OBJECT TRACKING

Edited by **Hanna Goszczynska**

INTECHWEB.ORG

Object Tracking
Edited by Hanna Goszczynska

Published by InTech
Janeza Trdine 9, 51000 Rijeka, Croatia

Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published articles. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

Publishing Process Manager Ivana Lorkovic
Technical Editor Teodora Smiljanic
Cover Designer Martina Sirotic
Image Copyright renkshot, 2010. Used under license from Shutterstock.com

First published February, 2011
Printed in India

A free online edition of this book is available at www.intechopen.com
Additional hard copies can be obtained from orders@intechweb.org

Object Tracking, Edited by Hanna Goszczynska
p. cm.
ISBN 978-953-307-360-6

Contents

Preface

Object tracking consists in estimation of trajectory of moving objects in the sequence of images. Automation of the computer object tracking is a difficult task. Dynamics of multiple parameters changes representing features and motion of the objects, and temporary partial or full occlusion of the tracked objects have to be considered.

This monograph presents the development of object tracking algorithms, methods and systems. Both, state of the art of object tracking methods and also the new trends in research are described in this book. Fourteen chapters are split into two sections. Part 1 presents new theoretical ideas whereas Part 2 presents real-life applications. This division was not easy because some chapters contain both, methods and applications. Editor has done at her best to keep the consistency of both Sections.

Part 1 presents the methods based on block matching and object identification using color and spatial information (Chapter 1). Chapter 2 presents methods that use the structural similarity measure in deterministic and probabilistic approaches based on hybrid particle filter. There adaptive tracking method is proposed in Chapter 3, which integrates online semi-supervised classification and particle filter. Local features and global appearances and shape of dynamic objects methods are described in Chapter 4. Chapter 5 covers local and global tracking using the mean-shift and covariance matching. Fusion of confidence maps in the visual object tracking procedure is described in Chapter 6. Solving the occlusion problems by coupling between optical flow and predictive algorithms using fuzzy control system is shown in Chapter 7.

Chapter 8 and Chapter 9 in Part 1 present the application of commercial software and hardware solutions for object tracking. Chapter 8 presents a method for extraction and indexing of moving objects in H.264/AVC bitstream domains and Chapter 9 presents real-time implementation of graph cuts calculation on GPUs using CUDA for simultaneous object tracking and segmentation.

Part 2 presents descriptions of the application of automatic object tracking to some special purposes. The frameworks targeted for digital video surveillance tasks in urban environment and over wide-areas such as an airport, downtown of a large city or any large public area are presented in Chapters 10 and Chapters 11. Automatic calibration method for distributed sensors by using mobile robots is presented in Chapter 12. The method for tracking surfaces which deformations are difficult to be described, such as drapery of textiles, facial expressions or complex organ motion in medical image sequences is presented in Chapter 13. Chapter 14 describes method for coronary artery segment tracking in angiographic images.

Despite the variety of topics contained in this monograph it constitutes a consisted knowledge in the field of computer object tracking. The intention of editor was to follow up the very quick progress in the developing of methods as well as extension of the application. Editor thanks the authors for their excellent contributions.

Hanna Goszczynska
Nałęcz Institute of Biocybernetics and Biomedical Engineering
Polish Academy of Sciences,
Warsaw

Part 1

Methods

A Block Matching Technique for Object Tracking Based on Peripheral Increment Sign Correlation Image

Budi Sugandi, Hyoungseop Kim., Joo Kooi Tan and Seiji Ishikawa
Graduate School of Engineering, Kyushu Institute of Technology
Japan

1. Introduction

Automatic detection and tracking of moving object is very important task for human-computer interface (Black & Jepson, 1998), video communication/expression (Menser & Brunig, 2000), and security and surveillance system application (Greiffenhagen et al., 2000) and so on. Various imaging techniques for detection, tracking and identification of the moving objects have been proposed by many researchers. Based on (Collins et al., 2000; Yilmaz, 2006), the object detection can be divided at least into five conventional approaches: frame difference (Lipton et al., 1998; Collins et al., 2000), background subtraction (Heikkila & Silven, 1999; Stauffer & Grimson, 1999; McIvor, 2000; Liu et al., 2001), optical flow (Meyer et al., 1998), skin color extraction (Cho et al., 2001; Phung et al., 2003) and probability based approaches (Harwood et al., 2000; Stauffer & Grimson, 2000; Paragios et al., 2000). Based on (Wu et al., 2004), the object tracking method can be categorized into four categories: region based tracking (Wren et al., 1997; McKenna, 2000), active contour based tracking (Paragois & Deriche, 2000), feature based tracking (Schiele, 2000; Coifman et al., 1998) and model based tracking (Koller, 2000). The object identification is performed to evaluate the effectiveness of the tracking object especially when the object occlusion happens. It can be done by measuring the similarity between the object model and the tracked object. Some of the researches rely on color distribution (Cheng & Chen, 2006; Czyz et al., 2007).

Regarding to our study, many of researchers have their own methods to solve the problem of object detection, object tracking and object identification.

In object detection methodology, many researchers have developed their methods. (Liu et al., 2001) proposed background subtraction to detect moving regions in an image by taking the difference between current and reference background image in a pixel-by-pixel. It is extremely sensitive to change in dynamic scenes derived from lighting and extraneous events etc. In another work, (Stauffer & Grimson, 1997) proposed a Gaussian mixture model based on background model to detect the object. (Lipton et al., 1998) proposed frame difference that use of the pixel-wise differences between two frame images to extract the moving regions. This method is very adaptive to dynamic environments, but generally does a poor job of extracting all the relevant pixels, e.g., there may be holes left inside moving entities. In order to overcome disadvantage of two-frames differencing, in some cases three-frames differencing is used. For instance, (Collins et al., 2000) developed a hybrid method that combines three-frame differencing with an adaptive background subtraction model for their VSAM (Video Surveillance and Monitoring) project. The hybrid algorithm successfully segments moving regions in video without the defects of temporal differencing and background subtraction. (Desa & Salih, 2004) proposed a combination of background subtraction and frame difference that improved the previous results of background subtraction and frame difference.

In object tracking methodology, regarding to our study, this article will describe more about the region based tracking. Region-based tracking algorithms track objects according to variations of the image regions corresponding to the moving objects. For these algorithms, the background image is maintained dynamically and motion regions are usually detected by subtracting the background from the current image. (Wren et al., 1997) explored the use of small blob features to track a single human in an indoor environment. In their work, a human body is considered as a combination of some blobs respectively representing various body parts such as head, torso and the four limbs. The pixels belonging to the human body are assigned to the different body part's blobs. By tracking each small blob, the moving human is successfully tracked. (McKenna et al., 2000) proposed an adaptive background subtraction method in which color and gradient information are combined to cope with shadows and unreliable color cues in motion segmentation. Tracking is then performed at three levels of abstraction: regions, people, and groups. Each region has a bounding box and regions can merge and split. A human is composed of one or more regions grouped together under the condition of geometric structure constraints on the human body, and a human group consists of one or more people grouped together.

Moreover, for object identification, (Cheng & Chen, 2006) proposed a color and a spatial feature of the object to identify the track object. The spatial feature is extracted from the bounding box of the object. Meanwhile, the color features extracted is mean and standard value of each object. (Czyz et al., 2007) proposed the color distribution of the object as observation model. The similarity of the object is measure using Bhattacharya distance. The low Bhattacharya distance corresponds to the high similarity.

To overcome the related problem described above, this article proposed a new technique for object detection employing frame difference on low resolution image (Sugandi et al., 2007), object tracking employing block matching algorithm based on PISC image (Satoh et al., 2001) and object identification employing color and spatial information of the tracked object (Cheng & Chen, 2006).

The study in this article includes three main building blocks for building an automated tracking system, which can be listed as object detection, object tracking and object identification as shown in Fig. 1. To cover all section, we organized our article as follows: in section 2, we give a detail of some image pre-processing techniques and filtering process to detect the moving object emerging in the scene. The tracking method is described in section 3 including the block matching algorithm. Object identification based on color and spatial feature is presented in section 4. Experimental results and conclusions are described in section 5 and 6, respectively.

Nearly, every tracking system starts with motion detection. Motion detection aims at separating the corresponding moving objects region from the background image. The first process in the motion detection is capturing the image information using a video camera. The motion detection stage includes some image preprocessing step such as; gray-scaling and smoothing, reducing image resolution using low resolution image technique, frame difference, morphological operation and labeling. The preprocessing steps are applied to reduce the image noise in order to achieve a higher accuracy of the tracking. The smoothing technique is performed by using median filter. The low resolution image is performed in three successive frames to remove the small or fake motion in the background. Then frame difference is performed on those frames to detect the moving object emerging in the scene. The next process is applying morphological operation such as dilation and erosion as filtering to reduce the noise that is remained in the moving object. Connected component labeling is then performed to label each moving object in different label.

The second stage is tracking the moving object. In this stage, we perform a block matching technique to track only the interest moving object among the moving objects emerging in the background. The blocks are defined by dividing the image frame into non-overlapping square parts. The blocks are made based on PISC image that considers the brightness change in all the pixels of the blocks relative to the considered pixel.

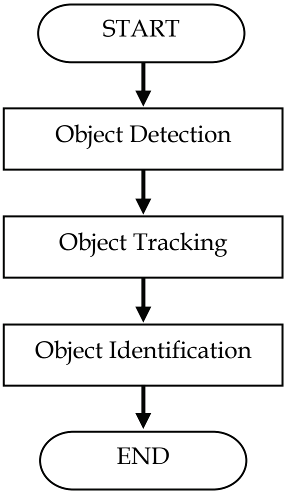


Fig. 1. Entire flow of procedure

The last stage is object identification. For this purpose we use spatial and color information of the tracked object as the image feature (Cheng & Chen, 2006). Then, a feature queue is created to save the features of the moving objects. When the new objects appear on the scene, they will be tracked and labeled, and the features of the object are extracted and recorded into the queue. Once a moving object is detected, the system will extract the features of the object and identify it from the identified objects in the queue.

The details of each stage are described as following.

2. Object detection

Performance of an automated visual surveillance system considerably depends on its ability to detect moving objects in the observed environment. A subsequent action, such as tracking, analyzing the motion or identifying objects, requires an accurate extraction of the foreground objects, making moving object detection a crucial part of the system. In order to decide on whether some regions in a frame are foreground or not, there should be a model for the background intensities. This model should also be able to capture and store necessary background information. Any change, which is caused by a new object, should be detected by this model, whereas un-stationary background regions, such as branches and leafs of a tree or a flag waving in the wind, should be identified as a part of the background. In this thesis we propose method handle those problem related to un-stationary background such as branches and leafs of a tree by reducing the resolution of the image.

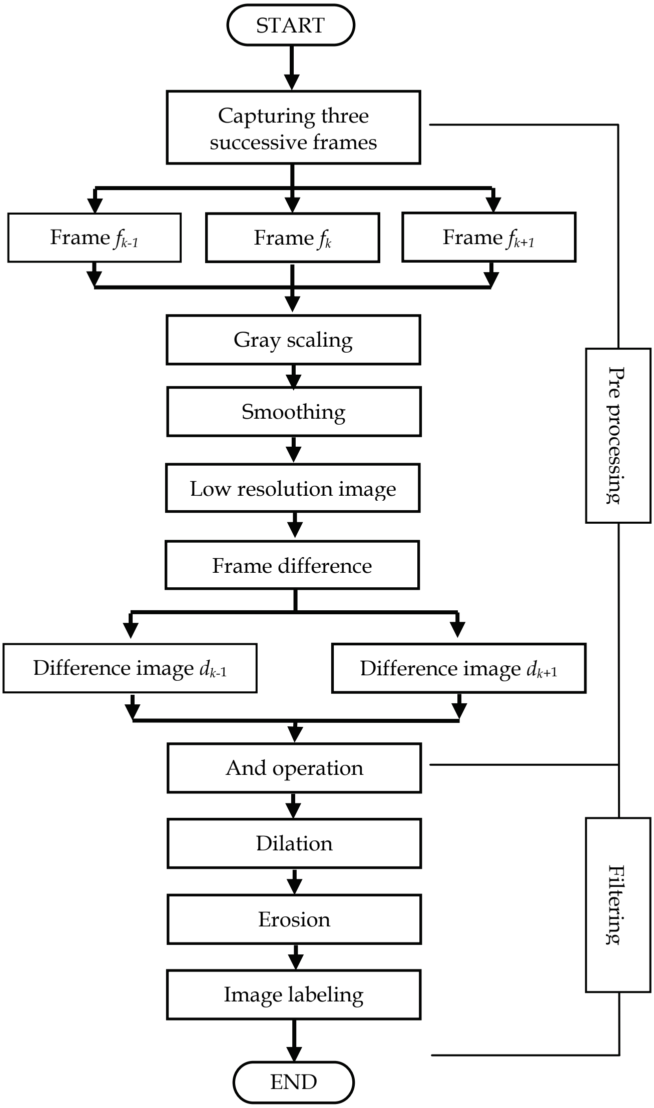


Fig. 2. Flow of object detection

Our object detection method consists of two main steps. The first step is pre-processing step including gray scaling, smoothing, and reducing image resolution and so on. The second step is filtering to remove the image noise contained in the object. The filtering is performed by applying the morphology filter such as dilation and erosion. And finally connected component labeling is performed on the filtered image.

The entire process of moving object detection is illustrated in Fig. 2.

2.1 Pre-processing

The first step on the moving object detection process is capturing the image information using a video camera. Image is capture by a video camera as 24 bit RGB (red, green, blue) image which each color is specified using 8-bit unsigned integers (0 through 255) that representing the intensities of each color. The size of the captured image is 320x240 pixels. This RGB image is used as input image for the next stage.

In order to reduce the processing time, gray-scale image is used on entire process instead of color image. The gray-scale image only has one color channel that consists of 8 bit while RGB image has three color channels. The color conversion between gray-scale image and RGB image is defined by the following equation:

$$Y = 0.3 \times R + 0.59 \times G + 0.11 \times B \tag{1}$$

where: Y is gray-scale image, R is red, G is green and B is Blue of RGB image, respectively.

Image smoothing is performed to reduce image noise from input image in order to achieve high accuracy for detecting the moving objects. The smoothing process is performed by using a median filter with $m \times m$ pixels.

We consider un-stationary background such as branches and leaf of a tree as part of the background. The un-stationary background often considers as a fake motion other than the motion of the object interest and can cause the failure of detection of the object. To handle this problem, we reduce the resolution of the image to be a low resolution image. A low resolution image is done by reducing spatial resolution of the image with keeping the image size (Gonzales & Woods, 2001) and (Sugandi et al., 2007). In this article, the low resolution image is done by averaging pixels value of its neighbors, including itself. We use a video image with resolution 320x240 pixels. The original image size is 320x240 pixels. After applying the low resolution image, the numbers of pixels will be 160x120, 80x60, or 40x30 pixels, respectively, while the image size is still 320x240 pixels. The low resolution image can be used for reducing the scattering noise and the small fake motion in the background because of un-stationary background such as leaf of a tree. These noises that have small motion region will be disappeared in low resolution image.

To detect the moving object from the background based of image subtraction, generally there are three approaches can be performed: (i) background subtraction as discussed in (Liu et al., 2001), (ii) frame difference as discussed in (Lipton et al., 1998), and (iii) combination of background subtraction and frame difference as discussed in (Desa & Salih, 2004). Background subtraction is computing the difference between the current and the reference background image in a pixel-by-pixel. Frame difference is computing the difference image between the successive frames image. In this article, we applied frame difference method to detect the moving objects. In our case, frame difference method is performed on the three successive frames, which are between frame f_k and f_{k-1} and between frame f_k and f_{k+1}. The output image as frame difference image is two difference images d_{k-1} and d_{k+1} as expressed in Eq. (2). Threshold is performed by threshold value T on the difference image d_{k-1} and d_{k+1} as defined in Eq. (3) to distinguish between the moving object and background.

$$d_{k-1} = \left| f_k - f_{k-1} \right|$$

$$d_{k+1} = \left| f_k - f_{k+1} \right| \tag{2}$$

$$d'_k(x,y) = \begin{cases} 1, & \text{if } d'_k(x,y) > T \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

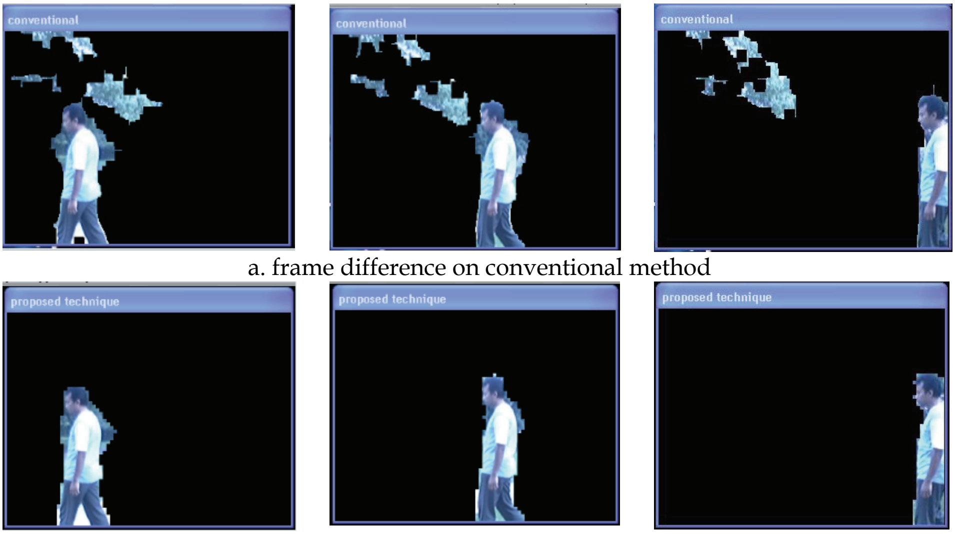
where $k' = k - 1$ and $k + 1$.
The process is followed by applying AND operator to $d_{k\text{-}1}$ and d_{k+1} as expressed in Eq. (4). The output image of this operation is named as motion mask m_p.

$$m_p = d_{k-1} \cap d_{k+1} \tag{4}$$

The comparison of moving object detection using a conventional method (frame different on normal resolution) and frame different method on low resolution image is shown in Fig 3. On those figures, we use same threshold to determine the object. As shown in those figures, using the conventional method the detected moving object is still greatly affected by small noise such as moving leaves. In the other hand, by reducing the resolution of the image before taking the difference frame, that kind of noise can be removed.

2.2 Filtering

In order to fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour, a morphological operation is applied to the image.



a. frame difference on conventional method

b. frame difference on a low resolution image

Fig. 3. Comparison of moving object detection technique

As a result, small gaps between the isolated segments are erased and the regions are merged. To extract the bounding boxes of detected objects, connected component analysis was used. We find all contours in image and draw the rectangles around corresponding contours with minimum area. Since the image may contain regions which are composed of background noise pixels and these regions are smaller than actual motion regions, we discard the region with a smaller area than the predefined threshold. As a result, the processing produces perfect bounding boxes.

Morphological operation is performed to fill small gaps inside the moving object and to reduce the noise remained in the moving objects (Stringa, 2000). The morphological operators implemented are dilation followed by erosion. In dilation, each background pixel that is touching an object pixel is changed into an object pixel. Dilation adds pixels to the boundary of the object and closes isolated background pixel. Dilation can be expressed as:

$$f(x,y)=\begin{cases}1, & \text{if there is one or more pixels of the 8 neighbors are 1}\\ 0, & \text{otherwise}\end{cases} \tag{5}$$

In erosion, each object pixel that is touching a background pixel is changed into a background pixel. Erosion removes isolated foreground pixels. Erosion can be expressed as:

$$f(x,y)=\begin{cases}0, & \text{if there is one or more pixels of the 8 neighbors are 0}\\ 1, & \text{otherwise}\end{cases} \tag{6}$$

Morphological operation eliminates background noise and fills small gaps inside an object. This property makes it well suited to our objective since we are interested in generating masks which preserve the object boundary.

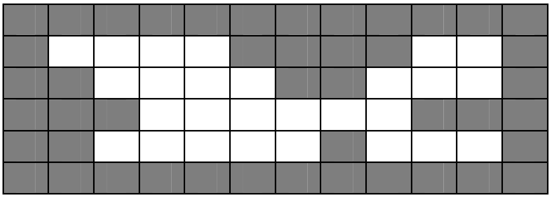


Fig. 4. Binary image

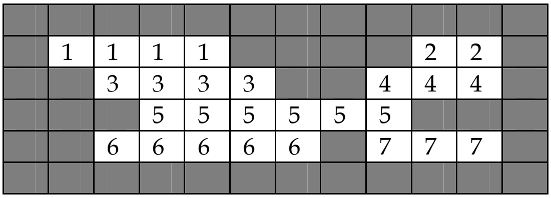
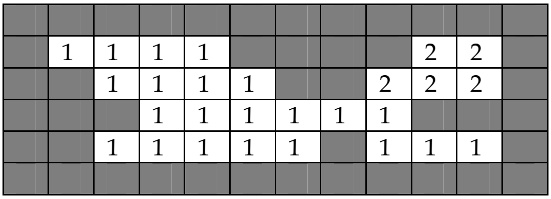
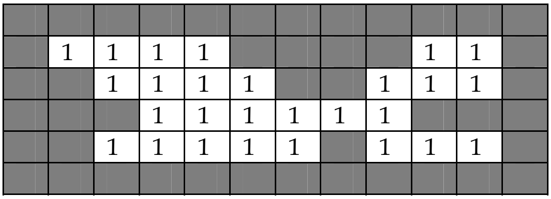


Fig. 5. Image is labeled in the same row



	1	1	1	1					2	2	
		1	1	1	1			2	2	2	
			1	1	1	1	1	1			
		1	1	1	1	1		1	1	1	

Fig. 6. Labeled image is scanned from top-left to bottom-right



	1	1	1	1					1	1	
		1	1	1	1			1	1	1	
			1	1	1	1	1	1			
		1	1	1	1	1		1	1	1	

Fig. 7. Final labeled image

Connected component labeling is performed to label each moving object emerging in the background. The connected component labeling (Gonzales & Woods, 2001) groups the pixels into components based on pixel connectivity (same intensity or gray level). In this article, connected component labeling is done by comparing the pixel with the pixel in four neighbors. If the pixel has at least one neighbor with the same label, this pixel is labeled as same as neighbor's label. The algorithm of connected component labeling algorithm is described as follows:

1. Firstly, image labeling is done on binary image as shown in Fig. 4 where object is shown as 1 (white) and background is shown as 0 (black).
2. The image is scanned from top-left to search the object pixel. The label is done by scanning the image from left to right and comparing label with the neighbor's label in the same line. If the neighbor has the same pixel value, the pixel is labeled as same as previous label as shown in Figure. 5.
3. Next, the labeled image is scanned from top-left to bottom-right by comparing with the four (or eight) neighbors pixel which have already been encountered in the scan (the neighbors (i) to the left of the pixel, (ii) above it, and (iii and iv) the two upper diagonal terms). If the pixel has at least one neighbor, then this pixel is labeled as same as neighbor's label as shown in Fig. 6.
4. On the last scanning, the image is scanned from bottom-right to top-left by comparing with the four neighbors pixel as step 3. The final labeled image is shown in Fig. 7.

3. Object tracking

After the object detection is achieved, the problem of establishing a correspondence between object masks in consecutive frames should arise. Indeed, initializing a track, updating it robustly and ending the track are important problems of object mask association during tracking. Obtaining the correct track information is crucial for subsequent actions, such as object identification and activity recognition. Tracking process can be considered as a region mask association between temporally consecutive frames and estimating the trajectory of an object in the image plane as it moves around a scene. In this article, we use block matching technique for this purpose. The entire process of tracking the moving object is illustrated in Fig 8. The details of the tracking mechanism of the interest moving object are described in the following sections.

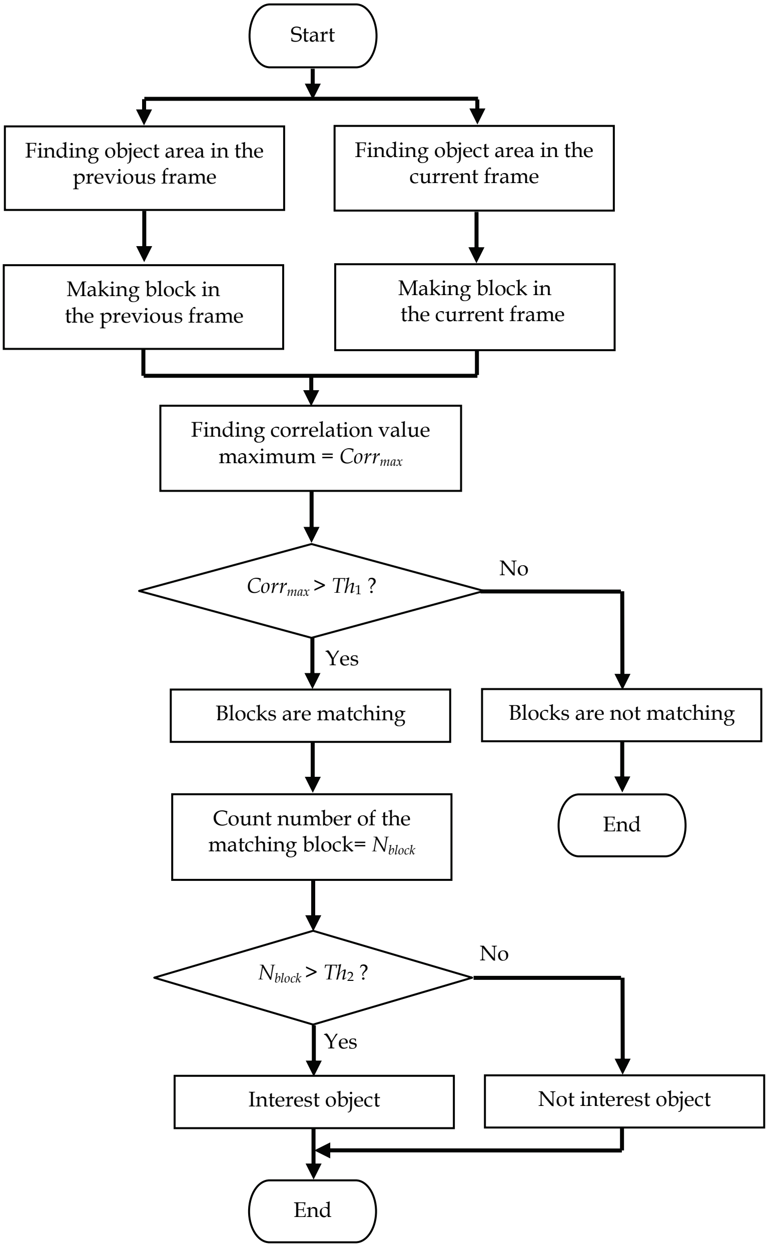

Start
Finding object area in the previous frame
Finding object area in the current frame
Making block in the previous frame
Making block in the current frame
Finding correlation value maximum = $Corr_{max}$
$Corr_{max} > Th_1$?
No
Yes
Blocks are matching
Blocks are not matching
Count number of the matching block= N_{block}
End
$N_{block} > Th_2$?
No
Yes
Interest object
Not interest object
End

Fig. 8. Entire flow of tracking of interest object

3.1 Block matching method

Block matching is a technique for tracking the interest moving object among the moving objects emerging in the scene. In this article, the blocks are defined by dividing the image frame into non-overlapping square parts. The blocks are made based on peripheral increment sign correlation (PISC) image (Satoh et al., 2001; Sugandi et al., 2007) that considers the brightness change in all the pixels of the blocks relative to the considered pixel. Fig. 9 shows the block in PISC image with block size is 5×5 pixels. Therefore, one block consists of 25 pixels. The blocks of the PISC image in the previous frame are defined as shown in Eq. (7). Similarly, the blocks of the PISC image in the current frame are defined in Eq. (8). To determine the matching criteria of the blocks in two successive frames, we evaluate using correlation value that expresses in Eq. (9). This equation calculates the correlation value between block in the previous frame and the current one for all pixels in the block. The high correlation value shows that the blocks are matched each other. The interest moving object is determined when the number of matching blocks in the previous and current frame are higher than the certain threshold value. The threshold value is obtained experimentally.

$$b_{np} = \begin{cases} 1, & \text{if } f_{np} \geq f(i,j) \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$b'_{np} = \begin{cases} 1, & \text{if } f_{np} \geq f(i,j) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$corr_n = \sum_{p=0}^{N} b_{np} * b'_{np} + \sum_{p=0}^{N} (1-b_{np}) * (1-b'_{np}) \tag{9}$$

where : b and b' are the block in previous and current frame, n is number of block and N is number of pixels of block, respectively.

$f_{i-2,j-2}$	$f_{i-1,j-2}$	$f_{i,j-2}$	$f_{i+1,j-2}$	$f_{i+2,j-2}$
$f_{i-2,j-1}$	$f_{i-1,j-1}$	$f_{i,j-1}$	$f_{i+1,j-1}$	$f_{i+2,j-1}$
$f_{i-2,j}$	$f_{i-1,j}$	$f_{i,j}$	$f_{i+1,j}$	$f_{i+2,j}$
$f_{i-2,j+1}$	$f_{i-1,j+1}$	$f_{i,j+1}$	$f_{i+1,j+1}$	$f_{i+2,j+1}$
$f_{i-2,j+2}$	$f_{i-1,j+2}$	$f_{i,j+2}$	$f_{i+1,j+2}$	$f_{i+2,j+2}$

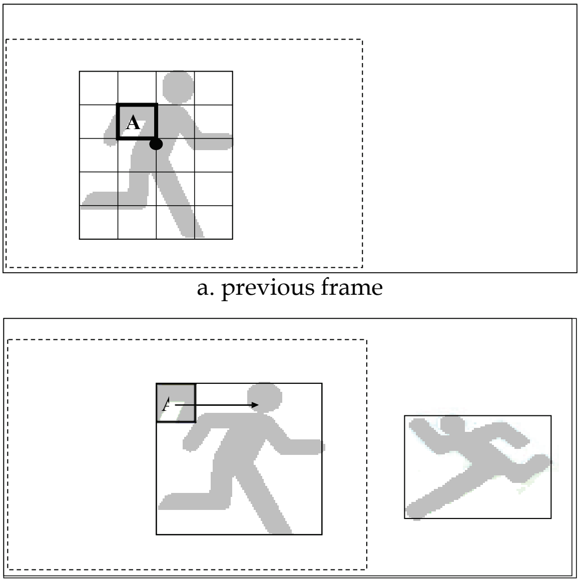
Fig. 9. PISC image for block size 5 × 5 pixels

3.2 Tracking method

The tracking method used in this article can be described as following. The matching process is illustrated in Fig. 10. Firstly, blocks and the tracking area are made only in the area of moving object to reduce the processing time. We make the block size (block *A*) with 9x9 pixels in the previous frame. We assume that the object coming firstly will be tracked as the interest moving object. The block *A* will search the matching block in each block of the current frame by using correlation value as expresses in Eq. (9). In the current frame, the interest moving object is tracked when the object has maximum number of matching blocks. When that matching criteria is not satisfied, the matching process is repeated by enlarging the tracking area (the rectangle with dash line). The blocks still are made inside the area of moving object. When the interest moving object still cannot be tracked, then the moving object is categorized as not interest moving object or another object and the tracking process is begun again from the beginning.

4. Object identification

The object identification is the last stage of our study. In this stage, firstly, we will explain about features extraction for object detection (Cheng & Chen, 2006). In this article, the extracted features are divided into the following two types; color and spatial information of the moving objects.



a. previous frame

b. curent frame

Fig. 10. Matching process

4.1 Spatial feature extraction

The feature of objects extracted in the spatial domain is the position of the tracked object. The spatial information combined with the features in time domain represents the trajectory of the tracked object, so we can estimate the movement and speed of the moving objects that we tracked. Therefore, the features of spatial domain are very important to object identification. The bounding box defined in Eq. (10) is used as spatial information of moving objects.

After getting the interest moving object, we extract the interest moving object by using a bounding box. The bounding box can be determined by computing the maximum and minimum value of x and y coordinates of the interest moving object according to the following equation:

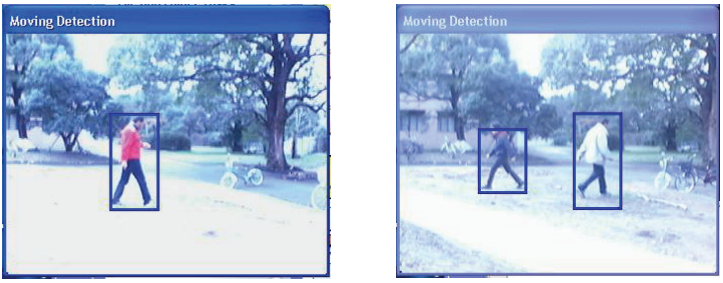
$$B^i_{\min} = \left\{ \left(x^i_{\min}, y^i_{\min} \right) \; | x, y \in O^i \right\}$$

$$B^i_{\max} = \left\{ \left(x^i_{\max}, y^i_{\max} \right) \; | x, y \in O^i \right\} \tag{10}$$

where O^i denotes the set of the coordinate of points in the interest moving object i, $B^i{}_{min}$ is the left-top corner coordinates of the interest moving object i and $B^i{}_{max}$ is the right-bottom corner coordinates of the interest moving object i, respectively. Fig. 11 shows an example of the bounding box of the object tracking.

4.2 Color feature extraction

The color feature extracted from the object is RGB color space as the RGB color information can be obtained from video capture device directly. We extract the information from upper



a. single moving object b. two moving objects

Fig. 11. Example of bounding box of moving object

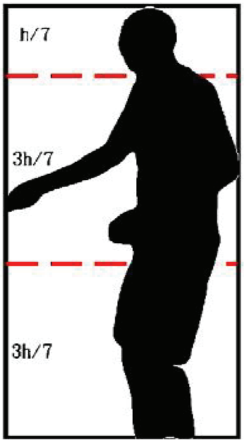


Fig. 12. Definition of a human body ratio

and lower part of the object to obtain more color information for identification. The ratio of these three parts can be defined as shown in Fig. 12. However, in this article, we only calculate the color information of the upper and lower part excluding the head part of the object. The first color information calculated is mean value of each human body part as calculated by Eq. (11) for upper part and Eq. (12) for lower part. The mean value is calculated for each color component of RGB space.

$$\mu_{f_k}^{O_U^i} = \frac{\sum_{x=x_{\min}^i}^{x_{\max}^i} \sum_{y=y_{\min}^i}^{y_{\max}^i} f_k(x,y)}{\#O_U^i} \tag{11}$$

$$\mu_{f_k}^{O_L^i} = \frac{\sum_{x=x_{\min}^i}^{x_{\max}^i} \sum_{y=y_{\min}^i}^{y_{\max}^i} f_k(x,y)}{\#O_L^i} \tag{12}$$

where *i* is number of the moving objects and *(x, y)* is the coordinate of pixels in moving object. $\left(x_{\max}^i, y_{\max}^i\right)$ and $\left(x_{\min}^i, y_{\min}^i\right)$ are the coordinates of the bounding box of moving object *i*, $f_k(x,y)$ denotes pixel value for each color component in RGB space of the current frame, O_U^i and O_L^i denote the set of coordinates of upper and lower part of human body of moving object *i* and #*Oi* is the number of pixels of moving object *i*.

The standard deviation has proven to be an extremely useful measure of spread in part because it is mathematically tractable. Standard deviation is a statistical term that provides a good indication of volatility. It measures how widely the values are dispersed from the average. Dispersion is the difference between the actual value and the average value. The larger the difference between the actual color and the average color is, the higher the standard deviation will be, and the higher the volatility. We can extract more useful color features by computing the dispersed color information from upper and lower part of body as shown in Eq. (13) for the upper part and Eq. (14) for the lower part.

$$SD_{f_k}^{O_U^i} = \sqrt{\frac{\sum_{x=x_{\min}^i}^{x_{\max}^i} \sum_{y=y_{\min}^i}^{y_{\max}^i} \left(f_k(x,y) - \mu_{f_k}^{O_U^i}\right)^2}{\#O_U^i}} \tag{13}$$

$$SD_{f_k}^{O_L^i} = \sqrt{\frac{\sum_{x=x_{\min}^i}^{x_{\max}^i} \sum_{y=y_{\min}^i}^{y_{\max}^i} \left(f_k(x,y) - \mu_{f_k}^{O_L^i}\right)^2}{\#O_L^i}} \tag{14}$$

where $SD_{f_k}^{O_U^i}$ and $SD_{f_k}^{O_L^i}$ denote the standard deviation of each color component of RGB space for upper and lower part of the human body of moving object *i*, respectively.

4.3 Identification process

After the feature extraction, we can represent the moving object *i* by the following feature vectors;

$$F^i = \left(\mu_{f_k}^{O_U^i}, \mu_{f_k}^{O_L^i}, SD_{f_k}^{O_U^i}, SD_{f_k}^{O_L^i}, B_{\min}^i, B_{\max}^i\right) \tag{15}$$

To identify a moving object, a feature queue is created to save the features of the moving objects. When a new object enters the system, it will be tracked and labeled, and the features of the object are extracted and recorded into the queue. Once a moving object *i* is detected, the system will extract the features F^i of the object *i* and identify it from the identified objects in the queue by computing the similarity $S\left(F^i,F^j\right)$, *j* =1… *n*, where *j* is one of the *n* identified objects. The similarity, $S\left(F^i,F^j\right)$ is computed as in Eq. (16),

$$\begin{aligned} S\left(F^i,F^j\right) = {} & Mc\left(\left|\mu_{f_k}^{O_U^i} - \mu_{f_k}^{O_U^j}\right|\right) + Mc\left(\left|\mu_{f_k}^{O_L^i} - \mu_{f_k}^{O_L^j}\right|\right) \\ & + Msd\left(\left|SD_{f_k}^{O_U^i} - SD_{f_k}^{O_U^j}\right|\right) + Msd\left(\left|SD_{f_k}^{O_L^i} - SD_{f_k}^{O_L^j}\right|\right) \\ & + 0.5Mp\left(\left|B_{\min}^i - B_{\min}^j\right|\right) + 0.5Mp\left(\left|B_{\max}^i - B_{\max}^j\right|\right) \end{aligned} \tag{16}$$

where *Mc* and *Msd* are the membership function of color information and standard deviation as defined in Eq. (17) and Eq. (18), *Mp* is the membership function of spatial information as defined in Eq. (19).

$$Mc(x) = \begin{cases} 1 - x / Thr & \text{if } x < Thr \\ 0 & \text{if } x \geq Thr \end{cases} \tag{17}$$

$$Msd(x) = \begin{cases} 1 - x / Thr & \text{if } x < Thr \\ 0 & \text{if } x \geq Thr \end{cases} \tag{18}$$

$$Mp(x) = \begin{cases} 1 - 3x / W & \text{if } x < W / 3 \\ 0 & \text{if } x \geq W / 3 \end{cases} \tag{19}$$

where *Thr* is threshold which is obtained experimentally and *W* is the width or height of the image frames. We compare the features of detected object with those of the objects in the feature queue. The one with the maximum similarity is identified as the same object. The whole object identification flow is shown in Fig. 13.

5. Experimental results

We have done the experiments using a video camera in outdoor environment and real time condition. The experiment is performed in 2.54 [GHz] Pentium 4 PC with 512 MB memory. The image resolution is 320 × 240 [pixels]. The size of each block is 9×9 [pixels]. The experimental results are shown in Fig. 14–Fig. 16. The rectangle area on the object shows the tracked object. The identification result is shown in Table 1. In the experimental results, we can extract the moving objects on the successive frame successfully and identification rates of 92.8% were achieved.

In our experiment, we tracked the interest object from two and three moving objects that occluded between each other when objects move in the same and different direction. We assumed that the first moving object emerging in the scene as the interest moving object. We did the experiments in three conditions. In the first condition, we tracked the interest

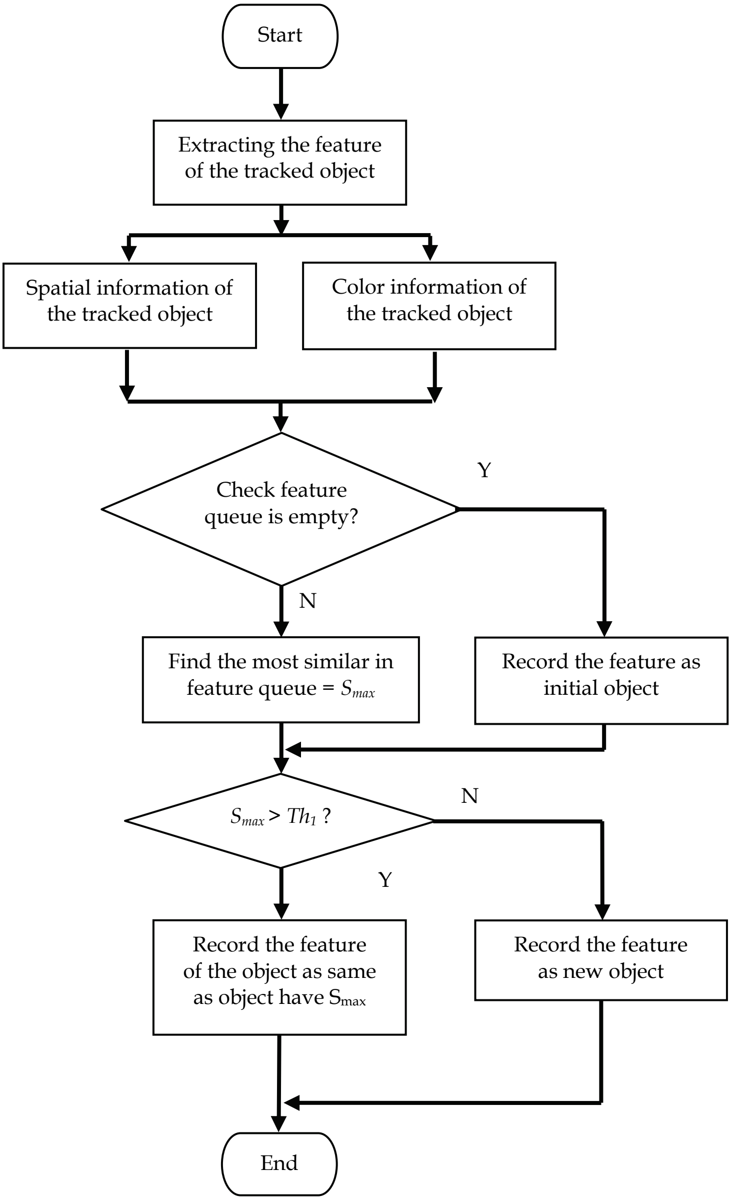

Start
Extracting the feature of the tracked object
Spatial information of the tracked object
Color information of the tracked object
Check feature queue is empty?
Y
N
Find the most similar in feature queue = S_{max}
Record the feature as initial object
$S_{max} > Th_1$?
N
Y
Record the feature of the object as same as object have S_{max}
Record the feature as new object
End

Fig. 13. Flow chart of object identification

moving object when it is occluded by another moving object when they move in the same direction as shown in Fig. 14. In the second condition, we tracked the interest moving object when it is occluded by another moving object when they move in different direction as shown in Fig. 15. In the third condition, we tracked the interest moving object among three moving object appear in the scene as shown in Fig. 16.

On the first case (Fig. 14), at first, the man wearing the white shirt enters the scene from the left side. This object is successfully detected as the interest moving object. While the first object is being tracked, another object (man wearing the blue shirt) enters the scene from the right side. They move in the different direction and overlap each other in the middle of the scene. We successfully track the first moving object as the interest moving object as our assumptions while the other moving object is not tracked.

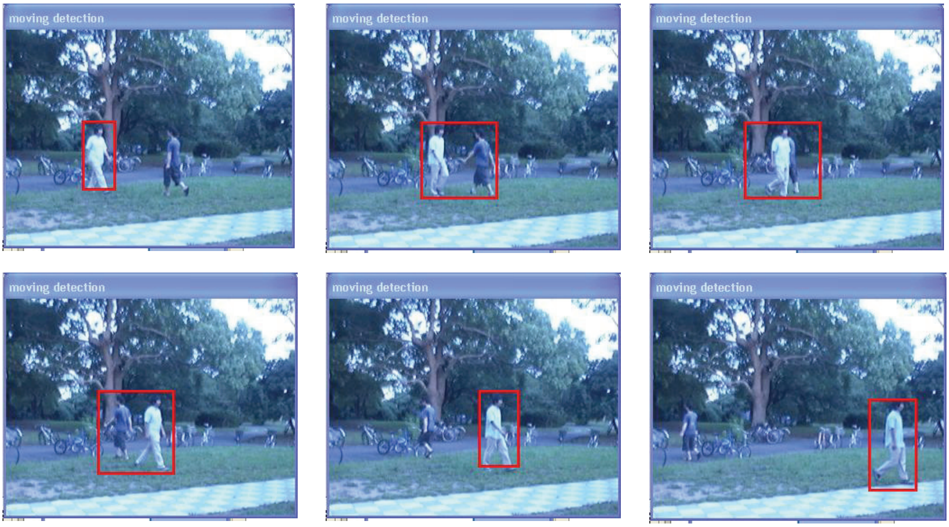


Fig. 14. Two moving objects move in different direction

On the second case (Fig. 15), at first, the man wearing the blue shirt enters the scene from the left side. This object is successfully detected as the interest moving object. Then on the next

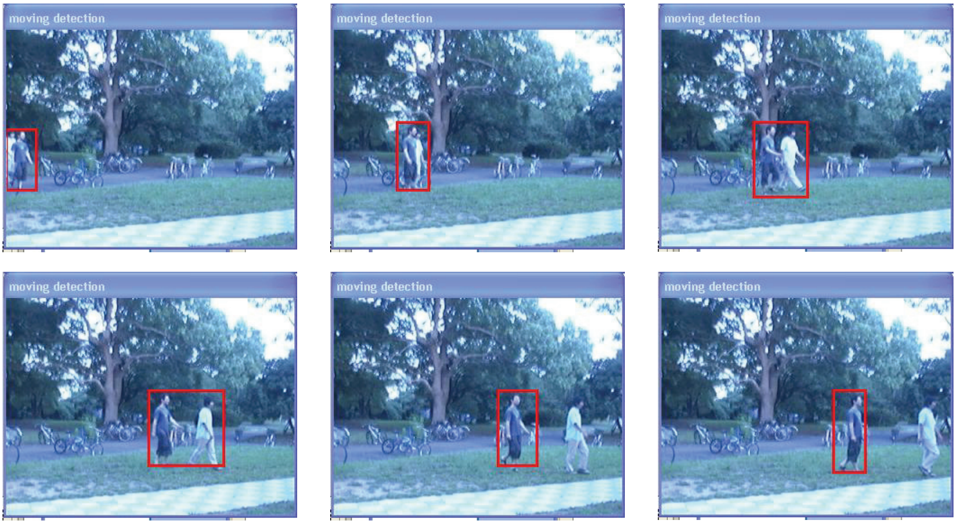


Fig. 15. Two moving objects move in same direction

frame, the man wearing the white shirt enters the scene from the same side. They move in the same direction and occlude each other in the middle of the scene. We successfully track the first moving object as the interest moving object as our assumptions while the other moving object is not tracked.

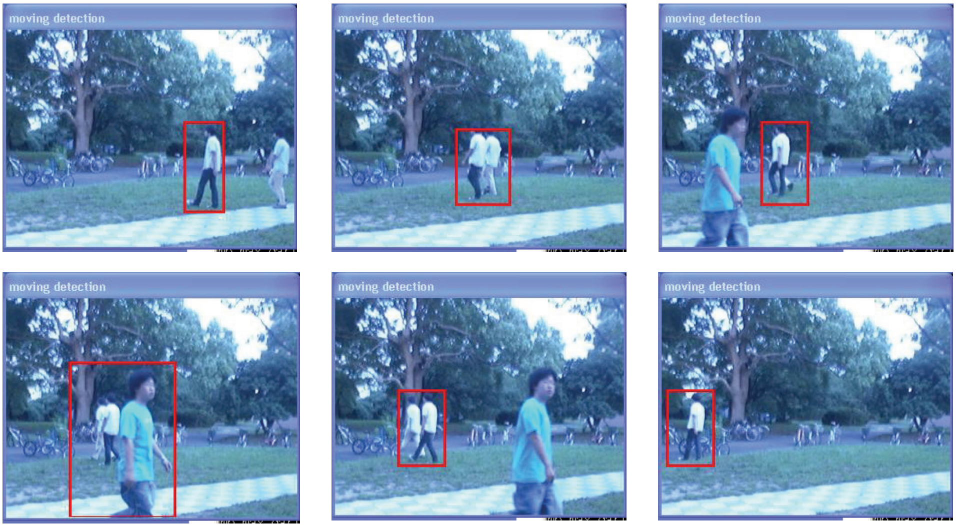
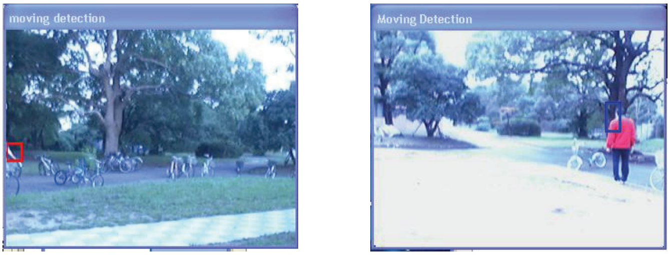


Fig. 16. Three moving objects occlude in the middle of the scene

Experiment	Object detected	Correct identification	Identification rates [%]
1	126	117	92.8
2	148	136	91.9
3	154	141	91.6

Table 1. Object identification results



a. an object entering scene

b. an object moves slow down

Fig. 17. Error in object identification

On the third case (Fig.16), at first, the man wearing the white shirt enters the scene from the right side. This person will be tracked as the interest moving object. Then on the next frame, another man enters the scene from right side. They move in the same direction. The third man enters the scene from the left side. They occlude each other in the middle on the scene. We successfully track the first moving object as interest moving object as our assumptions while the other moving object is not tracked.

To evaluate our method, we performed the object identification method based on spatial and color information of the tracked object. Table 1 shows the result of the object identification. From the table, we notice that some objects are not correctly identified in some frames of each experiment. The wrong identification occurs in two types. First, it occurs when the moving object just enters or leaves the scene. When the moving object is just entering the scene, the system can detect the moving object. However, the extracted features of the moving object in this case cannot represent the moving object very well, because only partial of the features of the moving objects are extracted. The second error occurs when the moving object is slowing down. In this situation, the frame difference of the object becomes smaller. Therefore, only smaller bounding box and less moving pixels are obtained. On this condition, the extracted features will lose its representative. Both of errors are illustrated in Fig. 17.

6. Conclusion

This article proposed a new method for detecting the moving object employing frame difference on low resolution image, tracking the moving object employing block matching technique based on peripheral increment sign correlation image for tracking the interest moving object among the moving objects emerging in the background and identifying the moving objects employing color and spatial information of the tracked object. The experiment results and data show the effectiveness and the satisfaction of the proposed methods. Using our method, we can achieve the identification rate of 92.1[%] in average.

However, the proposed method still has limitations. The limitations can be investigated as followings. Firstly, the detection method based on frame difference on low resolution image has a limitation when the moving object is too small to be detected. It is occurred because the low resolution image removes the small moving objects emerging in the background. To overcome this limitation, we can add another method such as skin color detection. By using this method, even if the moving object is too small, it can still be detected based on the skin color of the object. Secondly, the block matching technique has successfully tracked the interest moving object in the occlude condition. However, when the moving objects appear in the same time, we cannot judge any object to be an interest object. Moreover, when the interest moving object is covered by the occluded object, the image information of the interest moving object cannot be read by the camera. This condition cause the system cannot recognize the interest moving object. Those limitations can be solved by adding other information to the interest moving object such as flow of moving object based on optical flow, dimension or another feature and also we can add the color information to each object. So whenever the objects appear, they have their own model that different from each other. And we can track them based on the model.

Thirdly, color and spatial information method show the high correct identification rate. However, the system still cannot identify the objects sometimes when they are just entering or leaving the scene. The extracted features in this case are not enough to be used to identify the moving objects. The system also has limitation when the object is moving slowly. In this condition, the inter-frame difference image of the object will become smaller and we will get smaller bounding box and less moving pixels. Therefore, the extracted features will lose its representative. The correct identification rate highly depends on the correctness of the moving object detection and feature representation. This problem can be improved by a better feature selection method and moving object detection method.

By considering those limitations and implement some improvements to our method including speed up the processing time, they could lead to some improvements in the tracking system. These are remaining for future work.

7. References

Lipton, A; Fujiyoshi, H. & Patil, R.(1998) .Moving target classification and tracking from real-time video, *Proceeding of IEEE Workshop Applications of Computer Vision*, pp. 8-14.

Stauffer, C. & Grimson, W. (1999). Adaptive background mixture models for real-time tracking, *Proceeding of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 246-252.

LIU, Y.; Haizho, A. & Xu Guangyou.(2001). Moving object detection and tracking based on background subtraction, *Proceeding of Society of Photo-Optical Instrument Engineers (SPIE),* Vol. 4554, pp. 62-66.

Cheng, F. & Chen, Y. (2006). Real time multiple objects tracking and identification based on discrete wavelet transform, *Journal of the pattern Recognition Society*, pp 1126-1139.

Davis, J. W. & Bobick, A. F. (1997). The representation and recognition of action using temporal templates, *Proceeding of Comp Vision and Pattern Recognition*, pp. 928-934.

Meyer, D.; Denzler, J. & Niemann, H. (1998). Model based extraction of articulated objects in image sequences for gait analysis, *Proceeding of IEEE Int. Conf. Image Proccessing*, pp. 78-81.

Paragios, N. & Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving object, *IEEE Trans. on Pattern Analysis and Machine Intelligent*, pp. 266-280.

Satoh, Y.; Kaneko, S. & Igarashi, S. (2001). Robust Object Detection and Segmentation by Peripheral Increment Sign Correlation Image, *System and Computer in Japan*, pp.2585-2594.

Gonzalez, R. C. & Woods, R. E. (2001). *Digital Image Processing*, Addison-Wesley Longman Publishing Co., Inc.,Boston, MA.

Sugandi, B.; Kim, H.S.; Tan, J.K. & Ishikawa, S. (2007). Tracking of moving object by using low resolution image, *Proceeding of Int. Conf. on Innovative Computing, Information and Control (ICICIC07)*, Kumamoto, Japan, (4 pages).

Desa, S. M. & Salih, Q. A. (2004). Image subtraction for real time moving object extraction, *Proceeding of Int. Conf. on Computer Graphics, Imaging and Visualization (CGIV'04)*, pp. 41–45.

Stringa, E. (200). Morphological change detection algorithms for surveillance applications, *Proceeding of British Machine Vision Conf.*, pp. 402–412.

Sugandi, B.; Kim, H.S.; Tan, J.K. & Ishikawa, S. (2007).Tracking of moving persons using multi camera employing peripheral increment sign correlation image, *ICIC Express Letters, An International Journal of Research and Surveys*, Vol. 1, No. 2, pp. 177-184.

Black, M. & Jepson, A. (1998). A probabilistic framework for matching temporal trajectories: condensation-based recognition of gestures and expressions, *Proceeding of the European Conference on Computer Vision*, pp. 909-924.

Menser, B. & Brunig, V. (2000). Face detection and tracking for video coding applications, *Asilomar Conference on Signals, Systems, and Computers*, pp. 49-53.

Greiffenhagen, M.; Ramesh, V.; Comaniciu, D. & Niemann, H. (2000). Statistical modeling and performance characterization of a real-time dual camera surveillance system, *Proceeding of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 335-342.

McIvor, A. M. (2000). Background subtraction techniques, *Proceeding of Image and Vision Computing*, 6 pages.

Heikkila, J. & Silven, O. (1999). A real-time system for monitoring of cyclists and pedestrians, *Proceeding of Second IEEE Workshop on Visual Surveillance*, pp 74–81.

Collins, R. ; Lipton, A.; Kanade, T.; Fujiyoshi, H.; Duggins, D.; Tsin, Y.; Tolliver, D.; Enomoto, N. & Hasegawa. (2000). System for video surveillance and monitoring, *Technical Report CMU-RI-TR-00-12*, Robotics Institute, Carnegie Mellon University.

Meyer, D.; Denzler, J. & Niemann, H. (1998). Model based extraction of articulated objects in image sequences for gait analysis, *Proceeding of the IEEE Int. Conf. on Image Processing*, pp. 78–81.

Phung, S.; Chai, D. & Bouzerdoum, A. (2003). Adaptive skin segmentation in color images, *Proceeding of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 3, pp. 353-356.

Cho, K. M.; Jang, J. H. & Hong, K. S. (2001). Adaptive skin-color filter, *Pattern Recognition*, pp. 1067-1073.

Harwood, D. ; Haritaoglu, I. & Davis, L. S. (2000). W4: Real-time surveillance of people and their activities, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 809–830.

Stauffer, C. & Grimson, W. (2000). Learning patterns of activity using real-time tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 747–757.

Hu, W. ; Tan, T.; Wang, L. & Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors, *IEEE Trans. On System, Man, and Cybernetics*, Vol. 34, No. 3, pp. 334-352.

Wren, C. R.; Azarbayejani, A.; Darrell, T. & Pentland, A. P. (1997). Pfinder: real-time tracking of the human body, *IEEE Trans. Pattern Analysis. Machine Inteligent.*, Vol. 19, pp. 780–785.

McKenna, S.; Jabri, S.; Duric, Z.; Rosenfeld, A. & Wechsler, H. (2000). Tracking groups of people, *Computer Vision: Image Understanding*, Vol. 80, No. 1, pp. 42–56.

Paragios, N. & Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving objects, *IEEE Trans. Pattern Analysis and Machine Intelligent*, Vol. 22, pp. 266–280.

Schiele, B. (2000). Vodel-free tracking of cars and people based on color regions, *Proceeding of IEEE Int. Workshop Performance Evaluation of Tracking and Surveillance*, Grenoble, France, pp. 61–71.

Coifman, B.; Beymer, D.; McLauchlan, P. & Malik, J. (1998). A real-time computer vision system for vehicle tracking and traffic surveillance, *Transportation Research: Part C*, Vol. 6, No. 4, pp. 271–288.

Koller, D.; Danilidis, K. & Nagel. H. (2000). Model-based object tracking in monocular image sequences of road traffic scenes, *Int. Journal of Computer Vision*, Vol.10, No.3, pp.257-281.

Czyz, J.; Ristic, B. & Macq, B. (2007). A particle filter for joint detection and tracking of color objects”, *Image and Vision Computing*, Vol. 25, No. 8, pp 1271-1281.

Yilmaz, A.; Javed, O. & Shah, M. (2006). Object tracking: a survey, *ACM Computing Survey*, Vol.38, No.13, pp.1-45.

Structural Information Approaches to Object Tracking in Video Sequences

Artur Loza[1], Lyudmila Mihaylova[2], Fanglin Wang[3] and Jie Yang[4]
[2] *Lancaster University*
[1,3,4]*Shanghai Jiao Tong University*
[2]*United Kingdom*
[1,3,4]*China*

1. Introduction

The problem of object tracking has been a subject of numerous studies and has gained considerable interest (Aghajan & Cavallaro (2009); Gandhi & Trivedi (2007)) in the light of surveillance (Hu et al. (2004); Loza et al. (2009)), pedestrian protection systems (Forsyth et al. (2006); Gerónimo et al. (2010); Smith & Singh (2006)), vehicular traffic management, human vision systems (Chen & Yang (2007)) and others. The methods for object tracking can be subdivided into two main groups: deterministic (Bradski (1998); Cheng (1995); Comaniciu & Meer (2002); Comaniciu et al. (2000; 2003)) and probabilistic (e.g., Pérez et al. (2004)) within which the Bayesian techniques are the most prominent.

Most video tracking techniques are region based which means that the object of interest is contained within a region, often of a rectangular or circular shape. This region is then tracked in a sequence of video frames based on certain features (or their histograms), such as colour, texture, edges, shape, and their combinations (Brasnett et al. (2005; 2007); Pérez et al. (2004); Triesch & von der Malsburg (2001)).

This book chapter addresses the problem of object tracking in video sequences by using the recently proposed structural similarity-based image distance measure (Wang et al., 2005a; Wang & Simoncelli, 2004). Advantages of the Structural SIMilarity (SSIM) measure are its robustness to illumination changes and ability to extract the structural information from images and video frames. Real world videos are often recorded in unfavourable environments, for example with low or variable light exposure due to the weather conditions. These factors often cause undesired luminance and contrast variations in videos produced by optical cameras (e.g. the object entering dark or shadowy areas) and by Infrared (IR) sensors (due to varying thermal conditions or insufficient exposure of the object). Moreover, due to the presence of spurious objects or backgrounds in the environment, real-world video data may lack sufficient colour information needed to discriminate the tracked object against its background.

The commonly applied tracking techniques relying on colour and edge image features represented by histograms are often prone to failure in such conditions. In contrast, the SSIM reflects the distance between two video frames by jointly comparing their luminance, contrast and spatial characteristics and is sensitive to relative rather than absolute changes in

the video frame. It replaces histograms used for calculation of the measurement likelihood function within a particle filter. We demonstrate that it is a good and efficient alternative to histogram-based tracking. This work builds upon the results reported in Loza et al. (2006; 2009) with more detailed investigation including further extensions of the proposed method.
The remaining part of the book chapter is organised in the following way. Section 2 presents an overview of the main (deterministic and probabilistic) tracking approaches and outlines the Bayesian tracking framework. Section 3 presents the proposed approach, followed in Section 4 by the results obtained with real video data and Section 5 summarises the results and open issues for future research.

2. Video tracking overview

2.1 Deterministic methods

In this chapter an overview of selected deterministic & probabilistic tracking techniques is presented. Within the group of deterministic methods the mean shift (MS) algorithm (Cheng (1995); Comaniciu & Meer (2002); Comaniciu et al. (2000; 2003)) is one of the most widely used. The MS algorithm originally proposed by Fukunaga & Hostetler (1975) was further extended to computer vision problems in (Comaniciu & Meer (2002); Comaniciu et al. (2000)). It is a gradient based, iterative technique that uses smooth kernels, such as Gaussian or Epanechnikov for representing a probability density function. The similarity between the target region and the target candidates in the next video frame is evaluated using a metric based on the Bhattacharyya coefficient (Aherne et al. (1990)). The MS tracking algorithm from Comaniciu & Meer (2002) is a mode-finding technique that locates the local maxima of the posterior density function. Based on the mean-shift vector, utilised as an estimate of the gradient of the Bhattacharyya function, the new object state estimate is calculated. The accuracy of the mean shift techniques depends on the kernel chosen and the number of iterations in the gradient estimation process. One of the drawbacks of the MS technique is that sometimes local extrema are found instead of the global one. Moreover, the MS algorithm faces problems with multimodal probability density functions which can be overcome by some of the Bayesian methods (sequential Monte Carlo methods).
The MS algorithm has been combined with particle filtering techniques and as a result kernel particle filters (Chang & Ansari (2003; 2005)) and hybrid particle filters (Maggio & Cavallaro (2005)) were proposed combining the advantages of both approaches. The MS is applied to the particles in order to move them into more likely regions and hence the performance of these hybrid particle filters is significantly improved. Interesting implementation of this scheme has been proposed in (Cai et al., 2006) where the data association problem is formulated and the MS algorithm is "embedded seamlessly" into the particle filter algorithm: the deterministic MS - induced particle bias with a superimposed Gaussian distribution is considered as a new proposal distribution. Other related hybrid particle filters combined with the MS have been proposed in (Bai & Liu (2007); Cai et al. (2006); Han et al. (2004); Shan et al. (2007)).

2.2 Bayesian tracking framework

Bayesian inference methods (Doucet et al. (2001); Isard & Blake (1998); Koch (2010); Pérez et al. (2004); Ristic et al. (2004)) have gained a strong reputation for tracking and data fusion applications, because they avoid simplifying assumptions that may degrade performance in complex situations and have the potential to provide an optimal or sub-optimal solution (Arulampalam et al. (2002); Khan et al. (2005)). In case of the sub-optimal solution, the proximity to the theoretical optimum depends on the computational capability to execute numeric approximations and the feasibility of probabilistic models for target appearance, dynamics, and measurements likelihoods.

In the Bayesian tracking framework the best posterior estimate of the state vector $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is inferred from the available measurements, $\boldsymbol{z}_{1:k} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\}$, based on derivation of the posterior probability density function (pdf) of $\boldsymbol{x}_k$ conditioned on the whole set of observations: $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$. Assuming that the posterior pdf at time $k-1$ (the initial pdf) is available, the prior pdf of the state at time k is obtained via the Chapman-Kolmogorov equation:

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \int\limits_{\mathbb{R}^{n_x}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})d\boldsymbol{x}_{k-1} \tag{1}$$

where $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ is the state transition probability. Once the sequence $\boldsymbol{z}_{1:k}$ of measurements is available, the posterior pdf $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ is recursively obtained according to the Bayes update rule

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_{1:k}|\boldsymbol{z}_{1:k-1})} \tag{2}$$

where $p(\boldsymbol{z}_{1:k}|\boldsymbol{z}_{1:k-1})$ is a normalising constant and $p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)$ is the measurement likelihood. Thus, the recursive update of $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ is proportional to the measurement likelihood

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \propto p(\boldsymbol{z}_{1:k}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}). \tag{3}$$

Different strategies can be applied to estimate $\boldsymbol{x}_k$ from this pdf. Commonly used estimators of $\boldsymbol{x}_k$, include the maximum a posteriori (MAP) approach,

$$\hat{\boldsymbol{x}}_k = \arg\max_{\boldsymbol{x}_k} p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}), \tag{4}$$

and the minimum mean squared error (MMSE) approach, giving an estimate which is equivalent to the expected value of the state

$$\hat{\boldsymbol{x}}_k = \int \boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})d\boldsymbol{x}_k. \tag{5}$$

2.3 Particle filtering techniques for state vector estimation

Particle filtering (Arulampalam et al. (2002); Doucet et al. (2001); Isard & Blake (1996; 1998); Pérez et al. (2004); Ristic et al. (2004)) is a method relying on sample-based reconstruction of probability density functions. The aim of sequential particle filtering is to evaluate the *posterior* pdf $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ of the state vector $\boldsymbol{x}_k$, given a set $\boldsymbol{z}_{1:k}$ of sensor measurements up to time k. The quality (importance) of the ℓ^{th} particle (sample) of the state, $\boldsymbol{x}_k^{(\ell)}$, is measured by the weight associated with it, $W_k^{(\ell)}$. An estimate of the variable of interest can be obtained by the weighted sum of particles (cf. (5) and (9)). The pseudo-code description of a generic particle filter (PF) tracking algorithm is shown in Table 1.

Two major stages can be distinguished in the particle filtering method: *prediction* and *update*. During prediction, each particle is modified according to the state model of the region of interest in the video frame, including the perturbation of the particle's state by means of addition of white noise in order to simulate the effect of the random walk according to the

Table 1. Pseudocode of the particle filter algorithm

```
Input: target state $\boldsymbol{x}_{k-1}$ (previous frame)
Output: target state $\boldsymbol{x}_k$ (current frame)

Initialisation
$k = 0$, initialise $\boldsymbol{x}_0$.
Generate $N$ samples (particles) $\{\boldsymbol{x}_0^{(\ell)}\}, \ell = 1, 2, \dots, N$, from the initial distribution $p(\boldsymbol{x}_0)$.
Initialise weights $W_0^{(\ell)} = 1/N$.
• FOR $k = 1 : K_{\text{frames}}$
*   FOR $\ell = 1, 2, \dots, N$
    Prediction
 1. Sample the state from the object motion model
        $\boldsymbol{x}_k^{(\ell)} \sim p(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}^{(\ell)}).$   (6)
    Update
 2. Evaluate the importance weights based on the likelihood $\mathcal{L}(\boldsymbol{z}_k | \boldsymbol{x}_k^{(\ell)})$ of the cue from the measurement $\boldsymbol{z}_k$
        $W_k^{(\ell)} \propto W_{k-1}^{(\ell)} \mathcal{L}(\boldsymbol{z}_k | \boldsymbol{x}_k^{(\ell)}).$   (7)
*   END FOR
    Output
3.  Normalise the weights of each particle
        $\widehat{W}_k^{(\ell)} = W_k^{(\ell)} / \sum_{\ell=1}^{N} W_k^{(\ell)}.$   (8)
4.  Compute the posterior mean state estimate of $\boldsymbol{x}_k$ using the collection of samples
        $\hat{\boldsymbol{x}}_k = \sum_{\ell=1}^{N} \widehat{W}_k^{(\ell)} \hat{\boldsymbol{x}}_k^{(\ell)}.$   (9)
 Resampling
5.  Estimate the effective number of particles $N_{\text{eff}} = 1 / \sum_{\ell=1}^{N} \left(\widehat{W}_k^{(\ell)}\right)^2$. If $N_{\text{eff}} \le N_{\text{thr}}$ ($N_{\text{thr}}$ is a given threshold) then perform resampling: multiply samples $\boldsymbol{x}_k^{(\ell)}$ with high importance weights $\widehat{W}_k^{(\ell)}$ and suppress samples with low importance weights, in order to introduce variety and obtain $N$ new random samples. Set $W_k^{(\ell)} = \widehat{W}_k^{(\ell)} = 1/N$.
• END FOR
```

motion model $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{(\ell)})$, (6). The prior pdf of the state at time k is obtained in prediction stage via Chapman-Kolmogorov equation (1). Once a measurement $\boldsymbol{z}_k$ is available, $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ is recursively obtained in the update step according to (3), or equivalently, (7). The likelihood $\mathcal{L}(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)})$ is calculated for the respective image cue (e.g. colour). Consequently, the posterior mean state is computed using the collection of particles (9).

An inherent problem with particle filters is degeneracy (the case when only one particle has a significant weight). A *resampling* procedure helps to avoid this by eliminating particles with small weights and replicating the particles with larger weights. Various approaches for resampling have been proposed (see Doucet et al. (2001); Kitagawa (1996); Liu & Chen (1998); Wan & van der Merwe (2001), for example). In this work, the systematic resampling method (Kitagawa (1996)) was used with the estimate of the measure of degeneracy (Doucet et al. (2001)) as given in (Liu & Chen (1998)) (see Table 1).

2.4 Importance sampling and proposal distributions

In the PF framework, the pdf of the object state, $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$, is represented by a set of samples with associated weights $\{\boldsymbol{x}_k^{(\ell)}, W_k^{(\ell)}\}_{i=1}^N$ such that $\sum_{i=1}^N W_k^{(\ell)} = 1$. Then the posterior density can be approximated as

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \approx \sum_{\ell=1}^{N} W_k^{(\ell)} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{(\ell)}) \tag{10}$$

based on the likelihood $\mathcal{L}(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)})$ (see the following paragraph for details of the likelihood) of the measurement and particle weights. Here, $\delta(.)$ is the Dirac delta function. The particle weights in (10) are updated based on the principle of importance sampling (Arulampalam et al. (2002))

$$W_k^{(\ell)} \propto W_{k-1}^{(\ell)} \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)}) p(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)})}{q(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k})}, \tag{11}$$

where $q(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k})$ is a proposal, called an *importance density* and $p(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)})$ is the measurement likelihood function. It has been assumed that $q(.)$ is only dependent on $\boldsymbol{x}_{k-1}^{(\ell)}$ and $\boldsymbol{z}_k$. The most popular choice of the importance density is the prior, $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$. This choice results in a simple implementation of the weight update stage (cf. (11))

$$W_k^{(\ell)} \propto W_{k-1}^{(\ell)} p(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)}) . \tag{12}$$

However, using the transition information alone may not be sufficient to capture the complex dynamics of some targets. It has been shown that an optimal importance density is defined as function of the state and a new measurement/additional information $q(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k})$. Therefore, in this work, the use of a mixture distribution containing additional information as the importance density is proposed

$$q(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k}) = \sum_{m=1}^{M} \alpha_m f_m(\boldsymbol{x}_{1:k}^{(\ell)}, \boldsymbol{z}_{1:k}) \tag{13}$$

where $\alpha_m, \sum_{m=1}^M \alpha_m = 1$ are normalised weights of M components of the mixture. Among possible candidates for f_m are the prior, blob detection and data association distributions. For

$M = 1$ and $f_1(\boldsymbol{x}_{1:k}^{(\ell)}, \boldsymbol{z}_{1:k}) = p(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)})$ the generic PF is obtained. Examples of such mixture importance densities have been proposed in (Cai et al. (2006); Lu et al. (2009); Okuma et al. (2004)), consisting in an inclusion of the Adaboost detection information and a modification of the particle distribution with the use of a mode-seeking algorithm (M-S has been used). In this case the 'proposal distribution' has been defined as a mixture distribution between the prior and detection distributions:

$$q(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k}) = \alpha p_{\text{ada}}(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{z}_k) + (1-\alpha) p(\boldsymbol{x}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}) \tag{14}$$

In (Cai et al., 2006) the application of M-S optimisation to the particles is considered as a new proposal distribution:

$$\breve{q}(\breve{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k}) = \mathcal{N}(\breve{\boldsymbol{x}}_k^{(\ell)}|\tilde{\boldsymbol{x}}_k^{(\ell)}, \boldsymbol{\Sigma}) , \tag{15}$$

where $\tilde{\boldsymbol{x}}_k^{(\ell)}$ are M-S-modified samples of the original proposal distribution q and $\mathcal{N}(\breve{\boldsymbol{x}}_k^{(\ell)}|\tilde{\boldsymbol{x}}_k^{(\ell)}, \boldsymbol{\Sigma})$ is a Gaussian distribution, with mean $\tilde{\boldsymbol{x}}_k^{(\ell)}$ fixed covariance $\boldsymbol{\Sigma}$, superimposed on the results of M-S. The particle weights are then updated accordingly, i.e.

$$W_k^{(\ell)} \propto W_{k-1}^{(\ell)} \frac{p(\boldsymbol{z}_k|\breve{\boldsymbol{x}}_k^{(\ell)}) p(\breve{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)})}{q(\breve{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \boldsymbol{z}_{1:k})} . \tag{16}$$

3. The structural information approach

The recently proposed approach combining the SSIM and particle filtering for video tracking has been shown in (Loza et al., 2009) to outperform similar methods using the conventional colour or edge histograms and Bhattacharyya distance. However, the structural similarity combined with the particle filtering approach results in increased computational complexity of the algorithm due to the necessity of extracting the structural information at each point of the state space. In this book chapter, novel optimised approaches based on the SSIM are proposed for video tracking. Firstly, a fast, deterministic version of the SSIM-based tracking algorithm is developed. The deterministic tracking algorithm estimates the state of the target (location and size) combining a gradient ascent procedure with the structural similarity surface of the current video frame, thus avoiding computationally expensive sampling of the state space. Next, an optimisation scheme is presented, based on a hybrid PF with a deterministic mode search, applied to the particle distribution.

3.1 Structural similarity measure

The proposed method uses a similarity measure computed directly in the image spatial domain. This approach differs significantly from other particle filtering algorithms, that compare image distributions represented by their sample histograms (Nummiaro et al. (2003); Pérez et al. (2004); Shen et al. (2003)).

Although many simple image similarity measures exist (for example, mean square error, mean absolute error or peak signal-to-noise ratio), most of these have failed so far to capture the perceptual similarity of images/video frames under the conditions of varied luminance, contrast, compression or noise (Wang et al. (2004)). Recently, based on the premise that the HVS is highly tuned to extracting structural information, a new image metric has been developed, called the Structural SIMilarity (SSIM) index (Wang et al. (2004)). The SSIM index, between two images, I and J is defined as follows:

$$\begin{aligned} S(I,J) &= \left(\frac{2\mu_I\mu_J + C_1}{\mu_I^2 + \mu_J^2 + C_1}\right)\left(\frac{2\sigma_I\sigma_J + C_2}{\sigma_I^2 + \sigma_J^2 + C_2}\right)\left(\frac{\sigma_{IJ} + C_3}{\sigma_I\sigma_J + C_3}\right) \\ &= l(I,J)\, c(I,J)\, s(I,J), \end{aligned} \tag{17}$$

where $C_{1,2,3}$ are small positive constants used for the numerical stability purposes, μ denotes the sample mean

$$\mu_I = \frac{1}{L}\sum_{j=1}^{L} I_j, \tag{18}$$

σ denotes the sample standard deviation

$$\sigma_I = \sqrt{\frac{1}{L-1}\sum_{j=1}^{L}(I_j - \mu_I)^2} \tag{19}$$

and

$$\sigma_{IJ} = \frac{1}{L-1}\sum_{j=1}^{L}(I_j - \mu_I)(J_j - \mu_J) \tag{20}$$

corresponds to the sample covariance. The estimators are defined identically for images I and J, each having L pixels. The image statistics are computed in the way proposed in (Wang et al. (2004)), i.e. locally, within a 11×11 normalised circular-symmetric Gaussian window.

For $C_3 = C_2/2$, (17) can be simplified to obtain

$$S(I,J) = \left(\frac{2\mu_I\mu_J + C_1}{\mu_I^2 + \mu_J^2 + C_1}\right)\left(\frac{2\sigma_{IJ} + C_2}{\sigma_I^2 + \sigma_J^2 + C_2}\right). \tag{21}$$

3.2 Selected properties of the SSIM

The three components of (17), l, c and s, measure respectively the *luminance*, *contrast* and *structural similarity* of the two images. Such a combination of image properties can be seen as a fusion of three independent image cues. The relative independence assumption is based on a claim that a moderate luminance and/or contrast variation does not affect structures of the image objects (Wang et al. (2005a)).

In the context of the multimodal data used in our investigation, an important feature of the SSIM index is (approximate) invariance to certain image distortions. It has been shown in (Wang et al. (2005a; 2004)), that the normalised luminance measurement, l, is sensitive to the relative rather than to absolute luminance change, thus following the masking feature of the Hue, Saturation, Value (HVS).

Similarly, the contrast comparison function, c, is less sensitive to contrast changes occurring in images with high base contrast. Finally, the structure comparison, s, is performed on contrast-normalised signal with mean luminance extracted, making it immune to other (non-structural) distortions.

These particular invariance properties of the SSIM index make it suitable for the use with multimodal and surveillance video sequences. The similarity measure is less sensitive to

the type of global luminance and contrast changes produced by infrared sensors (results of varied thermal conditions or exposure of the object) and visible sensors (for example, the object entering dark or shadowy areas or operating in variable lighting conditions). Moreover, the structure comparison is expected to be more reliable in scenarios when spurious objects appear in the scene or when there is not enough discriminative colour information available. The latter may be the result of the tracked object being set against background of similar colour or when background-like camouflage is deliberately being used.
It can easily be shown that the measure defined in (17) is symmetric, i.e.

$$S(I,J) = S(J,I) \tag{22}$$

and has a unique upper bound

$$S(I,J) \leq 1,\ S(I,J) = 1 \text{ iff } I = J. \tag{23}$$

One way of converting such a similarity $S(I,J)$ into dissimilarity $D(I,J)$ is to take (Loza et al. (2009); Webb (2003))

$$D(I,J) = \frac{1}{|S(I,J)|} - 1. \tag{24}$$

Here a more natural way Webb (2003),

$$D(I,J) = (1 - S(I,J))/2. \tag{25}$$

is preferred, however, as it maps the dissimilarity into [0, 1] interval (0 when the images are identical). The measure (25) satisfies non-negativity, reflexivity and symmetry conditions. Although sufficient for our purposes, this dissimilarity measure is not a metric, as it does not satisfy the triangle condition. In the following Section we present a method of evaluating the likelihood function, based on the structural similarity between two greyscale images.

3.3 The structural information particle filter tracking algorithm

Below the main constituents of the structural similarity-based particle filter tracking algorithm (SSIM-PF), such us motion, likelihood and target model, are described. A pseudocode of the algorithm is shown in Table 2.

3.3.1 Motion model

The motion of the moving object can be modelled by the random walk model,

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1}, \tag{26}$$

with a state vector $\boldsymbol{x}_k = (x_k, y_k, s_k)^T$ comprising the pixel coordinates (x_k, y_k) of the centre of the region surrounding the object and the region scale s_k; $\boldsymbol{F}$ is the transition matrix ($\boldsymbol{F} = \boldsymbol{I}$ in the random walk model) and $\boldsymbol{v}_k$ is the process noise assumed to be white, Gaussian, with a covariance matrix

$$\boldsymbol{Q} = \text{diag}(\sigma_x^2, \sigma_y^2, {\sigma_s}^2). \tag{27}$$

The estimation of the scale permits adjustment of the region size of the moving objects, e.g., when it goes away from the camera, when it gets closer to it, or when the camera zoom varies. Depending on the type of the tracking object and the environment in which tracking is

Table 2. Pseudocode of the SSIM-based particle filter algorithm

```
Input: target state $\boldsymbol{x}_{k-1}$ (previous frame)
Output: target state $\boldsymbol{x}_k$ (current frame)

Initialisation
$k = 0$, initialise tracked region at $\mathbf{x}_0$.
Generate $N$ samples (particles) $\{\boldsymbol{x}_0^{(\ell)}\}, \ell = 1, 2, \ldots, N$, from the initial distribution $p(\boldsymbol{x}_0)$.
Initialise weights $W_0^{(\ell)} = 1/N$.
• FOR $k = 1 : K_{\text{frames}}$
*   FOR $\ell = 1, 2, \ldots, N$
    Prediction
 1. Sample the state from the object motion model $\boldsymbol{x}_k^{(\ell)} \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{(\ell)})$.
    Update
 3. Evaluate the importance weights according to 29:
        $W_k^{(\ell)} \propto W_{k-1}^{(\ell)} \mathcal{L}(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)}).$    (28)
*   END FOR

    Output
4.  Normalise the weights of each particle (8)
5.  Compute the posterior mean state estimate of $\boldsymbol{x}_k$ (9).
 Resampling
6.  Perform resampling as described in Table 1
• END FOR
```

performed, the state vector can be extended to include, for example, the acceleration variables, and the fixed ratio condition can be relaxed allowing independent changes of the height and the width of the object. However, increased dimensionality of the state vector requires finer sampling of the state space, and thus undesirably high number of particles, which may preclude real-time implementation of the tracking system.

3.3.2 Likelihood model

The distance between the reference (target) region $\boldsymbol{t}_{ref}$ and the current region $\boldsymbol{t}_{\boldsymbol{k}}$ is calculated by the similarity measure (25). The normalised distance between the two regions is then substituted into the likelihood function, modelled as an exponential:

$$\mathcal{L}(\boldsymbol{z}_k|\boldsymbol{x}_k^{(\ell)}) \propto \exp\left(-D^2(\boldsymbol{t}_{\text{ref}}, \boldsymbol{t}_k)/D_{\text{min}}^2\right), \tag{29}$$

where $D_{\text{min}} = \min_{\boldsymbol{x}}\{D(\boldsymbol{t}_{\text{ref}}, \boldsymbol{t}_k)\}$. Here $\boldsymbol{z}$ denotes the measurement vector, although with the SSIM a measurement in explicit form is not available. This smooth likelihood function, although chosen empirically by (Pérez et al. (2004)), has been in widespread use for a variety of cues ever since. The similarity-based distance proposed in this work is an alternative to the Bhattacharyya distance D, commonly used to calculate similarity between target and reference objects, described by their histograms h:

$$D(\boldsymbol{t}_{\text{ref}}, \boldsymbol{t}_k) = \left(1 - \sum_{i=1}^{B} h_{\text{ref},i} h_{k,i}\right)^{0.5}. \tag{30}$$

where the tracked image regions are described by their colour (Nummiaro et al. (2003)) or texture histograms (Brasnett et al. (2007)). The likelihood function is then used to evaluate the importance weights of the particle filter, to update the particles and to obtain the overall estimate of the centre of the current region.

3.3.3 Target model

The tracked objects are defined as image regions within a rectangle or ellipsoid specified by the state vector (i.e. spatial location and scale). In the particle filtering framework as specified in Table 1, a region corresponding to each particle, centred at location (x, y) and resized according to the scale parameter of the state, is computed. The extracted region is then compared to the target region using the distance measure D (25). The structural properties of the region extracted through SSIM (17) are related with the estimates of the centre of the region of interest and are used directly to calculate the distance D in (29) between the reference and current region as shown in (25).

3.4 Differential SSIM tracking algorithm

In the SSIM-PF tracking algorithm, described in Section 3.3, the SSIM is computed a large number of times, i.e. for each particle. This makes the SSIM-PF method computational expensive when a large number of particles is required. In this section, a low-complexity, deterministic alternative to the SSIM-PF is proposed, namely Differential SSIM-based tracker (DSSIM). The proposed algorithm tracks the object by analysing the gradient SSIM surface computed between the current video frame and the object model. This deterministic iterative gradient search procedure uses the structural information directly and does not rely on the probabilistic framework introduced in Section 2.2.

In order to achieve a computationally efficient tracking performance, whilst retaining the benefits of the original measure, a differential SSIM formula is proposed as follows. The object is tracked in the spatial domain of the subsequent video frames by maximising the measure (21) with respect to location $\mathbf{x}$, based on its gradient. In order to simplify the subsequent derivation, we choose to analyse the logarithm of (21) by defining a function $\rho(\boldsymbol{x})$:

$$\begin{aligned}\rho(\boldsymbol{x}) &= s\log(|S(\boldsymbol{x})|) \qquad (31)\\ &= s\log(2\mu_I\mu_J + C_1) - \log(\mu_I^2 + \mu_J^2 + C_1) + \log(2|\sigma_{IJ}| + C_2) - \\ &\qquad \log(\sigma_I^2 + \sigma_J^2 + C_2). \qquad (32)\end{aligned}$$

Table 3. Pseudocode of the proposed DSSIM tracking algorithm

```
Input: target state $\boldsymbol{x}_{k-1}$ (previous frame)
Output: target state $\boldsymbol{x}_k$ (current frame)

Initialisation
$k = 0$, initialise tracked region at $\boldsymbol{x}_0$.
• FOR $k = 1 : K_{\text{frames}}$

0. Initialise $\boldsymbol{x}_k^{(0)} = \boldsymbol{x}_k^{(1)} = \boldsymbol{x}_{k-1}$

*  WHILE $S(\boldsymbol{x}_k^{(1)}) \geq S(\boldsymbol{x}_k^{(0)})$
   1. Assign $\boldsymbol{x}_k^{(0)} = \boldsymbol{x}_k^{(1)}$
   2. Calculate $\nabla\rho(\boldsymbol{x}_k^{(0)})$ according to (39)
   3. Assign $\boldsymbol{x}_k^{(1)}$ the location of a pixel in $\boldsymbol{x}_k^{(0)}$ 8-connected neighbourhood, along the
      direction of $\nabla\rho(\boldsymbol{x}_k^{(0)})$
*  END WHILE
Output
4. Assign target location in the current frame $\boldsymbol{x}_k = \boldsymbol{x}_k^{(0)}$
• END FOR
```

where $S(\boldsymbol{x})$ denotes the similarity (21) between the object template J and a current frame image region I centered around the pixel location $\boldsymbol{x} = (x, y)$ and $s = \operatorname{sign}(S(\boldsymbol{x}))$. After a simple expansion of (31) we obtain the expression for the gradient of the function $\rho(\boldsymbol{x})$

$$\nabla\rho(\boldsymbol{x}) = s\left(A_1\nabla\mu_I + A_2\nabla\sigma_I^2 + A_3\nabla\sigma_{IJ}\right)\,, \tag{33}$$

where

$$A_1 = \frac{2\mu_J}{2\mu_I\mu_J + C_1} - \frac{2\mu_I}{\mu_I^2 + \mu_J^2 + C_1}\,, \tag{34}$$

$$A_2 = -\frac{1}{\sigma_I^2 + \sigma_J^2 + C_2}\,, \quad A_3 = \frac{1}{2\sigma_{IJ} + C_2}\,. \tag{35}$$

The gradients $\nabla\mu_I$ and $\nabla\sigma_I^2$ can be calculated as follows

$$\nabla\mu_I = \frac{1}{L}\sum_{i=1}^{L}\nabla I_i\,, \tag{36}$$

$$\nabla\sigma_I^2 = \frac{2}{L-1}\sum_{i=1}^{L}(I_i - \mu_I)\nabla I_i\,. \tag{37}$$

A simplified expression for the covariance gradient, $\nabla\sigma_{IJ}$, can be obtained, based on the observation that $\sum_{i=1}^{L}(J_i - \mu_J) = 0$:

$$\begin{aligned}\nabla\sigma_{IJ} &= \frac{1}{L-1}\sum_{i=1}^{L}(J_i - \mu_J)(\nabla I_i - \nabla\mu_I) \\ &= \frac{1}{L-1}\sum_{i=1}^{L}(J_i - \mu_J)\nabla I_i\end{aligned} \tag{38}$$

Finally, by defining the gradient of the pixel intensity as $\nabla I_i = \left(\frac{\partial I_i}{\partial x}, \frac{\partial I_i}{\partial y}\right)^T$, the complete formula for $\nabla\rho(\boldsymbol{x})$ is obtained

$$\nabla\rho(\boldsymbol{x}) = s\sum_{i=1}^{L}\left(\frac{A_1}{L} + \frac{2A_2(I_i - \mu_I) + A_3(J_i - \mu_J)}{L-1}\right)\nabla I_i \,. \tag{39}$$

The proposed algorithm, employing the gradient DSSIM function (39) is summarised in Table 3. In general terms, the estimated target location, $\mathbf{x}_k^{(0)}$ is moved along the direction of the structural similarity gradient by one pixel in each iteration until no further improvement is achieved. The number of SSIM and gradient evaluations depends on the number of iterations needed to find the maximum of the measure $S(\mathbf{x})$ and on average does not exceed 5 in our experiments. This makes our approach significantly faster than the original SSIM-PF. It should be noted that although the differential framework of the algorithm is based on a reformulation of the scheme proposed in (Zhao et al. (2007)), it utilises a distinct similarity measure.

3.5 The hybrid SSIM-PF tracker algorithm

An extension to the SSIM-PF, by deterministically modifying each particle according to the local structural similarity surface, referred to as hybrid SSIM-PF, is proposed in this correspondence. In the DSSIM procedure described in Section 3.4, the estimated target location, $\boldsymbol{x}_k^{(0)}$ is moved along the direction the structural similarity gradient by one pixel in each iteration until no further improvement is achieved, or the limit of iterations is reached. In the hybrid scheme proposed here this step is performed for each particle, following its prediction (step 1. in Table 1). In accordance with the principle of importance sampling (see Section 2.4), the prior distribution p resulting from the particle prediction and the proposal distribution q centred on the optimised position of the particle in the state space, are used to re-calculate the weight of a resulting particle $\tilde{\boldsymbol{x}}^{(\ell)}$:

$$W_k^{(\ell)} \propto W_{k-1}^{(\ell)} \frac{\mathcal{L}(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(\ell)})p(\tilde{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)})}{q(\tilde{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \mathbf{z}_k)} \,. \tag{40}$$

with the proposal distribution defined analogously to Lu et al. (2009)

$$q(\tilde{\boldsymbol{x}}_k^{(\ell)}|\tilde{\boldsymbol{x}}_{k-1}^{(\ell)}, \boldsymbol{z}_k) = \alpha p_{\mathrm{DSSIM}}(\tilde{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{z}_k) + (1-\alpha)p(\tilde{\boldsymbol{x}}_k^{(\ell)}|\tilde{\boldsymbol{x}}_{k-1}^{(\ell)}) \,. \tag{41}$$

In our implementation of this algorithm the mixing parameter is set to $\alpha = 0.5$ resulting in a uniform mixture distribution of two Gaussian distributions with identical covariances (27),

Table 4. Pseudocode of the hybrid particle filter algorithm

```
Input: target state $\boldsymbol{x}_{k-1}$ (previous frame)
Output: target state $\boldsymbol{x}_k$ (current frame)

Initialisation
$k = 0$, initialise tracked region at $\boldsymbol{x}_0$.
Generate $N$ samples (particles) $\{\boldsymbol{x}_0^{(\ell)}\}, \ell = 1, 2, \ldots, N$, from the initial distribution $p(\boldsymbol{x}_0)$.
Initialise weights $W_0^{(\ell)} = 1/N$.
• FOR $k = 1 : K_{\text{frames}}$
*   FOR $\ell = 1, 2, \ldots, N$
    Prediction
 1. Sample the state from the object motion model $\boldsymbol{x}_k^{(\ell)} \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{(\ell)})$.
    Optimisation
 2. Modify the particle associated with the state $\boldsymbol{x}_k^{(\ell)}$ by performing steps 0.–4., Table 3.
        Assign the modified state to $\tilde{\boldsymbol{x}}_k^{(\ell)}$.
    Update
 3. Evaluate the importance weights
        $$W_k^{(\ell)} \propto W_{k-1}^{(\ell)} \frac{\mathcal{L}(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(\ell)})p(\tilde{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)})}{q(\tilde{\boldsymbol{x}}_k^{(\ell)}|\boldsymbol{x}_{k-1}^{(\ell)}, \mathbf{z}_k)} \,. \qquad (42)$$
    with proposal distribution $q$ defined as in (41).
*   END FOR

    Output
 4. Normalise the weights of each particle (8)
 5. Compute the posterior mean state estimate of $\boldsymbol{x}_k$ (9).
  Resampling
 6. Perform resampling as described in Table 1
• END FOR
```

centred on the motion model-predicted particle and its optimised version, respectively. The proposed method is described in the form of a pseudocode in Table 4.

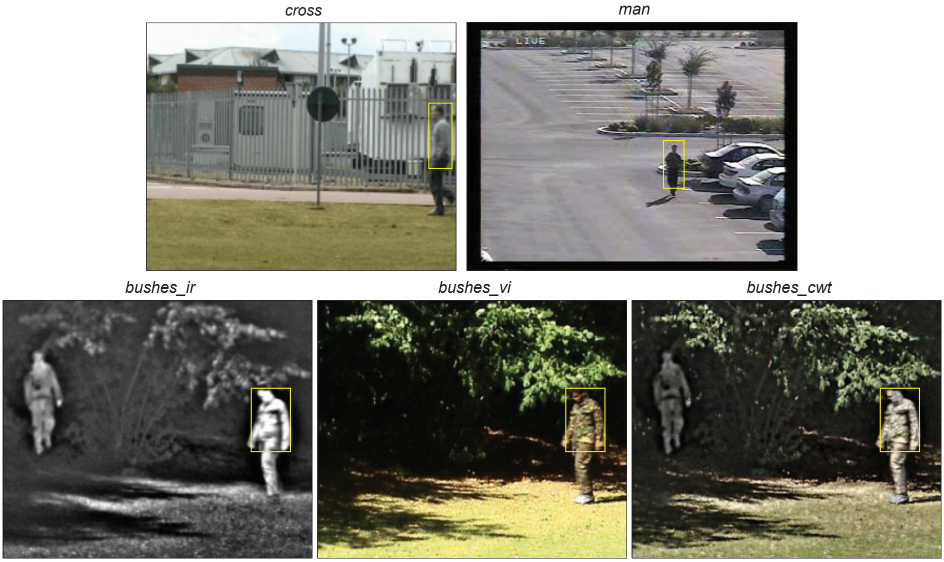


Fig. 1. Reference frames from the test videos

4. Tracking performance

4.1 Evaluation metrics

Tracking algorithms are usually evaluated based on whether they generate correct mobile object trajectories. In addition to the commonly applied visual assessment of the tracking performance, qualitative measures can be used to provide formal comparisons of the tested algorithms. In our work, the Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(k) = \left(\frac{1}{M} \sum_{m=1}^{M} (x_k - \hat{x}_{k,m})^2 + (y_k - \hat{y}_{k,m})^2 \right)^{\frac{1}{2}} \tag{43}$$

has been used as numerical measure of the performance of the developed techniques. In (43) $(\hat{x}_{k,m}, \hat{y}_{k,m})$ stand for the upper-left corner coordinates of the tracking box determined by both the object's central position and the scale estimated by the tracking algorithm in the frame k in m-th independent simulation (in our simulations $M = 50$ for probabilistic tracking algorithms and $M = 1$ for DSSIM and MS). The corresponding ground truth positions of the object, (x_k, y_k), have been generated by manually tracking the object.

4.2 Video sequences

The performance of our method is demonstrated over various multimodal video sequences, in which we aim to track a pre-selected moving person. The sequence *cross* (5 sec duration), taken from our multimodal database *The Eden Project Multi-Sensor Data Set* (2006), contains three people walking rapidly in front of a stationary camera. The main difficulties posed by this sequence are: the colour similarity between the tracked object and the background or other passing people, and a temporal near-complete occlusion of the tracked person by a passer-by.

Seq.	mean RMSE				std RMSE			
name	colour	edges	col.&edges	SSIM	colour	edges	col.&edges	SSIM
cross	150.5	77.4	39.6	8.3	98.4	70.1	58.2	5.1
man	71.5	27.7	48.4	8.0	46.1	23.7	34.3	6.5
bushes_ir	71.9	30.7	26.9	21.0	40.8	9.6	7.8	7.5
bushes_vi	98.4	36.0	36.4	19.1	13.1	15.7	16.7	7.1
bushes_cwt	92.6	45.4	32.0	20.7	54.6	21.0	12.1	7.5

Table 5. The performance evaluation measures of the tracking simulations

The sequence *man* (40 sec long), has been obtained from PerceptiVU, Inc. (n.d.). This is a video showing a person walking along a car park. Apart from the object's similarity to the nearby cars and the shadowed areas, the video contains numerous instabilities. These result from a shaking camera (changes in the camera pan and tilt), fast zoom-ins and zoom-outs, and a altered view angle towards the end of the sequence.

The three multimodal sequences *bushes* (*The Eden Project Multi-Sensor Data Set* (2006)), contain simultaneous registered infrared (*ir*), visual (*vi*) and complex wavelet transform fused (*cwt*, see Lewis et al. (2007) for details) recordings of two camouflaged people walking in front of a stationary camera (10 sec). The tracked individual looks very similar to the background. The video contains changes in the illumination (the object entering shadowy areas) together with nonstationary surroundings (bushes moved by strong wind). The reference frames used in tracking are shown in Figure 1.

4.3 Comparison of tracking cues

In this section the commonly used tracking cues (colour, edge histograms and their combination (Brasnett et al. (2007); Nummiaro et al. (2003))) are compared with the cue based on the structural similarity information. In order to facilitate easy and fair comparison the cues are evaluated in the same PF framework with identical initialisation and common parameters. The reference targets shown in Figure 1 were tracked in 50 Monte Carlo simulations and then the tracking output of each cue has been compared to the ground truth. The exemplary frames showing the tracking output are given in Figures 2–4 and the mean of RMSE and its standard deviation (std) were computed and are presented in Table 5.

From inspection of the video output in Figures 2–4 and the tracking error statistics in Table 5 it can clearly be seen that the SSIM-based method outperforms the other methods in all instances while never loosing the tracked object. The colour-based PF algorithm is the most prone to fail or give imprecise estimates of the object's state. Combining edge and colour cues is usually beneficial, however in some cases (sequences *man* and *bushes_vi*) the errors of the colour-based PF propagate through the performance of the algorithm, making it less precise than the PF based on edges alone. Another observation is that the 'structure' tracking algorithm has been least affected by the modality of *bushes* and the fusion process, which demonstrates the robustness of the proposed method to luminance and contrast alterations.

A closer investigation of the selected output frames illustrates the specific performance of the different methods. Figures 2–4 show the object tracking boxes constructed from the mean locations and scales estimated during the tests. Additionally, the particles and object location obtained from one of the Monte Carlo trials are shown. Since very similar performance has been obtained for all three *bushes* videos, only the fused sequence, containing complementary information from both input modalities, is shown. The visual difference between contents of

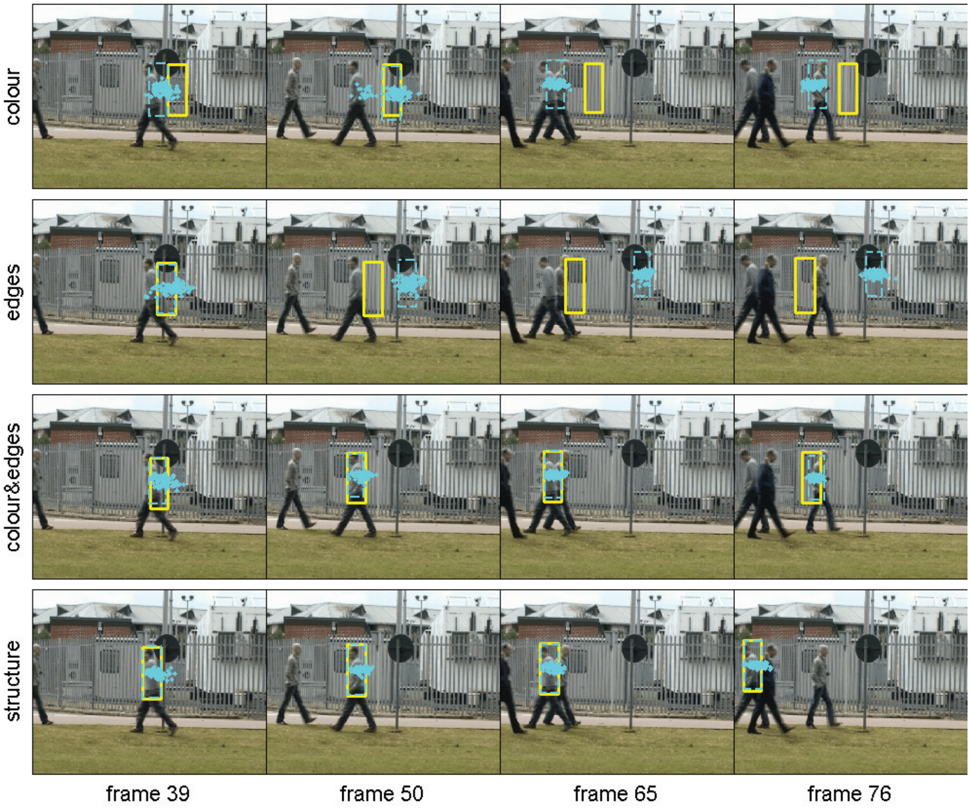


Fig. 2. Example video frames with average output of the tracking algorithm (solid line rectangle), a single trial output (dashed line rectangle and particles) superimposed, sequence *cross*

the input *bushes* videos (colour information, a person hidden in shaded area) can be seen by comparing the reference frames in Figure 1.

In the sequence *cross*, Figure 2, the 'colour' and 'edges' tracking algorithms are distracted by the road sign, which eventually leads to the loss of the object. Then, the first non-occluding passer-by causes the 'colour&edges' cue tracking algorithm to loose the object (frame 65). The 'structure' tracking technique is not distracted even by the temporary occlusion (frame 76).

The shaking camera in the sequence *man* (Figure 3, frame 162), has less effect on the 'structure' tracking technique than on the other compared algorithms, which appear to choose the wrong scale of the tracking box. Moreover, the other considered tracking algorithms do not perform well in case of similar dark objects appearing close-by (shadows, tyres, frame 478, where the 'colour' tracking algorithm permanently looses object) and rapid zoom-in (frame 711) and zoom-out of the camera (frame 790). Our method, however, seems to cope well with both situations. It should be noted, however, that 'colour&edges' (and 'edges') based algorithms show a good ability of recovering from some of the failings.

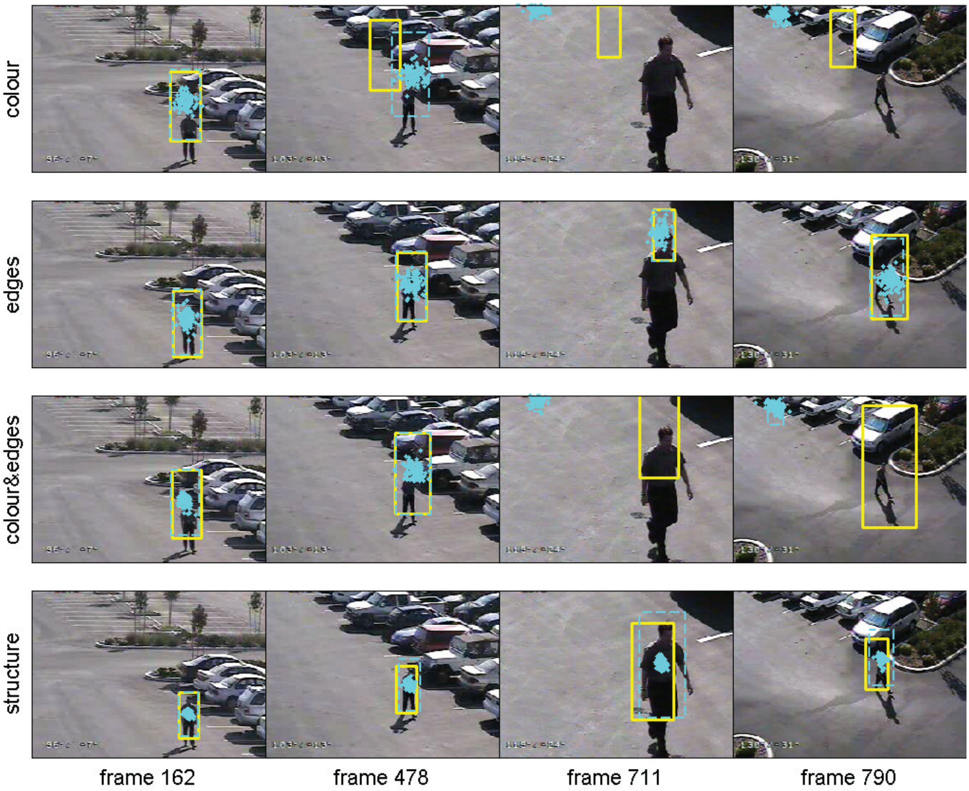


Fig. 3. Example video frames with average output of the tracking algorithm (solid line rectangle), a single trial output (dashed line rectangle and particles) superimposed, sequence *man*

Similarly, in the multimodal sequence *bushes*, Figure 4, the proposed 'structure' tracking algorithm is the most precise and the 'colour' tracking algorithm the least precise. The use of the fused video, although resulting in slightly deteriorated performance of the 'edges' based tracking algorithm, can still be motivated by the fact that it retains complementary information useful both for the tracking algorithm and a human operator (Cvejic et al. (2007); Mihaylova et al. (2006)): contextual information from the visible sequence and a hidden object location from the infrared sequence.

A single-trial output shown in Figures 2–4 exemplifies the spread of the spatial distribution of the particles. Typically, in the 'structure' tracking technique, particles are the most concentrated. Similar features can be observed in the output of the 'colour&edges' tracking algorithm. The particle distribution of the remaining PF tracking algorithms is much more spread, often attracted by spurious objects (see Figures 2 and 3, in particular).

It should also be noted that, the tracking performance varies between realisations, often giving different results compared with the output averaged over all Monte Carlo trials. Also in this

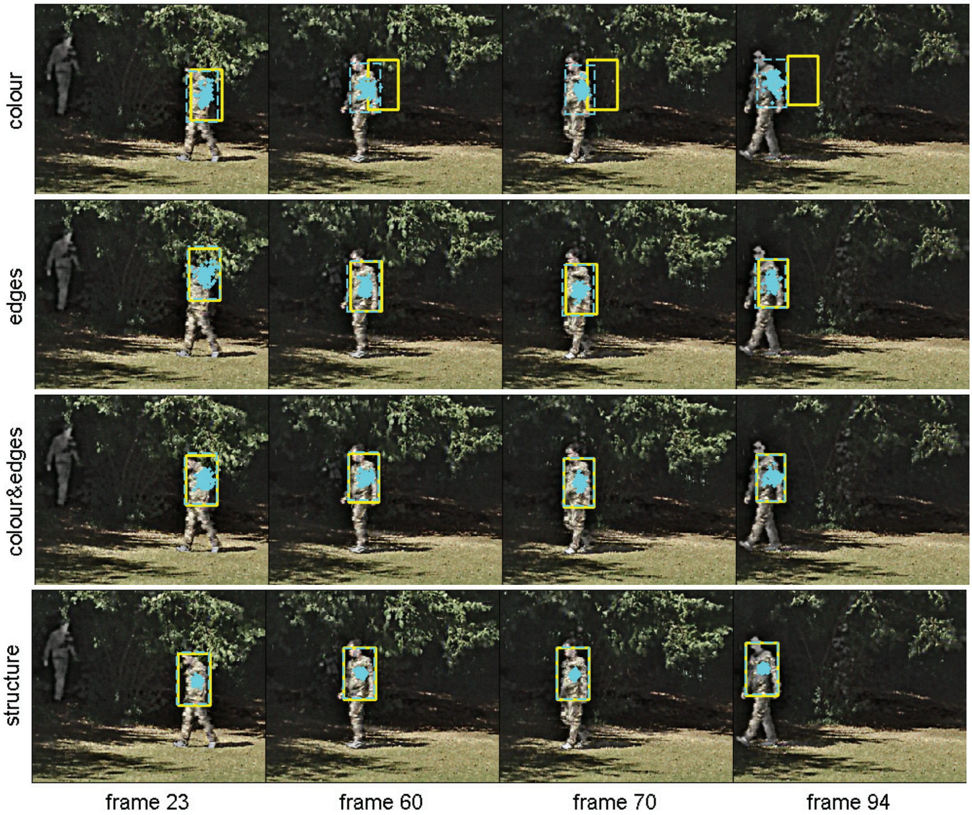


Fig. 4. Example video frames with average output of the tracking algorithm (solid line rectangle), a single trial output (dashed line rectangle and particles) superimposed, sequence *bushes_cwt*

respect the proposed method has been observed to be the most consistent, i.e., its results had the lowest variation, as illustrated by the low values of std RMSE in Table 5.

4.4 Comparison of SSIM-based probabilistic and deterministic techniques

In this section the probabilistic tracking SSIM-PF (Section 3.3) is compared with its deterministic counterpart, DSSIM-PF (Section 3.4). Since the main motivation for development of DSSIM technique was the reduction of the computational complexity, the algorithms are also evaluated with respect to their execution speed and therefore the rate at which the video frames are processed by the tracking algorithms, measured as frames per second (FPS), has been included in the results shown in Table 6. The proposed algorithms have been compared with another deterministic technique, the MS algorithm (Comaniciu & Meer (2002)). Analogously to PF-based methods, the MS and DSSIM algorithms are made scale-adaptive, by varying the object size by 5% and choosing the size giving the best match in terms of the similarity measure used.

Seq. name	Image size (pixels)	Speed (fps)			Mean RMSE (pixels)			std RMSE (pixels)		
		MS	SSIM-PF	DSSIM	MS	SSIM-PF	DSSIM	MS	SSIM-PF	DSSIM
cross	720 × 576	27	13	71	37.3	8.3	5.6	62.2	5.1	4.3
man	320 × 240	109	53	315	18.2	8.0	7.0	13.1	6.5	4.9

Table 6. The performance evaluation measures of the tracking simulations

Based on the performance measures in Table 5, it can be concluded that DSSIM outperforms the MS and SSIM-PF, both in terms of the processing speed and the tracking accuracy. It also appears to be more stable than the other two methods (lowest std). Although the example frames in Figure 5 reveal that in a number of instances the methods perform comparably, it can be seen that DSSIM method achieves the overall best performance in most of the frames. Admittedly, the difference between the accuracy and the stability of SSIM-PF and DSSIM is not large in most cases, however, in terms of the computational complexity, DSSIM method compares much more favourably with the other two techniques. The average tracking speed estimates were computed on PC in the following setup: CPU clock 2.66 GHZ, 1G RAM, MS and DSSIM requiring on average 20 and 5 iterations, respectively, and PF using 100 particles. In terms of the relative computational efficiency, the proposed method has been found to be approximately four times faster than SSIM-PF and twice as fast as MS.

The exemplary frames in Figure 5, where the ’difficult’ frames have been selected, offer more insight into the performance and robustness of the algorithms. In the *cross* sequence, neither SSIM-PF nor DSSIM are distracted by the temporary occlusion of the tracked person by other passer-by, whereas the MS algorithm locks onto a similar object moving in the opposite direction. Likewise, although all the three algorithms manage to follow the target in *man* sequences, the gradient structural similarity method identifies the scale and the position of the object with the best accuracy.

4.4.1 Performance evaluation of the extension of the SSIM-based tracking algorithm

Below, we present a performance analysis of the hybrid structural similarity-based PF algorithm. For the sake of completeness, six competing algorithms has been tested and compared: colour-based PF algorithm COL-PF, SSIM-PF, their hybridised versions, hybrid SSIM-PF-DSSIM (Section 3.5) and hybrid COL-PF-MS (based on procedure proposed in (Lu et al. (2009))), and two deterministic procedures themselves (DSSIM and MS). A discussion of the results based on the visual observation of tracking output in the *cross* sequence is provided below and the specific features of the algorithms tested are pointed out. Figure 6 presents the extracted frames of the output of the six tracking algorithms.

It should be noted that in order to illustrate the benefit of using the optimisation procedures, a very low number of the particles for PF-based methods has been chosen (20 for SSIM-based and 30 for colour-based PF). Consequently, it allowed us to observe whether the resulting tracking instability and failures are partially mitigated by the use of the optimisation procedures. Moreover, since the optimisation procedures are much faster than PFs, such a combination does not increase the computational load considerably. On the contrary, the appropriate combination of the two methods, results in a lower number of the particles required and thus reducing the processing time. Conversely, it can be shown that, a non-optimised tracking algorithms can achieve a similar performance to the optimised tracking algorithm utilising a larger number of particles and thus being more computationally demanding.

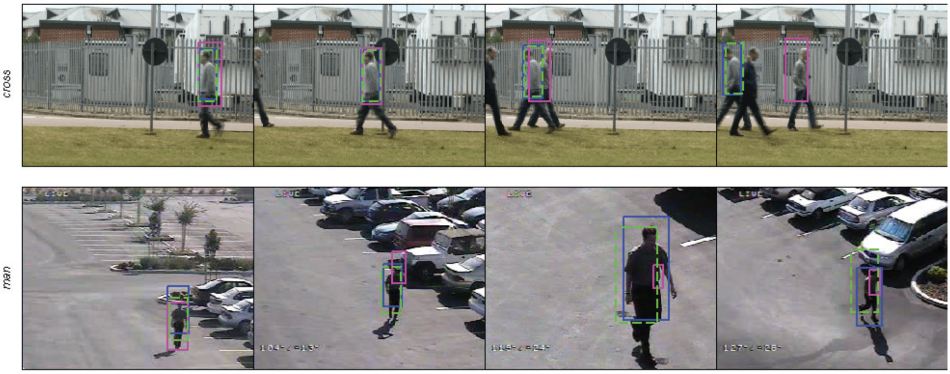


Fig. 5. Example video frames with output of the tracking algorithm output superimposed: DSSIM solid blue rectangle, SSIM-PF dashed green rectangle and MS solid magenta rectangle

Based on the observation of the estimated target regions in Figure 6, it can be concluded that the gradient structural similarity procedure locates the object precisely in majority of the frames. It fails, however, to recover from an occlusion towards the end of the sequence. The performance of the SSIM-PF is very unstable, due to a very low number of particles used and it looses the object half-way through the sequence. On the other hand, the combined algorithm, SSIM-PF-DSSIM, tracks the object successfully throughout the sequence. The MS algorithm has completely failed to track the object. Since the MS algorithm is a memory-less colour-based tracking algorithm, its poor performance in these sequence is due to the object's fast motion and its similarity to the surrounding background. The colour-based algorithm, COL-PF, performs similarly to SSIM-PF, however, it locates the object somewhat more precisely. Finally, the combined COL-PF-MS algorithm, appears to be more stable than its non-optimised version. Nevertheless, the objects is eventually lost as a result of the occlusion.

Finally, to illustrate a potential of further extension of the SSIM-PF, a type of target distortion, for which the state space can be easily extended, is considered: rotation of the target in the plane approximately perpendicular to the camera's line-of-sight. A simple solution to the tracking of the rotating objects is to include an orientation of the target in the state space, by taking $\boldsymbol{x} = (x_k, y_k, s_k, \alpha_k)^T$ as the state variable in the algorithm described in Table 2, where α_k is the orientation angle. The complexity of the algorithm is increased slightly due to the need to generate the rotated versions of the reference object (which can, possibly, be pre-computed). For some video sequences it may also be necessary to increase the number of particles, in order to sufficiently sample the state space. The results of tracking a rotating trolley in a sequence from PETS 2006 Benchmark Data (Nin (2006)), with the use of 150 particles are shown in Figure 7. The figure shows examples of frames from two best-performing tracking techniques, 'colour&edges' and 'structure'. Apart from the rotation scaling of the object, additional difficulty in tracking arose because the object was partially transparent and thus often took on the appearance of the non-stationary background. However, also in this case 'structure' tracking algorithm appears to follow the location, scale and rotation of the object more closely than the other algorithms.

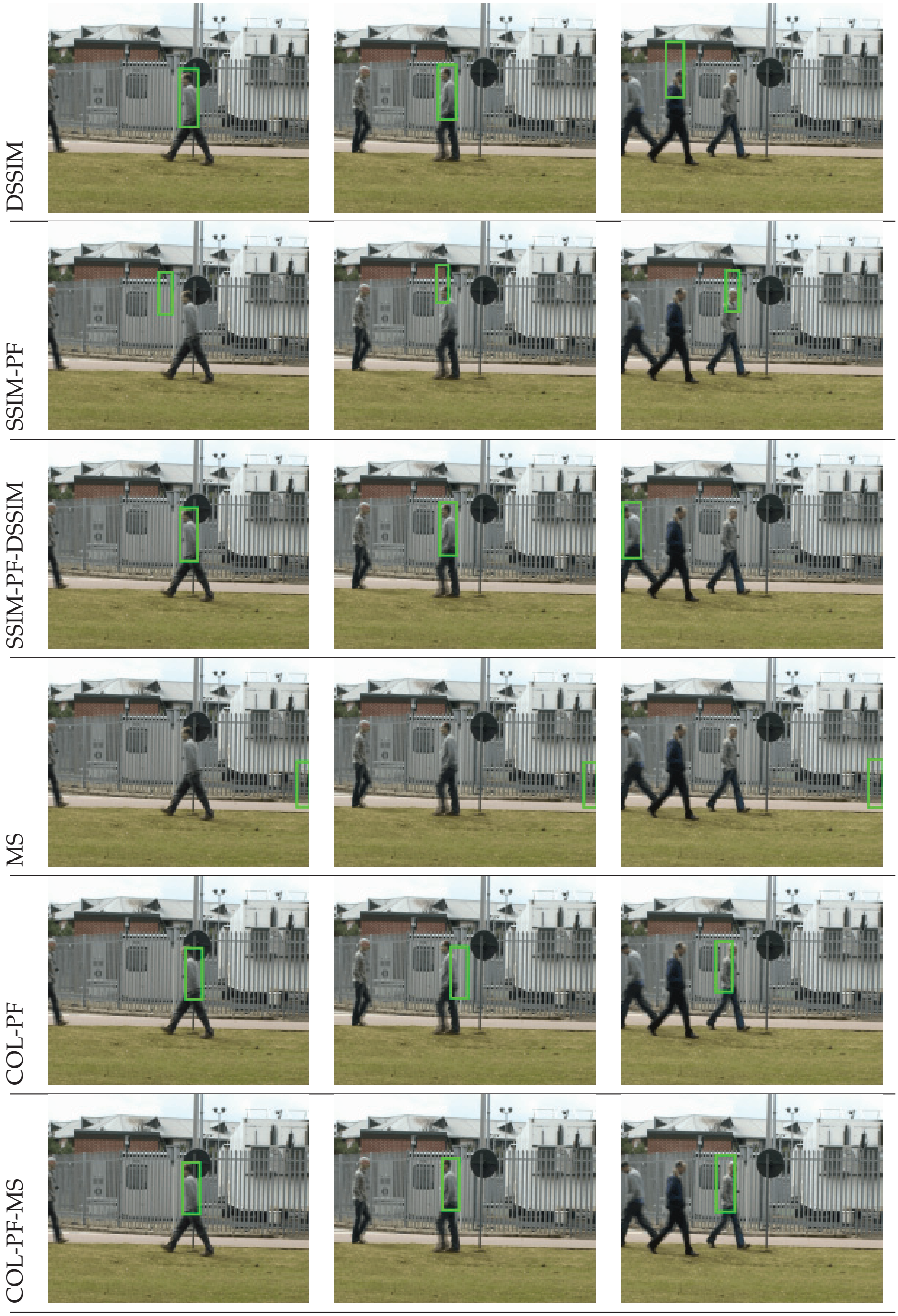


Fig. 6. Pedestrian tracking test results

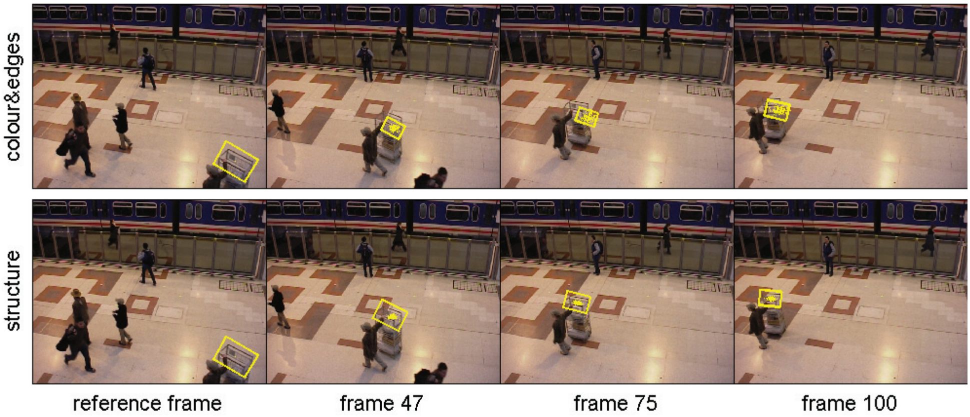


Fig. 7. Example video frames with a tracking output (tracking rectangle and particles) superimposed, sequence *S1-T1-C* containing a rotating object

5. Discussion and conclusions

The recently developed video tracking methods based on structural similarity and their new extensions have been presented in this work. Novel deterministic and hybrid probabilistic approaches to video target tracking have been investigated, and their advantages and mutual complementarities have been identified. First, a fast deterministic procedure that uses the gradient of the structural similarity surface to localise the target in a video frame has been derived. Next, a hybrid PF-based scheme, where each particle is optimised with the use of the aforementioned gradient procedure has been proposed.

The performance of the structural similarity-based methods has been contrasted with selected tracking methods based on colour and edge cues. The structural similarly methods, while being computationally less expensive, perform better, on average, than the colour, edge and mean shift, as shown in the testing surveillance video sequences. Specifically, the results obtained with the hybrid technique proposed indicate that a considerable improvement in tracking is achieved by applying the optimisation scheme, while the price of a moderate computational complexity increase of the algorithm is off-set by the low number of particles required.

The particular issue addressed herein is concerned with tracking object in the presence of spurious or similarly-coloured targets, which may interact or become temporarily occluded. All structural similarity-based method have been shown to perform reliably under difficult conditions (as often occurs in surveillance videos), when tested with real-world video sequences. Robust performance has been demonstrated in both low and variable light conditions, and in the presence of spurious or camouflaged objects. In addition, the algorithm copes well with the artefacts that may be introduced by a human operator, such as rapid changes in camera view angle and zoom. This is achieved with relatively low computational complexity, which makes these algorithms potentially applicable to real-time surveillance problems.

Among the research issues that will be the subject of further investigation is a further speed and reliability improvement of the proposed optimised hybrid technique. It is envisaged that this could be achieved by replacing the simple gradient search with a more efficient optimisation procedure and by more accurate modelling of the resulting proposal density. The structural similarity measure-based tracker, although giving very precise performance, may in some cases be sensitive to alteration of the tracked object, for example its significant rotation or long occlusion. Thus, the recovery and/or template update techniques will also be investigated in the future to improve reliability of the proposed tracker.

6. Acknowledgements

We would like to thank the support from the [European Community's] Seventh Framework Programme [FP7/2007-2013] under grant agreement No 238710 (Monte Carlo based Innovative Management and Processing for an Unrivalled Leap in Sensor Exploitation) and the EU COST action TU0702. The authors are grateful for the support offered to the project by National Natural Science Foundation of China Research Fund for International Young Scientists and EU The Science & Technology Fellowship Programme in China.

7. References

Aghajan, H. & Cavallaro, A. (2009). *Multi-Camera Networks: Principles and Applications*, Academic Press.

Aherne, F., Thacker, N. & Rockett, P. (1990). The quality of training-sample estimates of the Bhattacharyya coefficient, *IEEE Trans. on PAMI* 12(1): 92–97.

Arulampalam, M., Maskell, S., Gordon, N. & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, *IEEE Trans. on Signal Proc.* 50(2): 174–188.

Bai, K. & Liu, W. (2007). Improved object tracking with particle filter and mean shift, *Proceedings of the IEEE International Conference on Automation and Logistics, Jinan, China*, Vol. 2, pp. 221–224.

Bradski, G. (1998). Computer vision face tracking as a component of a perceptual user interface, *Workshop on Applic. of Comp. Vision*, Princeton, NJ, pp. 214–219.

Brasnett, P., Mihaylova, L., Canagarajah, N. & Bull, D. (2005). Particle filtering with multiple cues for object tracking in video sequences, *Proc. of SPIE's 17th Annual Symposium on Electronic Imaging, Science and Technology, V. 5685*, pp. 430–441.

Brasnett, P., Mihaylova, L., Canagarajah, N. & Bull, D. (2007). Sequential Monte Carlo tracking by fusing multiple cues in video sequences, *Image and Vision Computing* 25(8): 1217–1227.

Cai, Y., de Freitas, N. & Little, J. J. (2006). Robust visual tracking for multiple targets, *In Proc. of European Conference on Computer Vision, ECCV*, pp. 107–118.

Chang, C. & Ansari, R. (2003). Kernel particle filter: Iterative sampling for efficient visual tracking, *Proc. of ICIP* 2003, pp. III - 977-80, vol. 2.

Chang, C. & Ansari, R. (2005). Kernel particle filter for visual tracking, *IEEE Signal Processing Letters* 12(3): 242–245.

Chen, D. & Yang, J. (2007). Robust object tracking via online dynamic spatial bias appearance models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29: 2157–2169.

Cheng, Y. (1995). Mean shift, mode seeking and clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(8): 790–799.

Comaniciu, D. & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(5): 603–619.

Comaniciu, D., Ramesh, V. & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift, *Proc. of 1st Conf. Comp. Vision Pattern Recogn.*, Hilton Head, SC, pp. 142–149.

Comaniciu, D., Ramesh, V. & Meer, P. (2003). Kernel-based object tracking, *IEEE Trans. Pattern Analysis Machine Intelligence* 25(5): 564–575.

Cvejic, N., Nikolov, S. G., Knowles, H., Loza, A., Achim, A., Bull, D. R. & Canagarajah, C. N. (2007). The effect of pixel-level fusion on object tracking in multi-sensor surveillance video, *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Minneapolis, Minnesota, USA*.

Doucet, A., Freitas, N. & N. Gordon, E. (2001). *Sequential Monte Carlo Methods in Practice*, New York: Springer-Verlag.

Forsyth, D., Arikan, O., Ikemoto, L. & Ramanan, D. (2006). *Computational Studies of Human Motion: Part 1, Tracking and Motion Synthesis. Foundations and Trends in Computer Graphics and Vision*, Hanover, Massachusetts. Now Publishers Inc.

Fukunaga, K. & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition, *Information Theory, IEEE Transactions on* 21(1): 32 – 40.

Gandhi, T. & Trivedi, M. (2007). Pedestrian protection systems: Issues, survey and chllenges, *IEEE Transactions on Intelligent Transportation Systems* 8(3): 413–430.

Gerónimo, D., López, A. M., Sappa, A. D. & Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32: 1239–1258.

Han, B., Comaniciu, D., Zhu, Y. & Davis, L. S. (2004). Incremental density approximation and kernel-based bayesian filtering for object tracking., *CVPR (1)*, pp. 638–644.

Hu, W., Tan, T., Wang, L. & Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors, *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 34(3): 334 –352.

Isard, M. & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density, *European Conf. on Comp. Vision*, Cambridge, UK, pp. 343–356.

Isard, M. & Blake, A. (1998). Condensation – conditional density propagation for visual tracking, *Intl. Journal of Computer Vision* 28(1): 5–28.

Khan, Z., Balch, T. & Dellaert, F. (2005). MCMC-based particle filtering for tracking a variable number of interacting targets, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(11): 1805–1819.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, *J. Comput. Graph. Statist.* 5(1): 1Ű25.

Koch, W. (2010). On Bayesian tracking and data fusion: A tutorial introduction with examples, *IEEE Transactions on Aerospace and Electronics Magazine Part II: Tutorials* 7(July): 29–52.

Lewis, J. J., O'Callaghan, R. J., Nikolov, S. G., Bull, D. R. & Canagarajah, C. (2007). Pixel- and region-based image fusion using complex wavelets, 8(2): 119–130.

Liu, J. & Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems, *Journal of the American Statistical Association* 93(443): 1032–1044.

Loza, A., Mihaylova, L., Bull, D. R. & Canagarajah, C. N. (2006). Structural similarity-based object tracking in video sequences, *Proc. Intl. Conf. on Information Fusion, Florence, Italy*, pp. 1–6.

Loza, A., Mihaylova, L., Bull, D. R. & Canagarajah, C. N. (2009). Structural similarity-based object tracking in multimodality surveillance videos, *Machine Vision and Applications* 20(2): 71–83.

Lu, W.-L., Okuma, K. & Little, J. J. (2009). Tracking and recognizing actions of multiple hockey players using the boosted particle filter, *Image Vision Comput.* 27(1-2): 189–205.

Maggio, E. & Cavallaro, A. (2005). Hybrid particle filter and mean shift tracker with adaptive transition model, *Proc. of ICASSP* 2005, pp. 221-224, vol. 2.

Mihaylova, L., Loza, A., Nikolov, S. G., Lewis, J., Canga, E. F., Li, J., Bull, D. R. & Canagarajah, C. N. (2006). The influence of multi-sensor video fusion on object tracking using a particle filter, *Proc. Workshop on Multiple Sensor Data Fusion, Dresden, Germany*, pp. 354–358.

Nin (2006). PETS 2006 benchmark data, Dataset available on-line at: http://www.pets2006.net.

Nummiaro, K., Koller-Meier, E. B. & Gool, L. V. (2003). An adaptive color-based particle filter, *Image and Vision Computing* 21(1): 99–110.

Okuma, K., Taleghani, A., de Freitas, N., Little, J. & Lowe, D. (2004). A boosted particle filter: Multitarget detection and tracking, *In Proc. of European Conference on Computer Vision*, Vol. 1, pp. 28–39.

PerceptiVU, Inc. (n.d.). Target Tracking Movie Demos. http://www.perceptivu.com/MovieDemos.html.

Pérez, P., Vermaak, J. & Blake, A. (2004). Data fusion for tracking with particles, *Proceedings of the IEEE* 92(3): 495–513.

Ristic, B., Arulampalam, S. & Gordon, N. (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House, Boston, London.

Shan, C., Tan, T. & Wei, Y. (2007). Real-time hand tracking using a mean shift embedded particle filter, *Pattern Recogn.* 40(7): 1958–1970.

Shen, C., van den Hengel, A. & Dick, A. (2003). Probabilistic multiple cue integration for particle filter based tracking, *Proc. of the VIIth Digital Image Computing : Techniques and Applications*, C. Sun, H. Talbot, S. Ourselin, T. Adriansen, Eds.

Smith, D. & Singh, S. (2006). Approaches to multisensor data fusion in target tracking: A survey, *IEEE Transactions on Knowledge and Data Engineering* 18(12): 1696–1710.

The Eden Project Multi-Sensor Data Set (2006). http://www.imagefusion.org/.

Triesch, J. & von der Malsburg, C. (2001). Democratic integration: Self-organized integration of adaptive cues, *Neural Computation* 13(9): 2049–2074.

Wan, E. & van der Merwe, R. (2001). *The Unscented Kalman Filter, Ch. 7: Kalman Filtering and Neural Networks. Edited by S. Haykin*, Wiley Publishing, pp. 221–280.

Wang, Z., Bovik, A. C. & Simoncelli, E. P. (2005a). Structural approaches to image quality assessment, *in* A. Bovik (ed.), *Handbook of Image and Video Processing, 2nd Edition*, Academic Press, chapter 8.3.

Wang, Z., Bovik, A., Sheikh, H. & Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity, *IEEE Transactions on Image Processing* 13(4): 600–612.

Wang, Z. & Simoncelli, E. (2004). Stimulus synthesis for efficient evaluation and refinement of perceptual image quality metrics, *IS & T/SPIE's 16th Annual Symposium on Electronic Imaging. Human Vision and Electronic Imaging IX, Proc. SPIE, Vol. 5292*, San Jose, pp. 18–22.

Webb, A. (2003). *Statistical Pattern Recognition*, John Wiley & Sons.

Zhao, Q., Brennan, S. & Tao, H. (2007). Differential EMD tracking, *Proc. of IEEE 11th International Conference on Computer Vision*, pp. 1–8.

3

Bayesian Tracking by Online Co-Training and Sequential Evolutionary Importance Resampling

Lizuo Jin, Zhiguo Bian, Qinhan Xu and Zhengang Chen
School of Automation, Southeast University
China

1. Introduction

Object tracking is an indispensable ingredient of many machine vision applications such as intelligent visual surveillance, traffic monitoring, robot and vehicle navigation, human computer interactions, virtual and augmented realities, video compression and indexing, etc. and has drawn considerable attention from computer research communities in recent years. Actually, in the real world scenarios, that is a very challenging task due to the interference of noise, clutters, occlusions, illumination variations and dynamic changes of the object and the background appearance in the complex scene; a quite variety of tracking methods have been proposed to tackle these difficulties in decades (Yilmaz et al., 2006), which can be roughly divided into two categories: the deterministic method and the statistical methods.

The deterministic method performs tracking typically by seeking the local extreme of a matching function which measures the similarity between a template and a candidate image; the most widely used similarity measures include the sum of squared differences, the histogram intersection distance, the Kullback-Leibler divergence, the normalized cross correlation coefficient, and the Bhattacharyya coefficient. Some optimization techniques have been proposed to search the local extreme of the matching function such as the mean-shift method (Comaniciu et al., 2003) and the optical flow based method (Baker & Matthews, 2004). The drawback of these methods is if the matching function takes into account only the object and not the background, then it might not be able to correctly distinguish the object from the background and tracking might fail. More robust similarity measures are presented recently such as the posterior probability measure (Feng et al., 2008.) and the log likelihood ratio of features (Collins et al., 2005), which takes the background interference into account.

Recently, object tracking is treated as a binary classification problem, where the object have to be identified from the background with multiple image cues and better performance over the matching function based approaches such as the template-matching method (Lucas & Kanade, 1981), the view-based method (Black & Jepson, 1998), and the kernel-based method (Comaniciu et al., 2003), etc. was reported in literatures, where a discriminative model for separating the object from the background is trained offline and applied before tracking and termed tracking-by-detection method (Andriluka et al., 2008 ; Avidan 2004; Breitenstein et al., 2009 ; Choudhury et al., 2003; Leibe et al., 2008; Okuma et al., 2004; Wu & Nevatia, 2007). However, to formulate that as an object-background discrimination problem, two important factors need to be treated carefully: what features to choose and how to train the classifiers. Furthermore, since the object and background appearance may change greatly over time; online feature selection and classifier training are necessary to adapt the tracker to such variations. The appearance-based method deals with it by updating a holistic representation of object in a feature space with online learning, such as the incremental PCA (Wang et al., 2007; Ross et al., 2008), the incremental LDA (Lin et al., 2007; Li et al., 2008), the Expectation Maximization method (Jepson et al., 2003), etc. The online feature selection method chooses the most discriminative features that could distinguish the object from the background correctly rather than specifying features beforehand (Collins et al., 2005). The feature fusion method in (Yin et al., 2008) combines multiple features by weighting the likelihood maps with respect to their variance ratios. Additionally, other feature fusion methods for tracking such as the weighted averaging rule, the product rule, the maximum rule, the minimum rule and the dynamic weighting rule are compared in (Conaire et al., 2006). An ensemble of weak classifiers is trained with the offline boosting algorithm and updated online weakly in (Avidan, 2007) for identifying the pixels of object from that of background, and an improved version is proposed in (Grabner et al., 2006) with online boosting that endows the tracker the adaptability somehow. An ensemble of SVM classifiers for tracking is built in (Tian et al., 2007) by seeking the linear separating hyperplane which maximizes the margin between the positive and the negative sample in a kernel space, and that handles appearance changes somehow by heuristically updating a SVM queue online.

It is reported that the increment of the number of features used for tracking could benefit the performance of tracker (Isard & Blake, 1998; Wu & Huang, 2004); however, it depends on how the features are utilized. Online feature selection is advocated in (Collins et al., 2005; Grabner et al., 2006), but it is difficult to determine how many features should be chosen beforehand. Online feature fusion is proposed in (Yin et al., 2008) where the features are weighted and combined with respect to their variance ratios but the final combination is in fact not the most discriminative. Moreover, unlabelled data is valuable for classifier training, though they do not improve the performance always (Chapelle et al., 2006; Zhu & Goldberg, 2009). It is demonstrated recently that the performance of tracker could be significantly improved by training classifier on a labeled dataset augmented by unlabelled ones with the semi-supervised learning technologies; however, it depends on how to predict the labels on unlabeled data correctly. The self-training method updates the classifier online with its own predictions to adapt the tracker to the appearance changes in (Collins et al., 2005; Grabner et al., 2006), however incorrect predictions could deteriorate the tracker and even cause the tracking drifting, which is called tracking label jitter. Therefore, the unlabelled data need to be treated carefully when updating the classifier. The template is updated based on the geometric information heuristically in (Matthews et al., 2004) to make the tracker adaptive and avoid drifting. The semi-supervised online boosting method in (Grabner et al., 2008; Godec et al. 2009; Leistner et al. 2008) formulates the update process in a semi-supervised fashion as a combined decision of a given prior and an online classifier and can limit the drifting somehow while adaptive to appearance changes. An improved version is proposed in (Stalder et al., 2009) to adapt rapid appearance changes of the target object that may result in tracking label jitter, by optimizing a robust loss function based on a convex combination of a supervised and an unsupervised classifier. Recently online random forest is proposed for tracking in (Saffari et al. 2009; Godec et al. 2010), which outperform the online boosting based trackers in case of severe occlusions and large appearance changes. An online multi-classifier boosting algorithm is proposed in (Kim et al. 2010) for learning object multi-modal appearance models and tracking under rapid appearance changes. Tracking label jitter can also be handled by online multiple instance learning methods and in principle the classifier is still performing self-training (Babenko et al., 2009; Saffari et al., 2009). The co-training and the multi-view learning method in (Javed et al., 2005; Leistner et al., 2010; Liu et al. 2009; Tang et al., 2007; Yu et al. 2008) updates the classifiers online each other with the predictions generated on different features and can avoid tracking drifting somehow.

The statistical methods model the underlying dynamics of a tracking system in a state space. Assuming a linear Gaussian model of system dynamics, only one mode appears in the posterior probability density function (PDF); for example Kalman filter performs tracking by updating the mean and the covariance of the posterior PDF. However the dynamics of a practical tracking system is usually nonlinear and non-Gaussian; it is impossible to estimate the distribution analytically; therefore, many statistical approximation algorithms have been developed in recent years. Among them, particle filter, also called sequential Monte Carlo, is the most popular state estimation method, which constructs the posterior PDF recursively in a state space using Monte Carlo integration (Doucet et al. 2001; Cappe et al., 2007; Doucet & Johansen, 2010). It is developed for object tracking in computer vision research communities originally in (Isard & Black, 1998), also termed Condensation algorithm. During the tracking process, object state can be estimated recursively with particle filter, but over depletion of particles by sequential importance resampling (SIR) could cause tracking failure. It is very important to preserve particles diversity, which is measured usually by the effective sample size (Doucet et al. 2001). Various methods have been proposed to tackle this problem in recent years (Doucet et al. 2001; Cappe et al., 2007; Doucet & Johansen, 2010).

In this chapter we propose an adaptive object tracking method that integrates a particle filter with an online semi-supervised classifier to treat the appearance variations of the object and the background and occlusions. An online real AdaBoost algorithm is presented to train an ensemble of weak classifiers that endows the strong classifier faster convergence speed and higher classification accuracy. We further improve the classifiers by co-training operated on two groups of uncorrelated local image features in order to reduce tracking label jitter. To deal with the problem of particles depletion, an iterative importance resampling algorithm to maximize the effective sample size with evolutionary operations is proposed, which gives more accurate estimations than the SIR method, and we term it the sequential evolutionary importance resampling (SEIR) method. The final tracker combines online real AdaBoost, co-training and the SEIR particle filter all together. The experimental results on pedestrian and vehicles tracking in the real world scenarios demonstrate that our method is very robust to tracking objects undergoing large appearance changes and severe occlusions.

The remainder of this chapter is organized as follows: Section 2 introduces the local image features for representing object; Section 3 discusses the online real AdaBoost and co-training algorithm; Section 4 presents the SEIR particle filter; Section 5 gives the experimental results; and finally Section 6 concludes the chapter.

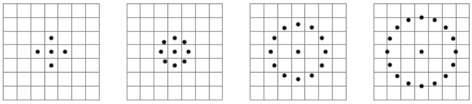
2. Local image features for object representation

In order to identify pixels of the object from that of the background in a predicted candidate region at each image frame, two types of local image features, namely color features and texture features are employed to represent the appearance of the object and the background. The color features include the most wildly used color intensity RGB and gray level intensity Y. The texture features are the recently proposed local ternary pattern (LTP) (Tan & Triggs, 2010), which is in fact an improved version of local binary patterns (LBP) that is originally applied to texture classification (Ojala et al., 2002) and later extended to face recognition

(Ahonen et al., 2006). Nowadays the LBP pattern is one of the best performing texture descriptors. It has proven to be highly discriminative, invariant to monotonic gray level changes and computational efficient, make it much suitable for demanding image analysis tasks, and it has been used in various applications. Concretely LBP pattern is based on the difference between the central and the neighbor pixels in a mask to form a representation of texture pattern by coding a binary sequence, which is defined by

$$LBP_{P,r}(i,j) = \sum_{u=0}^{P\text{-}1} 2^u \cdot S(g_u \text{ - } g_c) \tag{1}$$

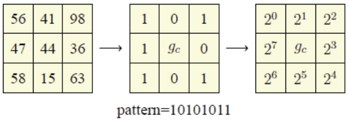
where $S(x)$ is an indicating function, $S(x)=1$ if $x \geq 0$ else $S(x)=0$; r is the radius of a mask, P is the number of neighbor pixels in the mask, g_c is the gray value of the central pixel, g_u is the gray value of the neighbor pixels, u is the label of the neighbor pixels and (i,j) is the position of the central pixel. The most widely used masks and their neighbor pixels labeled by the black dots are shown in Figure 1.



$(P=4, r=1)$ $(P=8, r=1)$ $(P=12, r=2)$ $(P=16, r=2.5)$

Fig. 1. The mask and the neighbor pixels

If set $P=8$, $r=1$ then we get the most widely used pattern LBP$_{8,\ 1}$, named 1-radius 8-neighborhood pattern. Figure 2 shows how the pattern LBP$_{8,1}$ is calculated.



$$LBP_{8,1}(g_c) = 2^0 + 2^2 + 2^4 + 2^6 + 2^7 = 213$$

Fig. 2. Calculate the texture pattern LBP$_{8,1}$

One flaw of LBP pattern is that it cannot reflect the relationship between the central and the neighbor pixels in a mask completely, for example it cannot distinguish bright spots, dark spots and other small sized patterns; furthermore, it is very sensitive to noise. To deal with it, local ternary pattern (LTP) is developed recently (Tan & Triggs, 2010), which is defined by

$$LTP_{P,r}(i,j) = \sum_{u=0}^{P-1} 3^u \cdot \mathrm{S}'(g_u - g_c, t) \tag{2}$$

where $S'(x)$ is an indicating function with a threshold t specified beforehand.
To simplify the computation, the LTP pattern is divided into two parts: the upper pattern $S^u(x)$ and the lower pattern $S^l(x)$, that are defined respectively by

$$\mathrm{S}^u(x,t) = \begin{cases} 1 & x \geq t \\ 0 & x < t \end{cases}, \quad \mathrm{S}^l(x,t) = \begin{cases} 1 & x \leq -t \\ 0 & x > -t \end{cases} \tag{3}$$

For each part, a similar operation as the calculation of LBP pattern is used to encode the pattern, and then they are combined into one unified representation. Figure 3 shows how the pattern $LTP_{8,1}$ is calculated.

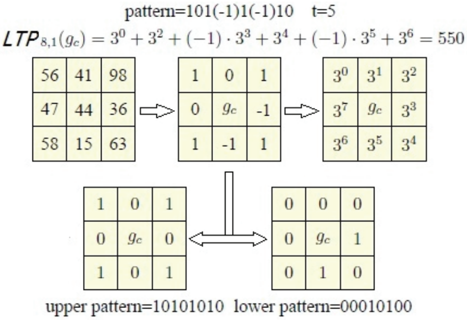

pattern=101(-1)1(-1)10 t=5
$LTP_{8,1}(g_c) = 3^0 + 3^2 + (-1) \cdot 3^3 + 3^4 + (-1) \cdot 3^5 + 3^6 = 550$
56 41 98
47 44 36
58 15 63
1 0 1
0 g_c -1
1 -1 1
3^0 3^1 3^2
3^7 g_c 3^3
3^6 3^5 3^4
1 0 1
0 g_c 0
1 0 1
0 0 0
0 g_c 1
0 1 0
upper pattern=10101010 lower pattern=00010100

Fig. 3. Calculate the texture pattern $LTP_{8,1}$

The texture is represented by nine $LTP_{8,1}$ patterns, which contains the in-channel texture feature R-R, B-B, G-G and the cross-channel texture feature R-G, R-B, G-R, G-B, B-R, and B-G. For example, when compute the R-G texture feature, around each pixel a 3x3 sized rectangle area is specified as the mask, the central pixel is taken from the red channel and the other eight neighbor pixels are taken from the green channel. Figure 4 shows the four color features and the nine texture features used in this chapter for representing the object and the background. It is obvious that the capability of each feature to discriminate the object from the background is different, and it may change greatly too when the object moves in the scene; we need to choose the best ones dynamically during tracking.

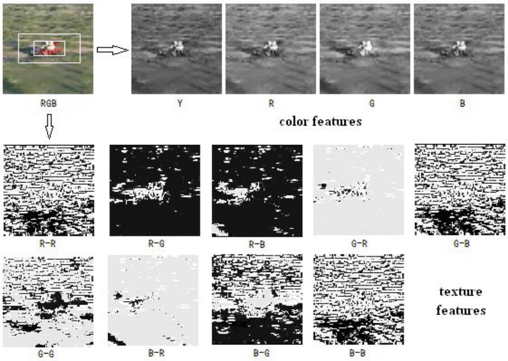


Fig. 4. The local image features

3. Locate object with online co-trained classifiers

3.1 Online real AdaBoost

To a practical tracking application, only a few training sample are available at each image frame, and the performance of the tracker depends heavily on the discriminative capability of the classifier. Boosting can improve the performance of weak learners by building a small number of weak classifiers iteratively and combining them into a strong one. AdaBoost is an adaptive boosting algorithm that combines weak classifiers together to adapt to the problem. Real AdaBoost deals with confidence-rated weak classifiers, mapping from a sample space to a real-valued prediction instead of a Boolean prediction. The real-valued weak classifiers have an advantage over the Boolean ones in discriminative capability. Although, the final classifier trained by discrete AdaBoost achieves a similar accuracy as that by real AdaBoost mostly, however, the former includes much more weak classifiers than the latter. Namely, real AdaBoost can achieve a higher accuracy if the number of weak classifiers is fixed beforehand (Schapire & Singer, 1999), exactly the case of online learning.

Unlike offline training, which uses all samples to train one weak classifier at the same time, the online version has a fixed-length classifier and uses only one sample on the entire stage. For each weak classifier on the online stage, there is a feature selector and each maintains its own information on features and updates it when a new sample is availab1e. The selector chooses the best feature with respect to the cumulated classification error; then updates the weights according to the output of the corresponding weak classifier and passes the sample to the next selector. The weights updating process boosts the online stage and enables the succeeding weak classifiers to focus on difficult sample. When the cumulated error is beyond a specified threshold, the weights updating steps are skipped. Online boosting is proposed originally in (Oza & Rusell, 2001) and later improved in (Grabner et al., 2006).

In this chapter, real AdaBoost is applied to select a group of optimal weak classifiers by minimizing the cumulated classification error on the available training sample at each image frame. In this way, the final strong classifier could adapt to appearance changes of the object and background somehow. A remarkable feature of the proposed method is that the optimal weak classifiers are chosen by real AdaBoost, which improves the classification performance significantly compared with that by discrete AdaBoost. The pseudo code of the proposed online real AdaBoost method is shown as Algorithm 1.

3.2 Online co-training

Co-training is a typical semi-supervised learning method originally introduced in (Blum & Mitchell, 1998) that allows starting with only a few labeled data to train classifiers initially and then apply more unlabeled data to improve accuracy of classification. The basic idea is that the features describing the data are redundant and could be split into different groups, each of which is sufficient for correct classification. One classifier is trained from one group of features, and the resulted two classifiers go through parallelly more unlabeled data, label them and add a part of the unlabeled ones having the most confident predictions to the labeled dataset; namely the classifier trains each other on the unlabeled data.

In this chapter we train two online ensemble classifiers parallelly by co-training, one from the group of color features and one from the group of texture features introduced in section 2, and each ensemble classifier is trained by online real AdaBoost. To predict the unlabeled data more reliably, a voting scheme for classifier combination is applied; logically, it is based on the degree of predicting agreement between the two ensemble classifiers. The final fused classifier is applied to locate the target object from each image frame. The pseudo code of the proposed online co-training method is shown as Algorithm 2.

Algorithm 1 : online real AdaBoost

```
Input:    training sample $(x,y) \in \chi \times \{-1,1\}$,
                    where $\chi$ is partitioned into several disjoint blocks $X_1, X_2, \cdots, X_k$.
          weights $\lambda_{n,m}^{corr}$, $\lambda_{n,m}^{wrong}$ (initialized with one)
Output:   strong classifier $h^{strong}$ (initialized randomly)

Initialize the importance weight $\lambda$ of the incoming sample with one.
for $n = 1,2,\cdots,N$ do // update all feature selector $h_n^{sel}$
  for $m = 1,2,\cdots,M$ do // one feature selector maintains $M$ features individually
    Initialize the weight of the sample $w_{n,m}(x,y) = w_{n,m}(x,y) + \lambda$,
            where $x \in X_j$, $j \in \{1,2,\cdots,k\}$.
  Update the weak classifier $h_{n,m}^{weak} = \frac{1}{2}\ln\left(\left(w_{n,m}(x,+1)+\varepsilon\right) / \left(w_{n,m}(x,-1)+\varepsilon\right)\right)$,
            where $\varepsilon$ is a small positive constant.
  Estimate the classification error $e_{n,m} = \lambda_{n,m}^{wrong} / \left(\lambda_{n,m}^{wrong} + \lambda_{n,m}^{corr}\right)$,
            if $h_{n,m}^{weak} \cdot y \ge 0$ then $\lambda_{n,m}^{corr} = \lambda_{n,m}^{corr} + \lambda\left|h_{n,m}^{weak}\right|$,
            else $\lambda_{n,m}^{wrong} = \lambda_{n,m}^{wrong} + \lambda\left|h_{n,m}^{weak}\right|$ end if.
  end for // feature selector
  Select the weak classifier having the lowest error $h_n^{sel} = h_{n,m^+}^{weak}$, $m^+ = \arg\min_m\left(e_{n,m}\right)$.
  if $e_{n,m^+} = 0$ or $e_{n,m^+} > 0.5$ then exit end if.
  Calculate the voting weight $\alpha_n = \ln\left(\left(1 - e_{n,m^+}\right) / e_{n,m^+}\right)$.
  Update the importance weight $\lambda$,
            if $h_{n,m}^{weak} \cdot y \ge 0$ then $\lambda = \lambda / \left(2\left(1 - e_{n,m^+}\right)\right)$,
            else $\lambda = \lambda / \left(2e_{n,m^+}\right)$ end if.
  Replace the worst weak classifier $h_{n,m^-}^{weak}$, $m^- = \arg\max_m\left(e_{n,m}\right)$ with a new one,
  set $\lambda_{n,m^-}^{corr} = 1$, $\lambda_{n,m^-}^{wrong} = 1$.
end for // update all feature selectors

The final classifier is $h^{strong} = sign\left(conf(x)\right)$, the confidence score is $conf(x) = \sum_{n=1}^{N} \alpha_n h_n^{sel}(x)$.
```

Algorithm 2 : online co-training

```
At the first image frame:
1. Locate the target object manually or automatically, and generate the labeled data.
2. Train two strong classifiers by online real AdaBoost shown in Table 1,
       One classifier from one group of features on the labeled data respectively.

At each newly coming image frame:
1. Apply the two strong classifiers pixel by pixel in a predicted candidate image region,
       generate two confidence score maps respectively.
2. Combine the two confidence score maps into one by voting:
       the label is $h^{fused} = sign\left(\min\left(conf_i(x)\right)\right)$,
       the confidence score is $conf^{fused}(x) = \max\left(\left|conf_i(x)\right|\right) \cdot h^{fused}(x)$ .
3. Locate the target object from the image frame based on the combined confidence map.
4. Add a part of the pixels having the most confident scores to the labeled dataset.
5. Retrain the two strong classifiers on the updated labeled data respectively.
```

3.3 Locate the target object

Giving the two strong classifiers, to a newly incoming image frame, we need to determine the location of the target object within it. To do this, each classifier is operated pixel by pixel in a predicted candidate region at that image frame individually, thus yields two confidence score maps from the two groups of local features; and then they are combined into one. Figure 5 illustrates how the combined confidence score map is generated. In Figure 5, (a) is the predicted candidate region containing the target object; (b) and (c) are the confidence score maps from the texture and color features respectively; (d) and (e) are the confidence score maps of (b) and (c) after applying the morphological operation OPENING respectively; (f) is the final combined confidence score map. It is obvious that each feature contributes the map differently; it has fewer clutters than either one before combining, which means it can reduce tracking label jitter and may lead to more stable tracking.

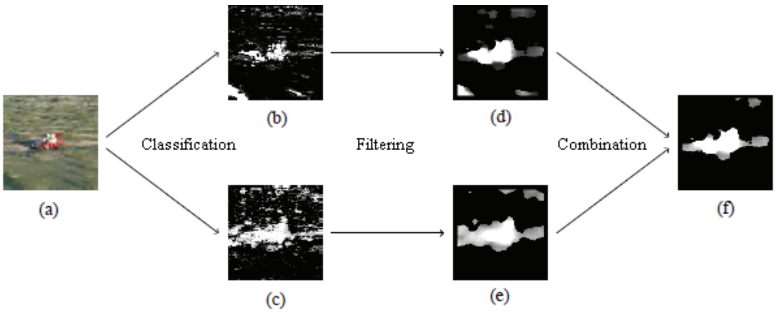


Fig. 5. Combine the confidence score maps into one

The location of the target object can be conjured from the combined confidence score map. Since local features are utilized, the confidence score can somehow measure the contribution of the pixel to the existence of the target object; the largest blob having highest accumulated confidence score locates the object most confidently. We check all the blobs within the predicted candidate region and choose the most confident blob as the target object and the center of the ellipse bounding the blob most closely is treated as object location. The whole locating procedure is visualized in figure 6, where, (a) is the predicted candidate region containing the target object; (b) is the confidence score map from the color features; (c) is the confidence score map from the texture features; (d) is the combined confidence score map; (e) is the result of (d) after thresholding; (f) shows the ellipse that bounding the blob most closely, of which the center indicates the object location.

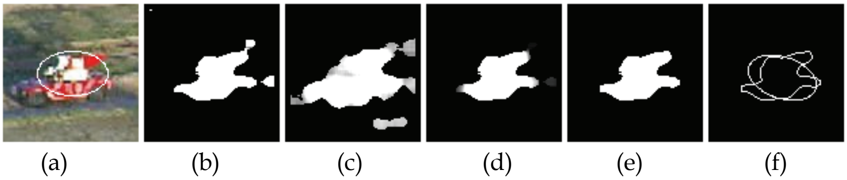


Fig. 6. Locate the target object from the combined confidence score map

Stable tracking depends on the discriminative ability of the strong classifiers trained by co-training, in fact which is heavily affected by the correctness of the sequentially predicted sample added to the labeled dataset. We must check the accuracy of classification carefully; however it is difficult to be evaluated online due to lacking ground-truth data. Fortunately, by checking the amount of object pixels within and background pixels outside of the object blob, we can guess that somewhat, though it is ad hoc. Usually the most confident blob is treated as the object; when the accumulated confidence score of pixels within it beyond a specified threshold, the pixels are reckoned as positive sample, and added to the labeled dataset, otherwise no one is selected; that happens mostly as occlusions appear. Meanwhile, the pixels near the boundary of the candidate region are treated as negative sample and added to the labeled dataset at any time to adapt the tracker to the varying background.

4. Predict object location with particle filter

4.1 Sequential importance resampling

To a practical object tracking system, we can describe the dynamics of the target object by

$$X_k = X_{k-1} + \delta_k \tag{4}$$

where X_k and X_{k-1} are the location of the target object at time k and $k-1$ respectively, δ_k is a random variable which subjects to the transition distribution $p\left(X_k \middle| X_{k-1}\right)$.

The posterior PDF is approximated recursively by a weighted sample, involving two steps mainly: prediction and update. Given the observations $Y_{1:k\text{-}1} = \{Y_1, \dots, Y_{k\text{-}1}\}$ up to time $k-1$, at the prediction step, the transition distribution $p\left(X_k \middle| X_{k-1}\right)$ is applied to predict the posterior PDF at time k, termed the prior as well,

$$p\left(X_k \middle| Y_{1:k-1}\right) = \int p(X_k \mid X_{k-1}) p\left(X_{k-1} \mid Y_{1:k-1}\right) dX_{k-1} \tag{5}$$

At time k as the observation Y_k is available, applying the Bayes rule, the posterior PDF is

$$p\left(X_k \middle| Y_{1:k}\right) = \frac{p(Y_k \mid X_k)p\left(X_k \mid Y_{1:k-1}\right)}{p(Y_k \mid Y_{1:k-1})} \tag{6}$$

The posterior PDF is approximated by N sample X_k^i drawn from the proposal distribution $q\left(X_k \mid X_{1:k\text{-}1}, Y_{1:k}\right)$ with the importance weight v_k^i,

$$v_k^i = v_{k-1}^i \frac{p(Y_k \mid X_k^i)p\left(X_k^i \mid X_{k-1}^i\right)}{q(X_k \mid X_{1:k-1}, Y_{1:k})} \tag{7}$$

In the case of sequential importance resampling (SIR), $q\left(X_k \mid X_{1:k\text{-}1}, Y_{1:k}\right) = p\left(X_k \mid X_{1:k\text{-}1}\right)$, the importance weight becomes the observation likelihood $p\left(Y_k \middle| X_k\right)$.

To the SIR particle filter, if it fails to generate new values for the states from the latest observations, only a few particles will have significant importance weights, the variance of weights will increase continuously and eventually cause tracking failure, which is termed particle degeneration. So it is very important to move particles towards the regions of high likelihood. This problem arises when the likelihood is too peaked or lies in the tail of the prior. Various approaches have been proposed to tackle this problem such as the auxiliary particle filter (APF), the unscented particle filter (UPF), the kernel particle filter (KPF), the regularized particle filter (RPF), to name a few (Doucet et al. 2001; Cappe et al., 2007).

4.2 Sequential evolutionary importance resampling

To keep the diversity of particles, we enhance importance resampling in this chapter with four evolutionary operations: copy, crossover, mutation and selection.

1. COPY: generate N new particles X_k^{l1} by duplicating all the existing N particles,

$$X_k^{l1} = X_k^i \tag{8}$$

2. CROSSOVER: generate N new particles X_k^{l2} based on N pair of particles X_k^i and X_k^j chosen randomly from the existing N particles according to the probability $p\left(Y_k \middle| X_k^i\right)$ and $1 - p\left(Y_k \middle| X_k^j\right)$,

$$X_k^{l2} = X_k^j + \mu \cdot (X_k^i - X_k^j) \tag{9}$$

where μ is a random number which subjects the standard uniform distribution.

3. MUTATION: generate N new particles X_k^{l3} by disturbing the existing N particles,

$$X_k^{l3} = X_k^i + \lambda \cdot \nu \tag{10}$$

where ν is a random number which subjects the standard normal distribution, and λ is a constant which controls the amplitude of interference.

4. SELECTION: resample N particles from the resulted $3N$ new particles $\left\{X_k^{l1}, X_k^{l2}, X_k^{l3}\right\}$ according to their importance weights.

In fact, the operation CROSSOVER and MUTATION can increase the amount of distinctive particles, the operation COPY and SELECTION can keep the particles staying in the regions of high likelihood. The combination of the four operations can make particles move towards the regions of high likelihood so as to overcome the degeneration of particles.

The effective sample size (ESS) is a function of the coefficient of the variation of importance weights, measuring the efficiency of an importance sampling algorithm, which is defined by

$$N_{eff} = 1 \Big/ \sum_{i=1}^{n} \left(\tilde{v}_k^i \right)^2 \tag{11}$$

where $\tilde{v}_k^i$ is the normalized importance weight, which is defined by

$$\tilde{v}_k^i = v_k^i \Big/ \sum_{i=1}^{n} \left(v_k^i \right) \tag{12}$$

When ESS is small, it means there is a risk of degeneration; importance resampling should be conducted to augment the particles by applying COPY, CROSSOVER, MUTATION and SELECTION until ESS becomes large enough. The pseudo code of the proposed sequential evolutionary importance resampling method is detailed as Algorithm 3.

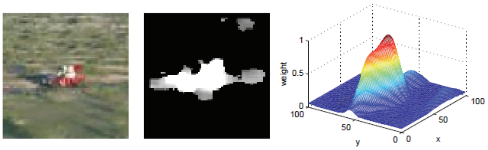
Algorithm 3: sequential evolutionary importance resampling

```
1: Initialization, $k = 1$
   Sample $X_1^i \sim p(X_1), i = 1, \cdots, N$ .
   Evaluate the importance weight $v_1^i = p\left(Y_1 \mid X_1^i\right)$ .
   Resample $\left\{X_1^i, v_1^i\right\}$ to generate the $N$ equally weighted particles $\left\{X_1^i, \frac{1}{N}\right\}$ .

2: Importance sampling, $k > 1$
   Sample $\tilde{X}_k^i \sim p\left(X_k \mid X_{k-1}^i\right), i = 1, \cdots, N$ .
   Evaluate the importance weight $v_k^i = p\left(Y_k \mid \tilde{X}_k^i\right)$ .
   Normalize the importance weight to obtain $\tilde{v}_k^i = v_k^i \Big/ \sum_{i=1}^{n} \left(v_k^i\right)$ .

3: Evolutionary importance resampling
   Compute the effective sample size $N_{eff}$ , and set the loop counter $T = 0$ .
   While ( $N_{eff}$ or $T$ is not large enough)
      Generate $N$ new particles $X_k^{l1}$ by COPY, $l1 = 1, \cdots, N$ ,
         evaluate the importance weight $v_k^{l1} = p\left(Y_k \mid X_k^{l1}\right)$ .
      Generate $N$ new particles $X_k^{l2}$ by CROSSOVER, $l2 = 1, \cdots, N$ ,
         evaluate the importance weight $v_k^{l2} = p\left(Y_k \mid X_k^{l2}\right)$ .
      Generate $N$ new particles $X_k^{l3}$ by MUTATION, $l3 = 1, \cdots, N$ ,
         evaluate the importance weight $v_k^{l3} = p\left(Y_k \mid X_k^{l3}\right)$ .
      Normalize the importance weight of the $3N$ particles $\left\{X_k^{l1}, X_k^{l2}, X_k^{l3}\right\}$ .
      Resample $N$ particles $X_k^i$ by SELECTION from the $3N$ particles $\left\{X_k^{l1}, X_k^{l2}, X_k^{l3}\right\}$ .
      Normalize the importance weight of the selected $N$ particles to obtain $\tilde{v}_k^i$ .
      Compute the effective sample size $N_{eff}$ and set $T = T + 1$ .
   end while
   Resample $\left\{X_k^i, \tilde{v}_k^i\right\}$ to generate the $N$ equally weighted particles $\left\{X_k^i, \frac{1}{N}\right\}$ .
   Set $k = k + 1$ and go to step 2.
```

4.3 Predict the location of the target object

When performing tracking, a particle is a candidate object state, actually corresponding to a possible location of the target object; of which the importance weight is proportional to the observation likelihood, that in fact indicates the possibility whether the object appears there; Theoretically, the likelihood must be proportional to the accumulated confidence scores of pixels classification around the position; practically, it is computed by adding the confidence scores together in a specified window at that position in this chapter.

Figure 7 exemplifies the distribution of accumulated confidence scores in a predicted region, where, (a) is the predicted image region containing the target object; (b) is the combined confidence score map yield by the co-trained classifier; (c) is the distribution of accumulated confidence scores. It is obvious in (c) that there is one mode holding a very high likelihood value, which may result in an accurate prediction on object location.



(a) (b) (c)

Fig. 7. The distribution of accumulated confidence scores

To track an object in complex scene, actually, pixels classification error is inevitable, means clutters must appear at the predicted object region. To reduce the interference of clutters, we use a gating function to penalize pixels by weighting; concretely, pixels near the predicted position get a higher weight, otherwise gets a lower weight. In this chapter, we choose a Gaussian $g(X;\mu,\sigma)$ as the gating function, let the distribution of accumulated confidence scores is $l(X)$, then the observation likelihood is defined by

$$p(Y|X)=l(X)\cdot g(X;\mu,\sigma) \tag{13}$$

5. Experimental results

This section presents two tracking examples on practical video sequences that illustrate the benefits of combining online feature selection and co-training techniques. Specifically, these benefits are the enhanced ability to track objects undergoing large appearance variations, the ability to adapt to the changing background and illumination conditions, the ability to treat severe occlusions, and the ability to avoid distraction by similar objects by emphasizing automatically certain features that are distinctive to the target object.

5.1 Results on pixels classification

Figure 8 shows the comparison result on pixels classification by the co-trained and the self-trained classifier, where, (a) is the predicted candidate image region containing the target object; (b) and (c) are the confidence score map respectively from the texture and the color features yield by the co-trained classifier; (d) is the combined confidence score map from (b) and (c); (e) is the confidence score map from the mixed texture and color features yield by the self-trained classifier; (f) is the result of (e) after applying the morphological operation OPENING. It is obvious that the score map yield by the co-trained classifier has much fewer clutters than the one by the self-trained classifier, which may lead to more stable tracking.

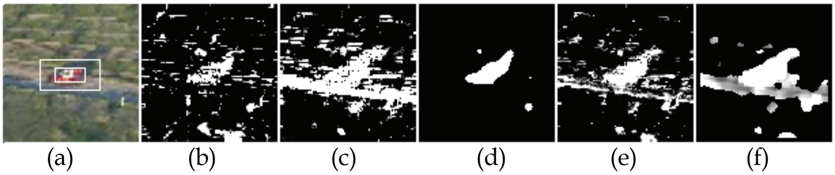


Fig. 8. Pixels classification by the co-trained and the self-trained classifier

5.2 Results on tracking objects in complex scene

The proposed method is applied to track vehicles and pedestrians in complex scene. The test video data consists of a vehicle sequence from PETS 2005 dataset (Collins et al., 2005) and a pedestrian sequence captured by us. Especially, the vehicle sequence involves significant appearance variations resulting from the change of object size, environmental lighting and distractions by similar objects in the scene. Besides, in the pedestrian sequence, the object is occasionally occluded severely by background clutters for several times.

Figure 9 (a) and (b) shows the results on tracking a vehicle and a pedestrian respectively, where the red box indicates the location of the target object yield by the proposed tracker. The experimental results demonstrate that the proposed method can enhance the ability to effectively track objects undergoing large appearance variations due to size change, and to adapt the background and illumination change, severe occlusions, and even the distractions by similar objects.

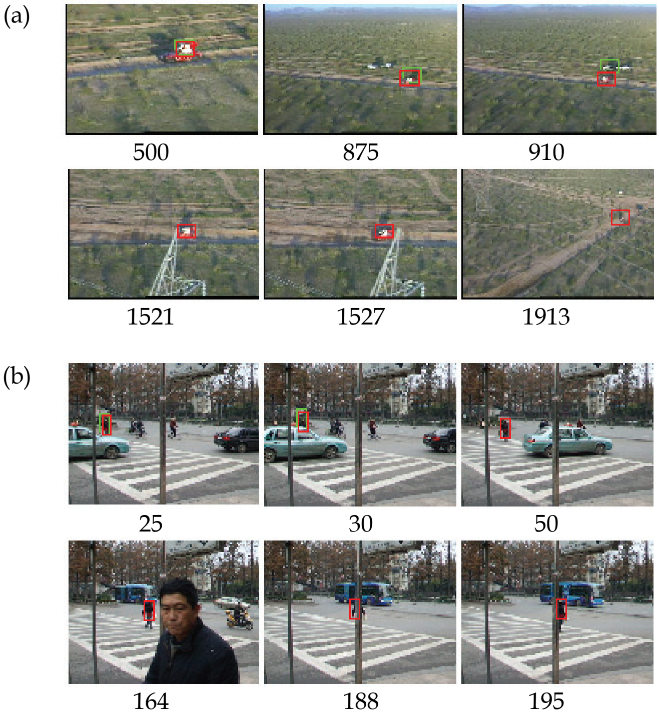


Fig. 9. Results on tracking vehicle and pedestrian

To check why it can do these, the results on pixels classification yield by the co-trained classifier in the predicted candidate regions on the selected frames from the two sequences are visualized in figure 10 and 11 respectively, where, (a) is the predicted candidate image region, (b) is the combined confidence score map. As can be seen, the combined map in (b) has very few clutters; based on that the target object can be located very accurately, in fact, the white bounding ellipse in (a) predict the object location correctly.

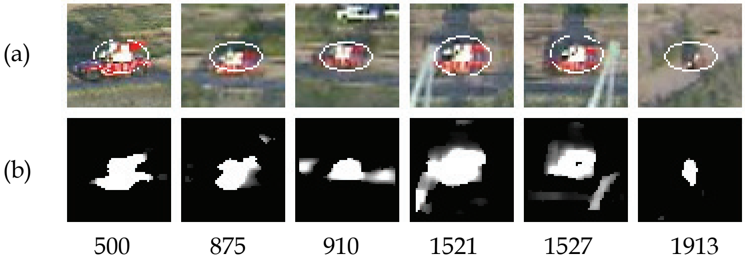


Fig. 10. Pixels classification by the co-trained classifier on the vehicle sequence

5.3 Results on importance resampling

Figure 12 shows the comparison results on vehicle tracking with state estimation by particle filter using different importance resampling method, where, the object locating results are shown in (a); the effective sample size and maximum loop number yield by the sequential evolutionary importance resampling (SEIR) particle filter are shown respectively in (b) and (c). The results demonstrate that the SEIR particle filter yields more accurate state estimation than the SIR particle filter. It is obvious in (a) that the red box (SEIR) encloses the object more closely than the green box (SIR) and the latter loses tracking at frame 1600, which may due to the inaccurate state estimation. The figure in (b) shows that the SEIR particle filter (red line) can maintain high efficiency of importance resampling while the SIR particle filter (green line) is much lower, where in total 300 particles are used. Furthermore within finite times, only about three loops averagely, the expected efficiency can be reached by the SEIR particle filter, means it is computationally very efficient.

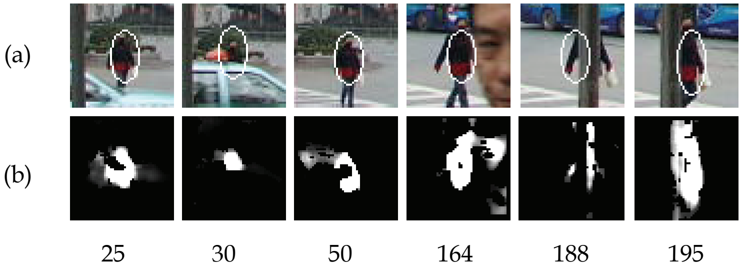


Fig. 11. Pixels classification by the co-trained classifier on the pedestrian sequence

Figure 13 shows the variation on scatter of particles when conducting the evolutionary importance resampling at some frame. In figure (a) to (d), the effective sample size is 62, 186,

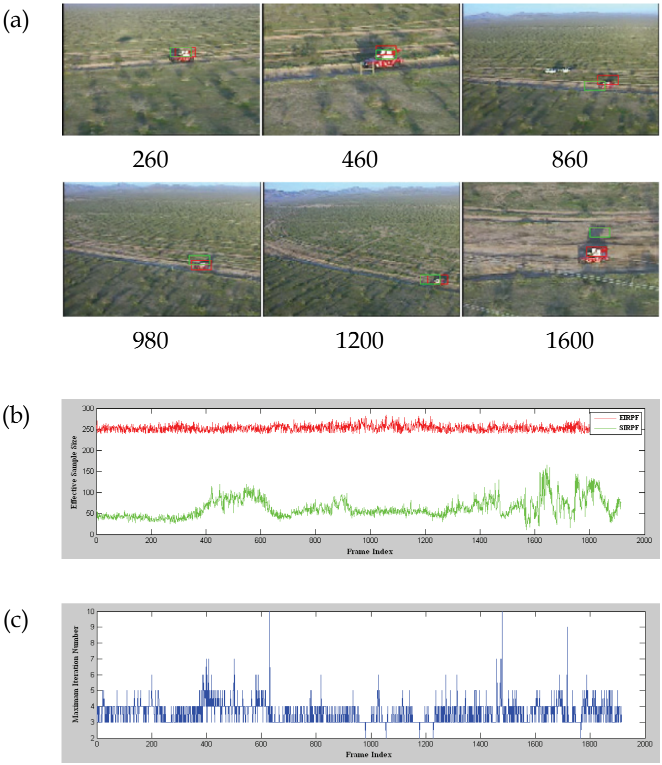


Fig. 12. Results on vehicle tracking with state estimation by particle filter

236 and 259 respectively. It is obvious that the particles move quickly towards the center of the target object and cover it more closely, meanwhile the effective sample size increases accordingly; and that may lead to more accurate state estimation than the SIR particle filter. Figure 14 shows the scatter of particles after importance resampling at some frame, where, (a) is the predicted candidate image region containing the target object; (b) is the combined confidence score map yield by the co-trained classifier; (c) shows the distribution of particles after SEIR; (d) shows the distribution of particles after SIR. It is obvious that though there are some clutters in (b), yield by inaccurate pixels classification, through conducting SEIR, particles move towards the true location of the target object, which is demonstrated by the number and the spread of the mode appearing in the distributions of particles. There are fewer modes in (c) than in (d) and the spread of mode is much lower in (c) than that in (d).

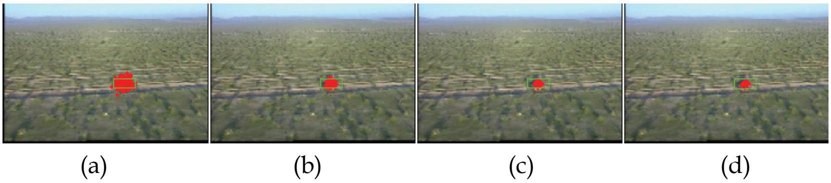


Fig. 13. Scatter of particles when conducting importance resampling

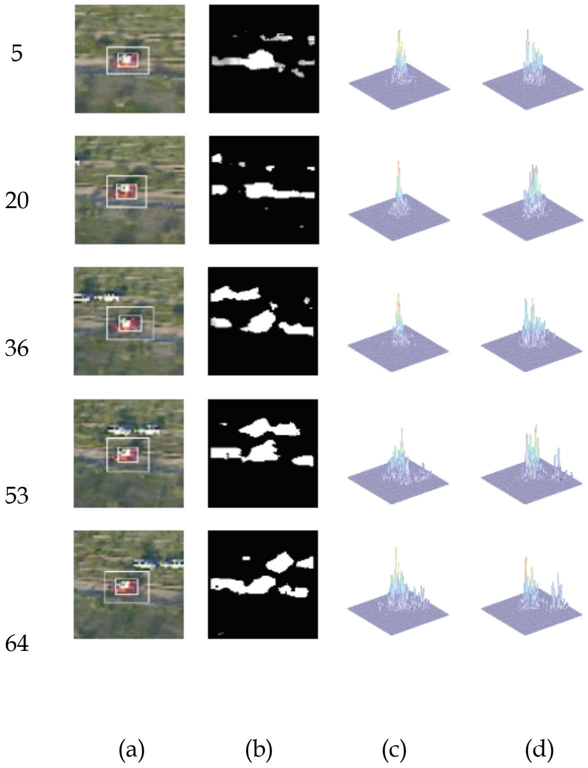


Fig. 14. Scatter of particles after importance resampling

6. Conclusions

Object tracking is still a challenging and open task in computer vision research, especially when dealing with large appearance variations and severe occlusions. In this chapter, an adaptive object tracking method is proposed, which integrates online semi-supervised classification and particle filter efficiently. Thanks to the feature selection capability achieved by online real AdaBoost and co-training, two classifiers are trained on two groups of features complementarily; the tracker consistently provides accurate classification of the object and the background for stable tracking, even under severe situations.

In fact the efficiency of the proposed method comes from three aspects. First of all, the real AdaBoost approach constructs ensemble classifiers with a soft decision scheme to adapt to aspects, occlusions and size variations of the object by online learning. Second, classifier co-training on two uncorrelated groups of features further improves the classification accuracy on the object from the background. Finally, the particle filter with sequential evolutionary importance resampling can adapt the nonlinearity of object dynamics and integrates the confidence scores updated online to predict object state more accurately even when it is occluded by background clutters completely.

7. References

Ahonen, T.; Hadid, A. & Pietikainen, M. (2006). Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 12, (Dec. 2006) pp. 2037-2041.

Andriluka, M.; Roth, S. & Schiele, B. (2008). People tracking by detection and people detection by tracking, *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, Anchorage, Alaska, USA, Jun. 2008.

Avidan, S.; Support vector tracking (2004). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 8, (Aug. 2004) pp. 1064-1072.

Avidan, S.; Ensemble Tracking (2007). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 2, (Feb. 2007) pp. 261-271.

Babenko, B.; Yang, M-H & Belongie, S. (2009). Visual tracking with online multiple instance learning, *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, Miami, FL, USA, Jun. 2009.

Baker, S. & Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, Vol. 56, No.3, (Mar. 2004) pp.221-255.

Black, M.J. & Jepson, A.D. (1998). EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision*, Vol. 26, No.1, (Jan. 1998) pp. 63-84.

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training, *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pp. 92-100, Madison, Wisconsin, USA, Jul. 1998.

Breitenstein, M.D.; Reichlin, F., Leibe, B., Koller-Meier, E. & Gool, L.V. (2009). Robust tracking-by-detection using a detector confidence particle filter, *Proceedings of the 12th IEEE International Conference on Computer Vision*, pp. 1515-1522, Kyoto, Japan, Sep. 2009.

Cappe, O.; Godsill, J.S. & Moulines, E. (2007). An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceeding of IEEE*, Vol. 95, No. 5, (May 2007) pp. 899-924.

Chapelle, O.; Scholkopf, B. & Zien, A. (2006). *Semi-Supervised Learning*. The MIT Press, ISBN: 978-0-262-03358-9, London, England.

Choudhury, R.; Schmid, C. & Mikolajczyk, K. (2003). Face detection and tracking in a video by propagating detection probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 10, (Oct. 2003) pp. 1215-1228.

Collins, R.T.; Liu, Y. & Leordeanu, M. (2005). Online Selection of Discriminative Tracking Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 10, (Oct. 2003) pp. 1631-1643.

Collins, R.T.; Zhou, X. & Teh, S.K. (2005). An Open Source Tracking Testbed and Evaluation Web Site, *Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2005)*, pp. 1-8, Breckenridge, USA, Jan. 2005.

Comaniciu, D.; Ramesh, V. & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, (May 2003) pp. 234-240.

Conaire, C.O.; O'Connor, N.E., Cooke E. & Smeaton, A.F. (2006). Comparison of Fusion Methods for Thermo-Visual Surveillance Tracking, *Proceedings of the 9th IEEE International Conference on Information Fusion*, pp.1-7, Florence, Italy, Jul. 2006.

Doucet, A.; Freitas, N.D. & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer, ISBN: 978-0-387-95146-1.

Doucet, A. & Johansen, A. M. (2010). A Tutorial on Particle Filtering and Smoothing: Fifteen years later, in *Oxford Handbook of Nonlinear Filtering*, D. Crisan, B. Rozovsky (Ed.), Oxford University Press, London UK.

Feng, Z.; Lu, N. & Jiang, P. (2008). Posterior probability measure for image matching. *Pattern Recognition*, Vol. 41, No. 7, (Jul. 2008) pp. 2422-2433.

Godec, M., Grabner, H., Leistner, C. & H. Bischof. (2009). Speeding up Semi-Supervised On-line Boosting for Tracking, *Proceedings of the 33rd annual Workshop of the Austrian Association for Pattern Recognition*, pp. 1-8, Stainz, Austria, May 2009.

Godec, M., Leistner, C., Saffari, A. & Bischof, H. (2010). On-line Random Naive Bayes for Tracking, *Proceedings of the IEEE 20th International Conference on Pattern Recognition*, pp. 1-4, Istanbul, Turkey, Aug. 2010.

Grabner, H.; Grabner, M. & Bischof, H. (2006). Real-time tracking via on-line boosting, *Proceedings of the 17th British Machine Vision Conference*, pp. 47-56, Edinburgh UK, Sep. 2006.

Grabner, H.; Leistner, C. & Bischof, H. (2008). Semi-supervised On-Line Boosting for Robust Tracking, *Proceedings of the 10th European Conference on Computer Vision*, pp. 234-247, Marseille, France, Oct. 2008.

Hieu, N. & Arnold, S. (2006). Robust Tracking Using Foreground-Background Texture Discrimination. *International Journal of Computer Vision*, Vol. 69, No. 3, (Jun. 2006) pp. 277-293.

Isard, M.; & Blake, A. (1998). CONDENSATION: Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, Vol. 29, No.1, (Jan. 1998) pp. 5-28.

Isard, M. & Blake, A. (1998). ICONDENSATION: Unifying Low-Level and High-Level Tracking in a Stochastic Framework, *Proceedings of the 6th European Conference on Computer Vision*, pp.893-908, Freiburg, Germany, Jun. 1998.

Javed, O.; Ali, S. & Shah, M. (2005). Online Detection and Classification of Moving Objects Using Progressively Improving Detectors, *Proceedings of the IEEE 2005 Conference on Computer Vision and Pattern Recognition*, pp. 1-8, San Diego, Jun. 2005.

Jepson, A.; Fleet, D. & El-Maraghi, T. (2003). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 10, (Oct. 2003) pp.1296-1311.

Kim, T-K; Woodley, T., Stenger, B. & Cipolla, R. (2010). Online Multiple Classifier Boosting for Object Tracking, Proceedings of the 2010 IEEE Conference on computer Vision and Pattern Recognition, pp. 1-6, San Francisco, CA, USA, Jun. 2010.

Leibe, B.; Schindler, K., Cornelis, N. & Gool, L.V. (2008). Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 10, (Oct. 2008) pp. 1683-1698.

Leistner, C., Godec, M., Saffari, A. & Bischof, H. (2010). On-line Multi-View Forests for Tracking, *Proceedings of the 32nd annual symposium of The German Association for Pattern Recognition*, pp.1-10, Darmstadt, Germany, Sep. 2010.

Leistner, C.; Grabner, H. & Bischof, H. (2008). Semi-Supervised Boosting using Visual Similarity Learning, *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, Anchorage, Alaska, USA, Jun. 2008.

Li, G.; Liang, D., Huang, Q., Jiang, S. & Gao, W. (2008). Object tracking using incremental 2D-LDA learning and Bayes inference, *Proceedings of the IEEE 15th International Conference on Image Processing*, pp. 1568-1571, San Diego, California, USA, Oct. 2008.

Lin, R-S; Yang, M-H & Levinson, S.E. (2004). Object Tracking Using Incremental Fisher Discriminant Analysis, *Proceedings of the IEEE 17th International Conference on Pattern Recognition*, pp. 757-760, Cambridge, UK, Aug. 2004.

Liu, R.; Cheng, J. & Lu, H. (2009). A robust boosting tracker with minimum error bound in a co-training framework, *Proceedings of the 12th IEEE International Conference on Computer Vision*, pp. 1459-1466, Kyoto, Japan, Sep. 2009.

Lucas B. & Kanade T. (1981). An iterative image registration technique with an application to stereo vision, *Proceedings of the 7th international joint conference on Artificial intelligence*, pp.674-679, Vancouver, British Columbia, Aug. 1981.

Matthews, I.; Ishikawa, T. & Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No.6, (Jun. 2004) pp. 810-815.

Mallapragada, P.K.; Jin, R., Jain, A.K. & Liu, Y. (2009). SemiBoost: Boosting for Semi-Supervised Learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 11, (Nov. 2009) pp. 2000-2014.

Ojala, T.; Pietikainen, M. & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 7, (Jul. 2002) pp. 971-987.

Okuma, K.; Taleghani, A., Freitas, N.D., Little, J., & Lowe, D. (2004). A boosted particle filter: Multitarget detection and tracking, *Proceedings of the 8th European Conference on* Computer Vision, pp. 28-39, Prague, Czech, May 2004.

Oza, N. & Rusell, S. (2001). Online bagging and boosting, *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, pp.105-112, Key West, FL, USA, Jan. 2001.

Ross, D.A.; Lim, J., Lin, R-S & Yang, M-H (2008). Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision*, Vol. 77, No. 3, (Mar. 2008) pp. 125-141.

Saffari, A.; Leistner, C., Santner, J., Godec, M. & Bischof, H. (2009). On-line random forests, *Proceedings of the 12th IEEE International Conference on Computer Vision*, pp. 1-8, Kyoto, Japan, Sep. 2009.

Schapire, R.E. & Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, Vol. 37, No. 3, (Dec. 1999) pp. 297-336.

Stalder, S.; Grabner, H. & Gool, L.V. (2009). Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but not Simpler than Recognition, *Proceedings of the 12th IEEE International Conference on Computer Vision*, pp. 1409-1416, Kyoto, Japan, Sep. 2009.

Tan, X. & Triggs, B. (2010). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, Vol. 19, No. 6, (Jun. 2010) pp. 1635-1650.

Tang F.; Brennan S., Zhao Q. & Tao H. (2007). Co-tracking using semi-supervised support vector machines, *Proceedings of the 11th IEEE International Conference on Computer Vision*, pp. 1-8, Rio de Janeiro, Brazil, Oct. 2007.

Tian, M.; Zhang, W. & Liu, F. (2007). On-Line Ensemble SVM for Robust Object Tracking, *Proceedings of the 8th Asian Conference on Computer Vision*, pp. 355-364, Tokyo Japan, Nov. 2007.

Wang, T.; Gu, I.Y.H. & Shi, P. (2007). Object Tracking Using Incremental 2D-PCA Learning and ML Estimation, *Proceedings of the IEEE 32nd International Conference on Acoustics, Speech, and Signal Processing*, pp. 933-936, Honolulu, Hawaii, USA, Apr. 2007.

Wu, B. & Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, Vol. 75, No. 2, (Feb. 2007) pp. 247-266.

Wu, Y. & Huang, T. S. (2004). Robust Visual Tracking by Integrating Multiple Cues Based on Co-Inference Learning. *International Journal of Computer Vision*, Vol. 58, No. 1, (Jan. 2004) pp. 55-71.

Yilmaz, A.; Javed, O. & Shah, M. (2006). Object Tracking: A Survey, *ACM Computing Surveys,* Vol. 38, No. 4, (Dec. 2006) pp. 1-45.

Yin, Z.; Porikli, F. & Collins, R.T. (2008). Likelihood Map Fusion for Visual Object Tracking, *Proceedings of the IEEE 2008 Workshop on Application of Computer Vision*, pp. 1-7, Copper Mountain, USA, Jan. 2008.

Yu, Q.; Dinh, T.B. & Medioni, G. (2008). Online Tracking and Reacquisition Using Co-trained Generative and Discriminative Trackers, *Proceedings of the 10th European Conference on Computer Vision*, pp. 678-691, Marseille, France, Oct. 2008.

Zhu, X. & Goldberg, A.B. (2009). *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, ISBN: 978-1-598-29548-1.

4

Object Tracking Based on Color Information Employing Particle Filter Algorithm

Budi Sugandi, Hyoungseop Kim, Joo Kooi Tan and Seiji Ishikawa
Graduate School of Engineering, Kyushu Institute of Technology
Japan

1. Introduction

The increasing interest in the object tracking is motivated by a huge number of promising applications that can now be tackled in real-time applications. These applications include performance analysis, surveillance, video-indexing, smart interfaces, teleconferencing and video compression and so on. However, object tracking can be extremely complex and time-consuming especially when it is done in outdoor environments. Here, we can mention some problems of object tracking in outdoor environments such as fake-motion background, illumination changes, shadows and presence of clutter. A variety of tracking algorithms have been proposed and implemented to overcome these difficulties. They can be roughly classified into two categories: deterministic methods and stochastic methods.

Deterministic methods typically track the object by performing an iterative search for a similarity between the template image and the current one. The algorithms which utilize the deterministic method are background subtraction (Heikkila & Silven, 1999; Stauffer & Grimson, 1999; McIvor, 2000; Liu et al., 2001), inter-frame difference (Lipton et al., 1998; Collins et al., 2000), optical flow (Meyer et al., 1998), skin color extraction (Cho et al., 2001; Phung et al., 2003) and so on. On the other hand, the stochastic methods use the state space to model the underlying dynamics of the tracking system such as Kalman filter (Broida & Chellappa, 1986; Arulampalam et al., 2002) and particle filter (Isard & Black, 1998; Kitagawa, 1996; Gordon et al., 1993; Ristic et al., 2004). Kalman filter is a common approach for dealing with target tracking in the probabilistic framework. In a linear-Gaussian model with linear measurement, there is always only one mode in the posterior probability density function (pdf). The Kalman filter can be used to propagate and update the mean and covariance of the distribution of this model (Arulampalam et al., 2002). But it cannot resolve the tracking problem when the model is nonlinear and non-Gaussian (Tanizaki, 1987). For nonlinear or non-Gaussian problems, it is impossible to evaluate the distributions analytically and many algorithms have been proposed to approximate them. The extended Kalman filter can deal to this problem, but still has a problem when the nonlinearity and non-Gaussian cannot be approximated accurately. To overcome those problems, particle filter has been introduced by many researchers and become popular algorithm to estimate the problem of nonlinear and non-Gaussian estimation framework. The particle filter, also known as sequential Monte Carlo (Kitagawa, 1996), is the most popular approach which recursively constructs the posterior pdf of the state space using Monte Carlo integration. It approximates a posterior probability density of the state such as the object position by using samples or particles. The probability distribution of the state of the tracked object is approximated by a set of particles, which each state is denoted as the hypothetical state of the tracked object and its weight. It has been developed in the computer vision community and applied to tracking problem and is also known as the Condensation algorithm (Isard & Black, 1998). For another particle filter, Bayesian bootstrap filter was introduced (Gordon et al., 1993).

Although particle filters have been widely used in recent years, they have important drawbacks (King & Forsyth, 2000) such as samples are spread around several modes pointing out the different hypotheses in the state space. Many improvements have been introduced, but there is still much ground to cover. Different approaches have been taken in order to overcome these problems. (Nummiaro et al., 2002) used a particle filter based on color histograms features. Histograms are robust to partial occlusions and rotations but no shape analysis is taken into account. Moreover, no multiple-target tracking is considered and complete occlusions are not handled. (Perez et al., 2002) proposed also a particle filter based on color histogram. They introduced interesting extension in multiple-part modeling, incorporation of background information and multiple targets tracking. Nevertheless, it requires an extremely large number of samples and no appearance model updating is performed, what usually leads to target loss in dynamic scenes. (Comaniciu et al., 2000) proposed to use mean shift in order to track non-rigid object. His work has real time capabilities. However, the problem for object tracking with color occurs when the region around the object is cluttered and illumination is change. In this way, a color feature based tracking does not provide reliable performance because it fails to fully model the target especially when occlusion occurs. In another work, (Comaniciu et al., 2003) approach relied on gradient-based optimization and color-based histograms. In this case, no dynamic model is used therefore no occlusion can be predicted. (Deutscher & Reid, 2005) presented an interesting approach called annealing particle filter which aims to reduce the required number of samples. However, it could be inappropriate in a cluttered environment. They combine edge and intensity measures but they focused on motion analysis, and thus, no occlusion handling is explored.

To overcome the above problems, in this article, we made some improvements on color based object tracking and employed it to track a moving object. We proposed an object state not only object position but also speed, size, object size scale and appearance condition of the object. We applied also target model update condition and adaptive likelihood, ensuring the proper tracking object. By applying the appropriate initialization of the sample, we successfully track the moving object both known and unknown initial position of the object. We utilized the appearance condition and variance of the samples position for the tracking purpose without any other object detection methods. The color histogram is utilized as the observation model which is measured using Bhattacharyya distance as a similarity measurement between target model and the samples.

The outline of this article is as follows. In section 2, we describe briefly a particle filter algorithm. In section 3, our approach employing color particle filter is described in detail. We make an extension of our design method for robustness of the tracking method. In section 4, we show some experimental results in order to demonstrate the effectiveness of our method. Finally, section 5 presents the conclusions of the article.

2. Particle filter

2.1 Bayesian filtering framework

In this sub section, we briefly discuss the Bayesian approach for state estimation (Simon, 2006). Suppose we have a nonlinear system described by equations,

$$x_{k+1} = f_k(x_k, \omega_k)$$

$$y_k = h_k(x_k, v_k) \tag{1}$$

where, k is the time index, x_k is the state, ω_k is the process noise, y_k is the measurement, and v_k is the measurement noise, respectively. The function $f_k(.)$ and $h_k(.)$ are time-varying non-linear system and measurement equation. The noise sequences $\{\omega_k\}$ and$\{v_k\}$ are assumed to be independent and white with known pdf's. The goal of a Bayesian estimator is to approximate the conditional pdf of $\boldsymbol{x}_k$ based on measurements $y_1, y_2, \ldots, y_k$. This conditional pdf is denoted as $p(\boldsymbol{x}_k \mid \boldsymbol{y}_k)$. The first measurement is obtained at $k = 1$, so the initial condition of the estimator is the pdf of x_0, which can be written as $p(\mathrm{x}_0) = p(x_0 \mid y_0)$ since y_0 is defined as the set of no measurements. Once we compute $p(\boldsymbol{x}_k \mid \boldsymbol{y}_k)$ then we can estimate $\boldsymbol{x}_k$.

The Bayesian estimator will find a recursive way to compute the conditional pdf $p(\boldsymbol{x}_k \mid \boldsymbol{y}_k)$. Before we find this conditional pdf, we will find the conditional pdf $p(\boldsymbol{x}_k \mid y_{1:k-1})$. This pdf can be calculated as

$$p(x_k \mid y_{1:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid y_{1:k-1}) dx_{k-1} \tag{2}$$

The second pdf on the right side of the above equation is not available yet, but it is available at the initial time ($p(x_0)$ = $\mathrm{p}(x_0 \mid y_0)$). The first pdf on the right side of the above equation is available. The pdf $p(x_k \mid x_{k-1})$ is simply the pdf of the state at time k given a specific state at time (k - 1). We know this pdf because we know the system equation $f_k(.)$ and we know the pdf of the noise ω_k. This pdf can be calculated according to dynamical model of the system $f_k(.)$

Next, we consider a *posteriori* conditional pdf of x_k. We can write this pdf as

$$p(x_k \mid y_{1:k}) = \frac{p(y_k \mid x_k) p(x_k \mid y_{1:k-1})}{p(y_k \mid y_{1:k-1})} \tag{3}$$

All of the pdf's on the right side of the above equation are available. The pdf $p(y_k \mid x_k)$ is available from our knowledge of the measurement equation $h_k(.)$ and our knowledge of the pdf of the measurement noise *vk*. The pdf $p(x_k \mid y_{1:k-1})$ is available from Eq. (2). Finally, the pdf $\mathrm{p}(y_k \mid y_{1:k-1})$ is obtained as follows:

$$p(y_k \mid y_{1:k-1}) = \int p(y_k \mid x_k) p(x_k \mid y_{1:k-1}) dx_k \tag{4}$$

Both of the pdf's on the right side of the above equation are available. The pdf $p(y_k \mid x_k)$ is available from our knowledge of the measurement equation *h(.)* and the pdf of *vk*, and $p(x_k \mid y_{1:k-1})$ is available from Eq. (2).

Summarizing the development of this section, the recursive equations of the Bayesian state estimation filter can be summarized as follows.

1. The system and measurement equations are given as follows:

$$x_{k+1} = f_k(x_k, \omega_k)$$

$$y_k = h_k(x_k, v_k)$$

where $\{\omega_k\}$ and $\{v_k\}$ are independent white noise processes with known pdf's.

2. Assuming that the pdf of the initial state $p(x_0)$ is known, initialize the estimator as follows:

$$p(x_0 \mid y_0) = p(x_0)$$

3. For $k = 1, 2, \ldots$, perform the following.
   a. The a *priori* pdf is obtained from Eq. (2).

$$p(x_k \mid y_{1:k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid y_{1:k-1}) dx_{k-1}$$

   b. The a *posteriori* pdf is obtained from Eq. (3) and (4).

$$p(x_k \mid y_{1:k}) = \frac{p(y_k \mid x_k) p(x_k \mid y_{1:k-1})}{\int p(y_k \mid x_k) p(x_k \mid y_{1:k-1}) dx_k}$$

Analytical solutions to these equations are available only for a few special cases. In particular, if $f(.)$ and $h(.)$ are linear, and x_0, $\{\omega_k\}$, and $\{v_k\}$ are additive, independent, and Gaussian, then the solution is the Kalman filter , otherwise it can be solved by particle filter.

2.2 Particle filter for object tracking

The particle filter was invented to numerically implement the Bayesian estimator which recursively approximates the posterior distribution using a finite set of weighted samples or particles. It has been introduced by many researchers to solve the estimation problem when the system is nonlinear and non-Gaussian. The basic idea behind the particle filter is Monte Carlo simulation, in which the posterior density is approximated by a set of particles with associated weights. As a Bayesian estimator, particle filter has two main steps: prediction and update. Prediction is done by propagating the samples based on the system model. The update step is done by measuring the weight of each samples based on the observation model. The implementation of particle filter can be described as follows.

1. Particle initialization.

Starting with a weighted set of samples at k-1 $\{\mathrm{x}_{k-1}^i, \pi_{k-1}^i; i = 1:N\}$ approximately distributed according to $p(\boldsymbol{x}_{k-1} \mid y_{1:k-1})$ as initial distribution $p(x_0)$, new samples are generated from a suitable proposal distribution, which may depend on the previous state and the new measurements.

2. Prediction step.

Using the probabilistic system transition model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, the particles are predicted at time k. It is done by propagating each particle based on the transition or system model.

$$x_{k+1} = f_k(x_k, \omega_k) = p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k\text{-}1})$$

3. Update step.

To maintain a consistent sample, the new importance weights are set to

$$\pi_k^i = \pi_{k-1}^i \frac{p\left(\mathrm{y}_k \middle| \mathrm{x}_k^i\right) p\left(\mathrm{x}_k^i \middle| \mathrm{x}_{k-1}^i\right)}{q\left(\mathrm{x}_k \middle| \mathrm{x}_{1:k-1}, y_{1:k}\right)} \tag{5}$$

It is done by measuring the likelihood of each sample based on the observation model.

4. Resample.

This step is performed to generate a new samples set according to their weight for the next iteration. The resample step will decrease the number of the samples with low weight and will increase the number of high weight sample. The new particles set is re-sampled using normalized weights $\bar{\pi}_k^i$ as probabilities. This sample set represents the posterior at time k, $p(\mathrm{x}_k|y_{1:k})$.

5. Then, the expectations can be approximated as

$$\mathrm{E}p(\mathrm{x}_k|\mathrm{y}_{1:k}) \cong \sum_{i=1}^{N} \bar{\pi}_k^i \mathrm{x}_k^i \tag{6}$$

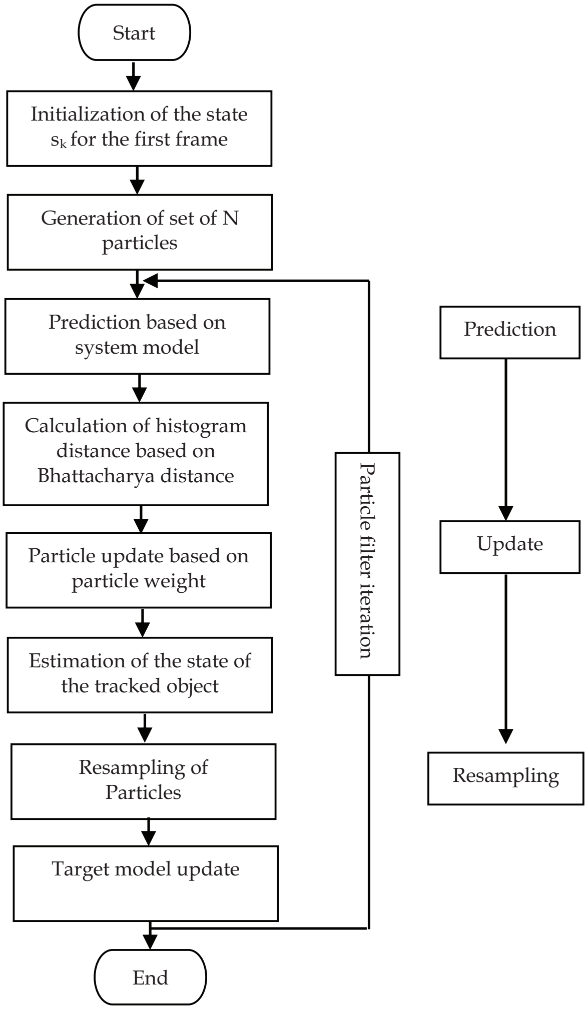

Start
Initialization of the state s_k for the first frame
Generation of set of N particles
Prediction based on system model
Calculation of histogram distance based on Bhattacharya distance
Particle update based on particle weight
Estimation of the state of the tracked object
Resampling of Particles
Target model update
End
Particle filter iteration
Prediction
Update
Resampling

Fig. 1. Entired flow of procedure

3. An approach to robust tracking

In this article, we proposed an algorithm based on particle filter in conjunction with a color feature. The motion of the central point of a bounding box is modeled using first-order dynamics model. In this article, the state of the object is defined as $s_k = (x_k, \dot{x}_k, w_k, \dot{w}_k, A_k)$ where the components are position, speed, bounding-box size, bounding-box scale and pixel appearance, respectively. The observations $\boldsymbol{y}_k$ is given by input images I_k.

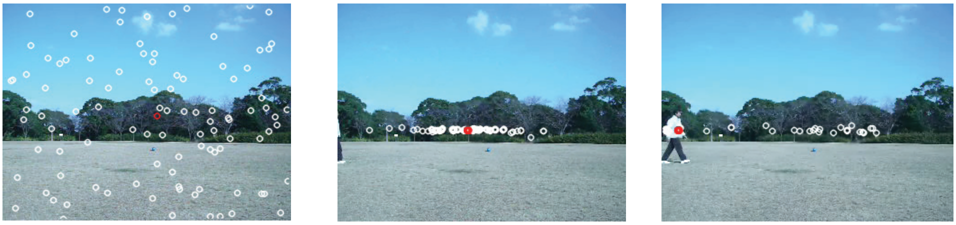
3.1 System flow diagram

In this article, we implement a particle filter in a color model-based framework to track the moving objects in outdoor environment. Firstly, the initialization of the samples done in the first frame is performed by drawing them randomly on entire scene or drawing them based on region where the object is expected to appear.

Next, the samples are predicted based on a system/transitional model by propagating each sample based on this model. The samples update is performed based on the observation model. In this article, we use color distribution of the object as the observation model. Then using the Bhattacharya distance, the similarity between the color distribution of the target and the samples can be measured. Based on the Bhattacharya distance, the weight of each sample is measured. The target sate estimation is performed based on the sample's weight. The resampling is performed for the next sample iteration to generate a new samples set. During the resample step, samples with a high weight may be chosen several times leading to identical copies, while others with relatively low weights may be ignored and deleted. And finally, the target model update is performed adaptively based on the best match of the target model. The overall flow diagram is shown in Fig. 1.

3.2 Initialization

In this article, the initialization of the particles is done on two approaches: (i) samples are initialized by putting the samples randomly on entire scene and (ii) by putting the samples around the region where the target is expected to appear such as edge of the scene or edge of the occluded object.



#1 #5 #10

Fig. 2. Initialization based on the variance of the samples

On the first approach (Fig. 2), the samples are initialized randomly on entire scene (frame #1). In this approach, we utilize the variance of the samples position to be a parameter of object tracking. We determine that the object is begun to track when the variance of the samples is below then certain threshold. Fig. 3 shows the variance of the samples position for each frame index. From that figure, we can understand that the object has not been

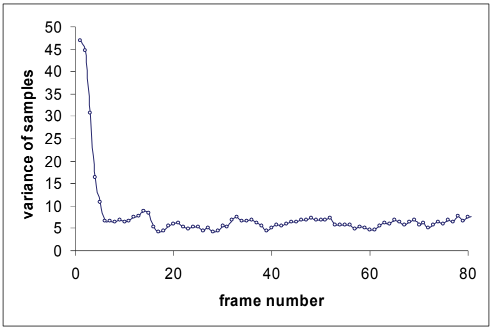
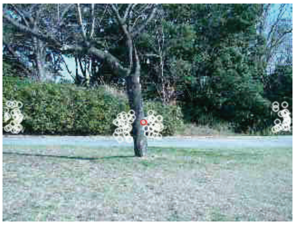
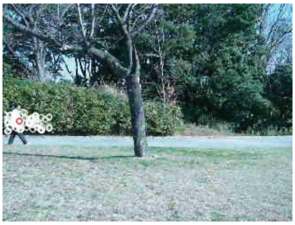


Fig. 3. Variance of the samples

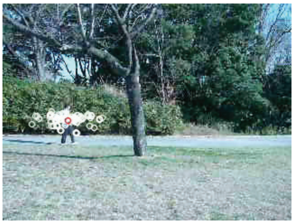
tracked yet before the variance of the sample less than the threshold, although the object appeared on the scene. We can see from Fig. 2 that the object is begun to be tracked in frame #10 when the variance of the sample position is less than threshold.



#1



#9



#25

Fig. 4. Initialization based on expected region

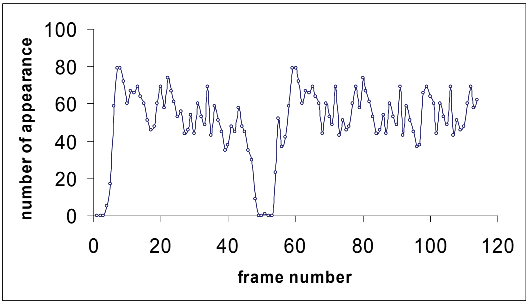


Fig. 5. Appearance of the samples

On the other hand on the second approach, the object tracking is begun when the samples satisfy some special appearance conditions. In this condition, the samples are put around the region where the target is expected to appear such as edge of the scene or edge of the occluded object. In this approach, if the target appears, the Bhattacharyya distances (later we will discuss it) of samples around the object position should be remarkable smaller than the average of sample set. Therefore, the mean value μ_b and the standard deviation σ_b of the Bhattacharyya distances of all initial samples are firstly calculated as,

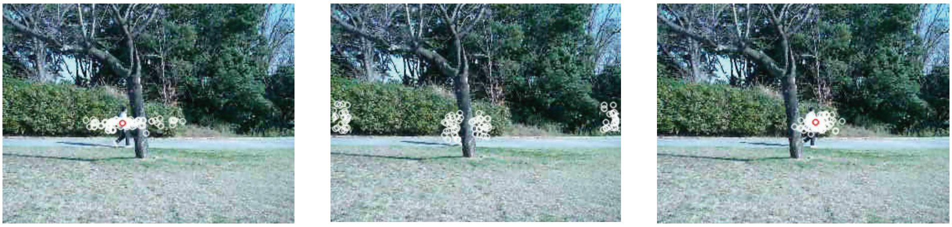
$$\mu_b = \frac{1}{I}\sum_{i=1}^{I}\sqrt{1-\rho[p_{x_i},q]} \tag{7}$$

$$\sigma_b{}^2 = \frac{1}{I}\sum_{i=1}^{I}\left(\sqrt{1-\rho[p_{x_i},q]}-\mu\right)^2 \tag{8}$$

and then an appearance condition is defined as,

$$d = \sqrt{1-\rho[p_{x_i},q]} > \mu + 2\sigma \tag{9}$$

indicating a 95[%] confidence that a sample belongs to the object. Here, $\rho[p_{xi}\ ,q]$ is the Bhattacharya distance between the target model and the object. Then, an appearing and disappearing threshold T is defined to indicate the quantity of the samples fulfilling the appearance condition during initialization. More than T means the target being found and starting tracking, contrary situation means the tracker will enter the initialization mode. Fig. 4 shows the initialization approach (frame #1). The object tracking is begun when the appearance condition is fulfilled (frame #9). The appearance of the samples is shown in Fig. 5. As presented in that figure, the object is detected and started to be tracked when the number of samples appearance more than threshold T. From that figure also, we can understand that object disappear behind the tree. It is shown by the number of samples appearance become zero. Thus, using the appearance condition, we not only can detect the object but also can predict the occlusion of the object. The predicted occlusion is illustrated in Fig. 6. The occlusion is predicted occurs on frame #50.



#45 #50 #60

Fig. 6. Occlusion prediction based on appearance condition

3.3 Transition/system model

In this article, we consider the motion of the object as the discrete time 2-dimensional (2D) motion with constant velocity assumption. The state vector at a time step k is denoted by $\mathbf{s}_k$, including position, speed, size and bounding box scale of each sample. For this purpose, we model the system based on the following expression.

$$\begin{aligned}
\hat{\mathrm{x}}_k &= \mathrm{x}_{k-1} + \dot{\mathrm{x}}_{k-1}\Delta t + \xi_{\mathrm{x}}\,, \\
\hat{\dot{\mathrm{x}}}_k &= \dot{\mathrm{x}}_{k-1} + \xi_{\dot{\mathrm{x}}}\,, \\
\hat{\mathrm{w}}_k &= \mathrm{w}_{k-1} + \dot{\mathrm{w}}_{k-i}\Delta t + \xi_{\mathrm{w}}\,, \qquad (10) \\
\hat{\dot{\mathrm{w}}}_k &= \dot{\mathrm{w}}_{k-1} + \xi_{\dot{\mathrm{w}}}
\end{aligned}$$

Here, $\hat{\mathrm{x}}_k, \hat{\dot{\mathrm{x}}}_k, \dot{w}_k, \hat{\dot{w}}_k$ are the sample position, speed, bounding box and bounding box scale of the state, respectively. The random vectors $\xi_{\mathrm{x}}, \xi_{\dot{\mathrm{x}}}, \xi_{\mathrm{w}}, \xi_{\dot{\mathrm{w}}}$ are Gaussian noise providing the system with a diversity of hypotheses. Then, each component of the samples is predicted by propagating the samples according to this transition model.

3.4 Observation model

The observation model is used to measure the observation likelihood of the samples. Many observation models have been built for particle filtering tracking. One of them is a contour based appearance template (Isard & Black, 1998). The tracker based on a contour template gives an accurate description of the targets but performs poorly in clutter and is generally time-consuming. The initialization of the system is relatively difficult and tedious. In contrast, color-based trackers are faster and more robust, where the color histogram is typically used to model the targets to combat the partial occlusion, and non-rigidity (Nummiaro et al., 2002; Perez et al., 2002).

In this article, the observation model is made based on color information of the target obtained by building the color histogram in the RGB color space. This section describes how the color features is modeled in a rectangular region *R*, where *R* can be a region surrounding the object to be tracked or region surrounding one of the hypothetical regions. A color histogram is commonly used for object tracking because they are robust to partial occlusion, rotation and scale invariant. They are also flexible in the types of object that they can be used to track, including rigid and non-rigid object.

The color distribution is expressed by an *m*-bins histogram, whose components are normalized so that its sum of all bins equals one. For a region *R* in an image, given a set of *n* samples in *R*, denoted by $\mathbf{X} = \{x_i, i = 1, 2, \ldots, n\} \in R$, the *m*-bins color histogram $H(R) = \{h_j\}$, $(j = 1, 2, \ldots, m)$ can be obtained by assigning each pixel x_i to a bin, by the following equation:

$$h_j = \frac{1}{n} \sum_{(x_i) \in \mathrm{X}} \delta_j[b(x_i)] \tag{11}$$

Here $b(x_i)$ is the bin index where the color component at x_i falls into, and δ is the Kronecker delta function.

To increase the reliability of the target model, smaller weight are assigned to the pixels that are further away from region center by employing a weighting function

$$g(r) = \begin{cases} 1 - r^2 & r < 1 \\ 0 & otherwise \end{cases} \tag{12}$$

Here, *r* is the distance from the center of the region.

Using this weight, the color histogram $p_{\mathrm{y}} = \left\{ p_{\mathrm{y}}^{(u)} \right\}$ $u = 1, \ldots, m$ at location ***y*** is calculated as

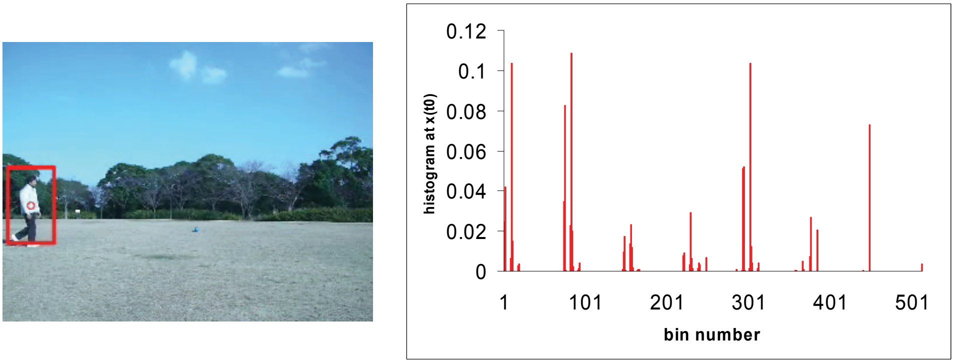
$$p_{\mathrm{y}}^{(u)} = f \sum_{j=1}^{I} g\left(\frac{\left\| \mathrm{y} - \mathrm{x}_j \right\|}{a} \right) \delta\left[h\left(\mathrm{x}_j \right) - u \right] \tag{13}$$

where, m is number of bins, I is the number of pixels in the region R, x_j is the position of pixels in the region R, δ is the Kronecker delta function, a is the normalization factor, f is the scaling factor to ensures that $\sum_{u=1}^{m} p_{\mathrm{y}}^{(u)} = 1$, and $g(.)$ is weighting function, respectively.
Fig. 7 shows an example of target histogram at time step k. In subsequent frames, at every time k, there are N particles that represent N hypothetical states need to be evaluated. The observation likelihood model is used to assign a weight associated to a specific particle (new observation) depending on how similar the model histogram q and the histogram $p(x_t)$ of object in the region described by the i^{th} particle x_k^i are.

The similarity between two color histograms can be calculated using Bhattacharya distance $d = \sqrt{1 - \rho[p,q]}$, where $\rho[p,q] = \sum_{u=1}^{m} \sqrt{p^{(u)} q^{(u)}}$. Similar histogram will have a small Bhattacharya distance which corresponds to high sample weight. Based on the Bhattacharya distance, the weight $\pi^{(i)}$ of the sample state $x^{(i)}$ is calculated as,

$$\begin{aligned}\pi^{(i)} &= \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{d^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{\left(1-\rho\left[p\left(\mathrm{x}^{(i)}\right),q\right]\right)}{2\sigma^2}\right)\end{aligned} \tag{14}$$

Where $p(x^{(i)})$ and q are the color histogram of the samples and target model, respectively. During the resample step, samples with a high weight may be chosen several times leading to identical copies, while others with relatively low weights may be ignored.



(a) target object at time k (b) histogram of a target

Fig. 7. An example of color histogram distribution of a target model at time k

3.5 State estimation

Each of the target state is estimated according their mean estimation as following equation,

$$\mathrm{x}_k = (1-\alpha_{\mathrm{x}})(\mathrm{x}_{k-1} + \dot{\mathrm{x}}_{k-1}\Delta t) + \alpha_{\mathrm{x}}\left(\sum_{i=1}^{N} \bar{\pi}_k^i \hat{\mathrm{x}}_k^i\right)$$

$$\mathrm{w}_k = (1-\alpha_{\mathrm{w}})\mathrm{w}_{k-1} + \alpha_{\mathrm{w}}\left(\sum_{i=1}^{N} \bar{\pi}_k^i \hat{w}_k^i\right)$$

$$\dot{\mathrm{x}}_k = (1-\alpha_{\dot{\mathrm{x}}})\dot{\mathrm{x}}_{k-1} + \alpha_{\dot{\mathrm{x}}}\left(\frac{\mathrm{x}_k - \mathrm{x}_{k-1}}{\Delta t}\right) \tag{15}$$

$$\dot{\mathrm{w}}_k = (1-\alpha_{\dot{\mathrm{w}}})\dot{\mathrm{w}}_{k-1} + \alpha_{\dot{\mathrm{w}}}\left(\frac{\mathrm{w}_k - w_{k-1}}{\Delta t}\right)$$

Here, $\alpha_{\mathrm{x}}, \alpha_{\dot{\mathrm{x}}}, \alpha_{\mathrm{w}}, \alpha_{\dot{\mathrm{w}}} \in [0,1]$ denote the adaptation rates.

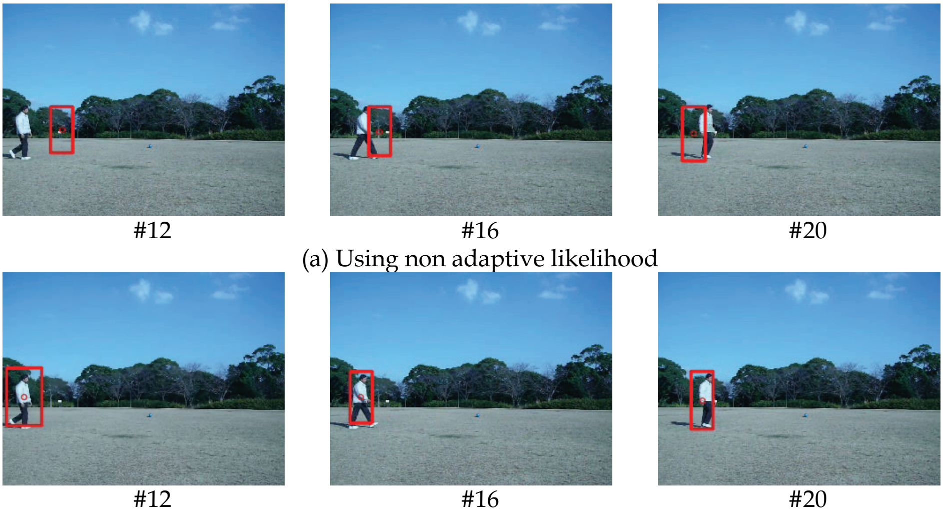
3.6 Resample step

The resample step is performed for the next sample iteration to generate a new samples set. During the resample step, samples with a high weight may be chosen several times leading to identical copies, while others with relatively low weights may be ignored and deleted. The resample step can be done in several different ways. One straightforward way is as the following steps [Ristic et al., 2004].

1. Generate a random number *r* that is uniformly distributed on [0, 1].
2. Calculate the normalized cumulative probability

$$c_k^{(0)} = 0, \quad c_k^{(i)} = c_k^{(i-1)} + w_k^{(i)}, \quad c_k^{\prime(i)} = \frac{c_k^{(i)}}{c_k^{(N)}} \tag{16}$$

3. By binary search, find the smallest *j* for which $c_k^{\prime} \geq r$ and set the new particle $x_k^i = x_k^j$



(a) Using non adaptive likelihood

(b) Using adaptive likelihood

Fig. 8. Object tracking using adaptive and non adaptive likelihood

3.7 Adaptive likelihood

The observation likelihood function is very important for the tracking performance of particle filter. The reason is that the observation likelihood function determines the weights of the particles and determines how the particles are resampled. Resample step is used to decrease the number of low-weighted particles and increase the number of particles with high weights. The observation likelihood function is expressed in Eq. (14). In this case, the observation likelihood function is the only function that contributes to particle's weight. We can rewrite the equation as,

$$\pi^{(i)} \propto \exp\left(-\frac{d^2}{2\sigma^2}\right) \tag{17}$$

here, the σ parameter determines the steepness of the function, i.e. how fast the function will decrease when the distance is large (bad particles).

It is reported that the value of the parameter σ has a major influence on the properties of the likelihood (Freeman & Adelson, 1992). Typically, the choice of this value is left as a design parameter to be determined experimentally. However, in this article, an adaptive scheme is proposed which aims to maximize the information available in the likelihood using the Bhattacharya distance *d*. For this purpose, we define the minimum squared distance $d_{\min}^2$ as the minimum distance d^2 of the set of distance calculated for all particles. Rearranging the likelihood function yields,

$$\log(\pi) \propto -\frac{d^2}{2\sigma^2} \tag{18}$$

from which we can obtain σ that gives maximum likelihood as,

$$\sigma \propto \frac{\sqrt{2}}{2} d_{\min}^2 \tag{19}$$

Fig. 8 shows a comparison the object tracking method using the adaptive and non-adaptive likelihood model. Using non adaptive likelihood, the particles with different Bhattacharya distance will be assigned with equal weights. As a result, the likelihood function will not differentiate the difference between particle with small distance and particle with big distance. This condition may assign the bad particles with high weight (Fig. 8 (a)). On the other hand, the adaptive likelihood ensures the best particles are assigned with the highest weights and the worst ones are assigned with small weight (Fig. 8 (b)).

3.8 Target model update

The problem for object tracking with color occurs when the region around the object is cluttered, illumination is change and the object appearance is change. In this way, a color feature based tracking does not provide reliable performance because it fails to fully model the target. To overcome this problem, the adaptive target model is applied based on the best match of the color model. However, this is a sensitive task. The target models are only updated when two conditions hold: (i) the target is not occluded and (ii) the likelihood of the estimated target's state suggests that the estimate is sufficiently reliable. In this case, they are updated using an adaptive filter

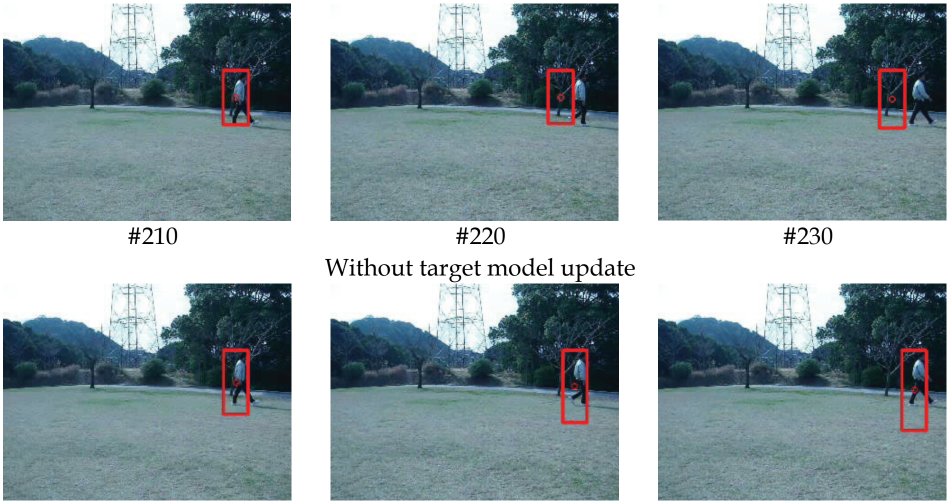
$$q_k = (1-\alpha_q)q_{k-1} + \alpha_q p_{est} \tag{20}$$

where $a_q \in [0, 1]$ is the learning rate that contribute to the updated histogram and p_{est} is histogram of the estimated state, respectively. In order to determine when the estimate is reliable, the likelihood of the current estimate is computed, π_{est}. The appearance is then updated when this value is higher than an indicator of the expected likelihood value and is calculated following an adaptive rule

$$\lambda_k = (1-\alpha_u)\lambda_{k-1} + \alpha_u \pi_{est} \tag{21}$$

here λ_k is expected likelihood, $a_u \in [0, 1]$ is the learning rate and π_{est} is estimated likelihood, respectively. This value indicates that the object has to be well matched to the model histogram before the update is applied.

Fig. 9 and Fig. 10 show the comparison of object tracking with and without target model update. As shown in Fig. 9, non-adaptive model fail to detect the object properly. However, the adaptive model can track the object successfully. It is because, using the adaptive model, the target model is always updated based on the best match of the color model. Moreover, as shown in Fig. 10, using adaptive model the scale of object appearance is getting adapted correctly compare to without using adaptive model.



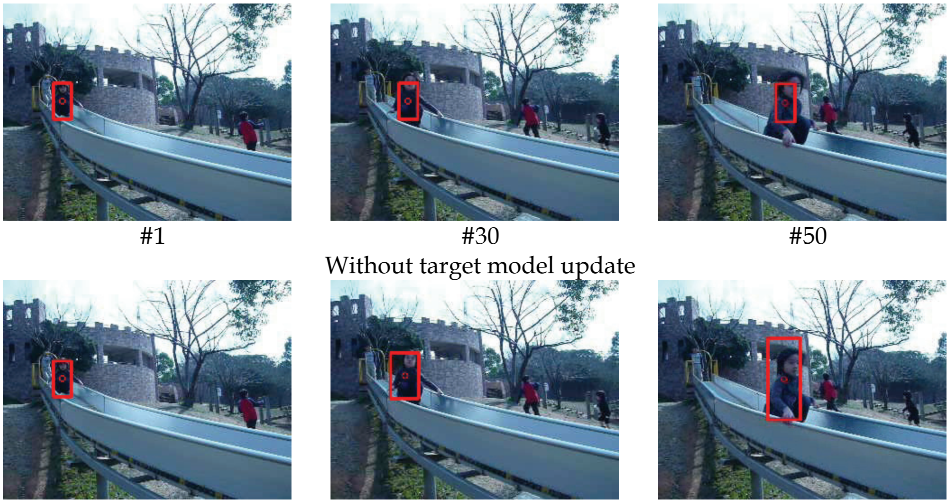
#210 #220 #230

Without target model update

#210 #220 #230

With target model update

Fig. 9. Object tracking with and without model update



#1 #30 #50

Without target model update

#1 #30 #50

With target model update

Fig. 10. Object tracking with model update shows the scale adaptation

4. Experimental results

In order to evaluate our proposed method, we have done the experiments using a video camera to track the objects in outdoor environment. The experiments are implemented on Pentium 4 with 2.54 [GHz] CPU and 512 [MB] RAM. The resolution of each frame is 320x240 [pixels] image. The color histogram is calculated in RGB space with $8 \times 8 \times 8$ [bins]. The experimental results are shown in Fig. 11 – Fig. 18. In each experimental result, the white dots represent the samples' distribution, the red dot represents the mean state of the samples' position and the red rectangle represents the bounding box of the tracked object used in calculation of the histogram distribution.

We did the experiments in three conditions. In the first condition, we tracked the moving object when the initial position of the object is unknown as shown in Fig. 12. In the second condition, we tracked the moving object when the initial position of the object is known as shown in Fig. 13 and Fig. 15. In the third condition, we tracked the moving object based on the appearance condition including the handling of object occlusion by another object as shown in Fig. 16 and Fig. 17.

4.1 Object tracking with unknown initial position

In this experiment, we track the moving object when the initial position of the object is unknown. We set the initial samples to be uniform distribution in $x_0 \sim U(1,320)$ and $y_0 \sim U(1,240)$. The experimental result is shown in Fig. 12. As presented in the figure, initially, the samples are distributed uniformly around the scene (frame #1). The object moves from left-hand to right-hand side and begin to appear on frame #5, however, the object starts to be tracked on frame #10. In this experiment, the variance of the samples' position distribution is utilized to judge whether the object has been tracked. We consider the object has been tracked by the system when the variance is below 10 and it occurs on the frame #10. Fig. 11 shows the variance positions of the samples. We can understand from that figure that at the beginning the variance is very high but after some while it decreases below the threshold. At that time, we determine that object is being tracked.

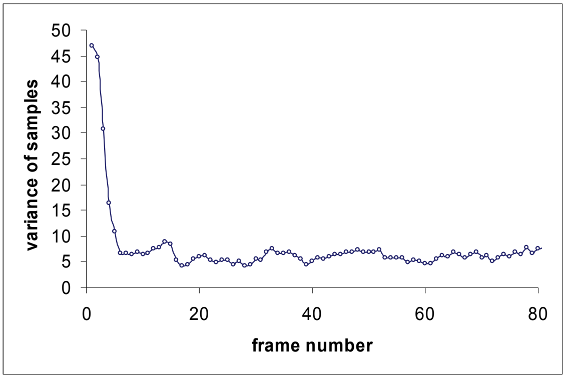
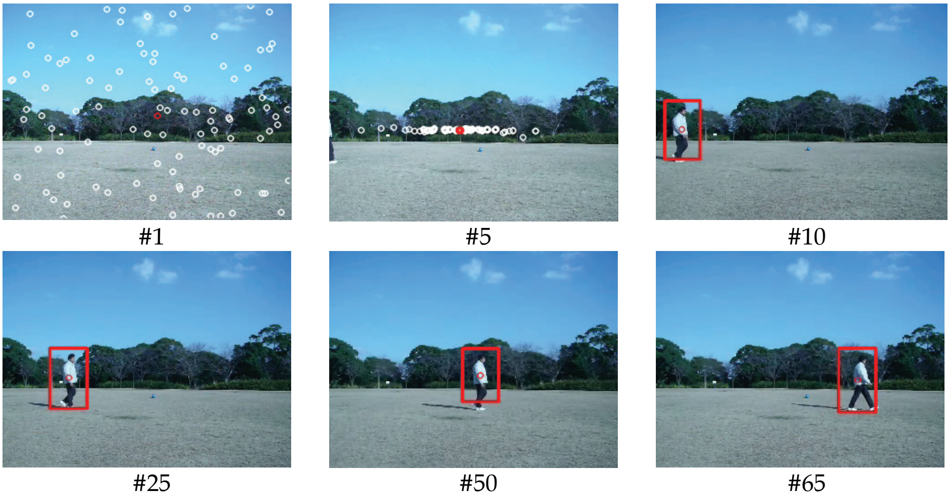


Fig. 11. Variance of samples' position


#1

#5

#10

#25

#50

#65

Fig. 12. Object tracking with unknown initial position

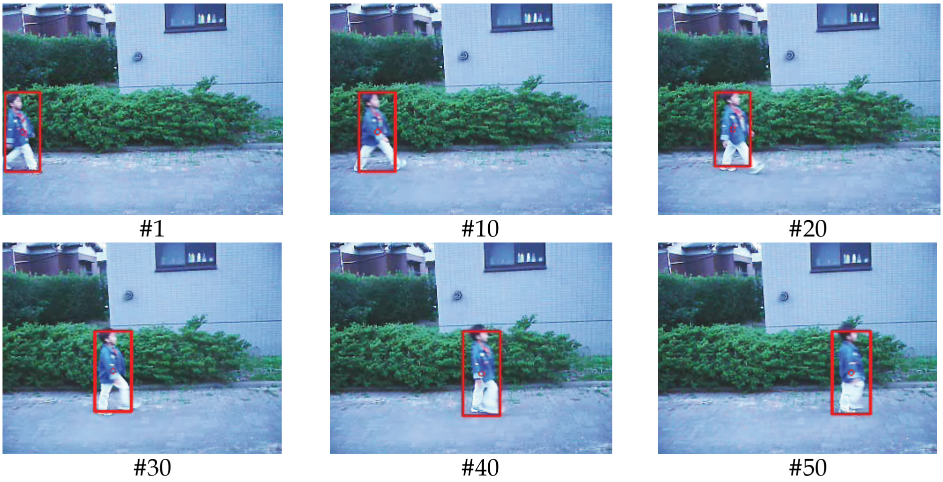

#1 #10 #20
#30 #40 #50

Fig. 13. Object tracking with known initial position

4.2 Object tracking with known initial position

In this experiment, the initial position of the moving object is known. We set the initial samples around the initial position of the object i.e. $x_0 \sim N(22,3)$ and $y_0 \sim N(145,3)$. The tracking result is presented in Fig. 13. Initially, the samples are distributed around initial position of the object. The object moves from left-hand to right-hand side and is tracked from the initial position (frame #1). We successfully tracked the object from the initial position until the end of the scene. Fig. 14 shows the comparison between true and estimated position of the tracked object. From that figure, we obtained the RMSE (root mean square error) of the estimated position is about 2.06 for X position and 2.18 for Y position.

Another result is shown in Fig. 15. Here, we set the initial samples around the initial position of the object i.e. $x_0 \sim N(65,3)$ and $y_0 \sim N(106,3)$. The obejct is tracked from the initial position. We can observe that the scale of object bounding box is adaptively updated. It is because, in this experiment, the target model is always updated using the adaptive model based on the best match of the color model.

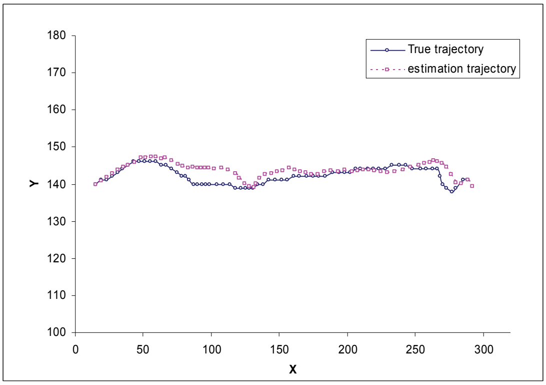
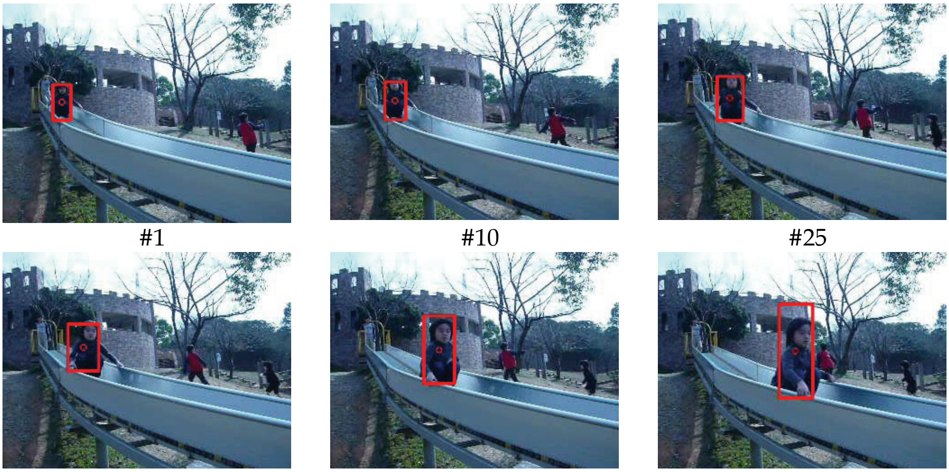


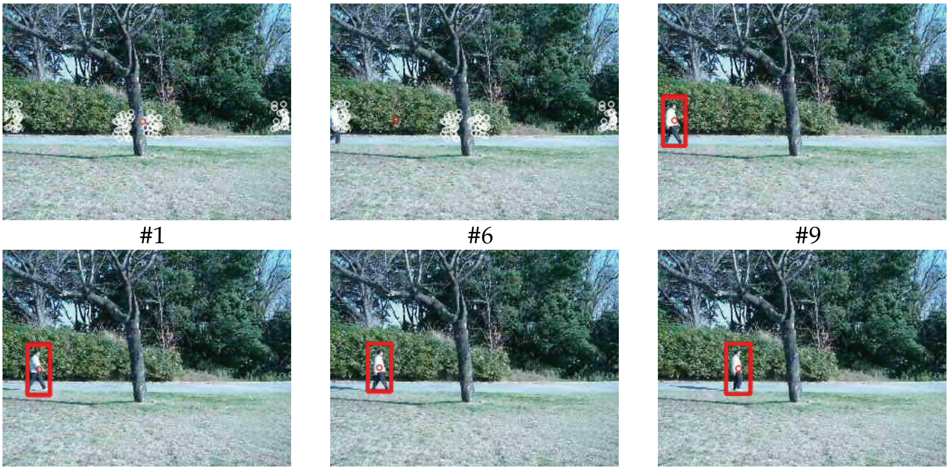
Fig. 14. True vs. estimate position



#1 #10 #25

#35 #45 #53

Fig. 15. Another result of object tracking with known initial position



#1 #6 #9

#15 #20 #30

Fig. 16. Object tracking based on appearance condition

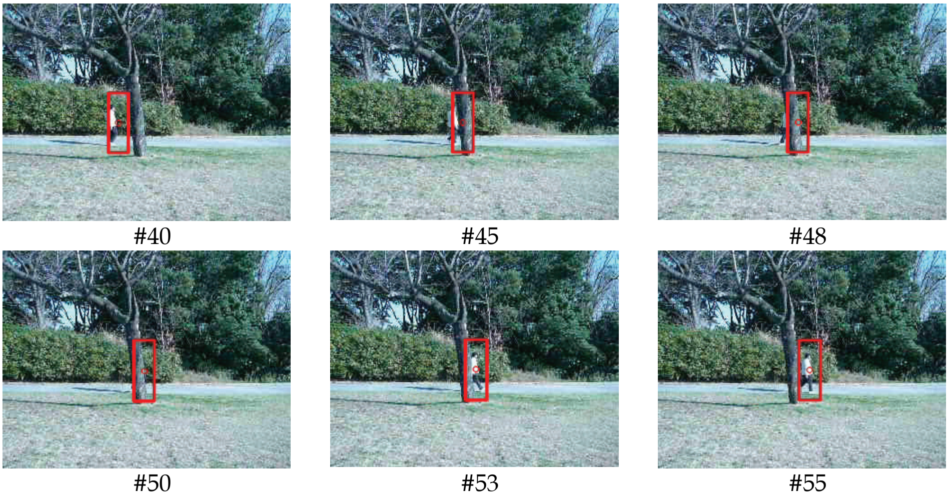
4.3 Object tracking with appearance condition and in the presence of occlusion

In this experiment, the samples initialize around the expected region where the target is most likely to appear, such as image borders and edge of the occluded object (tree). The experimental results are shown in Fig.16.

As presented in Fig. 16, the initial samples are placed at the position where the object is most likely to appear, such as image borders and edge of the occluded object (tree) as shown in frame #1. The object is started to appear on frame #6, but as the appearance condition is not fulfilled, the object is still judged not to be tracked. However, when the appearance condition is fulfilled (frame #9), the object starts to be tracked.

Fig. 17 shows the object occlusion handling based on appearance condition. As shown in that figure, we successfully track the object although the part and full occlusion occur. When occlusion occurs, the final mean state estimate is still calculated and the object is still tracked (frame #45 – frame #50). On that condition, the update and resample step are not performed due to no object color information can be represented the object. The samples are just propagated based on the system model and the target model is not updated until no more occlusion is detected. The object appears behind the tree (frame #55), due to the fulfilling of appearance condition, then the sample update and resample is performed again and the object continues to be tracked until the end of the frame.

The number of samples that fulfill the appearance condition is shown in Fig. 18. As presented in that figure, the object is detected and started to be tracked when the number of appearance more than 10. That means the object is detected and tracked when the number of samples fulfilling the appearance condition is more than 10 samples. We can see from the figure, the number of appearance is started by zero at the initial position. After some while, the number of appearance increases due to appearance of the object. Using the threshold 10 samples appearance, the object starts to be tracked on frame #9 (Fig. 16). The object continues to be tracked until it is occluded by tree (Fig. 17 frame #45 – frame #50). At this time, the number of samples at appearance condition is dropped to zero. At frame #55, the number of samples at appearance condition is higher again. It shows that the object is no more occluded by tree.



#40 #45 #48

#50 #53 #55

Fig. 17. Object tracking in the presence of oclusion

5. Conclusions

This paper presented a method to track the moving object employing particle filter in conjunction with color information in both known and unknown initial position of the object and in the presence of occlusion with static object. The robust color likelihood is used to evaluate samples properly associated to the target which present high appearance

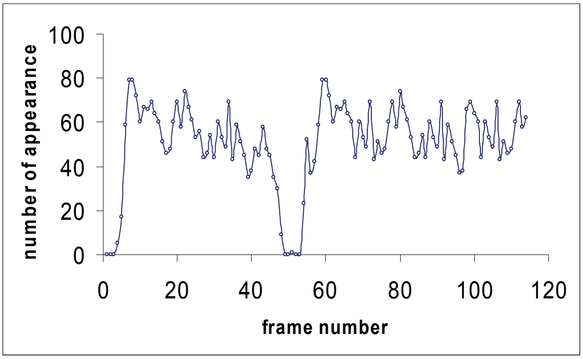


Fig. 18. Number of samples fulfilling the appearance condition

variability. We rely on Bhattacharya coefficient between target and sample histogram to perform this task. Model updating is carried to update the object in the presence of appearance change. The problem occlusion by static object can be tackled by prediction based on the dynamical model and appearance condition. The performance of the tracking algorithm was evaluated by the experiments. The experimental results show the algorithm can successfully track the single moving object in the presence of occlusion with static object and both known and unknown initial position of the object

Furthermore, the performance of the objects tracking can be improved in several ways such as adding the background modeling information in the calculation of likelihood, detection of appearing objects and extend the method to track multiple object in the presence of object occlusion and so on. Taking them into consideration could lead to some improvement. These are remaining for our future works.

6. References

Stauffer, C. & Grimson, W. (1999). Adaptive background mixture models for real-time tracking, *Proceeding of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 246-252.

LIU, Y.; Haizho, A. & Xu Guangyou.(2001). Moving object detection and tracking based on background subtraction, *Proceeding of Society of Photo-Optical Instrument Engineers (SPIE),* Vol. 4554, pp. 62-66.

McIvor, A. M. (2000). Background subtraction techniques, *Proceeding of Image and Vision Computing*, 6 pages.

Heikkila, J. & Silven, O. (1999). A real-time system for monitoring of cyclists and pedestrians, *Proceeding of Second IEEE Workshop on Visual Surveillance*, pp 74–81.

Lipton, A; Fujiyoshi, H. & Patil, R.(1998) .Moving target classification and tracking from real-time video, *Proceeding of IEEE Workshop Applications of Computer Vision*, pp. 8-14.

Collins, R. ; Lipton, A.; Kanade, T.; Fujiyoshi, H.; Duggins, D.; Tsin, Y.; Tolliver, D.; Enomoto, N. & Hasegawa. (2000). System for video surveillance and monitoring, *Technical Report CMU-RI-TR-00-12*, Robotics Institute, Carnegie Mellon University.

Meyer, D.; Denzler, J. & Niemann, H. (1998). Model based extraction of articulated objects in image sequences for gait analysis, *Proceeding of IEEE Int. Conf. Image Proccessing*, pp. 78-81.

Phung, S.; Chai, D. & Bouzerdoum, A. (2003). Adaptive skin segmentation in color images, *Proceeding of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 3, pp. 353-356.

Cho, K. M.; Jang, J. H. & Hong, K. S. (2001). Adaptive skin-color filter, *Pattern Recognition*, pp. 1067-1073.

Broida, T. & Chellappa, R. (1986). Estimation of object motion parameters from noisy images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 1, pp. 90–99.

Ristic, B.; Arulampalam, S. & Gordon, N. (2004). Beyond the Kalman filter: Particle filters for tracking applications, Artech House.

Isard, M. & Blake, A. (1998). CONDENSATION –Conditional Density Propagation for Visual Tracking, *Intl. Journal of Computer Vision*, Vol. 29, No. 1, pp. 5-28.

Kitagawa G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, *Journal of Comput. and Graph. Stat.*, Vol. 5, No. 1, pp. 1–25.

Gordon, N.; Salmon, D. & Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proc. on Radar and Signal Processing*, Vol. 140, No. 2, pp. 107-113.

Tanizaki, H. (1987). Non-Gaussian state-space modeling of non-stationary time series, *Journal of American Statistic Association*, Vol. 82, pp. 1032–1063.

King, O. & Forsyth, D. (2000). How does Condensation behave with a finite number of samples, *Proc. of 6th European Conference on Computer Vision*, pp. 695-709.

Nummiaro, K.; Koller, E. & Gool, L. (2002). An adaptive color-based particle filter, *Image and Vision Computing*, Vol. 25, No. 1, pp. 99–110.

Perez, P.; Hue, C. & Vermaak, J. (2002). Color-based probabilistic tracking, *Proc. of the 7th European Conference on Computer Vision*, 661–675.

Comaniciu, D.; Ramesh, V. & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift, *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recog*., pp. 142–149.

Comaniciu, D.; Ramesh, V. & Meer, P. (2003), Kernel-based object tracking, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 25, No.5, pp. 564-577.

Deutscher, J. & Reid, I. (2005). Articulated body motion capture by stochastic search, *Intl. Journal of Computer Vision*, Vol. 61, No. 2, pp. 185-205.

Simon, D. (2006). *Optimal State Estimation*, Jhon Wiey & Sons, Inc, Publication.

Freeman, W. & Adelson, E. (1991). The Design and Use of Steerable Filters, *IEEE Transactions on Pattern* Analysis *and Machine Intelligence*, Vol. 13, No. 9, pp. 891-906.

5

Online Learning and Robust Visual Tracking using Local Features and Global Appearances of Video Objects

Irene Y.H. Gu and Zulfiqar H. Khan

Dept. of Signals and Systems, Chalmers Univ. of Technology, Gothenburg, 41296 Sweden

1. Introduction

This chapter describes a novel hybrid visual object tracking scheme that jointly exploits local point features, global appearance and shape of target objects. The hybrid tracker contains two baseline candidate trackers and is formulated under an optical criterion. One baseline tracker, a spatiotemporal SIFT-RANSAC, extracts local feature points separatively for the foreground and background regions. Another baseline tracker, an enhanced anisotropic mean shift, tracks a dynamic object whose global appearance is most similar to the online learned distribution of reference object. An essential building block in the hybrid tracker is the online learning of dynamic object, where we employ a new approach for learning the appearance distribution, and another new approach for updating the two feature point sets. To demonstrate the applications of such online learning approaches to other trackers, we show an example in which online learning is added to an existing JMSPF (joint mean shift and particle filter tracking) tracking scheme, resulting in improved tracking robustness. The proposed hybrid tracker has been tested on numerous videos with a range of complex scenarios where target objects may experience long-term partial occlusions/intersections from other objects, large deformations, abrupt motion changes, dynamic cluttered background/occluding objects having similar color distributions to the target object. Tracking results have shown to be very robust in terms of tracking drift, accuracy and tightness of tracked bounding boxes. The performance of the hybrid tracker is evaluated qualitatively and quantitatively, with comparisons to four existing stat-of-the-art tracking schemes. Limitations of the tracker are also discussed.

2. Related work

2.1 Visual tracking

Visual object tracking has drawn increasing interest in recent years, partly due to its wide variety of applications, e.g., video surveillance in airports, schools, banks, hospitals, traffic, freight, and e-health cares. Tracking is often the first step towards a further analysis about the activities, behaviors, interactions and relationships between objects of interest. Many object tracking methods have been proposed and developed, e.g., state-space based tracking using Kalman filters and particle filters (Welch & Bishop,97; Rosales &

Sclaroff,99; Gordon et al.,01; Gordon,00; Wang et al.,08; Vermaak et al.,03; Okuma et al.,04), joint state-space representation and association (Bar-Shalom & Fortmann,98), multiple hypothesis tracking (MHT) (Reid,79), anisotropic mean shift tracking (Comaniciu et al.,03; Khan & Gu,10), optical flow-based tracking (Shi & Tomasi,94), and point feature-based tracking (Strandmark & Gu,09; Haner & Gu,10), among many others. An overview on visual tracking methods can be found in (Yilmaz et al.,06; Sankaranarayanan et al.,08).

In the state space-based tracking approach using Kalman Filters (KFs), the assumptions of Gaussian noise and linear models of state vector are made. A state vector typically includes different attributes of object, e.g. object appearance and shape, and/or other object features. G.Welch (Welch & Bishop,97) applies KFs to track user's poses in man-computer interactive graphics. R.Roosales (Rosales & Sclaroff,99) uses Extended Kalman Filters (EKFs) to estimate the 3D object trajectory from 2D motions. (Gordon et al.,01) uses Unscented Kalman Filters (UKFs) that enforces Gaussian distributions while keeps nonlinearity by using discrete samples to estimate the mean and covariance in posterior densities. Under Multiple Hypothesis Tests (MHTs), (Reid,79) uses an iterative process to track multiple objects and finds the best matching between the real object descriptors. Gordon et al. (Gordon,00) uses particle filers (PFs) to track 1D (one dimensional) signals. Tracking is achieved by estimating the probability density of state vector from synthetic nonlinear and non-Gaussian distributed 1D signals, and is formulated under the Bayesian framework by estimating the posterior probability using the rule of propagation of state density over time. Extension of PFs to visual tracking is not straight forwards, since the size of state vector for tracking a visual object is significantly larger than that of a 1D target. This requires a large number of particles and consequently a heavy computation, which often hampers the practical use of PFs. To overcome this, (Wang et al.,08) proposes to use Rao-Blackwellized PFs that marginalizes out the linear part of the state vector (the appearance), while the nonlinear shape and pose parts are then estimated by PFs, while (Khan et al.,09; Deguchi et al.,04) propose to embed the object appearance in the likelihood of PFs so that the size of the state vector can be kept small.

Visual tracking from mean shift has drawn much interest lately, partly due to its computational efficiency and relatively robust performance. Different from the conventional mean shift for nonlinear image smoothing or segmentation that seeks the local modes in the kernel estimate of pdf, mean shift tracking is an efficient and fast implementation of the similarity metric, the Bhattacharyya coefficient, that maximizes the similarity between the reference and a candidate object regions. It is worth mentioning that other similarity metrics, e.g., Kullback-Leibler divergence (Khalid et al.,05), or SSD measure (Hager et al.,04), can also be used as well. The main drawback of mean shift is that tracking may drift away or fail especially when the background clutter and the object of interest have similar color distributions, or when long term partial occlusions of objects, pose changes of large objects, and fast change of object motion occur. Following the pioneering work of mean shift tracking by (Comaniciu et al.,03), various attempts are made to address these issues. (Collins,03) extends the mean shift by introducing a normalizing factor to the bandwidth matrix to capture target variations in scales (Bretzner & Lindeberg,98). It performs extensive search within a range of ellipses and is computationally expensive. (Yilmaz,07) proposes tracking by using a level-set asymmetric kernel. It is performed in image coordinates by including the scale and orientation as additional dimensions and simultaneously estimating all unknowns. (Sumin & Xianwu,08) proposes to simultaneously track the position, scale and orientation of bounding box by using anisotropic mean shift, where the bandwidth matrix is used to compute the scale and orientation. (Zivkovic & Krose,04) proposes an EM-like algorithm that tracks a deformable object whose bounding box contains five degrees of freedom. It simultaneously estimates the center and the bandwidth matrix of kernel. (Maggio & Cavallaro,05; Xu et al.,05; Parameswaran et al.,07; Khan et al.,09) include the spatial information in the color histogram by dividing an ellipse shape bounding box into multiple parts to make the tracker more robust. (Maggio & Cavallaro,05; Khan et al.,09) further integrate the multi-part mean shift into the particle filter framework using overlapped, or non-overlapped regions where improved results are reported. The tracking performance is rather robust, however, tracking drift or tracking failure may still occur in some occasions, especially when a cluttered background or an intersecting object has similar color distributions to the target object.

While global appearance distributions are widely used in visual object tracking, local point features of object are often used as an alternative. One of the main advantages of using point features is their resilience to partial object occlusions. When one part of an object is occluded, point features from the non-occluded part can still be used for the tracking. Local appearance-based tracking usually involves detecting and characterizing the appearance of object by local features from points, lines or curves, establishing correspondences between detected feature points (lines, or curves) across frames and estimating the parameters of the associated transformation between two feature point (line, or curve) sets. Several strategies are used to select feature points that are invariant to affine or projective transformations. (Harris & Stephens,88) proposes to extract rotational and translational-invariant features by combining corner and edge detectors based on local autocorrelation functions. (Shi & Tomasi,94) proposes to threshold the minimum eigen values of image gradient matrices at candidate feature points and use them as the appropriate feature points for tracking. These methods generate rotational and translational invariant point features however are variant to affine or projective transformations. (Lowe,04) proposes a Scale-Invariant Feature Transform (SIFT) that is invariant to rotations, translations, scalings, affine transformations, and partially invariant to illuminations. Each point feature is described by a feature descriptor or a vector, formed from the gradient directions and the magnitudes around the point. (Bay et al.,06) proposes to use Speeded Up Robust Features (SURF), having similar performance as SIFT (Bauer et al.,07) but faster. Due to the robustness of SIFT features, various attempts (Skrypnyk & Lowe,04; Mondragon wt al.,07; Li et al,06; Xu et al,08; Battiato et al.,07) have been made to integrate SIFT in the tracking. (Skrypnyk & Lowe,04) proposes to use SIFT features to track camera poses and to register virtual objects in online videos. Since feature points are often sensitive to noise, the consensus of a group of points can be exploited. (Mondragon wt al.,07) uses SIFT and RANSAC to detect points of interest and reject outliers when estimating projective transformations, where videos are captured by an online UAV camera. (Li et al,06) handles object occlusions by matching local invariant features learned online rather than predicted from motions, since the local point features from a non-occluded object part can still be used for the tracking. (Xu et al,08) proposes vehicle tracking by using SIFT features extracted from detected moving object bounding boxes, followed by frame-by-frame matching. (Battiato et al.,07) proposes a video stabilizer by inferring the inter-frame motion through SIFT feature tracking in consecutive frames. These methods are efficient and invariant to scaling, rotation and moderate lighting changes, however require the appearance of object containing sufficient textures. Several attempts are made to solve this problem by combining SIFT features with other tracking methods. To extend the visual tracking from 2 images to a video sequence, efficient methods for establishing spatiotemporal local feature point correspondences through video frames are required. (Strandmark & Gu,09) proposes multiple motion models and feature point maintenance by employing an online updating process that may add new feature points, prune the existing points, or temporally freeze the updating, and (Haner & Gu,10) further improves the method by introducing two local feature point sets, one for the foreground and another for the background, where the background point set is used to provide priors on possible occlusions.

While both the global and local object models offer some attractive properties for visual tracking, hybrid models that combines these two types of models may offer better results as they can complement each other. (Zhou et al.,08) proposes an expectation-maximization algorithm that integrates SIFT features along with the color-based appearance in the mean shift, resulting in a better tracking performance even if one of the two methods becomes unreliable. (Wu et al.,08) enhances the performance of particle filters in cluttered background by taking into account the SIFT features in particle weights along with the color similarity measure. (Zhao et al.,08) uses feature point analysis to recover affine parameters, from which relative scales between two frames are estimated. It reconstructs target positions and relative scales using the affine parameters estimated from the SIFT feature correspondences. (Chen et al.,08) uses a similar method to handle the occlusion and scaling under the mean shift framework. While local feature points are promising for tracking, several problems remain, e.g. lacking of SIFT point features in smooth objects; lacking of sufficient feature point correspondences through video frames especially when the object contains pose changes, intersections and large deformations. (Khan & Gu,10) proposes to combine an enhanced anisotropic mean shift and a spatiotemporal SIFT-RANSAC procedure into a unified framework by using an optimal criterion, where the mean shift seeks global object appearance similarity and the spatiotemporal SIFT-RANSAC finds local feature points in the foreground/background. To further enhance the robustness against the tracking drift, the scheme also includes online learning of global appearances and local features.

2.2 Online learning

Online learning is another key issue that significantly impacts the performance of visual tracking. Most tracking methods require some kind of reference object models. Offline training videos from the same target object usually are not available, since video scenes from specific objects (e.g. suspicious actions of a particular person) captured by a surveillance camera are rarely repeatable. For tracking dynamic target objects, online learning of reference object appearances and/or shape is thus crucial. This is not trivial as the change of object can be caused by the object itself (e.g. change in colors, poses or shape), but also by partial/full occlusions from an occluding object or cluttered background, in addition to other changes such as lighting and illuminations. Online learning of dynamic objects requires that only the change associated with the target object itself is learned/updated into the model, while the remaining change from the background or other objects does not trigger the learning process. This is challenging since we have neither priors on the background/occluding objects, nor the information on when and where an occlusion may occur. Techniques for online learning vary depending on the attributes of object (e.g. global/local appearance, shape, motion) to be learned. Further, it depends on the technique used in the visual tracking as tracking and online learning are usually incorporated under a same framework. Many online learning techniques have been proposed. For example, (Lim et al.,04; Yang et al.,04; Wang et al.,07) perform incremental learning of 1D/2D PCA that describes the object appearance in a visual tracker. (Wang et al.,08) proposes an online Grassmann manifold learning scheme where dynamic object appearances are constrained on a smooth curved surface rather than a linear subspace. For online learning of pdf associated with each individual pixels, (Li et al.,08) suggests a color co-occurrence based method to learn the time-varying principal pdf of individual pixels that contain motions. For online learning of object appearance pdf used in the mean shift, (Khan & Gu,10) propose a robust online learning method in the regular frame intervals based on the criterion that determines whether the change is likely caused by the target object. For online learning of local features, (Strandmark & Gu,09) proposes multiple motion models and dynamic maintenance of the feature point set by using SIFT and RANSAC allowing online adding and pruning the feature points, or freezing the updating. (Haner & Gu,10; Khan & Gu,10) further improve the method by applying dynamic maintenance of separate foreground and background feature point sets under different criteria, where the background set is used to provide priors on occlusions to the foreground object.

2.3 Chapter outline

This chapter is focused on describing a hybrid visual tracking scheme, where both the local features and the global dynamic object appearance and shape are exploited. A key component of the tracker, the online learning, is employed to two baseline trackers: one is used to maintain the dynamic local feature point sets, and the other is used to learn the global appearance of dynamic object. The hybrid tracking scheme combines local point features and global appearances. It includes: (a) A local point feature-based candidate tracking method by employing consensus point feature correspondences separately in the foreground set and in the surrounding background set through using a spatiotemporal SIFT-RANSAC procedure. They are accompanied with an online maintenance process that can add, prune, freeze and re-initialize feature points in the sets; (b) A global appearance similarity-based candidate tracking method by using an enhanced anisotropic mean shift whose initial kernel is partially guided by the local point features, and is equipped with the online learning of reference object distribution; (c) The final hybrid tracker by combining the candidate trackers in (a) and (b) through using an optimal criterion.

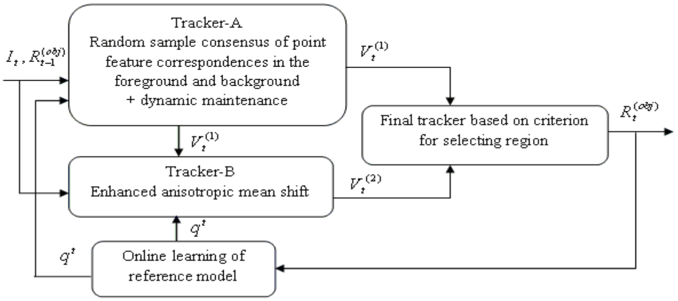
We then show that the online learning strategy for the candidate tracker in (b) can be directly applied to the online learning of another state-of-the-art visual tracking method, the joint anisotropic mean shift and particle filter (JMSPF) tracker, which may result in further tracking robustness in terms of tracking drift, tightness of tracked bounding boxes, and tracking failures in complex scenarios.

Experimental results on visual tracking of video objects with a range of difficult scenarios are included. Several distance metrics are used to quantitatively evaluate the robustness of the tracker, to evaluate the performance of the tracker with and without the online learning. Further performance evaluations are made qualitatively with three existing trackers, and quantitatively with two existing trackers. The computations are also compared for the hybrid tracker and three existing trackers.

The remainder of the chapter is organized as follows. The general structure and overall description of the hybrid tracker are given in Section 3. In Section 4, we describe two baseline trackers, one is based on using local feature point correspondences extracted from the spatiotemporal SIFT-RANSAC, and the other is based on the global object appearance similarities. In particular, Section 5 emphasizes two online learning techniques employed to these two baseline trackers. In Section 6, the hybrid tracker is formulated from the two baseline trackers under an optimal criterion. Section 7 describes a direct application of the online learning method to an existing joint anisotropic mean shift and particle filter (JMSPF) tracker. Section 8 is contributed to the experimental results and performance evaluations aimed at demonstrating the feasibility and robustness of the hybrid tracker. The advantages and limitations are also discussed. Finally, conclusions are given in Section 9.

3. A Robust hybrid visual tracking scheme: The big picture

This section describes the general structure and gives the big picture of the hybrid tracking scheme, where multiple issues are treated in a unified tracking framework. The tracking scheme, as shown in the block diagram of Fig.1, can be split into several basic modules. This is briefly summarized as follows:



$I_t, R_{t-1}^{(obj)}$
Tracker-A
Random sample consensus of point feature correspondences in the foreground and background + dynamic maintenance
$V_t^{(1)}$
$V_t^{(1)}$
Final tracker based on criterion for selecting region
$R_t^{(obj)}$
Tracker-B
Enhanced anisotropic mean shift
$V_t^{(2)}$
q^t
q^t
Online learning of reference model

Fig. 1. Block diagram of the hybrid tracking scheme, where $V_t^{(i)}$, $i = 1, 2$, is the parameter vector for the tracked candidate region $R_t^{(i)}$ from the baseline tracker-A and tracker-B at time t, I_t is the tth video frame, $R_{t-1}^{(obj)}$ is the finally tracked object region from the hybrid tracker at $t-1$, and q^t is the estimated appearance pdf for the reference object at t.

(a) *Two online learning methods.* Two novel methods for online learning of dynamic object are described: one is used for online maintenance of local point feature sets, and the other is for dynamically updating the object appearance distribution. This is aimed at keeping a time-varying reference object description thereby the tracking drift can be reduced. The method is based on seeking the best frame, indicated by reliable tracking without occlusions, in each fixed-size frame interval. This is achieved by using a criterion function in the interval.

(b) *Baseline tracker-A: exploit local point features for object tracking.* The baseline tracker-A in the hybrid tracker is realized by exploiting the local point features. Local point features are useful when partial occlusions occur: point features in non-occluded part remain unchanged, despite the global object appearance may experience significant changes. To make the point feature-based tracking robust, two sets of point features are utilized: one for the foreground region and the other for the background region (see Fig.1). The idea of using the background point features is to provide priors to the foreground on possible occlusions. The local point features are collected by introducing a novel spatiotemporal SIFT-RANSAC procedure followed by an online maintenance process that may add, prune and update feature points in both sets. To prevent drift and error propagation, a re-initialization process is also used.

(c) *Baseline tracker-B: exploit global object appearance for object tracking.* The baseline tracker-B in the hybrid tracker is realized by exploiting global object appearance distributions. To find the most similar appearance object compared with the reference (e.g., previously tracked object, or a reference object), anisotropic mean shift with a 5-degree parametric bounding box is used. An enhancement is added to the conventional mean shift by allowing its kernel partially guided by the local point features. This may reduce the tracking drift as the mean shift is sensitive to cluttered background/occluding objects having the similar color distribution to the foreground object. Online learning of object appearance distributions and re-initialization introduced for achieving more tracking robustness and preventing propagation of tracking drift across frames.

(d) *The hybrid tracking scheme: formulated from an optimal criterion.* The above local point feature-based tracker and global object appearance-based tracker in (b) and (c) are exploited jointly to form the hybrid tracker. It is formulated by using an optimal criterion that parallel employs the two baseline trackers and one from their weighted combination.

4. Baseline tracking methods using local features and global appearances

This section describes two baseline tracking methods, one is based on using local point features of object (Tracker-A in Fig.1), and the other is based on using global object appearance (Tracker-B in Fig.1).

4.1 Local feature point-based visual tracking

This section describes one baseline tracking method, tracker-A, in the hybrid tracker shown in Fig.1. It is a local feature point-based tracking method realized by a *spatiotemporal SIFT-RANSAC* procedure. It generates separate local feature point sets in the foreground and the surrounding background respectively.

The use of local point features is motivated by problems encountered in tracking partially occluded objects, or objects having similar color distributions to the cluttered background. In these scenarios, local salient point features from non-occluded object parts, or salient point features of object may be exploited for tracking. For matching local point features, two well known computer vision techniques are employed: SIFT (scale-invariant feature transform) (Lowe,04) and RANSAC (random sample consensus) (Fischler & Bolles,81) are used. The former is used to match scale-invariant feature points, while the latter is used to remove outliers. A brief review of SIFT and RANSAC is given in Section 4.1.1. For introducing more robustness to partial occlusions/itersections, two sets of feature points, one to the foreground area and another to the background area surrounding the candidate object, are employed. The background set is used to provide priors on occlusions. This is described in Section 4.1.2.

It is worth emphasizing that one of the key steps to realize the spatiotemporal SIFT-RANSAC is the online maintenance of point feature sets (in Section 5.1), while the conventional SIFT and RANSAC usually cannot be applied successfully to a long video sequence (e.g. of a few hundreds of frames).

4.1.1 Review: SIFT and RANSAC for feature point correspondences

Two standard computer vision techniques, SIFT (Lowe,04) and RANSAC (Fischler & Bolles,81), are briefly reviewed in this subsection.

In SIFT, each point feature is described by a feature vector

$$\mathbf{f}_i = \{p_i, \Phi_i\} = \{p_i, \sigma_i, \varphi_i, \mathbf{g}_{h_i}\} \tag{1}$$

where $p_i = (x_i, y_i)$ is the 2D position of SIFT keypoint, $\Phi_i = \{\sigma_i, \varphi_i, \mathbf{g}_{h_i}\}$ is the parameter vector associated with the point p_i, including the scale σ_i, the main gradient orientation within the region φ_i and the gradient orientation histogram $\mathbf{g}_{h_i}$ (128 bins). First, the original image I is convolved with a bandpass filter h to obtain a filtered image I_o, $I_o(x, y, \sigma) = I(x, y) * h(x, y, k\sigma)$, where $h(\cdot)$ is formulated from the difference of two scale Gaussian shape kernels $h(x, y, \sigma) = g(x, y, k\sigma) - g(x, y, \sigma)$. The location of each feature point (SIFT keypoint) $p_i, i = 1, 2, \cdots$, at scale σ_i corresponds to the thresholded local extrema of $I_o(x, y, \sigma)$. For each SIFT keypoint, one or more principal orientations θ_i are assigned by computing the gradient magnitudes and orientations in a region surrounding the point and finding 80% peaks in the orientation histogram. The orientation histogram is computed from a 16×16 region centered at p_i, partitioned into 4×4 blocks each consisting of 8 bins. This results in a total of $8 \times 16 = 128$ bins. The value of orientation histogram $\mathbf{h}_i$ is obtained by summing up the gradient magnitudes in each bin.

Matching SIFT keypoints in two image frames is obtained by searching the Euclidian distance from the nearest neighboring keypoints with the minimum errors. Under a pre-defined motion model, corresponding SIFT keypoints across two image frames are related. For example, if two images are related to an affine transform $T(\beta, \theta, (d_x, d_y))$, then each pair of SIFT keypoints is related by $\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = \beta \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$. Equivalently, given two sets of keypoints $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and $\{(\tilde{x}_1, \tilde{y}_1), \cdots, (\tilde{x}_n, \tilde{y}_n)\}$ from two images (e.g. at $(t-1)$ and t), these pairs of keypoints are related by,

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \vdots \\ \tilde{x}_n \\ \tilde{y}_n \end{bmatrix} = \begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & -y_n & 1 & 0 \\ y_n & x_n & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta\cos\theta \\ \beta\sin\theta \\ d_x \\ d_y \end{bmatrix} \tag{2}$$

The LS (Least Squares) estimation, $argmin_{(\beta,\theta,d_x,d_y,\text{ index order of } p_j)} ||\tilde{p}_i - T(p_j)||_2$, can be applied to find the best matching and to estimate the transform parameters β, θ, (d_x, d_y) at t. To make the matching robust against the false positive, the best matching is selected if the ratio of distances between the first and the second nearest neighbors is less than some empirically determined threshold (Lowe,04).

Since individual SIFT keypoints are prune to noise, RANSAC is often followed to remove outliers through finding the maximum number of consensus correspondences and estimating the associated motion parameters. RANSAC contains a two-step iterative process: estimate the parameters of the transform $T_i^{(t)}$ and find a subset of inlier points from the SIFT keypoints that yield the maximum consensus under $T_{i^*}^{(t)}$. In the first step, it differentiates outliers from inliers by selecting a minimum number of points needed to estimate the transform from the SIFT keypoint set at random. Then, the parameters of the transform are estimated. Using the estimated parameters, more points are picked up if they fit to this specific transform. This is done by calculating the error for each pair of keypoints and comparing with a small error threshold T_e. The iteration is repeated until the error is smaller than T_e, or the total number of iterations exceeds a pre-specified maximum iteration number T_{iter}. In the second step, it fits the transform to the inliers while ignoring the outliers. The transform parameters are updated using all collected inlier points. A tracked candidate object region is then obtained by drawing a tight rectangle surrounding the consensus points.

4.1.2 Using separate feature point sets for the foreground and the background

In the local point feature-based baseline tracker, two separate feature point sets, $\mathcal{P}^F$ and $\mathcal{P}^B$, are formed. $\mathcal{P}^F = \{p_i^F \mid p_i^F : \Phi_i^F\}$ is for the foreground, and $\mathcal{P}^B = \{p_i^B \mid p_i^B : \Phi_i^B\}$ is for the background surrounding the candidate object. In each set, the parameter vector Φ_i^F (or, Φ_i^B) is defined according to (1). The basic idea of using background points is to extract priors on possible object occlusions or intersections. As shown in Fig.2, a *searching area* (the black rectangle) is defined to be larger than that of an object region (the red rectangle). The region between the searching area and the candidate object region (between the black and red rectangles) is defined as the background region.

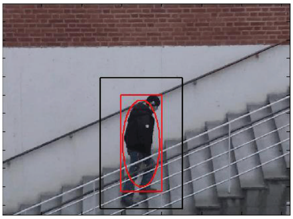


Fig. 2. Foreground and background regions. Red rectangle: a foreground object region; Black rectangle: the searching area. The area between the black and red rectangles is the background region. Red ellipse: maybe used for some objects (e.g humans) to minimize the possible inclusion of background points around the four corner areas.

Both sets of feature points are extracted by combining SIFT and RANSAC as described above, however, in different regions. Finally, a tight outer rectangular boundary surrounding the selected foreground feature points in $\mathcal{P}^F$ is drawn as the tracked candidate object region $R^{(1)}$ for the baseline tracker-A. The region is specified by the parameter vector $V^{(1)} = [p_c = (y_{1,c}^{(1)}, y_{2,c}^{(1)}), w^{(1)}, h^{(1)}, \theta^{(1)}, \mathcal{P}^F]^T$ including the 2D center position, the width, height and orientation of the region, and the foreground sets. To reduce the possibility of including background points in the foreground set, the shape of object type may be considered. For example, for human objects, foreground feature points may be constrained within an ellipse that is tightly made within the rectangular region $R^{(1)}$.

It is worth mentioning that the resulting region $R^{(1)}$ is also provided to the baseline tracker-B (in Section 4.2) to enhance the conventional mean shift which is prune to the background or occluding objects with similar color distributions.

4.1.3 Re-initialization

A re-initialization process may be applied to some frames to prevent tracking drift or tracking error propagation across frames. The idea can be analogue to using I (intra-coding) frames in video compression. A re-initialization process for the baseline tracker-A is used to avoid severe errors, e.g., when the number of corresponding points is very small, unreliable tracking or accumulated tracking drift may occur. In the former case, a small number of feature points may lead to an ill-posed RANSAC estimation, or a unreliably tracked region. In the latter case, accumulated drift may eventually lead to tracking failure. A tracker thus needs to be re-initialized to avoid the propagation of errors across frames.

Based on the observation that a bounding box does not change significantly in consecutive frames, and that a very low similarity value between the tracked region and the reference object indicates a possible tracking drift or unstable tracking, the frames for re-initialization can be selected. Two conditions, the distance of consecutive box shape and the Bhattacharyya coefficient between the tracked and the reference object regions, are used to determine whether a re-initialization is applied. That is, if one of the following two conditions is satisfied,

$$dist_t^{(1)} = \sum_{i=1}^{4} \|\mathbf{x}_{t,i}^{(1)} - \mathbf{x}_{t-1,i}^{(1)}\|^2 > T_1^{(1)}, \text{ or } \rho_t^{(1)} < T_2^{(1)} \tag{3}$$

then the baseline tracker-A is re-initialized at t by:

$$R_t^{(1)} \leftarrow R_{t-1}^{(obj)}, \; V_t^{(1)} \leftarrow V_{t-1}^{(obj)}, \text{ and } \tilde{\rho}_t^{(1)} \leftarrow 0 \tag{4}$$

where $R_{t-1}^{(obj)}$ is the tracked bounding box from the final hybrid tracker at $(t-1)$, and $V_{t-1}^{(obj)}$ is the corresponding parameter vector, $\tilde{\rho}_t^{(1)}$ is the normalized Bhattacharyya coefficient defined in (21), $\mathbf{x}_{t,i}^{(1)}$ and $\mathbf{x}_{t-1,i}^{(1)}$ are the four corners of the object bounding box at t and $(t-1)$ from the baseline tracker-A, $\rho_t^{(1)}$ is the Bhattacharyya coefficient for the baseline tracker-A, and $T_1^{(1)}$ and $T_2^{(1)}$ are the empirically determined thresholds.

4.2 Global appearance-based visual tracking

This section describes another baseline tracking method, tracker-B, in the hybrid tracker shown in Fig.1. It is a global object appearance-based tracking method realized by an enhanced anisotropic mean shift. Based on the observation that mean shift tracking yields reasonably good tracking results, however, tracking drift may occur when nearby objects or background clutter have similar color distributions. To tackle the problem, two strategies are introduced here to the mean shift: The first one is to employ an *enhanced* anisotropic mean shift, where the result from the point feature-based tracking (the baseline tracker-A) is used to partially guild the location of mean shift kernel. The second strategy is to add online learning of reference object appearance (in Section 5.2), as well as a re-initialization process to prevent the propagation of tracking drift. This baseline tracker-B results in a tracked candidate object region $R^{(2)}$ specified by a parameter vector $V^{(2)} = [y^{(2)} = (y_1^{(2)}, y_2^{(2)}), w^{(2)}, h^{(2)}, \theta^{(2)}, \mathbf{h}_{rgb}^{(2)}]^T$, where $y^{(2)} = (y_1^{(2)}, y_2^{(2)})$, $w^{(2)}$, $h^{(2)}$ and $\theta^{(2)}$ are the 2D center, width, height and orientation of $R^{(2)}$, and $\mathbf{h}_{rgb}^{(2)}$ is the color histogram.

4.2.1 Anisotropic mean shift

This subsection briefly reviews the anisotropic mean shift for visual tracking. More details of mean shift can be found in (Sumin & Xianwu,08; Comaniciu et al.,03).

Let the pdf estimate $p(y,\Sigma) = \{p_u(y,\Sigma), u = 1, \cdots, m\}$ for a candidate object be the spatial kernel-weighted color histogram within the bounding box in the image $I(y)$, and the spatial kernel-weighted color histogram $q(x_c,\Sigma_c) = \{q_u(x_c,\Sigma_c), u = 1, \cdots, m\}$ for the reference object within the bounding box in $I_0(x)$. The corresponding histogram bins for the candidate and reference objects are described respectively as follows:

$$\begin{aligned} p_u(y,\Sigma) &= \frac{c}{|\Sigma|^{\frac{1}{2}}} \sum_{j=1}^{n} k(\tilde{y_j}^T \Sigma^{-1} \tilde{y_j}) \delta[b_u(I(y_j)) - u] \\ q_u(x_c,\Sigma_c) &= \frac{c_0}{|\Sigma_c|^{\frac{1}{2}}} \sum_{j=1}^{n} k(\tilde{x_j}^T \Sigma_c^{-1} \tilde{x_j}) \delta[b_u(I_0(x_j)) - u] \end{aligned} \tag{5}$$

where $\tilde{y_j} = (y_j - y)$, $\tilde{x_j} = (x_j - x_c)$, Σ (or, Σ_c) is a kernel bandwidth matrix, $b_u(I(y_j))$ (or, $b_u(I_0(x_j))$) is the index of color histogram bin at the location y_j (or, x_j) associated with the candidate (or, reference) object region, y_j (or, x_j) is summed over all pixels within the bounding box, c (or, c_0) is a constant used for the normalization, m is the total number of bins, $k(\cdot)$ is the spatial kernel profile, and y (or, x_c) is the center of the kernel (or, bounding box).

To measure the similarity between a candidate and the reference object region, the Bhattacharyya coefficient ρ defined below, is used:

$$\rho(p,q) = \sum_{u=1}^{m} \sqrt{p_u(y,\Sigma)q_u} \tag{6}$$

Applying the first-order Taylor series expansion to (6) around (y_0, Σ_0), (where y_0 and Σ_0 are the kernel center and bandwidth in the previous frame) yields $\rho \approx \sum_u \frac{1}{2}\sqrt{q_u p_u(y_0,\Sigma_0)} + \frac{c}{2|\Sigma|^{\frac{1}{2}}} \sum_j \omega_j k(\tilde{y}_j^T \Sigma^{-1} \tilde{y}_j)$, where $\omega_j = \sum_u \sqrt{\frac{q_u}{p_u(y_0,\Sigma_0)}} \delta[b_u(I(y_j)) - u]$. The kernel center (or, bounding box center) can be estimated by setting $\nabla_y \rho(p,q) = 0$. This leads to:

$$\hat{y} = \frac{\sum_{j=1}^{n} g(\tilde{y}_j^T \Sigma^{-1} \tilde{y}_j)\omega_j x_j}{\sum_{j=1}^{n} g(\tilde{y}_j^T \Sigma^{-1} \tilde{y}_j)\omega_j} \tag{7}$$

where $g(\cdot) = -k'(\cdot)$ is the shadow of the kernel. To estimate $\hat{\Sigma}$, a γ-normalized kernel bandwidth Σ (in (Bretzner & Lindeberg,98)) is applied to ρ, where y in $\tilde{y}_j$ is substituted by $\hat{y}$ that is obtained from (7). The kernel bandwidth matrix is estimated by setting $\nabla_\Sigma(|\Sigma|^{\gamma/2}\rho(p,q)) = 0$, yielding,

$$\hat{\Sigma} = \frac{2}{1-\gamma} \frac{\sum_{j=1}^{n} \omega_j \, g(\tilde{y}_j^T \Sigma^{-1} \tilde{y}_j)\tilde{y}_j^T \tilde{y}_j}{\sum_{j=1}^{n} \omega_j \, k(\tilde{y}_j^T \Sigma^{-1} \tilde{y}_j)} \tag{8}$$

where γ is empirically determined, and $\tilde{y}_j = (y_j - \hat{y})$. The estimation of (7) and (8) are done alternatively in each iteration. The iteration repeats until the estimated parameters converge, or a pre-specified maximum number of iterations is reached.

4.2.2 Estimating shape parameters of bounding box

For estimating the parameters of bounding box in the baseline tracker-B, a simple approach different from (Sumin & Xianwu,08) is employed. The anisotropic mean shift used in the baseline tracker-B contains a fully tunable affine box with five degrees of freedom, i.e., the 2D central position, width, height and orientation of the box.

Let the orientation of the box be defined as the angle between the long axis of bandwidth matrix and the horizontal-axis of the coordinate system. Let the height h and width w of the bounding box be defined as the radii along the long and short axes of an ellipse, as depicted in Fig.3. Since h, w, and θ are related to the kernel bandwidth matrix Σ by,

$$\Sigma = \mathbf{R}^T(\theta) \begin{bmatrix} (h/2)^2 & 0 \\ 0 & (w/2)^2 \end{bmatrix} \mathbf{R}(\theta) \tag{9}$$

where $\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$, computing these parameters can be efficiently done by applying eigenvalue decomposition to Σ,

$$\Sigma = V \Lambda V^{-1} \tag{10}$$

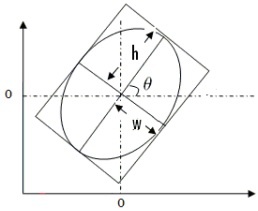


Fig. 3. Definition of the width, height and orientation of a bounding box.

where $V = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$, $\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$, and by relating these parameters with eigenvectors and eigenvalues using (9) and (10) as follows,

$$\hat{\theta} = \tan^{-1}(v_{2,1}/v_{1,1}), \;\; \hat{h} = 2\sqrt{\lambda_1}, \;\; \hat{w} = 2\sqrt{\lambda_2} \tag{11}$$

where v_{11} and v_{21} are the two components from the largest eigenvector. The first five parameters in $V^{(2)}$ are obtained from the estimates in (7), (8) and (11).

4.2.3 Enhancing the anisotropic mean shift

The basic idea of enhanced mean shift is to partially guild the location of mean shift kernel from the result of local feature point-based tracker (or, baseline tracker-A). This is designed for correcting possible tracking drift due to, e.g. similar color distributed object / background clutter, or partial object occlusions/intersections. Enhancement is done by assigning the mean shift tracker to an area that is also agreeable with that from the local feature points of target.

Areas used for the mean shift and for the candidate object: To limit the number of background pixels entering the foreground region, one may use a slightly smaller ellipse area inside the rectangular box (e.g., scaled by K, K=0.9 in our tests) of candidate object region $R^{(2)}$. This is based on the observation that an ellipse box can be tighter for some object types and would therefore exclude the background pixels around the four corners of the rectangular box.

The result from the local feature-based tracker (i.e. baseline tracker-A) is employed to guide the kernel position of mean shift, if the result from the baseline tracker-A is shown to be reliable. This is done by examining the Bhattacharyya coefficient and the number of consensus feature points in the tracked object region $R_t^{(1)}$. If they are both high (indicating a reliable result), then the initial parameter vector in $R_t^{(2)}$ for the baseline tracker-B is assigned by that in the tracker-A, otherwise by the parameter vector from the tracker-B at $t-1$, i.e.

$$V_t^{(2)} = \begin{cases} V_t^{(1)} & \text{if } |\mathcal{P}^F| > T_1^{(2)} \text{ and } \rho_t^{(1)} > T_2^{(2)} \\ V_{t-1}^{(2)} & \text{otherwise} \end{cases} \tag{12}$$

where $T_1^{(2)}$ and $T_2^{(2)}$ are thresholds determined empirically.

4.2.4 Re-initializing the region

A re-initialization process is added to tackle the issue of tracking drift or tracking error propagation across frames. The idea can be analogue to applying intra video coding in each fixed frame interval. The following criterion, in a similar spirit to that in the baseline tracker-A, is used to determine whether or not the re-initialization is applied to the tracked region $\hat{R}_t^{(2)}$ for the baseline tracker-B. That is, if one of the following two conditions is satisfied:

$$dist_t^{(2)} = \sum_{i=1}^{4} \|\mathbf{x}_{t,i}^{(2)} - \mathbf{x}_{t-1,i}^{(2)}\|^2 \; > \; T_3^{(2)}, \text{ or } \; \rho_t^{(2)} \; < \; T_4^{(2)} \tag{13}$$

then, the baseline tracker-B is re-initialized at t by:

$$R_t^{(2)} \leftarrow R_{t-1}^{(obj)}, \;\; V_t^{(2)} \leftarrow V_{t-1}^{(obj)} \text{ and } \tilde{\rho}_t^{(2)} \leftarrow 0 \tag{14}$$

where $R_{t-1}^{(obj)}$ is the previous tracked bounding box from the final hybrid tracker at $(t-1)$ and $V_{t-1}^{(obj)}$ is the corresponding parameter vector, $\tilde{\rho}_t^{(2)}$ is the normalized Bhattacharyya coefficient defined in (21), $\mathbf{x}_{t,i}^{(2)}$ and $\mathbf{x}_{t-1,i}^{(2)}$ are the four corners of tracked object regions $R_t^{(2)}$ and $R_{t-1}^{(2)}$ from the baseline tracker-B, $T_3^{(2)}$ and $T_4^{(2)}$ are two empirically determined thresholds.

5. Online learning in the spatiotemporal SIFT-RANSAC and the enhanced anisotropic mean shift

Online learning is an important step that may significant impact the tracking performance. In this section, we describe two online learning techniques, one is utilized to maintain two feature point sets from the spatiotemporal SIFT-RANSAC in the baseline tracker-A, another is applied to update the reference object distribution from the enhanced anisotropic mean shift in the baseline tracker-B.

5.1 Online learning of local point feature sets

This subsection describes a key step in the spatiotemporal SIFT-RANSAC: online learning of two feature point sets across video frames. The use of spatiotemporal SIFT-RANSAC is motivated by the problems encountered in the conventional SIFT-RANSAC for tracking objects through long video sequences. Often, the number of corresponding feature points reduce significantly through video frames due to object pose changes, partial occlusions or intersections. In these cases, some feature points on the object may disappear, consequently only a subset of feature points finds their correspondences across two image frames. This phenomenon may propagate through video frames, and may eventually lead to a dramatically reduced number of corresponding points. When the number of points is very small, the region surrounding these points may become very small and unreliable. Further, motion parameter estimation can become ill-posed if the number of equations is less than the unknown parameters.

The online learning procedure in the spatiotemporal SIFT-RANSAC is designed to dynamically maintain the corresponding point sets through video frames. This includes the possibility of adding new points, pruning weak points with low weights, and freezing the adaptation when partial occlusions are suspected. This online learning is applied separately to the foreground candidate object region and the surrounding background region. The method for online maintenance of spatiotemporal point correspondences is similar to the work in (Haner & Gu,10; Khan & Gu,10), which contains the following main steps:

- Assign weights to all corresponding feature points;
- Add new candidate feature points;

- Prune feature points with low weights;
- Freeze the updating when a partial occlusion of object is likely to occur;
- Separate maintenance of the foreground feature point set $\mathcal{P}^F$ and the background feature point set $\mathcal{P}^B$.

5.1.1 Maintenance of foreground feature point set $\mathcal{P}^F$

For this feature point set, online learning of the dynamic feature points contains the following main steps:

Assigning weights: First, a set of feature points $\mathcal{P}^F = \{p_i = (x_i, y_i),\ i = 1, 2, \cdots\}$ at the frame t is selected from the SIFT, within the transformed bounding box area obtained by using the estimated affine transform parameters from the RANSAC to the tracked object box at (t-1). Feature points from RANSAC that are outside the transformed bounding box (i.e. belong to the background) are then removed. After that, a candidate consensus point set $\mathcal{P}^F$ is created. $\mathcal{P}^F$ consists of three subsets of feature points according to how consensus correspondences in the foreground set are established,

$$\mathcal{P}^F = \{\mathcal{P}_a^F \cup \mathcal{P}_b^F \cup \mathcal{P}_c^F\} \tag{15}$$

In (15), $\mathcal{P}_a^F$ contains matched consensus points, i.e. the feature points selected by the RANSAC, $\mathcal{P}_b^F$ contains the outliers that fail to agree with the best estimated transform parameters in the RANSAC (which could be the result from noise influence or object dynamics), and $\mathcal{P}_c^F$ is the set of newly added feature points from the SIFT that are initiated within the candidate object region at t and do not correspond to any background feature points. Each feature point p_i in the candidate set $\mathcal{P}^F$ is assigned to a weight according to:

$$W_t^i = \begin{cases} W_{t-1}^i + 2 & p_i \in \mathcal{P}_a^F \\ W_{t-1}^i - 1 & p_i \in \mathcal{P}_b^F \\ W_0^i & p_i \in \mathcal{P}_c^F \end{cases} \tag{16}$$

where the initial weight for a new feature point W_0^i is set to be the median weight value of all points in the subsets $\mathcal{P}_a^F$ and $\mathcal{P}_b^F$, i.e. $W_0^j = \text{median}(W_t^j | p_j \in \mathcal{P}_a^F \cup \mathcal{P}_b^F)$, and W_{t-1}^i for $\mathcal{P}_a^F$ and $\mathcal{P}_b^F$ is initialized to zero in the first frame. Once the maximum consensus points are selected at t, their weights in (16) are increased. For those corresponding points that do not fit to the estimated transform parameters in the RANSAC (i.e. matched outliers), their weights in (16) are reduced.

Adding or pruning consensus points: After updating the weights in (16), feature points in $\mathcal{P}^F$ are then updated. This is done by first sorting out the feature points in $\mathcal{P}^F$ according to their weights. New feature points in $\mathcal{P}_c^F$ are added with the median weight values, so that they may remain in the set after the subsequent pruning process. The pruning process is then applied to keep a reasonable size of $\mathcal{P}^F$ by removing low weight feature points in the set. This is done by keeping the L_F (L_F empirically determined, L_F=1000 in our tests) highest weight points in the set and removing the remaining ones.

Freezing the updating when a partial occlusion is highly probable: If an object is occluded by cluttered background or intersected by other objects, object appearance within the bounding box may change significantly. The Bhattacharyya coefficient value may indicate the existence of such scenarios, as the images between the tracked object and the reference object become less similar. When such a scenario occurs, the online maintenance process should be temporally frozen in order to not include the feature points from the background clutter or the occluding object. The Bhattacharyya coefficient is computed by using the histograms from the reference object and from the tracked object region $R^{(1)}$ at t obtained in RANSAC as $\rho_t^{(1)} = \sum_{u=1}^{m} \sqrt{p_u^{t,(1)}(y, \Sigma)\, q_u^t}$, where $p_u^{t,(1)}$ and q_u^t are the uth bin of spatial kernel-weighted color histogram from $R^{(1)}$ of tracker-A, and of the reference object region, respectively. The kernel center $y = (y_1, y_2)$ and the anisotropic kernel bandwidth matrix Σ can be computed using the method described in Section 4.2. The Bhattacharyya coefficient $\rho_t^{(1)}$ is used to indicate whether or not the region $R^{(1)}$ is likely to contain occluding objects/background area, e.g. from partial occlusions or object intersections. If $\rho_t^{(1)}$ is small, indicating that the global appearance of object is significantly different from that of the reference object, then the dynamic maintenance is temporally frozen, so that no new feature points would be wrongly added. This is done as follows: If $\rho_t^{(1)} \leq T_F$ (T_F is a threshold determined empirically), then the maintenance process freezes, otherwise the maintenance process proceeds.

5.1.2 Maintenance of background feature point set $\mathcal{P}^B$

Online learning is also performed to the background set to maintain the dynamic background feature points. The background set contains feature points in between the large searching box and the candidate object region (see the area between the black and red rectangles in Fig.2). Comparing with the foreground case, the following differences exist for maintaining the background set:

The searching area and its re-initialization: The search area at t (see the black rectangle in Fig.2) is a rectangular area, that is larger than the tracked object region $R_{t-1}^{(1)}$. This is done by extending both the left and right side of $R_{t-1}^{(1)}$ by $k_x w^{(1)}$ ($w^{(1)}$ is the width of $R_{t-1}^{(1)}$, and k_x=0.5 in our tests), and extending both the top and bottom side of $R_{t-1}^{(1)}$ by $k_y h^{(1)}$ ($h^{(1)}$ is the height of $R_{t-1}^{(1)}$, and $k_y \in [0.1, 0.5]$ in our tests). This results in a searching area of $(2k_x + 1)w^{(1)}$ in width, and $(2k_y + 1)h^{(1)}$ in height. Correspondingly, the search area is re-initialized immediately after the re-initialization of the foreground object region $R_t^{(1)}$.

Shift foreground points to the background set: Those feature points in the foreground set that find their correspondences in the background set are re-assigned to the background set.

Adding and pruning new background points: New feature points that are found in the background region at the current frame t are added into this set. A maximum number L_B is then assigned to the background point set (L_B=1500 in our tests, empirically determined). Feature points in this set are sorted out according to their aging: If the total number of feature points exceeds L_B, then only the newest L_B feature points are kept while the remaining old aging points are removed.

5.2 Online learning of dynamic reference object appearance

Since the appearance of a target object may change in time, using a time-independent appearance distribution $q^t = q$ for the reference object may lead to tracking drift or tracking failures in the mean shift especially when the pdf of a visual object changes (e.g. significant changes in the object color distribution). Despite much research work in the mean shift-based object tracking, online learning of reference object pdf q remains an open issue. This is mainly due to the ambiguity on the change which could be caused either by object itself or by some background interferences (e.g. occlusions/intersections or background clutter) and the lack of occlusion priors. An efficient method for online learning of reference object pdf for the mean shift has so far been reported in (Khan & Gu,10).

The basic idea behind this online learning method is to *only* update the reference appearance distribution at those frames where reliable tracking without occlusion is indicated. Further, the updating frequency does not need to be very high, since object changes are usually gradual due to the mechanical movement. The online learning is done by using a criterion function and seeking the local maximum point that corresponds to good tracking performance in each individual frame interval of fixed length. Let $\rho_t = \sum_u \sqrt{q_u^{j-1} p_u^t}$ be the Bhattacharyya coefficient between the current tracked object from the final tracker and the reference object in the previous $(j-1)$th interval, and $\mathbf{x}_{t,i}$ be the four corners of the tracked region $R_t^{(obj)}$ from the final tracker. Noting that q_u^{j-1} implies that q_u^t is in the $(j-1)$th interval, $t \in [(j-2)S+1, (j-1)S]$, where S is the total frames in the interval (S is empirically determined depending on the motion speed and video frame rate, S=25 frames in our tests). If the following conditions are both satisfied:

$$\text{dist}_t = \sum_{i=1}^{4} \|\mathbf{x}_{t,i} - \mathbf{x}_{t-1,i}\|^2 < T_1, \quad \text{and} \quad \rho_t > T_2 \tag{17}$$

then the reference object appearance distribution q^j in the jth interval is updated

$$q^j = \kappa p^{t^*} + (1-\kappa) q^{j-1} \tag{18}$$

where $j = 1, 2, \cdots$, and t^* is the highest performance frame chosen from

$$t^* = \text{argmax}_{t \in [(j-1)S+1, jS]} \rho_t \tag{19}$$

and q^j is the updated reference object pdf in the jth interval that is related to the time interval $t \in [(j-1)S+1, jS]$, κ is the constant controlling the learning rate ($\kappa = 0.1$ in our tests), p^{t^*} is the appearance distribution of the candidate object where t^* is chosen from (19). If (17) is not satisfied, then the reference object distribution remains unchanged, i.e., $q^j \leftarrow q^{j-1}$. Key steps for updating q^j in the jth interval can be summarized as:

1. Check whether conditions in (17) are satisfied in $t \in [(j-1)S+1, jS]$;
2. If satisfied, updating q^j using (18) where the frame t^* is selected from (19);
3. Otherwise, freezing the update by assigning $q^j \leftarrow q^{j-1}$.

As an example, Fig.4 shows the two performance curves in (17) for the video "stair walking", where the thresholds T_1 and T_2 (blue dash line) and the updated frames (red dots) are marked. To demonstrate the effect of online learning, Fig.5 shows the results from the hybrid tracker with and without adding online learning to the reference object appearance. It shows that the improvement of tracking performance is most visible with the increase of video frame number. Since object appearance changes gradually in time, online learning of dynamic reference object distribution has indeed yielded visible improvement in tracking.

6. Hybrid tracker formulated from a criterion function

This section describes the formulation of hybrid tracker through combining the two baseline trackers under a given criterion function.

For the baseline tracker-A in Section 4.1, feature point correspondences are estimated by using spatiotemporal SIFT-RANSAC in the foreground and background regions. A tight rectangle

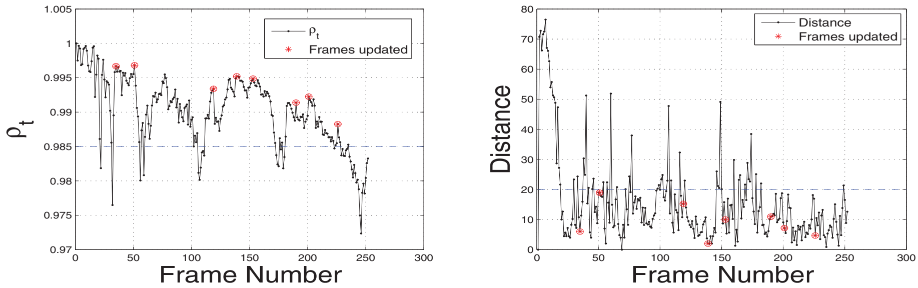


Fig. 4. Online learning of q^t for video "stair walking". Left: the curve of Bhattacharyya coefficient ρ_t in (17) vs. frame number, where the blue dash line is the threshold T_2, and the red dots are the frames updated. Right: the curve of distance of four corners $dist_t$ in (17) vs. frame number, where the blue dash line is T_1 and the corresponding updated frames are in red dots.

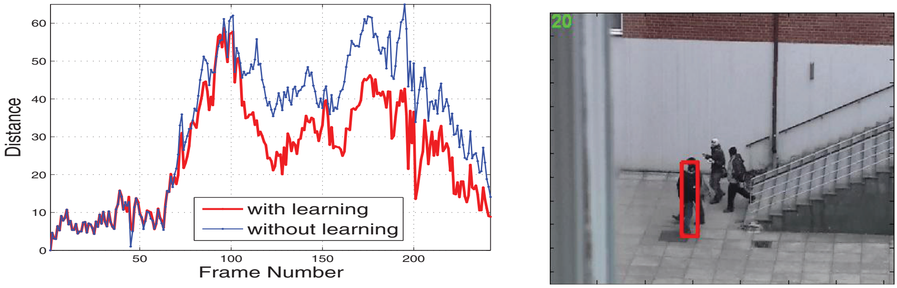


Fig. 5. Left: Tracking errors for the hybrid tracker as a function of video frame number. Distance d_1 in the curve is defined between the tracked and the ground-truth regions, according to (25). Red: with online learning; Blue: without online learning. Both curves are for the "stair walking" video; Right: an example frame of "stair walking".

surrounding the foreground points is then drawn as the candidate object region $R_t^{(1)}$ that is described by a parameter vector,

$$R_t^{(1)} : V_t^{(1)} = [y_c = (y_{1,c}^{(1)}, y_{2,c}^{(1)}), w^{(1)}, h^{(1)}, \theta^{(1)}, \mathcal{P}^F]_t^T$$

containing the 2D center position, width, height and orientation of the bounding box, and the foreground point set. For the baseline tracker-B in Section 4.2, an image region $R_t^{(2)}$ whose image content is most similar to the reference object appearance is sought by using the enhanced anisotropic mean shift with its kernel partially guided by the local feature points. This enhanced mean shift tracker generates a parameterized candidate region

$$R_t^{(2)} : V_t^{(2)} = [y = (y_1^{(2)}, y_2^{(2)}), w^{(2)}, h^{(2)}, \theta^{(2)}, \mathbf{h}_{rgb}^{(2)}]_t^T$$

A third candidate object region $R_t^{(3)}$ is then formed whose parameter vector is a weighted combination of the parameter vectors of the above two baseline trackers, i.e.,

$$R_t^{(3)} : \quad V_t^{(3)} = \sum_{i=1}^{2} \tilde{\rho}_t^{(i)} V_t^{(i)} \qquad (20)$$

where $\rho_t^{(i)}$ is the Bhattacharyya coefficient (defined in (23)), and $\tilde{\rho}_t^{(i)}$ is the normalized Bhattacharyya coefficients for the two baseline trackers,

$$\tilde{\rho}_t^{(i)} = \frac{\rho_t^{(i)}}{\rho_t^{(1)} + \rho_t^{(2)}} \tag{21}$$

For the final hybrid tracker, the parameter vector associated with the optimal target object region $R_t^{(obj)}$ is selected by maximizing the following criterion,

$$V_t^{(obj)} = \arg\max_{i:V_t^{(i)}} \left\{ \rho_t^{(i)}, i = 1, \cdots, 3 \right\} \tag{22}$$

where $\rho_t^{(i)}$, i=1,2,3, is the Bhattacharyya coefficient measuring the similarity between the reference object and the candidate object from the tracked candidate region $R_t^{(i)}$ at time t,

$$\rho_t^{(i)} = \sum_{u=1}^{m} \sqrt{q_u^t \, p_u^{t,(i)}} \tag{23}$$

$p_u^{t,(i)}$ is the uth bin of candidate object pdf estimate either from the baseline tracker-A (i=1) or from the baseline tracker-B (i=2), q_u^t is the uth bin of reference object pdf estimate. Noting that the superscript t in q_u^t indicates that the reference object pdf is dynamic. Table 1 summarizes the algorithm of the entire hybrid tracking scheme.

```
Initialization::
  Frame $t = 0$: mark a bounding box for the object and compute $q_0$;
For frame $t = 1, 2, \cdots$, do:
  1. Baseline tracker-A: Local feature correspondences by the spatiotemporal SIFT-RANSAC.
   1.1 Compute correspondence points by SIFT in the searching area;
   1.2 Find consensus points, estimate the transform, compute scores by RANSAC;
   1.3 Perform dynamic point maintenance to $\mathcal{P}_F$ and $\mathcal{P}_B$;
   1.4 Compute $V_t^{(1)}$, $R_t^{(1)}$, and $\rho_t^{(1)}$;
   1.5 If (3) is satisfied, re-initialize $V_t^{(1)}$, $R_t^{(1)}$, and $\tilde{\rho}_t^{(1)}$;
  2. Baseline tracker-B: Enhanced anisotropic mean shift:
   2.1 Using (12) to determine the initial $V^{(2)}$;
   2.2 Compute $\hat{y}^{(t)}$ using (8), and $\hat{\Sigma}^{(t)}$ using (9);
   2.3 Repeat Step 2.2 until convergence;
   2.4 Compute $\hat{w}_t^{(2)}$, $\hat{h}_t^{(2)}$, $\hat{\theta}_t^{(2)}$ from (11), and form $V_t^{(2)}$;
   2.5 Compute $\rho_t^{(2)}$ and form $R_t^{(2)}$;
   2.6 If (13) is satisfied, re-initialize $V_t^{(2)}$, $R_t^{(2)}$ and $\tilde{\rho}_t^{(2)}$;
  3. Compute the combined region parameters $V^{(3)}$ using (20);
  4. Determine $R_t^{(obj)}$ for the hybrid tracker according to (22);
  5. Online learning of object appearance pdf:
    If $mod(t, S) = 0$ (i.e., boundary of an interval), then online
    learning of $q^j$ using (18), if conditions in (17) are satisfied;
END (For)
```

Table 1. The algorithm for the hybrid tracking scheme

7. Application: Employ online learning in the joint mean shift and particle filter-based tracker

In this section, we show a new application example where the online learning approach in Section 5.2 is directly added to an existing joint anisotropic mean shift and particle filter-based (JMSPF) tracking scheme (Khan et al.,09), in order to further improve the tracking performance.
In the JMSPF tracking scheme, a particle filter is used to track the parameters of object shape (or, the bounding box of object), while the multi-mode anisotropic mean shift is embedded in the the particle filter through shifting its kernel location to the most similar object area and subsequently forming up the conditional probability based on the appearance distance metric. In such a way, PF weights are updated by using the likelihood obtained from the mean-shift, that re-distribute particles according to the distance metric through exploiting the most similar object appearance from the mean shift, rather than using the random sampling. This leads to more efficient utilizing of particles, hence a significantly reduction of the required number of particles: from $N_P = 800$ particles when the state vector contains both the shape and the appearance of object (Wang et al.,08), to N_P=15 particles in this JMSPF tracking scheme. Details of the JMSPF tracking scheme is referred to (Khan et al.,09).
Due to the lack of effective online learning methods, the JMSPF tracker in (Khan et al.,09) uses a time-invariant reference object appearance. Adding online learning of object appearance to the JMSPF tracker can be done by applying (18) in fixed-length frame intervals, with a small modification to include the superscript i for particles,

$$q^{j,i} = \kappa p^{t^*,i} + (1-\kappa)q^{j-1,i}, \quad i = 1, \cdots, N_P \tag{24}$$

if the both conditions in (17) are satisfied. Further, the re-initialization process in Section 4.2.4 can also be applied. A JMSPF tracking scheme after adding the online learning and re-initialization process can be shown schematically in Fig.6.

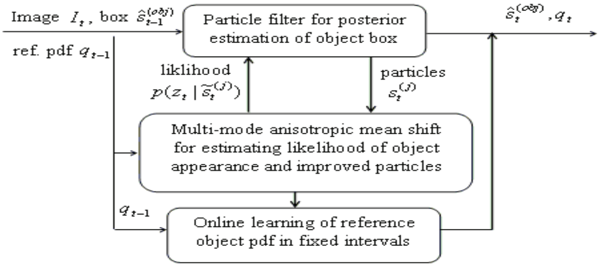


Image I_t, box $\hat{s}_{t-1}^{(obj)}$
ref. pdf q_{t-1}
Particle filter for posterior estimation of object box
$\hat{s}_t^{(obj)}, q_t$
liklihood $p(z_t \mid \tilde{s}_t^{(j)})$
particles $s_t^{(j)}$
Multi-mode anisotropic mean shift for estimating likelihood of object appearance and improved particles
q_{t-1}
Online learning of reference object pdf in fixed intervals

Fig. 6. Block diagram of the joint multi-mode anisotropic mean shift and particle filter-based tracking scheme (JMSPF) with online learning. Notations used in the block diagram: I_t: image frame at t; $\hat{s}_{t-1}^{(obj)}$ and $\hat{s}_t^{(obj)}$: tracked box parameters at (t-1) and t; $s_t^{(j)}$: jth particle at t; q_{t-1} and q_t: the estimated reference object pdf at (t-1) and t.

Fig.7 shows the tracking errors on the two videos "ThreePastShop2Cor" (CAVIAR Dataset) and "Pets2006_S07_C4" (PETS2006) with and without online learning. One can see that adding online learning to the scheme is able to further improve the tracking robustness and reduce the tracking drift in these cases.

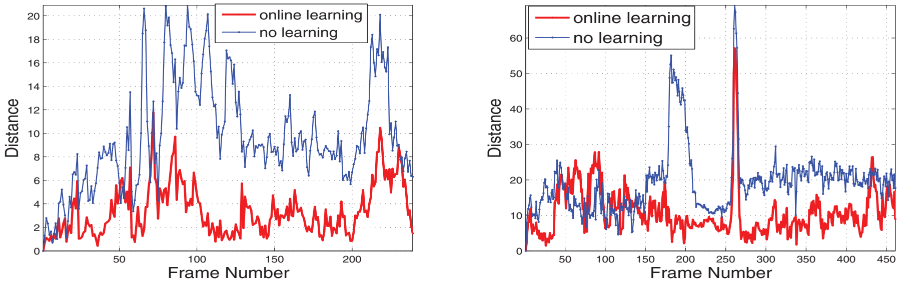


Fig. 7. Tracking errors for the JMSPF tracker: the distance d_1 in the curve is defined between the tracked and ground-truth regions according to (25). Red: with online learning; Blue: without online learning. Left: from the video "ThreePastShop2Cor"; Right: from the video "Pets2006_S07_C4". The ground truth boxes are manually marked. For "ThreePastShop2Cor", only the first 250 frames of ground truth boxes are marked and compared.

8. Experimental results and performance evaluation

The hybrid tracking scheme (summarized in Section 6) has been tested on numerous videos containing a range of complex scenarios. Our test results have shown that the hybrid tracking scheme is very robust and has yielded a marked improvement in terms of tracking drift, tightness and accuracy of tracked bounding boxes. This is especially obvious when complex scenarios in videos contain long-term partial occlusions, object intersections, severe object deformation, or cluttered background / background objects with similar color distributions to the foreground object.

8.1 Experimental setup

For testing the effectiveness of the hybrid tracking scheme, test videos that contain difficult scenarios in a range of complexities (e.g. long-term partial occlusion, object intersection, deformation or, pose changes) are selected. These videos are either captured from a dynamic or a static camera. In the tests, the initial bounding box is manually marked. Methods for automatic initial bounding box is beyond the scope of this chapter, readers can exploit other techniques, e.g. multiple hypothesis tests (Reid,79), active shape models or polygon vertices (Cootes et al.,01). For the mean shift tracking, a $32 \times 32 \times 32$ bin histogram is used for the RGB color images. The maximum number of iterations is 10 for the enhanced mean shift for all videos and is determined empirically. Table 2 summarizes the thresholds used for re-initialization thresholds as well as the γ values for normalizing the kernel bandwidth matrix of the mean shift in the hybrid tracker. $(T_1^{(2)}, T_2^{(2)})$ in (12) are set to (10, 0.95) in all cases. Further, Table 3 summarizes the online learning thresholds used for the hybrid tracker and the improved JMSPF tracker.

8.2 Qualitative evaluation and comparison of tracking results

The hybrid tracking scheme has been tested on numerous videos that contain a variety of difficult tracking scenarios. Fig.8 shows the tracking results (key video frames) from the hybrid scheme (marked by red solid line rectangles) on 5 videos. In all cases, online learning is included in the hybrid tracker.

Video	in baseline tracker-A		in baseline tracker-B		
	$T_1^{(1)}$	$T_2^{(1)}$	$T_3^{(2)}$	$T_4^{(2)}$	γ
walking lady	50	0.73	30	0.950	0.33
OneShopOneWait2Cor	300	0.65	15	0.975	0.33
ThreePastShop2Cor	60	0.98	15	0.975	0.33
Pets2006_S7_C4	150	0.87	12	0.975	0.31
Pets2007_DS5_C1	100	0.88	15	0.975	0.23

Table 2. Parameters in the hybrid tracker: re-initializing thresholds in (3) and (13), and γ-normalization in (8).

Video	Hybrid tracker		JMSPF tracker	
	T_1	T_2	T_1	T_2
walking lady	10	0.95	50	0.90
OneShopOneWait2Cor	30	0.95	20	0.90
ThreePastShop2Cor	30	0.96	15	0.96
Pets2006_S7_C4	30	0.95	20	0.95
Pets2007_DS5_C1	30	0.95	20	0.90

Table 3. Online learning thresholds in (17) for the hybrid tracker and the JMSPF tracker.

The video "walking lady" captured from a moving camera contains several long-term partial occlusions when the lady walks behind cars. Further, colors from a part of the object sometimes appear to be similar to the occluding car.

The video "OneShopOneWait2Cor" is downloaded from the CAVIAR dataset (CAVIAR Dataset). The selected target object is a walking man with dark colored clothes. During the course of walking, there is intersection where another man partially occludes the target man, also there are pose changes while the man is waiting, and scale changes during the course of walking.

The video "ThreePastShop2Cor" is also from the CAVIAR dataset (CAVIAR Dataset). In the video, the selected target (a man wearing a red coat with a backpack) walks in parallel with two other persons before intersecting with one of them by suddenly changing his walking direction. The man continues his walking and passes another nearby person with a red coat coming from the opposite direction. After a while, several other intersections appear when the man walks continuously away from the camera (depth changes).

The video "Pets2006_S7_C4" is from the Pets 2006 dataset (PETS2006), named "Dataset S7 (Take 6-B)" by the camera 4. The selected target object is a walking man with dark clothes. During the course of walking, there are several intersections with partial occlusions, pose changes. The man also passes over other walking persons with similar color clothes.

The video "Pets2007_S05_C1" is from the Pets 2007 dataset (PETS2007), named "Dataset S5" from the camera one. The video ia probably captured around a check-in desk in an airport, where there are many walking persons. The selected target object is a man with white shirt carrying a backpack. Tracking this single target through the crowds (containing around 20 persons where some are moving, some stand still) is a rather challenging task, as there are many and frequent partial occlusions, intersections and pose changes.

The aim of these tests is to qualitatively evaluate the robustness of the hybrid tracking scheme, especially in video scenes containing long term partial occlusions, object intersections, deformations and fast motion, cluttered background or background object.

Comparisons: Comparisons are made with three state-of-the-art methods that are closely-related to the hybrid tracker described in this chapter. They are:

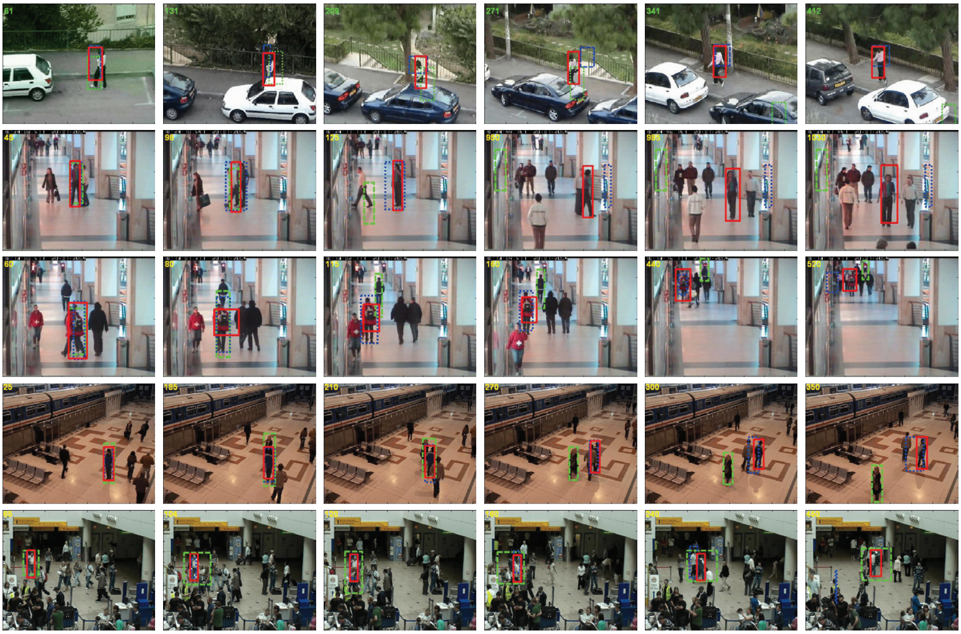
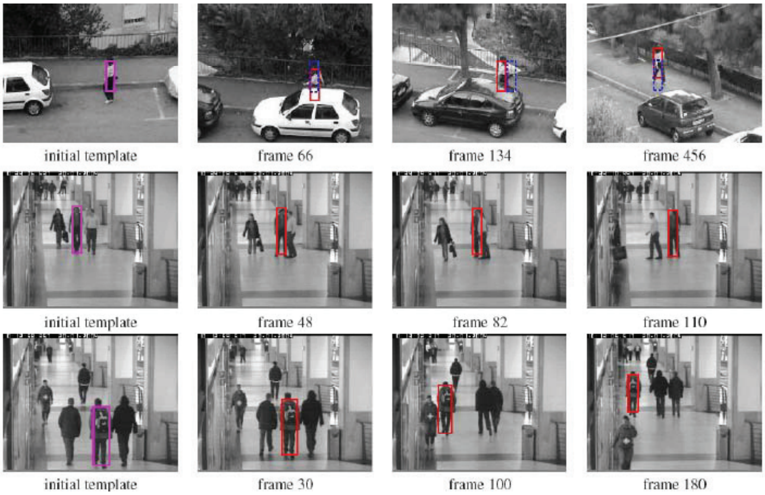


Fig. 8. Comparing tracking results from 3 trackers: the hybrid tracker (red solid line box), *Tracker-1* (green short dash line); *Tracker-2* (blue long dash line). From rows 1-6: results for videos (key frames) "walking lady", "OneShopOneWait2Cor", "ThreePastShop2Cor", "Pets2006_S5_C4" and "Pets2007_DS5_C1".

- *Tracker-1:* an anisotropic mean shift tracker in (Sumin & Xianwu,08) that is formed entirely based on using global object modeling.
- *Tracker-2:* a spatiotemporal SIFT-RANSAC tracker that is entirely based on using local feature points.
- *Tracker-3:* a fragments-based tracker that uses integral histograms (Adam et al.,06).

Since we do not have the program code of *Tracker-3*, comparison is made by using the same videos shown in the *Tracker-3* (Adam et al.,06). Three videos (rows 1-3 in Fig.8) in (Adam et al.,06) are found by the Internet search, and are used for comparisons with *Tracker-3*. Fig.8 shows the tracking results from the three tracking methods: the hybrid tracker (marked in red solid line boxes), *Tracker-1* (marked in green short dash line boxes), and *Tracker-2* (marked in blue long dash line boxes). Observing the tracked results in Fig.8, the hybrid tracker is shown to be very robust with tightly tracked bounding boxes and without tracking drift. Comparing the results from the hybrid tracker and the two existing *Tracker-1* and *Tracker-2*, the hybrid tracking scheme is shown to be much more robust with marked improvement, especially in difficult video scenarios that contain long partial occlusions, object intersects, fast object motions, nearby persons/cluttered background with similar color distributions and shape. For the video "Pets2007_S05_C1", the hybrid tracker has eventually failed after 490 frames as the scenes contain too many persons (around 15) with high frequency of partial occlusions. Comparing with *Tracker-1* and *Tracker-2*, the two trackers have failed in about 400 and 240 frames, respectively, while the hybrid tracker has managed to track the object somewhat longer.

Fig.9 shows the tracking results in (Adam et al.,06) (referred to as: *Tracker-3*). Comparing the tracking results (marked in red box) shown in Fig.9 and the tracking results in Fig.8 (rows 1-3, marked in red box), one can see that the two trackers have somewhat similar tracking performance in these 3 videos, both have tracked the target object rather well. Comparisons using more complex videos, e.g., "Pets2007_DS5_C1" would probably be able to distinguish the performance differences of these 2 trackers, however, no tests are made as this would require to run the program of (Adam et al.,06).



initial template, frame 66, frame 134, frame 456

initial template, frame 48, frame 82, frame 110

initial template, frame 30, frame 100, frame 180

Fig. 9. Results from *Tracker-3* (courtesy from (Adam et al.,06)): results from *Tracker-3* (Red); manually selected target (Pink). Top to bottom: frames from the videos "walking lady", "OneShopOneWait2Cor" and "ThreePastShop2Cor".

8.3 Quantitative evaluation and comparisons of performance

To quantitatively evaluate and compare the performance of the hybrid tracker and the two existing trackers (*Tracker-1* and *Tracker-2*), three distance metrics are used.

8.3.1 Distance metrics

The Euclidian distance: is defined between the four corners of the tracked object bounding box and the manually marked ground truth box as follows,

$$d_1 = \frac{1}{4}\sum_{i=1}^{4}\sqrt{(x_{i,1}-x_{i,1}^{GT})^2+(x_{i,2}-x_{i,2}^{GT})^2} \qquad (25)$$

where $(x_{i,1}, x_{i,2})$ and $(x_{i,}^{GT}, x_{i,2}^{GT})$, $i = 1, \cdots, 4$, are the corners of rectangular box from the final hybrid tracker and the manually marked Ground Truth (GT), respectively.

The MSE (Mean Square Error): is defined between the 5 parameters (2D center, width, height and orientation) of tracked object box and the manually marked Ground Truth (GT) object bounding box over all frames in each video,

$$MSE = \frac{1}{N}\sum_{t=1}^{N}\sqrt{\left(v_i^t - v_i^{t,GT}\right)^2} \tag{26}$$

where v_i^t is the ith parameter of a tracked box at t, $v_i^{t,GT}$ is the ith ground truth parameter at t, N is the total number of frames in the video.

The Bhattacharyya distance: is defined between the tracked object box and the reference object box as follows:

$$d_2 = \sqrt{1 - \sum_u \rho(p_u, q_u)} \tag{27}$$

where u is the index of histogram bin. Under this criterion, good performance is indicated by small d_2 values. The average Bhattacharyya distance $\bar{d}_2$ is computed by averaging the Bhattacharyya distances over all frames in each video.

In the first row of Fig.10, we compare the tracking errors for the 3 trackers (the hybrid tracker, *Tracker-1* and *Tracker-2*), in terms of the Euclidian distance d_1 (in (25)) between the tracked box and the ground truth box as a function of image frames, on the video "face" and "walking lady". Comparing the results from the two videos, the hybrid tracker has clearly shown better performance than those from the *Tracker-1* and *Tracker-2* in these cases. In the 2nd row of Fig.10, we compare the hybrid tracker and the JMSPF tracker with online learning. Comparing the results from the two videos, the JMSPF tracker seems better in "ThreePastShop2Cor" and slightly worse in "walking lady" to that obtained from the hybrid tracker. The performance of these two methods varies depending on the test videos.

Table 4 shows the tracking errors (the MSEs defined in (26)) for the four trackers: the hybrid tracker, the JMSPF tracker, *Tracker-1*, and *Tracker-2*. Comparing the results in the table, the hybrid tracker and JMSPF tracker have shown clearly better performance than those from the two existing *Tracker-1* and *Tracker-2*. Further, the JMSPF tracker is shown to be much better than that of the hybrid tracker on the video "ThreePastShop2Corthe" and slightly worse on the video "walking lady".

Video	Box Parameters	Hybrid tracker	JMSPF tracker	*Tracker-1*	*Tracker-2*
walking lady	x-position	1.6851	2.9454	51.575	19.854
	y-position	1.6020	4.7661	66.222	10.357
	width w	0.8935	1.0221	4.7926	3.3857
	height h	1.1682	2.6063	2.5973	55.973
	θ (in radius)	0.0011	0.0047	0.0627	0.0123
ThreePastShop2Cor	x-position	2.8815	0.9940	23.835	4.8997
	y-position	3.3784	2.7138	37.878	7.6360
	width w	1.9675	1.0327	5.8228	2.4200
	height h	16.836	2.1415	9.8112	4.3763
	θ (in radius)	0.0067	0.0001	0.0023	0.0012

Table 4. Tracking errors, the MSE defined in (26), for 4 different trackers.

Table 5 shows the tracking errors, the average Bhattacharyya distances d_2 in (27), for the four trackers: the hybrid tracker, the JMSPF tracker, *Tracker-1* and *Tracker-2*.

Video	Hybrid Tracker	JMSPF Tracker	*Tracker-1*	*Tracker-2*
walking lady	0.2076	0.2652	0.3123	0.3654
OneShopOneWait2Cor	0.1466	0.2037	0.4549	0.5240
ThreePastShop2Cor	0.1580	0.1444	0.3241	0.2861
Pets2006_S7_C4	0.1007	0.2267	0.2473	0.2032
Pets2007_DS5_C1	0.1455	0.2133	0.2840	0.2370

Table 5. Tracking errors, the average Bhattacharyya distances $\bar{d}_2$ in (27), for 4 different trackers. The smaller the d_2, the better the performance.

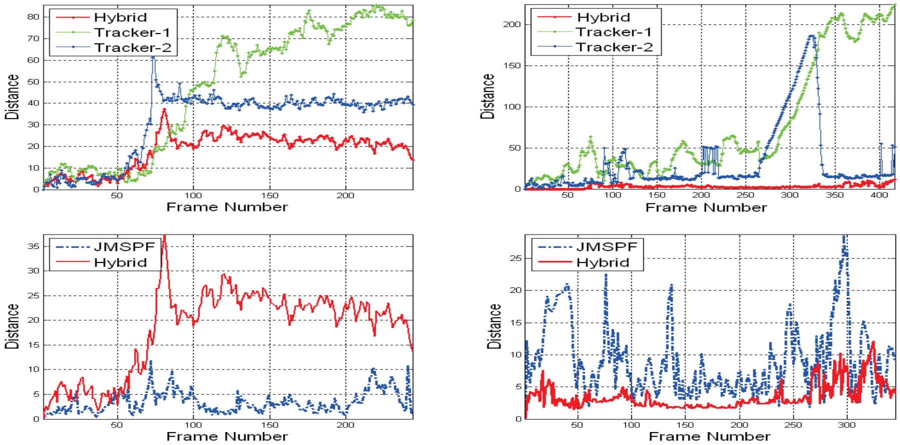


Fig. 10. Comparison of tracking errors (the Euclidian distance d_1 in (25)) on the videos "ThreePastShop2Cor" (column 1) and "walking lady" (column 2). 1st row: comparison among 3 trackers: hybrid tracker (red), *Tracker-1* (blu) and *Tracker-2* (green); 2nd row: comparison between hybrid tracker (red solid) and JMSPF tracker (blue dash). Noting the scale difference in vertical axis for "walking lady" in the 2nd column.

8.4 Computational cost

To give an indication on the computational cost, the execution times are recorded for four tracking methods: the hybrid tracker (summarized in Section 6), *Tracker-1*, *Tracker-2*, and JMSPF tracker (in Section 7). Table 6 shows the average time (in Second) required for tracking one object in one video frame, where the average is done over all frames in each video. Noting that tracking time varies dependent on the complexity of video scenes. All these tracking schemes are implemented by Matlab programs, and run on a PC with a Intel Pentium Dual 2.00 GHz processor. Observing Table 6 one can see that the hybrid tracker requires

Video	Hybrid tracker	*Tracker-1*	*Tracker-2*	JMSPF tracker
	(in sec)	(in sec)	(in sec)	(in sec)
OneShopOneWait2Cor	0.1498	0.0934	0.049	1.4356
ThreePastShop2Cor	0.1501	0.0954	0.053	1.3945
Pets2006_S7_C4	0.1352	0.0935	0.041	1.5423
Pets2007_DS5_C1	0.1489	0.0845	0.050	0.9870

Table 6. Average required time to track a visual object in one video frame, for 4 different visual trackers. All programs are implemented in Matlab without optimizing the program codes.

more computations as comparing with *Tracker-1* or *Tracker-2*, as the result of combining baseline trackers, adding online learning and computing the criterion in order to make the final tracking. Despite this, the hybrid tracker achieves an average tracking speed of 10 frames/second using the Matlab program. One may also observe that the JMSPF tracker requires rather heavy computations, approximately 10 times of that required by the hybrid tracker.

8.5 Comparison between the hybrid tracker and the JMSPF tracker

Both tracking schemes, the hybrid tracker (summarized in Section 6) and the JMSPF tracker (in Section 7) are shown to be very robust in tracking visual objects. Fig.11 shows some results of tracked video frames (key frames are selected) on 5 videos from these two methods. Qualitatively evaluation of these tracking results through visual comparisons, both the hybrid tracker and the JMSPF tracker are shown to be very robust. From the general visual impression, the JMSPF tracker has a slightly better performance in terms of the tightness and the accuracy of the bounding box in some frames. For the video "Pets2007_S05_C1", similar to that in the hybrid tracker, the JMSPF tracker has eventually failed after about 490 frames as the scenes contain too many persons with high frequency of partial occlusions. Quantitatively evaluations of the performance by comparing d_2 values (defined in (27)) in Table 5 (columns 1 and 2) and d_1 values (defined in (25)) in Fig.10 (the right sub-figure), and comparing the computational speed in Table 6 (columns 1 and 4), show that the hybrid tracker has slightly smaller (average) d_2 values and a much fast computational speed (about 10 times faster) on the tested videos. While d_1 values in the two trackers vary depending on the videos. Overall, the hybrid tracker seems a more attractive choice, as the tradeoff between the average performance, tracking robustness and computational speed.

8.6 Limitations

Despite very robust tracking performance from the hybrid tracking scheme, several weak points are observed from the experiments. (a) If a target object in the video experiences a long-duration partial occlusions over a large percentage of area (e.g.$>$60%), then the tracking performance can be degraded, especially if the visible part is rather smooth and lacks of local feature points. (b) For images contain a relatively large object, e.g., a face, large pose changes could potentially cause tracking degradation. This is probably due to the complexity of face movement (many possible local motions) and the use of pdf as the face appearance model (that may not be the best choice). Improvement through using object-type-specific appearance models and feature point correspondences under multiple local motion models could be considered. (c) When full object occlusion occurs. Although our tests occasionally contain a few frames of full occlusion, it causes the tracker temporally frozen or tracking failure, however, the tracking is able to immediately recover or resume tracking soon after the partial appearance of the object. In principle, full occlusions with a long duration is beyond the limit of this scheme. The problem may be better tackled by trackers using videos from multiple cameras.

9. Conclusion

A novel hybrid visual tracking scheme is presented in this chapter, which jointly exploits local features and global appearances and shape of dynamic objects. The hybrid tracker is formulated using a criterion function that optimally combines the results from two baseline trackers: the spatiotemporal SIFT-RANSAC and the enhanced anisotropic mean shift.

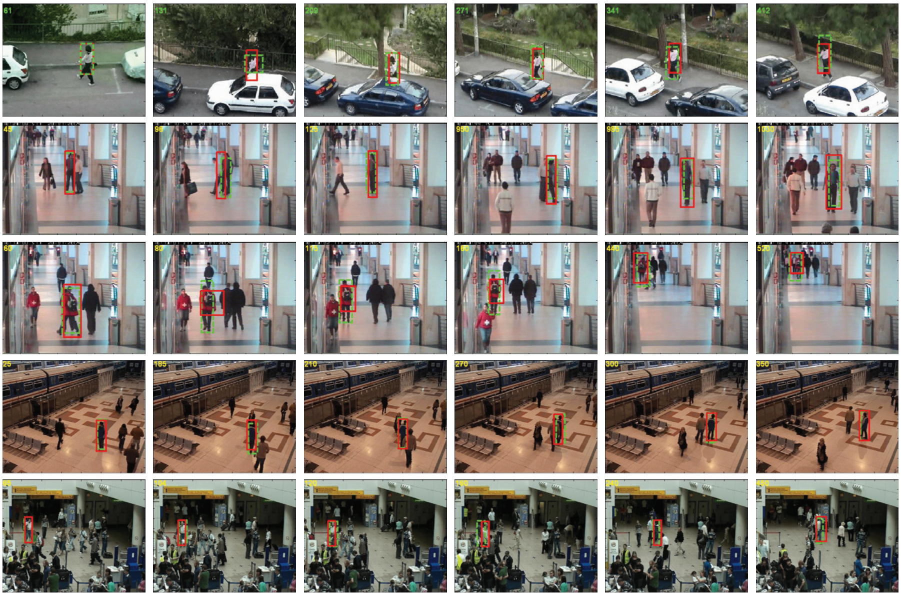


Fig. 11. Comparing tracking results from 2 trackers: the hybrid tracker (red solid line box) and the JMSPF tracker with online learning (green dash line box). From rows 1-5: result frames (key frames) from videos "walking lady", "OneShopOneWait2Cor", "ThreePastShop2Cor", "Pets2006_S5_C4" and "Pets2007_DS5_C1".

Online learning of dynamic object is introduced to the global object appearance and local object feature points separately: For object appearances, online learning of the appearance distribution of reference object is performed in each fixed-length frame interval where the ambiguity between the object change and the change due to partial occlusions is addressed. For object feature points, online maintenance of two feature point sets (foreground and background) is performed in each video frame, where the background set is used as priors on the occlusion. It is worth noting that the online maintenance of feature point sets is a key step for the realization of spatiotemporal SIFT-RANSAC. It is also worth mentioning that the enhanced mean shift, by allowing the kernel position partially guided by local feature points, significantly reduces the mean shift sensitivity to similar color distributed background/other objects.

Experimental results on numerous videos with a range of complexities have shown that the hybrid tracker has yielded very robust tracking performance. This is especially evident when tracking objects through complex scenarios, for example, video scenes where the target object experiences long-term partial occlusions or intersections from other objects, large object deformations, abrupt motion changes of object, dynamic cluttered background/occluding objects having similar color distributions to the target object. Results of quantitative and qualitative evaluations and comparisons of the hybrid tracker and the two existing tracking methods, (*Tracker-1* and *Tracker-2*), have shown a marked tracking improvement from the hybrid tracker, in terms of reduced tracking drift and improved tightness of tracked object bounding box. Comparisons by visual inspecting the tracking results of 3 videos from the hybrid tracker and from (Adam et al.,06) have shown that both trackers perform rather well in these cases. Further comparisons on complex video scenarios requires to run the program from (Adam et al.,06) and hence not performed. Comparisons of the hybrid tracker and the JMSPF tracker with online learning (in Section 7) have shown that latter has rather similar however occasionally slightly better performance but at the cost of significantly increase in computations (approximately 10 times). Comparisons of hybrid tracker with and without online learning have shown that adding online learning has significantly reduced the tracking drift especially for long video sequences. Overall, the hybrid tracking scheme is shown to be very robust and yielded marked improvements over the existing trackers (*Tracker-1* and *Tracker-2*). Comparing with the JMSPF tracker, the hybrid tracker provides a better tradeoff between the tracking robustness and tracking speed (≈ 10 frames/second in our Matlab program).

10. References

[Adam et al.,06] Adam A, Rivlin E, Shimshoni I (2006), "Robust Fragments-based Tracking using the Integral Histogram", vol.1, pp.798-805, in Proc.*IEEE int'l conf CVPR*.

[Bay et al.,06] Bay H., Tuytelaars T. & Gool L.V.(2006), "SURF: Speeded Up Robust Features", in proc. *European conf. ECCV*, pp.404-417.

[Bauer et al.,07] Bauer J, Sunderhauf N & Protzel P (2007), "Comparing Several Implementations of Two Recently Published Feature Detectors", in proc. *Int'l Conf. Intelligent and Autonomous Systems, Toulouse, France*.

[Bar-Shalom & Fortmann,98] Bar-Shalom Y. and Fortmann T. (1998), Tracking and Data Association. New York: Academic.

[Battiato et al.,07] Battiato S., Gallo G., Puglisi G. & Scellato S. (2007), "SIFT Features Tracking for Video Stabilization", in *Proc of Int'l Conf Image Analysis and Processing*, pp. 825-830.

[Bretzner & Lindeberg,98] Bretzner L. & Lindeberg T. (1998), "Feature Tracking with Automatic Selection of Spatial Scales", in proc. *Comp. Vision and Image Understanding*, vol. 71, pp. 385-392.

[CAVIAR Dataset] http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/

[Chen et al.,08] Chen A, Zhu M, Wang Y & Xue C. (2008), "Mean shift tracking combining SIFT", in proc. *Int'l conf. Audio, Language and Image Proc.*, pp.1532-1535.

[Cootes et al.,01] Cootes TF, Edwards GJ, Taylor CJ (2001), "Active appearance models", *IEEE trans. TPAMI*, vol.23, no.6, pp.681Ű685.

[Collins,03] Collins R.T.(2003), "Mean-shift blob tracking through scale space", in proc. *IEEE Int'l conf. CVPR'03*, vol. 2, pp. 234-240.

[Comaniciu et al.,03] Comaniciu D., Ramesh V. & Meer P. (2003), "Kernel-based object tracking", *IEEE Trans Pattern Analysis and Machine Intelligence*, Vol.5, pp.564-577.

[Deguchi et al.,04] Deguchi K., Kawanaka O. & Okatani T. (2004), "Object tracking by the mean-shift of regional color distribution combined with the particle-filter algorithm", in proc. *Int'l conf. ICPR*, vol. 3, pp. 506-509.

[Fischler & Bolles,81] Fischler MA & Bolles RC (1981), "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartographys", in *Communications of the ACM*, vol. 24, pp. 381-395.

[Gordon et al.,01] Gordon N.J., Doucet A. and Freitas N.D. (2001), "Sequential Monte Carlo Methods in Practice", *New York: Springer*.

[Gordon,00] Gordon N.J., Doucet A. and de Freitas N. (2000), "On sequential monte carlo sampling methods for Bayesian filtering", *Statistics and Computing*, vol. 10, pp. 197-208.

[Haner & Gu,10] Haner S and Gu IYH (2010), "Combining Foreground / Background Feature Points and Anisotropic Mean Shift For Enhanced Visual Object Tracking", in proc. *IEEE Int'l conf. ICPR*, 23-26 August, Istanbul, Turkey.

[Harris & Stephens,88] Harris C. & Stephens M.(1988), "A Combined Corner and Edge Detector", in *Proc. 4th Alvey Vision Conf., Manchester*, pp.147-151.

[Hager et al.,04] Hager G.D., Dewan M., Stewart C.V. (2004), ŤMultiple Kernel Tracking with SSD,Ť vol. 1, pp.790-797, in proc. *IEEE Int'l conf. CVPR'04*.

[Khan & Gu,10] Khan, Z.H.; Gu, I.Y.H.(2010), "Joint Feature Correspondences and Appearance Similarity for Robust Visual Object Tracking", *IEEE Trans. Information Forensics and Security*, Vol.5, No. 3, pp. 591-606.

[Khan et al.,09] Khan ZH, Gu IYH, Backhouse AG (2010), "Robust Visual Object Tracking using Multi-Mode Anisotropic Mean Shift and Particle Filters", to appear, IEEE trans. Circuits and Systems for Video Technology.

[Khalid et al.,05] Khalid M.S., Ilyas M.U., Mahmoo K., Sarfaraz M.S., Malik M.B.(2005), "Kullback-Leiber divergence measure in correlation of gray-scale objects", in Proc. *2nd IntŠl conf. on innovations in Information Technology (IIT05)*.

[Li et al,06] Li. Y., Yang J., Wu R. & Gong F. (2006), "Efficient Object Tracking Based on Local Invariant Features", in *Proc. of Int. Symposium on Comm. and Information Technologies (ISCIT)*, pp. 697-700.

[Li et al.,08] Li L, Huang W, Gu IYH, Luo R & Tian Q (2008), "An efficient sequential approach to tracking multiple objects through crowds for real-time intelligent CCTV systems", *IEEE trans. Systems, Man and Cybernetics*, part B, vol.38, No.5, pp.1254-1269.

[Lim et al.,04] Lim J, Ross D,Lin R-S, Yang M-H (2004), "Incremental learning for visual tracking", in Proc. *Int'l conf NIPS*.

[Lowe,04] Lowe D.G (2004), "Distinctive Image Features from Scale-Invariant Keypoints", *Int. Journal of Computer Vision*, vol.60, pp. 91-110.

[Maggio & Cavallaro,05] Maggio E. & Cavallaro A. (2005), "Multi-part target representation for colortracking", in proc. *IEEE Int'l conf. ICIP*, pp.729-732.

[Mondragon wt al.,07] Mondragon I.F., Campoy P., Correa J. F. , & Mejias L.(2007), "Visual Model Feature Tracking For UAV Control", in *Proc. Int'l Symposium Intelligent Signal Processing*, pp.1-6.

[Okuma et al.,04] Okuma K., Taleghani A., Freitas N., Little J.J. & Lowe D.G. (2004), "A boosted particle filter: multitarget detection and tracking", in Proc. *Int'l conf. ECCV*, pp.28-39.

[PETS2006] http://www.cvg.rdg.ac.uk/PETS2006/data.html

[PETS2007] http://pets2007.net/

[Parameswaran et al.,07] Parameswaran V, Ramesh V & Zoghlami I (2006), "Tunable kernels for racking", in proc. *IEEE Int'l Conf. CVPR*, pp.2179-2186.

[Reid,79] Reid D.B. (1979), "An algorithm for tracking multiple targets", *IEEE Trans. Autom. Control*, vol. 24, no. 2, pp. 843-854.

[Rosales & Sclaroff,99] Rosales R. and Sclaroff S. (1999), "3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions", in *proc. IEEE Int'l Conf. CVPR*, pp. 117-123.

[Sankaranarayanan et al.,08] Sankaranarayanan A.C., Veeraraghavan A., Chellappa R. (2008), "Object Detection, Tracking and Recognition for Multiple Smart Cameras", *Proc. of the IEEE*, Vol.96, No.10, pp. 1606-1624.

[Shi & Tomasi,94] Shi J. & Tomasi C. (2008), "Good features to track", in proc. *IEEE Int'l conf. CVPR*, pp. 593-600.

[Strandmark & Gu,09] Strandmark P. & Gu I.Y.H. (2009), "Joint Random Sample Consensus and Multiple Motion Models for Robust Video Tracking", in Springer *LNCS* Vol. 5575, pp. 450-459.

[Skrypnyk & Lowe,04] Skrypnyk I.& Lowe D.G.(2004), "Scene modelling, recognition and tracking with invariant image features", in *Proc. Int. Symposium Mixed and Augmented Reality (ISMAR)*, pp. 110-119, 2004.

[Sumin & Xianwu,08] Sumin Q. & Xianwu H. (2008), "Hand tracking and gesture gecogniton by anisotropic kernel mean shift", in proc. *IEEE Int'l. conf. NNSP*, vol. 25, pp. 581-585.

[Vermaak et al.,03] Vermaak J., Doucet A., Perez P. (2003), "Maintaining multimodality through mixture tracking", in Proc. *IEEE Int'l conf. ICCV*, pp.1110-1116.

[Wang et al.,07] Wang T, Gu IYH, Shi P (2007), "Object tracking using incremental 2D-PCA learning and ML estimation", in proc. *IEEE int'l conf. ICASSP*.

[Wang et al.,08] Wang T., Backhouse A.G., & Gu I.Y.H. (2008), "Online subspace learning on Grassmann manifold for moving object tracking in video", in proc.*IEEE int'l conf. ICASSP*.

[Wang et al.,08] Wang T., Gu I.Y.H., Backhouse A.G. and Shi P. (2008), "Face Tracking Using Rao-Blackwellized Particle Filter and Pose-Dependent Probabilistic PCA", in proc. *IEEE int'l conf ICIP*, San Diego, USA, Oct. 12-15.

[Welch & Bishop,97] Welch G. and Bishop G. (1997), "Scaat: incremental tracking with incomplete information", in proc. *24th Annual Conf. Comp. Graphics & Interactive Techniques*.

[Wu et al.,08] Wu P, Kong L, Zhao F & Li X(2008), "Particle filter tracking based on color and SIFT features", in Proc. *IEEE int'l conf Audio, Language and Image Processing*, pp. 932-937.

[Xu et al,08] Tu Q., Xu Y., & Zhou M. (2008), "Robust vehicle tracking based on Scale Invariant Feature Transform", in proc. *Int. Conf. Information and Automation (ICIA)*, pp. 86-90.

[Xu et al.,05] Xu D, Wang Y & An J (2005), "Applying a new spatial color histogram in mean-shift based tracking algorithm", in proc. *Int'l conf. Image and Vision Comp.* New Zealand.

[Yang et al.,04] Yang J, Zhang D, Frangi AF, Yang J-Y (2004), "Two-dimensional PCA: a new approach to appearance-based face representation and recognition", *IEEE Trans. PAMI*, vol.26, no.1, pp.131-137.

[Yilmaz,07] Yilmaz A., "Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection", in Proc. *IEEE conf. CVPR'07*.

[Yilmaz et al.,06] Yilmaz A., Javed O. and Shah M.(2006), "Object tracking: A survey", *ACM Computing Surveys*, vol. 38, no. 4.

[Zhao et al.,08] Zhao C, Knight A & Reid I (2008), "Target tracking using mean-shift and affine structure", in Proc. *IEEE Int'l conf. ICPR*, pp.1-5.

[Zhou et al.,08] Zhou H., Yuan Y., Shi C.(2008), "Kernel-Based method for tracking objects with rotation and translation", *Int. Journal Computer Vision*.

[Zivkovic & Krose,04] Zivkovic Z. & Krose B. (2004), "An EM-like algorithm for color-histogram-based object tracking", in proc. *IEEE Int'l conf. CVPR*, vol.1, pp. I-798-808.

6

Switching Local and Covariance Matching for Efficient Object Tracking

Junqiu Wang and Yasushi Yagi
Osaka University
Japan

1. Introduction

Object tracking in video sequences is challenging under uncontrolled conditions. Tracking algorithms have to estimate the states of the targets when variations of background and foreground exist, occlusions happen, or appearance contrast becomes low. Trackers need to be efficient and can track variant targets. Target representation, similarity measure and localization strategy are essential components of most trackers. The selection of components leads to different tracking performance.

The mean-shift algorithm Comaniciu et al. (2003) is a non-parametric density gradient estimator which finds local maxima of a similarity measure between the color histograms (or kernel density estimations) of the model and the candidates in the image. The mean-shift algorithm is very fast due to its searching strategy. However, it is prone to failure in detecting the target when the motion of the target is large or when occlusions exist since only local searching is carried out.

The covariance tracker Porikli et al. (2006) represents targets using covariance matrices. The covariance matrices fuse multiple features in a natural way. They capture both spatial and statistical properties of objects using a low dimensional representation. To localize targets, the covariance tracker searches all the regions; and the region with the highest similarity to the target model is taken as the estimation result. The covariance tracker does not make any assumption on the motion. It can compare any regions without being restricted to a constant window size. Unfortunately, the Riemannian metrics adopted in Porikli et al. (2006) are complicated and expensive. Since it uses a global searching strategy, it has to compute distances between the covariance matrices of the model and all candidate regions. Although an integral image based algorithm that requires constant time is proposed to improve the speed, it is still not quick enough for real time tracking. It is difficult for the covariance tracker to track articulated objects since computing covariance matrices for articulated objects is very expensive.

In this work, we propose a tracking strategy that switches between local tracking and global covariance tracking. The switching criteria are determined by the tracking condition. Local tracking is carried out when the target does not have large motion. When large motion or occlusions happen, covariance tracking is adopted to deal with the issue. The switching between local and covariance matching makes the tracking efficient. Moreover, it can deal with sudden motions, distractions, and occlusions in an elegant way. We compute covariance matrices only on those pixels that are classified as foreground. Therefore we can track articulated objects.

To speed up the global searching process, we use Log-Euclidean metrics Arsigny et al. (2005) instead of the Riemannian invariant metrics Pennec et al. (2006); Porikli et al. (2006) to measure the similarity between covariance matrices. The model update in covariance tracking Porikli et al. (2006) is also expensive. We update the model by computing the geometric mean of covariance matrices based on Log-Euclidean metrics. The computation is simply Euclidean in the logarithmic domain, which reduces the computational costs. The final geometric mean is computed by mapping back to the Riemannian domain with the exponential. Log-Euclidean metrics provide results similar to their Riemannian affine invariant equivalent but takes much less time.

We arrange this chapter as follows. After a brief review of previous works in Section 2, we introduce the local tracking method based on foreground likelihood computation in Section 3. In specific, we discuss target representation for local tracking using color and shape texture information in Section 3.1; we describe our feature selection for local tracking in Section 3.2, and our target localization strategy for local tracking in Section 3.3. In Section 4, we apply Log-Euclidean metric in covariance tracking. We introduce a few basic concepts that are important for our covariance matching in Section 4.1. The extended covariance matching method using Log-Euclidean metric is described in Section 4.2. In Section 5, we give the switching criteria for the local and global tracking. Experimental results are given in Section 6. Section 7 concludes the paper.

2. Related work

Many tracking algorithms assume that target motion is continuous. Given this assumption, we can apply local tracking algorithms Comaniciu et al. (2003); Isard & Blake (1998); Wang & Yagi (2008b). In the local tracking algorithms, the mean-shift algorithm Comaniciu et al. (2003) aims at searching for a peak position using density gradient estimation, whereas particle filtering techniques Isard & Blake (1998); Rathi et al. (2005); Wang & Yagi (2009); Zhao et al. (2008); Zhou et al. (2006) use a dynamic model to guide the particle propagation within a limited sub-space of target state. Particle filtering tracking algorithms have certain robustness against sudden motions. The mean-shift algorithm can deal with partial occlusions.

Tracking can be formulated as template matching Hager & Belhumeur (1998). A target is characterized by a template that can be parametric or non-parametric. The task of a template matching tracking is to find the region that is the most similar to the template. Template matching techniques do not require the continuous motion assumption. Therefore, it is possible to handle occlusions and sudden motions. We will introduce local tracking and global matching techniques. The objective of our algorithm in this chapter it to combine the advantages of the local and global matching techniques.

2.1 Local tracking

There are many local tracking methods. Tracking was treated as a binary classification problem in previous works. An adaptive discriminative generative model was suggested in Lin et al. (2004) by evaluating the discriminative ability of the object from the foreground using a Fisher Linear Discriminant function. Fisher Linear Discriminant function was also using in Nguyen & Smeulders (2006) to provide good discrimination. Comaniciu et al. Comaniciu et al. (2003) take of the advantage of this method to their mean-shift algorithm, where colors that appear on the object are down weighted by colors that appear in the background. Collins et al. Collins & Liu (2005) explicitly treat tracking as a binary classification problem. They apply mean-shift algorithm using discriminative features selected by two online discriminative evaluation methods (Variance ration and peak difference). Avidan Avidan (2007) proposes ensemble tracking that updates a collection of weak classifiers. The Collection of weak classifiers are assembled to make a strong classifier, which separates the foreground object from the background. The weak classifiers are maintained by adding or removing at any time to deal with appearance variations.

Temporal integration methods include particle filtering to properly integrate measurements over time. The WSL tracking that maintains short-term and long term

2.2 Exhaustive matching

Describing a target by one or many templates, tracking can be formulated as exhaustive searching. A target represented by its whole appearance can be matched with each region in the input image by comparing the Sum of Squared Distances (SSD). Template using SSD matching is not flexible because it is sensitive to viewpoint, illumination changes. To deal with these problems, histograms are employed for characterizing targets. Histogram representation is extended to a spatiogram-based tracking algorithm Birchfield & Rangarajan (2005), which makes use of spatial information in addition to color information. A histogram contains many bins which are spatially weighted by the mean and covariance of the location of the pixels that contribute to that bin. Since the target is presented by one histogram, the tracking is not reliable when occlusion exist. The computational cost is also high due to the exhaustive matching. Tuzel et al. Tuzel et al. (2006) introduce covariance matrix to describe the target. This descriptor contains appearance and spatial information. The target localization process is formulated as an expensive exhaustive searching. Moreover, the similarity measure in Tuzel et al. (2006) is adopted from Pennec et al. (2006), which is an affine invariant metric. The affine invariant metric used in Tuzel et al. (2006) is computationally expensive.

3. Local tracking

3.1 Target representation for local tracking

The local tracking is performed based on foreground likelihood. The foreground likelihood is computed using the selected discriminative color and shape-texture features Wang & Yagi (2008a). The target is localized using mean-shift local mode seeking on the integrated foreground likelihood image.

We represent a target using color and shape-texture information. Color information is important because of its simplicity and discriminative ability. Color information only is not always sufficiently discriminative. Shape-texture information is helpful for separating a target and its background. Therefore, the target representation for our local tracking consists of color and shape-texture features.

3.1.1 Multiple Color Channels

We represent color distributions on a target and its background using color histograms. We select several color channels from different color spaces. Among them, we compute color histograms for the R, G, and B channels in the RGB space; the H, S, and V channels in the HSV space. Different from the approach in Wang & Yagi (2008a), we do not use the *r* and *g* channels in the normalized *rg* space because they are found not discriminative in many sequences. Although the *r* and *g* channels have good invariant ability to illumination changes, the gain from this advantage is not very important in our approach since we use global matching and local matching. The histograms computed in R,G, B, H, and S channels are quantized into 12 bins respectively. The color distribution in the V channel is not used here because we found that intensity is less helpful in our tracking tasks. The *rg* space has been shown to be reliable when the illumination changes. Thus *r* and *g* are also employed. There are 5 color features in the candidate feature set.

A color histogram is calculated using a weighting scheme. The contributions of different pixels to the object representation depend on their position with respect to the center of the target. Pixels near the region center are more reliable than those further away. Smaller weights are given to those further pixels by using Epanechnikov kernel Comaniciu et al. (2003) as a weighting function:

$$k(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2), \text{ if } \|\mathbf{x}\|^2 \leq 1; \\ \\ 0, \text{ otherwise}, \end{cases} \tag{1}$$

where c_d is the volume of the unit d-dimensional sphere; $\mathbf{x}$ the local coordinates with respect to the center of the target. Thus, we increase the reliability of the color distribution when these boundary pixels belong to the background or get occluded.

The color distribution $h_f = \{p_f^{(b_{in})}\}_{b_{in}=1\ldots m}$ of the target is given by

$$p_f^{(b_{in})} = C_f \sum_{\mathbf{x}_i \in R_f} k(\|\mathbf{x}_i\|)\delta[h(\mathbf{x}_i) - b_{in}], \tag{2}$$

where δ is the Kronecker delta function and $h(\mathbf{x}_i)$ assigns one of the m-bins ($m = 12$) of the histogram to a given color at location $\mathbf{x}_i$. C_f is a normalization constant. It is calculated as

$$C_f = \frac{1}{\sum_{\mathbf{x}_i \in R_f} k(\|\mathbf{x}_i\|^2)}. \tag{3}$$

The tracking algorithm searches for the target in a new frame from the target candidates. The target candidates are represented by

$$p_c^{(b_{in})}(\mathbf{y}) = C_b \sum_{\mathbf{x}_i \in R_f} k(\frac{\|\mathbf{y} - \mathbf{x}_i\|}{h})^2 \delta[h(\mathbf{x}_i) - b_{in}], \tag{4}$$

where C_b is

$$C_b = \frac{1}{\sum_{\mathbf{x}_i \in R_c} k(\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\|)^2}. \tag{5}$$

and R_f is the target region.

3.1.2 Shape-texture information

Shape-texture information plays an important role for describing a target. Shape-texture information has a few nice properties such as certain invariant ability to illumination changes. Shape-texture information can be characterized by various descriptors Belongie et al. (2002); Berg & Malik (2001); Lowe (1999). We describe a target's shape-texture information by orientation histograms, which is computed based on image derivatives in x and y directions. We did not use the popular Sobel masks in this calculation. Instead, the Scharr masks (S_x and S_y) are employed here because they give more accurate results than the Sobel kernel.

The gradients at the point (x, y) in the image I can be calculated by convolving the Scharr masks with the image:

$$D_x(x,y) = S_x * I(x,y),$$

and

$$D_y(x,y) = S_y * I(x,y).$$

The strength of the gradient at the point (x, y)

$$D(x,y) = \sqrt{D_x(x,y)^2 + D_y(x,y)^2}.$$

In order to ignore noise, a threshold is given

$$D^{'}(x,y) = \begin{cases} D(x,y), \text{if } D(x,y) \geq T_D, \\ 0, \text{ otherwise,} \end{cases} \tag{6}$$

where T_D is a threshold given empirically.
The orientation of the edge is

$$\theta(x,y) = \arctan(\frac{D_y(x,y)}{D_x(x,y)}). \tag{7}$$

The orientations are also quantized into 12 bins. A orientation histogram can be calculated using a approach similar to the calculation of a color histogram, as introduced in the previous subsection.

3.2 Feature selection for local tracking

We select a subset of features from the feature pool in the 5 color channels and 1 shape-texture representation. We evaluate the discriminative ability of each feature based on the histograms calculated on the target and its background. The discriminative ability of each feature is dependent on the separability between the target and its background. The weighted histograms introduced in the last section do not directly reflect the descriptive ability of the features. A log-likelihood ratio histogram can be helpful for solving this problem Collins (2003); Swain & Ballard (1991); Wang & Yagi (2006). We calculate likelihood images for each feature. Then, we compute likelihood ratio images of the target and its background. Finally, we select good features by ranking the discriminative ability of different features.

3.2.1 Likelihood images

Given target representation using a specific feature, we want to evaluate the probability on an input image. The probability indicates the likelihood of appearance of the target. we We compute foreground likelihood based on the histograms of the foreground and background with respect to a given feature. The frequency of the pixels that appear in a histogram bin is calculated as $\zeta_f^{(b_{in})} = p_f^{(b_{in})}/n_{fg}$ and $\zeta_b^{(b_{in})} = p_b^{(b_{in})}/n_{bg}$, where n_{fg} is the pixel number of the target region and n_{bg} the pixel number of the background.
The log-likelihood ratio of a feature value is given by

$$L^{(b_{in})} = \max(-1, \min(1, \log\frac{\max(\zeta_f^{(b_{in})}, \delta_L)}{\max(\zeta_b^{(b_{in})}, \delta_L)})), \tag{8}$$

where δ_L is a very small number. The likelihood image for each feature is created by back-projecting the ratio into each pixel in the image Swain & Ballard (1991); Wang & Yagi (2008a).

3.2.2 Color and shape-texture likelihood ratio images

Based on the multi-cue representation of the target and its background, we can compute the likelihood probability in an input image. The values in likelihood images have large variations since they are not normalized. We need good representation of different features and evaluate their discriminative ability. Log-likelihood ratios of the the target and background provide such representation. We calculate log-likelihood ratios based on the histograms of the foreground and background with respect to a given feature. The likelihood ratio produces a function that maps feature values associated with the target to positive values and the background to negative values. The frequency of the pixels that appear in a histogram bin is calculated as

$$\zeta_f^{(b_{in})} = \frac{p_f^{(b_{in})}}{n_{fg}}, \tag{9}$$

and

$$\zeta_b^{(b_{in})} = \frac{p_b^{(b_{in})}}{n_{bg}}, \tag{10}$$

where n_{fg} is the pixel number of the target region and n_{bg} the pixel number of the background. The log-likelihood ratio of a feature value is given by

$$L^{(b_{in})} = \max(-1, \min(1, \log\frac{\max(\zeta_f^{(b_{in})}, \delta_L)}{\max(\zeta_b^{(b_{in})}, \delta_L)})), \tag{11}$$

where δ_L is a very small number. The likelihood image for each feature is created by back-projecting the ratio into each pixel in the image.

We use likelihood ratio images as the foundation for evaluating the discriminative ability of the features in the candidate feature set. The discriminative ability will be evaluated using variance ratios of the likelihood ratios, which will be discussed in the next subsection.

3.2.3 Feature selection using variance ratios

Given m_d features for tracking, the purpose of the feature selection module is to find the best subset feature of size m_m, and $m_m < m_d$. Feature selection can help minimize the tracking error and maximize the descriptive ability of the feature set.

We find the features with the largest corresponding variances. Following the method in Collins (2003), based on the equality $\mathrm{var}(x) = E[x^2] - (E[x])^2$, the variance of Equation(11) is computed as

$$\mathrm{var}(L;p) = E[(L^{b_{in}})^2] - (E[L^{b_{in}}])^2.$$

The variance ratio of the likelihood function is defined as Collins (2003):

$$\mathrm{VR} = \frac{\mathrm{var}(B \cup F)}{\mathrm{var}(F) + \mathrm{var}(B)} = \frac{\mathrm{var}(L; (p_f + p_b)/2)}{\mathrm{var}(L; p_f) + \mathrm{var}(L; p_b)}. \tag{12}$$

We evaluate the discriminative ability of each feature by calculating the variance ratio. In the candidate feature set, the color feature includes 7 different features: the color histograms of R, G, B, H, S, r, and g, while the appearance feature includes a gradient orientation histogram. These features are ranked according to the discriminative ability by comparing the variance ratio. The feature with the maximum variance ratio is taken as the most discriminative feature.

3.3 Location estimation for local tracking

We select discriminative features from the color and shape-texture feature pool. These features are employed to compute likelihood images. We extend the basic mean-shift algorithm to our local tracking framework. We combine the likelihood images calculated using different discriminative features. The combined likelihood images are used for our location estimation. In this section, we will introduce the localization strategy in the basic mean-shift algorithm. Then, we discuss how many features are appropriate for the local tracking. Finally, we will describe the localization in our local tracking.

3.3.1 Localization using the standard mean-shift algorithm

The localization process for our local tracking can be described as a minimization process, which aims at searching for the position with maximum similarity with the target. The minimizing process can be formulated as a gradient descent process in the basic mean-shift algorithm. The mean-shift algorithm is a robust non-parametric probability density gradient estimation method. It is able to find the mode of the probability distributions of samples. It can estimate the density function directly from data without any assumptions about underlying distribution. This virtue avoids choosing a model and estimating its distribution parameters Comaniciu & Meer (2002). The algorithm has achieved great success in object tracking Comaniciu et al. (2003) and image segmentation Comaniciu & Meer (2002). However, the basic mean shift tracking algorithm assumes that the target representation is sufficiently discriminative against the background. This assumption is not always true especially when tracking is carried out in a dynamic background such as surveillance with a moving camera. We extend the basic mean shift algorithm to an adaptive mean shift tracking algorithm that can choose the most discriminative features for effective tracking.

The standard mean shift tracker finds the location corresponding to the target in the current frame based on the appearance of the target. Therefore, a similarity measure is needed between the color distributions of a region in the current frame and the target model. A popular measure between two distributions is the Bhattacharyya distance Comaniciu et al. (2003); Djouadi et al. (1990). Considering discrete densities such as two histograms $p = \{p^{(u)}\}_{u=1...m}$ and $q = \{q^{(u)}\}_{u=1...m}$ the coefficient is calculated by:

$$\rho[p,q] = \sum_{b_{in}=1}^{m} \sqrt{p^{(b_{in})} q^{(b_{in})}}. \tag{13}$$

The larger ρ is, the more similar the distributions are. For two identical histograms we obtain $\rho = 1$, indicating a perfect match. As the distance between two distributions, the measure can be defined as Comaniciu et al. (2003):

$$d = \sqrt{1 - \rho[p,q]}, \tag{14}$$

which d is the Bhattacharyya distance.

The tracking algorithm recursively computes an offset value from the current location $\hat{\mathbf{y}}_0$ to a new location $\hat{\mathbf{y}}_1$ according to the mean shift vector. $\hat{\mathbf{y}}_1$ is calculated by using Comaniciu &

Meer (2002); Comaniciu et al. (2003)

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} x_i w_i g(\frac{y-x_i}{h})}{\sum_{i=1}^{n_h} w_i g(\frac{y-x_i}{h})}. \tag{15}$$

where $w_i = \sum_{u=1}^{m} \sqrt{\frac{q^{(u)}}{p^{(u)}(\mathbf{y}_0)}} \delta[h(\mathbf{x}_i) - b_{in}]$ and $g(x) = -k'(x)$.

3.3.2 How many features are appropriate?

We evaluate the discriminative abilities of different features in the feature pool. In the Evaluation, we rank the features according to their discriminative ability against the background. Features with good discriminative ability can be combined to represent and localize the target. The combination of features needs to be carried out carefully. Intuitively, the more features we use, the better the tracking performance; however, this is not true in practice. According to information theory, the feature added into the system can bring negative effect as well as improvement of the performance Cover & Thomas (1991). This is due to the fact that the features used are not totally independent. Instead, they are correlated. In our implementation, two kinds of features are used to represent the target, a number, which according to the experimental results, is appropriate in most cases. We have tested a system using 1 or 3 features, which gave worse performances. During the initialization of the tracker, the features ranked in the top two are selected for the tracking. The feature selection module runs every 8 to 12 frames. When the feature selection module selects features different from those in the initialization, only one feature is replaced each time. Only the second feature of the previous selection will be discarded and replaced by the best one in current selection. This strategy is very important in keeping the target from drifting.

3.3.3 Target localization for local tracking

The proposed tracking algorithm combines the top two features through back-projection Bradski (1998) of the joint histogram, which implicitly contains certain spatial information that is important for the target representation. Based on Equation(4), we calculate the joint histogram of the target with the top two features,

$$p_f^{(b_{in}^{(1)}, b_{in}^{(2)})} = C \sum_{\mathbf{x}_i \in R_f} k(\|\mathbf{x}_i\|)\delta[h(\mathbf{x}_i) - b_{in}^{(1)}]\delta[h(\mathbf{x}_i) - b_{in}^{(2)}], \tag{16}$$

and a joint histogram of the searching region

$$p_b^{(b_{in}^{(1)}, b_{in}^{(2)})} = C \sum_{\mathbf{x}_i \in R_b} k(\|\mathbf{x}_i\|)\delta[h(\mathbf{x}_i) - b_{in}^{(1)}]\delta[h(\mathbf{x}_i) - b_{in}^{(2)}]. \tag{17}$$

We get a division histogram by dividing the joint histogram of the target by the joint histogram of the background,

$$p_d^{(b_{in}^{(1)}, b_{in}^{(2)})} = \frac{p_f^{(b_{in}^{(1)}, b_{in}^{(2)})}}{p_b^{(b_{in}^{(1)}, b_{in}^{(2)})}}. \tag{18}$$

The division histogram is normalized for the histogram back-projection. The pixel values in the image are associated with the value of the corresponding histogram bin by histogram back-projection. The back-projection of the target histogram with any consecutive frame generates a probability image $p = \{p_w^i\}_{i=1...n_h}$ where the value of each pixel characterizes the probability that the input pixel belongs to the histograms. The two images of the top two features have been computed for the back-projection. Note that the H, S, r, and g images are calculated by transferring the original image to the HSV and the rg spaces; the orientation image has been calculated using the approach introduced in section III(B).

Since we are using an Epanechnikov profile the derivative of the profile, $g(x)$, is constant. The target's shift vector in the current frame is computed as

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i p_w^i}{\sum_{i=1}^{n_h} p_w^i}. \tag{19}$$

The tracker assigns a new position to the target by using

$$\hat{\mathbf{y}}_1 = \frac{1}{2}(\hat{\mathbf{y}}_0 + \hat{\mathbf{y}}_1). \tag{20}$$

If $\|\hat{\mathbf{y}}_0 - \hat{\mathbf{y}}_1\| < \varepsilon$, this position is assigned to the target. Otherwise, compute the Equation(19) again. In our algorithm, the number of the computation is set to less than 15. In most cases, the algorithm converges in 3 to 6 loops.

3.4 Target model updating for local tracking

The local tracker needs adaptivity to handle appearance changes. The model is computed by mixing the current model with the initial model which is considered as correct Wang & Yagi (2008a). The mixing weights are generated from the similarity between the current model and the initial model Wang & Yagi (2008a). The initial model works in a similar way to the stable component in Jepson et al. (2003). But the updating approach in Wang & Yagi (2008a) takes less time.

Updating the target model adaptively may lead to tracking drift because of the imperfect classification of the target and background. Collins and Liu Collins (2003) proposed that forming a pooled estimate allows the object appearance model to adapt to current conditions while keeping the overall distribution anchored to the original training appearance of the object. They assume that the initial color histogram remains representative of the object appearance throughout the entire tracking sequence. However, this is not always true in real image sequences.

To update the target model, we propose an alternative approach that is based on similarities between the initial and current appearance of the target. The similarity s is measured by a simple correlation based template matching Atallah (2001) performed between the current and the initial frames. The updating is done according to the similarity s:

$$H_m = (1 - s)H_i + sH_c, \tag{21}$$

where the H_i is the histogram computed on the initial target; the H_c the histogram of the target current appearance, the H_m the updated histogram of the target.

The template matching is performed between the initial model and the current candidates. Since we do not use the search window that is necessary in template matching-based tracking, the matching process is efficient and brings little computational cost to our algorithm. The performance of the proposed algorithm is improved by using this strategy, which will be shown in the next section.

4. Covariance matching in riemannian manifold

We describe our covariance matching in Riemmannian manifold in this section. We introduce some important theories on Riemannian manifold. Since the affine invariant metric used in Tuzel et al. (2006) is computationally expensive, we apply the efficient Log-Euclidean metric in the manifold. Finally, we give the updating strategy for the covariance matching.

4.1 Basic concepts for global matching in riemannian manifold

We will introduce some basic concepts of Riemannian geometry, which is important for our global tracking formulation. We describe differentiable manifold, Lie groups, Lie algebras, and Riemannian manifold. The details of the theories are referred to Gilmore (2006); Jost (2001).

4.1.1 Differentiable manifold

A manifold $\mathcal{M}$ is a Hausdorff topological space, such that for every point $\mathbf{x} \in \mathcal{M}$ there exists a neighborhood $\mathcal{N} \subset \mathcal{M}$ containing x and an associated homeomorphism from $\mathcal{N}$ to some Euclidean space R^m. The neighborhood $\mathcal{N}$ and its associated mapping ϕ together form a coordinate chart. A collection of chart is named as an atlas.

If a manifold is locally similar enough to Euclidean space, it is allowed to do calculus. A differentiable manifold is such kind of manifold that is also a topological manifold with globally defined differential structure. Any topological manifold can be given a differential structure locally by using the homeomorphisms in this atlas. One may apply ideas from calculus which working within the individual charts, since these lie in Euclidean spaces to which the usual rules of calculus apply.

4.1.2 Lie groups

Lie groups are finite-dimensional real smooth manifold with continuous transformation group properties Rossmann (2003). Group operations can be applied into Lie groups.

Assuming we have two groups, G_1 and G_2, we can define a homomorphism $f_A : G_1 \rightarrow G_2$ for them. The homomorphism f is required to be continuous (not necessarily to be smooth). If we have another homomorphism $f_B : G_3 \rightarrow G_4$, the two homomorphisms are combined into a new homomorphism. A category is formulated by composing all the Lie groups and morphisms. According to the type of homomorphisms, there are two kinds of Lie groups: isomorphic Lie groups with bijective homomorphisms.

Homomorphisms are useful in describing Lie groups. We can represent a Lie group on a vector space V. We chose a basis for the vector space, the Lie group representation is expressed as a homomorphisms into $GL(n, K)$, which is known as a matrix representation. If we have two vector spaces V_1 and V_2, the two representations of G on V_1 and V_2 are equivalent when they have the same matrix representations with respect to some choices of bases for V_1 and V_2.

4.1.3 Lie algebras

We may consider Lie groups as smoothly varying families of symmetries. Small transformation is an essential property of Lie groups. In such situations, Lie algebras can be defined because Lie groups are smooth manifold with tangent spaces at each point. Lie algebra, an algebraic structure, is critical in studying differentiable manifolds such as Lie groups. Lie algebra is able to replace the global object, the group, with its local or linearized version. In practice, matrices sets with specific properties are the most useful Lie groups.

Matrix Lie groups is defined as closed subgroups of general linear groups $\mathbf{GL}(n, \mathcal{R})$, the group of $n \times n$. nonsingular matrices.

We associate a Lie algebra with every Lie group. The underlying vector space is the tangent space of Lie group at the identity element, which contains the complete local structure of the group. The elements of the Lie algebra can be thought as elements of the group that are infinitesimally close to the identity. The Lie algebra provides a commutator of two such infinitesimal elements with the Lie bracket. We can connect vector space with Lie algebra preserves the Lie bracket.

Each Lie group has a identity component, which is an open nomal subgroup. All the connected Lie groups forms the universal cover of these groups. Any Lie group G can be decomposed into discrete abelian groups.

We can not define a global structure for a Lie group using its Lie algebra. However, if the Lie group is simply connected, we can determine the global structure based on its Lie algebra.

Tensors are defined as multidimensional arrays of numbers. It is an extension of matrix, which is a 2D definition. The entries of such arrays are symbolically denoted by the name of tensor with indices giving the position in the array. Covariance

4.1.4 Exponential maps

A Lie algebra homomorphism is a mapping: every vector v in Lie algebra g is a linear map from R taking 1 to v. Because R is the Lie algebra of the simply connected Lie group R, this induces a Lie group homorphism $f : R \rightarrow G$. The operation of c is

$$c(s + t) = c(s)c(t) \tag{22}$$

for all s and t. We easily find that it is similar to exponential function

$$exp(v) = c(1). \tag{23}$$

This exponential function is name as exponential map which maps the Lie algebra g into the Lie group G. Between a neighborhood of the identity element of g, there is a diffeomorphism. The exponential map is a generalization of the exponential function for real numbers. In fact, the exponential function can be extended into complex numbers and matrices, which is important in computing Lie groups and Lie algebras.

Since we are interested in symmetric matrices, matrix operators are important for the computation on Lie algebra. The exponential map from the Lie algebra is defined by

$$\exp(A) = \sum_{i=0}^{\infty} \frac{1}{i!} A^i, \tag{24}$$

It is possible to decompose A into an orthogonal matrix U and a diagonal matrix ($A = UDU^T$, $D = DIAG(d_i)$), we compute power k of A using the same basis

$$A^k = UD^kU^T, \tag{25}$$

where the rotation matrices in the computation is factored out. The mapping of exponential to each eigenvalue:

$$\exp(A) = U\mathrm{DIAG}(\exp(d_i))U^T. \tag{26}$$

An inverse mapping is defined in the neighborhood: $\log_{\mathbf{X}} : G \rightarrow T_{\mathbf{X}}R$. This definition is unique. For certain manifolds, the neighborhood can be extended more regions in the tangent space and manifold.

This operator is able to be applied to any square matrix. The definitions above are meaningful only for matrix groups. Since we concern matrix groups in this work, the definitions are very important for understanding our algorithm.

4.2 Improving covariance tracking

The covariance tracker Porikli et al. (2006) describes objects using covariance matrices. The covariance matrix fuses different types of features and modalities with small dimensionality. Covariance tracking searches all the regions and guarantees a global optimization (Up to the descriptive ability of the covariance matrices). Despite of these advantages, covariance tracking is relatively expensive due to the distance computation and model updating in Riemannian manifold. We speed up the global searching and the model updating by introducing Log-Euclidean metrics.

4.2.1 Target representation

The target is described by covariance matrices that fuse multiple features. We adopt the features used in Porikli et al. (2006), which consist of pixel coordinates, RGB colors and gradients. The region R is described with the $d \times d$ covariance matrix of the feature points in R

$$C_R = \frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{z}_k - \bar{\ })(\mathbf{z}_k - \bar{\ })^T, \tag{27}$$

where $\bar{\ }$ is the mean of the points.

The covariance of a certain region reflects the spatial and statistical properties as well as their correlations of a region. However, the means of the features are not taken into account for tracking. We use the means by computing the foreground likelihoods and incorporate them into the covariance computation.

4.2.2 Similarity measuring for covariance matrices

The simplest way for measuring similarity between covariance matrices is to define a Euclidean metric, for instance, $d^2(C_1, C_2) = \mathrm{Trace}((C_1 - C_2)^2)$ Arsigny et al. (2005). However, the Euclidean metric can not be applied to measure the similarity due to the fact that covariance matrices may have null or negative eigenvalues which are meaningless for the Euclidean metrics Forstner & Moonen (1999). In addition, the Euclidean metrics are not appropriate in terms of symmetry with respect to matrix inversion, e.g., the multiplication of covariance matrices with negative scalars is not closed for Euclidean space.

Since covariance matrices do not lie on Euclidean space, affine invariant Riemannian metrics Forstner & Moonen (1999); Pennec et al. (2006) have been proposed for measuring similarities between covariance matrices. To avoid the effect of negative and null eigenvalues, the distance measure is defined based on generalized eigenvalues of covariance matrices:

$$\rho(C_1, C_2) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(C_1, C_2)}, \tag{28}$$

where $\{\lambda_i(C_1, C_2)\}_{i=1\ldots n}$ are the generalized eigenvalues of C_1 and C_2, computed from

$$\lambda_i C_1 \mathbf{x}_i - C_1 \mathbf{x}_i = 0, i = 1 \ldots d, \tag{29}$$

and $\mathbf{x}_i \neq 0$ are the generalized eigenvectors. The distance measure ρ satisfies the metric axioms for positive definite symmetric matrices C_1 and C_2. The price paid for this measure is a high computational burden, which makes the global searching expensive.

In this work, we use another Riemannian metrics – Log-Euclіean metrics proposed in Arsigny et al. (2005). When only the multiplication on the covariance space is considered, covariance matrices have Lie group structures. Thus the similarity can be measured in the domain of logarithms by Euclidean metrics:

$$\rho_{LE}(C_1, C_2) = \| \log(C_1) - \log(C_2) \|_{Id}. \tag{30}$$

This metric is different from the classical Euclidean framework in which covariance matrices with null or negative eigenvalues are at an infinite distance from covariance matrices and will not appear in the distance computations.

Although Log-Euclіean metrics are not affine-invariant Arsigny et al. (2005), some of them are invariant by similarity (orthogonal transformation and scaling). It means that the Log-Euclidean metrics are invariant to changes of coordinates obtained by a similarity Arsigny et al. (2005). The properties of Log-Euclidean make them appropriate for similarity measuring of covariance matrices.

4.3 Model updating

Covariance tracking has to deal with appearance variations. Porikli et al. Porikli et al. (2006) construct and update a temporal kernel of covariance matrices corresponding to the previously estimated object regions. They keep a set of previous covariance matrices $[C_1 \ldots C_T]$. From this set, they compute a sample mean covariance matrix that blends all the previous matrices. The sample mean is an intrinsic mean Porikli et al. (2006) because covariance matrices do not lie on Euclidean spaces. Since covariance matrices are symmetric positive definite matrices, they can be formulated as a connected Riemannian manifold. The structure of the manifold is specified by a Riemannian metric defined by collection of inner products. The model updating is computationally expensive due to the heavy burden of computation in Riemannian space.

In this work, we use the Log-Euclidean mean of T covariance matrices with arbitrary positive weights $(w_i)_{i=1}^{T}$ such that $\sum_{i=1}^{T} w_i = 1$ is a direct generalization of the geometric mean of the matrices. It is computed as

$$C_m = \exp(\sum_{i=1}^{T} \log(C_i)). \tag{31}$$

This updating method need much less computational costs than the method used in Porikli et al. (2006).

5. Switching criteria

The local tracking strategy is adopted when the tracker runs in steady states. When sudden motion, distractions or occlusions happen, local tracking strategy tends to fail due to its limited searching region. We switch to the global searching strategy based on the improved covariance tracker described in the previous section. Motion prediction techniques such the Kalman filter have been used to deal with occlusions. However, when the prediction is far away from the true location, a global searching is preferred to recover from tracking failure.

Algorithm	Seq1	Seq2	Seq3
Meanshift	72.6	78.5	35.8
Covariance	89.7	90.4	78.8
TheProposed	91.3	88.1	83.1

Table 1. Tracking percentages of the proposed and other trackers.

The detection of sudden motion and distraction is performed using the effective methods proposed in Wang & Yagi (2007). Occlusions are announced when the objective function value of the local tracking is lower than some threshold t_l. The threshold for switching between local and covariance tracking is computed by fitting a Gaussian distribution based on the similarity scores (Bhattacharyya distances) of the frames labeled as occlusion. The threshold is set to $3\sigma_t$ from the mean of the Gaussian. The covariance tracking is applied when the above threats are detected.

6. Experiments

We verify our approach by tracking different objects in some challenging video sequences.
We compare the performance of the mean-shift algorithm and the proposed method in Figure. 1. The face in the sequence moves very fast. Therefore, the mean-shift tracker fails to capture the face. The proposed method combines multiple features for local tracking. It is possible to track the target thorough the sequence. The example in Figure. 1 demonstrate the power of the local tracking part in our approach.
In Figure. 2, we show the tracking results on the street sequence Leibe et al. (2007). Pedestrians are articulated objects which are difficult to track. The occlusions in frame 7574 brings more difficulty to the tracking. The proposed tracker successfully tracks through the whole sequence.
We compare the proposed tracker with the mean-shift and covariance trackers. Different objects in the three sequences Leibe et al. (2007) are tracked and the tracking percentages are given in Table. 1. The proposed tracker provides higher or similar correct ratio.

6.1 Computation complexity

The tracking is faster when the local tracking method is applied since the searching of local tracking is only performed on certain the regions. It takes less than 0.02 seconds to process one frame.
The covariance tracking is also sped up thanks to the efficiency of Log-Euclidean distance computation adopted in this work. The iterative computation of the affine invariant mean leads to heavy computational cost. In contrast, the Log-Euclidean metrics are computed in a closed form. The computation of mean based on Log-Euclidean distances takes less than 0.02 seconds, whereas the computation based on Riemannian invariant metrics takes 0.4 seconds.

7. Conclusions

We propose a novel tracking framework taking the advantages of local and global tracking strategies. The local and global tracking are performed by using the mean-shift and covariance matching. The proposed tracking algorithm is efficient because local searching strategy is adopted for most of the frames. It can deal with occlusions and large motions for the switching

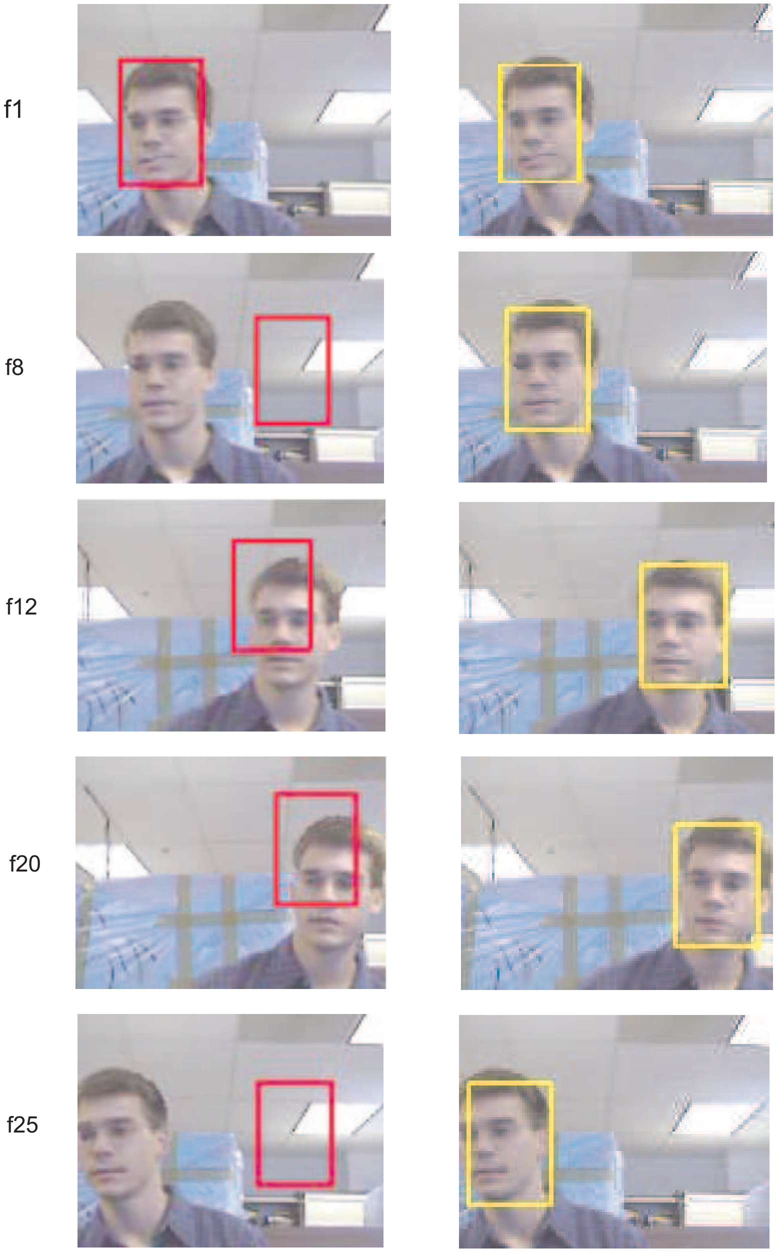


Fig. 1. Face tracking results using the basic mean shift algorithm (in the first row) and the proposed method (in the second row). The face in the sequence moves quickly.

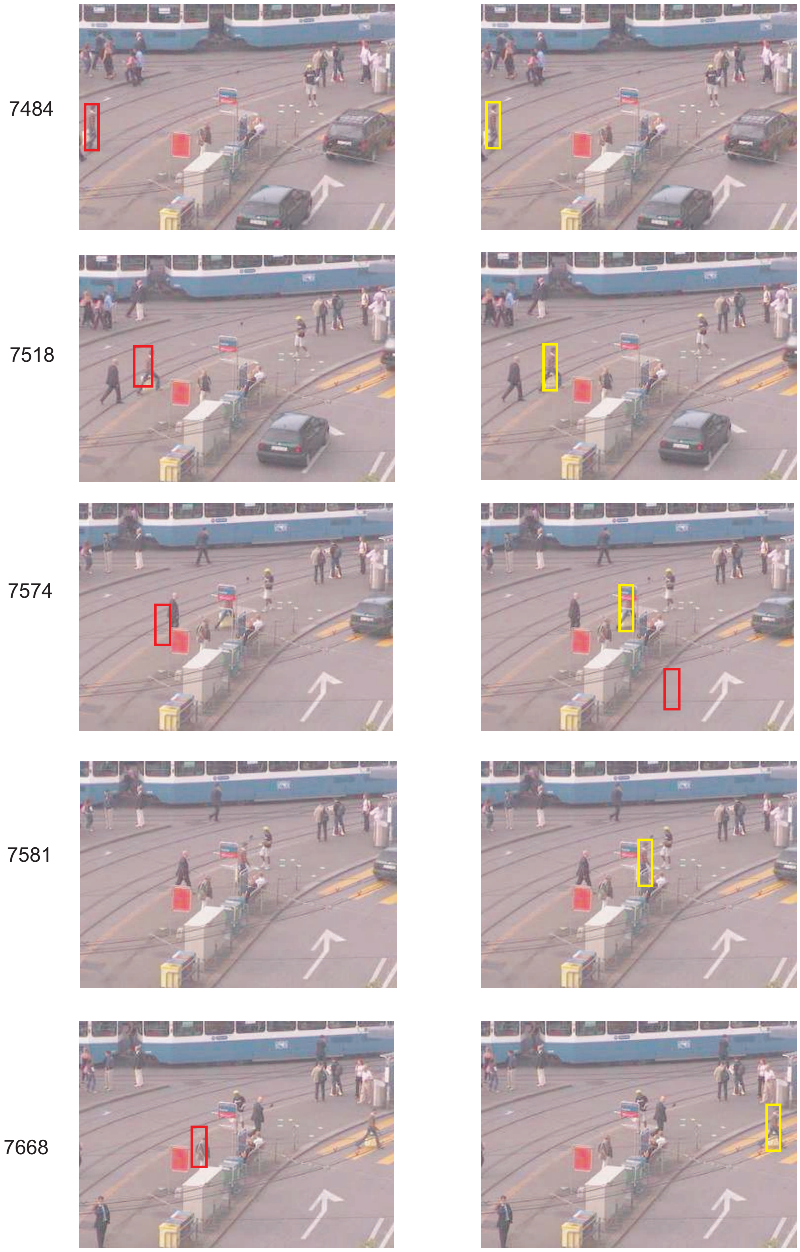


Fig. 2. Tracking pedestrian in the complex background. No background subtraction is applied in the tracking.

from local to global matching. We adopt Log-Euclidean metrics in the improved covariance tracking, which makes the global matching and model updating fast.

8. References

Arsigny, V., Fillard, P., Pennec, X. & Ayache, N. (2005). Fast and simple calculus on tensors in the log-euclidean framework, *Proc. MICCAI'05*, pp. 115–122.

Atallah, M. J. (2001). Faster image template matching in the sum of the absolute value of differences measure, *IEEE Transactions on Image Processing* 10(4): 659–663.

Avidan, S. (2007). Ensemble tracking, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 29(2): 261–271.

Belongie, S., Malik, J. & Puzicha, J. (2002). Shape matching and object recognition using shape contexts, *IEEE Trans. Pattern Anal. Mach. Intell.* 24(4): 509–522.

Berg, A. C. & Malik, J. (2001). Geometric blur for template matching, *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 607–614.

Birchfield, S. & Rangarajan, S. (2005). Spatiograms versus histograms for region-based tracking, *Proc. of Conf. Computer Vision and Pattern Recognition*, pp. 1158–1163.

Bradski, G. (1998). Computer vision face tracking as a component of a perceptural user interface, *Proc. of the IEEE Workshop Applications of Computer Vision*, pp. 214–219.

Collins, R. T. (2003). Mean-shift blob tracking through scale space, *Proc. CVPR*, pp. 234–240.

Collins, R. T. & Liu, Y. (2005). On-line selection of discriminative tracking features, *IEEE Trans. Pattern Anal. Mach. Intell.* 27(10): 1631–1643.

Comaniciu, D. & Meer, P. (2002). Mean shift: a robust approach toward feature space analysis, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 24(5): 603–619.

Comaniciu, D., Ramesh, V. & Meer, P. (2003). Kernel-based object tracking, *IEEE Trans. Pattern Anal. Mach. Intell.* 25(5): 564–577.

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory*, John Wiley and Sons Press.

Djouadi, A., Snorrason, O. & Garber, F. D. (1990). The quality of training sample estimates of the bhattacharyya coefficient, *IEEE Trans. Pattern Anal. Mach. Intell.* 12(1): 92–97.

Forstner, W. & Moonen, B. (1999). A metric for covariance matrices, *Technical report, Dept. of Geodesy and Geoinformatics* .

Gilmore, R. (2006). *Lie Groups, Lie Algebras, and Some of Their Applications*, Dover Publications.

Hager, G. D. & Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination, *IEEE Trans. Pattern Anal. Mach. Intell.* 20(10): 1025–1039.

Isard, M. & Blake, A. (1998). Condensation - conditional density propagation for tracking, *Intl. Journal of Computer Vision* 29(1): 2–28.

Jepson, A. D., Fleet, D. J. & EI-Maraghi, T. (2003). Robust online appearance models for visual tracking, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 25(10): 1296–1311.

Jost, J. (2001). *Riemannian Geometry and Geometric Analysis*, Springer.

Leibe, B., Schindler, K. & Gool, L. V. (2007). Coupled detection and trajectory estimation for multi-object tracking, *Proc. of Int'l Conf. on Computer Vision*, pp. 115–122.

Lin, R.-S., Ross, D. A., Lim, J. & Yang, M.-H. (2004). Adaptive discriminative generative model and its applications, *Proc. Conf. Neural Information Processing System.*

Lowe, D. G. (1999). Object recognition from local scale-invariant features, *Proc. ICCV'99*, pp. 1150–1157.

Nguyen, H. T. & Smeulders, A. W. M. (2006). Robust tracking using foreground-background texture discrimination, *International Journal of Computer Vision* 69(3): 277–293.

Pennec, X., Fillard, P. & Ayache, N. (2006). A riemannian framework for tensor computing, *Intl. Journal of Computer Vision* 66: 41–66.

Porikli, F., Tuzel, O. & Meer, P. (2006). Covariance tracking using model update based on lie algebra, *Proc. of Intl Conf. on Computer Vision and Pattern Recognition*, pp. 728–735.

Rathi, Y., Vaswani, N., Tannenbaum, A. & Yezzi, A. (2005). Particle filtering for geometric active contours with application to tracking moving and deforming objects, *Proc. of Conf. Computer Vision and Pattern Recognition*, pp. 2–9.

Rossmann, W. (2003). *Lie groups: an introduction through linear groups*, London: Oxford University Press.

Swain, M. & Ballard, D. (1991). Color indexing, *Intl. Journal of Computer Vision* 7(1): 11–32.

Tuzel, O., Porikli, F. & Meer, P. (2006). Region covariance: A fast descriptor for detection and classification, *ECCV*, pp. 589–600.

Wang, J. & Yagi, Y. (2006). Integrating shape and color features for adaptive real-time object tracking, *Proc. of Conf. on Robotics and Biomimetrics*, pp. 1–6.

Wang, J. & Yagi, Y. (2007). Discriminative mean shift tracking with auxiliary particles, *Proc. 8th Asian Conference on Computer Vision*, pp. 576–585.

Wang, J. & Yagi, Y. (2008a). Integrating color and shape-texture features for adaptive real-time tracking, *IEEE Trans. on Image Processing* 17(2): 235–240.

Wang, J. & Yagi, Y. (2008b). Patch-based adaptive tracking using spatial and appearance information, *Proc. International Conference on Image Processing*, pp. 1564–1567.

Wang, J. & Yagi, Y. (2009). Adaptive mean-shift tracking with auxiliary particles, *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics* 39(6): 1578–1589.

Zhao, T., Nevatia, R. & Wu, B. (2008). Segmentation and tracking of multiple humans in crowded environments, *IEEE Trans. Pattern Anal. Mach.* 30(7): 1198–1211.

Zhou, S. K., Georgescu, B., Comaniciu, D. & Shao, J. (2006). Boostmotion: boosting a discriminative similarity function for motion estimation, *Proc. of CVPR*, pp. 1761–1768.

7

Fuzzy Control System to Solve Coupling Between Tracking and Predictive Algorithms

Eduardo Parrilla, Jaime Riera, Juan R. Torregrosa and José L. Hueso
Universidad Politécnica de Valencia
Spain

1. Introduction

In this work, we analyze different topics in the study of object tracking. In Section 2, we study algorithms for tracking objects in a video sequence, based on the selection of landmark points representative of the moving objects in the first frame of the sequence to be analyzed. The movement of these points is estimated using a sparse optical-flow method. Methods of this kind are fast, but they are not very robust. Particularly, they are not able to handle the occlusion of the moving objects in the video. To improve the performance of optical flow-based methods, we propose the use of adaptive filters to predict the expected instantaneous velocities of the objects, using the predicted velocities as indicators of the performance of the tracking algorithm. The efficiency of this strategy to handle occlusion problems are tested with a set of synthetic and real video sequences. In (Parrilla et al., 2008) we develop a similar study using neural networks.

Video tracking deals with the problem of following moving objects across a video sequence (Trucco & Plakas, 2006), and it has many applications as, for example, traffic monitoring and control (Iñigo, 1989), (Erdem et al., 2003), robotic tasks, surveillance, etc. Simple algorithms for video tracking are based on the selection of regions of interest in the first frame of a video sequence, which are associated with moving objects to be followed and a system for estimating the movement of these regions across the sequence. More demanding methods impose constrains on the shape of the tracked objects (Erdem et al., 2003; Shin et al., 2005), or use methods to separate the moving objects from the background of the images (Ji et al, 2006; Wren et al., 1997). Most of these more demanding algorithms deal with partial occlusion of the moving objects, that is, the algorithms are not lost when the target temporarily disappears from the frame and they resume correctly when the target reappears. Generally, this kind of algorithms include an a priori training of the possible shape of the object to be followed and occlusion is handled following the movement of the contour. This is expensive from the computational point of view and makes these algorithms difficult to be implemented in a real-time application.

We have used adaptive filters (Haykin, 1986; Ljung, 1999) to predict the velocities of the object to track. These expected velocities are compared with the ones computed with the optical flow method and are used as indicators of the performance of the method. If the optical flow method fails to compute reasonable values for the velocities, the velocity values predicted by the filter can be used as reliable values for the velocities of the object. A system of this kind can be useful in different applications, such as economic systems for traffic monitoring and control using images provided by cameras installed in the roads (Iñigo, 1989).

The critical point of the system for solving the occlusion problems is the coupling between optical flow and predictive algorithms. This coupling is governed by different parameters. In Section 3, we propose the use of a fuzzy control system to solve the mentioned coupling.

Section 4 is devoted to stereoscopic video sequences. We analyze simple algorithms for 3D tracking objects in a stereo video sequence, by combining optical flow and stereo vision. This method is not able to handle the occlusion of the moving objects when they disappear due to an obstacle. To improve the performance of this method, we propose the use of adaptive filters or fuzzy control techniques to predict the expected instantaneous velocities of the objects.

2. Handling occlusion in optical flow algorithms for object tracking

2.1 Optical flow

For computing the optical flow of a frame of a video sequence, it is necessary to assume that intensity variations of the image are only caused by displacements of the objects, without considering other causes such as changes of illumination. This hypothesis, proposed initially by Horn and Schunck (Horn & Schunck, 1981), supposes that present intensity structures in the image, at local level, stay constant throughout the time, at least during small temporal intervals. Formally, if $I(\bar{x},t) = I(x(t),y(t),t)$ denotes the continuous space-time intensity function of an image, it has to be fulfilled that

$$I(x(t),y(t),t) \approx I(x(t+\Delta t),y(t+\Delta t),t+\Delta t). \tag{1}$$

By expanding the right-hand side of equation (1) using the Taylor series yields

$$I(t) \approx I(t) + \frac{dI(t)}{dt}\Delta t + O^2(\Delta t) \tag{2}$$

that is,

$$\frac{dI}{dt} = 0, \tag{3}$$

where $O^2(\Delta t)$ are the 2^{nd} and higher order terms, which are assumed to be negligible. Finally, applying the Chain rule, we obtain the optical flow equation

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{4}$$

or, in another way

$$I_x u + I_y v + I_t = 0, \tag{5}$$

where I_x, I_y are the components of the spatial gradient ∇I, I_t denotes partial differentiation with respect to time and $\mathbf{v} = (u,v)$ denotes the components of the image velocity field. The distribution of the velocity in each point of the image is known as the optical flow.

Constraint (5) is not sufficient to solve the optical flow equation for a given frame. The problem is ill-posed because we have to compute two components, u and v, and we only have one equation. This phenomenon is known as the aperture problem. It is necessary to consider additional restrictions to the problem to estimate the motion at every image location. There are many techniques for computing the optical flow, which differ one from each other in the different assumptions that are taken into account to determine the solution. In our analysis we use the technique proposed by Lucas and Kanade in 1981 (Lukas & Kanade, 1981).

2.1.1 Lucas and Kanade algorithm

The algorithm proposed by Lucas and Kanade uses a local constant model of velocity, obtaining good results in different applications (Bourel et al., 2001; Parrilla et al., 2005;?). It is based on supposing the values of optical flow in a local neighborhood, R, to be constant, minimizing the following expression

$$\sum_{(x,y)\in R} W^2(x,y)(\nabla I(x,y,t)\mathbf{v} + I_t(x,y,t))^2, \tag{6}$$

where $W(x,y)$ denotes a window function and R is a spatial neighborhood. Equation (6) has the following solution (Barron et al., 1994)

$$A^T W^2 A \cdot \mathbf{v} = A^T W^2 b, \tag{7}$$

where, for n points (x_i, y_i) in R at a moment t,

$$A = \left[\nabla I(x_1,y_1), \cdots, \nabla I(x_n,y_n)\right]^T \tag{8}$$

$$W = diag\left[W(x_1,y_1), \cdots, W(x_n,y_n)\right] \tag{9}$$

$$b = -\left[I_t(x_1,y_1), \cdots, I_t(x_n,y_n)\right]^T \tag{10}$$

We can track a window from frame to frame if the system (7) represents good measurements, and if it can be solved reliably. This means that the 2 × 2 coefficient matrix $A^T W^2 A$ of the system must be both above the image noise level and well conditioned. In turn, the noise requirement implies that both eigenvalues of $A^T W^2 A$ must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. In practice, if the two eigenvalues of $A^T W^2 A$ are λ_1 and λ_2, we accept a window if $min(\lambda_1, \lambda_2) > \lambda$, where λ is a predefined threshold.

2.2 Adaptive filters

To obtain an indicator of the performance of the tracking algorithm, each component of the velocity estimated by using optical flow, for the different frames of a sequence, is considered as a time series. We use adaptive filters to predict instantaneous velocities in order to compare them with the velocities obtained by using the optical flow algorithm.

An adaptive filter is a filter which self-adjusts its transfer function according to an optimizing algorithm. Because of the complexity of the optimizing algorithms, most adaptive filters are digital filters that perform digital signal processing and adapt their performance based on the input signal. In our work, we concentrate on the recursive least squares (RLS) filter (Ljung, 1999).

2.2.1 RLS filter

We start from a signal x_n, $n = 0, \ldots, N_T$, and we assume for the signal an AR(p) model

$$x_k = -(a_1)_k\, x_{k-1} - (a_2)_k\, x_{k-2} - \cdots - \left(a_p\right)_k x_{k-p} + \varepsilon_k\,. \tag{11}$$

Assuming this model for $k = p, \ldots, N$, we obtain the system of equations

$$X_N = C_N a_N + \epsilon_N\,, \tag{12}$$

where

$$X_N = \begin{bmatrix} x_p \\ x_{p+1} \\ \vdots \\ x_N \end{bmatrix}, \; C_N = \begin{bmatrix} x_{p-1} & x_{p-2} & \cdots & x_0 \\ x_p & x_{p-1} & \cdots & x_1 \\ \vdots & & & \vdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-p} \end{bmatrix},$$

$$a_N = \begin{bmatrix} -\left(a_1\right)_N \\ -\left(a_2\right)_N \\ \vdots \\ -\left(a_p\right)_N \end{bmatrix}, \; \epsilon_N = \begin{bmatrix} \varepsilon_p \\ \varepsilon_{p+1} \\ \vdots \\ \varepsilon_N \end{bmatrix}.$$

The model coefficients, $\left(a_j\right)_N$, are obtained as those coefficients that make minimum the error function

$$J_N = \frac{1}{2}\sum_{k=p}^{N} \lambda^{N-k}\varepsilon_k\varepsilon_k =$$
$$= \frac{1}{2}\epsilon_N^T\Lambda_N\epsilon_N = \frac{1}{2}\left(X_N - C_N a_N\right)^T \Lambda_N \left(X_N - C_N a_N\right) . \tag{13}$$

We have introduced the weights matrix

$$\Lambda_N = \begin{bmatrix} \lambda^{N-p} & 0 & \cdots & 0 \\ 0 & \lambda^{N-p-1} & & \vdots \\ \vdots & \vdots & & \\ 0 & \cdots & & 1 \end{bmatrix},$$

where $0 < \lambda \leq 1$ is a constant that controls the importance on the error function J_N of the older samples of the signal, and is called the forgetting factor of the method. The coefficients a_N are the solutions of the equation

$$\frac{\partial J_N}{\partial a_N} = 0 ,$$

that is,

$$a_N = \left[C_N^T\Lambda_N C_N\right]^{-1} C_N^T\Lambda_N X_N . \tag{14}$$

Let us consider now $N+1$ samples of the signal. The system (12) becomes

$$\begin{bmatrix} X_N \\ x_{N+1} \end{bmatrix} = \begin{bmatrix} C_N \\ c_{N+1}^T \end{bmatrix} a_{N+1} + \epsilon_{N+1} , \tag{15}$$

where

$$c_{N+1}^T = \left[x_N, x_{N-1}, \cdots, x_{N-p+1}\right] .$$

a_{N+1} will be the coefficients that minimize the new error function

$$J_{N+1} = \frac{1}{2}\sum_{k=p}^{N+1} \lambda^{N+1-k}\varepsilon_n\varepsilon_N = \frac{1}{2}\epsilon_{N+1}^T\Lambda_{N+1}\epsilon_{N+1} .$$

The solution of this problem is

$$a_{N+1} = \left[C_{N+1}^T \Lambda_{N+1} C_{N+1}\right]^{-1} C_{N+1}^T \Lambda_{N+1} X_{N+1} \,. \tag{16}$$

To obtain the inverse of matrix $C_{N+1}^T \Lambda_{N+1} C_{N+1}$ is an expensive process from the computational point of view, especially when a large number of samples are considered. To overcome this problem, we take into account that

$$\begin{aligned} C_{N+1}^T \Lambda_{N+1} C_{N+1} &= \begin{bmatrix} C_N^T & c_{N+1} \end{bmatrix} \begin{bmatrix} \lambda \Lambda_N & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} C_N \\ c_{N+1}^T \end{bmatrix} = \\ &= C_N^T \lambda \Lambda_N C_N + c_{N+1} c_{N+1}^T \,. \end{aligned}$$

By using the inversion lemma (Ogata, 1987)

$$(A + BD)^{-1} = A^{-1} - A^{-1} B \left(I + DA^{-1} B\right)^{-1} DA^{-1} \,,$$

with

$$A = \lambda C_N^T \Lambda_N C_N \,, \;\; B = c_{N+1} \,, \;\; D = c_{N+1}^T \,,$$

we rewrite

$$\begin{aligned} &\left[C_{N+1}^T \Lambda_{N+1} C_{N+1}\right]^{-1} = \\ &\frac{1}{\lambda} \left(\left[C_N^T \Lambda_N C_N\right]^{-1} - \frac{\left[C_N^T \Lambda_N C_N\right]^{-1} c_{N+1} c_{N+1}^T \left[C_N^T \Lambda_N C_N\right]^{-1}}{\lambda + c_{N+1}^T \left[C_N^T \Lambda_N C_N\right]^{-1} c_{N+1}} \right) \,. \end{aligned} \tag{17}$$

Introducing the notation

$$P_N = \left[C_N^T \Lambda C_N\right]^{-1} \,, \;\; K_{N+1} = \frac{P_N c_{N+1}}{\lambda + c_{N+1}^T P_N c_{N+1}} \,,$$

we obtain the recurrence

$$P_{N+1} = \frac{1}{\lambda} \left(P_N - K_{N+1} c_{N+1}^T P_N\right) \,. \tag{18}$$

Therefore, using (16), we can write

$$\begin{aligned} a_{N+1} &= P_{N+1} \lambda C_N^T \Lambda_N X_N + P_{N+1} c_{N+1} x_{N+1} \\ &= P_N C_N^T \Lambda_N X_N - K_{N+1} c_{N+1}^T P_N C_N^T \Lambda_N X_N \\ &\quad + \frac{1}{\lambda} P_N c_{N+1} x_{N+1} - \frac{1}{\lambda} K_{N+1} c_{N+1}^T P_N c_{N+1} x_{N+1} \\ &= a_N - K_{N+1} c_{N+1}^T a_N + \frac{1}{\lambda} P_N c_{N+1} x_{N+1} \\ &\quad - \frac{1}{\lambda} \frac{P_N c_{N+1} c_{N+1}^T P_N c_{N+1} x_{N+1}}{\lambda + c_{N+1}^T P_N c_{N+1}} \\ &= a_N + K_{N+1} \left(x_{N+1} - c_{N+1}^T a_N\right) \,. \end{aligned}$$

To use the recursive least square (RLS) method, we first perform an initialization step, considering N_{in} samples of the signal and computing

$$\begin{aligned} P_{N_{in}} &= \left[C_{N_{in}}^{T} \Lambda_N C_{N_{in}} \right]^{-1} \; , \\ a_{N_{in}} &= P_N C_{N_{in}}^{T} \Lambda_{N_{in}} X_{N_{in}} \; . \end{aligned}$$

After this step, the following recursion is used

$$\begin{aligned} K_{N+1} &= \frac{P_N c_{N+1}}{\lambda + c_{N+1}^{T} P_N c_{N+1}} \; , \\ P_{N+1} &= \frac{1}{\lambda} \left(P_N - K_{N+1} c_{N+1}^{T} P_N \right) \; , \\ a_{N+1} &= a_N + K_{N+1} \left(x_{N+1} - c_{N+1}^{T} a_N \right) \; , \end{aligned} \qquad (19)$$

for $N = N_{in}, \ldots, N_T$, obtaining in this way a new set of coefficients for the AR(p) model each time a new sample of the signal is considered. This kind of autoregressive method where the coefficients change in time is called a Dynamic Autoregressive method of order p, DAR(p), (Young, 1999).

In order to combine adaptive filters with optical flow calculation, we follow the next steps:

1. We calculate velocities for N_{in} frames of each sequence by using Lucas and Kanade algorithm and we use this samples to inicialize the filter coefficients.
2. For the N-th frame, we calculate the velocity v_N in the following way:

   a. We calculate v_N^{of} by using optical flow.

   b. We estimate v_N^{af} by using an adaptive filter.

   c. If $|v_N^{of} - v_N^{af}| < tol_N$, then $v_N = v_N^{of}$. Else $v_N = v_N^{af}$, where tol_N are certain tolerance factors, which depend on the sequence to be analyzed.

2.3 Numerical results

The method explained above has been used to analyze synthetic and real video sequences. The result has been satisfactory, since the algorithm has allowed to track the objects along the whole sequence.

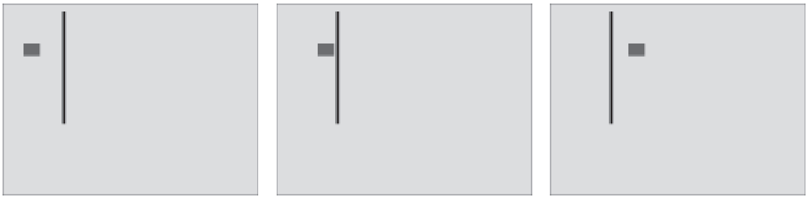


Fig. 1. Moving object

In all the examples, we have used windows of 7 × 7 pixels to calculate optical flow by using Lucas and Kanade algorithm. The point to track has been selected by indicating to the program the zone where the object that we want to track is. In a final application, we would

select a greater number of points of the image, discarding those that did not have movement. In this way, we would only consider the points that belong to objects in movement.
Parameter values are a critical factor for the correct operation of the proposed algorithms. A very small value of the tolerance will cause the methods to select predicted velocities when there is no obstacle. On the other hand, a large value of the tolerance will make that the methods can not detect any occlusion.
We will analyze the occlusion problem by using the example observed in figure 1. It consists in an object with a horizontal movement and a velocity *vel* pixels per frame.

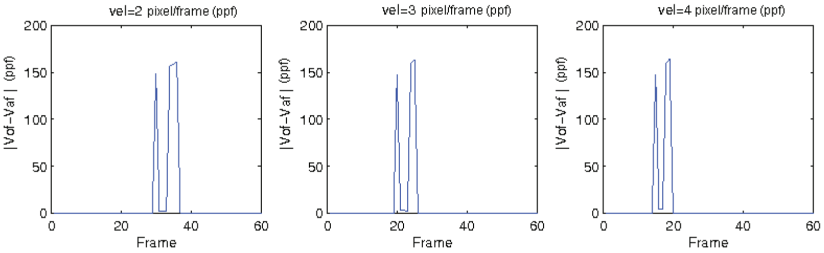


Fig. 2. Difference between optical flow and prediction

We have calculated $|v_x^{of} - v_x^{af}|$ for $vel = 2$, $vel = 3$ and $vel = 4$ pixels per frame (ppf) for the different frames of the sequence. The results can be observed in figure 2. We can see that there are two different peaks for each value of *vel* that corresponds with the input and output frontiers of the obstacle. In these points there are two discontinuities that produce a large error in the calculation of the optical flow. These peaks will always be detected. If we analyze what happens in the region corresponding to the transition of the object through the obstacle, we obtain the figure 3.

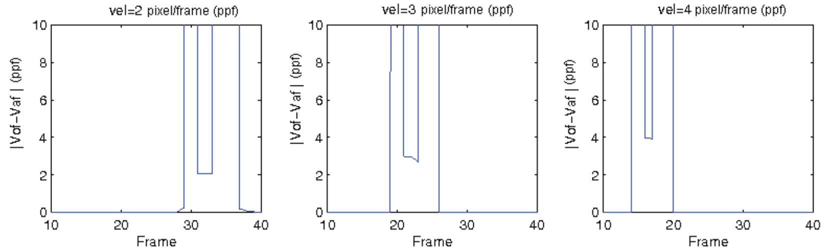


Fig. 3. Difference between optical flow and prediction (zoom)

As we can see, the values of $|v_x^{of} - v_x^{af}|$ are very similar to the velocity values of the object because the optical flow algorithm considers that obstacle is a static object so $v_{x,y}^{of} = 0$ in this region. In this way, we will use a variable value of tolerance in our algorithm for each frame N

$$tol_N = k\,|v_{N-1}| \tag{20}$$

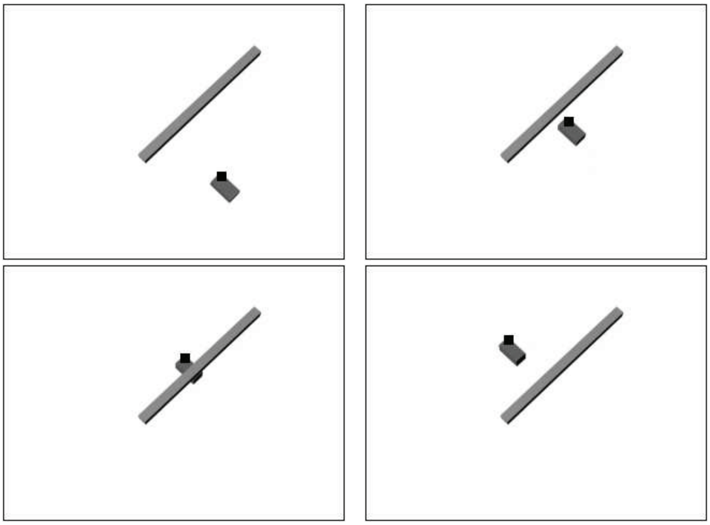


Fig. 4. RLS tracking in a synthetic sequence

In the different sequences that have been analyzed we have used a value of $k = 0.75$.
In figure 4, we can observe differents frames of a synthetic video sequence that consists in an object that follows a curved trajectory passing under an obstacle. Another example is shown in figure 5. We can see an urban route where there is a partial vehicle occlusion because of a street light. In these two examples, occlusions have been handled by using a RLS filter. We have used a value of $N_{in} = 7$ frames, an order filter of 2 and a forgetting factor of $\lambda = 0.99$.
In both sequences, without using the RLS filter, an error occurs when the object arrives at the obstacle. As we can see, by using the RLS filter, objects can be tracked along the whole sequences.
This method has demonstrated to be very efficient to handle occlusion because of its adaptability and tolerance to noisy data.

3. Fuzzy control for obstacle detection in object tracking

As we have said in the former section, tolerance values are a critical factor for the correct operation of the proposed algorithms. A very small value of the tolerance will cause the methods to select predicted velocities when there is no obstacle. On the other hand, a large value of the tolerance will make that the methods can not detect any occlusion. For this reason, the choice of parameter k is very critical. In this section, we propose to use a variable value for parameter k controlled by a fuzzy system, instead of selecting its value a priori.

3.1 Fuzzy controller

A fuzzy control system is a control system based on fuzzy logic (Zadeh, 1965). Fuzzy control provides a formal methodology for representing, manipulating and implementing a human's heuristic knowledge about how to control a system (Passino & Yurkovich, 2000). The fuzzy controller block diagram is given in figure 6, where we can see a fuzzy controller embedded in a closed-loop control system.

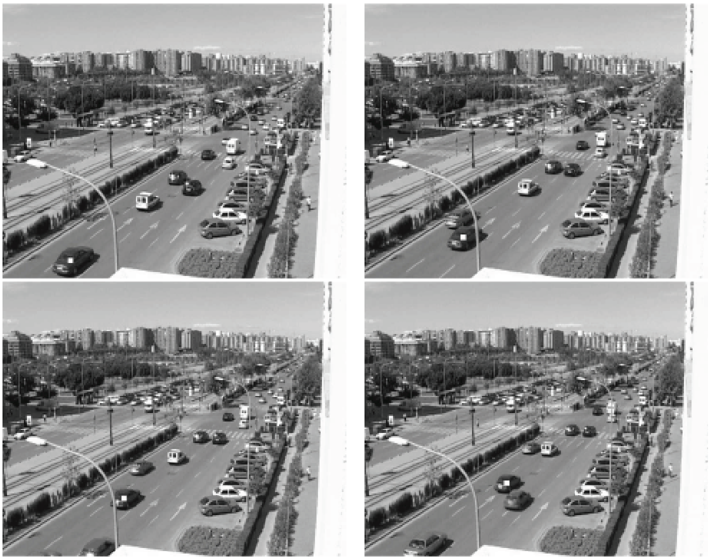


Fig. 5. RLS tracking in a real sequence

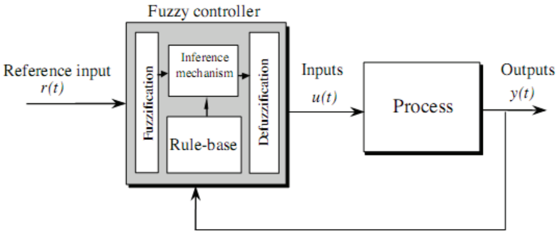


Fig. 6. Fuzzy controller architecture

To design a fuzzy controller, we have to specify the different blocks of the system:

- Input variables LHS
- Output variables RHS
- Input fuzzification method
- Fuzzy rules
- Inference engine parameters
- Defuzzification method

3.1.1 Fuzzification

The function of the fuzzificator is the conversion of the real input x into a fuzzy set. To do this work, it is neccessary to define a set of membership functions. In figure 7, we can see an example of this set.

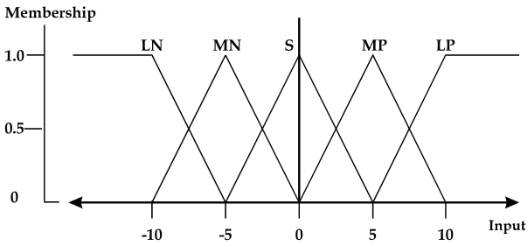


Fig. 7. Membership functions. LN: large negative, MN: medium negative, S: small, MP: medium positive, LP: large positive

The values of each membership function are labeled by $\mu(x)$ and determined by the input value x and the shape of each membership function. In the example of figure 7, if we have an input value $x = 2$, the set of fuzzy variables will be:
$\mu_{LN} = 0$
$\mu_{MN} = 0$
$\mu_S = 0.6$
$\mu_{MP} = 0.4$
$\mu_{LP} = 0$

3.1.2 Fuzzy decision blocks

Fuzzy control requires a specification of the rules in terms of linguistic variables, that are conditional statements of type IF-THEN. These rules relate input fuzzy values to output fuzzy value. A set of fuzzy rules R^l is:

$$R^l : \ IF\ x_1\ is\ F_1^l\ and\ ...\ and\ x_n\ is\ F_n^l\ THEN\ y\ is\ G^l,$$

where x_i and y are input and output linguistic variables, and F_i^l and G^l are fuzzy sets.
In this case, we consider a system with multiples inputs and one output. A system with multiple outputs can be separated in a set of simple fuzzy systems.

3.1.3 Inference engine

The inference engine produces a global output $\mu_{out}(y)$, by using the set of rules and considering the membership degree of the input with each membership function. The global output $\mu_{out}(y)$ is a fuzzy set that is calculated from a set of membership functions for the output y.
Continuing the previous example, if we have the two following rules:
If x is small (S), Then y is small (S)
If x is medium positive (MP), Then y is positive (P)
, and we know that
$\mu_S = 0.6$
$\mu_{MP} = 0.4$
by using the set of membership functions for the output y observed in figure 8, we will obtain the global output of the figure 9.
Depending of the method of defuzzification, we can consider an only global output $\mu_{out}(y)$ as the union of all the contributions, or a global output $\mu_{out}^k(y)$ for each contribution.

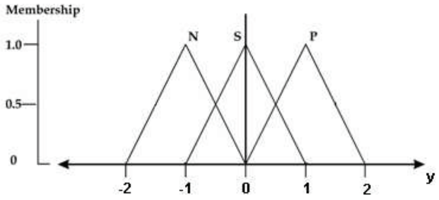


Fig. 8. Membership output functions. N: negative, S: small, P: positive

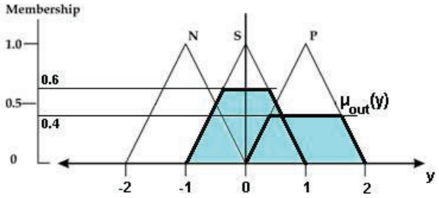


Fig. 9. Global output

3.1.4 Defuzzification

The defuzzification is a process to obtain a real value u^* representative of $\mu_{out}(u)$. There are different methods for defuzzification and method selection is very important for the efficiency and precision of the controller.
The most used defuzzification techniques are:

- COA o Center of Areas

  This technique calculates u^* as the geometric center of the output fuzzy value $\mu_{out}(u)$, where $\mu_{out}(u)$ is the combination of all the contributions. u^* divides the area in two regions with the same area. Some problems of this method are the computational cost, the preference for central values and that the overlapping area is only counted once.

- COS o Center of Sums

  This method operates separately for each contribution. The overlapping areas are considered more than once. It is efficient and it is the most used method.

- MOM o Mean of Maxima

  This technique calculates the mean value of points with maxima membership degree in the global output $\mu_{out}(u)$. It is very fast, but it does not consider the shape of $\mu_{out}(u)$.

- CA o Center Average

  This method operates separately for each contribution and calculates a pondered average of the centers of the contributions.

3.2 Obstacle detection

In our system, we want the value of tol to be small when there is an obstacle and large in any other case. The value of tol can be controlled by the variable k. In this way, we have designed a fuzzy system to control the value of k.

The system has two inputs, $\Delta\varepsilon$ and qV, and one output, k.
The variable $\Delta\varepsilon$ is the increment error (EI). If we call error ε to the absolute value of the difference between the velocity calculated by optical flow and the velocity estimated by the adaptive filter, we have that:

$$\varepsilon_n = \left| v_n^{of} - v_n^{af} \right| , \tag{21}$$

$$\Delta\varepsilon_n = \varepsilon_n - \varepsilon_{n-1} \,. \tag{22}$$

The variable qV is a binary variable that indicates if we are using the velocities calculated by optical flow in the previous frames ($qV = 1$) or the ones estimated by the adaptive filter ($qV = 0$).
In figure 10, we can see the set of membership functions used to the fuzzification of the input $\Delta\varepsilon$. Fuzzy values for this input can be negative big (NB), medium (M) or positive big (PB). NB values correspond to the final of the peaks that exist in the frontiers of the obstacle as we can see in figure 2, PB values correspond to the beginning of these peaks and M value correspond to the other situations.

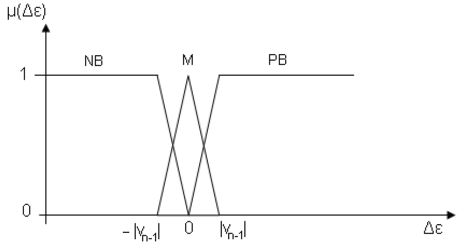


Fig. 10. Membership functions for the input $\Delta\varepsilon$

In this system, the value of variable k must be high (H), to select the velocity calculated by optical flow, when there is no occlusion (EI=M, qV=1) and when the target is in the output of the obstacle (EI=NB, qV=0). In all the other cases, the value of k will be low (L) to select the velocity estimated by the adaptive filter.
In this way, the system fuzzy rules are the following:

- IF EI is NB and qV=0 THEN k is H
- IF EI is M and qV=0 THEN k is L
- IF EI is PB and qV=0 THEN k is L
- IF EI is NB and qV=1 THEN k is L
- IF EI is M and qV=1 THEN k is H
- IF EI is PB and qV=1 THEN k is L

The inference engine calculates the global output $\mu_{out}(k)$ by using these conditions and the set of output membership functions that we can observe in figure 11. In this way, the value of the output variable k can oscillate between 0.5 and 1.
Finally, the defuzzificator calculates from the global output $\mu_{out}(k)$ the real value for the variable k by using the center of sums technique, that provides fast and satisfactory results.

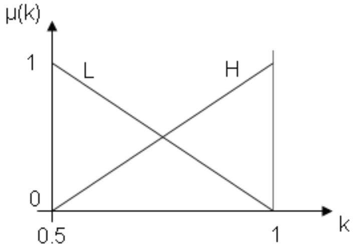


Fig. 11. Membership functions for the output k

3.3 Results

The algorithm exposed has been tested in different video sequences, studying synthetic and real traffic videos. We have used windows of 7×7 pixels for Lucas and Kanade algorithm, and a value of $N_{in} = 7$ frames, an order of 2 and a forgetting factor of $\lambda = 0.99$ for the adaptive filter.

In figure 12, we can observe differents frames of a synthetic video sequence designed with 3D Studio Max ® that consists in an object that follows a curved trajectory passing under an obstacle.

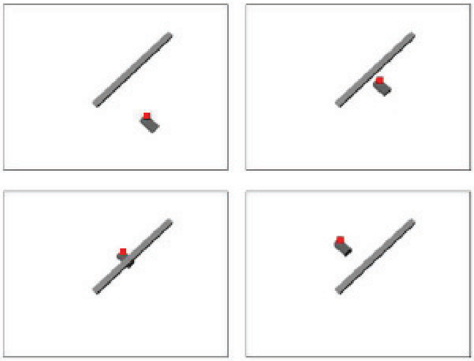


Fig. 12. Fuzzy object tracking in a synthetic sequence

In this example, the occlusion is produced in frames 37-43. As we can observe in figure 13, the value of k decreases in those frames. In this way, the predicted velocity is selected while the target is hidden by the obstacle and the object is correctly tracked along all the sequence.

The same result is obtained analyzing the example shown in figure 14. In this figure, we can see an urban route where there is a partial vehicle occlusion because of a street light.

In this case, the occlusion occurs in frames 21-27 and, as we can see in figure 15, k decreases in this position. On the other hand, in this figure, we can observe that the value of k also decreases in frames 51-54. This change is due to the fact that there is an error in the optical flow calculation. The performance of the system in this point is very interesting since it is able to correct possible errors in the optical flow calculation, besides handling occlusion.

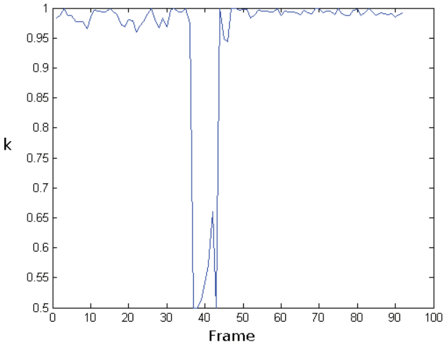


Fig. 13. *k* values in synthetic sequence

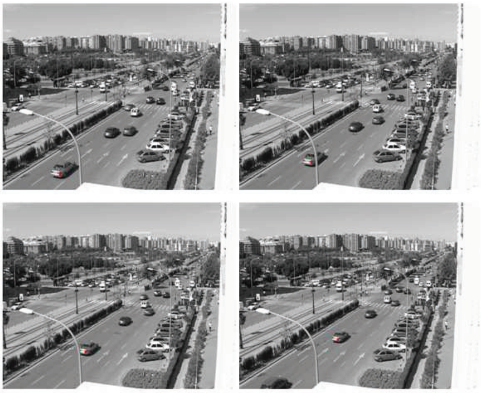


Fig. 14. Object tracking in a real sequence

4. Handling occlusion in object tracking in stereoscopic video sequences

Stereo vision is a part of the field of computer vision that uses two or more cameras to obtain two or more images from which distance information can be obtained by using the triangulation principle (Cochran & Medioni, 1992). We can obtain three-dimensional information of static scenes by using this technique. On the other hand, we can use optical flow algorithms for object tracking in two dimensional video sequences along the time (Trucco & Plakas, 2006).

In (Parrilla et al., 2005), we have studied a system that combines stereoscopic vision and optical flow algorithms for object tracking in a three-dimensional space. We select a set of points to be tracked in the first frame of one video of the stereo sequence and, by using optical flow algorithms, we track them in that sequence. For each frame, we calculate the disparity of these points by using the other video and stereo algorithms. In this way, we know the position of each point in a three dimensional space (disparity) and along the time (optical flow). One of the most important problems of this technique is that this method is not able to handle the occlusion of the moving objects.

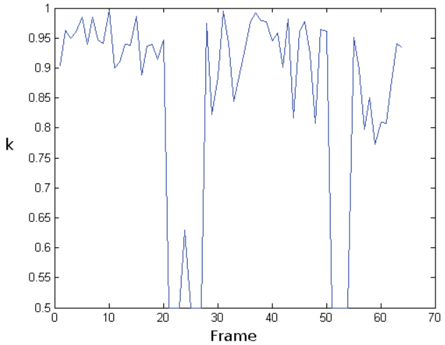


Fig. 15. k values in real sequence

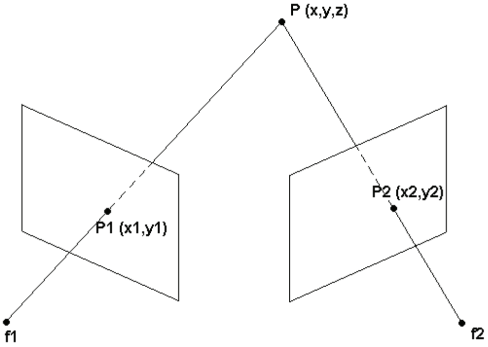


Fig. 16. Triangulation principle

For solving this occlusion problem, we propose the use of adaptive filters (Haykin, 1986; Ljung, 1999) to predict the expected 3D velocities of the objects. In this section, we combine adaptive filters, optical flow algorithms and stereo vision, in order to handle occlusion in object tracking in stereoscopic video sequences.

Stereo vision (Cochran, 1990) consists in, given a point P from which we know their projections P_1 and P_2 in the images of a pair of cameras, calculating its position in the space by using the triangulation principle (see figure 16). If we designate the cartesian coordinates of the projections as (x_1, y_1) and (x_2, y_2) in the respective coordinate systems of the cameras, we can obtain their relationship with the coordinates (x, y, z) of point P.

Without loss of generality, we will assume that the coordinate system of the left camera and the real world is the same and, to simplify the problem, we will consider that the axes of the cameras are parallel and the coordinate system of the right camera is similar but moved a distance B along the axis x. We will also assume that the two cameras are equal with a focal length f. In this way we have the following equations:

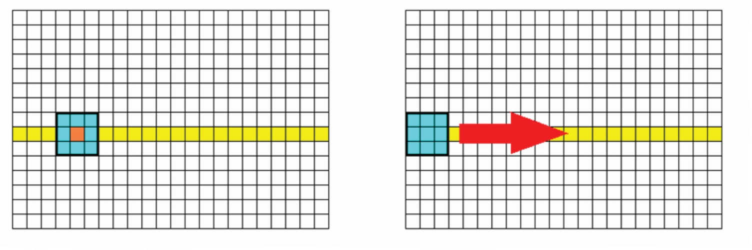


Fig. 17. Correlation search

$$\begin{cases} \dfrac{x_1}{f} = \dfrac{x}{z} \\ \dfrac{x_2}{f} = \dfrac{x-B}{z} \end{cases} \tag{23}$$

By solving the system of equations we obtain the following results:

$$x = \frac{x_1 B}{x_1 - x_2}\,, \tag{24}$$

$$z = \frac{Bf}{x_1 - x_2}\,. \tag{25}$$

By proceeding in a similar way we also can calculate that:

$$y = \frac{y_1 B}{x_1 - x_2}\,. \tag{26}$$

The main problem of stereo vision is to find the correspondences between the points of the different images (disparity), i.e, given the projection P_1 in the left image, to find what point of the right image corresponds to the projection P_2, to situate in the space the point P.

We have programmed a correlation search method that has provided satisfactory results. This technique selects a neighborhood around the point P_1 and calculates the correlation with a window that will displace over the different points of the right image (see figure 17). Finally we select that point where the correlation is maximum.

Because of the geometry of the problem, we only have to displace the window along a line (epipolar line) because projections P_1 and P_2 will be situated in the same horizontal (see figure 18).

4.1 Two-dimensional adaptive filter

To obtain an indicator of the performance of the tracking algorithm, each component of the velocity estimated by using optical flow and stereo vision, for the different frames of the sequences, is considered as a time series. We use adaptive filters to predict instantaneous velocities in order to compare them with the velocities obtained by using optical flow and stereo vision. For the component v_y, we use a simple adaptive filter because this component is the same in the two video sequences. On the other hand, there are two components v_{Lx} and v_{Rx} that correspond to the velocities in the left and right images. In this case, we will adapt the RLS filter to operate with two signals simultaneously.

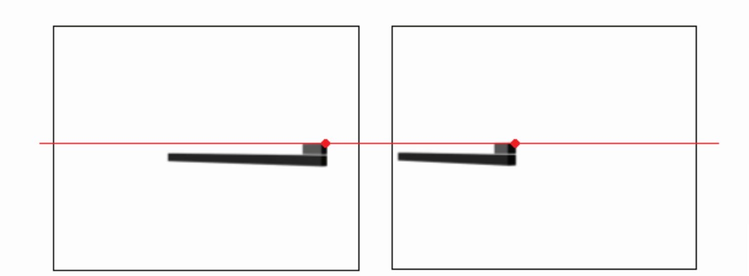


Fig. 18. Epipolar line

We start from two signals x_{Ln} and x_{Rn}, $n = 0, \ldots, N_T$, and we assume for the signals an AR(p) model

$$\begin{aligned} x_{Lk} = &-(a_1)_k\, x_{Lk-1} - (a_2)_k\, x_{Rk-1} - \cdots \\ &\cdots - \left(a_{2p-1}\right)_k x_{Lk-p} - \left(a_{2p}\right)_k x_{Rk-p} + \varepsilon_k\,, \\ x_{Rk} = &-(b_1)_k\, x_{Lk-1} - (b_2)_k\, x_{Rk-1} - \cdots \\ &\cdots - \left(b_{2p-1}\right)_k x_{Lk-p} - \left(b_{2p}\right)_k x_{Rk-p} + \varepsilon'_k\,, \end{aligned} \tag{27}$$

In this case, x_{Lk} and x_{Rk} will depend on the previous samples of both signals. Assuming this model for $k = p, \ldots, N$, we obtain the system of equations

$$\begin{bmatrix} X_{LN} \\ X_{RN} \end{bmatrix} = C_N \begin{bmatrix} a_N \\ b_N \end{bmatrix} + \epsilon_N\,,$$

where

$$X_{LN} = \begin{bmatrix} x_{Lp} \\ x_{Lp+1} \\ \vdots \\ x_{LN} \end{bmatrix}, \quad X_{RN} = \begin{bmatrix} x_{Rp} \\ x_{Rp+1} \\ \vdots \\ x_{RN} \end{bmatrix}, \quad C_N = \begin{bmatrix} x_{Lp-1} & x_{Rp-1} & \cdots & x_{L0} & x_{R0} \\ x_{Lp} & x_{Rp} & \cdots & x_{L1} & x_{R1} \\ \vdots & & & \vdots & \\ x_{LN-1} & x_{RN-1} & \cdots & x_{LN-p} & x_{RN-p} \end{bmatrix},$$

$$a_N = \begin{bmatrix} -(a_1)_N \\ -(a_2)_N \\ \vdots \\ -\left(a_{2p}\right)_N \end{bmatrix}, \quad b_N = \begin{bmatrix} -(b_1)_N \\ -(b_2)_N \\ \vdots \\ -\left(b_{2p}\right)_N \end{bmatrix}, \quad \epsilon_N = \begin{bmatrix} \varepsilon_p \\ \vdots \\ \varepsilon_N \\ \varepsilon'_p \\ \vdots \\ \varepsilon'_N \end{bmatrix}.$$

This system of equations can be considered as two separated systems of equations with the same matrix C_N, that contains samples of both signals x_{LN} and x_{RN}

$$\begin{aligned} X_{LN} &= C_N a_N + \epsilon_N\,, \\ X_{RN} &= C_N b_N + \epsilon'_N\,. \end{aligned} \tag{28}$$

If we apply the RLS filter theory to these systems of equations, we will obtain an adaptive filter for two correlated signals. To use this filter, we first perform an initialization step, considering N_{in} samples of the signals and computing

$$\begin{aligned} P_{N_{in}} &= \left[C_{N_{in}}^T \Lambda_N C_{N_{in}}\right]^{-1} , \\ a_{N_{in}} &= P_N C_{N_{in}}^T \Lambda_{N_{in}} X_{LN_{in}} , \\ b_{N_{in}} &= P_N C_{N_{in}}^T \Lambda_{N_{in}} X_{RN_{in}} . \end{aligned}$$

After this step, the following recursion is used

$$\begin{aligned} K_{N+1} &= \frac{P_N c_{N+1}}{\lambda + c_{N+1}^T P_N c_{N+1}} , \\ P_{N+1} &= \frac{1}{\lambda}\left(P_N - K_{N+1} c_{N+1}^T P_N\right) , \\ a_{N+1} &= a_N + K_{N+1}\left(x_{LN+1} - c_{N+1}^T a_N\right) , \\ b_{N+1} &= b_N + K_{N+1}\left(x_{RN+1} - c_{N+1}^T b_N\right) , \end{aligned} \tag{29}$$

where

$$c_{N+1}^T = \left[x_{LN}, x_{RN}, x_{LN-1}, x_{RN-1}, \cdots, x_{LN-p+1}, x_{RN-p+1}\right] ,$$

for $N = N_{in}, \ldots, N_T$, obtaining in this way a new set of coefficients for the AR(p) model each time a new sample of the signals is considered.
In this way, we can estimate the following sample of the signals

$$\begin{aligned} x_{LN+1} &= c_{N+1}^T a_{N+1}, \\ x_{RN+1} &= c_{N+1}^T b_{N+1}. \end{aligned} \tag{30}$$

4.2 Handling occlusion

In order to predict the velocities of the objects, we follow the next steps:

1. We calculate velocities for N_{in} frames of each sequence by using Lucas and Kanade algorithm and stereo vision, and we use this samples to inicialize the filter coefficients.
2. For the N-th frame, we calculate the velocities v_{LN} and v_{RN} in the following way:
   a. We calculate v_{LN}^{of} and v_{RN}^{of} by using optical flow and stereo vision.
   b. We estimate v_{LN}^{es} and v_{RN}^{es} by using an adaptive filter.
   c. If $|v_{L,RN}^{of} - v_{L,RN}^{es}| < tol_{L,RN}$, then $v_{L,RN} = v_{L,RN}^{of}$. Else $v_{L,RN} = v_{L,RN}^{es}$
3. We use $v_{L,RN}$ and $N_{in} - 1$ previous samples to update the filter coefficients.

Tolerance values are a critical factor for the correct operation of the proposed algorithm. A very small value of the tolerance will cause the method to select predicted velocities when there is no obstacle. On the other hand, a large value of the tolerance will make that the method can not detect any occlusion. In Section 2, we have demonstrated that an optimum value of tolerance is

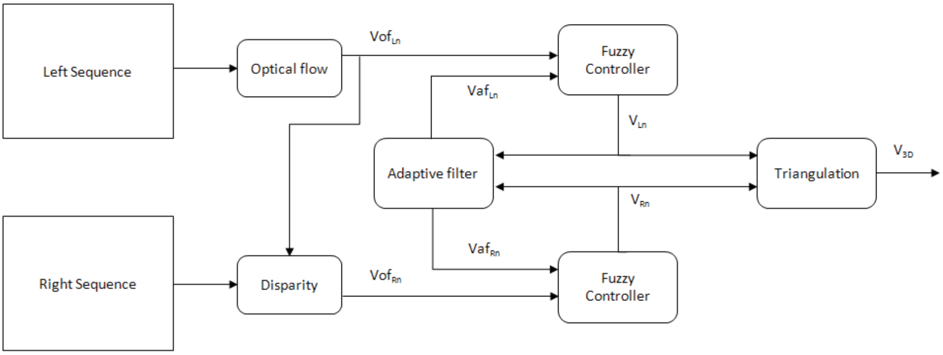


Fig. 19. Tracking system diagram

$$tol_{L,RN} = k\,|v_{L,RN-1}|\,. \tag{31}$$

In the same way as in the case of two-dimensional tracking, the values of k will be controlled by a fuzzy system.

In Figure 19, we can see a diagram of the entire system. An optical flow algorithm is applied to the left sequence in order to perform a 2D tracking of the object. From the obtained results, by calculating the disparity, we track the object in the right sequence. Moreover, the velocity values are predicted by using the adaptive filter. The velocity values calculated using optical flow and the predicted values are the inputs of two identical fuzzy controllers that are responsible for deciding which velocities are selected according to whether there is occlusion. These selected velocities will also be the inputs of the adaptive filter that will update the coefficients in order to make the prediction in the next frame. Once the object has been tracked in both sequences, using the triangulation principle, we can obtain the 3D velocity of the object.

4.3 Numerical results

The method explained above has been used to analyze synthetic and real video sequences. The result has been satisfactory, since the algorithm has allowed to track the objects along the whole sequence.

In all the examples, we have used windows of 7 × 7 pixels to calculate optical flow and disparity.

In Figure 20, we can observe differents frames of a synthetic video sequence that consists in an object that moves over a rail and passes behind an obstacle. Another example is shown in Figure 21. We can see an urban route where there is a partial vehicle occlusion because of a street light. In these two examples, occlusions have been handled by using an adaptive filter. We have used a value of $N_{in} = 7$ frames, an order filter of 2 and a forgetting factor of $\lambda = 0.99$. As we can observe in the images, objects are successfully tracked along all the frames, there is not any error when they disappear behind the obstacles and their trajectory is predicted correctly.

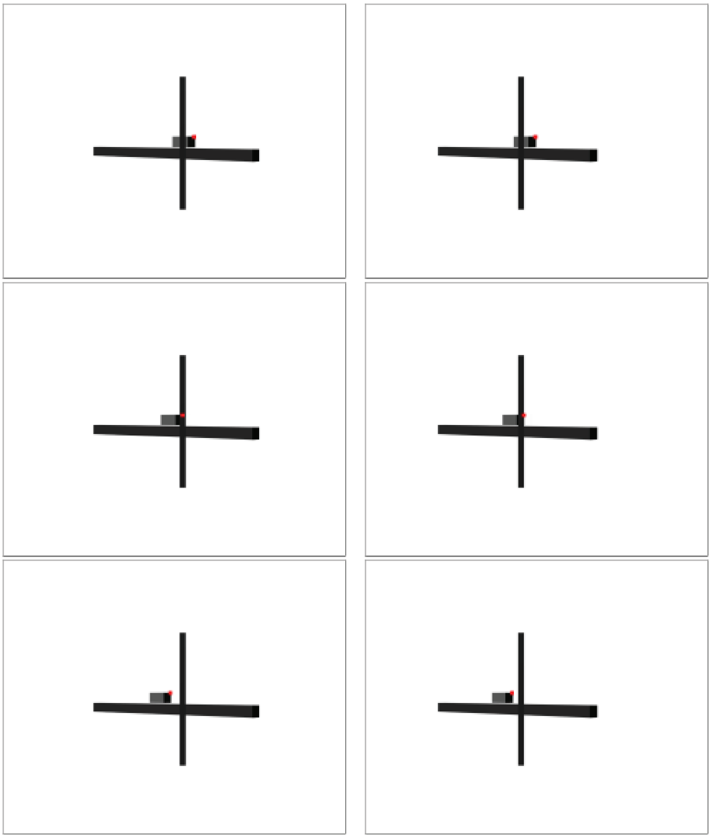


Fig. 20. RLS tracking stereo in a synthetic sequence

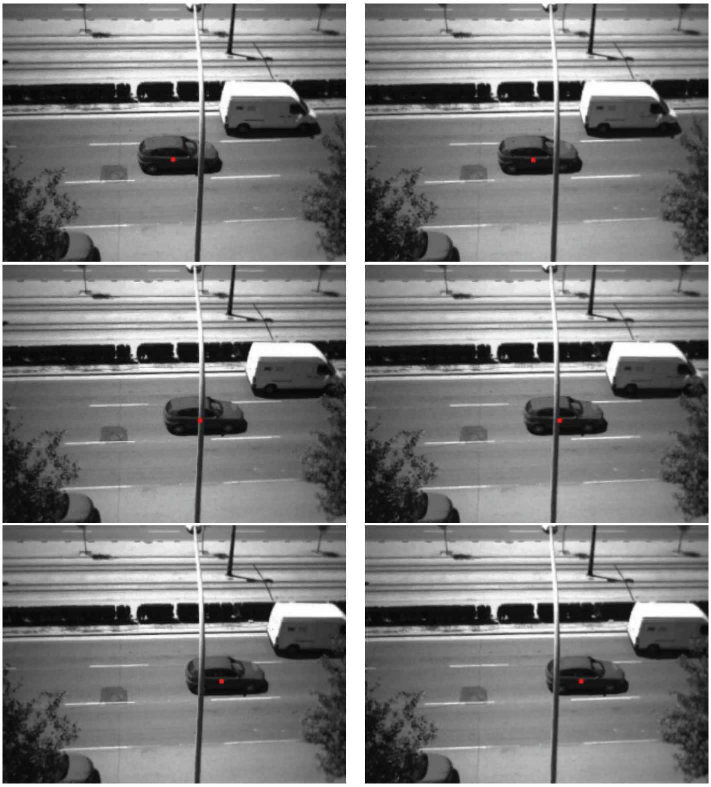


Fig. 21. RLS tracking stereo in a real sequence

5. References

Barron, J.L., Fleet, D.J. & Beauchemin, S.S. (1994) Performance of Optical Flow Techniques, *International Journal of Computer Vision*, Vol. 12(1): 43-77.

Bourel, F., Chibelushi, C.C. & Low, A.A. (2001) Recognition of facial expressions in the presence of occlusion, *Proceedings of the Twelfth British Machine Vision Conference*, Vol. 1: 213-222.

Cochran, S.D. (1990) Surface description from binocular stereo, *Dissertation presented in partial fulfillment of the requirements for the degree Doctor of Philosophy*, Vol. 1.

Cochran, S.D. & Medioni, G. (1992) 3-D Surface Description from Binocular Stereo, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14(10): 981-994.

Erdem, Ç.E., Tekalp, A.M. & Sankur, B. (2003) Video Object Tracking with Feedback of Performance Measures, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 13(4): 310-324.

Haykin, S. (1986) *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs NJ.

Horn B.K.P. & Schunck B. G. (1981) Determining optical flow, *Artificial Intelligence*, Vol. 17(1-3): 185-203.

Iñigo, R.M. (1989) Application of machine Vision to Traffic Monitoring and Control, *IEEE Transactions on Vehicular Technology*, Vol. 38(3): 112-122.

Ji, X., Wei, Z. & Feng, Y. (2006) Effective Vehicle Detection Technique for Traffic Surveillance Systems, *J. Vis. Commun. Image R.*, Vol. 17: 647-658.

Ljung, L. (1999) *System Identification. Theory for the User*, Prentice-Hall, Upper Saddle River NJ.

Lukas, B.D. & Kanade, T. (1981) An iterative image registration technique with an application to stereovision, *Proceedings of Imaging Understanding Workshop*, 121-130.

Ogata, K. (1987) *Discrete-Time Control Systems*, Prentice Hall. Upper Saddle River, NJ.

Parrilla, E., Riera, J., Giménez, M., Torregrosa, J.R. & Hueso, J.L. (2005) Vehicle tracking in urban routes by means of Lucas&Kanade algorithm, *Proceedings of the International Conference on Computational and Mathematical Methods in Science and Engineering*, 438-445.

Parrilla, E., Riera, J., Giménez, M., Torregrosa, J.R. & Hueso, J.L. (2005) Cálculo de velocidades mediante un sistema estereoscópico y algoritmos de flujo óptico, *Proceedings of the Congreso de Ecuaciones Diferenciales y Aplicaciones*.

Parrilla, E., Ginestar, D., Hueso, J.L., Riera, J. & Torregrosa, J.R. (2008) Handling occlusion in optical flow algorithms for object tracking, *Computers and Mathematics with Applications*, Vol. 56(3): 733-742.

Parrilla, E., Riera, J., Torregrosa, J.R. & Hueso, J.L. (2009) Handling occlusion in object tracking in stereoscopic video sequences, *Mathematical and Computer Modelling*, Vol. 50(5-6): 823-830.

Passino, K.M. & Yurkovich, S. (2000) Fuzzy control, *International Journal of General Systems*, Vol. 29(2): 341-342.

Rumelhart, D. & McClelland, J. (1986) *Parallel Distributed Processing*. MIT Press, Cambridge, MA.

Shin, J., Kim, S., Kang, S., Lee, S., Paik, J., Abidi, B., & Abidi, M. (2005) Optical flow-based real-time object tracking using non-prior training active feature model, *Real-Time Imaging*, Vol. 11: 204-218.

Trucco, E. & Plakas, K. (2006) Video Tracking: A Concise Survey, *IEEE Journal of Oceanic Engineering*, Vol. 31(2): 520-529.

Wren, C.R., Azerbayejani, A., Darrell, T. & Pentland, A.P. (1997) Pfinder: Real-time tracking of the human body, *IEEE Trans. Pattern Anal. Machine Intell*, Vol. 19: 780-785.

Young, P.C. (1999) Nonstationary time series analysis and forecasting, *Progress in Environmental Science*, Vol. 1(1): 3-48.

Zadeh, L.A. (1965) Fuzzy sets, *Inf. Control*, Vol. 8: 338-353.

8

Object Tracking and Indexing in H.264/AVC Bitstream Domains

Muhammad Syah Houari Sabirin and Munchurl Kim
Korea Advanced Institute of Science and Technology
Korea

1. Introduction

The use of surveillance video recording is mainly based on the activity of monitoring the areas in which the surveillance cameras are located. Consequently, one of the motivations of observing the footage from surveillance video is to locate and identify regions of interest (e.g. suspected moving objects) in the covered areas. The visual information of objects such as color, shape, texture and motion etc. enables users to easily navigate and browse regions of interest in the surveillance video data. However, this requires the semi-automatic or automatic analysis and description of visual features for surveillance video data, which is usually very challengeable and time consuming.

To describe the visual information of the surveillance video contents in a structured way for fast and accurate retrieval over visual queries, MPEG has standardized a set of visual descriptors in forms of metadata schema. The MPEG-7 Visual descriptors, as Part 3 of MPEG-7 (Multimedia Content Description Interface, standardized under ISO/IEC 15938), provide the standardized descriptions of visual features that enable users to identify, categorize or filter the image or video data (MPEG57/N4358). The MPEG-7 Visual descriptors enable the structured descriptions of color, texture, shape, motion, location and face recognition features. Moreover, since they are described in low-level, the retrieval process can be relatively easier by comparing a given query with the pre-generated descriptions of MPEG-7 Visual descriptors in database.

In surveillance video contents, MPEG-7 Visual descriptor enables the description of the characteristics of objects. The motion trajectory can be used to describe the behavior of objects. It can track and identify when and where the objects move. The motion information for an object enables the retrieval of the object by querying the intensity of the motion or the moving direction.

The automatic extraction of visual features from video contents has also become an emerging issue with MPEG-7 Visual descriptors. Recently, H.264/AVC has become a popular video compression tool for video surveillance due to its high coding efficiency and the availability of its real-time encoding devices. By incorporating content analysis into the surveillance applications, intelligent surveillance can be possible for efficient searching and retrieval of surveillance video data. In this case, the content analysis for the surveillance video data is usually required to be performed in real-time for metadata generation. For real-time generation of metadata, the content analysis is usually performed in the bitstream domain of compressed video instead of its raw pixel domain. In this chapter, we explain a feature extraction of motion trajectory which detects and tracks moving objects (ROI's) and also propose improvement to the feature extraction of motion activity of the moving objects in H.264/AVC bistreams. Our target is fast indexing of object retrieval in the metadata. Object retrieval can be performed only in interested time frame in which the metadata describes the existence of an object with high motion activity. Thus the motion trajectory of the object can be detected in the given time frame, without having to search the entire metadata, localizing objects of interest by effectively limiting the scope of searching subjects, thus reducing computation complexity in similarity matching process

Much research has been made object detection and tracking in MPEG-1 or MPEG-2 compressed video, and recently some related works (Zeng, 2005; Thilak 2004; You 2007; You 2009) have been tried for moving object detection and tracking in H.264/AVC bitstream domains. Nevertheless the methods do not consider identifying the objects into different identity, thus the objects cannot be consistently tracked along the sequence. To overcome this problem, a novel technique is introduced which uses motion information and quantized residual coefficients in every 4x4 block partition units of a macroblock. Using smallest block partition, our method can detect small sized object. Furthermore, by observing both motion and residue coefficients, we develop a similarity model to keep track of object when occlusion occurred.

We use motion activity of detected object as a point of interest in indexing the tracking using MPEG-7 descriptor. The MPEG-7 motion activity descriptor can describe the degree of motion activity caused by moving objects in the video. It has a motion intensity element to describe how much active the moving objects are. The normative of MPEG-7 motion activity descriptor only expresses the motion intensity into 5 levels of motion activity. However, how to categorize an amount of motion intensity into one of the five motion activity levels is non-normative and is left for application vendors.

In this chapter, the proposed motion activity categorization method takes into account the moving and non-moving parts in each frame. The non-moving parts are usually stationary background, which are more likely to be encoded in the partition of 16x16 macroblock size. On the other hand, the moving objects are likely to be encoded in sub-macroblock partitions due to their motion. Therefore weighting factors are assigned on encoded macroblock types by giving more emphasis on sub-macroblock partitions. Details of motion activity description are described in Section 2.2.

This chapter is organized as follows: We first describe the proposed method of extracting visual features from H.264/AVC bitstream domain in Section 2; the experimental results are described in Section 3; the example of conceptual structure of MPEG-7 metadata with the descriptions of extracted features is presented in Section 4; finally, the conclusion and future works are described in Section 5.

2. Feature extraction for MPEG-7 visual descriptor

2.1 Feature extraction of MPEG-7 motion trajectory descriptor

The MPEG-7 motion trajectory descriptor describes the transition of the relative coordinates of moving region in a frame during a sequence. The object detection and tracking method will provide the coordinates of the detected object for each frame. Feature extraction of motion trajectory is performed by observing each of 4x4 block partition units in a macrbolock in the bitstream. We take into account only the blocks having nonzero motion vectors or nonzero quantized residue coefficient by assumption that the camera is static so the observed blocks have high probability of being the region of moving objects.

Define B_{ij} as a 4x4 block partition in frame f having nonzero motion vectors and nonzero quantized residue coefficient values, where $\{i \in I, j \in J\}$ is the location of the block in the 4×4 block-unit of a $4I \times 4J$-sized frame. Thus by definition, all blocks having zero motion vectors are spatially segmented into background and $K \prec 4I \times 4J$ denotes the number of 4x4 block partition that belongs to foreground that contains a nonzero motion vector and nonzero quantized residual coefficient values. We then define a block group C_f as the group of 4×4 block partitions that are adjacent to each other (we will use the term "group" to refer to as the block group hereafter). Each group is representation of individual objects to be tracked.

Let $\hat{O}_f^{\ell} = \left\{C_f^u ; u \in K\right\}$ be a set of groups where ℓ denotes the index of group and $u \in U$ denotes the block partition index. A group may have U member block partitions, where $U \in K$ but not necessarily in order of K (that is, one group may contain any arbitrary block partition regardless its index order). For implementation, we keep the maximum number of groups to be recognized to ten objects. For each group, the parameters are classified into two categories: static parameters and dynamic parameters. Static parameters are the parameters that have been defined by the bitstreams and its values will not be changed by the algorithm throughout the sequence. Dynamic parameters are the parameters that will be updated adaptively to the change of the static parameters.

The static parameters are position, direction, and energy. Position of the group is denoted by x and y, where $x \le width$ and $y \le height$. The position value is defined from the leftmost x and topmost y positions of the member block partitions of the group. For the *l*-th group in frame f, its position is defined as x_f^{ℓ}, y_f^{ℓ}.

The direction of the group, $D \in \Re, 0 \le D \le 360$, is calculated from the motion vector average of member blocks of the group. For the *l*-th in frame f, its direction is defined as

$$D_f^{\ell} = \frac{1}{U} \sum_{u \in U} \tan^{-1}\left(\frac{(mv_y)_f^u}{(mv_x)_f^u}\right) \tag{1}$$

where $(mv_x)_f^u$ and $(mv_y)_f^u$ are the motion vectors of C_f^u for each block partition u. For the *l*-th group in frame f its energy is defined as

$$e_f^{\ell} = \frac{1}{U} \sum_{u \in U} \left(\frac{1}{16} \sum_{j \in 16} (R_f^{uj})^2\right) \tag{2}$$

where R_f^{uj} is the residue data of C_f^u for each j-th pixel in block partition u.

The dynamic parameters are event status and label. The status of events occurred in the group is given by $S = \{0,1\}$. The status events defined for a group are: (1) *occlusion*, (2) *out of frame*, (3) *stop*, and (4) *in to frame*. Additionally we add default status (0) if none of the mentioned status occurred to the group. For the *l*-th group in frame f its status is defined as $S_f^{\ell}(0)$, $S_f^{\ell}(1)$, $S_f^{\ell}(2)$, $S_f^{\ell}(3)$ and $S_f^{\ell}(4)$ according to the event respectively. Restriction applies to the event status where one object can only have at most one status in a frame.

The label of the group is denoted as l with $l > 0$. For the *l*-th group in frame f its label is defined as l_f^{ℓ}. Note that label l is not the same index as ℓ, although it is possible that a group may have the same value for the label and index, for example, in initial frame where we set the label equal to the index, or if the relative positions of objects are interchanged over the sequence (for example, if the objects are moving toward the same direction). Label

l defines the actual identification of groups as object candidates regardless their positions in frames, while index ℓ defines the order of groups for which the top-leftmost group has the smaller index value. Similar to groups, we limit the maximum number of objects to ten objects.

To this point, we have object candidates as groups with static and dynamic parameters which are then to be correctly identified frame to frame, i.e. tracking the objects by correctly recognize whether the candidate of objects in the current frame are the same as the identified object in the previous frame. In the next section we will describe the method of tracking the groups and set their label appropriately using a similarity model.

Object tracking process begins with pre-processing the groups in order for inconsistent groups to prevent from incorrect tracking. In addition, temporal and spatial filtering is applied to remove noise that resulted from inconsistent, isolated and dangled block. In the temporal refinement, we filter (remove) the groups that inconsistently appear in frames and interpolate (insert) the groups that inconsistently disappear over frames.

Let h be the number of frames where a group $\hat{O}^l$ exists over five consecutive frames including the current frame, two frames prior and two frames, thus $0 \le h \le 5$. The filtering is simply to remove the groups that appears only in 2 frames or less as given by

$$\hat{O}_f^\ell \begin{cases} keep & if\, h > 2 \\ remove & else \end{cases} \tag{3}$$

Similarly, $\hat{O}_f^\ell$ is interpolated when there is missing groups in frame f by taking the average value of static parameters of groups in frames $f' = \{f-2, f-1, f, f+1, f+2\}$ so the interpolated $\hat{O}_f^\ell$ will have average values of positions, directions and energies. The interpolation is defined by

$$\hat{O}_f^\ell = \varnothing \Leftrightarrow \hat{O}_f^\ell = \frac{1}{h}\sum_{f'} \hat{O}_{f'}^l, \hat{O}_{f'}^l \ne \varnothing \tag{4}$$

From the filtering and interpolating process, we expect to have consistent groups (object candidates) ready to be tracked. In this process we track the object candidates by updating their dynamic parameters. At the end of the process, the object candidates will be determined as the real objects O_f^ℓ .

The first step of tracking is to observe the difference (increment or decrement) of number of groups from one frame to the next. Let L_f be the total number of groups in the current frame and L_{f-1} be the total number of groups in the previous frame. Then we define the difference as γ and set its value as follows

$$\gamma = \begin{cases} 0 & \Leftrightarrow L_{f-1} = L_f \\ 1 & \Leftrightarrow L_{f-1} < L_f \\ 2 & \Leftrightarrow L_{f-1} > L_f \end{cases} \tag{5}$$

Assessing the changes in the number of groups in the first step is performed when a new frame is tracked. Following the assessment we may then update the status event of every candidate of objects in a frame. The status event will determine the way of finding the similarity of an object candidate in the current frame with its corresponding one in the previous frame.

The status event is updated when the number of groups in the current frame is larger than the number of groups in the previous frame. For the other γ we set status events of all groups as 0, that is, $S_f^{\ell}(0)=1$. Thus for γ =2 we update the status event of a group based on the position of the group relative to the other groups or to the the edge of the frame, or based on its existence in the current and previous frames.

The relative position of a group to other groups is determined based on the Euclidean distance of its parameters. We define

$$d(O_{f-1}^{\ell},O_{f-1}^{m})_{xy}=\left(\left(x_{f-1}^{\ell}-x_{f-1}^{m}\right)^{2}+\left(y_{f-1}^{\ell}-y_{f-1}^{m}\right)^{2}\right)^{1/2} \tag{6}$$

as the distance between object ℓ and object m in frame $f-1$,

$$d(O_{f-1}^{\ell},O_{f-1}^{m})_{x}=\left(\left(x_{f-1}^{\ell}-x_{f-1}^{m}\right)^{2}\right)^{1/2} \tag{7}$$

as the distance between position x of object ℓ and object m in frame $f-1$, and

$$d(O_{f-1}^{\ell},O_{f-1}^{m})_{y}=\left(\left(y_{f-1}^{\ell}-y_{f-1}^{m}\right)^{2}\right)^{1/2} \tag{8}$$

as the distance between position y of object ℓ and object m in frame $f-1$; $m\in L$ denotes the pair index of object ℓ in frame $f-1$ where $\ell\neq m$.

The relative position of a group to the frame edge is determined by its outer boundaries. We define $p_{\min}(\hat{O}_f^{\ell})_{xy}$ as the leftmost x and topmost y positions of member block partitions of group ℓ in frame f, $p_{\max}(\hat{O}_f^{\ell})_x$ as the leftmost x position of member block partitions of group ℓ in frame f, and $p_{\max}(\hat{O}_f^{\ell})_y$ as the topmost y position of member block partitions of group ℓ in frame f.

Finally, the criterion is given by

$$\begin{aligned}
S_f^{\ell}(1)&=\begin{cases}1 & \text{if } d(O_{f-1}^{\ell},O_{f-1}^{m})_{xy}<\left(d(O_{f-1}^{\ell},O_{f-1}^{m})_{x}\wedge d(O_{f-1}^{\ell},O_{f-1}^{m})_{y}\right)\\ 0 & \text{otherwise}\end{cases}\\
S_f^{\ell}(2)&=\begin{cases}1 & \text{if } \left(p_{\min}(\hat{O}_f^{\ell})_{xy}\le 8\right)\wedge\left(p_{\max}(\hat{O}_f^{\ell})_{x}\ge width-8\wedge p_{\max}(\hat{O}_f^{\ell})_{y}\ge height-8\right)\vee O_{f-1}^{\ell}\neq\varnothing\\ 0 & \text{otherwise}\end{cases}\\
S_f^{\ell}(3)&=\begin{cases}1 & \text{if } S_{f-1}^{\ell}(1)=0\vee S_{f-1}^{\ell}(2)=0\vee \hat{O}_f^{\ell}=\varnothing\\ 0 & \text{otherwise}\end{cases}\\
S_f^{\ell}(4)&=\begin{cases}1 & \text{if } \left(p_{\min}(\hat{O}_f^{\ell})_{xy}\le 8\right)\wedge\left(p_{\max}(\hat{O}_f^{\ell})_{x}\ge width-8\wedge p_{\max}(\hat{O}_f^{\ell})_{y}\ge height-8\right)\vee O_{f-1}^{\ell}=\varnothing\\ 0 & \text{otherwise}\end{cases}
\end{aligned} \tag{9}$$

where O_{f-1}^{ℓ} denotes object ℓ in the previous frame. Based on γ and the status event, the next step of tracking is to update the label l_f^{ℓ} by finding the most similar object between two frames and to set the same label for the most similar objects. The similarity is computed by finding index ℓ that gives the minimum distance between the group in the current frame $\hat{O}_f^\ell$ and the object in the previous frame O_{f-1}^ℓ for given previous frame's event S and the changes of group γ . We define the similarity model as

$$l_f^\ell = \left\{ l_{f-1}^\ell \mid \min_\ell \left(\Delta\left(O_{f-1}^\ell, \hat{O}_f^\ell, S, \gamma \right) \right) \right\} \tag{10}$$

where $\Delta\left(O_{f-1}^\ell, \hat{O}_f^\ell, S, \gamma\right)$ denotes the similarity function between O_{f-1}^ℓ and $\hat{O}_f^\ell$ for given S and γ . When γ =0, the similarity is defined as

$$\Delta\left(O_{f-1}^\ell, \hat{O}_f^\ell, S, \gamma\right) = d(O_{f-1}^\ell, \hat{O}_f^\ell)_{xy} \Gamma(\hat{O}_f^\ell) \tag{11}$$

where $\Gamma(\hat{O}_f^\ell) = \{0,1\}$ denotes the existence of group l in the current frame. If γ =1, the similarity is defined as

$$\begin{aligned} \Delta\left(O_{f-1}^\ell, \hat{O}_f^\ell, S, \gamma\right) &= d(O_{f-1}^\ell, \hat{O}_f^\ell)_{xy} S_f^\ell(0) \\ &+ d(O_{f-t}^\ell, \hat{O}_f^\ell)_{De} S_f^\ell(1) + l_f^{\ell_{new}} S_f^\ell(4) \end{aligned} \tag{12}$$

where

$$d(O_{f-t}^\ell, \hat{O}_f^\ell)_{De} = \left(\left(\tilde{D}_{f-t}^\ell - \tilde{D}_f^\ell \right)^2 + \left(\tilde{e}_{f-t}^\ell - \tilde{e}_f^\ell \right)^2 \right)^{1/2} \tag{13}$$

which denotes the similarity between O_{f-t}^ℓ and $\hat{O}_f^\ell$ based on their direction and energy. t denotes number of elapsed frames since $S_f^\ell(1)$ is set (i.e. when occlusion has occurred), and $l_f^{\ell_{new}}$ is a new label.

In finding the distance in direction and energy between two objects, instead of using the actual values, we normalize the parameter values in logarithmic scale, where $\tilde{D} = \log_{10}(D)$ and $\tilde{e} = \log_{10}(e)$, to make fair comparison of two values in different ranges (as can be seen from (1) and (2), the direction D has possible values from 0 to 360 while energy e has possible values from 0 to virtually unlimited). New label $l_f^{\ell_{new}}$ is determined by observing the list of labels already used or currently being used in the frame so the used labels or the current labels are not defined as new labels. When γ =2,

$$\Delta\left(O_{f-1}^\ell, \hat{O}_f^\ell, S, \gamma\right) = d(O_{f-1}^\ell, \hat{O}_f^\ell)_{xy} \Gamma(O_{f-1}^\ell) \tag{14}$$

where $\Gamma(O_{f-1}^\ell) = \{0,1\}$ denotes the existence of the object in the previous frame.

By this step, the groups are now labeled with the same label of the most similar object in the previous frame and the groups are then defined as the real objects for which their parameter values will then be used in the next frame to identify the candidate of objects.

2.2 Feature extraction of MPEG-7 motion activity descriptor

The MPEG-7 motion activity descriptor describes the degree of motion activity in a frame in terms of the intensity, direction, spatial distribution, spatial localization, and temporal distribution features of motion activity. Here, we especially take into consideration the intensity feature of the descriptor because surveillance observation tends to give more attention to the intensity of motion. Furthermore, the usage of intensity descriptor is mandatory in annotating MPEG-7 motion activity.

The intensity feature indicates the degree of motion into five levels (*very low*, *low*, *medium*, *high*, and *very high*). However, the feature extraction itself is outside the standard. So, in this chapter, we propose a feature extraction scheme for the computation of such feature values in the H.264/AVC bitstream domain.

Feature extraction of motion intensity in a frame can be performed by computing the standard deviation of the magnitudes of motion vectors of the macroblocks. The standard deviation can then be quantized into one of the five levels in which the quantization parameters are determined experimentally by the training data of motion vectors from H.264/AVC bitstreams. In this chapter, we use the quantization parameters in Table 1 as defined in (Manjunath, 2002).

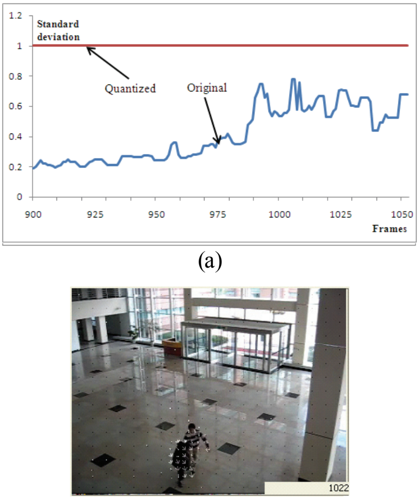
Activity value	Range of σ
1	$0 \leq \sigma < 3.9$
2	$3.9 \leq \sigma < 10.7$
3	$10.7 \leq \sigma < 17.1$
4	$17.1 \leq \sigma < 32$
5	$32 \leq \sigma$

Table 1. Quantization threshold of motion intensity

The result of using the method described in (MPEG57/N4358) for surveillance video recording has drawback to describe the information of object activity. This justification is depicted in the following case. Fig. 1(a) shows the curve of standard deviation of the magnitudes of motion vectors for a surveillance video sequence with its corresponding quantized motion intensity, while Fig. 1(b) shows one frame from the sequence with its representation of motion vectors. While the frame shows the objects actively moving, the resulting level of motion activity corresponds to very low activity.

This problem is caused by the size of the objects that are relatively small, compared to the resolution of the frame. Even though the method described in (MPEG57/N4358) takes into account the resolution of frame for the computation of standard deviation, it is still hard to correctly categorize such motion. Therefore, small objects with high activity cannot be identified as having high activity motion. On the other hand, it is also possible that large objects (e.g., the object that located very near to the camera) with slow activity will be identified as having high activity motion.

To overcome this problem, we propose a method of preprocessing the standard deviation values before it is quantized. Our method utilizes the macroblock partition types (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 block types) and applies different values of the weighting factor for different block classes. Our method is applicable for surveillance video contents that are commonly captured by still cameras. These surveillance video usually has still background (i.e., no abrupt changes in intensity) and all motions are likely to happen by object movement. The weighting factor for each MB partition is defined based on the size of the partition itself. For example, an MB with 4x4 block types can have 16 motion vectors, so it can contribute more to the standard deviation of motion intensity, compared to the MB's with a smaller number of block partitions. For such an MB, we multiply a larger value of weighting factor to its magnitude of motion vectors so the resulting standard deviation increases large enough to represent the motion activity of the object.



(a)

(b)

Fig. 1. (a) The curve of standard deviation of the magnitudes of motion vectors for a surveillance video sequence and its corresponding quantized value identifies the sequence has very slow activity of objects (intensity = 1) (b) a snapshot frame from the same sequence with its corresponding motion vectors of macroblocks

However, a slowly moving object close to a surveillance camera can be identified as having high motion intensity due to the large portion of the object size, compared to the background. Therefore, we need to reflect the partition of the object into the weighting factor by estimating the object size with respect to its distance to the camera. That is, the larger the estimated object size is, the larger a penalty factor value is taken into account in the weighting factor. So, the weighting factor is defined as

$$w_i = M - r\sum_{k=1}^{n_{PB}} e^{k \cdot r} \tag{15}$$

Here M reflects the size of the i-th partition block or macroblock, assuming that the square-sized blocks (4x4 and 8x8) contribute more to the motion activity due to more numbers of those block types in each MB, while the remaining block types (4x8, 8x4, 16x8, 8x16) are contribute less. Therefore M can be calculated as follows:

$$M = \begin{cases} m \times n, \\ if\ mb\_type = class\ 1\ (8 \times 8\ or\ 4 \times 4) \\ \\ \sqrt{m \times n}, \\ if\ mb\_type = class\ 2\ (16 \times 8, 8 \times 16, 8 \times 4\ or\ 4 \times 8) \end{cases} \tag{16}$$

where m and n indicate the width and height of one partitioned block in the *i*-th partition block or MB. r is a multiplication factor of the penalty term in (15) based on the partition ratio of partition block, with similar assumptions to M where square-sized blocks have smaller penalty than the other block type. Thus r is given by

$$r = \begin{cases} \frac{1}{4}, & if\ mb\_type = 8\times 8\ or\ 4\times 4 \\ \frac{1}{2}, & if\ mb\_type = 16\times 8, 8\times 16, 8\times 4\ or\ 4\times 8 \end{cases} \tag{17}$$

n_{PB} is the accumulated number of all the partition blocks for a same class of the block partition types to the *i*-th partition block or macroblock. In (15), the penalty term contains exponents in summation. This is to put more penalties on the partition blocks of a same class coming in the subsequent MB's. In this regard, smaller values of the weighting factor are then applied for the partition blocks of having the same class with more frequent occurrence. By doing so, the slowly moving object of a large size can be regarded as having low motion activity.

In (16), the square block types are more emphasized for the calculation of standard deviation than the non-square block types because the square block type allows for more sub-block partitions which results in more motion vectors. Therefore, with the weighting factor in (15), the standard deviation of the intensity in motion activity can be calculated as

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{18}$$

where x_i is a weighted magnitude of motion vector and is given by

$$x_i = w_i \times |mv_i| \tag{19}$$

mv_i is the motion vector of the *i*-th partition block or macroblock and N is the total number of the partition blocks and macroblocks (no partitions) in a frame.

Sometimes, due to the noise of motion caused by camera shaking, abrupt light intensity change or unexpected shadow, the standard deviation undergoes abrupt change in a certain frame over time. Therefore, we remove such outliers by applying a median filter on standard deviation values for every five frames. Finally the median filtered standard deviation values are quantized into the five levels of motion activity.

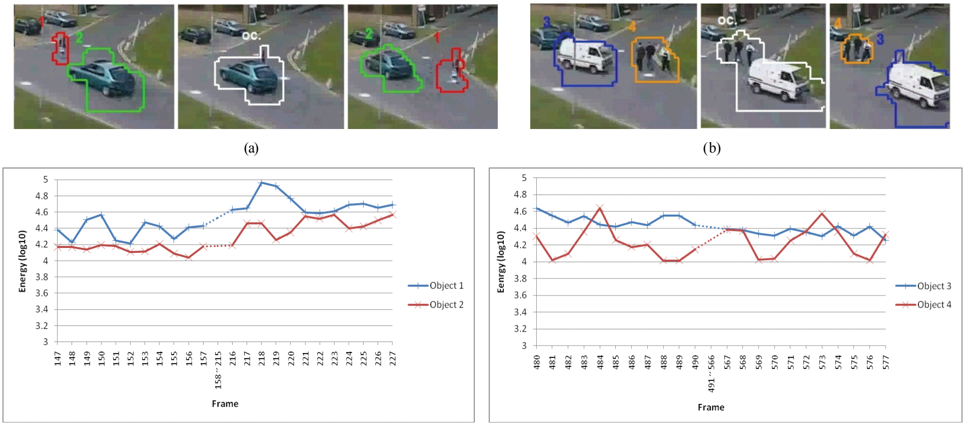
3. Experimental results

3.1 Feature extraction of motion trajectory

(a) Three sequences were used in our experiments: *Corridor* and *Speedway* for low and high camera, respectively; and *PETS2001* for occlusion handling experiments. *Corridor* sequence is a QVGA (320x240), 670 frames, indoor surveillance video with single object (a person) moves toward the camera. *Speedway* sequence is a CIF (352x288), 198 frames, outdoor surveillance video of traffic in a speedway with several vehicles. *PETS2001* is a QPAL (384x288), 600 frames, outdoor surveillance video in a building complex, with several objects and group of objects and having occlusions between some of them.

We first present the observation of using energy of object to determine the similarity between objects before and after occlusion as defined in (12). Fig. 2 shows the example of occlusion and disocclusion occurred in *PETS2001* sequence with the graphs showing the energy value for occluded objects before and after occlusion.

In Fig. 2(a), object 1 and object 2 starts to occlude in frame 157 and disocclude in frame 216. As depicted by the graph in Fig. 2(c) we can see that the energy value of each object is distributed fairly different so we can easily match the energy of object 1 and object 2 before and after occlusion. However, there is also case when the energy value alone is not enough to determine the similarity of object. As shown in Fig. 2(d), the energy value of object 4 fluctuates and in some points overwhelming the energy value of object 3. Furthermore, it makes the energy value almost has no significant differences for the similarity calculation to find similarity (and dissimilarity) of both objects after occlusion in frame 567, although energy value of both objects fairly different just before occlusion in frame 490. For this reason we add direction value term in the similarity model as the weighting factor to control the energy value for the model to not depend only one feature of object. Fig. 2(b) shows the result of handling occlusion for object 3 and object 4.



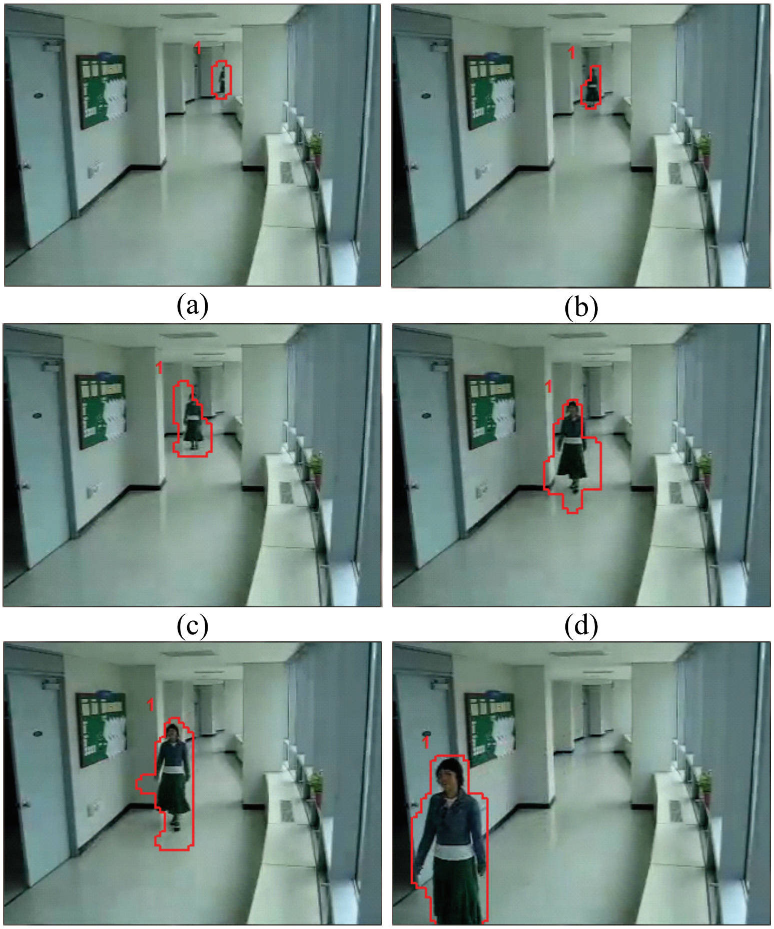
(a) (b)

(c) (d)

Fig. 2. Computing energy similarity of two occluded objects in PETS2001 sequence (a) between object 1 and object 2, (b) between object 3 and object 4. Graphs show the value of energy of (c) objects 1 and 2, (d) object 3 and 4. Dashed lines in the graphs imply the occlusion.

The results from of overall object detection and tracking process are shown. Fig. 3 shows the result of overall object detection and tracking process for *Corridor* sequence. The sequence is taken from video surveillance camera located near the ceiling of a corridor and features one person moving toward the camera. In this sequence the object's size changes gradually from small to large until the person finally walks approximately below the camera. As shown in Fig. 3, when the object starts to appear in the frame, its motion and residue information is available and the algorithm can recognize it correctly until the object is large and located near the camera.

Fig. 4 shows the result of overall object detection and tracking process for *Speedway* sequence. The sequence is taken from video surveillance camera located in high position and features four vehicles: two vehicles move away from camera and two vehicles move toward the camera. In this sequence the objects' size changes rapidly from large to small for



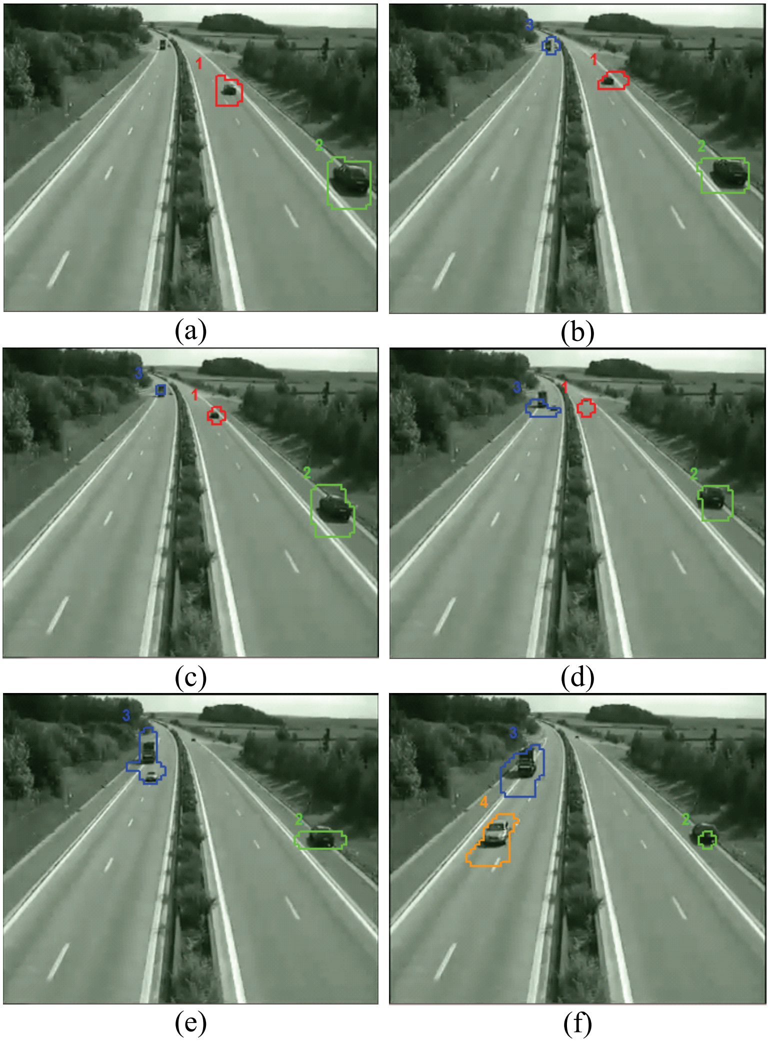
(a) (b)

(c) (d)

(e) (f)

Fig. 3. Object detection and tracking results for 18th (a), 298th (b), 447th (c), 498th (d), 535th (e) and 650th (f) frame of Corridor sequence.

the objects that move away from camera, and vice versa. In the bitstreams, the small objects are too small to have motion and residue information. As a result, those objects cannot be recognized by the segmentation process, as can be seen in Fig. 4(a). In frame 54, shown in Fig. 4(b), the small objects moving toward the camera are first recognized into single object because their motion and residue information are represented in small amount of adjacent block partitions. As the frame advances, the two objects become large and finally can be recognized as two objects in the frame 186 as shown in Fig. 4(f). Similarly, object one (red) is getting small as the frame advances and finally unable to be recognized by the algorithm (Fig. 4(e)). From this experiment it can be seen that the algorithm is able to detect small object with block partitions take up less than one macroblock as long as its appearance is consistent along several frames in the sequence.



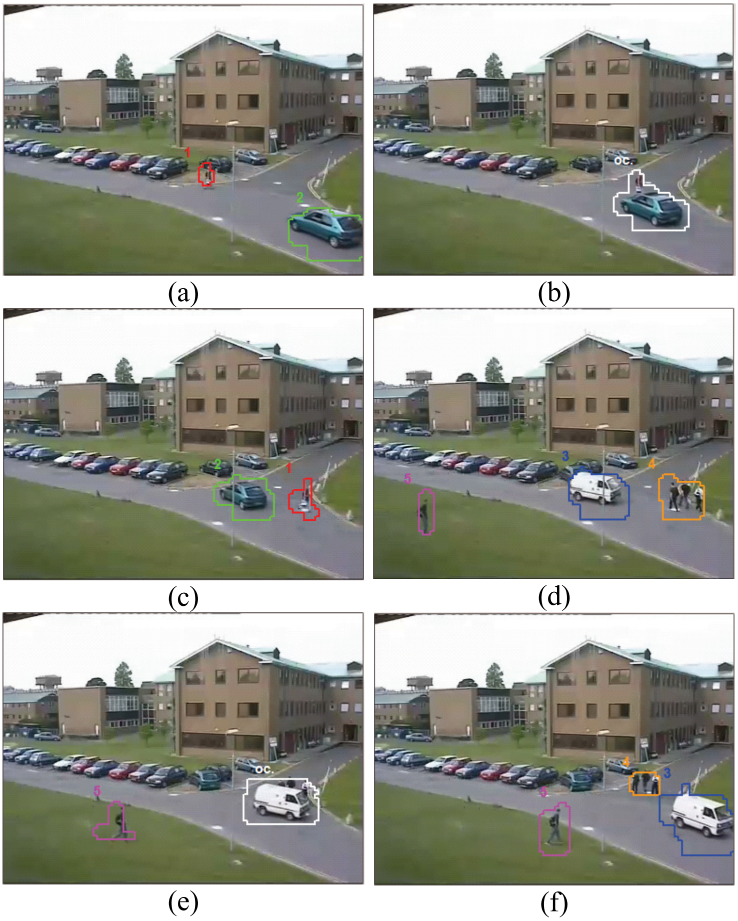
(a) (b)

(c) (d)

(e) (f)

Fig. 4. Object detection and tracking results 43rd (a), 54th (b), 70th (c), 92nd (d), 138th (e) and 186th (f) frame of Speedway sequence.

Fig. 5 shows the result of overall object detection and tracking process for *PETS2001* sequence. The sequence features five objects: two persons, two vehicles, and one group of persons where there are two times occlusions between the objects. The first occlusion is occurred between a person and a vehicle (Fig. 15(b)). During the occlusion, both objects are identified as single object and labeled as “occluded objects”. Because we kept their information prior the occlusion, the label information correctly identified both object to their original label after the occlusion. Similar result also occurred in the second occlusion when another vehicle occluded with group of persons, as shown in Fig. 5(e). The identification of all objects can be correctly detected prior to occlusion (Fig. 5(d)) and after occlusion (Fig. 5(f)).



(a) (b) (c) (d) (e) (f)

Fig. 5. Object detection and tracking results for 104th (a), 165th (b), 207th (c), 462nd (d), 510th (e) and 556th (f) frame of PETS2001 sequence.

3.2 Feature extraction of motion activity

In this section we present the result of the extraction of motion activity using the proposed method with a weighting factor and its comparison with the one without the weighting factor. The experiments were performed for surveillance video of QVGA size encoded at 30 fps by an H.264/AVC baseline profile encoder.

As shown in Fig. 1, the quantization of standard deviation of motion vector magnitudes cannot appropriately describe the real motion activity. In this experiment, the surveillance video camera covers a large area, i.e., the mounted camera is far away from the objects. As the result, quantization of the standard deviation without the weighting factor results in incorrect classification of motion intensity.

Fig. 6 shows the quantization of the standard deviation of the motion vector magnitudes using a weighting factor. As a result we can identify the motion of the object. The frames that have relatively high motion are quantized as higher intensity level.

We present another result where the position of the camera is relatively low from the floor, which means there is a possibility that the size of objects becomes large so that it might lead to incorrect motion activity quantization.

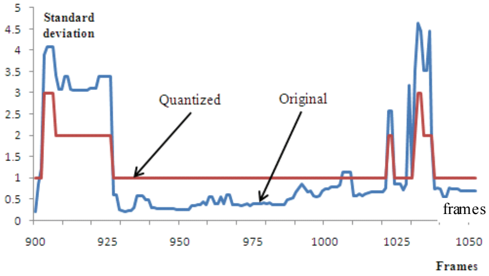
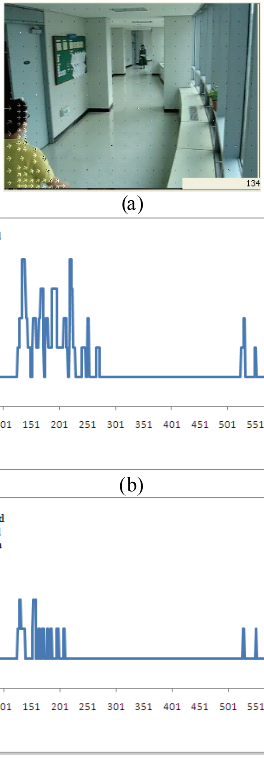


Fig. 6. Curves of original and quantized standard deviations

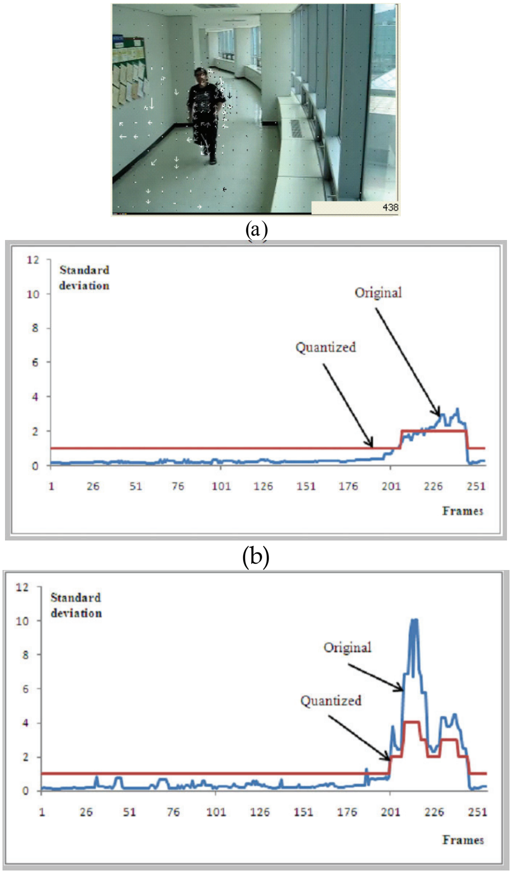
In order to see the effect of assigning penalty value in computing the standard deviation of the motion vector magnitudes, we present a screenshot with an object that moves slowly near the camera as shown in Fig 7(a).



(c)

Fig. 7. The effect of assigning penalty value: (a) A snapshot frame (b) The quantized standard deviation without the penalty value and (c) The quantized standard deviation with penalty value.

For this, the quantized standard deviation without penalty values results in higher intensity levels as shown in Fig 7(b), compared to those with penalty values in Fig. 7(c). The object motion activity is abnormally high as having level 4 and 5 in the interval between 134th frame and 230th frame, and the interval between 570th frame to 670th frame. Assigning penalty values for this sequence gives better quantization result for the same frame as shown in Fig 7(c).



(a)

(b)

(c)

Fig. 8. (a) A snapshot frame of surveillance video sequence shows high motion activity. (b) Quantized standard deviation without weighting factor and (c) Quantized standard deviation with weighting factor.

Fig 8(a) shows the screenshot of the sequence in which an object moves fast. Fig. 8(b) shows the quantization results of the standard deviation of the magnitude of motion vectors without applying the weighting factor. The quantized standard deviation seems to correctly represent the activity of the objects. However, the video sequence actually consists of the scene where an object moves fast. Fig. 8(c) shows the result of the quantization by applying the weighting factor and penalty values, which results in relatively more appealing quantization results of motion intensity.

4. Implementations

Here we present the conceptual structure of MPEG-7 Visual for describing the result of motion trajectory and motion activity description to illustrate the practical usage of our method.

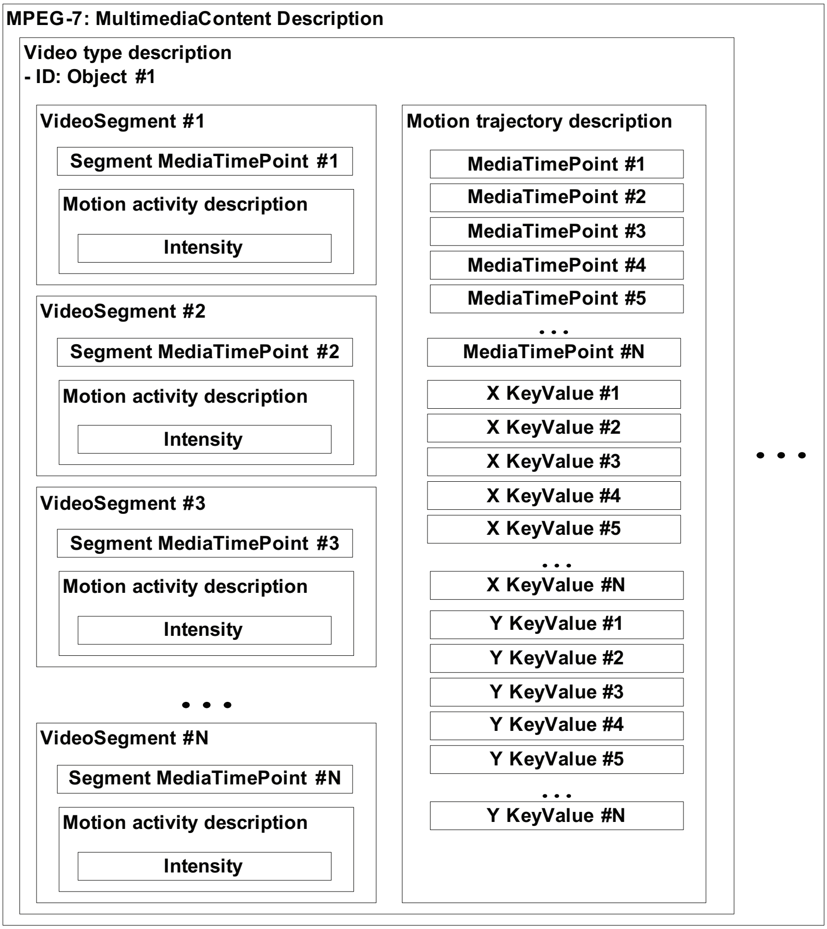


Fig. 9. The example of conceptual structure of MPEG-7 metadata to describe object motion trajectory and motion intensity.

Fig. 9 shows the example of conceptual structure of MPEG-7 metadata. A multimedia content descriptor from MPEG-7 Multimedia Description Scheme (MDS) (MPEG57/N4242) is the container for video type object description that can be used to describe each of detected objects in the bitstream. As shown in Fig. 9, the identity of object, which obtained from labeling process described in section 2.1, can be described by an identifier attributes, thus all descriptions inside the video descriptor belong to this object. A video segment descriptor in a video descriptor is used to index the segment of the video having the motion intensity description of this object. Each video segment has media time information, describing the frame number where the intensity is annotated. Effective indexing can be generated by describing video segment description for every time frame when the intensity is high. A motion trajectory description describes the coordinate x and y of tracked objects, which can be represented by the interpolation of the centroid of the region resulted by the tracking algorithm. As shown in Fig. 9, each of x and y coordinates of tracked object is referred by each media time point.

Referencing the motion trajectory time point and video segment time point can be performed to find the location of detected object from frame to frame. Suppose the motion intensity description of an object describes a high intensity motion at frame 10, then the corresponding segment media time point of video segment acts as pointer to the media time point of motion trajectory. Then the trajectory of this object can be found by tracing the x and y coordinate values referred by media time point of motion trajectory started from frame 10.

5. Conclusions

In this chapter, a novel approach for extracting the ROI's of multiple moving objects in H.264/AVC bitstream domains is proposed which includes spatial and temporal filtering, similarity model for object detection and tracking and motion activity extraction to enhance indexing.

In order to extract the motion activity feature values, we proposed to utilize the macroblock types of H.264/AVC bitstreams by assigning the weighting factor to the block partitions so that the standard deviation of the magnitudes of the motion vectors can be appropriately quantized to represent the motion activity levels.

Finally, some examples for the usage of the extracted features are shown for visual information retrieval in the form of MPEG-7 structure that can make it easy the detected object indexing and information retrieval. Future study on the current can be extended to deal with occlusion problems during object detection and tracking in H.264/AVC bitstream domains.

6. References

MPEG57/N4358, *ISO/IEC FDIS 15938-3 Visual*, ISO/IEC JTC 1/SC 29/WG 11, July 2001, Sydney

MPEG57/N4242, *ISO/IEC FDIS 15938-5 Multimedia Description Schemes*, ISO/IEC JTC 1/SC 29/WG 11, July 2001, Sydney

Manjunath, B. S.; Salembier, P. & Sikora, T., *Introduction to MPEG-7, Multimedia Content Description Interface*, John Wiley & Sons, 2002

Zeng, W.; Du, J.; Gao, W. & Huang, Q., Robust moving object segmentation on H.264|AVC compressed video using the block-based MRF model, *Real-Time Imaging*, vol. 11(4), 2005, pp. 290-299

Thilak, V. & Creusere, C. D., Tracking of extended size targets in H.264 compressed video using the probabilistic data association filter, *EUSIPCO 2004*, pp. 281-284

You, W.; Sabirin, M. S. H. & Kim, M., Moving Object Tracking in H.264/AVC bitstream, *MCAM 2007, LNCS*, vol. 4577, pp. 483-492

You, W.; Sabirin, M. S. H. & Kim, M. (2009) Real-time detection and tracking of multiple objects with partial decoding in H.264/AVC bitstream domain, In: *Proc. Of SPIE*, N. Kehtarnavaz & M. F. Carlsohn (Ed.), 72440D-72440D-12, SPIE, San Jose, CA, USA

9

Accelerating Live Graph-Cut-Based Object Tracking Using CUDA

Ismael Daribo, Zachary A. Garrett, Yuki Takaya and Hideo Saito
Keio University
Japan

1. Introduction

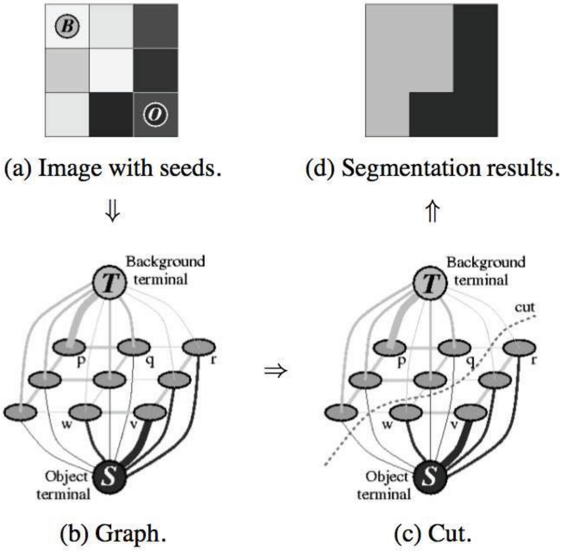
Graph cuts have found many applications that address the problem of energy minimization, which occur frequently in computer vision and image processing. One of the most common applications is binary image segmentation, or silhouette extraction.

Image segmentation is the process of applying a labeling to each pixel in an image to determine a list of boundaries of objects, areas of interest, or silhouettes of objects in the scene. The resulting pixel labeling enables higher level vision systems to perform complex actions such as gesture recognition for human-computer interfaces Ueda et al. (2003), real-time 3D volume reconstructions Laurentini (1994), and mixed reality applications Takaya et al. (2009). Without accurate and coherent segmentations, these higher end-to-end systems will fail to perform their required functions. This necessitates a method which can correctly identify and extract objects from a scene with high accuracy. However, accuracy usually comes with serious performance penalties, particularly in image processing which examines hundreds of thousands of pixels in each image. This is unacceptable for high-level vision systems, which generally require the processing to be completed in real time.

By creating a graph using information in the image, as in Fig. 1, researchers have recast the problem of energy minimization and image segmentation as a graph theory problem. Then, methods such as flow analysis and finding shortest paths give researchers information about the structure of the underlying image. The results of these methods have been very promising, although the problem of computation complexity (speed) remains. Techniques such as *livewire* Mortensen & Barrett (1995), *normalized cuts* Shi & Malik (1997), and *graph cuts* have been developed to produce more accurate image segmentations and silhouette extractions.

Applying graph cuts to layered frames of a video sequence has shown to be robust for object tracking and object segmentation. In such scenarios, the entire video sequence is treated as a single 3D volume of layered frames, and a cut is performed across the entire sequence at once Boykov & Funka-Lea (2006). In sight of the large size of the graphs, which is increased with the video sequence length, performance of the cut is of paramount importance. To solve this problem, advances in the graph cut algorithm have produced dramatic performance improvements Boykov & Kolmogorov (2004); Juan & Boykov (2006).

However, regarding live video applications, there is no prior knowledge of subsequent frames. Techniques that build single graph across the entire video Boykov & Funka-Lea (2006) become then inapplicable in live video scenarios. Instead, a graph must be maintained or



(a) Image with seeds.

(d) Segmentation results.

(b) Graph.

(c) Cut.

Fig. 1. Graph based silhouette extraction, from Boykov & Funka-Lea (2006).

build at each frame. When considering live real-time video, using faster cut algorithms, is not enough sufficient. The creation of the graph, as well as the structure for utilizing temporal data from previous frames must be reconsidered.

Although recent researches have given much attention to graph cuts algorithms, most of them are too computational expensive to reach real-time, limiting their applications to off-line processing. Moreover, with the introduction of the Compute Unified Device Architecture (CUDA), GPUs are no longer exclusively designed for 3D rendering. In CUDA, a GPU can be explicitly programmed as general-purpose shared memory Single Instruction Multiple Data (SIMD) multi-core processors, which allows a high level of parallelism. Thus, many applications that are not yet able to achieve satisfactory performance on Central Processing Units (CPUs), can get benefit from that massive parallelism provided by such devices. Nevertheless, only specific algorithms can be adapted and efficiently designed on GPUs.

In this chapter, we propose the design and implementation of graph cuts that is capable of handling both tracking and segmentation simultaneously, leading to real-time conditions. We first propose to track and segment objects in live video thereby utilizing regional graph cuts and object pixel probability maps, where the graph is dynamically adapted to the motion of objects. We then introduce a fast and generic novel method for graph cuts on GPUs that places no limits on graph configuration, and which is able to reach real-time performance.

2. Background on graph cuts

Graph-based segmentation has a history originally based in mathematics and graph theory. Since graphs are an abstraction of the problem space, they are often applied to many different domains. In addition to image segmentation, graph cuts have been applied to computer networking, selection maximization, and stereo vision. The graph theory that supports graph-based segmentation methods is described, followed by a survey of frequently used algorithms, and a discussion of common graph construction techniques.

Before applications to image segmentation were demonstrated, mathematicians had been interested in computation of flows through networks. Networks were modeled as directed graphs with weights (also called *capacities*) assigned to the edges. The first important discovery in graph-based segmentation occurred in 1956, when Elias *et al* Elias et al. (1956), and Ford and Fulkerson Ford & Faulkerson (1956) independently proved the *max-flow min-cut theorem*. This theorem proved that solving for the maximum flow across a network is equivalent to solving for the minimum cut required to separate a network into two disjoint components.

2.1 Max-flow min-cut theorem

The max-flow min-cut theorem defines the relationship between the maximum flow on a *flow network*, and the minimum cut required to separate the network into two disjoint components. A *flow network* is a graph $G =< V, E >$ composed of edge set E and a vertex set V. The vertex set V contains two labeled vertices, the source s, and the sink t, and each edge $\vec{uv} \in E$ has a capacity c_{uv}, and the flow f between two vertices u and v is never more than the capacity, $f(u,v) \leq c_{uv}$. Then for a path $P_{u_0 u_n} = \{u_0\vec{u}_1, u_1\vec{u}_2, \ldots, u_{n-1}\vec{u}_n\}$ where $u_0, u_1, \ldots, u_n \in V$, the flow is defined as:

$$f(P_{uv}) = \min_{\vec{ij} \in P_{uv}} c_{ij} \tag{1}$$

Then the maximum flow of the network G is defined as the sum of path flows for all paths from s to t:

$$f(G) = \sum_{\forall P_{st} \in G} f(P_{st}) \tag{2}$$

Next, an *s-t cut* is defined as a partition of the vertices V into two sets $(W, \bar{W})$ such that $s \in W$ and $t \in \bar{W}$ and no edges exists from W to $\bar{W}$, as shown in Fig. 2 if the *forward edges* are removed. The max-flow min-cut theorem states that the value v of the maximum flow is equal to the value of the minimum *s-t cut* Papadimitriou & Steiglitz (1998).

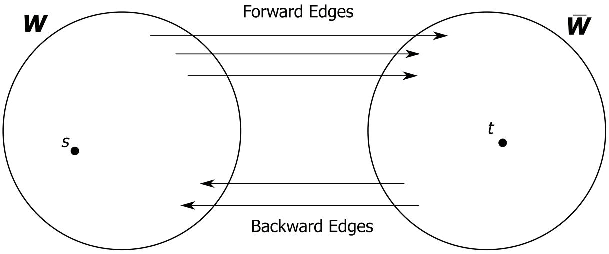


Fig. 2. Graph *s-t* partitioning

While the theorem was proved in 1956, algorithms allowing for efficient computer computation of very large graphs continued to be a focus of much research. Today, general categories of algorithms for solving the max-flow/min-cut exist and are often the basis of additional research.

2.2 Max-flow algorithms

Along with the proof of the max-flow min-cut theorem in Ford & Faulkerson (1956), Ford and Fulkerson presented an algorithm for computing the maximum flow of a flow network. This algorithm forms the basis of a set of algorithms known as *Ford-Fulkerson* or *augmenting paths* algorithms. Then, in 1988 Goldberg and Tarjan presented a new way of approaching the computation of flow with their *push-relabel* algorithm Goldberg & Tarjan (1988).

The time complexities for two *augmenting paths* algorithms and two *push-relabel* algorithms are presented in Table 1. The two classes of algorithms differ in time complexity based on the number of vertices V, or edges E. Ford-Fulkerson algorithms would generally perform better on sparse graphs (fewer edges) than do push relabel algorithms. At the time of writing, one of the fastest (and freely available) implementations is an *augmenting paths* algorithm utilizing breadth-first-search (BFS) search trees Kolmogorov & Zabin (2004).

Algorithm	Complexity	Notes
Ford & Faulkerson (1956)	$O(VE^2)$	
Ford & Faulkerson (1956)	$O(Ef)$	Only for
Edmonds & Karp (1972)	$O(V^2E)$	integer capacities BFS
Push-Relabel, Goldberg & Tarjan (1988)	$O(V^2E)$	
Push-Relabel w/ FIFO Queue, Goldberg & Tarjan (1988)	$O(V^3)$	

Table 1. Time Complexity of Maximum Flow Algorithms.

2.3 Augmenting paths algorithm

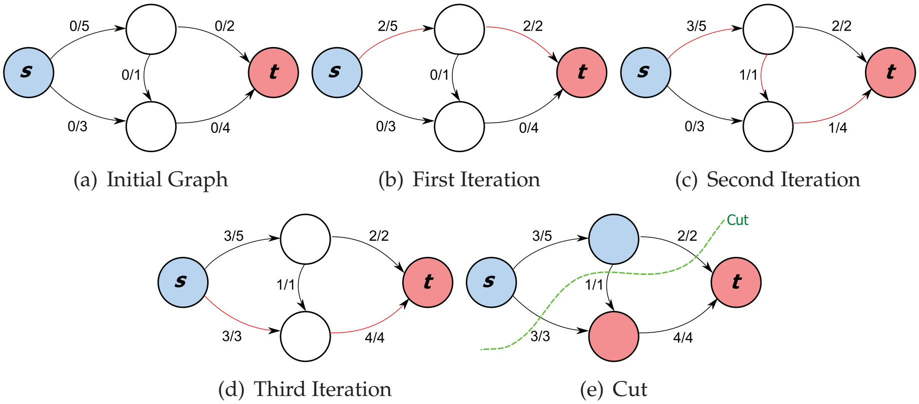
The *augmenting paths* algorithm solves the maximum flow problem by iterating searches for paths from s to t and pushing flow equal to the minimum capacity along the path. The algorithm terminates when no paths with remaining capacity exist from s to t. The pseudo-code for the algorithm is presented in Algorithm 1.

Algorithm 1: Augmenting Paths Algorithm.

```
   input : Graph $G$ with flow capacity $c$, source vertex $s$, sink vertex $t$
   output: A flow $f$ from $s$ to $t$ which is maximum
 1 forall the $\vec{uv} \in E$ do
 2     $f(u,v) \leftarrow 0$
 3 end
 4 while $\exists$ path $P$ from $s$ to $t$ such that $c_{uv} > 0$ for all $\vec{uv} \in P$ do
 5     $c_f(p) = \min(c_{uv} | \vec{uv} \in P)$
 6     foreach $\vec{uv} \in P$ do
 7         $f(u,v) \leftarrow f(u,v) + c_f(p)$        /* Send flow along the path */
 8         $f(v,u) \leftarrow f(v,u) - c_f(p)$
 9     end
10 end
```

The difference between the Edmonds-Karp algorithm and the Ford-Fulkerson algoirthm in Table 1 lies in the method used to find paths from s to t. In step 2 in Algorithm 1, the Edmonds-Karp algorithm uses a BFS search method originating from vertex s.

Fig. 3 presents a simple case of an augmenting paths algorithm. Fig. 3(a) shows the initial graph. The graph has been constructed with a source vertex and a sink vertex, and all flow has been initialized to zero. In the next panel, the first path from the source to sink is highlighted in red, and the minimum capacity has been pushed as flow. The result is that the link connecting to the sink has been fully saturated, and this path is no longer usable. This continues through panel three and four, iteratively pushing flow along *s-t paths* until no unsaturated path remains. Then the cut separates those vertices that still have an unsaturated path back to the source from those that do not.



(a) Initial Graph (b) First Iteration (c) Second Iteration

(d) Third Iteration (e) Cut

Fig. 3. Example Augmenting Paths Algorithm Execution.

2.4 Push relabel algorithm

Whereas the *augmenting paths* algorithm examines entire *s-t paths*, the *push relabel* algorithm examines individual vertices and attempts to *push* flow through the graph network. The *push relabel* algorithm relies on generating a *pseudo-flow* (a flow that does not obey the properties of the graph) and finding a convergence on a true flow. The algorithm adds two pieces of information to each vertex, a height h and an excess e. The height of the vertex is a distance measurement, reflecting how *close* the vertex is to the source. The excess is the difference between incoming and outgoing flow of a vertex, or in other words, the flow that is trapped or waiting at the vertex. A vertex with $e > 0$ is said to be *overflowing*. The algorithm consists of two main operations: *push* (also called *discharge*) and *relabel* (also called *raise*). Typically a queue of overflowing vertices is checked each iteration, with the first vertex in the list being dequeued and either *discharged* or *relabeled*. The loop terminates once there are no further viable *discharge* or *relabel* operations to perform (i.e. no remaining overflowing vertices).

Discharging a vertex causes some of the excess to be pushed to neighboring vertices when the incident edge has not completely saturated (has residual capacity). The push decreases the excess in the current vertex, increasing excess in the destination, and decreasing the residual capacity of the edge. The destination vertex is then enqueued into the overflowing list, along with the current vertex if it is still overflowing.

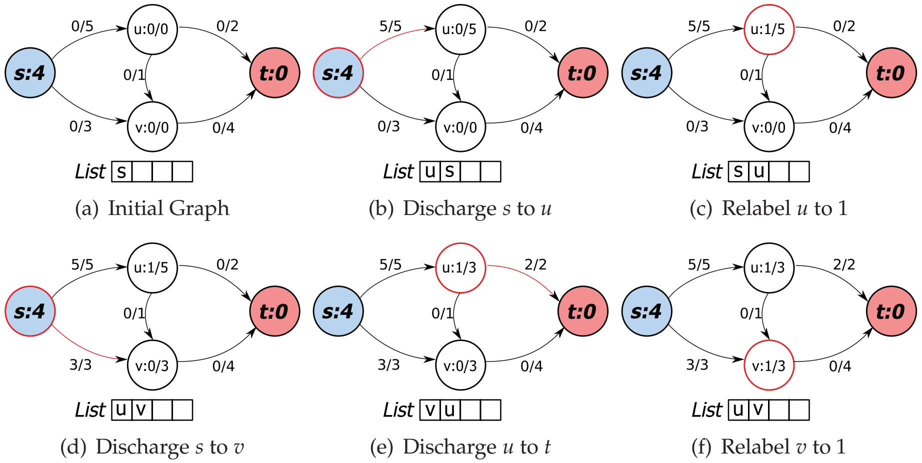
A vertex must be *relabeled* when it is overflowing but has no valid neighbors. A neighbor is valid if the edge incident to both vertices has remaining capacity, and the destination vertex's height is lower than the current vertex. The current vertex's height is increased to one more than the lowest neighbor, which allows further *discharging*. If a vertex has no remaining outgoing capacity, then it is relabeled to the height of the source plus one, so that the excess may be pushed back to the source.

Algorithm 2: Push Relabel Algorithm.

```
input  : Graph $G$ with flow capacity $c$, source vertex $s$, sink vertex $t$
output: A flow $f$ from $s$ to $t$ which is maximum
list.enqueue($s$)
while not list.isEmpty() do
    $u \leftarrow$ list.dequeue()
    if $u$.excess $> 0$ then
        amtPushed $\leftarrow$ discharge($u$)
        if amtPushed $== 0$ then
            relabel($u$)
        end
    end
end
```

In Algorithm 2, the enqueueing of more vertices into the list would occur inside the *discharge* and *relabel* functions. Various techniques to change or improve performance have been proposed, such as pushing to the highest neighbors first, or using a first-in-first-out (FIFO) queue for vertex selection. The general time complexity given in Table 1 would seem to say that the *push relabel* algorithm is more efficient than *augmenting paths*, especially for sparse graphs, but in practice this did not seem to be the case, until recently, with parallelized implementations of the algorithm.

Fig. 4 demonstrates the operations of the *push relabel* algorithm. All vertices have their information printed alongside their label. The source and sink have their height listed after the semicolon, while vertices u and v have their height listed, follow by their current excess (after the slash). The currently active vertex is highlighted in red. There appear to be many more steps than *augmenting paths*, but this is not necessarily the case, since the path finding steps were not detailed in the previous section.



(a) Initial Graph (b) Discharge s to u (c) Relabel u to 1

(d) Discharge s to v (e) Discharge u to t (f) Relabel v to 1

Fig. 4. Example Push Relabel Algorithm Execution.

2.4.1 Parallel push relabel algorithm

As recently as 2008, researchers discovered methods for parallelization of the *push relabel* family of maximum flow algorithms. Most of the strategies implement methods so that the vertices in the overflowing list can be processed concurrently, vastly speeding up the performance of the algorithm. The parallelization of any algorithm often involves many complex concurrency issues, requiring control structures (often referred to as locks) so that the order of execution of certain statements is guaranteed. However, a lock-less implementation of the push relabel has been presented by Hong Hong (2008), which will be presented here. The algorithm constructs the graph with a *pre-flow*, effectively saturating all outgoing links from s.

Algorithm 3: Lock-free, Parallel Push Relabel Algorithm: Master Thread.

```
   input  : Graph $G$ with flow capacity $c$, source vertex $s$, sink vertex $t$
   output: A flow $f$ from $s$ to $t$ which is maximum
   /* Initialization, performed by master thread                    */
 1 $s$.height $\leftarrow |V|$
 2 foreach $u \in V - \{s\}$ do
 3     $u$.height $\leftarrow 0$
 4 end
 5 foreach $\vec{uv} \in E$ do
 6     $f(u,v) \leftarrow 0$
 7 end
 8 foreach $\vec{su} \in E$ do
 9     $f(s,u) \leftarrow c_{su}$
10     $u$.excess $\leftarrow c_{su}$
11 end
```

As stated by Hong, lines 14-17 must be performed by atomic *read-update-write* operations. These operations are usually available on multi-core systems, or are implementable via software (though at a cost). Lastly, Hong notes that while a single thread may terminate because of no available operations, it does not mean that the algorithm has terminated. Other vertices may push flow back into the vertex, making it necessary for the thread to continue operations. This problem will be address in the GPU implementation in Section 3.2.

2.5 Building graphs from images

While the max-flow min-cut theorem was first proved in 1956, it was not applied to computer vision and image segmentation for more than 30 years. The key to applying the max-flow min-cut theorem image segmentation is the ability to represent the image as a meaningful flow network, so that the min-cut can be translated back into an image segmentation. One of the first successful image processing applications was developed by Greig et al Greig et al. (1989), which demonstrated success in estimating solutions for noise removal. Then Boykov et al Boykov et al. (2001) demonstrated that the min-cut of a graph can correspond to the maximum a posteriori distribution of a Markov Random Field, which had been used for image segmentation in the past. By formulating image segmentation as an energy minimization problem, and then a max-flow min-cut problem, Boykov et al created *graph cuts* for image segmentation.

Algorithm 4: Lock-free, Parallel Push Relabel Algorithm: Child Threads.

```
input  : A vertex $u$
output: None
/* Main loop performed per thread, one thread per vertex      */
while $u$.excess $> 0$ do
    $\acute{e} \leftarrow u$.excess
    $\hat{v} \leftarrow$ null
    $\hat{h} \leftarrow \infty$
    foreach $\vec{uv} \in E_f$ do                          /* edges with capacity */
        $\acute{h} \leftarrow v$.height
        if $h\prime <\hat{v}$.height then
            $\hat{v} \leftarrow v$
            $\hat{h} \leftarrow \acute{h}$
        end
    end
    if $u$.height $> \hat{h}$ then                                  /* Discharged */
        $d \leftarrow \min\left(\acute{e}, c_{uv} - f(u,v)\right)$
        $f(u,v) \leftarrow f(u,v) + d$
        $f(v,u) \leftarrow f(v,u) - d$
        $u$.excess $\leftarrow u$.excess $-d$
        $v$.excess $\leftarrow v$.excess $+d$
    else                                                          /* Relabel */
        $u$.height $\leftarrow \hat{h} + 1$
    end
end
```

The graph cut creation process consists of splitting the edges of the graph into two categories. All edges that are incident to the source or sink are called *t-links* (terminal links), while the other links are called *n-links* (neighbor links). The two different types of links are assigned weights (energy) using different methods, and give rise to the energy equation which is to be minimized by the graph cut.

$$E(A) = \lambda R(A) + B(A) \tag{3}$$

where

$$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p) \tag{4}$$

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{p,q} \cdot \delta_{A_p \neq A_q} \tag{5}$$

Equation 4 defines the regional energy, which corresponds to the t-links of the graph. This energy is often a model of the object to be extracted. Research has been conducted into various object models such as Gaussian mixture models Rother et al. (2004) and color histograms Garrett & Saito (2008). One simple method for t-link energy computation uses histograms. For a given pixel p and a histogram H, the histogram value for p, $H(p)$, is divided by the maximum histogram value to obtain an energy between $[0.0, 1.0]$ as demonstrated in Equation 6.

$$R(p) = \frac{H(p)}{\max\limits_{\forall q \in V} H(q)} \tag{6}$$

Equation 5 is the boundary energy, corresponding to the n-links in the graph. The most common equation used for the boundary energy is the square of the Euclidean distance between two pixels. Equation 7 demonstrates the energy between two RGB pixels ($p^T = [r, g, b]$), normalized to the range $[0.0, 1.0]$. This produces a linear energy model, which is often not optimized for practical use, as the difference between colors captured in real scenes does not exhibit enough variance in the RGB color space. Instead negative exponential functions (Equation 8) can replace the Euclidean difference.

$$B(\{p,q\} \in V) = 1.0 - ||p - q||/3.0 \tag{7}$$

$$B(\{p,q\} \in V) = e^{-||p-q||} \tag{8}$$

Fig. 5 demonstrates the differences in boundary energy equations. The x axis is the difference between the pixels ($||p - q||$), and the y axis is the resultant energy. The exponential function decreases much faster for small differences of pixel intensity earlier than does the linear function.

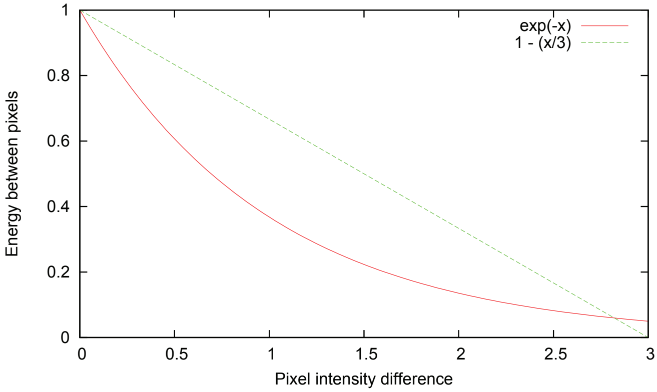


Fig. 5. Boundary Equation Comparisons

Finally, graph construction can be summed up by the energy table by Boykov et al Boykov & Funka-Lea (2006), which has been reproduced here. The $\mathcal{K}$ in Table 2 refers to a maximum energy level, computed as:

$$\mathcal{K} = 1 + \max_{p \in \mathcal{P}} \sum_{q \in \mathcal{N}_p} B_{pq} \tag{9}$$

This equation defines that links may have hard constraints, limiting them to either the foreground or background by ensuring that their neighbor links are fully saturated due to the large amount of energy in the t-link from the source, or the large drain to the sink.

Edge Weight (Energy)		For
$\vec{pq}$	B_{pq}	$\{p, q\} \in V$
$\vec{ps}$	$\lambda \cdot R_p(''bkg'')$	$p \in \mathcal{P}, p \notin O \cup B$
	$\mathcal{K}$	$p \in O$
	0	$p \in B$
$\vec{pt}$	$\lambda \cdot R_p(''obj'')$	$p \in \mathcal{P}, p \notin O \cup B$
	0	$p \in O$
	$\mathcal{K}$	$p \in B$

Table 2. Edge Energy Computations per Edge Type.

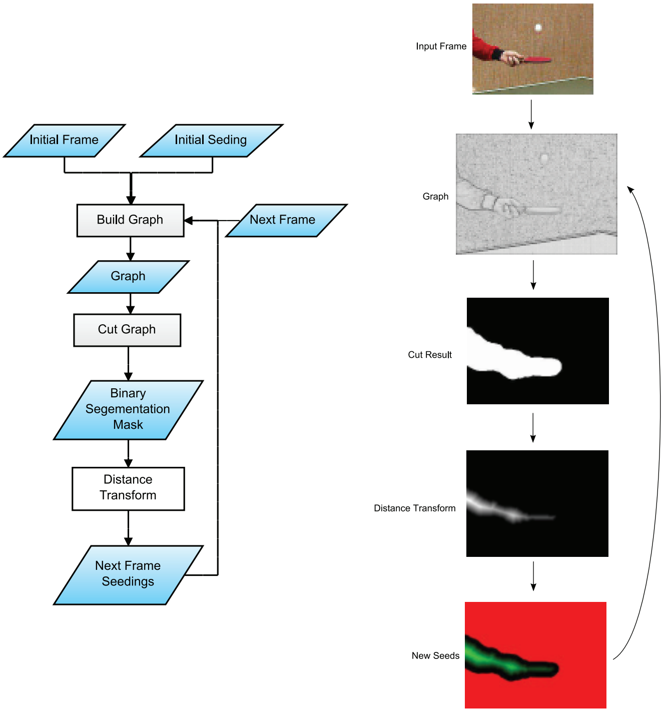


Fig. 6. Flow Chart of Tracking System.

3. Proposed system

3.1 Live video extraction

The method for tracking and silhouette extraction is composed of three main steps: building the graph, cutting the graph, and creating temporal seed information for the next frame. These three steps are performed in a cycle, with the temporal seeds being used in the next build phase, shown in Fig. 6. The images on the right show the image representation of data at each step of the algorithm.

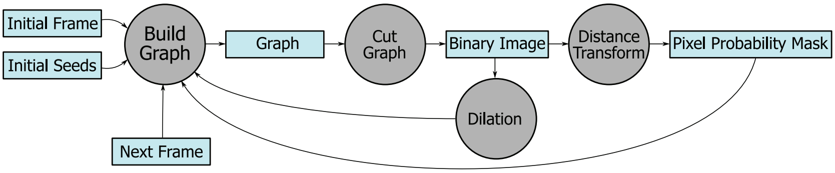


Fig. 7. System Operation Flow Chart.

3.1.1 Build the graph

Each edge in the graph requires an energy computation, and as the size of the image increases, the number of edges increases dramatically. For example, an 8-neighborhood graph on a 640x480-resolution video has almost 2 million edges. This results in poor performance on live video because 2 million edges must be recomputed for every frame. If an object is tracked and segmented that does not fill the entire frame, many of the pixels will not need to be computed, such as those far from the object, since they will not likely change unless the object moves closer. To reduce the number of computations required, the graph is restricted to a region-of-interest.

For the very first cut on the graph, the region-of-interest will cover the entire video frame. However, subsequent regions will be smaller, depending on the object size.

3.1.2 Cut the graph

The second step is to cut the graph. For this step, a freely available graph cut implementation by Kolmogorov *et al* Kolmogorov & Zabin (2004) was used. This implementation relies on a modified *augmenting paths* algorithm, as described earlier.

3.1.3 Temporal seeding

Seeding the graph for the next frame is not a simple matter of marking the previous frame's object pixels as object seeds, and all other pixels as background seeds. If the object were to move within the next frame, then some seeds would be mislabeled, also causing the cut to incorrectly label some pixels. This research describes a new method for seeding pixels based on probabilities of future pixel locations.

When examining any two consecutive frames of a video sequence, many of the pixels of the object will overlap between frames. These will most likely be those pixels closer to the center of the object, as the object must travel farther before these pixels change from object to background. Thus, a probability distribution can be created over the object pixels such that the pixels near the edge will have a lower probability of remaining object pixels in the next video frame. Pixels near the center of the object have a high probability of remaining object pixels in the next video frame. This distribution we call the object pixel probability map.

To compute the pixel probability map, the binary mask of the object segmentation from the previous frame is computed. Then a distance transform is applied to the mask to obtain the pixel probability map, shown on the right hand side of Fig. 7. The distance transform computes the distance from a given white pixel to the closest black pixel. This gives us a gradient from the center of the object (greatest distance) to the cut edge (shortest distance).

Using the probability map, the t-links of the next frame's graph are computed using a modified version of Region Equation R described by Equation 5. Table 3 shows the new energy computations for t-links.

Edge	Weight (Energy)	For
$\{p,s\}$	$\min(\lambda+1, \lambda \cdot R_p(bkg) + \lambda \cdot D_p)$	$p \in P, p \notin O$
	0	$p \in O$
$\{p,t\}$	$\min(\lambda+1, \lambda \cdot R_p(bkg) + \lambda \cdot D_p)$	$p \in P, p \notin B$
	0	$p \in B$

Table 3. Modified Boundary Energy Equation

Where D_p is the value of the probability map for pixel p divided by the largest D, and thus $D_p \in [0.0, 1.0]$. The technique also requires that $R_p \in [0.0, 1.0]$ to make sure that the maximum value of the t-link weight is $\lambda + 1$ (labeled as K in Table 2).

Finally, a region-of-interest is constructed around the object area of the binary image mask. The region-of-interest is then expanded to contain probable future locations of the object. Metrics for determining the size of potential for the region-of-interest include camera frame rate, previous object velocities, and object scaling speeds.

3.1.4 Focused graph cuts

Graph cuts are notoriously computationally expensive. The graph size grows exponentially with image size (image size growth means growing in width and height), and easily includes hundreds of thousands of vertices and millions of edges, as seen in Fig. 8. To alleviate this problem, the proposed system focuses only on those pixels which are most likely to contain the object from the previous frame. By utilizing a dilation of the previous graph cut's resultant binary image, the operations of function graph cuts ca be masked, resulting in a smaller graph on which to operate. Similar methods which use banded graph cuts, cutting only around the boundary of the object, have been proposed Lambaert et al. (2005); Mooser et al. (2007), however these methods have been shown to lose information and misclassify pixels in certain topologies Garrett & Saito (2008); Sinop & Grady (2006).

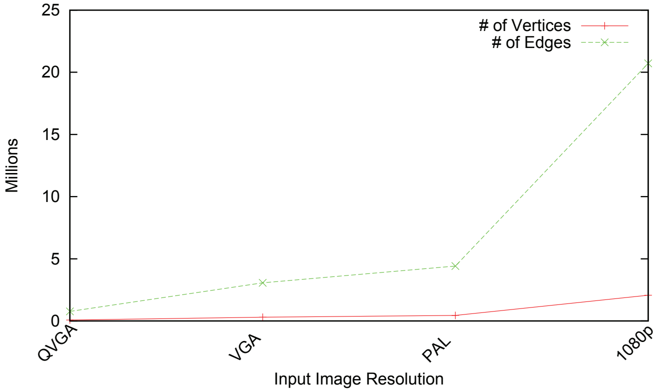


Fig. 8. Graph Growth Size.

Initial timing analysis of graph cuts shown in Fig. 9 demonstrates that the building of the graph consumes a majority of the time, as many floating point operations must be performed (typically slow on CPUs). To speed up the graph cuts even more, the next section details how to harness the parallelization and floating computations of the GPU.

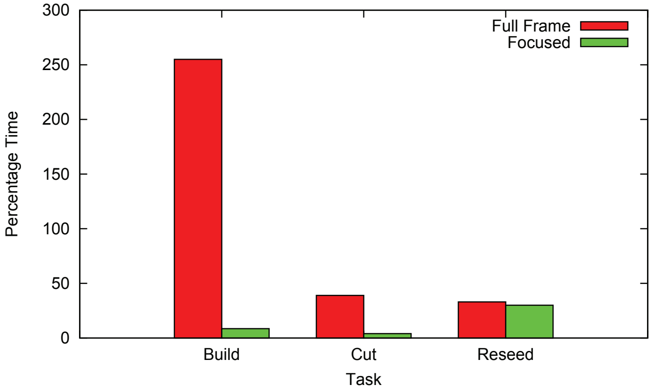


Fig. 9. Graph Timing per Task.

3.2 GPU implementation using CUDA

Driven by the insatiable market demand for real-time, high-definition 3D graphics, the programmable GPU has evolved into a highly parallel, multi-threaded, many-core processor with tremendous computational horsepower and very high memory bandwidth.

The reason behind the discrepancy in floating-point capability between the CPU and the GPU is that the GPU is specialized for compute-intensive, high-parallel computation, and therefore designed such that more transistors are devoted to data processing rather that data caching and flow control.

NVidia's CUDA allows researchers to easily parallelize their computing task to take advantage of the GPU. CUDA provides a software library used for interfacing with the graphics processor, and a specialized compiler to create executable code to run on the GPU. CUDA presents programmers with two other types of memory access, shared memory and global memory, on top of the texture cache units. Being able to access to all the memory on the GPU (albeit with varying access times) has made it much easier to perform research into GPGPU computing. The method presented in this paper was developed using the CUDA architecture and run on nVidia hardware. There are many technical resources available on the nVidia CUDA homepage nVidia (2009).

The following description of the CUDA model is stated in terms of its hardware and software components. The hardware model, shown in Fig. 10, illustrates how each processor *core* is laid out in relation to the different memory buses of global memory, texture cache, and shared memory. Device (global) memory is the slowest to access while the texture cache is only available for read operations. Shared memory is used by all of the threads in a block, as described by the software model.

The software model, depicted in Fig. 11, shows the how a *block* organizes *threads*, and how the *grid* of a kernel is organized into *blocks*. When an invocation of kernel code is to be called, CUDA requires specifying the dimensions of the *grid* and the size of *blocks* within the grid (each block is the same size). A collection of *threads* that will be run in parallel on the SIMD cores is defined as a *warp*. Operations such as branching will cause threads to diverge into separate warps which must be run serially in time-shared environments. Each thread and each block have a two dimensional ID that can be accessed at any time. This ID is often used to reference memory locations pertaining to the particular thread.

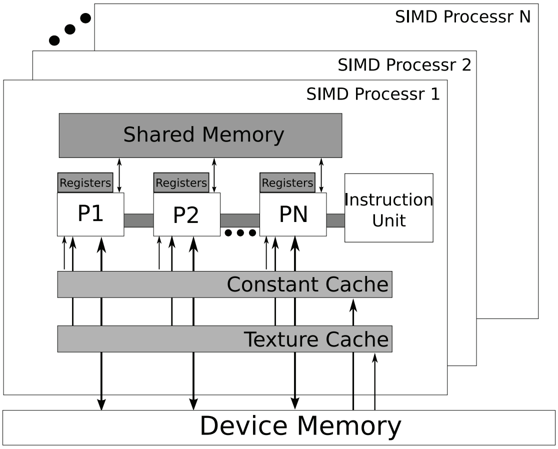


Fig. 10. CUDA Hardware Model

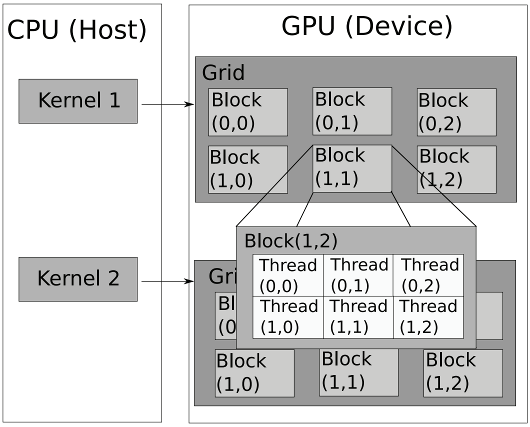
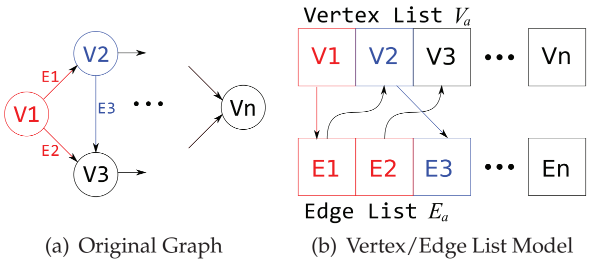


Fig. 11. CUDA Programming Model

3.2.1 GPU Programming paradigm

To develop a GPU implementation of a traditional algorithm, both data structure layout (the graph representation in memory) and concurrency bottlenecks must be carefully considered to realize the full potential of the GPU. CUDA implementations of graph cuts for image segmentation have traditionally paired the 2D lattice structure of graphs with the 2D grid structure of the CUDA programming model. However this leads to restrictions on the types of graphs that can be processed by the routine. Furthermore, traditional graph cut algorithms contain too much cyclomatic complexity in the form of branching and looping. In CUDA, divergent code produces poor performance because the SIMD model cannot perform the instructions in parallel, making new techniques for graph cuts necessary.

Vineet and Narayanan presented a method of graph cuts in CUDA that has shown improved performance by pairing the 2D lattice graphs with the CUDA programming model. They conceded that this method would not be feasible for other types of graphs Vineet & Narayanan



(a) Original Graph (b) Vertex/Edge List Model

Fig. 12. Conversion of a direct graph into vertex and edge lists.

(2008). In this research, the graphs were structured using a technique similar to that presented by Harish et al Harish & Narayanan (2007) that improved performance in distance calculations on arbitrary graphs using the GPU by representing graph $G(V, E)$ as a vertex array V_a and an edge array E_a. Edges in the edge list are grouped so that all edges that originate at a vertex are contiguous, and the vertex contains a pointer to the first edge in the group, as in Fig. 12. Each edge structure holds information about its capacity, current flow, and destination vertex.

3.2.2 Regular graphs

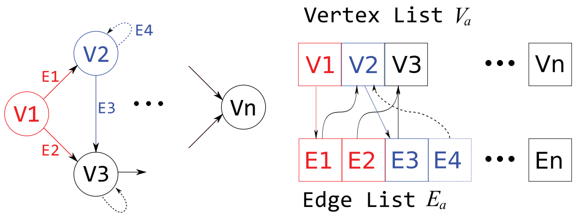
The technique used in this research improved upon the basic implementation of vertex and array lists. To make the list-based graph representation more SIMD friendly, the input graph is made *regular* by adding null edges to any vertex that has fewer edges than the vertex with the highest degree. Second, the code is simplified by only having one kernel, which performs both the discharging and relabeling steps.

Since the GPU is treated as an array of SIMD processors, it is important that the algorithm is able to perform the same operation across multiple data locations. However, this assumes that any kind of graph configuration could be used as input, requiring that each vertex be treated uniquely, as the vertex set is not homogeneous. Particularly in the case of image segmentation, there are three classes of vertices: center vertices, edge vertices, and corners vertices, which have 10, 7, and 5 edges respectively (in an 8-neighborhood graph). Since the vertices must be treated differently, the CUDA framework cannot perform the operations in parallel and it serializes the kernel calls, negating the benefit of using the GPU.

To overcome this problem, the new method used in this research adds extra edges to the graph so that each vertex has the same degree. In Fig. 13, the dotted line of edge $E4$ is an example of a null edge. These null edges point back to the originating vertex and have a capacity of zero. This causes the null edges to be ignored by the graph cut algorithm, as the destination of the edge never has a height lower than the origin (a condition for discharging), and the capacity of the edge restricts the amount of flow discharged to zero.

3.2.3 GPU graph cuts on regular graphs

Algorithm 5 details the steps of the computation that take place on the CPU. The program operates on a list of vertices V, and a list of edges E. The *initialize* function starts by discharging the maximum possible amount of flow from the source to the neighboring vertices. Then the GPU kernel is invoked so that one thread is excuted for each non-source and non-sink vertex. The CPU continues to invoke the GPU kernel until no discharge or relabel operations can be performed.



(a) 2-regular Original Graph (b) Vertex/Edge List Model

Fig. 13. Addition of null edges to achieve a k-regular graph.

Algorithm 5: Proposed System Host Graph Cut Loop

```
finished ← false
Initialize(V, E)
while not finished do
    // GraphCutKernel() is performed in parallel on the GPU

    finished ← GraphCutKernel(V,E)
end
```

The GPU kernel is detailed in Algorithm 6. Since the graph is regular, the loop over the neighbors is fully unrolled. The null edges are ignored because the height of u will never be less than the current vertex's height (since they are equal). In addition the discharge can be modified to push a conditionally assigned amount of flow. Conditional assignment allows us to use of the ternary operator, which prevents divergent code since it can be optimized to a zeroing of either operand of an addition.

Algorithm 6: Proposed System GPU Kernel

```
tId ← GetThreadID()
amtPushed ← 0
foreach Neighbor u of V[tId] do
    if u.height < V[tId].height and $\overrightarrow{(V[tId], u)} \in E$ has capacity > 0 then
        amtPushed ← amtPushed + Discharge(V[tId], u)
    end
end
if amtPushed > 0 and V[tId].excess > 0 then
    V[tId].height = FindLowestNeighbor(V[tId]).height + 1
end
```

4. Experimental results

The GPU experiment analyzes the performance of the GPU push relabel implementation described in Section 3.2. The goal of the implementation is to be able to realize the results of the graph cut faster than the traditional CPU implementations. To measure the performance benefit of the GPU approach, a pure GPU implementation was compared against the previous CPU implementation and against a hybrid CPU/GPU system.

This section begins with the drawbacks of the GPU approach. Then, as remarked at the end of the CPU test analysis, the two areas identified for improvement are the graph building and the graph cutting areas, which are covered in the following two sections respectively.

These tests utilized two video sequences scaled to QVGA (320x240), VGA (640x480), HD 720p (1280x720), and HD 1080p (1920x1080) resolutions.

All tests in this section were performed on an 2.5 GHz Intel Xeon processor with 2 GB of RAM and a nVidia GTX 280 graphics card running Windows XP. This information is important in analyzing the performance results in the following sections, as results are likely to differ with hardware.

4.1 GPU transfer latency

Due to computer hardware architecture, the memory on the GPU is not directly accessible from the CPU. An important limitation of the older GPUs was the time required to transfer the input to the GPU memory and the time required to transfer the GPU output back to the CPU. This latency from transferring data is not applicable to the CPU-only implementations of graph cuts, since all the data resides in the CPU main memory. Traditional Advanced Graphics Port (AGP) technology facilitated fast data transfer to the GPU, but the return path to the CPU was considered unimportant, since the output would then be written to the display device. Even with the new Peripheral Component Interconnect Express (PCIe) bus architecture, at very large image resolutions, minor latency is observed. Fig. 14 demonstrates that for even large HD 1080p (1920x1080) resolution images the absolute maximum frame rate (due to transfer latency) would be approximately 30 frames-per-second.

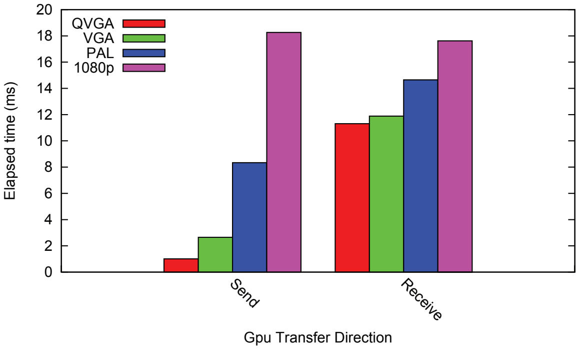


Fig. 14. Graph of Elapsed Time per GPU Task

4.2 GPU graph building

Due to the parallelism in GPUs, even greater performance benefits can be obtained using CUDA. The GPU performed so well during this task that the histogram bars cannot be seen, and instead the values are written. Furthermore, unlike focused graph cuts, this task is content independent since it works on the entire frame at the same time.

Table 4 gives exact performance numbers for the results shown in Fig. 15. In the final row, the speed up multiplier is computed as the ratio between the GPU implementation and the focused graph cuts implementation. For HD 1080p video, GPU graph building achieves as

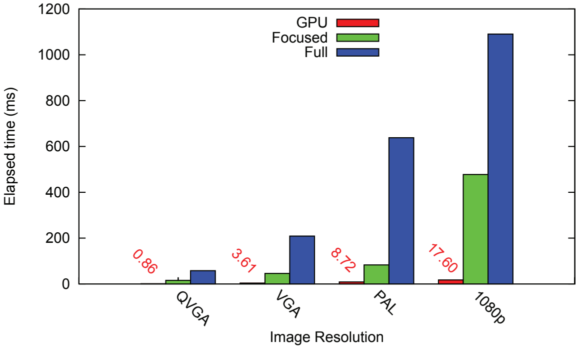


Fig. 15. Build Times for Various Image Resolutions

high as a 26x speed increase over the focused graph building, and even greater over traditional graph building methods.

	QVGA	VGA	720p	1080p
Full	57.96	208.91	638.07	1090.52
Focus	15.59	45.82	82.99	477.55
GPU	0.86	3.61	8.72	17.60
Speedup	17.5x	12.7x	9.5x	26.1x

Table 4. Speed Results for GPU Graph Building.

4.3 GPU push relabel

This section compares the cut speeds of the GPU implementation of the system. Fig. 16 shows that at smaller resolutions, the GPU implementation does not see significant improvement in speed. As the raw clock speed of the CPU is higher than that of the GPU (4x greater), this is to be expected. However, at the higher resolutions (720p and 1080p), the GPUs parallelization capabilities start to have an impact on the performance.

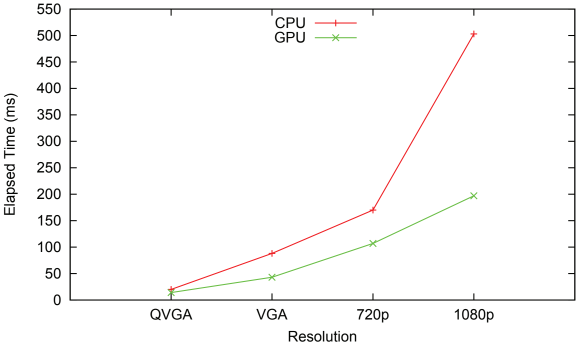


Fig. 16. Cut Times for Various Image Resolutions.

Table 5 gives exact performance numbers for the results shown in Fig. 16.

		QVGA	VGA	720p	1080p
	Avg	20	88	170	503
CPU	Min	15	78	156	453
	Max	32	94	170	503
	Avg	14	43	107	197
GPU	Min	11	39	91	180
	Max	18	47	117	207

Table 5. Speed Results for GPU Graph Building.

4.4 GPU Summary

Experimental data show that the GPU implementation is able to perform on average 9 times faster than the CPU implementation for large resolutions (see Table 6). Standard video broadcasts (up to VGA) could be segmented with very little delay as the frame rates approach 15 fps. However, HD video is still beyond the reach of real-time systems, averaging only 3 fps.

	QVGA	VGA	720p	1080p
CPU	150	739	1739	3094
GPU	23	69	186	336
Speedup	6.5x	10.7x	9.3x	9.2x

Table 6. Summary Results for Full System Performance Tests.

Fig. 17 contains sample frames from output video, scaled to fit the page, and scaled relative to each other in order to realize the size difference in resolutions. The segmentation boundary is designated by the red border around the yellow airplane. The ocean and islands are segmented as background, and the airplane is segmented as the object of interest. The videos are samples obtained from the Microsoft *WMV HD Content Showcase* website Microsoft (2009).

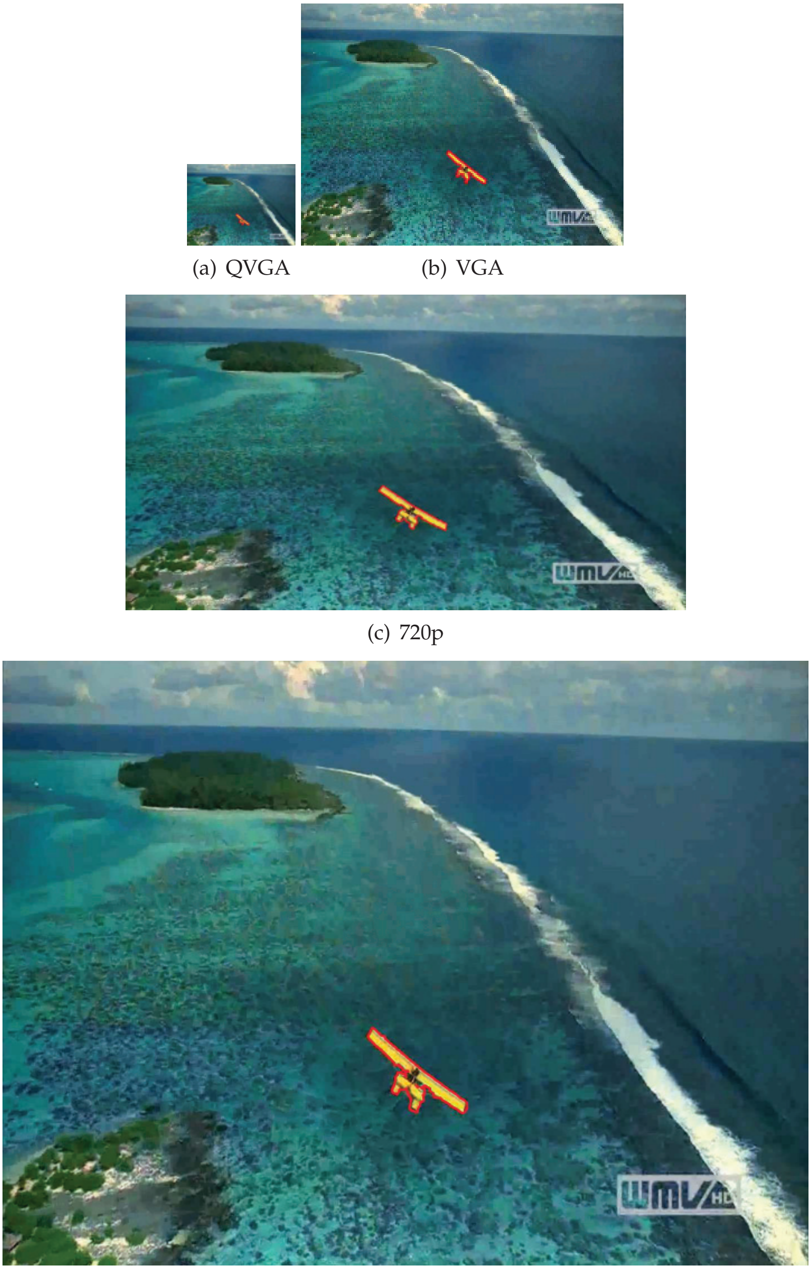
5. Conclusions

This chapter has presented research on a novel system for object silhouette extraction. The originality of the system comes from three distinct method improvements. First, the research developed a method for tracking and segmenting objects across a sequence of images by using temporal information and distance transforms. Second, the method performance was increased by focusing the graph cut on likely image pixels, reducing the graph size to only those pixels that are considered useful. Finally, creating an implementation on the GPU using vertex and edge lists, as opposed to 2D memory arrays, and making the graph regular by adding *null* edges further improved performance.

Through testing on various image sequences and the standardized Berkeley segmentation data set, the research showed the new system provides fast and accurate results. Future research extensions include alternative color models for objects and new boundary energy equations. Both hold promise for better segmentation in the occlusion case, but must be balanced with computational complexity, otherwise the improvement in speed realized by the system will be negated.

6. Acknowledgments

This work is supported by the National Institute of Information and Communications Technology (NICT), Japan.



(a) QVGA

(b) VGA

(c) 720p

(d) 1080p

Fig. 17. Sample Output of GPU Video Sequence Performance Experiment.

7. References

Boykov, Y. & Funka-Lea, G. (2006). Graph cuts and efficient n-d image segmentation, *International Journal of Computer Vision* 70(2): 109–131.

Boykov, Y. & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, *IEEE Trans. Pattern Anal. Mach. Intell.* 26(9): 1124–1137.

Boykov, Y., Veksler, O. & Zabih, R. (2001). Fast approximate energy fast approximate energy minimization via graph cuts, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(11): 1222–1239.

Edmonds, J. & Karp, R. (1972). Theoretical improvements in algorithmic efficiency for network flow problems, *Journal of the ACM* 19(2): 248–264.

Elias, P., Feinstein, A. & Shannon, C. (1956). A note on the maximum flow through a network, *IRE Transactions on Information Theory* 2(2): 117–119.

Ford, L. & Faulkerson, D. (1956). Maximal flow through a network, *Canadian Journal of Mathematics* 8: 399–404.

Garrett, Z. & Saito, H. (2008). Live video object tracking and segmentation using graph cuts, *International Conference on Image Processing*, IEEE, pp. 1576–1579.

Goldberg, A. & Tarjan, R. (1988). A new approach to the maximum-flowproblem, *Journal of the ACM* 35(4): 921–940.

Greig, D., Porteous, B. & Seheult, A. (1989). Exact maximum a posteriori estimation for binary images, *Journal of the Royal Statistical Society* 51(2): 271–279.

Harish, P. & Narayanan, P. J. (2007). Accelerating large graph algorithms on the gpu using cuda, *IEEE High Performance Computing*, pp. 197–208.

Hong, B. (2008). A lock-free multi-threaded algorithm for the maximum flow problem, *Workshop on Multithreaded Architectures and Applications, in Conjunction With International Parallel and Distributed Processing Symposium*, IEEE Computer Society.

Juan, O. & Boykov, Y. (2006). Active graph cuts, *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, pp. 1023–1029.

Kolmogorov, V. & Zabin, R. (2004). What energy functions can be minimized via graph cuts?, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(2): 147–159.

Lambaert, H., Sun, Y., Grady, L. & Xu, C. (2005). A multilevel banded graph cuts method for fast image segmentation, *IEEE Conference on Computer Vision*, Vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, pp. 259–265.

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(2): 150–162.

Microsoft (2009). Wmv hd content showcase, *Microsoft Corporation* .
URL: *http://www.microsoft.com/windows/windowsmedia/musicandvideo/hdvideo/contentshowcase.aspx*

Mooser, J., You, S. & Neumann, U. (2007). Real-time object tracking for augmented reality combining graph cuts and optical flow, *International Symposium on Mixed and Augmented Reality*, Vol. 00, IEEE, IEEE Computer Society, pp. 1–8.

Mortensen, E. & Barrett, W. (1995). Intelligent scissors for image composition, *22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 191–198.

nVidia (2009). Cuda zone, *nVidia* .
URL: *http://www.nvidia.com/object/cuda_home.html*

Papadimitriou, C. & Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*, Courier Dover Publications.

Rother, C., Kolmogorov, V. & Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts, *ACM Transactions on Graphics* 23: 309–314.

Shi, J. & Malik, J. (1997). Normalized cuts and image segmentation, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 731–737.

Sinop, A. & Grady, L. (2006). Accurate banded graph cut segmentation of thin structures using laplacian pyramids, *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Vol. 9, pp. 896–903.

Takaya, Y., Garrett, Z. & Saito, H. (2009). Player segmentation in sports scenes using graph cuts with a playing field constraint, *Meeting on Image Recognition and Understanding*.

Ueda, E., Matsumoto, Y., Imai, M. & Ogasawara, T. (2003). A hand-pose estimation for vision-based human interfaces, *IEEE Transactions on Industrial Electronics* 50(4): 676–684.

Vineet, V. & Narayanan, P. J. (2008). Cuda cuts: Fast graph cuts on the gpu, *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 1–8.

Part 2

Applications

10

Parsing Object Events in Heavy Urban Traffic

Yun Zhai, Rogerio Feris, Arun Hampapur, Stephen Russo, Sharath Pankanti
Physical Security, IBM Corp.
United States

1. Introduction

Due to rapid development of video capturing and management technologies and decreasing cost of the hardware infrastructure, digital cameras are nowadays widely deployed in cities, industry campuses, public facilities, etc. In particular, a vast amount of cameras are installed in urban environments to monitor traffic flows. One important video analytics task in traffic monitoring is to capture and track moving objects, and archive their trajectories in the database for future forensic activities, such as event search. Different from conventional event capture technologies, which are mostly based on explicit object tracking, we propose a new technique for capturing moving object using a specialized virtual tripwire.

Virtual boundaries (many times referred as tripwires) are defined as continuous lines or curves drawn in the image space to capture directional crossings of target objects. Due to their intuitive concept and reliable performance under controlled conditions, they are widely used in many digital surveillance applications. In border security, tripwires are set-up along the border line to detect illegal entries from outer perimeter. In urban surveillance, tripwires are often used to achieve many goals, including vehicle/pedestrain counting, wrong-way traffic detection, alerting with pre-defined object attributes, etc. In retail environments, store managements usually obtain the display-effectiveness of a certain merchandize by analyzing the triggering frequencies of the tripwire placed around that item. Conventional tripwire techniques consist of three main steps:

- Object detection: during this process, the target object is spatially segmented in each input image.
- Object tracking: once objects are localized in the frames, they are further correlated to each other within the temporal domain to achieve the trajectories representing the spatial paths of the objects.
- Tripwire crossing detecting: this is the process of determining if the object trajectory intersects with the tripwire in the defined crossing direction(s).

Since reliable object detection and tracking are the corner stones of conventional tripwires, its performance is extremely dependent upon the success of the first two steps. When there is a dense traffic pattern in the scene, object detection and tracking generally fail, and consequently, the tripwire crossing detection no longer provides accurate and meaningful results.

1.1 Related work

Object detection and tracking are long studied problems in the field of computer vision. In this section, we briefly review some of the well known techniques in these areas. One popular type of object detection techniques is the background subtraction (BGS), where active objects (foreground) are detected as the difference regions between the current image and the background reference. Color and gradient distributions are often applied to model the pixels. Some representative work in this area are Stauffer and Grimson (8), Tian *et.al.* (9) and Cucchiara *et.al.* (2). Another common trend in object detection is to localize the target objects in single images by applying pre-learned templates. The object templates are convolved with the input image with scale and orientation variations. One of the most cited work in this category is by Viola and Jones (12) in real-time face detection.

Once objects are localized in images, their correspondences in the temporal domain are established by object tracking techniques. Tomasi and Kanada (10) proposed the KLT tracker for point object tracking. Another popular interest-point tracking method, Scale-Invariant Feature Transform (SIFT), is proposed by Lowe (6). Comaniciu *et.al.* (1) have proposed the Mean-Shift tracking by utilizing the mode-seeking and kernel density estimation techniques. A geometric shape based matching method is developed by Fieguth and Terzopoulos (3). A survey on object tracking methods can be found at Yilmaz *et.al.* (13). Utilizing different object tracking techniques, tripwire is widely applied in many commercial digital video surveillance systems (5; 7; 11).

1.2 Proposed approach

As aforementioned, object detection and tracking usually fail in the dense traffic scenarios. To address the problem of degraded tripwire performance in these situations, we propose a new mechanism to approach the tripwire crossing detection. The proposed new type of tripwire is referred as the “trackerless tripwire”, i.e., it does not rely on explicit object tracking results. Different from conventional tripwires which have zero width, our proposed tripwire contains an “envelope” that covers a spatial extend on both sides of the tripwire, and crossing detection is carried out by analyzing the spatiotemporal and appearance properties of the tripwire envelope. Two sets of *ground patches*, called *inner patches* and *outer patches*, are automatically calibrated and generated along the envelope on both sides of the defined tripwire. Each of the patches maintains a history of appearance and motion models. Patch models are cross-checked to find matching candidates. The incorrect candidates are further filtered out by applying a motion consistency test. A set of spatiotemporal rules are also applied to eliminate potential false positive detections. The new proposed trakerless tripwire has been deployed in scenarios with different traffic loads, and promising results have been obtained for both low and high activity scenes.

The remainder of this paper is organized as follows: Section 2 presents the proposed new tripwire, including tripwire construction, feature extraction and crossing detection; Section 3 demonstrates the performance of the proposed method and its comparison against conventional tracking-based tripwire method; finally, Section 4 concludes our work.

2. Trackerless tripwire crossing detection

Numerous object tracking techniques have been developed over the past decades. The common goal of these techniques is to achieve accurate spatiotemporal segmentation of the foreground at the object-level. In crowded videos, such segmentation is extremely challenging and very difficult to accomplish. In the context of tripwire crossing detection, full-scale object tracking across the entire image field-of-view (FOV) many times is an overkill for the problem.

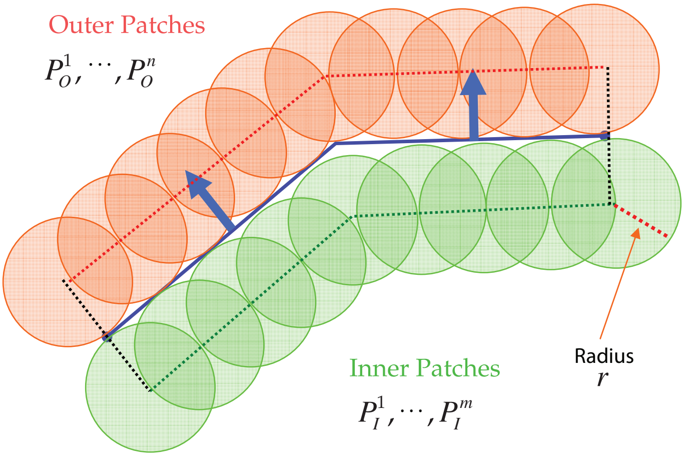


Fig. 1. The proposed trakerless tripwire with inner and outer patches. An envelope is generated along the defined tripwire, and the ground patches are equally sampled on the envelope. The thick arrows represent the defined tripwire crossing direction.

Rather, localized analysis on the video content around the tripwire area is often more reliable and efficient. In our method, we take this as our guiding direction and propose a new type of tripwire detection method that does not reply on explicit object tracking output.

2.1 Ground patches

The proposed tripwire is defined as a set of connected line segments. To model the regions around the tripwire, an envelope is created with a width of $2r$, where r is a scale factor representing the spatial extend of the tripwire region. To capture the local contents of the tripwire envelope, two sets of circular ground patches are equally sampled on both sides of the tripwire. They are denoted as the *Inner Patches* $\mathbf{P}^I$ and *Outer Patches* $\mathbf{P}^O$, where $\mathbf{P}^I = \{P_1^I, \cdots, P_m^I\}$ and $\mathbf{P}^O = \{P_1^O, \cdots, P_n^O\}$. The directions "inner" and "outer" are defined with respect to the order of the tripwire vertices. A graphical illustration is shown in Figure 1. Each ground patch P_i has a center $c_i = (x_i, y_i)$ and a unified radius r.
The centers of the ground patches lie on the edges of the tripwire envelope such that they have the same distance to the tripwire line. In addition to the center and radius, each patch P_i also has a directional vector w_i pointing to the tripwire from the patch center. The purpose of this vector is to assist in the crossing direction determination. Overall, a ground patch P_i is represented as a triplet,

$$P_i = (c_i, r, w_i). \tag{1}$$

An object crosses the tripwire in the *positive* direction if it moves from one of the inner patches to one of the outer patches. Same concept applies to the *negative* crossing. In this paper, the proposed tripwire crossing detection method is demonstrated in the context of *positive* crossings.

2.2 Feature extraction

Both appearance and motion features are extracted for each ground patch. We use the color histogram in the RGB space for the patch appearance model and the motion vector to represent the moving direction of the object inside the patch. A ground patch P_i maintains a history H_i of its appearances,

$$H_i = \{h_i^{T_0}, \cdots, h_i^{T_t}\}, \tag{2}$$

where h^T is the color histogram of the target patch at time T and these histograms are recorded in the chronological order, i.e., $T_x < T_y$ if $x < y$. To avoid including the background information (such as the road surface in a street scene) in the patch model, the model history is updated only if when the patch is occupied by one or more foreground objects. This is determined by analyzing the percentage of the foreground area inside the target patch based on common motion detection techniques such as frame differencing or Multi-Gaussian background subtraction (MG-BGS). If the foreground occupies a sufficient proportion of the patch, the patch is declared to have a "foreground" status, and its current color histogram is extracted for potential model update. In our implementation, we utilize the MG-BGS output of (9) for the motion detection.

The patch history model is an on-line process. When a new color histogram h^{T_+} is extracted at time T_+, it is compared to the latest model h^{T_t} in the history, if there is any. The distance measure used in the model comparison is defined as the Bhattacharyya distance, where $D(h_1, h_2) = \sum_{\text{all bins i}} \sqrt{h_1^i \times h_2^i}$. If the distance between the new model and the last model in the history is significant, then the new model is included in the model history. One exception to this rule is when there is a period that the patch is classified as "background" between the current time and the last time the model history was updated. In this situation, even if the new model has small distance to the last model in the history, they may be caused by two different objects due to the discontinuity in the patch status. Thus, the new patch model should also be included in the history.

Once the color histogram feature is extracted at time T_+ and successfully added to the model history, the motion vector of the target patch is also estimated to determine the moving direction of the object inside the patch. Simple approaches like the Sum of Squared Differences (SSD) could be used to estimate the motion vectors. Let this vector be v^{T_+} and coupled with the color histogram h^{T_+}. The overall feature representation of a patch P_i is,

$$\mathbf{\Gamma}_i = \{\Gamma_i^{T_0}, \cdots, \Gamma_i^{T_t}\}, \tag{3}$$

where $\Gamma_i^\tau = (h_i^\tau, v_i^\tau)$ is the feature vector at time τ.

2.3 Tripwire crossing detection

To determine if the tripwire crossing occurs in the positive direction, patch feature matching is performed by computing the similarity between the target outer patch P_o at the current time T_+ and the inner patches P_i on the other side of the tripwire. In order to comply with the crossing direction, the object in patch P_o should possess a motion opposite to the direction pointing to the tripwire from the patch center,

$$\text{Condition } C_d := v_o^{T_+} \cdot w_o < 0. \tag{4}$$

If condition C_d is not satisfied, that means the object is moving in the opposite direction as the tripwire crossing direction definition, and no match is performed for this patch.

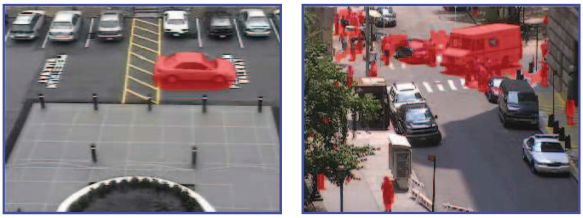


Fig. 2. Keyframes of example videos. (a) Sparse traffic scene; (b) Crowded activity scene. Red overlaid regions in the image represent the areas occupied by foreground moving objects.

If condition C_d is satisfied, the current feature $\Gamma_o^{T_+}$ is compared against with the history features $\mathbf{\Gamma}_i$. The feature matching is composed of two parts: (1) color similarity $S_{\mathbf{c}}$ and (2) motion direction consistency $S_{\mathbf{m}}$, where

$$S_{\mathbf{m}}(\tau) = \frac{v_o^{T_+} \cdot v_i^{\tau}}{\parallel v_o^{T_+} \parallel \times \parallel v_i^{\tau} \parallel}, \tag{5}$$

and,

$$S_{\mathbf{c}}(\tau) = \begin{cases} 0, & \text{if } S_{\mathbf{m}}(\tau) \leq 0, \\ 1 - D(h_o^{T_+}, h_i^{\tau}), & \text{otherwise.} \end{cases} \tag{6}$$

The overall patch-to-patch similarity $\mathbf{S}(P_o, P_i)$ is then computed as a weighted sum of both the similarity measures,

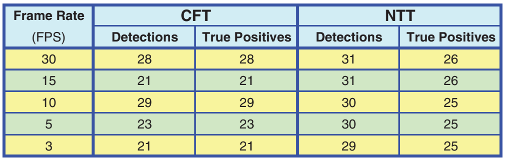
$$\mathbf{S}(P_o, P_i) = max\big(\alpha_{\mathbf{c}} S_{\mathbf{c}}(\tau) + \alpha_{\mathbf{m}} S_{\mathbf{m}}(\tau)\big), \forall \tau < T_+ - T_{th}. \tag{7}$$

where $\alpha_{\mathbf{c}}$ and $\alpha_{\mathbf{m}}$ are the fusion weights of the appearance similarity and motion consistency, respectively. Threshold T_{th} indicates the minimum possible temporal interval for an object to travel from a patch on one side of the tripwire to another patch on the other side. Based on the empirical analysis, we have found that the patch appearance is more distinctive than the motion consistency cue. Thus, a higher weight is specified for the color similarity $S_{\mathbf{c}}$. If the patch similarity $\mathbf{S}$ is above the desired threshold, a tripwire crossing in the positive direction is detected with τ contributing the maximum matching score.

3. Performance evaluation

To demonstrate the effectiveness of the proposed tracker-less tripwire, we have compared the performances of the proposed method and a conventional tracker-based tripwire in two cases: (1) sparse traffic scenes and (2) dense traffic scenes. For the conventional tripwire, we utilize the tripwire feature provided by the IBM Smart Surveillance System (4). The core components in this feature are (1) a background subtraction based object detection process, (2) an appearance based object tracking method and (3) a conventional tripwire based on the object tracking outputs. For the proposed trakerless tripwire, the same foreground modelling technique is used to classify the status of the patches. In this section, the tripwire of the IBM SSS solution is referred as "CFT", and the proposed tripwire is called "NTT".

The activity level of the scene is qualitatively defined. In sparse traffic scenes, the moving objects are apart from each other, i.e., there is a minimal amount of occlusions in the scene. In this case, object detection process is able to achieve clean object-level segments in the video, and consequently, the object tracking result is clean and reliable. A keyframe of a sparse traffic



Frame Rate	CFT		NTT	
(FPS)	Detections	True Positives	Detections	True Positives
30	28	28	31	26
15	21	21	31	26
10	29	29	30	25
5	23	23	30	25
3	21	21	29	25

Fig. 3. **Sparse Traffic**: Accuracy summary of the trackerless tripwire (NTT) and the conventional tripwire (CFT).

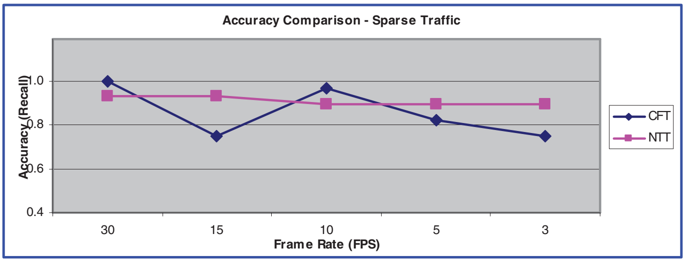


Fig. 4. **Sparse Traffic**: Accuracy comparison between the trackerless tripwire (NTT) and the conventional tripwire (CFT).

video is shown in Figure 2.a. On the other hand, high activity scenes are those that contain a significant amount of occlusions and/or the objects are close to each other, such that object detection can only produce a group-level segmentation. Therefore, it is very challenging to the object tracking module to generate meaningful trajectories that could be used by the tripwire detection module. A keyframe of one crowded scene is shown in Figure 2.b.

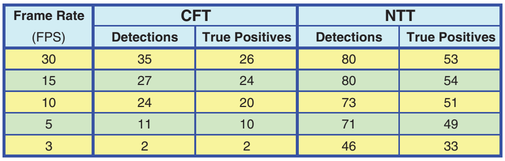
3.1 Performance in sparse traffic situations

The purpose of this test is to determine whether the proposed trackerless tripwire detection method could maintain the same performance as the conventional methods, as the object tracking in low activity scenes are reliable. The goal of the test is to accurately count the objects that cross the tripwire in the defined direction. Our testing video is obtained from a street scene in an urban environment. In this video[1], there are totally 28 vehicles in the ground truth, including sedans, SUVs, buses, etc. They are all moving in the same direction. Furthermore, since there is no traffic light, the vehicles are continuously moving forward without any stop. Two tripwires are deployed at the same location to capture the number of passing vehicles. One is the CFT and the other one is NTT. We used the standard recall measure for the accuracy analysis.

$$\text{Accuracy} = \frac{\text{Num. of True Positives}}{\text{Num. of Ground Truth}}. \tag{8}$$

We have also generated a set of videos with different frame rates using the same testing video. The purpose of this is to understand the robustness of the approaches against frame-dropping defects in real situations. The testing frame-rates are 30 frames-per-second (FPS), 15, 10, 5 and 3 FPS.
Detailed tripwire detection results are presented in Figure 3. This table lists the total detections and the true positive samples produced by CFT and NTT. Accuracy comparison between these

[1] Due to privacy issue, the images of the testing data used in this paper cannot be disclosed for public use.



Frame Rate	CFT		NTT	
(FPS)	Detections	True Positives	Detections	True Positives
30	35	26	80	53
15	27	24	80	54
10	24	20	73	51
5	11	10	71	49
3	2	2	46	33

Fig. 5. **High Activity**: Accuracy summary of the trackerless tripwire (NTT) and the conventional tripwire (CFT).

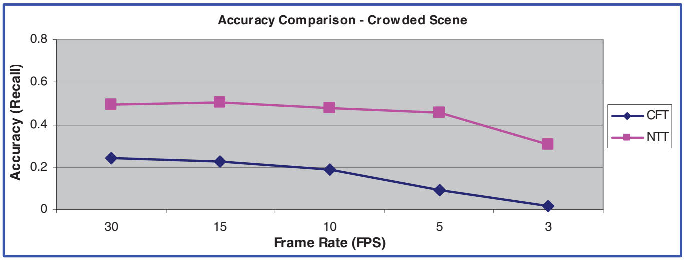


Fig. 6. **High Activity**: Accuracy comparison between the trackerless tripwire (NTT) and the conventional tripwire (CFT).

two methods is illustrated in Figure 4. Based on the results, the mean accuracies of the CFT and NTT methods are 0.85 and 0.91, respectively. From the frame-rate comparison graph, NTT performs in a more stable manner. This is because that as the frame-rate drops, object tracking quality drops as well.

3.2 Performance in crowded scenes

The second test is to compare the performances of CFT and NTT in a high activity environment. In high activity scenarios, object detection and tracking often do not provide meaningful object-level spatiotemporal segmentation. Therefore, conventional tracker-based tripwire crossings cannot be detected in this case. In this test, we apply both CFT and NTT on a video recorded at a heavy traffic street. There are totally 107 vehicles in the video, and all traffic are going in the same direction. In addition to the high density of the traffic, there is a traffic light in the scene so that cars are periodically stopped and piled up. That is extremely challenging for the object tracking module. On the other hand, our proposed tripwire does not rely on the tracking results. It analyzes the appearance and motion patterns just around the target tripwire region. That way, many of the irrelevant information is filtered out automatically, and cleaner detection results can be obtained. Figure 5 shows the detailed tripwire detection results of CFT and NTT, and their accuracy measure comparison is shown in Figure 6.

The performance of the conventional tripwire CFT is merely acceptable to the end users (0.15 in average). The superior performance of the proposed trackerless tripwire is apparent from the performance summary (average 0.45 for NTT). For all frame-rates, NTT over-performed CFT by detecting 25% more tripwire crossing events.

3.3 Failing case analysis

From the performance results presented in Figure 5, a high percentage of false positives is observed for the trackerless tripwire detection method. There are several reasons associate with this. The main reason is the unified patch size. This parameter is fixed in the detection process. If a single object has a size that is much greater than this parameter (e.g., a big bus), what the patch captures is only a small portion of the object. Thus, multiple events are detected for different parts of the same object based on the current feature extraction and matching formulation. One way to improve this is to have a grouping method to intelligently cluster the events that correspond to the same object into a single trigger. Patch motions can be applied to discover the relative movement of the detected objects. Events could be clustered if they possess steady spatial layout.

Another cause for the false positives is the background modelling error. In our testing video, there is a traffic light, and cars are often stopped and be static for awhile. Due to background updates, the object pixels actually became the background model. Therefore, after the vehicles leave the scene, the true background, road surface, is now classified as foreground by the BGS process. Since road surface has strong similarity in color, they are often mis-classified as objects that cross the tripwire. This could be improved by incorporating a more sophisticated BGS process.

4. Conclusions

In this paper, we have proposed a new tripwire detection method. The proposed framework is targeting the challenging problem of reliably detecting tripwire crossings in situations where explicit object tracking fails (e.g., in crowded scenes). Two sets of ground patches are generated along the tripwire to represent the content of the tripwire's proximity. Both appearance and motion information are incorporated in the feature models of the patches. By analyzing the similarities between the patches on the opposite sides of the tripwire, a strong matching candidate can be found to indicate the crossing event. The proposed new tripwire has been deployed in both low and high activity scenes, and very promising results have been obtained to demonstrate its effectiveness.

5. References

[1] D. Comaniciu, V. Ramesh and P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean-Shift", *IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

[2] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting Moving Objects, Ghosts, and Shadows in Video Streams", *IEEE T-PAMI*, 25:(10), 2003.

[3] P. Fieguth and D. Terzopoulos, "Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates". *IEEE Int'l Conf. on CVPR*, 1997.

[4] A. Hampapur *et.al.,*, "Multi-Scale Tracking for Smart Video Surveillance", *IEEE T-SP*, Vol.22, No.2, 2005.

[5] IntelliVision, http://www.intelli-vision.com/.

[6] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *IJCV*, 2004.

[7] ObjectVideo VEW, http://www.objectvideo.com/products/ vew/capabilities/.

[8] C. Stauffer and W. Grimson, "Learning Patterns of Activity Using Real-Time Tracking", *IEEE T-PAMI*, 1999.

[9] Y-L. Tian, M. Lu and A. Hampapur, "Robust and Efficient Foreground Analysis for Real-time Video Surveillance", *IEEE Int'l Conf. on CVPR*, 2005.

[10] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features". CMU Tech Report CMU-CS-91-132, 1991.

[11] Verint, http://www.verint.com/video_solutions/.

[12] P. Viola and M. Jones, "Robust Real-time Object Detection", *IJCV*, 2001.

[13] A. Yilmaz, O. Javed and M. Shah, "Object Tracking: A Survey", *Computing Surveys of ACM*, Vol.38, No.4, 2006.

11

Object Tracking in Multiple Cameras with Disjoint Views

Pier Luigi Mazzeo and Paolo Spagnolo
Istituto sui Sistemi Intelligenti per l'automazione (CNR)
Italy

1. Introduction

The specific problem we address in this chapter is object tracking over wide-areas such as an airport, the downtown of a large city or any large public area. Surveillance over these areas consists of the search for suspicious behavior such as persons loitering, unauthorized access, or persons attempting to enter a restricted zone. Until now, these surveillance tasks have been executed by human being who continually observe computer monitors to detect unauthorized activity over many cameras.It is known that the attention level drastically sinks after few hours. It is highly probable that suspicious activity would be unregistered by a human operator. A computer vision system, can take the place of the operator and monitor both immediate unauthorized behavior and long-term suspicious behavior. The "intelligent" system would, then, alert a responsible person for a deeper investigation. In most cases, it is not possible for a single camera to observe the complete area of interest because sensor resolution is finite and structures in the scene limit the visible areas. In the realistic application, surveillance systems which cover large areas are composed by multiple cameras with non overlapping Field of Views (FOVs). Multiple camera tracking is important to establish correspondence among detected objects across different cameras. Given a set of tracks in each camera we want to find which of these tracks belong to the same object in the real world. Note that tracking across multiple cameras is a challenging task because the observations are often widely separated in time and space, especially when viewed from non overlapped FOVs. Simple prediction schemes cannot be used to estimate the object position. It could be observed that object's appearance in one camera view might be very different from its appearance in another camera view due to the difference in illumination, pose and camera properties. It is preferable that any multi-view tracking approach does not require camera calibration or complete site modelling. In fact, the benefit of calibrated cameras or site models is unavailable in most situations. Maintaining calibration between a large sensors network is a discouraging task, it is very onerous recalibrate any sensor when it (sensor) slight changes its position.

This chapter presents an algorithm that deals with all described constraints and tracks objects across multiple un-calibrated cameras with non-overlapping FOVs. We investigate different techniques to evaluate intra-camera and inter-camera tracking algorithm based on object appearances. Object appearance can be modelled by its color or brightness, and it is a function of the scene illumination, object geometry, object surface material properties and camera parameters. Only the object surface, among these variables, remains constant whereas an object moves across cameras.

We compare different methods to evaluate the color Brightness Transfer Function (**BTF**) between non overlapping cameras. These approaches are based on the color histogram mapped among pairs of images of the same person in different FOVs. It is important to point up that our proposed inter-camera appearance models for tracking do not assume:

- Explicit camera calibration,
- A site model,
- Presence of a single ground plane across cameras,
- A particular non-overlapping camera topology,
- Constant illumination,
- Constant camera parameters, for example, focal length or exposure.

It is, definitively, a flexible calibration-free model.

1.1 Related work

Most of the approaches presented in literature suppose the use of calibrated cameras and the availability of the site model. In (Javed et al., 2008) the conformity in the traversed paths of people and car is used to establish correspondence among cameras. The algorithm learns this conformity and hence the inter-camera relationships in the form of multivariate probability density of spacetime variables (entry and exit locations, velocities, and transition times) using kernel density estimation. To handle the appearance change of an object as it moves from one camera to another, the authors demonstrate that all brightness transfer functions, which map one camera in an another, lie in a low dimensional subspace. This subspace is learned by using probabilistic principal component analysis and used for appearance mapping. In (Du & Piater, 2006) particle filters and belief propagation are combined in a unified framework. In each view, a target is tracked by a dedicated particle-filter-based local tracker. The trackers in different views collaborate via belief propagation so that a local tracker operating in one view is able to take advantage of additional information from other views. In (Du & Piater, 2007) a target is tracked not only in each camera but also in the ground plane by individual particle filters. These particle filters collaborate in two different ways. First, the particle filters in each camera pass messages to those in the ground plane where the multi-camera information is integrated by intersecting the target principal axes. This largely relaxes the dependence on precise foot positions when mapping targets from images to the ground plane using homographies. Secondly, the fusion results in the ground plane are then incorporated by each camera as boosted proposal functions. A mixture proposal function is composed for each tracker in a camera by combining an independent transition kernel and the boosted proposal function. Kalman filters are used in (Chilgunde et al., 2004) to robustly track each target shape and motion in each camera view and predict the target track in the blind region between cameras. For multi-camera correspondence matching, the Gaussian distributions of the tracking parameters across cameras for the target motion and position in the ground plane view are computed. Targets matching across camera views uses a graph based track initialization scheme, which accumulates information from occurrences of target in several consecutive video frames. Geometric and intensity features are used in (Cai & Aggarwal, 1999) to match objects for tracking in a multiple calibrated camera system for surveillance. These features are modelled as multivariate Gaussian, and Mahalanobis distance measure is used for matching. A method to match object appearances over non-overlapping cameras is presented in (Porikli, 2003).In his approach, a brightness transfer function (BTF) is computed for every pair of cameras. Once such mapping function is known, the correspondence problem is reduced to the matching of transformed histograms or appearance models. However, this mapping, i.e., the BTF varies from frame to frame depending on a large number of parameters which include illumination, scene geometry, exposure time, focal length and aperture size of each camera. Thus, a single pre-computed BTF cannot usually be used to match objects for moderately long sequences. An unsupervised approach to learn edge measures for appearance matching between non-overlapping views is presented by (Shan et al., 2005). The probability of two observations from two cameras being generated by the same or different object is computed to perform the matching. The main constraint of this approach is that the edge images of vehicles have to be registered together. Note that this requirement for registering object images could not be applicable for non rigid objects like pedestrians. A Cumulative Brightness Transfer Function (CBTF) is proposed by (Prosser et al., 2008) for mapping color between cameras located at different physical sites, which makes use available color information from a very sparse training set. A bi-directional mapping approach is used to obtain an accurate similarity measure between pairs of candidate objects. An illumination-tolerant appearance representation, based on online k-means color clustering algorithm is introduced in (Madden et al., 2007), which is capable of coping with the typical illumination changes occurring in surveillance scenarios. A similarity measurement is also introduced to compare the appearance representation of any two arbitrary individuals. In (Jeong & Jaynes, 2008) the distortion function is approximated as general affine transformation and the object appearance is represented as mixture of Gaussians. Appearance models are put in correspondence by searching a bijection function that maximizes a metric for model dissimilarity.

A common characteristic of the above related works is that the knowledge of model sites and particular camera positions in various scenarios allow the usage of geometrical and temporal constraints on the entry/exit image areas. In this way the appearance matching among different cameras is carried out on a sets of individuals that are candidate by their positions to be observed by distributed cameras.

2. Wording of disjoint views multi-camera tracking

Let us suppose that we have a system composed by n cameras $C_1, C_2, ..., C_n$ with non-overlapping views. Let us assume that q objects $P_1, P_2, ..., P_q$ are presented in the scene (the number of the objects in the scene is unknown). Each object is viewed from different cameras at different time instants. Let us call $O_j = \{O_{j,1}, O_{j,2}, ..., O_{j,m_j}\}$ the set of m_j observations that were viewed by the camera C_j. Each observation $O_{j,k}$ with $k = 1..m_j$ is generated by a moving object in the FOV of camera C_j. These observations consist of two part: object appearance $O_{j,k}(a)$ and space-time constraints of the object $O_{j,k}(st)$ (position, velocity, time, and so on). Let us assume, furthermore, that both $O_{j,k}(a)$ and $O_{j,k}(st)$ are independent of each other. Multi-camera tracking problem is centered on finding which of the observations in the system of cameras belong to the same object.

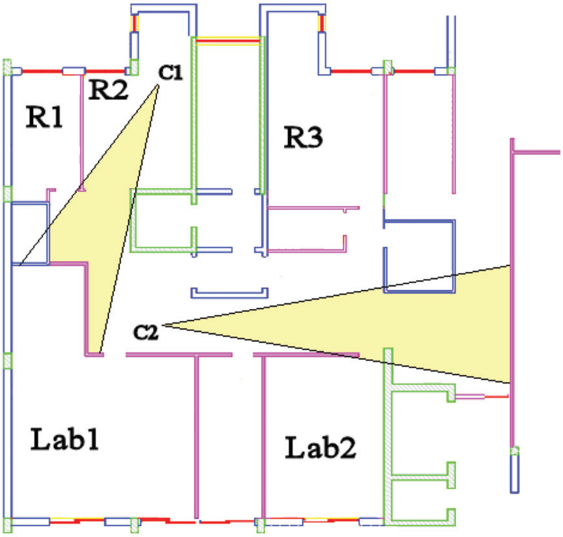


Fig. 1. The cameras configuration inside our office building

Seeing that we have assumed that the single camera tracking problem is solved, the multi-camera tracking task is to link the observations of an object exiting one camera to its observations entering another camera, as the object moves among different camera FOVs.
For a formal problem definition let consider a correspondence among two consecutive observations, i.e., exiting from one camera and entering into another,$O_{i,k}$ and $O_{j,l}$ will be denoted as $\lambda_{j,l}^{i,k}$. Let us consider $\psi(\lambda_{j,l}^{i,k})$ a random variable which is true if and only if $\lambda_{j,l}^{i,k}$ is a valid hypothesis, i.e. $O_{i,k}$ and $O_{j,l}$ are two observations of the same object. We want to find a set of correspondences $\Lambda = \{\lambda_{j,l}^{i,k}, ...\}$ where

$$\lambda_{j,l}^{i,k} \in \Lambda \Longleftrightarrow \beta(\lambda_{j,l}^{i,k}) = true$$

Let Σ be the solution space of the multi-camera tracking problem. If Λ is a candidate solution in Σ, then for all $\{\lambda_{j,l}^{i,k}, \lambda_{q,s}^{p,r}\} \subseteq \Lambda$ where $(j,l) \neq (i,k) \wedge (q,s) \neq (p,r)$. This is because we established that each observation of an object is preceded or succeeded by a maximum of one observation of the same object. Now, let consider Ψ_Λ that is a random variable which is true if and only if Λ represents a valid set of correspondences (all correspondences are correctly established). Ultimately, we want to find a solution in the space Σ of all possible solutions that maximizes the likelihood:

$$\Lambda' = \arg\max_{\Lambda \in \Sigma} P(O|\Psi_\Lambda = true) \tag{1}$$

In other words we can exploit previous equation (eq. 1) obtaining:

$$\Lambda' = \arg\max_{\Lambda \in \Sigma} \prod_{\lambda_{j,l}^{i,k} \in \Lambda} Similarity(O_{i,k}, O_{j,l}) \tag{2}$$

In this way by eq.2 the solution of the multi-camera re-identification problem is defined as the $\Lambda' \in \Sigma$ which maximizes an observation similarity measure. $Similarity()$ is a similarity measure between $O_{i,k}$ and $O_{j,l}$ in the testing data.

3. Appearances changing evaluation across cameras

Here, we want to model the changes in the appearances of an object from one camera to another. Basically, the idea is to learn the changes in the color objects when they move between the cameras, using a set of training data. Based on this set we can extract a function which is able to establish correspondence among object appearance coming from different FOVs. A possible way is proposed in (Porikli, 2003).In his approach, a brightness transfer function (BTF) f_{ij} is estimated for every pair of cameras C_i and C_j such that f_{ij} maps an observed brightness value in camera C_i to the corresponding value in camera C_j. Once that mapping function is known, the correspondence problem is reduced to match the transformed color histogram or the transformed appearance models. It should be noted that, a necessary condition for the existence of the one to one brightness mapping function among two different cameras, is that the object is planar and only has diffuse reflectance. This function, is not unique and it varies from frame to frame depending on different parameters including illumination, scene geometry, exposure time. focal length, aperture size and so on, of each camera. Therefore a single pre-computed mapping cannot normally be used to match objects for any frame sequences. It is possible to show how despite a large number of unknown parameters, all BTFs from a given camera to another one lie in a low dimensional subspace. Moreover, we describe a method to learn this subspace from training data and use this information to determine how, different observations in different cameras belong to the same object. Namely, given observations $O_{i,k}(a)$ and $O_{j,l}(a)$ from cameras C_i and C_j respectively, and given all possible brightness transfer functions (BTFs) from camera C_i to camera C_j we want to compute the probability that the observations $O_{i,k}(a)$ and $O_{j,l}(a)$ belong to the same object.

3.1 The Brightness transfer function space

Let $R_i(p,t)$ be the scene reflectance at a world point $p(x,y,z)$ of an object that is lighted by white light, when it is viewed from camera C_i at time instant t. Assuming that the objects do not have any specular reflectance, we can write $R_i(p,t)$ as a product of a term related to the material $M_i(p,t) = M(p)$ (i.e. albedo) and illumination/camera interaction, geometry and object shape related terms, $S_i(p,t)$, so we have:

$$R_i(p,t) = M(p)S_i(p,t) \tag{3}$$

The above model is used for the description of the Bidirectional Reflectance Distribution Function (BRDF), such as, the Lambertian model and the generalized Lambertian model (Oren & Nayar, 1995) (See Table 1). With the assumption of planarity we have, $S_i(p,t) = S_i(q,t) = G_i(t)$, for all points p and q of a given object. So, we can write eq(3) as

$$R_i(p,t) = M(p)S_i(p) \tag{4}$$

The image irradiance $I_i(p,t)$ is, of course, proportional to the scene radiance $R_i(p,t)$ (Horn, 1986) and we can obtain this:

$$I_i(p,t) = R_i(p,t)Y_i(t) = M(p)S_i(p)Y_i(t) \tag{5}$$

where

$$Y_i(t) = \frac{\pi}{4}\left(\frac{d_i(t)}{h_i(t)}\right)^2 \cos^4 \alpha_i(p,t) \tag{6}$$

is function of some camera parameters at time t. Here we list these intrinsic parameters: $h_i(t)$ is the focal length of the lens; $d_i(t)$ is a lens diameter (aperture); $\alpha_i(p,t)$ is the angle that the light ray from point p makes with the optical axis. However, the sensitivity reduction due to the term $\cos^4 \alpha_i(p,t)$ over an object is consider negligible (Horn, 1986) and can be replaced with a constant c.

Let us now consider $X_i(t)$ which is the time of exposure, and r_i which is the radiometric response function of the camera C_i, then the brightness point measure $B_i(p,t)$, is related to the image irradiance as follow:

$$B_i(p,t) = r_i(I_i(p,t)X_i(t)) = r_i(M(p)S_i(t)Y_i(t)X_i(t)) \tag{7}$$

In other words the image brightness $B_i(p,t)$ of a world point p at time instant t, is a nonlinear function of the product of its materials $M(p)$, its geometric properties $S_i(t)$ and camera parameters, $Y_i(t)$ and $X_i(t)$. Now let consider two cameras, C_i and C_j and let assume that a world point p is observed by both camera C_i and C_j at time instants t_i and t_j, respectively. A material properties M of world point does not change over the time so we have:

$$M(p) = \frac{r_i^{-1}(B_i(p,t_i))}{S_i(t_i)Y_i(t_i)X_i(t_i)} = \frac{r_j^{-1}(B_j(p,t_j))}{S_j(t_j)Y_j(t_j)X_j(t_j)} \tag{8}$$

Therefore the brightness transfer function from the image of C_i camera at time t_i to the camera C_j at time t_j, using equations 7 and 8, become:

$$B_j(p,t_j) = r_j\left(\frac{S_j(t_j)Y_j(t_j)X_j(t_j)}{S_i(t_i)Y_i(t_i)X_i(t_i)} r_i^{-1}(B_i(p,t_i))\right) = r_j(\omega(t_i,t_j)r_i^{-1}(B_i(p,t_i))) \tag{9}$$

where $\omega(t_i,t_j)$ is function of camera parameters and the illumination and scene geometry of cameras C_i and C_j at two different time instant t_i and t_j. So because eq. 9 is valid for all object point p visible by the two cameras, we can eliminate the p argument from the notation. Alike, since it is implicit that the BTF is different for any different pair of frames, we can remove the arguments t_i and t_j for make simpler the equations readability. Let denote with f_{ij} a BTF from camera C_i to camera C_j then starting from eq. 9 we have:

$$B_j = r_j(\omega r_i^{-1}(B_i)) = f_{ij}(B_i) \tag{10}$$

3.2 Inter-camera BTFs estimation

Now let us consider a pair of cameras C_i and C_j. The observations corresponding to the same object across this camera pair can be used to estimate an inter-camera BTF. A way to compute this BTF is to estimate the pixel to pixel correspondence among the object appearance in the two cameras (see eq. 10). However, self occlusion, change of scale and shape, and object

Model	**M**	**S**
Lambertian	ρ	$\frac{I}{\pi}\cos\theta_i$
Generalized Lambertian	ρ	$\frac{I}{\pi}\cos\theta_i\left[1-\frac{0.5\sigma^2}{\sigma^2+0.33}+\frac{0.15\sigma^2}{\sigma^2+0.09}\cos\left(\phi_i-\phi_r\right)\sin\alpha\tan\beta\right]$

Table 1. Commonly used BRDF models that satisfy eq. 3. The subscripts i and r denote the incident and the reflected directions measured with respect to normal surface. I is the source intensity, ρ is the albedo, σ is the surface roughness, $\alpha = max(\theta_i, \theta_r)$ and $\beta = min(\theta_i, \theta_r)$. Note that for generalized Lambertian model to satisfy eq.3, we must assume that surface roughness σ is constant over the plane.

deformation, make finding correspondences among different camera pixels very hard. In order to mitigate these problems, we use normalized histograms of object brightness values for the BTF estimation. The histograms are relatively robust to the changes of the object pose (Swain & Ballard, 1990). Assuming that the percentage of image points on the observed object $O_{i,k}(a)$ with brightness less than or equivalent to B_i is equal to the percentage of image points in the observation $O_{j,l}(a)$ with brightness less than or equivalent to B_j. It should be noted that a similar strategy was adopted in another work to obtain a BTF between images acquired by the same camera in the same FOV but in different illumination conditions (Grossberg & Nayar, 2003). Let be H_i and H_j the normalized cumulative histograms of object oservations I_i and I_j respectively, then

$$H_i(B_i) = H_j(B_j) = H_j(f_{ij}(B_i)) \tag{11}$$

Consequently we obtain,

$$f_{ij}(B_i) = H_j^{-1}(H_i(B_i)) \tag{12}$$

where with H^{-1} we indicate the inverted cumulative histogram. As discussed in the previous subsection, the BTF between two cameras changes instant by instant due to illumination conditions, camera parameters and so on. We apply eq. 12 to estimate the brightness transfer function f_{ij} for every pair of the observations contained in the training set. Also we denote with F_{ij} the collection of all the brightness functions obtained in the described way: $F_{ij} = \{f_{ij}^n\}, n \in \{1, \ldots, N\}$. Note that the discussion has been done dealing with only the brightness values of the images and estimating the brightness transfer functions. To deal with color images we have to compute each channel separately. It should be noted also that the knowledge of any camera parameters and response function for the calculation of these transfer function is not assumed.

3.3 Object color similarity estimation across camera using BTF

It is natural that the observed color of an object can vary widely across multiple non-overlapping camera due to change in scene illumination or of some of the different camera parameters like focal length, CCD gain, and so on. The training phase provides us the set of color transfer functions among the cameras, which models how colors of an object change across cameras. If the mapping function between the colors of two observations is correctly learned, in the test phase it is likely to find the observations generated by the same object. In particular for two observations $O_{i,k}(a)$ and $O_{j,l}(a)$ with color transfer functions f_{ij}^R, f_{ij}^G and f_{ij}^B, we define the probability of the observations belonging to the same object as:

$$P_{ij}(O_{i,k}(a), O_{j,l}(a) | \lambda_{j,l}^{i,k}) = \prod_{ch \in \{R,G,B\}} \gamma e^{-\gamma d(f_{ij}^{ch}(O_{i,k}^{ch}), O_{j,l}^{ch})} \tag{13}$$

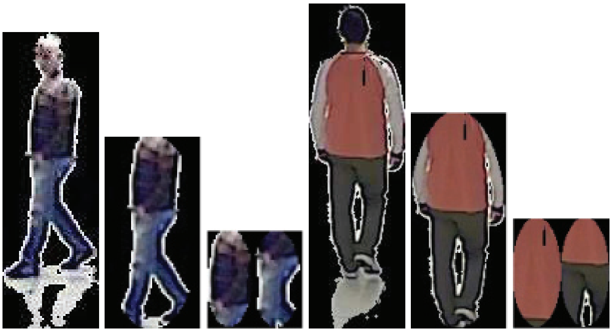
where γ is an arbitrary constant and d is a distance between an object appearance in C_j and the transformed one in C_i. The *ch* superscript denote the color channel for which appearance model and brightness transform were calculated.

4. Object appearance modelling

The proposed tracking algorithm models the object appearance using color histogram statistics. The task of finding the same object from the foreground region in current frame can be formulated as follows: the color histogram feature is assumed to have a density function and a candidate region also had a color histogram feature distributed by a certain density. The problem is to find a candidate region which is associated to a density most similar to the target one. A Bhattacharya coefficient measure is used as a similarity metric among the distributions.

4.1 Color histogram extraction

We have implemented and tested different methods to extract the color histogram from the foreground patches in order to remove noise and possible shadow from each object patch. We used various elliptic masks to reach this aim. The ellipse parameters (major and minor axis) are inferred using the patch dimensions and the distance of the object from the camera. Basing on the person position in each FOV, we have assessed, by a mapping function, his body measure in the foreground patches. Our intention is to build the elliptic masks in order to capture more useful information. We want to discard any possible part of the patch that could confuse the histogram distribution. The ellipses are drawn to cover most of the body cutting the head of the person and his eventual shadow (see 2(b), 3(b)). We have compared different combinations of these masks (see pictures 2(b), 2(c), 2(e), 2(f), 3(b), 3(c), 3(e), 3(f)) in order to estimate the potentialities of their performances.



(a) (b) (c) (d) (e) (f)

Fig. 2. Six images of two persons in the camera C1. a) Foreground patch extracted of the first person; b) Elliptic mask of the first person; c) Double Elliptic masks of the first person; d) Foreground patch extracted of the second person; e) Elliptic mask of the second person; f) Double Elliptic masks of the second person.

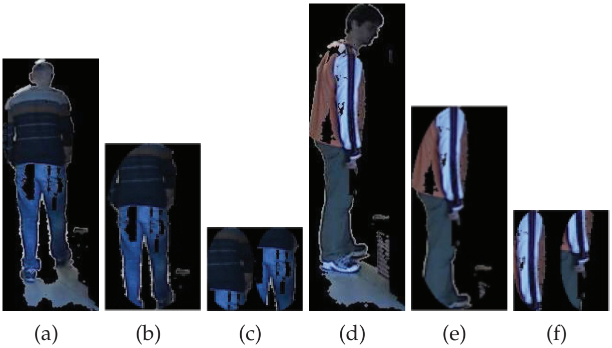


Fig. 3. Six images of two persons in the camera C2. a) Foreground patch extracted of the first person; b) Elliptic mask of the first person; c) Double Elliptic masks of the first person; d) Foreground patch extracted of the second person; e) Elliptic mask of the second person; f) Double Elliptic masks of the second person.

4.2 Positional histograms

As it should be noted, conventional color descriptors fail in the presence of clothes with the same main colors, but distributed in different way on the whole clothes. So we need to detect a features set able to maintain a level of relationship between global distribution and the displacement of colors on the silhouette. In the presence of well differentiated clothes, conventional histograms perform well, as well as other more refined features, like correlograms, even if this last one is onerous in terms of computational load. Our goal is to detect a feature set able to: perform in a acceptable way (compared with histograms) in presence of easily distinguishable uniforms; outperform histograms in the presence of hardly distinguishable uniforms; maintain a low level of computational load, that allows us to integrate this module in a higher level real team events detection system.

For these reasons we have chosen to work with a modified version of classic histograms, called Positional Histograms. These feature descriptors maintain basic characteristics of histograms (fast evaluation, scale invariance, rotation invariance, and so on); in addition, they introduce a dependance from the position of each point in the image: the global image is partitioned according to a geometrical relationship; the histograms are then evaluated for each region, and concatenated to obtain the final region descriptor.

Formally, the image I is partitioned in n subregions R_i, that satisfy the rules:

$$\bigcup_{i=1}^{n} R_i = I \tag{14}$$

$$R_i \cap R_j = \emptyset \quad \forall i \neq j \tag{15}$$

The first equation guarantees that each point of the image contributes to the final feature set construction, while the second one guarantees that each point gives its contribution just to one partition of histogram. In this way the final feature set contains exactly the same main information, as conventional histograms, but arranged in a different way, maintaining a level of information about the spatial distribution of points in the image.

The geometric rule for the partition should be fixed according to the nature of the problem to be solved. Our experience, and also experimental results we obtained, suggests us to use two main geometrical partitions: the angular sectors and the circular rings, and their fusion version (circular sectors). Polar coordinates allow to easily define the partitions. Each region R_i is composed by points (x,y) that satisfy:

$$R_i = \{(x,y) \| x = r\cos\theta,\ y = r\sin\theta \\ r^i_{MIN} < r < r^i_{MAX},\ \theta^i_{MIN} < \theta < \theta^i_{MAX}\} \quad (16)$$

With this notations, we can now explore details of each partition used in this paper. The starting point of each partition is the center of the image, where reasonably is concentrated the main informative content (a good object detector/tracker is able to maintain the subject in the center of the image).

4.2.1 Angular sectors

In this case each partition is obtained by varying the angle in a given range, according to the desired details level, while the radius ranges in all available values. So, considering D as the main diagonal of the image, and n the number of desired sectors, we have:

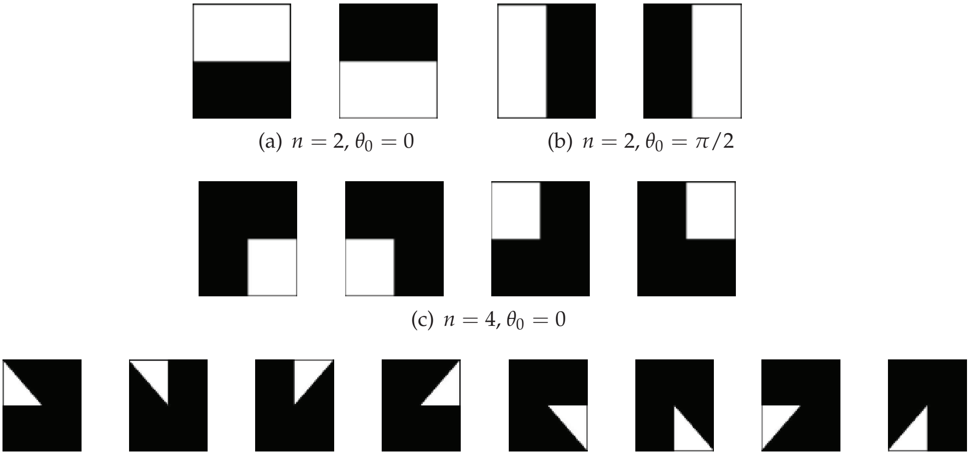
$$r^i_{MIN} = r_{MIN} = 0 \quad (17a)$$

$$r^i_{MAX} = r_{MAX} = D/2 \quad (17b)$$

$$\theta^i_{MIN} = \theta_0 + \frac{2\pi}{n}(i-1) \quad (17c)$$

$$\theta^i_{MAX} = \theta_0 + \frac{2\pi}{n} * i \quad (17d)$$

$$i = 1..n \quad (17e)$$



(a) $n = 2, \theta_0 = 0$

(b) $n = 2, \theta_0 = \pi/2$

(c) $n = 4, \theta_0 = 0$

(d) $n = 8, \theta_0 = 0$

Fig. 4. Plot of some Angular Sectors.

In figure 4 we have plotted some examples of masks for the regions creation in presence of Angular Sectors partitions. In the first rows we have plotted masks for $n = 2$, $\theta_0 = 0$, and $n = 2$, $\theta_0 = \pi/2$. Similarly, in the following rows we propose partitions for $n = 4$ and $\theta_0 = 0$, and $n = 8$ and $\theta_0 = 0$.

4.2.2 Circular rings

Each partition is obtained by varying the radius in a given range, according to the desired details level, while the angle varies in order to cover all possible values between 0 and 2π. So, considering D as the main diagonal of the image, and n the number of desired sectors, we have:

$$r^i_{MIN} = \frac{D * (i-1)}{2n} \tag{18a}$$

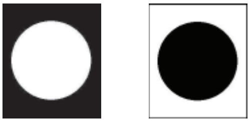
$$r^i_{MAX} = \frac{D * i}{2n} \tag{18b}$$

$$\theta^i_{MIN} = \theta_{MIN} = 0 \tag{18c}$$

$$\theta^i_{MAX} = \theta_{MAX} = 2\pi \tag{18d}$$

$$i = 1..n \tag{18e}$$

In figure 5 the masks in presence of Circular Rings partitions with $n = 2$ are plotted.



(a) $n = 2$

Fig. 5. Plot of some Circular Rings.

4.2.3 Circular sectors

The previously exposed partition rules can be combined (overlapped) in order to obtain another set of features that satisfies the conditions of equations 14 and 15. Now radius and angle various simultaneously tracing circular sectors across the image. So it is necessary to define two levels of partitions: the number n_S of desired angular sectors (that influences the range of the angle θ) and the number n_R of desired circular rings (that influences the range of the radius).

$$r^i_{MIN} = \frac{D * (i-1)}{2n} \tag{19a}$$

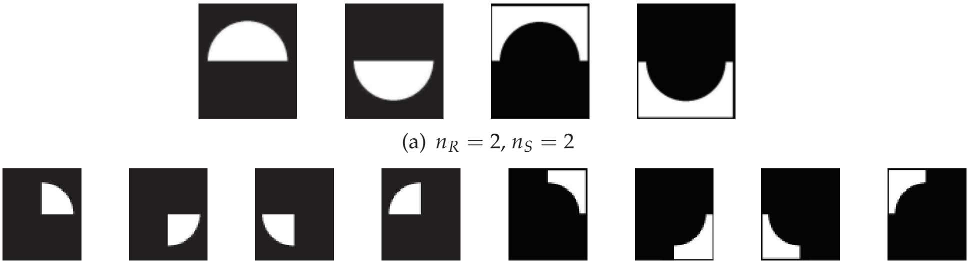
$$r^i_{MAX} = \frac{D * i}{2n} \tag{19b}$$

$$\theta^j_{MIN} = \theta_0 + \frac{2\pi}{n}(j-1) \tag{19c}$$

$$\theta^{j}_{MAX} = \theta_0 + \frac{2\pi}{n} * j \tag{19d}$$

$$i = 1..n_R \qquad j = 1..n_S \tag{19e}$$

In figure 6 some examples of masks in presence of Circular Sectors partitions are plotted: first row refers to rings obtained for $n_R = 2$, $n_S = 2$ while the second one refers to rings for $n_S = 4$, $n_R = 2$.



(a) $n_R = 2, n_S = 2$

(b) $n_S = 4, n_R = 2$

Fig. 6. Plot of some Circular Sectors.

5. Discussion on tracking in multiple cameras with disjoint views

Starting from two cameras C_1 and C_2 that are localized in different places in our building, we explore some methodology to establish correspondence among these FOVs. As you can see in Figure 1, the cameras' field of views cover different non-overlapping areas. People observed in camera C_2 can take a path across camera C_1 turning right or also turning left into *Lab1* without entering the C_1 field of view. Likewise, people coming from the *Lab1* are observed in camera C_1 without passing through camera C_2. As we described in the previous section the task of the multi-camera tracking algorithm is to establish correspondence across cameras finding which observations, coming from different FOVs, belong to the same object.Because of the cameras' positions, it is not always possible to use space-time constraints among the exits and entrances areas of the different cameras. Obviously people can take many paths across C_1 and C_2 producing different observations of the same objects in the two cameras. The multi-view tracking algorithm we want to investigate *relies just on the object appearances in the two cameras FOV.* It can be noted that the lack of entry/exit constraints make the association task more difficult. We should consider that color distribution of an object can be fairly different when it moves in a single camera FOV. For this reason, matching appearances between different cameras is still more difficult. It is necessary to find the transformation that maps the object's appearance in one camera with its appearance in the other one. In this chapter we consider a training phase in which know objects pass trough both cameras and their appearances are used to estimate a Brightness Transfer Function (BTF). During this phase we tested two different BTFs, ie. the mean BTF (MBTF) and the cumulative BTF (CBTF).In the succeeding phase we test implemented transformation in order to evaluate the object matches that produced the lowest value of the Bhattacharya distance for the appearance of the considered person in one camera compared with the appearances of all the possible persons who had walked through the second camera.

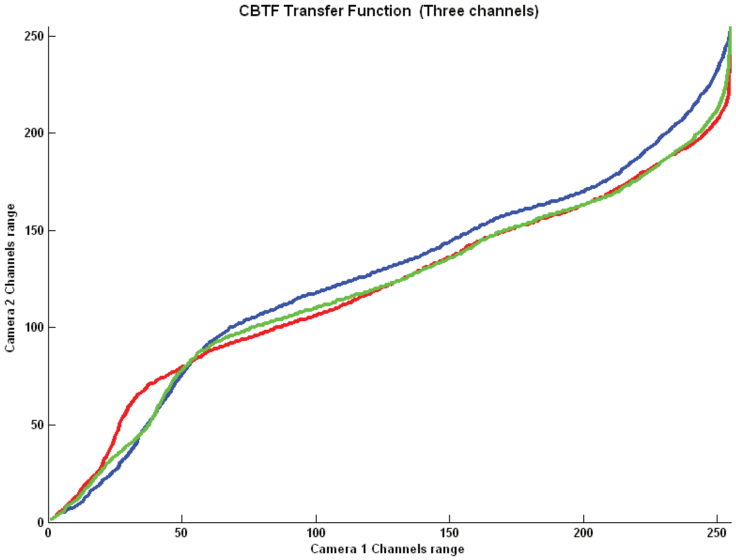


Fig. 7. The transfer functions for the R,G and B color channels from Camera 1 to Camera 2 obtained using Cumulative Brightness Transfer Transform on the objects appearances training set. Note that mostly lower color values from Camera 2 are being mapped to higher color values in Camera 1, indicating that the same object is appearing much brighter in Camera 1 as compared to Camera 2.

5.1 Establish correspondence across multiple cameras

The correspondence problem occurs when an object enters the camera FOV. We need to determine if the object is already being tracked by another camera or it is a new object in the scene. As described in previous sections there are many approaches which are able to estimate the BTFs among different cameras. We compare a mean BTF (see section 21) approach with the cumulative BTF proposed in (Prosser et al., 2008). In figure 2(a) some images of the the tracks of two people in the camera C_1 FOV are showed, while in figure 3(a) the same people are observed in in the camera C_2 FOV. We computed the three channels RGB histograms for each image in the C_1 tracks. We did the same, also, for the C_2 tracks. The histogram were generated using all the 256 bins for each color channel. We want to estimate a BTF f_{12} between the cameras C_1 and C_2 such that, for each couple of objects observations $O_{1,k}(a)$ and $O_{2,l}(a)$ given the brightness value $B_{O_{1,k}}(v)$ and $B_{O_{2,l}}(v)$ we have $B_{O_{2,l}}(v) = f_{12}(B_{O_{1,k}}(v))$ where $v = 0, \ldots, 255$ represents the number of bins, $k = 1, \ldots, M$ represents the number of object appearance in the camera C_1, $l = 1, \ldots, N$, the number of object appearance in the C_2. In order to evaluate the BTF f_{12} we collected a total of $N + M$ histograms obtained on the N object appearances tracked in the camera C_1 and on the same object appearances tracked in the camera C_2. Let be $H_{O_{1,k}}$ and $H_{O_{2,l}}$ the object appearance histograms obtained in the cameras C_1 and C_2 respectively. Now, for each possible different cameras object appearance couple $(O_{1,k}, O_{2,l})$ we want to compute the brightness transfer function using the inverted cumulative histogram (see equation 12) we obtain:

$$f_{O_{1,k}O_{2,l}}(B_{O_{1,k}}) = H^{-1}_{O_{2,l}}(H_{O_{1,k}}(B_{O_{1,k}})) \tag{20}$$

and finally the mean BTF (referred in the following sections as MBTF) $\overline{f}_{12}$

$$\overline{f}_{12} = \sum_{k=1}^{M}\sum_{l=1}^{N} f_{O_{1,k}O_{2,l}} \tag{21}$$

We estimated also a cumulative BTF (CBTF) as described in (Prosser et al., 2008). As it is known, the CBTF generation involves an amalgamation of the training set before computing any BTFs. An accumulation of the brightness values is computed an all the training images of the camera C_1 obtaining a cumulative histogram $\hat{H}_1$. The same is done for all the corresponding training images of the camera C_2 obtaining $\hat{H}_2$. The CBTF $\hat{f}_{12}$ using eq. 12 is

$$\hat{f}_{12}(O_{1,k}) = \hat{H}_2^{-1}(\hat{H}_1(O_{1,k})) \tag{22}$$

also in this case evaluated by using the inverted cumulative histogram. In figure 7 the CBTF obtained on the training set is plotted. It should be noted that mostly lower color values from Camera C_2 are being mapped to higher color values in Camera C_1, indicating that the same object is appearing much brighter in Camera C_1 as compared to Camera C_2.

6. Multi-camera tracking using different BTFs

In order to solve the multi-camera people re-identification problem we have to choose among a set of possible correspondence hypotheses the one that produces the best match. Since, our cameras' configuration allows people to enter into one camera field of view without passing across the other camera's FOV, we consider also the problem of finding a proper method to discard false matches. The common method is to match people appearances by estimating the similarity among color histograms. Once both $O_{i,k}$ and $O_{j,l}$ are converted into the same FOV by eq. 21 and eq. 22 we can compare them directly using the well known Bhattacharya distance measure $D_B()$ and thus the similarity measure from eq. 2 can be defined as follows:

$$Similarity(O_{i,k}, O_{j,l}) = 1 - D_B(O_{i,k}, O_{j,l}) \tag{23}$$

If we denote with H_i and H_j the normalized histograms of the observations $O_{i,k}$ and $O_{j,l}$ the Bhattacharya distance $D_B()$ (Comaniciu et. al, 2003) between two histograms is given as

$$D_B(H_i, H_j) = \sqrt{1 - \sum_{v=1}^{m} \sqrt{H_i(v)H_j(v)}} \tag{24}$$

where m is the total number of histogram bins. The Bhattacharya coefficient ranges between zero and one and is a metric.

Note that in order to compare two color objects, we must apply this process to each of the three RGB channels. Thus the overall similarity measure becomes the mean of the similarity values obtained in all three channels.

Let us, now, consider $\left\{ H_{O_{1,k_1}}, H_{O_{1,k_2}}, \ldots, H_{O_{1,k_{N_k}}} \right\}$ the N_k object appearances histograms of the $k-th$ person in the camera C_1. Suppose that we have P, i.e. $k \in \{1, \ldots, P\}$, people moving in the camera C_1. When a new observation is taken in the camera C_2 we have to decide either if it could be associated with one among the P individuals moving in the camera C_1 or if it is a new person entering the scene. For each person k, in the camera C_1 with $k \in \{1, \ldots, P\}$, we evaluated the mean color histograms among the N_k appearance observations of the k^{th} person obtaining $\overline{H}_{1,k}$. Anyway the mean histograms cannot be compared with those obtained by the camera C_2 unless the transformation with the BTFs are applied. By using the BTFs described in section 5 we projected the P mean histograms in the new space as follows:

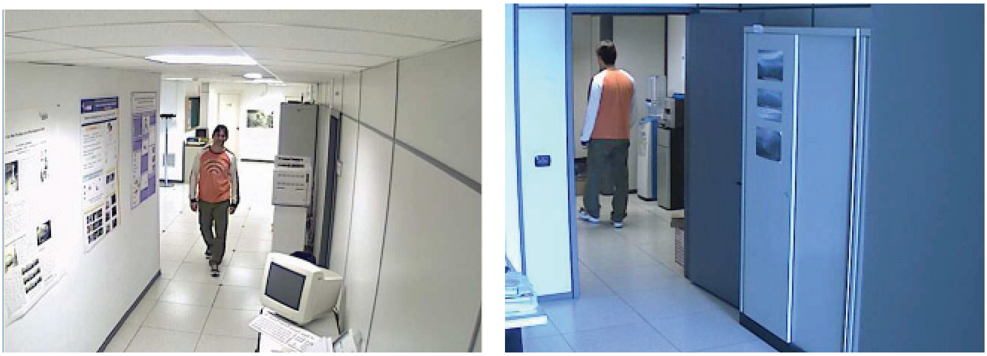
$$\breve{H}_k = \overline{f}_{12}(\overline{H}_{1,k}) \tag{25}$$

where $\breve{H}_k$ represents the new histogram obtained by using the mean BTF described in eq.21. We, also, have using eq. 22:

$$\tilde{H}_k = \hat{f}_{12}(\overline{H}_{1,k}) \tag{26}$$

which is the histogram transformation using CBTF. Let be $H_{O_{2,l_1}}$ the first observation histogram in the camera C_2. We evaluated the similarity between couple of histogram by using eq. 27. The association is done with the k^{th} person who produces the maximum similarity measure, i.e.

$$\arg\max_{k} Similarity(H_{O_{2,l_1}}, \tilde{H}_k) \tag{27}$$



(a) Camera 2 Field of view

(b) Camera 1 Field of view

Fig. 8. Frames from both camera views. The same person walks from camera 1 to camera 2

7. Obtained results

Different experiments were carried out to test the multi-camera tracking algorithm. The scenario was composed of two cameras located in two different points of our office (see map on figure 1). We used two wireless Axis IP camera with 640x480 VGA color jpg resolution with an acquisition frame rate of 10 fps. The camera network topology is shown in figure 1, while two images acquired by the two cameras are shown in figures 8(a) and 8(b). Note that the illumination conditions and color quality vary greatly between these views. We divided the experiments into two different parts: in the first part we investigate different kind of method to extract the color histogram from each foreground patch. In the second part we evaluated different approaches to establish correspondence across disjointed views. In both parts we used the same data set. The data-set consisted of synchronized *mjpeg* videos acquired simultaneously by two different cameras containing eight persons. The patches come from a single camera object detection method described in (Mazzeo et. al, 2008).The data set was obtained by extracting people from whole Field of View of each camera. Note that we did not consider any geometrical constraint on the exiting and entering areas of people moving in the observed scenario. We carried out different experiments using different sets of samples as follows:

- First experiment ten people giving 1456 appearance sample (coming from the same FOV) are used as testing data in order to evaluate the performance of different color histogram extraction methods (See section 4).
- In second experiment we have a training and a testing data set: seven individuals giving 1023 appearance samples in both views were used in the training step, while eight individuals with 1247 appearance samples in both views were used in the testing step (Note that in this case we added one person in the testing phase).

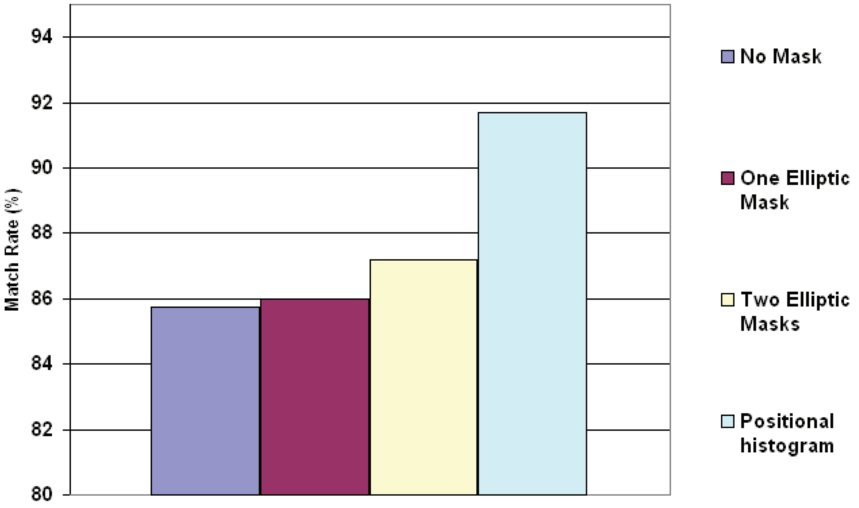


Fig. 9. A matching success comparison in the same FOV (Intra-camera) using different color histogram extraction method.

In the figure 9 are presented the results relative to the intra-camera histogram color tracking. As described in section 4 we evaluated four different approaches to estimate the color histogram from extracted people patches. The similarity between color histogram features belonging to different foreground patches, was measured by means of eq. 27 based on the Bhattacharyya distance (eq. 24). The highest value of eq. 27, among the designated patch and all the possible candidates (seven different people in the same FOV), determines the tracking association. As it is shown in figure 9 it is possible to notice how the positional histogram approach gives better result in term of match rate. By using positional histogram, in fact, it is possible to preserve the color histogram spatial information. In particular we used partition masks based on circular sector with $n_r = 2$ and $n_s = 4$ (see figure 6a). In this context this kind of masks gave best performance in terms of correct matching rate. In this way, in fact, color histograms of different body parts of each patch are compared with the correspondent parts of the another persons (see subsection 4.2). Results confirm that this color histogram extraction approach discriminates better among the different possible candidates.

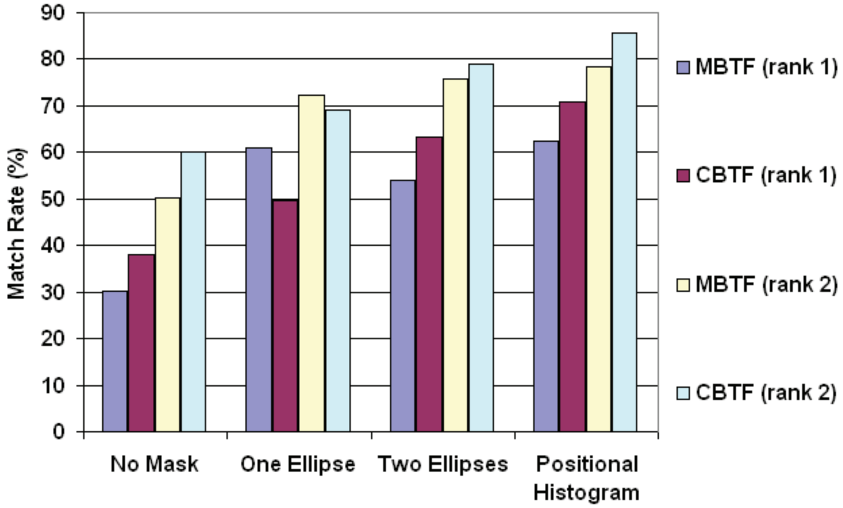


Fig. 10. A matching success comparison in establishing correspondence across cameras using varying color histogram extraction method and different Brightness Transfer Function Estimation.

In figure 10, the results relative to the tracking across different non overlapping cameras, are shown. The experiments consist of a training phase and a testing phase. During the training phase we supposed that the correspondence between the same object in the different cameras'FOV was known and this information was used to estimate the Mean Brightness Transfer Function (MBTF) and the Cumulative Brightness Transfer Function (CBTF). In the testing phase the correspondences between cameras were computed using eq. 27 based on Bhattacharyya distance, and the correct match was associated with the highest value of eq. 27 among all the possible couple of candidates. As it can be noticed, in the testing phase we consider seven people that were present in both views. As described in section 5 we tested two approaches to estimate BTFs among the two cameras: the MBTF and the CBTF. For each people patch we converted his RGB histogram (extracted in the different way described in section 4) into the target region color space (i.e. from camera $C1$ to camera $C2$). They were compared against all individuals observed in this region. In particular we estimated the mean color histogram for the same person in each view and we compared each converted histogram against it. In figure 10 we report both the $rank1$ and $rank2$ results indicating the correct match presence as the highest and the second highest similarity score respectively. As figure shows both transfer functions (MBTF and CBTF) gave quite similar behaviors in term of match rates but it should be noted that the CBTF outperform MBTF in the $rank1$ and $rank2$ results (we matched the histogram also against the mean histogram of the people contained in the training set). Note that, only in the case of the one elliptic mask approach MBTF gives better results than CBTF (the difference is very narrow). However the overall performances confirmed that

CBTF retained more color information than MBTF and produced a more accurate mapping function. The same figure 10 shows a comparison among different color histogram extraction approaches, the match rates confirmed that positional histogram gave the best results. And this is what we expect for the reason explained in the first part of this section. Finally, figure 11 shows the mean and standard deviation of the different color histogram extraction method applied on the patch set used in both part of the experiment. Even these values demonstrate that positional histogram gives the greatest mean score with the lowest standard deviation. It means that the positional histogram has the capability to catch more useful information from the people patches in order to establish correspondences among different same people appearances. Definitely positional histogram method maps the data among the people classes better than the others.

8. Final considerations and future work

This book chapter presented a dissertation on the feasibility of multi-camera tracking algorithms based on the appearance similarity. We considered only two non overlapping cameras located inside an office building. We investigated the reliability of appearance similarity methods to track people in the same FOV and among different FOVs. Then we evaluated different color histogram extraction methods, with different elliptic masks and positional histogram. The obtained results demonstrated that using positional histograms improved overall results in terms of matching rate. Also, we compared two different kinds of Brightness Transfer Function, i.e. the MBTF and the CBTF. The experiments demonstrated quite similar behaviors of the two transferring functions when the simple association problem has to be solved.

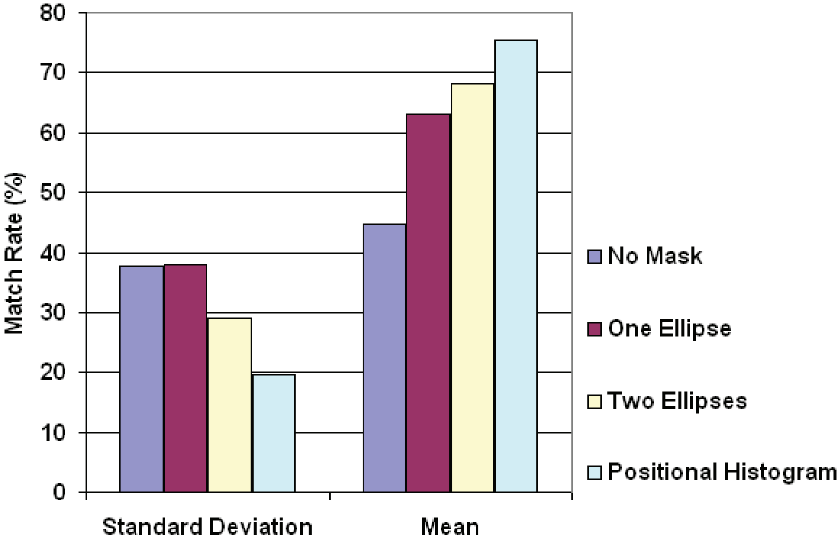


Fig. 11. Mean and standard deviation of the matching rate using different color extraction method to establish the correspondence between the two cameras.

Future work will be addressed on the study of new methodologies for more reliable appearance modelling. As we described in this chapter it is known that people's appearances can be similar in some parts of the body and differ in other parts. So we are thinking to apply some methodologies based on the extraction patch histogram graph. Then, we use different weights in the correspondence matches in order to consider different body parts reliability and highlight only the significant differences among the people appearances.

9. References

O. Javed, K.Safique,Z,Rasheed,M. Shah Modeling inter camera space-time and apperacnce relationships for tracking across non-overlapping views, *Computer Vision and Image Understanding*, Vol.109, 146-162, 2008

W. Du, J. Piater, Data Fusion by Belief Propagation for Multi-Camera Tracking, *The 9th International Conference on Information Fusion*, 2006.

W. Du, J. Piater, Multi-Camera People Tracking by Collaborative Particle Filters and Principal Axis-Based Integration *Asian Conference on Computer Vision*, Springer LNCS 4843 pp. 365-374, 2007.

A. Chilgunde, P. Kumar, S. Ranganath, H. WeiMin, Multi-Camera Target Tracking in Blind Regions of Cameras with Non-overlapping Fields of View, *British Machine Vision Conference BMVC*, Kingston, 7th-9th Sept, 2004.

Q. Cai, J.K. Aggarwal, Tracking human motion in structured environments using a distributed camera system, IEEE Trans. Pattern Anal. Mach. Intell. 2 (11)1241-1247, 1999.

F. Porikli, Inter-camera color calibration using cross-correlation model function, *IEEE Int. Conf. on Image Processing*, 2003.

Y. Shan, H.S. Sahwney, R. Kumar, Unsupervised learning of discriminative edge measures for vehicle matching between nonoverlapping cameras, *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

B. Prosser, S. Gong, T. Xiang, Multi-camera Matching using Bi-Directional Cumulative Brightness Transfer Functions, *British Machine Vision Conference*, 2008.

C. Madden, E.D. Cheng, M. Piccardi Tracking people across disjoint camera views by an illumination-tolerant appearance representation, *Machine Vision and Application*, 18, pp 233-247, 2007.

K. Jeong, C. Jaynes Object matching in disjoint cameras using a color transfer approach, *Machine Vision and Application*, 19 , pp 443-455, 2008.

M. Oren, S.K. Nayar Generalization of the Lambertian model and implications for machine vision, *International Journal of Computer Vision*, 14(3) , pp 227-251, 1995.

Horn, B.K.P. (1986). *Robot Vision*,MIT Press Cambridge.

Swain, M.J.; Ballard, D.H. (1990). Indexing via color histograms, in *IEEE International Conference on Computer Vision*.

Grossberg, M.D.; Nayar, S.K. (2003). Determing the camera response from images: what is knowable?, in *IEEE Transaction in Pattern Analysis and Machine Intelligence*, 25(11), pp. 1455-1467.

Comaniciu, D.; Ramesh, V.; Meer P. (2003). Kernel-based object tracking, in *IEEE Transaction in Pattern Analysis and Machine Intelligence*, 25, pp. 564-575.

Mazzeo, P.L.; Spagnolo, P.; Leo, M.; D'Orazio T (2008). Visual Player Detection and Tracking in Soccer Matches, in *IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance AVSS*, pp. 326-333.

12

Object Tracking for Calibration of Distributed Sensors in Intelligent Space

Takeshi Sasaki and Hideki Hashimoto
The University of Tokyo
Japan

1. Introduction

In recent years, the research field on smart environments, which are spaces with multiple embedded and networked sensors and actuators, has been expanding (Cook & Das, 2004). The main purpose for the introduction of smart environments is to support humans in both physical and informative ways (Johanson et al., 2002), (Mynatt et al., 2004), (Mizoguchi et al., 1999). In smart environments, the distributed sensor nodes observe the spaces, extract useful information from the obtained data and the actuators including mobile robots provide various services to users. Moreover, robots in the space can get necessary information from the smart environment and operate without restrictions due to the capability of on-board sensors and computers. In fact, mobile robot control is easier in smart environments since the global positions can be directly measured by using distributed sensors and Simultaneous Localization And Mapping (SLAM) problem (Durrant-Whyte & Bailey, 2006), where the robot tries to simultaneously estimate its own location and build a map of the environment, can be completely avoided (Lee & Hashimoto, 2003).

However, one of the major problems in developing smart environments is calibration of the sensors. Calibration is needed for proper calculation from the local (sensor's) coordinate system to the world (smart environment's) coordinate system and it is usually done by using calibration objects that are objects with known positions, shapes and so on. For example, researches on camera calibration based on geometrical features including 3D points (Tsai, R. Y., 1987), 2D points (Sturm & Maybank, 1999), (Zhang, 2000), lines (Zhang, 2004), circles (Wu et al., 2004), and spheres (Agrawal & Davis, 2003) are being pursued actively. Several researchers extended such a single camera calibration algorithm to a multiple-camera calibration algorithm. An extension of a planar point pattern based calibration algorithm to the multi-camera systems with an arbitrary number of cameras is presented in (Ueshiba & Tomita, 2003). The algorithm is based on factorization of homography matrices between the model and image planes, which is expressed as a composition of a camera projection matrix and a plane parameter matrix. If a calibration object is put in three or more places, the relative positions and orientations between cameras as well as the intrinsic camera parameters are determined. Another work also utilized a known 2D calibration object for stereo camera calibration (Malm & Heyden, 2001). The technique uses both stereo camera constraints and single camera constraints, and therefore noise-robust calibration is realized.

In the case of smart environments, it takes a great deal of time and efforts to calibrate the sensors since multiple sensors are distributed in the space. Although the researches mentioned above aim to lighten this enormous work, our research purpose is to automate the calibration process with satisfactory accuracy. In order to solve this problem, mobile robots have been used for realizing automated calibration. Mobile robots can cover wide areas of the environment by moving from place to place so there is no need to place many landmarks in exactly known positions beforehand. Some researchers focus on node localization in wireless sensor networks using mobile robots (Shenoy & Tan, 2005), (Sreenath et al., 2006). In the methods, each mobile robot broadcasts its position information and if a wireless sensor node can get the information, the node is considered to be located adjacent to the robot. An effective path planning of mobile robots for wireless node localization is also addressed in (Koutsonikolas et al., 2006). Although the sensor nodes just receive the position information from the robots in these researches, the measurement of the sensors can also be used for calibration. In (Rekleitis & Dudek, 2005), which is closely related to our work shown in this chapter, camera sensor network is calibrated based on grid patterns attached to a mobile robot. We also introduce a mobile robot assisted calibration method and use the position information of the robot to calibrate distributed sensors including laser range finders and cameras. Hereby we can add a calibration function without major changes in the system because we can use existing moving object tracking and mobile robot localization functions of smart environments.

Next section gives a brief introduction of our smart environment which is called "Intelligent Space (iSpace)." Section 3 and 4 describe the automated calibration method based on the mobile robot tracking for distributed laser range finders and cameras, respectively. Mobile robot localization used in the calibration method is explained in section 5. Experimental results of the proposed method and comparison with the manual calibration are shown in section 6. Finally, a conclusion is given in section 7.

2. Intelligent space

"Intelligent Space (iSpace)" has been proposed by Hashimoto laboratory at The University of Tokyo (Lee & Hashimoto, 2002). Figure 1 shows the concept of iSpace. We call the sensor node devices distributed in the space DINDs (Distributed Intelligent Network Device). A DIND consists of three basic components: sensors, processors and communication devices. The processors deal with the sensed data and extract information about objects (type of object, three dimensional position, etc.), users (identification, posture, activity, etc.), and the environment (geometrical shape, temperature, emergency, etc.). The network of DINDs can realize the observation and recognition/understanding of the events in the whole space. Based on the extracted and fused information, actuators such as displays or projectors embedded in the space provide informative services to users. In iSpace, mobile robots are also used as actuators to provide physical services to the users and for them we use the name mobile agents.

Figure 2 shows a configuration of the iSpace implementation in our laboratory. CCD cameras, laser range finders, and a 3D ultrasound positioning system are used as sensors of DINDs. The laser range finders are placed close to the ground horizontally (about 20 cm above the floor). The advantage of the low position is that it is possible to scan also relatively short objects, assuring that all objects on the floor will enter the scan. The 3D ultrasound positioning system involves 112 ultrasonic receivers installed on the ceiling. This system can measure the three dimensional position of an ultrasonic transmitter (tag) to an accuracy of 20-80 millimeters using triangulation. Moreover, differential wheeled robots are used as mobile agents. For estimating the position and orientation of the robot, an ultrasonic transmitter is installed on the top of the mobile robot. The mobile robot is also equipped with a wireless network device to communicate with iSpace.

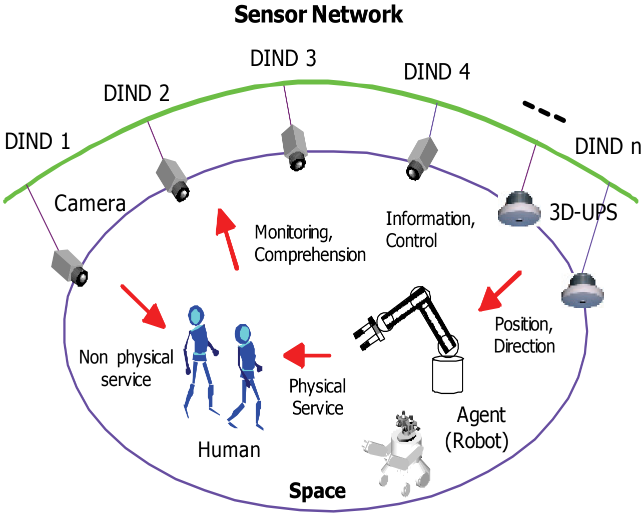


Fig. 1. Concept of Intelligent Space (iSpace)

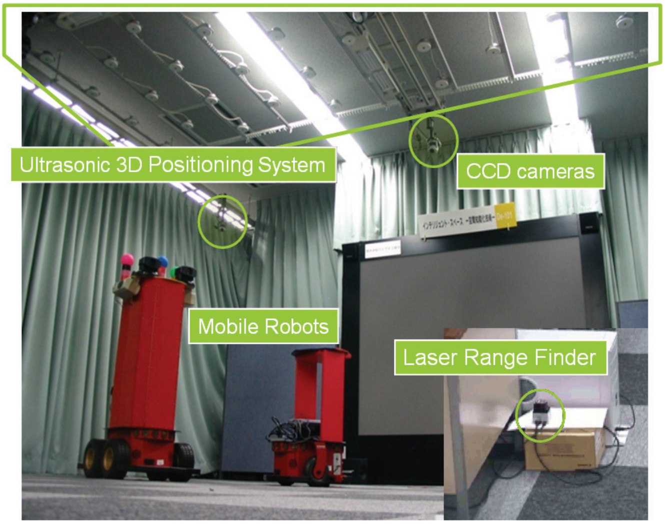


Fig. 2. Configuration of Intelligent Space – sensors and mobile robots

3. Automated calibration of distributed laser range finders

3.1 Overview of the method

Our goal is to find transformation parameters (translation vector ***T*** and rotation matrix ***R***) from the laser range finder coordinates to the world coordinates. Since the laser range finders are placed horizontally as noted in section 2, the calibration parameters are position and orientation of the laser range finder in 2D plane (T^i_{xg}, T^i_{yg}, θ^i_g) (i=1, 2, ..., N), where N and i denote the number of laser range finders in the environment and index of each laser range finder, respectively. As mentioned in section 1, we utilize mobile robots in iSpace to realize the automated calibration. Figure 3 shows the overview of the calibration method.

Let $^{W}O{-}^{W}x^{W}y$ be the coordinate system fixed to iSpace (world coordinate system) and $^{Li}O{-}^{Li}x^{Li}y$ be the coordinate system fixed to the i^{th} laser range finder (laser range finder i coordinate system). First, each DIND tracks the mobile agents and gets the position of the mobile robots in the local coordinate system (x^i_k, y^i_k). The DINDs also request the position server in iSpace where the postion information of objects is collected and stored to send the position of the robot in the world coordinate system (x^g_k, y^g_k). The calibration process is then performed based on the set of corresponding points {(x^g_k, y^g_k), (x^i_k, y^i_k)} (k = 1, 2, ..., n).

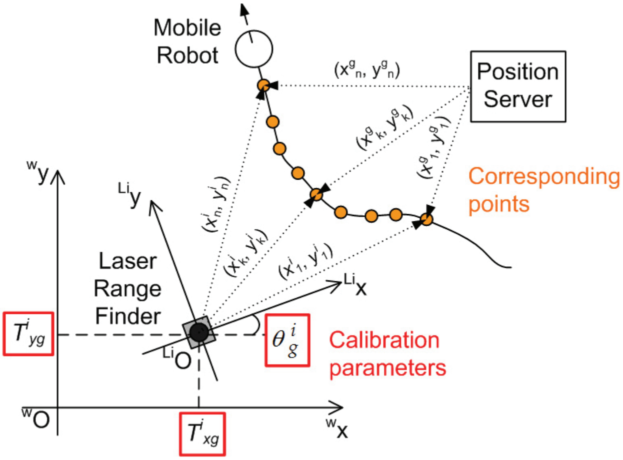


Fig. 3. Calibration of a laser range finder using a mobile robot

In the following subsections, the functions needed for the proposed calibration method are described. Tracking of moving objects and pose estimation from a set of corresponding points are explained in section 3.2 and 3.3, respectively.

3.2 Object tracking using a laser range finder

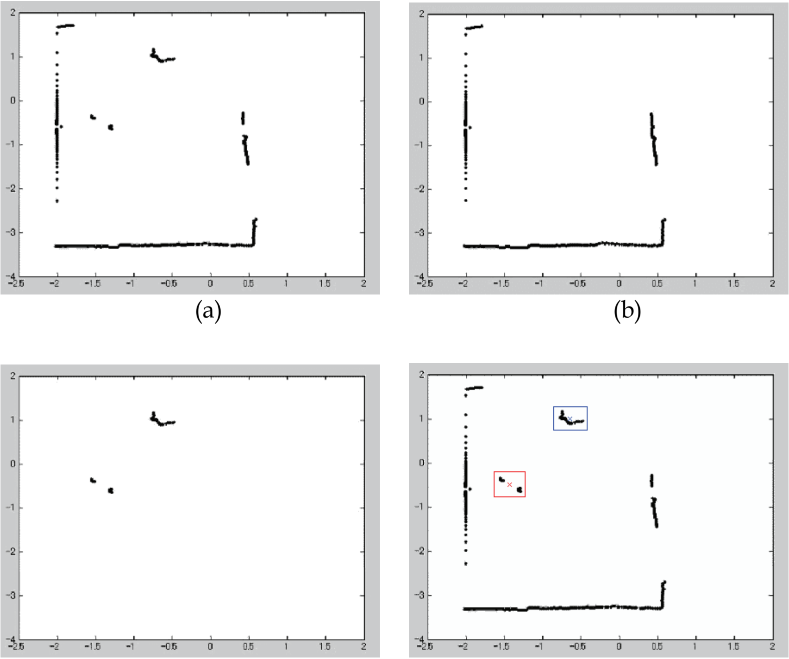
The tracking process consists of extraction of objects, estimation of the object centers and tracking using Kalman Filter.

Figure 4 shows the the object extraction. In the extraction process, background subtraction is first applied. The static parts of scan (background) are subtracted from the scan data in order for determining which parts of the scan are due to moving objects (foreground). The scan points in the foreground are then clustered based on the Euclidean distance between them using a nearest neighbor classifier. This divides the foreground to a number of clusters, each belonging to one of the tracked objects. Clusters with a small number of scan points are discarded as measurement noise. Data association is based on the Euclidean distance. The position of cluster centers are compared with the positions of currently tracked objects and each cluster is assigned to the closest object. The clusters that are far from all currently tracked objects are considered as new objects, and a new tracking process is started for them.

From the previous step the positions of cluster centers were obtained. But since the objects are scanned from one side, the center of the obtained cluster of points (x_{cl}, y_{cl}) in general does not coincide with the center of the tracked object (x_{obj}, y_{obj}). So, as shown in Fig. 5, the object center is assumed to be at a fixed distance d from the center of the cluster of scan points belonging to it, that is,

$$\begin{aligned} x_{obj} &= x_{cl} + d\cos(\alpha) \\ y_{obj} &= y_{cl} + d\sin(\alpha) \end{aligned} \tag{1}$$

where α is the angle of the line between the laser range finder and the center of the cluster, and d is a parameter depending on the radius of the object. In our experiments d was set to 6 centimeters for human (i.e. human's leg) and 15 centimeters for the mobile robot.



(a)

(b)

(b)

(d)

Fig. 4. Process of object extraction using laser range finder (a) raw scan data, (b) static part of the scan (background), (c) result of background subtraction (foreground), (d) result of clustering and centers of the clusters. The units of x and y are in meters.

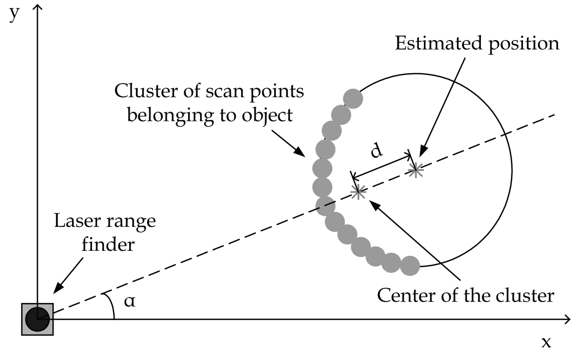


Fig. 5. Estimation of object center

Moreover, in order to distinguish between humans and mobile robots, the number of clusters belonging to an object is used since two clusters belonging to his/her legs are detected in the case of tracking a human. In our implementation we gradually determine the type of object by filtering the number of clusters. Only the positions of mobile robots are used for calibration purpose.

Finally, the Kalman filter is applied to track the objects.

The details of our laser range finder based tracking method and evaluation of the method are described in (Brscic & Hashimoto, 2006).

3.3 Calibration of laser range finders based on the corresponding points

In the calibration process, the position and orientation of the laser range finders in the world coordinate system (T^i_{xg}, T^i_{yg}, θ^i_g) is calculated. We solve the least square error problem denoted by the following equation:

$$\varepsilon^2 = \sum_{k=1}^{n}\left((x_k^g - \cos\theta_g^i x_k^i + \sin\theta_g^i y_k^i - T_{xg}^i)^2 + (y_k^g - \sin\theta_g^i x_k^i - \cos\theta_g^i y_k^i - T_{yg}^i)^2\right) \tag{2}$$

where the indices *g* and *i* represent the obtained global and local coordinates respectively. If more than one set of corresponding points are obtained, we can derive the following estimates from $\frac{\partial\varepsilon^2}{\partial T_{xg}^i} = 0$, $\frac{\partial\varepsilon^2}{\partial T_{yg}^i} = 0$, and $\frac{\partial\varepsilon^2}{\partial\theta_g^i} = 0$:

$$T_{xg}^i = \mu_x^g - \cos\theta_g^i \mu_x^i + \sin\theta_g^i \mu_y^i \tag{3}$$

$$T_{yg}^i = \mu_y^g - \sin\theta_g^i \mu_x^i - \cos\theta_g^i \mu_y^i \tag{4}$$

$$\theta_g^i = \operatorname{atan2}\left(\sum_{k=1}^{n}\frac{-\left(x_k^g y_k^i - y_k^g x_k^i\right)}{n} + \mu_x^g \mu_y^i - \mu_y^g \mu_x^i,\right.$$
$$\left.\sum_{k=1}^{n}\frac{\left(x_k^g x_k^i + y_k^g y_k^i\right)}{n} - \mu_x^g \mu_x^i - \mu_y^g \mu_y^i\right) \tag{5}$$

where atan2(·) denotes the four-quadrant inverse tangent function and μ's stand for mean values, for example:

$$\mu_x^g = \frac{1}{n}\sum_{k=1}^{n} x_k^g \tag{6}$$

The problem with least-squares estimation as given by equations (3)-(5) is sensitivity to outliers. Since robot tracking is done online it is possible that outliers, such as miscorrespondence between data can easily appear. In order to eliminate the effect of outliers instead of simple least squares we use the least median of squares (LMedS) based estimation. In the LMedS method, the estimation error is evaluated by not the sum of the square error but its median value.

The automated calibration process is summarized as follows:

1. Store corresponding points (x^g_k, y^g_k), (x^i_k, y^i_k) acquired by the robot tracking process
2. Sample 2 data randomly from the set of corresponding points
3. Calculate (T^i_{xg}, T^i_{yg}, θ^i_g) from the sampled data using equations (3)-(5)
4. Evaluate the estimation error by the median of the square error for all corresponding points
5. Repeat steps 2) – 4) enough times
6. Select (T^i_{xg}, T^i_{yg}, θ^i_g) which has minimum estimation error as the estimate

4. Automated calibration of distributed cameras

4.1 Overview of the method

The idea of the automated camera calibration is similar to the one explained in the previous section. Two measurements, the result of robot localization and the tracking result of the robot, are stored as corresponding points and the calibration is performed based on them. But, in this case, the position information obtained by a camera is the positions of the robots on the image plane. In addition, we need to calibrate both the intrinsic and the extrinsic camera parameters.

In the following subsections, tracking of mobile robots using a camera is explained in section 4.2 and the calibration method is mentioned in section 4.3.

4.2 Visual tracking of mobile robots

To detect moving objects in iSpace using CCD cameras, a similar algorithm to the one using laser range finders is implemented. In addition, we utilize color markers installed on a mobile robot to identify the mobile robot. Since color histogram is stable to deformation and occlusion of the objects relatively (Swain & Ballard, 1991), it is qualified as unique feature value to represent each object. Compared with the contour and so on, color histogram of the object stays largely unchanged against the various images that are captured by the distributed cameras.

Following three processes are performed to detect color markers.

1. Background subtraction: The background subtraction process compares the current image frame and an adaptive background image frame to find parts of the image that have changed due to the moving object.

2. Color histogram matching: The color histogram matching process searches over the current image and finds target colors which are registered in advance.
   In the previous processes, the system does not discriminate if each feature point belongs to the color marker or is just noise. So we need the following segmentation process to detect target objects and remove noise.
3. Segmentation: The overlapped areas of results from both background subtraction and color histogram matching are clustered. Clustering algorithm is based on nearest neighbor method. If the distance in both the x and y direction between two pixels is lower than a given threshold, these pixels are considered as part of the same object. In case that the number of pixels, height or width of the cluster does not get to a certain value, the cluster is removed as noise.

Figure 6 (left) and (right) show a captured image and the result of the color marker detection, respectively. We can find that three color markers on the top of the mobile robot are detected successfully.

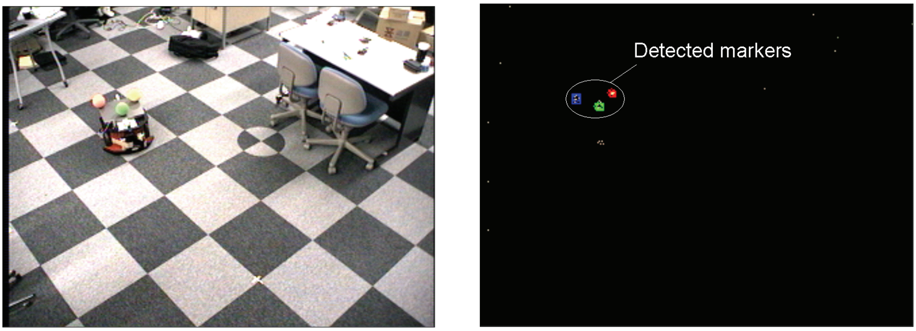


Fig. 6. Detection of color markers on a mobile robot using a camera DIND

4.3 Calibration of cameras based on the corresponding points

The automated camera calibration is performed based on the positions of the robots in the world coordinate system (x^g_k, y^g_k) and their corresponding points in the image coordinate system (u^i_k, v^i_k) (k = 1, 2, ..., n). Here we apply Tsai's method (Tsai, 1987) for the calibration. Although the method requires a 3D calibration object, we can use mobile robots with different heights or multiple markers attached to different heights.

In the Tsai's method, following 11 parameters are calibrated.

Five intrinsic parameters are:

- f: effective focal length
- κ_1: 1st order lens distortion coefficient
- s_x: uncertainty scale factor for x
- (C_x, C_y): computer image coordinate for the origin in the image plane

Six extrinsic parameters are:

- R_x, R_y, R_z: rotation angles for the transform between the world and camera coordinate frames
- T_x, T_y, T_z: translational components for the transform between the world and camera coordinate frames

5. Mobile robot localization

In calibration methods based on corresponding points shown in section 3 and 4, the position information of robots in the world coordinate system is required. In our mobile robot localization method, the position of the mobile robot measured by the 3D ultrasonic positioning system installed in the space and the mobile robot on-board sensor data (wheel encoder) are fused to minimize the position error. In the following subsections, the detail of the localization method is given.

5.1 Model of the mobile robot

We consider a two-wheeled mobile robot model shown in Fig. 7. Let ${}^{w}O$-${}^{w}x^{w}y$ be the coordinate system fixed to iSpace (world coordinate system) and ${}^{R}O$-${}^{R}x^{R}y$ be the coordinate system fixed to the mobile robot (robot coordinate system). The position and orientation of the mobile robot are denoted by (x, y, θ) in the world coordinate system. The control inputs for the mobile robot are the translational velocity v and rotational velocity ω. Here, the kinematic model for the mobile robot is expressed as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{7}$$

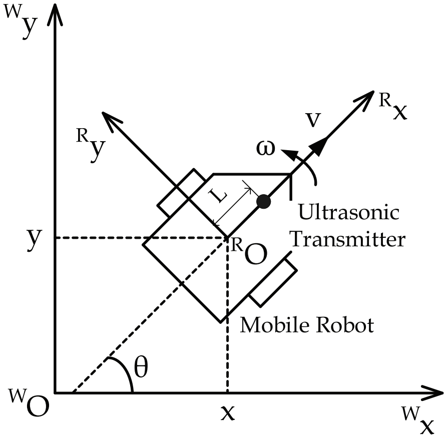


Fig. 7. Model of a mobile robot

In addition, an ultrasonic transmitter used with the ultrasonic positioning system is installed on the mobile robot. Its coordinate in the robot coordinate system is $(L, 0)$.

5.2 Localization using extended Kalman filter

The position and orientation of the mobile robot are estimated based on data from iSpace (the 3D ultrasonic positioning system) and the mobile robot (wheel encoder). These two measurement data are fused using extended Kalman filter. Extended Kalman filter has been widely applied to sensor fusion problems and it is computationally-efficient compared to other probabilistic methods, e.g. particle filter (Thrun et al., 2005).

In order to implement the extended Kalman filter, the model of the system has to be developed. Discretizing (7), we obtain the following state equation:

$$\begin{bmatrix} x_{\mathrm{k}} \\ y_{\mathrm{k}} \\ \theta_{\mathrm{k}} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{k}-1} + v\Delta t\cos\theta_{\mathrm{k}-1} \\ y_{\mathrm{k}-1} + v\Delta t\sin\theta_{\mathrm{k}-1} \\ \theta_{\mathrm{k}-1} + \omega\Delta t \end{bmatrix} + \mathbf{W}_{\mathrm{k}} w_{\mathrm{k}} \tag{8}$$

where x_{k}, y_{k} and θ_{k} denote position and orientation of the mobile robot at time *k*, Δt is the sampling rate, v and ω are the translational velocity and the rotational velocity obtained from encoders, w_k represents the process noise.
The observation equation is expressed as follows:

$$\begin{bmatrix} x_{\mathrm{zps}} \\ y_{\mathrm{zps}} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{k}} + L\cos\theta_{\mathrm{k}} \\ y_{\mathrm{k}} + L\sin\theta_{\mathrm{k}} \end{bmatrix} + \mathbf{V}_{\mathrm{k}} v_{\mathrm{k}} \tag{9}$$

where (x_{zps}, y_{zps}) is the position of the ultrasonic transmitter in the world coordinate system observed by iSpace and v_k represents the measurement noise. *L* is the distance on the central axis between the tag and robot center, as noted in Fig. 7.
Linearizing the state equation, Jacobian matrix $\boldsymbol{A}_k$ is obtained:

$$\mathbf{A}_{\mathrm{k}} = \begin{bmatrix} 1 & 0 & -v\Delta t\sin\theta_{\mathrm{k}-1} \\ 0 & 1 & v\Delta t\cos\theta_{\mathrm{k}-1} \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

We consider that the noise on the encoder is white noise with a normal distribution. Here, the matrix $\boldsymbol{W}_k$ is expressed as follows:

$$\mathbf{W}_{\mathrm{k}} = \begin{bmatrix} -\Delta t\cos\theta_{\mathrm{k}-1} & 0 \\ -\Delta t\sin\theta_{\mathrm{k}-1} & 0 \\ 0 & -\Delta t \end{bmatrix} \tag{11}$$

From the observation equation, the matrix $\boldsymbol{H}_k$ is

$$\mathbf{H}_{\mathrm{k}} = \begin{bmatrix} 1 & 0 & -L\sin\theta_{\mathrm{k}} \\ 0 & 1 & L\cos\theta_{\mathrm{k}} \end{bmatrix} \tag{12}$$

The matrix $\boldsymbol{V}_k$ is determined as follows:

$$\mathbf{V}_{\mathrm{k}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{13}$$

In this research, we assume the process noise covariance $\boldsymbol{Q}$ and measurement noise covariance $\boldsymbol{R}$ are constant and use diagonal matrices. The values are tuned experimentally.

6. Experiment

6.1 Calibration of distributed laser range finders

In the environment shown in Fig. 8, three laser range finders are calibrated using a mobile robot. The arrangement of the laser range finders and the actual path of the mobile robot, which is estimated by the mobile robot localization method described in the previous section, are also shown in the figure. The mobile robot entered the room from the right part of Fig. 8 and circled around the room in clockwise direction. In the mobile robot navigation system installed in iSpace, to find the best way for the robot to move through the space towards goals (path planning), the Field D* method (Ferguson & Stentz, 2005) is used. Moreover, in order for the robot to follow the calculated path and at the same time avoid bumping into obstacles, the Dynamic Window Approach (DWA) (Fox et al., 1997) is applied as a local control algorithm.

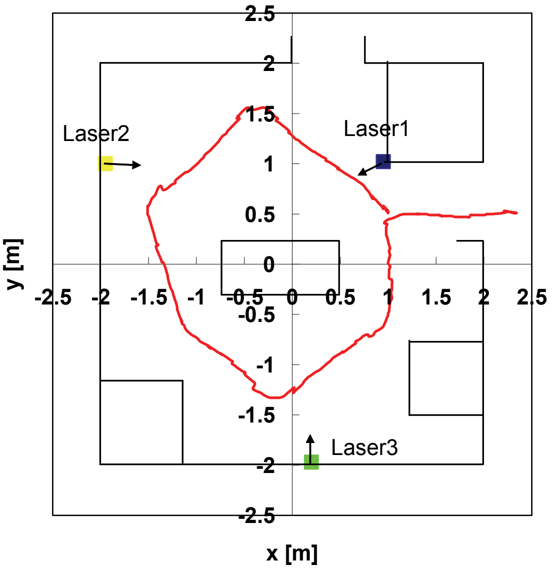


Fig. 8. Experimental environment for calibration of laser range finders – arrangement of the sensors and path of the mobile robot estimated by the extended Kalman filter

In order to evaluate the accuracy of the proposed method, we make a comparison between the automated calibration and manual calibration. In the case of manual calibration, a calibration object (an object which can be well detected by a laser range finder) is placed in turn on several points with known global coordinates and the calibration parameters are calculated by (3)–(5).

Table I shows the result of the automated calibration compared to that of manual one. The maximum difference between manual calibration and automated calibration is 0.11 meters for translation and 0.06 radians for rotation. The laser range fingers can only measure the edge of the mobile robot, but the tracking process works well and almost the same result as manual calibration case is achieved. The tracked positions of the mobile robot in each sensor node, which are transformed into the global coordinates using the estimated parameters are shown in Fig. 9. It is obvious that there is a good correspondence between the tracks, which shows that the calibrated parameters are correct.

LRF ID i	Automated / Manual		
	T^i_{xg} [m]	T^i_{yg} [m]	θ^i_g [rad]
1	−1.91 / −1.95	1.10 / 1.00	−0.19 / −0.15
2	0.99 / 0.95	0.95 / 1.02	−2.60 / −2.54
3	0.09 / 0.20	−2.01 / −1.97	1.56 / 1.60

Table 1. Estimated pose of the distributed laser range finders - comparison of automated and manual calibration results

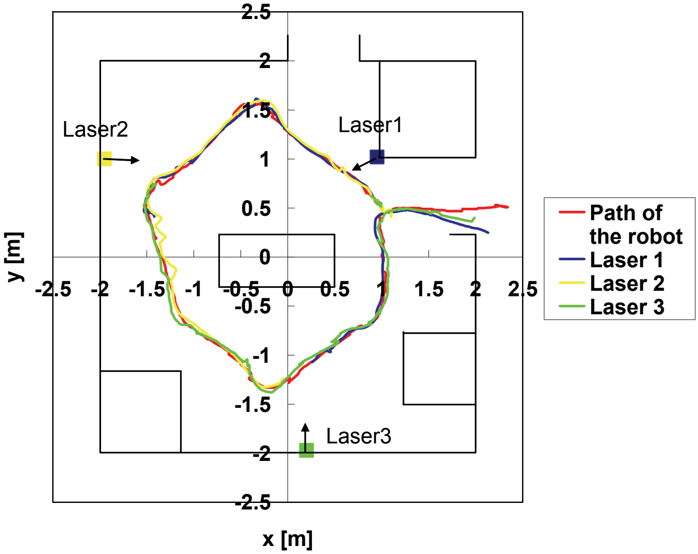


Fig. 9. The actual path of the mobile robot (presented again) and the tracked positions of the robot in each sensor node, obtained by transformation into the global coordinates using the estimated parameters

6.2 Calibration of distributed cameras

Figure 10 shows reference path followed by mobile robots and camera arrangement. In this experiment, four CCD cameras are calibrated and two mobile robots with different height followed the same "∞" shaped path defined as

$$\begin{aligned} x &= \alpha_x \sin(\beta t / 2) \\ y &= \alpha_y \sin(\beta t) \end{aligned} \tag{14}$$

where α_x, α_y, and β are constants, and t is a time step. Note that we can use an arbitrary path for calibration because we utilize only position information of the color markers. The observable area of each camera on the ground plane is about 4 meters × 4 meters. The captured image size is 320 pixels × 240 pixels.

In order to evaluate the accuracy of the proposed method, we made a comparison between the automated calibration and manual calibration. If the height of a place is known, the three

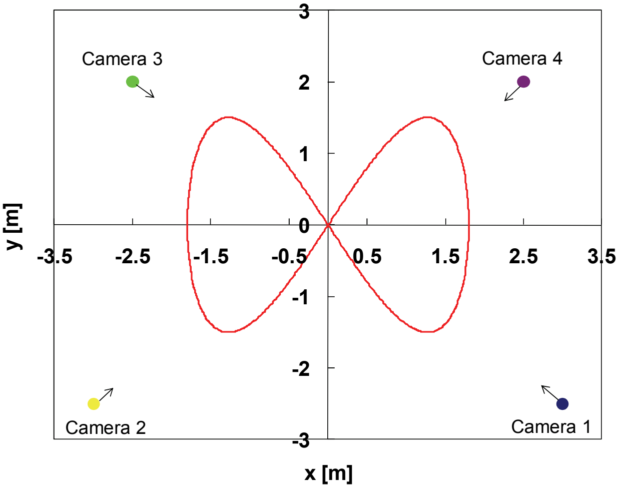


Fig. 10. Reference path followed by mobile robots and camera arrangement

dimensional position is reconstructed from one camera image. Therefore, we selected sample points shown in Fig. 11 from the image, and the corresponding positions on the ground plane (z=0) are reconstructed. The reference positions of sampled points are obtained from the ultrasonic 3D positioning system.

Table 2 shows the comparison of mean and maximum errors, which are the average and maximum Euclidean distance between the reference positions and the reconstructed positions of the nine sampled points, respectively. Although the difference of the average error is less than 8 millimeters, the manual calibration is more precise than the automated calibration in most cases. Especially, the maximum error is relatively large in the case of the automated calibration. This is mainly due to the fact that the obtained positions of the markers are not widely distributed on the image plane. This means that the path of the mobile robot is important for the calibration. In iSpace, this problem would be solved by mutual cooperation of DINDs and mobile agents - DINDs should ask mobile agents to move to the area where the corresponding points are not acquired enough.

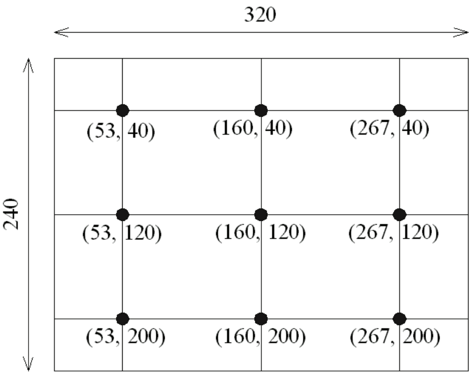


Fig. 11. Sampled points for evaluation of the camera calibration. The unit is in pixels.

Camera ID	Automated / Manual	
	Mean error [mm]	Maximum error [mm]
1	67.5 / 40.2	128.4 / 72.8
2	81.4 / 29.0	140.8 /54.7
3	54.6 / 64.6	116.4/ 91.3
4	32.6 /72.5	73.2 / 116.5
Average	59.0 / 51.6	114.7 / 83.8

Table 2. Comparison of mean and maximum error between automated and manual calibration of distributed cameras

7. Conclusion

We described an automated calibration method for distributed sensors in Intelligent Space (iSpace) by using mobile robots. The proposed method utilizes the positions of the robots in the world coordinate system and their corresponding points in the local coordinate system. The mobile robot localization and moving object tracking functions of iSpace are extended and calibration of distributed laser range finders and vision sensors is realized. In our work we used the ultrasound positioning system to localize the mobile robot. In a real environment this type of global positioning system is not always available, but it can still be possible to estimate the robot's position by using other already calibrated sensors or implementing a self localization method based on a preexisting map.

Performance of the proposed method was demonstrated experimentally. The experimental result shows that the method can provide enough accuracy for the various applications in iSpace such as mobile robot localization, human tracking, iSpace-human interface (Niituma et al., 2007) and so on.

For future work, since the proposed calibration method is affected by the error of the object tracking, we will develop a more accurate tracking algorithm. In addition, as mentioned in section 5.2, optimization of paths of mobile robots for calibration is another research direction. We will also consider utilization of the inter-sensor information for accuracy improvement and computational stability.

8. References

Agrawal, M. & Davis, L. (2003). Camera calibration using spheres: A semidefinite programming approach, *Proceedings of 9th IEEE International Conference on Computer Vision,* Vol.2, pp.782-789, ISBN 0-7695-1950-4, Nice, France, Oct. 2003.

Brscic, D. & Hashimoto, H. (2006). Tracking of Objects in Intelligent Space Using Laser Range Finders, *Proceedings of IEEE International Conference on Industrial Technology,* pp.1723-1728, ISBN 1-4244-0726-5, Mumbai, India, Dec. 2006.

Cook, D. J. & Das, S. K. (2004). Smart Environments: Technologies, Protocols, and Applications (Wiley Series on Parallel and Distributed Computing), Wiley-Interscience, ISBN 0-471-54448-7, USA.

Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part I, *IEEE Robotics and Automation Magazine,* Vol.13, No.2, (Jun. 2006) pp.99-110, ISSN 1070-9932.

Ferguson, D. & Stentz, A. (2005). The Field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments, *Technical Report CMURI-TR-05-19,* Robotics Institute, Jun. 2005.

Fox, D.; Burgard, W. & Thrun, S. (1997). The dynamic window approach to collision avoidance, *IEEE Robotics and Automation Magazine,* Vol.4, No.1, (Mar. 1997) pp.23-33, ISSN 1070-9932.

Johanson, B.; Fox, A. & Winograd, T. (2002). The Interactive Workspaces project: experiences with ubiquitous computing rooms, *IEEE Pervasive Computing,* Vol.1, No.2, (Apr.-Jun. 2002) pp.67-74, ISSN 1536-1268.

Koutsonikolas, D.; Das, S. M. & Hu, Y. C. (2006). Path planning of mobile landmarks for localization in wireless sensor networks, *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops,* pp.86(1)-86(7), ISBN 0-7695-2541-5, Lisboa, Portugal, 2006.

Lee, J.-H. & Hashimoto, H. (2002). Intelligent Space - concept and contents, *Advanced Robotics,* Vol.16, No.3, (Apr. 2002) pp.265-280, ISSN 0169-1864.

Lee, J.-H. & Hashimoto, H. (2003). Controlling mobile robots in distributed intelligent sensor network, *IEEE Transaction on Industrial Electronics,* Vol.50, No.5, (Oct. 2003) pp.890-902, ISSN 0278-0046.

Malm, H. & Heyden, A. (2001). Stereo head calibration from a planar object, *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Vol.2, pp.657-662, ISBN 0-7695-1272-0, Kauai, Hawaii, USA, Dec. 2001.

Mizoguchi, F.; Ohwada, H.; Nishiyama, H. & Hiraishi, H. (1999). Smart office robot collaboration based on multi-agent programming, *Artificial Intelligence,* Vol.114, No.1-2, (Oct. 1999) pp.57-94, ISSN 0004-3702.

Mynatt, E. D.; Melenhorst, A.-S.; Fisk, A.-D. & Rogers, W. A. (2004). Aware technologies for aging in place: understanding user needs and attitudes, *IEEE Pervasive Computing,* Vol.3, No.2, (Apr.-Jun. 2004) pp.36-41, ISSN 1536-1268.

Niitsuma, M.; Hashimoto, H. & Hashimoto, H. (2007). Spatial Memory as an aid system for human activity in Intelligent Space, *IEEE Transactions on Industrial Electronics,* Vol.54, No.2, (Apr. 2007) pp.1122-1131, ISSN 0278-0046.

Rekleitis, I. & Dudek, G. (2005). Automated calibration of a camera sensor network, *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp.3384-3389, ISBN 0-7803-8912-3, Edmonton, Alberta, Canada, Aug. 2005.

Shenoy, S. & Tan, J. (2005). Simultaneous localization and mobile robot navigation in a hybrid sensor network, *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp.1636-1641, ISBN 0-7803-8912-3, Edmonton, Alberta, Canada, Aug. 2005.

Sreenath, K.; Lewis, F. L. & Popa, D. O. (2006). Localization of a wireless sensor network with unattended ground sensors and some mobile robots, *Proceedings of the 2006 IEEE Conference on Robotics, Automation and Mechatronics,* pp.1-8, ISBN 1-4244-0025-2, Bangkok, Thailand, Dec. 2006.

Sturm, P. F. & Maybank, S. J. (1999). On plane-based camera calibration: A general algorithm, singularities, applications, *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Vol.1, pp.432-437, ISBN 0-7695-0149-4, Fort Collins, Colorado, USA, Jun. 1999.

Swain, M. J. & Ballard, D.H. (1991). Color indexing, *International Journal of Computer Vision,* Vol.7, No.1, (Nov. 1991) pp.11-32, ISSN 0920-5691.

Thrun, S; Burgard, W. & Fox, D. (2005). *Probabilistic Robotics,* MIT Press, ISBN 0-262-20162-3, USA.

Tsai, R. Y. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE Journal of Robotics and Automation,* Vol.RA-3, No.4, (Aug. 1987) pp.323-344, ISSN 0882-4967.

Ueshiba, T. & Tomita, F. (2003). Plane-based calibration algorithm for multi-camera systems via factorization of homology matrices, *Proceedings of 9h IEEE International Conference on Computer Vision,* Vol.2, pp.966-973, ISBN 0-7695-1950-4, Nice, France, Oct. 2003.

Wu, Y.; Zhu, H.; Hu, Z. & Wu, F. (2004). Camera calibration from the quasi-affine invariance of two parallel circles, *Proceedings of the 8th European Conference on Computer Vision,* Vol.1, pp.190-202, ISBN 3-540-21984-6, Prague, Czech Republic, May 2004.

Zhang, Z. (2000). A flexible new technique for camera calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.22, No.11, (Nov. 2000) pp.1330-1334, ISSN 0162-8828.

Zhang, Z. (2004). Camera calibration with one-dimensional objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.26, No.7, (Jul. 2004) pp.892-899, ISSN 0162-8828.

13

Image-based Tracking of Deformable Surfaces

Anna Hilsmann, David C. Schneider and Peter Eisert
Fraunhofer Institute for Telecommunications Heinrich-Hertz-Institute
Humboldt Universität zu Berlin
Germany

1. Introduction

Deformable objects and surfaces are ubiquitous in modern computer vision applications, including medical imaging [Metaxas (1996)], object-based video compression [Ostermann (1994)] and augmented reality [Hilsmann et al. (2010); Pilet et al. (2008)]. In this chapter, we are particularly interested in tracking surfaces whose deformations are difficult to describe, such as drapery of textiles, facial expressions or complex organ motion in medical image sequences. Capturing non-rigid motion of such objects in 2D images is a challenging task because the deformation field complexity is unknown a priori. Therefore, the assumed model has to cope with various types of deformations.

Our framework makes use of image-based deformation models and formulates a robust cost function that can be minimized with common optimization methods, like Gauss-Newton or Levenberg-Marquardt. It is highly modular and can therefore be applied to a broad variety of motion estimation problems, as the motion model can individually be formulated according to the given problem. To avoid error accumulation over the image sequence, we use an analysis-by-synthesis approach, where the error minimization is always carried out between a synthesized reference image built from previous parameter estimates and the actual camera frame. However, in real scenes, tracking is often influenced by varying lighting. If intensity changes due to changes in the lighting conditions in the scene are not considered, the intensity difference between the synthesized reference image and the current frame increases and causes errors in the motion estimation. To avoid these errors, it is essential to also model intensity changes between the images and warp the reference image not only spatially but also photometrically. We will present results from different fields of applications, such as real-time augmented reality (Figure 1), facial expression analysis (Figure 11), and medical applications (Figure 10).

This chapter is structured as follows. Section 2 will briefly review existing literature in the field of deformable tracking and discuss the contribution of our work. Section 3 will give an overview of the mathematical notation used in this chapter. Following, we present different types of deformable motion models in section 4 before we explain our approach to image-based optimization of the model parameters in section 5. Section 6 will present results achieved with our approach for different types of surfaces, such as cloth, faces, and medical images.

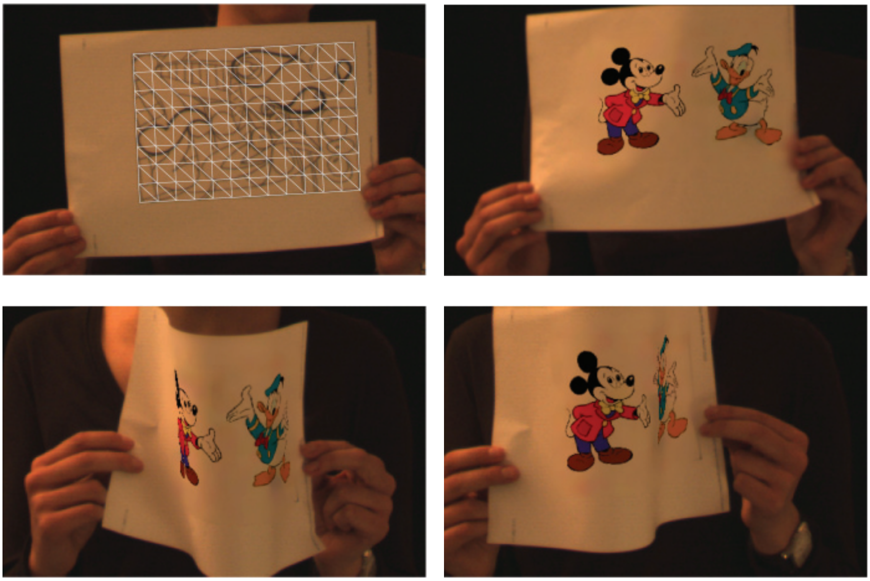


Fig. 1. One application we are targeting: Tracking and retexturing a piece of paper in a video sequence (original frame with overlaid deformation mesh and retextured frames).

2. Related work

Motion between two frames is commonly described by a dense displacement vector field which links the location of each pixel in a given frame to its location in the next frame. Registering a model to an image and tracking of deformable objects is a large field of research and there exists a multitude of methods. Basically, the literature on deformable surface tracking from monocular image sequences distinguishes between marker-based [Scholz & Magnor (2006); White & Forsyth (2006)], feature-based [Pilet et al. (2008)], and image-based [Bartoli & Zisserman (2004); Lim & Yang (2005); Torresani et al. (2001)] methods. As markers are not always available in real scenes, the assumption of such a-priori knowledge limits the applicability of these methods to very special cases. Feature-based methods minimize a distance between a few corresponding feature points, whereas direct methods minimize an error measure that is based on image information of all pixels in the image.

Feature-based methods determine correspondences between distinct feature points and use them to estimate the best transformation between these correspondences. Various features have been described in the literature. Image-based feature points are e.g. local curvature extrema or saddle points, edges or corners [Harris & Stephens (1988); Thirion (1996)] or SIFT [Lowe (2003)] and SURF [Bay et al. (2006)]features. Popular choices of shape-features are the shape context [Belongie et al. (2002)] or statistical moments of the shape [Chen (1993)]. Feature- based methods are mostly used to find rigid or affine transformation. If the set of points is large enough, also more complex transformations can be determined using e.g. radial basis functions [Bookstein (1989)]. The type of the RBF determines overall characteristics of the transformation such as the smoothness or the locality. [Pilet et al. (2008)] proposed a feature-based real-time method for deformable object detection and tracking that uses a wide baseline matching algorithm and deformable meshes. They estimate the irradiance of the surface separately by warping the reference image to the current frame and estimating the luminance ratio between both images.

Generally, image-based methods yield more accurate results in non-rigid deformation estimation than feature-based methods because they exploit the entire image instead of distinct points. [Bartoli & Zisserman (2004)] presented an optical flow based approach that uses radial basis functions to regularize the flow field. They iteratively insert new center-points for the radial basis functions based on examination of the error image after each iteration. The number of centers grows until the algorithm converges. Recently, [Gay Bellile et al. (2007)] proposed a direct method to estimate deformable motion under self-occlusions by establishing occlusion maps and penalizing a variation in the spatial warp derivative along some direction to prevent mesh-foldings and cope with self-occlusions. [Hilsmann & Eisert (2008)] utilized the idea of mesh-shrinking in an optical flow-based approach by exploiting topological relationships of a mesh-based warp to force the mesh to shrink instead of fold at the occlusion boundary.

While, in general, marker- or feature-based methods are more robust against illumination changes – assuming markers or features are illumination invariant – direct methods usually minimize an error function based on the intensity differences between the aligned images. Therefore, these methods are sensitive to illumination variations. [Gennert & Negahdaripour (1987)] were among the first to propose a direct method robust to illumination variations. They assume that the brightness at time $t + \delta t$ is related to the brightness at time t through a set of parameters that can be estimated from the image sequence. Several other researchers [Bartoli (2008); Haussecker & Fleet (2001); Nastar et al. (1996); Silveira & Malis (2007)] have exploited their ideas to make tracking more robust against lighting changes. [Bartoli (2008)] proposed a dual inverse compositional algorithm to estimate a homography and an affine photometric registration. The registration results are improved by the photometric registration but only global changes are modeled and thus specular reflections or local shadows are not taken into consideration. [Silveira & Malis (2007)] model local illumination changes to make a homography estimation more robust against generic illumination changes. [Pizarro & Bartoli (2007)] proposed to transform images into a 1D shadow invariant space to achieve direct image registration in the presence of even sharp shadows. [Hilsmann & Eisert (2009)] used an extended optical flow constraint and mesh-based models not only as a correction factor for spatial registration but also to actually retrieve local photometric parameters for convincing retexturing purposes in augmented reality applications.

Besides the optimization scheme, state-of-the-art methods differ in the type of the assumed motion model. Tracking arbitrary deformable surfaces without any knowledge about the type of deformation in monocular video sequences is an ill-posed problem as the deformation yields too many ambiguities. Therefore, if no 3D information is required, one approach is to track deformations in the image plane. Some researchers use 2-dimensional mesh-based models [Pilet et al. (2008), Gay Bellile et al. (2007), Hilsmann & Eisert (2009)], others radial basis functions [Bartoli & Zisserman (2004), Bookstein (1989)] to model deformations in 2D. If the type of deformation or a 3D shape model is known a priori, also more specific deformation models can be used in the 3D space. Often, deformation is modeled by weighted superposition of basis functions like superquadrics [Terzopoulos & Metaxas (1991)] or PCA-based models [Salzmann et al. (2007)]. An important application for deformable surface analysis is face tracking [Metaxas (1996), Eisert & Girod (1998)], where specific face models are used to constrain the parameters to model facial deformations.

In this chapter we will present a general framework to deformable surface tracking. In our formulation we separate the motion model from the optimization framework and the regularization term. This makes the framework highly modular and thus adaptive to the given problem.

3. Notation

In general, we denote scalar valued variables with lower case roman letters, e.g. x, vector valued variables with bold lower case letters, e.g. $\mathbf{x}$, and matrices with bold upper case letters, e.g. $\mathbf{X}$. We denote a function f of a pixel position $\mathbf{x}$ parameterized by the parameter vector $\boldsymbol{\theta}$ by $f(\mathbf{x};\boldsymbol{\theta})$. The following list briefly introduces the mathematical notation used in this chapter.

$\mathbf{A}$	Adjacency matrix of a mesh	β	Barycentric coordinates
$\boldsymbol{\mathcal{D}}$	Displacement field	$\delta\boldsymbol{\theta}$	Parameter update
η	Bilinear coordinates	$\mathcal{E}$	Cost function
$\mathcal{E}_D$	Data term of the cost function	$\mathcal{E}_S$	Smoothness term of the cost function
f_x, f_y	scaled focal length	$\mathbf{g}_f$	$1 \times n$ gradient vector
$\mathbf{H}_f$	$n \times n$ Hessian matrix	$\mathcal{I}(\mathbf{x})$	Image intensity at location $\mathbf{x}$
$\mathbf{J_f}$	$m \times n$ Jacobian matrix	K	Number of vertices
$\mathbf{L}$	Laplacian matrix	$\mathbf{l}$	Direction of light source
$\mathbf{n}$	Surface normal	N	Number of parameters
$\mathcal{N}_k$	Neighborhood of a vertex $\mathbf{v}_k$	$\boldsymbol{\psi}_g$	Geometric warp function
ψ_p	Photometric warp function	$\boldsymbol{\psi}_d^3$	Warp function for 3D deformation
$\mathbf{p}$	3D object point	$\boldsymbol{\theta}$	Parameter vector
$\hat{\boldsymbol{\theta}}$	Estimated parameter vector	$\boldsymbol{\theta}_g$	Geometric parameter vector
$\boldsymbol{\theta}_p$	Photometric parameter vector	$\boldsymbol{\theta}_d$	Parameters for 3D deformation
$\mathcal{R}$	Region	$\mathbf{R}$	rotation matrix
$\mathbf{t}$	3d translation vector	$\mathbf{V}$	Diagonal matrix of vertex valences
$\mathbf{v}$	vertex	$\mathbf{x}$	Pixel coordinate

4. Deformable models

In our framework, the spatial deformation and motion of a deformable surface in an image are described by a geometric warp function

$$\boldsymbol{\psi}_g(\mathbf{x};\boldsymbol{\theta}) = \mathbf{x} + \boldsymbol{\mathcal{D}}(\mathbf{x};\boldsymbol{\theta}) \tag{1}$$

of the pixel coordinates. $\boldsymbol{\mathcal{D}}(\mathbf{x};\boldsymbol{\theta})$ is a dense 2-dimensional pixel displacement field defined at each image pixel $\mathbf{x}$ and parameterized by a parameter vector $\boldsymbol{\theta}$. Usually, two successive frames in a natural video sequence do not only differ geometrically, but also the intensity of a scene point can vary due to changes in the scene lighting, shading properties etc. For deformable surfaces, the intensity of a scene point varies when the surface normals change. If not handled correctly, varying lighting influences the geometric tracking result, especially for tracking approaches that are based on a brightness constancy assumptions (see section 5.2). Hence, we model intensity changes in an additional photometric warp function

$$\psi_p(\mathcal{I}(\mathbf{x});\boldsymbol{\theta}) = \psi_p(\mathbf{x};\boldsymbol{\theta}) \cdot \mathcal{I}(\mathbf{x}) \tag{2}$$

which is applied multiplicatively to the image intensities. Hence, the parameter vector $\boldsymbol{\theta}$ consists of a geometric part $\boldsymbol{\theta}_g$ and a photometric part $\boldsymbol{\theta}_p$ such that the resulting parameter vector in our framework is given by

$$\boldsymbol{\theta} = \left[\boldsymbol{\theta}_g^T \; \boldsymbol{\theta}_p^T\right]^T .$$

Based on the given tracking problem (type of motion, required accuracy of the deformation field, real-time constraints etc.), different types of parameterization can be chosen. In the following, we will present different motion models used for different types of application.

4.1 2D models

The simplest 2-dimensional model is a dense displacement field, i.e. one displacement vector for each pixel. This *dense model* holds two parameters per pixel. To reduce the number of parameters for e.g. real-time applications or to introduce prior knowledge on the deformation field, we can also model the warp using a 2-dimensional *mesh-based model* with K vertices $\mathbf{v}_k$. Each vertex is associated with two parameters, i.e. its displacements in x- and y-direction $\mathbf{d}_k = [d_{kx} \; d_{ky}]^T$. The warps $\psi_g(\mathbf{x};\boldsymbol{\theta})$ and $\psi_p(\mathbf{x};\boldsymbol{\theta})$ can then be parameterized by arbitrary basis functions defining the deformation field. Different parameterizations will be presented below. In each case, the geometric parameter vector is given in the following form by concatenating the x- and y-coordinates of the displacements:

$$\boldsymbol{\theta}_g = \left[d_{1x} \; ... \; d_{Kx} \; d_{1y} \; ... \; d_{Ky}\right]^T . \tag{3}$$

The photometric warp is parameterized similarly to the geometric warp by K photometric parameters

$$\boldsymbol{\theta}_p = \left[\rho_1 \; ... \; \rho_K\right]^T . \tag{4}$$

Having defined the parameter vector, we introduce a matrix notation of the warps:

$$\begin{aligned} \psi_g(\mathbf{x}_i;\boldsymbol{\theta}) &= \mathbf{x}_i + \mathbf{M}_g^{\mathbf{x}_i} \cdot \boldsymbol{\theta} \\ \psi_p\left(\mathbf{x}_i;\boldsymbol{\theta}\right) &= \mathbf{m}_p^{\mathbf{x}_i} \cdot \boldsymbol{\theta} \end{aligned} \tag{5}$$

where $\mathbf{M}_g^{\mathbf{x}_i}$ and $\mathbf{m}_p^{\mathbf{x}_i}$ are $2 \times N$ and $1 \times N$ matrices defining the parameterization. The superscript $^{\mathbf{x}_i}$ denotes that these matrices differ for each pixel. They depend on the type of parameterization as well as pixel position and will be defined below. We can now easily determine the warp Jacobians, which are required for optimization:

$$\begin{aligned} \mathbf{J}_{\psi_g}(\mathbf{x}_i;\boldsymbol{\theta}) &= \mathbf{M}_g^{\mathbf{x}_i} \\ \mathbf{J}_{\psi_p}(\mathbf{x}_i;\boldsymbol{\theta}) &= \mathbf{m}_p^{\mathbf{x}_i} \end{aligned} \tag{6}$$

4.1.1 Dense model

The dense model holds one displacement vector per pixel, such that the number of parameters is $N = 2p$ for a region of interest with p pixels. The matrices $\mathbf{M}_g^{\mathbf{x}_i}$ and $\mathbf{m}_p^{\mathbf{x}_i}$ are zero matrices with only one entry in the i^{th}, the$(i+p)^{th}$ and the $(i+2p)^{th}$ column:

$$\begin{aligned} \mathbf{M}_g^{\mathbf{x}_i} &= \underbrace{\begin{bmatrix} 0... & 1... & 0... & 0... & 0... & 0... & 0... & 0... & 0... \\ 0... & 0... & 0... & 0... & 1... & 0... & 0... & 0... & 0... \end{bmatrix}}_{(2\times K)\quad(2\times K)\quad(2\times K)} \\ \mathbf{m}_p^{\mathbf{x}_i} &= [\underbrace{0...\ 0...\ 0...}_{(1\times K)}\ \underbrace{0...\ 0...\ 0...}_{(1\times K)}\ \underbrace{0...\ 1...\ 0...}_{(1\times K)}] \end{aligned} \tag{7}$$

This model is equivalent to the classic optical flow. Due to the aperture problem, the normal equations that arise from the matrices $\mathbf{M}_g^{\mathbf{x}_i}$ during optimization are rank deficient and regularization is a necessity. The Laplacian smoothness term addressed in section 4.3 is one possible choice for regularization.

4.1.2 Affine mesh-based model

In case of a mesh-based model, the number of parameters is determined by the number of vertices in the mesh, such that $N = 2K$ where K is the number of vertices. If a pixel $\mathbf{x}_i$ is surrounded by a triangle consisting of the three mesh vertices $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c$ with indices a, b, c and $\beta_a, \beta_b, \beta_c$ are the three corresponding Barycentric coordinates, it can be represented by the weighted sum of its enclosing vertices:

$$\mathbf{x}_i = \sum_{j\in\{a,b,c\}} \beta_j \mathbf{v}_j \qquad \beta_a + \beta_b + \beta_c = 1,\ 0 \leq \beta_{a,b,c} \leq 1 \tag{8}$$

A warp with piecewise affine interpolation between the respective three surrounding vertex positions keeps the Barycentric coordinates constant, such that the warps $\psi_g(\mathbf{x};\boldsymbol{\theta})$ and $\psi_p(\mathbf{x};\boldsymbol{\theta})$ can then be parameterized similarly by

$$\begin{aligned} \psi_g\left(\mathbf{x}_i;\boldsymbol{\theta}\right) &= \mathbf{x}_i + \boldsymbol{\mathcal{D}}\left(\mathbf{x}_i;\boldsymbol{\theta}\right) = \mathbf{x}_i + \sum_{j\in\{a,b,c\}} \beta_j \mathbf{d}_j \\ \psi_p\left(\mathbf{x}_i;\boldsymbol{\theta}\right) &= \sum_{j\in\{a,b,c\}} \beta_j \rho_j \end{aligned} \tag{9}$$

This can be formulated in matrix notation as in equation (5) with the following matrices

$$\begin{aligned} \mathbf{M}_g^{\mathbf{x}_i} &= \underbrace{\begin{bmatrix} \beta_a... & \beta_b... & \beta_c... & 0... & 0... & 0... & 0... & 0... & 0... \\ 0... & 0... & 0... & \beta_a... & \beta_b... & \beta_c... & 0... & 0... & 0... \end{bmatrix}}_{(2\times K)\quad(2\times K)\quad(2\times K)} \\ \mathbf{m}_p^{\mathbf{x}_i} &= [\underbrace{0...\ 0...\ 0...}_{(1\times K)}\ \underbrace{0...\ 0...\ 0...}_{(1\times K)}\ \underbrace{\beta_a...\ \beta_b...\ \beta_c...}_{(1\times K)}] \end{aligned} \tag{10}$$

4.1.3 Bilinear mesh-based model

Similar to the affine parameterization, we can use a bilinear parameterization between the respective four surrounding vertex positions

$$\begin{aligned}\psi_g\left(\mathbf{x}_i;\boldsymbol{\theta}\right) &= \mathbf{x}_i + \boldsymbol{\mathcal{D}}\left(\mathbf{x}_i;\boldsymbol{\theta}\right) = \mathbf{x}_i + \sum_{j\in\{a,b,c,d\}} \eta_j \mathbf{d}_j \\ \psi_p\left(\mathbf{x}_i;\boldsymbol{\theta}\right) &= \sum_{j\in\{a,b,c,d\}} \eta_j \rho_j \\ \eta_a = (1-s)(1-t),\ \eta_b &= s(1-t),\ \eta_c = t(1-s),\ \eta_d = st\end{aligned} \tag{11}$$

$\mathbf{M}_g^{\mathbf{x}_i}$ and $\mathbf{m}_p^{\mathbf{x}_i}$ are matrices composed similarly as for the affine parameterization above except that each row now has four entries $(\eta_a, \eta_b, \eta_c, \eta_d)$ in the corresponding columns.

4.2 3D models

Rather than modelling the motion directly in the 2D image plane, the displacement field (1) can be regarded as the projection of an object's 3D motion and deformation into the 2D domain. Similar to the 2D case, we can specify the deformation of a 3D object, given by its 3D surface points $\mathbf{p}_i$, as a function of deformation parameters $\boldsymbol{\theta}_d$

$$\psi_d^3(\mathbf{p};\boldsymbol{\theta}_d) = \mathbf{p} + \boldsymbol{\mathcal{D}}^3(\mathbf{p};\boldsymbol{\theta}_d). \tag{12}$$

For linear deformation fields, the geometrical warp can then be expressed by a deformation matrix $\mathbf{D}_d^{\mathbf{p}_i}$

$$\psi_d^3(\mathbf{p}_i;\boldsymbol{\theta}_d) = \mathbf{p}_i + \mathbf{D}_d^{\mathbf{p}_i} \cdot \boldsymbol{\theta}_d. \tag{13}$$

This deformation matrix can contain any basis functions that define the deviations of object points from a neutral shape $\mathbf{p}_i$. The amount of deformation performed in the local object coordinate system is controlled by the deformation parameter vector $\boldsymbol{\theta}_d$. Similarly, 3D deformation can be modelled by PCA analysis, with $\mathbf{p}_i$ being a mean shape and $\mathbf{D}_d^{\mathbf{p}_i}$ holding the Eigenvectors of the covariance matrix formed by several sample shapes.

A rigid body transform, specified by rotation matrix $\mathbf{R}_0$ and translation vector $\mathbf{t}_0$, can be used to position and orient the 3D object in a world coordinate system. In addition to deformation, the object is allowed to move which can be described by an update of rotation $\mathbf{R}$ with Euler angles R_x, R_y, R_z and translation $\mathbf{t} = [t_x, t_y, t_z]^T$. The entire geometrical warp in the world coordinate system is then given as

$$\psi_g^3(\mathbf{p}_i;\boldsymbol{\theta}_d) = \mathbf{R} \cdot \mathbf{R}_0(\mathbf{p}_i + \mathbf{D}_d^{\mathbf{p}_i} \cdot \boldsymbol{\theta}_d) + \mathbf{t} + \mathbf{t}_0. \tag{14}$$

For small changes of the parameter vector $\boldsymbol{\theta}_d$ this can be approximated linearly by approximating small rotations by a deformation along the object points' tangents, resulting in

$$\psi_g^3(\mathbf{p}_i;\boldsymbol{\theta}_g) \approx \mathbf{p}_i + \mathbf{D}_g^{\mathbf{p}_i} \cdot \boldsymbol{\theta}_g \tag{15}$$

with the extended parameter vector $\boldsymbol{\theta}_g$.

$$\boldsymbol{\theta}_g = [\boldsymbol{\theta}_d^T, R_x, R_y, R_z, t_x, t_y, t_z]^T \tag{16}$$

holding both deformation and pose parameters. The new matrix $\mathbf{D}_g^{\mathbf{p}_i}$ now covers information from both deformation and rigid body transform

$$\mathbf{D}_g^{\mathbf{p}_i} = \left[\mathbf{R}_0 \cdot \mathbf{D}_d^{\mathbf{p}_i}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times (\mathbf{R}_0 \mathbf{p}_i), \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times (\mathbf{R}_0 \mathbf{p}_i), \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (\mathbf{R}_0 \mathbf{p}_i), \mathbf{I}_3 \right]. \tag{17}$$

The geometrical warp in 3D space given by (14) or (15) can be related with the 2D geometric warp $\boldsymbol{\psi}_g$ via the camera projection. For perspective projection

$$\begin{aligned} x_i &= -f_x \cdot \frac{p_{x,i}}{p_{z,i}} \\ y_i &= -f_y \cdot \frac{p_{y,i}}{p_{z,i}} \end{aligned} \tag{18}$$

with scaled focal lengths f_x and f_y as camera parameters, the linearized displacement field is given by [Eisert & Girod (1998)]

$$\boldsymbol{\psi}_g(\mathbf{x}_i, \boldsymbol{\theta}) \approx \mathbf{x}_i + \underbrace{\begin{bmatrix} -\frac{1}{z(\mathbf{x}_i)}(f_x \mathbf{D}_{g,x}^{\mathbf{p}_i} + x_i \mathbf{D}_{g,z}^{\mathbf{p}_i}) \\ -\frac{1}{z(\mathbf{x}_i)}(f_y \mathbf{D}_{g,y}^{\mathbf{p}_i} + y_i \mathbf{D}_{g,z}^{\mathbf{p}_i}) \end{bmatrix}}_{J_{\psi_g}} \cdot \boldsymbol{\theta}_g \tag{19}$$

with $\mathbf{D}_{g,x,y,z}^{\mathbf{p}_i}$ being the $x-$, $y-$, and $z-$ component of the deformation matrix $\mathbf{D}_g^{\mathbf{p}_i}$ and $z(\mathbf{x}_i)$ the distance of the object point to the camera at pixel $\mathbf{x}_i$.
The geometric warp in the 3D case is attached in the local object coordinate system allowing a semantically meaningful modelling of surface deformation. Similarly, the photometric scaling can be described by compact illumination and reflection models, since geometry and surface normal information can be derived. For a simple shading model consisting of ambient and diffuse light with Lambertian reflection, the photometric warp ψ_p in (2) is given as

$$\psi_p(\mathbf{x}_i; \boldsymbol{\theta}) = c_{amb} + c_{diff} \max\{\mathbf{n}(\mathbf{x}_i) \cdot \mathbf{l}(\mathbf{x}_i), 0\}. \tag{20}$$

Here, $\mathbf{n}$ is the normalized surface normal of the object at a certain pixel position, while $\mathbf{l}$ is a normalized direction vector pointing from the surface point to the light source. The maximum function ensures that surface points facing away from the light source are not illuminated. For analysis of shading parameters, the setup of equations can be limited to surface points being known as illuminated or in shadow, thus making the non-linear maximum function unnecessary. The resulting photometric parameters are then given as

$$\boldsymbol{\theta}_p = [c_{amb}, c_{diff} n_x l_x, c_{diff} n_y l_y, c_{diff} n_z l_z]^T \tag{21}$$

and the individual components of $c_{diff} n_x l_x$ can be derived exploiting that both $\mathbf{n}$ and $\mathbf{l}$ have unit length. In the same way, higher order reflection models can be considered or even PCA spaces modelling pre-computed variations of a lightmap attached to the object surface as described in [Eisert (2003)].

4.3 Model smoothness

It is a reasonable a priori assumption for tracking deformable objects like cloth or tissue that the shape and its motion and deformation is smooth and continuous rather than rough and erratic. This assumption can be expressed by a *smoothness term*, i.e. a function that will associate a cost with undesired, non-smooth model states. Including this function in the optimization later on will lead to a preference of smooth results over rough ones. Algebraically, a smoothness term can also be interpreted as a regularization, especially in the case of the dense model where the data term leads to rank deficiency in the normal equations due to the aperture problem.

Often, smoothness is associated with a vanishing second derivative of a function. Hence, to force a model to be smooth, one can penalize the discrete second derivative of the motion parameters by applying a discrete Laplace operator [Wardetzky et al. (2007)]. The Laplace Matrix of a 2-dimensional grid is often defined as

$$\mathbf{L} = \mathbf{V} - \mathbf{A}, \tag{22}$$

where $\mathbf{V}$ is a diagonal matrix of the vertex valences and $\mathbf{A}$ is the adjacency matrix of the grid. Hence, the Laplace Matrix is based on the neighborhood definition in a grid and for a mesh with K vertices it is a $K \times K$ matrix with one row and one column per vertex. $\mathbf{L}_{ij} = -1$ if vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ are connected and $\mathbf{L}_{ii} = |\mathcal{N}_i|$, where the first subscript denotes the row and the second the column, respectively. $|\mathcal{N}_i|$ denotes the number of vertices in the neighborhood $\mathcal{N}_i$ of vertex $\mathbf{v}_i$. There exist several versions of a scaled Laplace matrix with entries $\mathbf{L}_{ij} = w_{ij}$ if vertices $\mathbf{v}_k$ and $\mathbf{v}_l$ are connected and $\mathbf{L}_{ii} = -1$ with different weighting schemes and different definitions of the neighborhood which influence the *smoothing behavior* of the Laplacian [Taubin (1995)]. The simplest one is a uniform scaling with

$$w_{ij} = \frac{1}{|\mathcal{N}_i|} \tag{23}$$

where $|\mathcal{N}_i|$ denotes the number of vertices in the neighborhood $\mathcal{N}_i$. In order to give closer neighbors a higher influence on vertex $\mathbf{v}_i$ than neighbors with a larger distance, we can also weight each neighbor according to its distance to vertex $\mathbf{v}_i$. In this case the weight is

$$w_{ij} = \frac{1/D_{ij}^{\text{Euc}}}{\sum_{n \in \mathcal{N}_i} 1/D_{in}^{\text{Euc}}} \tag{24}$$

where D_{ij}^{Euc} denotes the Euclidian distance between the two vertex positions $\mathbf{v}_i$ and $\mathbf{v}_j$.

The Laplacian can be applied to any parameter $\boldsymbol{\theta}$ associated with the mesh vertices. The distribution of these parameters over the mesh is perfectly smooth in the aforementioned sense if

$$\nabla\boldsymbol{\theta} = \mathbf{L} \cdot \boldsymbol{\theta} = \mathbf{0}$$

where $\boldsymbol{\theta}$ denotes the set of parameters $\theta_1 ... \theta_K$.

4.3.1 Laplacians and mesh boundaries

The Laplacian matrix explained above does not treat the mesh borders differently from the vertices inside the mesh. For example, the product of the Laplacian and the mesh vertex coordinates of an undeformed 2-dimensional mesh is not zero at the borders, $\|\mathbf{L} \cdot \mathbf{v}_x\| > 0, \|\mathbf{L} \cdot \mathbf{v}_y\| > 0$. This is due to the asymmetric vertex neighborhood at the mesh boundaries.

One solution is to build *Directional Laplacian matrices* $\mathbf{L}^d$ of the mesh which penalize the second derivatives in different directions, i.e. vertical and horizontal, separately. For each direction of vertex connections one Laplacian is built, with one row for each vertex that has two neighbors in the same direction. For example $\mathbf{L}^d_{ii} = 2$ and $\mathbf{L}^d_{ij} = -1$ if vertices $\mathbf{v}_k$ and $\mathbf{v}_l$ are connected in the direction d. Each row in $\mathbf{L}^d$ corresponds to one vertical vertex triple. The complete Laplacian is build by concatenating the *Directional Laplacian matrices* $\mathbf{L} = \left[\mathbf{L}^{d1} \ ... \ \mathbf{L}^{dn}\right]^T$. This type of Laplacians has several advantages. First, the product of this Laplacian with the mesh vertices of an undeformed mesh is zero as only symmetric neighborhoods, i.e. neighbors which have a *directional counterpart*, are taken into account. Second, by building separate Laplacians for each direction, we provide more constraints at the border vertices than one Laplacian for the full neighborhood.

5. Image-based optimization of the model parameters

Independent of the individual parameterization, the parameters $\boldsymbol{\theta}$ of the underlying model are estimated by minimizing a cost function:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} (\mathcal{E}_D(\boldsymbol{\theta}) + \lambda^2 \mathcal{E}_S(\boldsymbol{\theta})) \tag{25}$$

where $\mathcal{E}_D(\boldsymbol{\theta})$ is the *data term* and $\mathcal{E}_S(\boldsymbol{\theta})$ represents prior assumptions and is often called the *smoothness term*. We formulate both terms in a robust non-linear least-squares sense. This allows to minimize the cost function with common optimization algorithms such as Gauss-Newton (GN) or Levenberg-Marquardt (LM), i.e. by iteratively solving for a parameter update $\delta\hat{\boldsymbol{\theta}}$ and updating the parameter vector $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} + \delta\hat{\boldsymbol{\theta}}$. The optimization is performed hierarchically on an image pyramid where each level yields a more accurate parameter estimate. This hierarchical framework has several advantages. It speeds up the iteration time on lower and coarser levels and also allows us to cope with large displacements.

This section will concentrate on the formulation of the data and smoothness terms for different scenarios. For the data term, we exploit a relaxed version of a brightness constancy assumption accounting not only for geometric but also intensity differences between two images. The smoothness term in our framework makes use of a discrete formulation of the Laplacian defined on a 2-dimensional grid as formulated in section 4.3. We will describe the characteristics of different formulations of the Laplacian. Furthermore, we will cover numerical aspects for the estimation of the dense model.

5.1 Optimization framework

Both data term and smoothness term of equation (25) can be expressed in the following form:

$$\mathcal{E}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \rho(r_i(\boldsymbol{\theta})) \tag{26}$$

$\rho(r_i(\boldsymbol{\theta}))$ is a *norm-like function* [McCullagh & Nelder (1998); Wedderburn (1974)], i.e. a symmetric, positive-definite function with a unique minimum at zero, which is chosen to be less increasing than square. The r_i are the residuals of the particular term. The minimization is computed by an iteratively reweighted least-squares (IRLS) method which is known to be a Gauss-Newton type method for minimizing a sum of norm-like functions of the residuals [Huber (1981)]. In this scheme, the gradient and the (approximated) Hessian of the error function are given by

$$\begin{aligned} \mathbf{g}_{\mathcal{E}} &= \mathbf{r}^T \mathbf{W} \mathbf{J}_\mathbf{r} \\ \mathbf{H}_{\mathcal{E}} &\approx \mathbf{J}_\mathbf{r}{}^T \mathbf{W} \mathbf{J}_\mathbf{r} \end{aligned} \tag{27}$$

where $\mathbf{r} = \mathbf{r}(\boldsymbol{\theta})$, $\mathbf{J}_\mathbf{r} = \mathbf{J}_\mathbf{r}(\boldsymbol{\theta})$ and $\mathbf{W} = diag(w(r_i(\boldsymbol{\theta})))$ is a weight matrix computed in each iteration with

$$w(r_i) = \frac{1}{r_i} \frac{\partial \rho(r_i)}{\partial r_i} \tag{28}$$

An efficient and well known norm-like function is the Huber estimator [Huber (1981)]

$$\rho(r_i) = \begin{cases} \frac{1}{2} r_i^2 & if\ |r_i| \leq v \\ v_\mathrm{H} |r_i| - \frac{1}{2} v & \text{otherwise} \end{cases} \tag{29}$$

which is a parabola in the vicinity of zero, and increases linearly at a given level $|r_i| > v$. The weight function for the Huber kernel is given by

$$w(r_i) = \begin{cases} 1 & if\ |r_i| \leq v_\mathrm{H} \\ \frac{v_\mathrm{H}}{|r_i|} & \text{otherwise} \end{cases} \tag{30}$$

For $\rho(r_i) = \frac{1}{2} r_i^2$, the weight matrix is the identity and the method is equivalent to the (non-robust) Gauss Newton algorithm for non-linear least squares problems. In each iteration a parameter update is estimated by solving the *normal equations*

$$\delta \hat{\boldsymbol{\theta}} = -\left(\mathbf{J}_\mathbf{r}{}^T \mathbf{W} \mathbf{J}_\mathbf{r} \right)^{-1} \mathbf{r}^T \mathbf{W} \mathbf{J}_\mathbf{r} \tag{31}$$

The parameter is then updated by $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} + \delta \hat{\boldsymbol{\theta}}$.
For deformable surface tracking, we use the Huber kernel as a robust estimator for the data term and the two norm for the smoothness term. The matrices and vectors of our normal equations therefore are

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_\mathbf{r}(\hat{\boldsymbol{\theta}}) \\ \lambda \mathbf{J}_\mathbf{s}(\hat{\boldsymbol{\theta}}) \end{bmatrix}^T \quad \mathbf{b} = \begin{bmatrix} \mathbf{r}(\hat{\boldsymbol{\theta}}) \\ \mathbf{s}(\hat{\boldsymbol{\theta}}) \end{bmatrix}^T \quad \tilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \tag{32}$$

where $\mathbf{r}$ and $\mathbf{s}$ are the vectors of the residuals of the data and the smoothness term, respectively, which will be explained in detail in the following sections.

5.2 Data Term

The data term is based on a relaxed brightness constancy assumption. Methods exploiting the brightness constancy constraint assume that an image pixel $\mathbf{x}$ representing an object point does not change its brightness value between two successive frames $\mathcal{I}_{n-1}$ and $\mathcal{I}_n$:

$$\mathcal{I}_{n-1}\left(\psi_g(\mathbf{x}; \boldsymbol{\theta}_n)\right) = \mathcal{I}_n(\mathbf{x}) \tag{33}$$

However, this assumption is almost never valid for natural scenes. For this reason, we relax the optical flow constraint equation allowing for multiplicative deviations from brightness constancy:

$$\psi_p(\mathbf{x};\boldsymbol{\theta}_n) \cdot \mathcal{I}_{n-1}\left(\boldsymbol{\psi}_g(\mathbf{x};\boldsymbol{\theta}_n)\right) = \mathcal{I}_n\left(\mathbf{x}\right) \tag{34}$$

with a photometric warp $\psi_p(\mathbf{x};\boldsymbol{\theta}_n)$ and a geometric warp $\boldsymbol{\psi}_g(\mathbf{x};\boldsymbol{\theta}_n)$ as given in equation (5). Hence, the data term is formulated via the differences between the original frame $\mathcal{I}_n$ and the spatially and photometrically warped previous frame $\mathcal{I}_{n-1}$:

$$\mathcal{E}_D(\boldsymbol{\theta}) = \sum_{\mathbf{x}_i \in \mathcal{R}} \rho(\underbrace{\psi_p(\mathbf{x}_i;\boldsymbol{\theta}_{n-1}) \cdot \mathcal{I}_{n-1}\left(\boldsymbol{\psi}_g(\mathbf{x}_i;\boldsymbol{\theta}_{n-1})\right) - \mathcal{I}_n(\mathbf{x}_i)}_{r_i}) \tag{35}$$

where $\rho(r_i)$ is a norm-like function. Both warp functions are parameterized by the parameter vector $\boldsymbol{\theta}$ which comprises a geometric part $\boldsymbol{\theta}_g$ and a photometric part $\boldsymbol{\theta}_p$ as in equation (4). The Jacobian $\mathbf{J_r}$ of the residuals r_i is a $p \times N$ matrix (with p being the number of pixels in the region of interest) with the following i^{th} row:

$$\frac{\partial r_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \psi_p(\mathbf{x}_i;\boldsymbol{\theta}) \cdot \nabla\mathcal{I}_{n-1}(\boldsymbol{\psi}_g(\mathbf{x}_i;\boldsymbol{\theta})) \cdot \mathbf{J}_{\boldsymbol{\psi}_g}(\mathbf{x}_i;\boldsymbol{\theta}) + \mathcal{I}_{n-1}(\boldsymbol{\psi}_g(\mathbf{x}_i;\hat{\boldsymbol{\theta}})) \cdot \mathbf{J}_{\psi_p}(\mathbf{x}_i;\boldsymbol{\theta}) \tag{36}$$

with $\nabla\mathcal{I} = \left[\frac{\partial\mathcal{I}}{\partial x}\ \frac{\partial\mathcal{I}}{\partial y}\right]$. The Jacobians of the warps $\mathbf{J}_{\boldsymbol{\psi}_g}(\mathbf{x}_i;\boldsymbol{\theta})$ and $\mathbf{J}_{\boldsymbol{\psi}_g}(\mathbf{x}_i;\boldsymbol{\theta})$ depend on the warp parameterization. In general, it is a $2 \times N$ matrix where N is the number of parameters in the parameters vector θ as given in equation (6) or (19).

5.3 Smoothness term

The smoothness term is defined to regularize all parameters, geometric and photometric ones, using some suitably chosen Tikhonov regularizer

$$\mathcal{E}_S(\boldsymbol{\theta}) = \|\boldsymbol{\Gamma} \cdot \boldsymbol{\theta}\|^2 . \tag{37}$$

In the simplest case (zeroth-order Tikhonov regularization) $\boldsymbol{\Gamma}$ is the identity matrix $\boldsymbol{\Gamma} = \mathbf{I}$, giving preference to solutions with smaller norms. Other possible regularizations include first-order or second-order Tikhonov regularizations, where Γ approximates first- or second-order derivatives of the parameter, favoring *flat* or *smooth* results.

For the 2-dimensional models, the Laplacian matrix for 2-dimensional grids explained in section 4.3 define such an approximation of the second derivative of the mesh. As the parameter vector consists of three parts which should be regularized independently, the Tikhonov matrix is given by

$$\boldsymbol{\Gamma} = \begin{bmatrix} \mathbf{L} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \lambda_p \mathbf{L} \end{bmatrix} \tag{38}$$

A similar regularization matrix can be formulated for the 3-dimensional models. λ_p weights the smoothing of the geometric parameters (vertex displacements in $x-$ and y-direction) against the smoothing of the photometric scale. This is necessary due to the different scaling of the pixel displacement and the photometric parameter, the former being additive and the latter multiplicative. The Jacobian of the smoothness term is $\mathbf{J_s}(\boldsymbol{\theta}) = \boldsymbol{\Gamma}$.

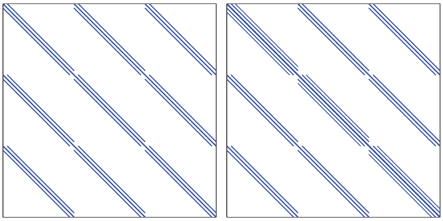


Fig. 2. Example structure of $\mathbf{H_r} \approx \mathbf{J_r}^T\mathbf{J_r}$ and $\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$ for an affine mesh-based model.

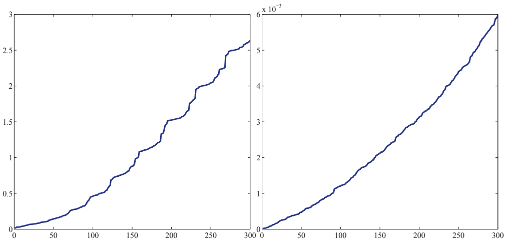


Fig. 3. Eigenvalues of the matrix with and without the smoothness term. Without the smoothness term the matrix is ill-conditioned.

By using a regularizer as given in equation (38), $\mathcal{E}_S(\boldsymbol{\theta})$ penalizes the discrete second derivative of the mesh and regularizes the flow field in addition to the motion model itself, especially in case of insufficient information in the data due to e.g. homogeneous regions with low image gradient. One important advantage of using direct image information instead of features is, that a lack of image information, i.e. image gradients of small magnitude in regions of little texture, automatically lead to a higher local weighting of the smoothness constraint in these regions. Also, the function dominates for vertices detected as outliers when using e.g. the Huber function for robust estimation.

5.4 Numerical issues

For all models presented in section 4 the approximation of the Hessian $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ is a sparse banded matrix (see Figure 2). The sparsity can be exploited to solve the resulting normal equations in each iteration, especially for the dense model where the matrix becomes very large but also very sparse: There are two unknowns per pixel, but there is no coupling of unknowns in the data term. This is in contrast to the affine warp, where all pixels under a certain mesh triangle simultaneously contribute to the six unknowns associated with the triangle's three vertices.

To solve the normal equations for the dense warp in an efficient way, multigrid algorithms can be used. The multigrid scheme exploits the fact that certain iterative solvers, e.g. the computationally inexpensive Gauss-Seidel method, reduce high frequency errors quickly despite of overall slow convergence. Therefore, these algorithms can be used as *smoothers*. After a few smoothing iterations the problem is subsampled to a lower resolution pixel grid and the process is restarted. Only at the bottom stage, the problem, which is now significantly reduced in size, is solved exactly. The exact solution is then propagated upwards to the higher resolution levels. The whole process is called a *V-cycle* in the multigrid literature.

5.5 Analysis by synthesis

Generally, intensity-based differential techniques, which estimate the motion only between two successive frames, often suffer from drift because they accumulate errors indefinitely. This limits their effectiveness when dealing with long video sequences. To avoid error accumulation we make use of an analysis-by-synthesis approach, where the error minimization is always carried out between a synthesized reference image and the actual camera frame. We use the previous parameter sets $\{\hat{\boldsymbol{\theta}}_1, ...\hat{\boldsymbol{\theta}}_{n-1}\}$ to generate a synthetic version of the previous frame $\mathcal{I}_{n-1}$ from a model image $\mathcal{I}_0$. The new parameters $\hat{\boldsymbol{\theta}}_n$ are then estimated from this synthetic previous frame $\hat{\mathcal{I}}_{n-1}$ to the current camera frame $\mathcal{I}_n$ (see Figure 4). This

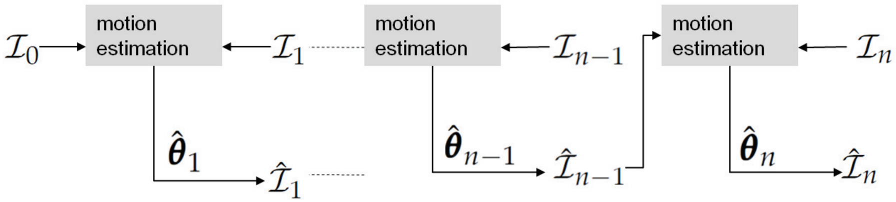


Fig. 4. Analysis by Synthesis framework

way, the model frame serves as reference frame and we assure that no misalignment of the model and the previous frame occurs. Thereby, we allow for recovery from small inaccuracies during parameter estimation.

6. Experimental results

Our approach has been applied to several tracking problems, such as tracking cloth in augmented reality applications, medical image analysis, and facial expression analysis. The different applications will be explained in detail in section 6.3. Section 6.1 and 6.2 focus on experiments on the registration accuracy achieved with our framework and a mesh-based 2-dimensional model as well as on choosing the regularization parameter in the optimization framework.

6.1 Registration accuracy

We evaluated the registration results of real image sequences based on the Root Mean Squared Error (RMSE) between the synthetic image $\hat{\mathcal{I}}_n$ generated from the parameter estimates $\hat{\boldsymbol{\theta}}_n$ and the original current frame $\mathcal{I}_n$ computed over all image pixels in the mesh region $\mathcal{R}$ for several video sequences and compared our approach with the classical optical flow approach. With classical optical flow approach we refer to the original optical flow constraint that does not account for illumination changes, the geometric deformation model and optimization method are equal. Experiments with nine sequences showed that taking illumination parameters into account significantly improves the spatial tracking results. The table in Figure 5 sums up the mean RSME values over the entire sequence for these nine sequences. It shows that taking illumination parameters into account significantly reduces the mean RMSE over the entire sequence by up to 74%. The right image shows the RSME plot of one of the sequences and compares the results achieved with our approach (black solid line) to the classical optical flow method (dashed red line). Additionally, we manually labeled prominent feature points in every 50th frame of two test sequences which serve as ground truth points. We then warped the ground truth points of the reference frame with the geometric deformation parameters of these frames. The mean difference between the estimated positions and the manually labeled ground truth position describes the geometric registration error. This additional evaluation approach is chosen to evaluate geometric registration accuracy separately from photometric registration. We can reduce the mean distance between the estimated and the ground truth position by approximately 40% when taking illumination into account. In all these experiments we used a 2-dimensional mesh-based affine motion model and a regularization parameter $\lambda = 0.5$.

Sequence	classical OF	our approach	%
Picasso	0.0306	0.0204	33.33%
Flower1	0.1364	0.0415	69.57%
Flower2	0.1245	0.0376	69.80%
Art	0.1016	0.0258	74.61%
Flower3	0.1286	0.0385	70.06%
Shirt	0.0630	0.0405	35.71%
Pattern	0.0632	0.0213	66.30%
Cloth1	0.0877	0.0443	49.49%
Cloth2	0.0976	0.0513	47.44%

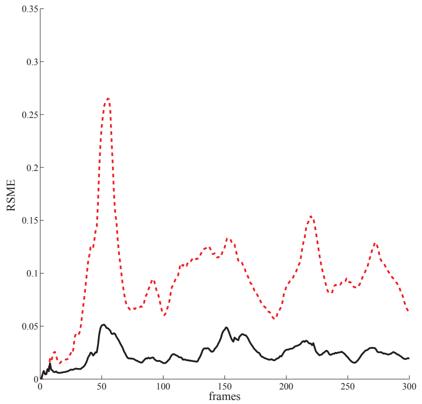


Fig. 5. Left: Comparison of the average RMSE over the entire video sequence with our approach and classical optical flow. Right: Plots of the RMSE between the synthetic frame $\hat{\mathcal{I}}_n$ and the original current frame $\mathcal{I}_n$ with classical optical flow (dashed red) and our method (solid black) for an example sequence.

6.2 Choosing the regularization parameter

Choosing the regularization parameter λ in the optimization framework is not an easy task. The choice of λ describes a trade-off between fitting to the data term –which may be corrupted by noise– and the *smoothness* of the result. Hence, there is no correct λ as the trade-off depends on the considered problem. We can only define a *reasonable* λ, i.e. a λ which represents the best balance between both sides. The influence of the regularization parameter λ can be analyzed using the L-curve [Hansen (1992)], which plots $\mathcal{E}_D(\hat{\boldsymbol{\theta}})$ against $\mathcal{E}_S(\hat{\boldsymbol{\theta}})$ for different values of λ. The L-curve (see Figure 7) is basically made up of two parts which correspond to *oversmoothed* and *undersmoothed* solutions. The more horizontal part corresponds to the solution where the regularization parameter is very large and the solution is dominated by the regularization errors. The more vertical part corresponds to solutions where the regularization parameter is very small and the solution is dominated by the data error. Often, a *reasonable* λ, i.e. a good trade-off between fitting to the data and smoothness term, is considered to be the point of maximal curvature of the L-shaped curve or the point nearest to the origin. Figure 6 shows the influence of different choices for λ. It shows examples of difference images before optimization (left) and for different values of λ. If λ is chosen too small, the result is dominated by the (noise corrupted) data term and if λ is chosen too small, the result is too smooth to be fitted to the data term. Figure 7 shows a typical L-Curve and associated values for the data and smoothness terms for different values for λ.

6.3 Applications

We applied the proposed framework to different fields of applications, presented in the following.

Augmented reality

The proposed method was used for augmented reality applications where a deformable surface, like a piece of paper or cloth, is retextured in a monocular image sequence. The

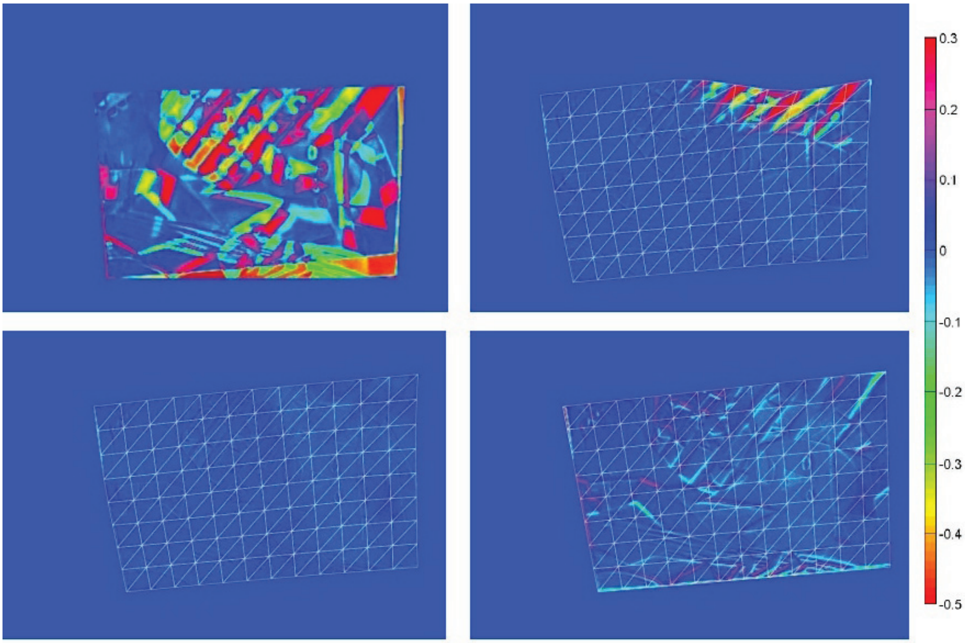
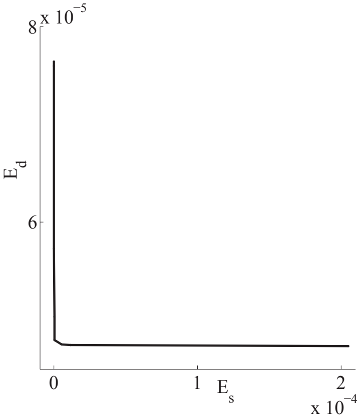


Fig. 6. Difference image between the model frame and an example current frame before motion estimation (left). Difference images after optimization with very small λ (undersmoothed), *reasonable* λ and very large λ (oversmoothed).



λ	$\mathcal{E}_D/10^{-5}$	$\mathcal{E}_S/10^{-5}$
0.01	4.7357	20.535
0.1	4.7449	1.1588
0.2	4.7494	0.5137
1	4.7957	0.0285
5	5.0331	0.0106
10	5.7310	0.0043
100	7.5944	0.0000
1000	7.6385	0.0000

Fig. 7. A typical L-curve and associated values for $\mathcal{E}_D$ and $\mathcal{E}_S$ for different values for λ

intention is to blend a virtual texture into a real video sequence such that its deformation as well as lighting conditions and shading in the final image remain the same (see e.g. Figure 1). We recover geometric and photometric parameters that describe the 2-dimensional

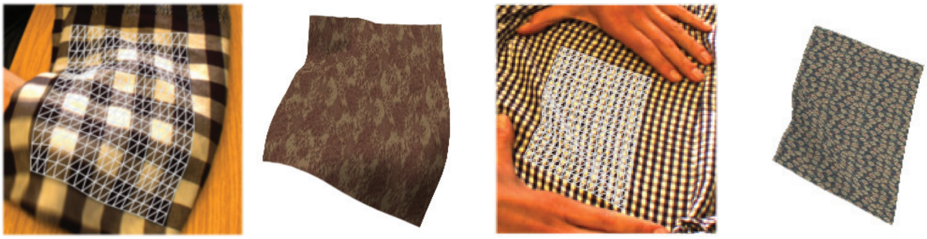


Fig. 8. Tracking and synthesizing different kinds of cloth. The left example shows thick cloth with very smooth deformations while the right example shows cloth that produces small crinkles and creases.

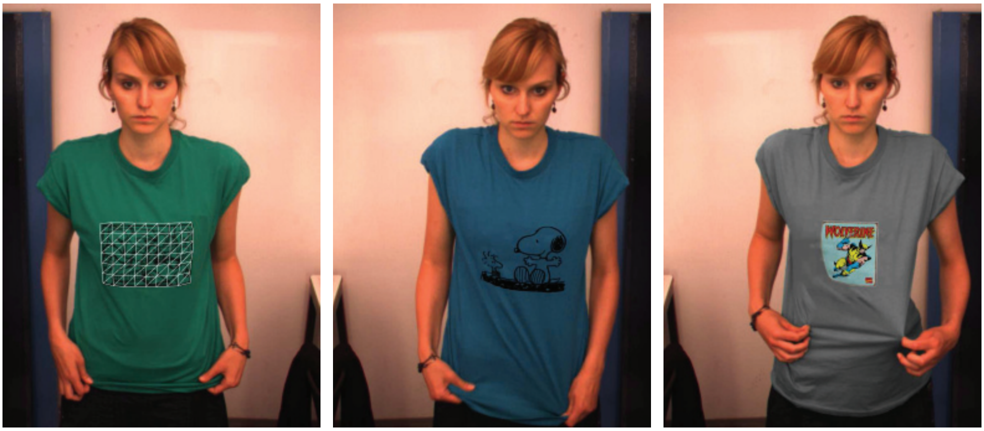


Fig. 9. Tracking and retexturing clothes in an augmented reality application (left: original frame with mesh)

deformation and shading properties of the projected surface in the image plane with the proposed method. The virtual texture is then deformed with the estimated deformation parameters and blended into the original video sequence. A shading map estimated from the photometric parameters is applied to the virtual texture increasing the realistic impression of the augmented video sequence.

Figure 1, 8, and 9 show augmentation results under different lighting conditions using the estimated illumination parameters to establish a shading map. These examples demonstrate how crucial illumination recovery is for convincing texture augmentation of deforming surfaces. The addition of realistic lighting increases the perception of spatial relations between the real and virtual objects. Note, that spatial deformation is purely 2D and the 3-dimensional impression comes from shading. The results from Figure 9 are images from a real-time augmented reality application which we built. In this Virtual Mirror a user is captured with a camera and a rectangular pattern on his shirt is retextured with a user-defined logo or image with correct deformation and illumination. The new logo follows the users movements as if attached to his shirt. The system runs at 25 fps and the images have a resolution of 768 × 1024 pixels. To this end we use 4 levels of resolution and experiments with synthetic image sequences with this hierarchical scheme showed that it is able to estimate displacements of up to 25 pixels between two frames with a mean error of 0.2 pixels. On a 2.4 GHz Pentium 4 based system the parameter estimation takes about 40 ms, leading to a frame rate of about 25 fps.

Medical imaging

Our framework is used to stabilize kymographic images. Kymographic imaging is a method for visualizing and analyzing motion. In our use case, it is the vibration of the human vocal folds that is to be visualized. The source material are endoscopic video recordings showing the vocal folds while the patient tries to produce a sound at a certain frequency. Instead of analyzing video images of the whole moving vocal folds, a single line of each frame in the recorded image sequence is extracted. The lines are combined to create a kymogram which is then analysed by the physician. A kymogram is a *time slice* of the vibratory behaviour of the vocal folds, i.e. an X-t-image rather than an X-Y image. Camera movement and motion of the patient as a whole disturb the creation of the kymographic images which relies on the assumption that a certain scanline of the frames displays roughly the same part of the vocal fold throughout the entire endoscopic video sequence. Therefore, kymographic imaging can be greatly improved by compensating camera motion.

We use the methods described in the chapter to compute a deformation field between successive frames in the endoscopic video sequence. The field is computed with the mesh-based affine model. The stiffness, i.e. the weighting of the Laplacian smoothness term, is chosen such that the deformation field adapts to global image motion due to camera movement as well as to the parallax motion induced by different depth layers in the scene. It is set too stiff, however, to compensate for the motion of the vocal folds, which must not be altered by the image stabilization. From the deformation fields, the global motion of a region of interest, which is annotated manually in the first frame, is tracked throughout the sequence. A stabilizing transformation for each frame is found by estimating a rigid transformation that maps the region of interest to its correspondence in the first frame. Figure 10 illustrates the vast increase in kymogram quality achieved by the motion compensation.

Our dense, warp-based approach to image stabilization has several advantages over the classic approach based on feature tracking. Our approach allows us to optimize for a dense motion field using all available image information. This is particularly effective for images of low quality. The endoscopic video we stabilize, for example, suffers from artifacts due to interlacing, image noise, occasional color artifacts and motion blur. We found simple feature detectors such as Harris corners virtually unusable on this material. Furthermore, the optimization approach gives a complete and accurate estimate of nonrigid image motion at a certain scale as the result of a single optimization process. Outliers in the data term are handled during the optimization by the robust error metric (in this case, the Huber function) and there is no need to perform RANSAC or other data selection schemes. Finally, the density of the mesh and the weight of the stiffness term allows us to regulate precisely to which scale of nonrigid motion the algorithm adapts.

Facial Expression Analysis

The human face is a well known example for a deformable object. Head pose can be described by rigid body transforms whereas facial expressions lead to local surface deformations. These deformations, however, are constrained and depend on face muscle and soft tissue properties. Analyses have shown [Ekman & Friesen (1978)], that there are only about 50 groups of muscles in the face, that can be controlled independently when performing facial expressions.

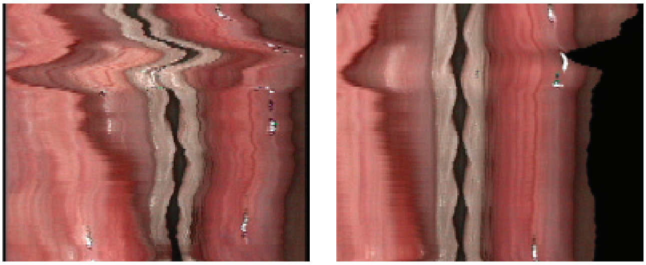


Fig. 10. Kymography: uncompensated and compensated kymographic images.

Therefore, surface deformations as specified in (12) can compactly be described by a limited number of facial expression parameters.

We have applied the framework described in this chapter to the estimation of facial expression parameters [Eisert (2003); Eisert & Girod (1998)]. The geometry of a head is represented by a parameterized generic triangle mesh model, which is individualized by applying surface deformations represented by shape parameters to a standard face geometry. The model is fit to the first frame of a video sequence, which is also projected onto the 3D model as texture. For the description of facial expressions, we have adopted a subset of the MPEG-4 Facial Animation Parameters (FAPs according to [MPG (1999)]), which describe facial expressions by a superposition of simple local surface deformations related to movements of lips, chin, cheeks, etc. In our experiments, we use 21 facial expression parameters $\boldsymbol{\theta}_d$ together with 6 pose parameters to specify surface deformation and orientation according to (14). The mapping from FAPs to arbitrary surface points deformation given by $\mathbf{D}_d$ is modelled using triangular B-splines.

Given the three-dimensional model description, the facial expression parameters are jointly estimated with the global pose parameters in an analysis by synthesis frame work as described in section 5.5. The per-pixel depth information required in (19) as well as the surface normal data are rendered and read back from the graphics card. Smoothness terms penalize FAP differences between the left and right side of the face, favoring symmetric facial expressions. In addition to the geometric deformations, parameters related to illumination are estimated to compensate for lighting changes between the model created for the first frame and the current scene. A simple model with ambient and diffuse colored light is used to describe the photometric changes. Figure 11 gives examples for the facial expression analysis of the proposed framework. From the original frames in the leftmost column, surface deformations are estimated. The deformed model is shown by means of a wireframe and textured representation in the middle columns. The facial animation parameters can also be applied to other face models, implementing expression cloning as illustrated in the rightmost columns of figure 11.

7. Conclusion

We presented a complete framework for tracking of deformable surfaces using image-based optimization methods. We use a relaxed brightness constancy assumption and model both geometrical as well as photometrical changes in an image sequence. In our framework formulation we separated the motion model from the optimization method which makes

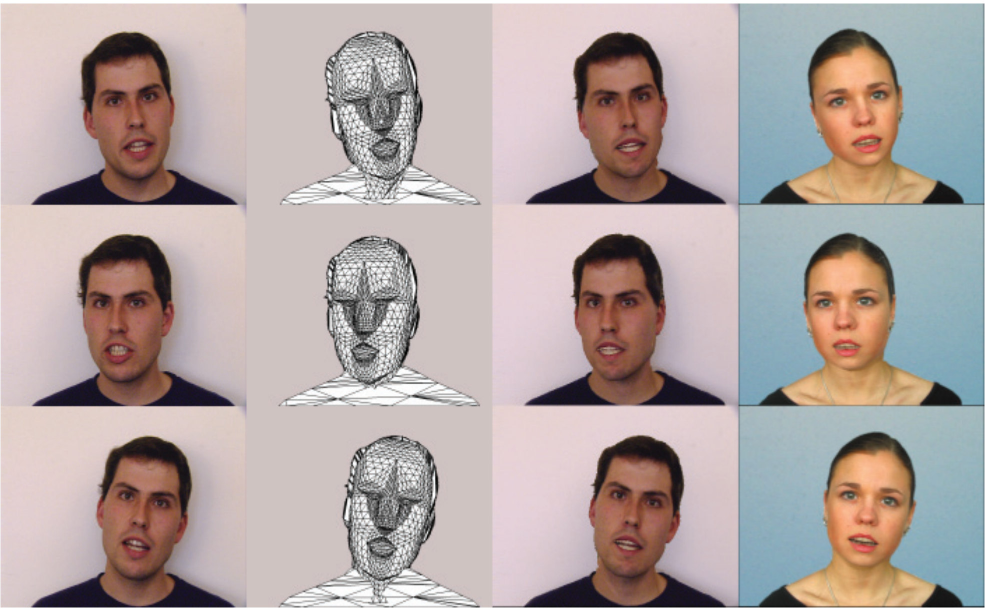


Fig. 11. 3D facial expression analysis. The left colum shows example frames of the original sequence. Facial expression parameters of a deformable face model are estimated. The second column shows the animated mesh, while the third column depicts the textured model. The rightmost column refers to expression cloning where the deformation parameters are mapped onto a different person's model.

it highly adaptive to a broad variety of motion estimation problems, as the motion model can be formulated individuallyfor the given problem. This is shown by a broad variety of applications where we applied the proposed framework.

8. References

Bartoli, A. (2008). Groupwise geometric and photometric direct image registration, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 30(12): 1098–2108.

Bartoli, A. & Zisserman, A. (2004). Direct estimation of non-rigid registrations, *Proc. British Machine Vision Conf. (BMVC 2004)*, London, UK.

Bay, H., Tuytelaars, T. & Gool, L. V. (2006). Surf: Speeded up robust features, *In ECCV*, pp. 404–417.

Belongie, S., Malik, J. & Puzicha, J. (2002). Shape matching and object recognition using shape contexts, *IEEE Trans. Pattern Analysis and Machine Intelligence* 24(4): 509–522.

Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations, *IEEE Trans. Pattern Analysis and Machine Intelligence* 11(6): 567–585.

Chen, C. (1993). Improved Moments Invariants for Shape Discrimination, *Pattern Recognition* 26(5): 683–686.

Eisert, P. (2003). MPEG-4 facial animation in video analysis and synthesis, *International Journal of Imaging Systems and Technology* 13(5): 245–256.

Eisert, P. & Girod, B. (1998). Analyzing facial expressions for virtual conferencing, *IEEE Computer Graphics and Applications* 18(5): 70–78.

Ekman, P. & Friesen, W. V. (1978). *Facial Action Coding System*, Consulting Psychologists Press, Inc., Palo Alto.

Gay Bellile, V., Bartoli, A. & Sayd, P. (2007). Direct estimation of non-rigid registrations with image-based self-occlusion reasoning, *Proc. Int. Conf on Computer Vision (ICCV 2007)*, pp. 1–6.

Gennert, M. A. & Negahdaripour, S. (1987). Relaxing the brightness constancy assumption in computing optical flow, *Technical report*, Cambridge, MA, USA.

Hansen, P. (1992). The use of the l-curve in the regularization of discrete ill-posed problems, *SIAM Journ. Sci. Comp.* 34: 561–580.

Harris, C. & Stephens, M. (1988). A combined corner and edge detection, *Proc. 4th Alvey Vision Conference*, pp. 147–151.

Haussecker, H. & Fleet, D. (2001). Computing optical flow with physical models of brightness variation, Vol. 23, Washington, DC, USA, pp. 661–673.

Hilsmann, A. & Eisert, P. (2008). Tracking deformable surfaces with optical flow in the presence of self occlusions in monocular image sequences, *CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, Anchorage, USA.

Hilsmann, A. & Eisert, P. (2009). Joint estimation of deformable motion and photometric parameters in single view video, *ICCV 2009 Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, Kyoto, Japan, pp. 1–6.

Hilsmann, A., Schneider, D. & Eisert, P. (2010). Realistic cloth augmentation in single view video under occlusions, *Computers & Graphics* .

Huber, P. (1981). *Robust Statistics*, John Wiley & Sons,.

Lim, J. & Yang, M.-H. (2005). A direct method for modeling non-rigid motion with thin plate spline, *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR 2005)*, pp. 1196–1202.

Lowe, D. (2003). Distinctive Image Features from Scale-Invariant Keypoints, *Int. Journal of Computer Vision* 60(2): 91–110.

McCullagh, P. & Nelder, J. A. (1998). *Generalized linear models*, Chapman & Hall, London.

Metaxas, D. N. (1996). *Physics-Based Deformable Models: Applications to Computer Vision, Graphics, and Medical Imaging*, Kluwer Academic Publishers, Norwell, MA, USA.

MPG (1999). *ISO/IEC 14496-2: Coding of audio-visual objects - Part 2: Visual, (MPEG-4 visual).*

Nastar, C., Moghaddam, B. & Pentland, A. (1996). Generalized image matching: Statistical learning of physically based deformations, pp. 589–598.

Ostermann, J. (1994). Object-oriented analysis-synthesis Coding (OOASC) based on the source model of moving flexible 3D-objects, *IEEE Trans. on Image Processing* 3(5).

Pilet, J., Lepetit, V. & Fua, P. (2008). Fast non-rigid surface detection, registration and realistic augmentation, *Int. Journal of Computer Vision* 76(2): 109–122.

Pizarro, D. & Bartoli, A. (2007). Shadow resistant direct image registration, *Proc. of the 15th Scandinavian Conference on Image Analysis (SCIA 2007)*, pp. 928–937.

Salzmann, M., Pilet, J., Ilic, S. & Fua, P. (2007). Surface deformation models for nonrigid 3d shape recovery, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 29(8): 1481–1487.

Scholz, V. & Magnor, M. (2006). Texture replacement of garments in monocular video sequences, *Rendering Techniques 2006: Eurographics Symposium on Rendering*, pp. 305–312.

Silveira, G. & Malis, E. (2007). Real-time visual tracking under arbitrary illumination changes, *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR 2007)*, pp. 1–6.

Taubin, G. (1995). A signal processing approach to fair surface design, *Proc. of ACM SIGGRAPH 1995*, pp. 351–358.

Terzopoulos, D. & Metaxas, D. (1991). Dynamic 3d models with local and global deformations: deformable superquadrics, *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* 13(7): 703–714.

Thirion, J. (1996). New Feature Points Based on Geometric Invariants for 3D Image Registration, *Int. Journal of Computer Vision* 18(2): 121–137.

Torresani, L., Yang, D., Alexander, E. & Bregler, C. (2001). Tracking and modeling non-rigid objects with rank constraints, *Proc. of Int. Conf on Computer Vision and Pattern Recognition, CVPR*.

Wardetzky, M., Mathur, S., Kälberer, F. & Grinspun, E. (2007). Discrete laplace operators: No free lunch, *Proc. of 5th Eurographics Symposium on Geometry Processing*, Aire-la-Ville, Switzerland, pp. 33–37.

Wedderburn, R. W. M. (1974). Quasi-likelihood functions, generalized linear models, and the gauss-newton method, *Biometrika* 61(3): 439–447.

White, R. & Forsyth, D. A. (2006). Retexturing single views using texture and shading., *Proc. European Conf. on Computer Vision (ECCV 2006)*, pp. 70–81.

14

Tracking of Moving Coronary Artery Segment in Sequence of X-Ray Images

Goszczynska Hanna and Kowalczyk Leszek
Nałęcz Institute of Biocybernetics and Biomedical Engineering PAS
Poland

1. Introduction

The variety of motion artefacts like situations when the patient is not able to hold his or her breath, natural organ movements and others cause great problems in current medical imaging techniques, such as image subtraction technique (eg. Digital Subtraction Angiography), registration technique, morphological and functional analysis, e.g. flow or perfusion estimation, and support of therapeutic intervention, eg. surgical tools and organ movement tracking. There are numerous reports in the literature mentioning possible types of motion artefacts. Many techniques are described to avoid and to correct such artefacts by means of digital image processing. The most important tasks utilized in all these techniques are the construction of a 2D geometrical transformation, which refers to original 3D deformation, finding correspondence between two images using only grey-level information, and efficient computer application of this correspondence [Meijering, 2000; Wang et al., 1999].

The procedure of finding the correspondence or estimation of the level of similarity between two images is the most important part of the above mentioned techniques, which has been applied either in static or dynamic approaches. This work presents an application of a correspondence finding technique to the densitometric analysis performed for coronary flow estimation on the image of a moving artery segment in X-ray image sequences of coronary arteries (Fig. 1).

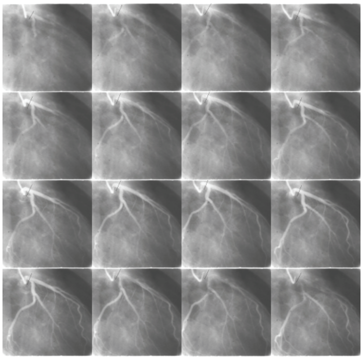


Fig. 1. Sequence of coronarographic images (every 6th image) from negative coronarographic film with visible contrast medium passing

Coronary image sequences are collected during coronarographic examination with X-ray contrast medium injection into coronary arteries for visualization purposes. Based on coronarographic sequences, both the morphometry of the arteries and the velocity of washing out of the indicator can be assessed [Nissen & Gurley, 1991; Goszczyńska & Rewicki, 2008].

For densitometric analysis the measurements of the image gray levels of the pixels forming a measuring window (called the cross-sectional line) are performed for each image of a coronarographic sequence. The measurements are performed along the same cross-sectional line with its adjacent background. Fig. 2 shows the cross-sectional line positioned on the left coronary artery (LCA).

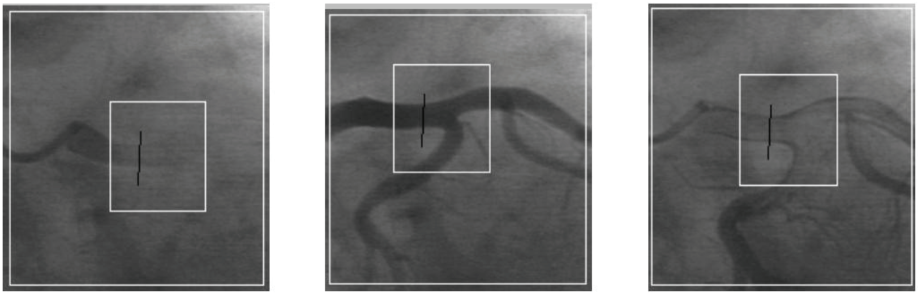
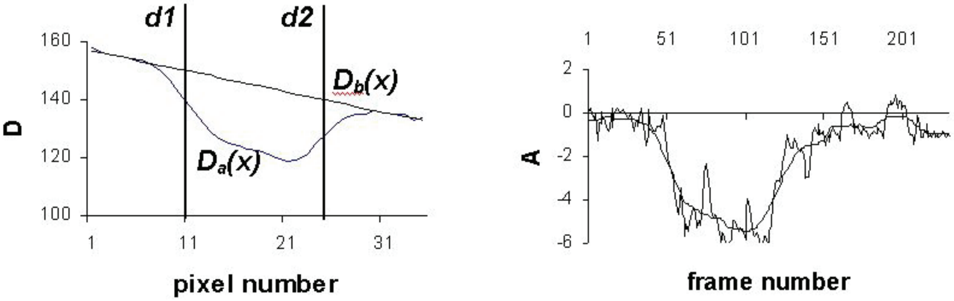


Fig. 2. Fragments of images of the left coronary artery (LCA) from a coronarographic sequence during injection, filling and washing out of a contrast medium



a) b)

Fig. 3. Densitometric measurements performed on the segment of LCA: a) brightness profile $D_a(x)$ along the artery cross-sectional line and the background estimation line $D_b(x)$, $d = d_2$-d_1, which is the inner artery diameter, b) densitometric curve $A(t)$ and its low pass version

Figure 3a shows the brightness curve $D_a(x)$ along the cross-sectional line (the measuring window) and the approximated background image intensity line $D_b(x)$, which is a straight line passing through two interpolation nodes [Goszczynska, 2005].

The $D_a(x)$ and $D_b(x)$ curves make it possible to calculate the $A(t)$ curve (with a temporal resolution equal to the image acquisition rate) proportional to the amount of the indicator present under the measuring window (i.e. segment d1-d2, which is the inner diameter of the artery) at time t:

$$A(t) = \int_{x=d_1}^{d_2} \ln\left(\frac{D_a(x,t)}{D_b(x,t)}\right) dx \tag{1}$$

where x is the coordinate within the cross-sectional line. Fig. 3b presents an example of the densitometric curve *A(t)* for a sequence of coronarographic images.

Densitometric curve *A(t)* is a basis for coronary flow estimation in indicator dilution method (Appendix A). In coronary flow estimation method based on indicator dilution presented in [Goszczynska & Rewicki, 2008] the number of frames needed for the calculation of the densitometric curve *A(t)* depends on contrast medium injection parameters and the coronary flow rate, being usually larger than 100. Sections 1.1 covers problems concerning data collection from coronarographic image sequences. In Section 1.2 the techniques of object movement estimation in image sequences are briefly described. Section 2 presents a proposed method of automatic tracking of an artery segment and Section 3 brings the results obtained with this method. The error of the method is estimated in Section 4, and other solutions for the problem of artery movement tracing are mentioned in Section 5.

1.1 Issues related to automatic drawing of the cross-sectional line

The estimation of coronary blood flow, besides the difficulties concerning densitometric measurements, is affected by an additional problem of the cyclic movement of an artery segment or part of the myocardium. In the following sections we describe some issues related to the automatic and manual drawing of the artery cross-sectional line. In the manual case, the operator marks a cross-sectional line at the same point of the artery on each frame of the sequence. Unfortunately the phrase “the same point” is not precise due to at least two reasons:

- the myocardium movement results in the changes of the location and shape of the particular part of the artery which makes ensuring that the cross-sectional line on the frame is in the same place as in the previous frame difficult,
- the subjective nature of manual positioning of the same point, in the absence of some characteristic points, is very error-prone, even when the location and the shape of the object remain unchanged.

The above issues suggest that the process of drawing the cross-sectional line should be automated. The manual solution, besides being inaccurate, is also very exhaustive and boring for the operator - the cross-sectional line has to drawn on at least hundred of consecutive frames for a single measurement.

For a clinical study of the flow measurement method presented in [Goszczynska & Rewicki, 2008; Goszczynska, 2008] the analysis of more than ten thousand images was required, and that is why certain type of automatic aid for cross-sectional line positioning has been strongly suggested.

The algorithms performing the task mentioned above are described as movement correction (the 3D transformation of each frame in sequence resulting in apparent lack of heart movement) or tracing the object (relocation of the ROI, so that the particular detail of the heart image – the artery fragment – is traced). The movement correction algorithms may be applied to local areas of the whole frame. For heart images processing movement correction for the whole frame is used to reduce the distortions made by patient movement. This makes it impossible to precisely determine a mask frame within the DSA technique [Meijering, 1999; Meijering, 2000].

The crucial difficulty preventing an effective incorporation of those algorithms to coronarographic frames analysis is a low densitometric contrast of the heart areas of interest. As an example an artery fragment image can be considered, across the whole sequence of the frames, also covering frames before contrast medium injection, and during contrast medium washing. In the instances when the area of interest covers only a part of the myocardium with tiny arteries, the frame analysis results in the analysis of the areas with nearly homogenous brightness, similar to the surrounding area. In such situation the movement detection can be performed indirectly, by tracing the movement of the nearest artery segment and applying a 2D interpolation.

In the analysis of movement in frames containing the image of clearly distinguishable arteries another difficulty appears. Local examination of continuous movement for contour in the plane perpendicular to the image detection plane may be hardly possible. The local movement analysis of a continuous contour had first been described by Stumpf in 1911 [Hildreth, 1987]. Obviously the observed movement of an artery using 2D systems lacks component of the real movement perpendicular to the image plane (the observed movement is barely visible). Computing any corrections of the event on the basis of only single plane images is a difficult task.

1.2 Movement estimation techniques

Artery segmentation and/or contour and central line detection methods are frequently implemented as a preliminary, but not mandatory step, in the movement detection algorithms described in literature [Coatrieux, 1995; Rus, 1995; Benayoun & Ayache, 1998; Konrad, 2000]. These methods can be divided into two categories:

- those limited to the area being a subject of segmentation only, tracing the artery assuming the continuity of the arteries tree structure, and
- those analyzing the complete image and sharpening the structures investigated, so that the background can be removed.

The first step of the analysis utilizing the methods of the first category is manual positioning of the point of the center line or on the contour of the element investigated. Then the operator proceeds in certain direction according to some criteria, and positions the next points of the line or contour. An example of such a criterion may be the maximum gradient of the grayscale.

The methods of the second category are generally two types of filtration - linear and nonlinear. The linear filters utilized for contour detection are based on testing two-dimensional derivatives of the first degree (Sobel or Roberts operators) or the second degree (Marr-Hildreth and Laplace operators) for the brightness change function within the selected neighborhood of an indicated point or, in a more general way, on the operation analysis between the defined operator mask and the neighborhood of a certain point. The adaptive linear filters, meaning those in which a mask is adapted to the shape of the object tested, maximize the output value for the central line of the object. The morphological filters belonging to the nonlinear filter class are based on the comparison of the object tested with a structuring element. For segmentation purposes, opening operation i.e. erosion and dilation is usually used. Some other methods may be mentioned here like image decomposition by eigenvalues, a curvature analysis, vector detection of the central line and the method of geometric moments.

The non-subtractive images, in opposite to subtractive ones are distinguished by high non-homogeneity of the background, which causes the artery segmentation to be a very difficult task. Due to this difficulty the main criterion for estimation of the results provided by filters is their ability to equalize the background level. The artery segmentation phase, setting their contours and central lines increases the effectiveness of movement detection methods implemented in the solutions, which are more sensitive to the structure and less sensitive to the brightness. Unfortunately it also increases the time of the movement analysis process.

The local movement detection methods or movement measurement methods in images may be generally divided into two categories: an optical flow procedure, based on the brightness gradient analysis, and a template matching procedure, based on image similarity measures.

The first procedure, suggested by Horn and Schnuck [Horn, 1981], is based on the continuity principle, also known in physics, in the movement analysis of the frame sequence is described by the formula:

$$\frac{\partial I}{\partial t} + v_x \frac{\partial I}{\partial x} + v_y \frac{\partial I}{\partial y} = 0 \tag{2}$$

where I is the brightness of the pixel tested, v_x is the x component of the tested pixel velocity, and v_y is the y component of the pixel tested velocity. The pixel velocity is determined according to the spatial and time resolution of the image tested sequence. By such processing we obtain the estimation of the apparent movement at every point of the frame. Application of differential techniques based on this method results in a relatively fast algorithm. There are two important conditions (considered disadvantages) that have to be met: the brightness of the moving object should be constant in time and the movement of the object between two consecutive frames should be relatively small. In the case of arteries, both of the mentioned conditions are not fulfilled for the entire frame sequence. Based on some presumptions, this technique may provide acceptable results for the estimation of the general myocardium shape change during its working cycle [Meunier et al, 1989].

In the next chapter we will describe a method of automatic movement trajectory estimation for an artery segment based on template matching.

2. Method

In the frame sequence studied for marked artery segment three stages can be indicated:

- before contrast medium appearance - T_a,
- with contrast medium filling - T_c and
- the contrast medium wash-out period - T_p.

The movement analysis of a fragment of the artery under investigation was divided into three steps:

- movement estimation for the marked fragment of the artery during contrast medium filling period T_c,
- correction of faulty detections,
- movement estimation in stages T_a and T_p.

2.1 Estimation of movement of an artery segment in the indicator filling period T_c

Below a method of automatic data collection based on a popular block-matching technique is described [Gonzalez & Wintz, 1987; Rong, 1989; Maragos & Pessoa, 2000; Buzug, 1998].

This method was used in the similarity analysis of a few selected fragments of original frames sequence (without performing any pre-processing) with a chosen template, i.e. the fragment of an artery as it was shown on the frame where the arteries are clearly visible (which means the end of the contrast medium injection phase). The sum of difference squares C_{SSD} was used (*R, S* denotes the compared fragments of the images in the area of *m, n* dimensions):

$$C_{SSD} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(R(i,j) - S(i,j))^2 \tag{3}$$

In order to find the area in frame *I(t+1)* which is similar to the given area *R* in frame *I(t)*, the area *S* is defined in frame *I(t+1)* with the same coordinates and size as the area *R* in frame *I(t)*. The next step is the area enlargement by values k_{max} and l_{max} which are greater or equal to the maximum movement of the selected fragment between the two frames (Fig. 4).

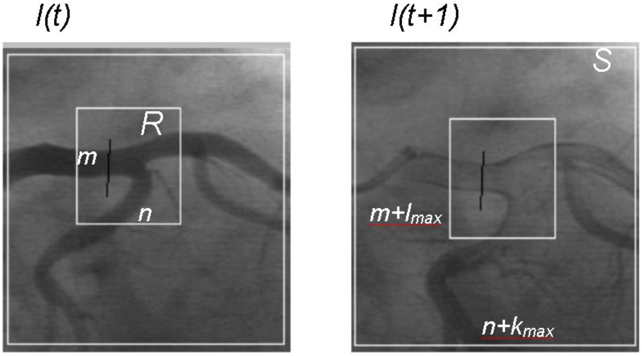


Fig. 4. Determining the template and investigation windows

The search for the area in frame *I(t+1)* which is most similar to the chosen area *R* in frame *I(t)* involves moving the reference area *R* on the magnified area *S* and computing the coefficient of mutual correlation ($0 \le k \le k_{max}$, $0 \le l \le l_{max}$):

$$C(k,l) = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} R(i,j) * S(i+k, j+l)}{\sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} R^2(i,j) * \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} S^2(i+k, j+l)}} \tag{4}$$

The maximum value C_{max} of this coefficient tells the location of the area most similar to the reference area. The indexes *k* and *l* corresponding to the maximum value of the coefficient determine the coordinates of the area investigated (Fig. 4).

Other measures of similarity have been also proposed:

- variance normalized correlation:

$$C = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(R(i,j) - I_R) * (S(i,j) - I_S)}{\sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(R(i,j) - I_R)^2 * \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(S(i,j) - I_S)^2}} \tag{5}$$

- the sum of absolute differences C_{SDA} :

$$C_{SDA} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} | R(i,j) - S(i,j) | \tag{6}$$

This measure refers to a nonlinear signal correlation. As the following equations is true $|a-b|=a+b-2min(a,b)$, then under some conditions (i.e. when the correlation value is normalized by dividing it by the average brightness value for windows *R* and *S*), the minimum value of C_{SDA} is equal to the maximum value of the morphological mutual correlation *CM* defined by equation (7)*:*

$$CM = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \min(R(i,j), S(i,j)) \tag{7}$$

- mutual information method , incorporating entropy *E*:

$$E = -\sum_{i=0}^{I} P(g_i) \log P(g_i) \tag{8}$$

where $P(g_i)$ is the probability of occurrence for the i^{th} grade on the grayscale in the range of *0-I*. The mutual information coefficient denoted by C_v is defined by equation (9):

$$Cv = -\sum_{i=0}^{I} \sum_{j=0}^{J} P(g_i, g_j) \log \frac{P(g_i, g_j)}{P(g_i) \bullet P(g_j)} \tag{9}$$

where $P(g_i, g_j)$ is the probability of occurrence of the pixel with the *i*-level gray in one frame and the pixel with the same coordinates and *j*-level gray in the range of *0-J* in the second frame.

2.2 Correction of faulty detections and movement estimation in periods T_a and T_p

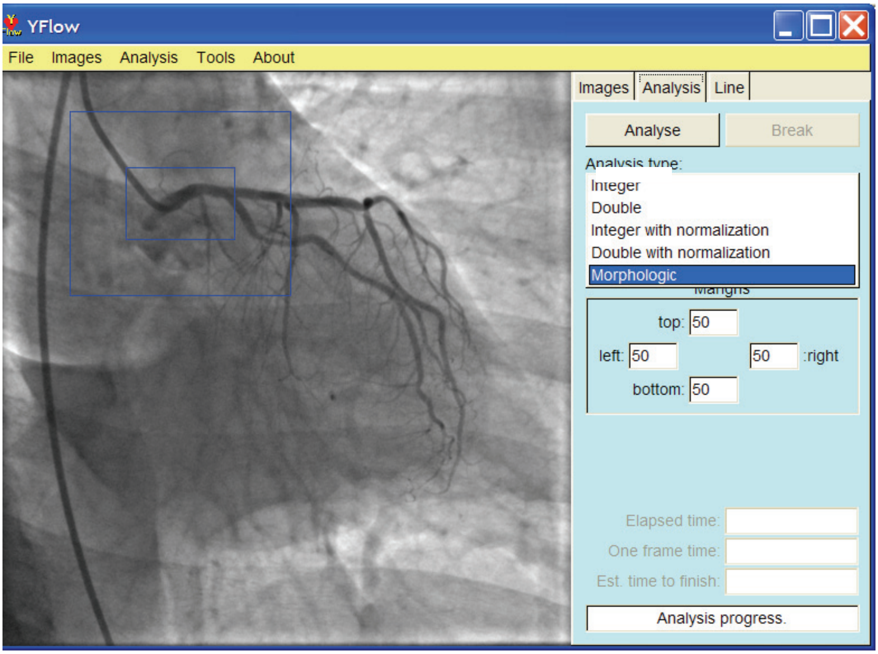
In the measurements of coronary flow with automatic data collection method [Goszczyńska, 2004] manual correction was applied for faulty artery movement detections during the contrast period T_c (medium presence) using time-related interpolation. To draw the artery movement trajectory in the period T_a (before contrast medium injection) and during the period T_p (contrast medium washing-out) a manual method of time extrapolation was applied.

2.3 Algorithm and technical solution

Fig. 5 describes the semiautomatic procedure of establishing a cross-sectional line for an artery in the *N*-frame sequence. It starts with the operator establishing frame n_r where he positions window *R* containing the template and determines the size *S* of the area to be investigated in the process (see Fig. 4). Next the procedure is initiated to find the values for indexes *k, l* (for each frame in sequence), for which the similarity measure *C* of the window *R* and subarea of window *S* reaches the maximum value.

```
• image sequence reading  n: 1 - N
• pattern window R marking in image $n_r$
• searching window S  marking in image $n_r$
• n=1
• for N images in sequence:

        o  finding of index $k_{max}$, $l_{max}$
        o  determining the tables of index value: $k[n]=k_{max}$ $l[n]=l_{max}$

• manual determining of the cross-sectional line l with p pixels of lenght in image $n_r$: $l[x_1,y_1...x_p,y_p]$
• calculation the coordinates of cross-sectional lines in  N images in sequence:

        o  line_1[...]
        o  line_n[$x_1$-k[$n_r$]+k[n], $y_1$-l[$n_r$]+l[n]... $x_p$-k[$n_r$]+k[n], $y_p$-l[$n_r$]+l[n]
        o  line_N[...]
```

Fig. 5. Procedure for determining semiautomatic cross-sectional lines.



top: 50

left: 50

50 :right

bottom: 50

Fig. 6. User interface of *YFlow* software package

After the operator positioned the cross-section line *l* with *k* pixels in the frame n_r, the procedure determining the coordinates of cross-sectional lines on *N* frames of the sequence is initiated according to the index table computed in the previous stage. The next stage is the automatic brightness data collection along the preset lines (Fig. 3a – line $D_a(x)$).

The MS Windows application *YFlow* (Fig. 6) implementing the automatic movement tracking of a chosen artery fragment using the method of template matching has been developed (the C++ language, authors: H. Goszczyńska and L. Kowalczyk) [Goszczynska et al., 2006].

3. Results

Fig. 7 shows an example of a proper matching performed with mutual correlation method for all frames in the interesting range of frames, that is from the moment of contrast appearance until it is washed-out from the chosen artery segment.

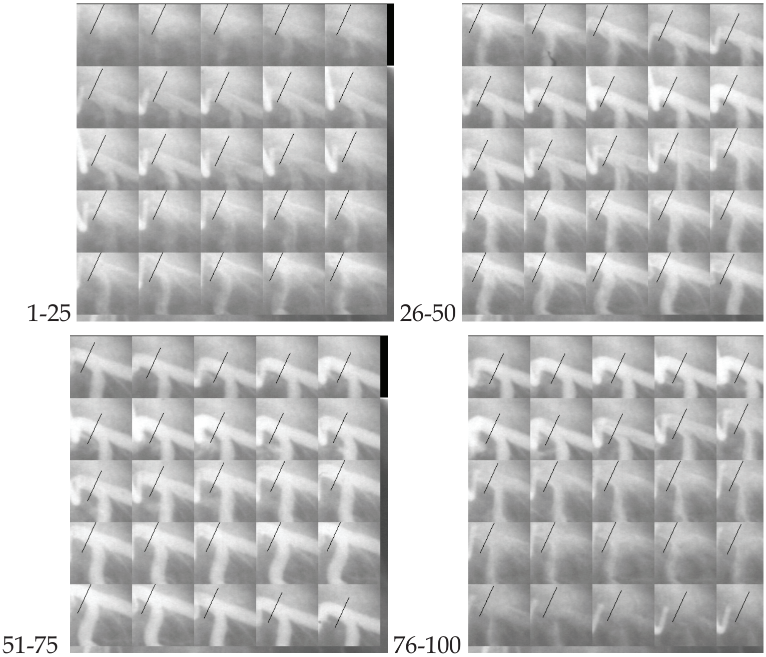


Fig. 7. Parts of 100 images from a negative coronarographic film with an automatically marked cross-sectional line

There were seventy coronarographic sequences analyzed in [Goszczynska, 2004] (for 16 patients, 4-5 sequences per patient). A single case of faulty matching in the analyzed material was observed. In [Goszczynska, 2008], examples of mismatch of template *R* with window *S* in a coronarographic sequence (Fig. 8) and the comparison of the matching results calculated for the matching concordance measurements listed in Section 2.1 are presented (Fig. 9).

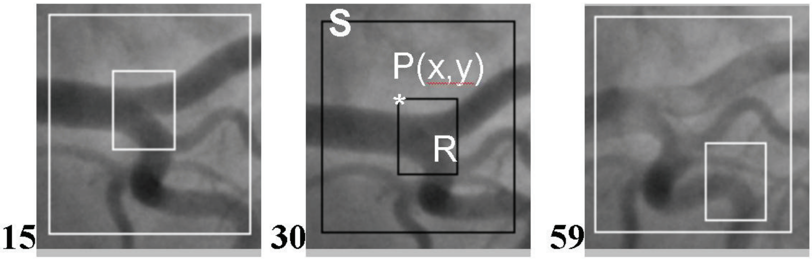
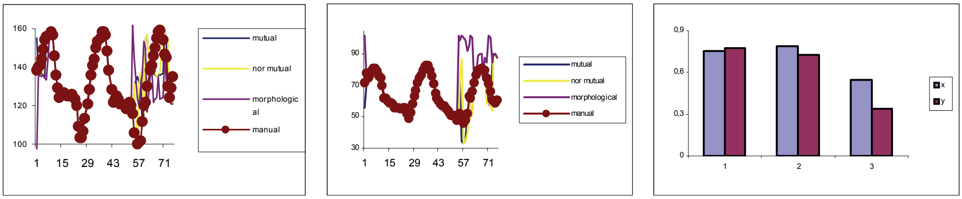


Fig. 8. Example of mismatch of template *R* with window *S i*n image 59



a) b) c)

Fig. 9. Curves *x(t)* (a) and *y(t)* (b) for the point *P* (Fig. 8) for automatic positioning (according to matching measurements) and for manual drawing and correlation coefficients for *x(t)* and *y(t)* curves between automatic and manual positioning (c): mutual correlation (1), normalized mutual correlation (2), morphological correlation (3)

Faulty matchings were detected on the basis of visual perception. During the tests described in [Goszczynska, 2005], in the case of faulty matching during the T_c period (concerning single frames only) the correction was introduced by replacing the densitometric data, i.e. *A(t)* values, obtained from the nearest frames with the proper matching.

For all the frames before and after the T_c period the time extrapolation was used by taking the repetition of the data of the last heart cycle of the T_c period within the T_p period and the first heart cycle of T_c period in the T_a period.

The trajectories of some selected artery segments were calculated. The examples of trajectories for one heart cycle for the left and right coronary arteries (RCA) are shown in Fig. 10. The degree of perfusion of the part of the myocardium can be estimated using these trajectories, and the analysis of the changes in duration of particular heart cycle phases can be performed.

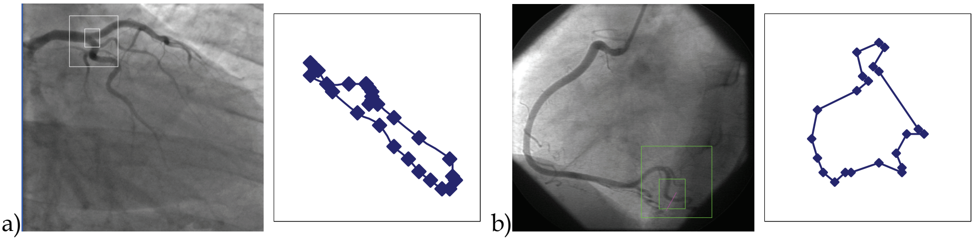


Fig. 10. Examples of trajectories for one heart cycle for a selected part of image of the LCA (a) and RCA (b)

4. Discussion

The occurrence of faulty matching and computational complexity are the main problems of the method of automatic tracing of the artery segment movement.

The most frequent reason for mismatching is too small size of searching area *S*. The proper positioning of this area must be done according to the results of the analysis of the maximum movements of the *R* mask with a chosen artery segment. Such faults can easily be avoided during the analysis by choosing a bigger searching area (taking care not to include more objects similar to the one which is necessary for the detection). However this operation requires more time for the analysis, so for calculation of C_{max} *S* could be kept small but displaced slightly for each consecutive frame, based on the previously estimated movement. This solution, unfortunately falls into the same trap as that described below.

The changing template algorithm has also been tested. As a primary template a fragment of an artery from the frame at the end of injection was selected. In such case, it works as a template only for comparison with the previous and the next frame. For the remaining frames the template is a result of the previous searching. Such an algorithm, intuitively better, still has a trap: due to an accidental mismatching the template may change entirely. Various solutions of this problem are proposed in the literature, mainly based on the reliability of the match [Guest, 1999].

Besides the problem of finding the best template or templates when choosing an optimal window size (*m, n*) containing the template and an optimal size of the searching area k_{max} and l_{max}, the detection faults may also be reduced by adopting a stronger criterion of similarity. The tests we have conducted showed that the best correlation coefficient is a normalized one. A morphological mutual correlation (a sharper matching peak) works better than a linear mutual correlation in detection of the area *R* within area *S*. Moreover, the morphological correlation algorithm (the sum of minima) is faster than the linear correlation algorithm (sum of products). Yet it is not determined by our tests what are the advantages of one algorithm over the other in regards to the precision of the matching.

The results obtained with our method should be compared with those based on other criteria, when for example two templates are used or other similarity measures, i.e. those based on histogram analysis applied for the difference (subtraction) of two adjacent frames [Buzug et al., 1998].

Our algorithm inability to analyze fragments of the object with no characteristic morphological details such as bifurcations or curvatures is a known issue. The solution here would be the incorporation of 3D interpolation methods [Schrijver, 2002].

Regarding the problem of faulty matching correction a 2D interpolation for the template positioning based on the neighbor proper matching seems to be a better solution than the one suggested in the present study. Yet specifying the point for which the automatic template matching was performed, the process of replacing densitometric data was easier and led to a smaller error (this is due to the continuity of the process of changing the indicator concentration in the artery cross-section investigated).

Controlling of the faulty matching can be computer-assisted, e.g. by comparing the values of indexes (l_n,k_n) obtained with the range limit, and finding the maximum movement of the artery fragment during the heart cycle, or by the analysis of movement direction, i.e. the direction of the vector defined by indexes (l_{n-1},k_{n-1}) and (l_n,k_n).

A proper matching for the T_p period is essential in the case when a part of the myocardium perfusion is estimated, which movement, on the basis of the closest artery displacement, is calculated. The degree of perfusion can be estimated from the image brightness changes of the part of the myocardium. The analysis of an artery movement trajectory may also allow to estimate changes in duration of the heart cycle phases and may serve as a basis for modeling epicardial strains [Young & Hunter, 1992].

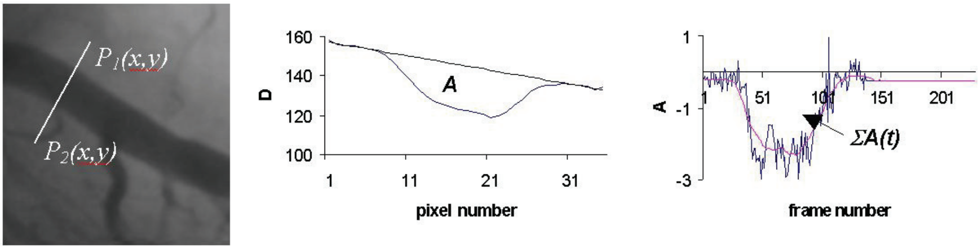
In the next section described is the estimation of the error in the densitometric curve calculation caused by the use of the automatic procedure for data collection.

4.1 Method of error estimation

The presented here application of the correspondence finding technique was used in coronary flow estimation by computer analysis of angiographic image sequences. For the estimation of the error caused by the automatic artery tracking method one may define as the reference either the values of the coordinates obtained during manual cross-sectional line drawing or other values calculated during the estimation of the flow rate. The following options for the reference value were analyzed:

- the coordinates of the P_1 and P_2 ends of the cross-sectional line (Fig. 11a)
- the value of the A field defined by equation (1) (Fig. 11b)
- the value of the field under the curve $A(t)$ as defined by equation (A11) (Fig. 11c)

The Fig.11 below depicts the three above-mentioned values.



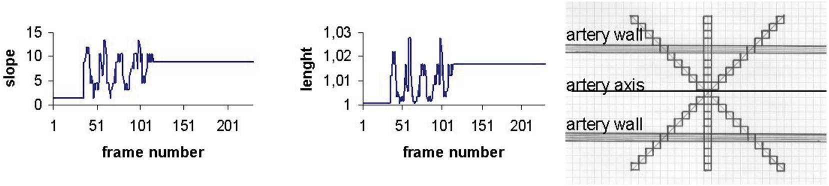
a) b) c)

Fig. 11. Examples of the reference values for estimation of the error caused by automatic cross-sectional line positioning (description in the text)

The following section discusses the analysis of the error caused by the automatic drawing of the cross-sectional line for the three above listed reference values.

4.1.1 The coordinates of cross-sectional line ends

The coordinates of the cross-sectional line end values were used for the estimation of the slope angle of the positioned cross-sectional line and for the construction of the movement trajectory. For image sequence of RCA (Fig. 10b), the changes in the slope angle of a manually positioned cross-sectional line with the axis of the artery segment were measured. It allowed to determine the error level of the automatic analysis caused by the fact that for the implemented algorithm the cross-sectional line has the same slope. For the example tested the difference between the two methods was not greater than 14° (Fig. 12a).
There was also the influence of those changes on the length of the cross-sectional line for one pixel (Fig. 12b) calculated (with a unit being the pixel size).



a) b) c)

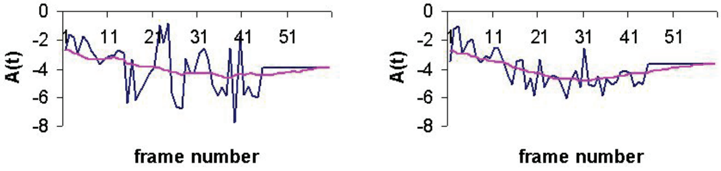
Fig. 12. The changes in the slope angle for the manually positioned cross-sectional line in the T_c period: a) the slope angle, b) the line length / (number of pixels – 1), c) a diagram of the influence of the cross-sectional line slope with the axis of an artery fragment on the line length in pixels.

As presented in Fig. 12c for the three positions of the line its length expressed in pixels is constant. This is due to the line generation algorithm in the discrete space (for the simplest position of the artery axis on the digitized plane). Thus in our case the fact that a constant slope of the cross-sectional line is assumed has no influence on the calculation results, that is on the range of the integration in equation (1). Despite the number of pixels in the segment remaining constant, the pixels that create the segment are not identical, which is also a source of error.

The only exception to the above results would be such a positioning of the cross-sectional line that the range of change of 14° covers the line drawn at 45° with the artery axis, which in practice means a wrong positioning of the cross-sectional line.

4.1.2 The value of field *A*

The changes in the background of the artery fragment investigated make the value of the *A* field sensitive to the slope angle of the cross-sectional line and the artery axis and to the interpolation method of the background lines [Goszczynska, 2004]. Fig. 13 shows two densitometric curves for the same cross-section of a catheter filled with a contrast medium, computed with two background line interpolation algorithms. Although method I (Fig. 13a) gives larger errors compared with method II (Fig. 13b), the final result, that is for the field under the *A(t)* curve, is computed with significantly smaller error. The conclusion based on the above results was to use the calculation of the field under the *A(t)* curve as the reference values for the estimation of accuracy of automatic cross-sectional line positioning.



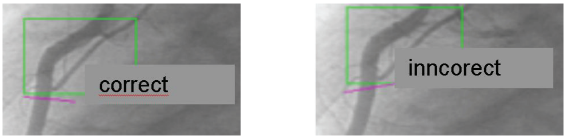
a)

b)

Fig. 13. Densitometric curves *A(t)* and their low pass version for the same cross-sectional line of the catheter calculated for two background line interpolation algorithms: a) method I, b) method II

4.1.3 The area under the densitometric curve *A(t)*

To estimate the error resulting from automatic data collection, densitometric analyses of the same artery segment were performed for automatic and manual positioning of the cross-sectional lines. The lines were positioned in three ways: automatically correctly (i.e. perpendicularly to the artery axis) on the pattern image, automatically purposely incorrectly (i.e. not perpendicularly to the artery axis) and manually correctly (Fig. 14).



a)

b)

Fig. 14. The position of the artery cross-sectional line on the reference image: a) correct position (visual estimation), b) incorrect position (intentional)

There were analysed three widths of the cross-sectional line and three algorithms of the value *A* calculation [Goszczynska, 2004]. Fig. 15 presents the ranges of the calculated area values with marked maximum and minimum values. Two types of errors were assessed. Error of type 1 caused by the automatic data collection was defined as:

$$er1 = \frac{\max_c - \min_c}{\min_m} \tag{10}$$

where max_c, min_c are the maximum and minimum values of the area for "correct" lines, min_m is the minimum area value for "manual" lines. Error of type 1 was estimated as less than 11%. This type of error is associated with the imperfection of the matching algorithm, e.g. when rotation is disregarded in equation (4). Error of type 2 (also called the total error of the flow estimation method) calculated for all the measurements performed was defined as:

$$er2 = \frac{\max_T - \min_T}{\min_T} \tag{11}$$

where max_T , min_T are the maximum and minimum values of the area for all lines. The estimation of this error, including both the previous error and the error caused by the operator, was assessed to less than 24%. The results of the error calculation of the maximum possible error for the results of the measurements are presented in Fig. 15.

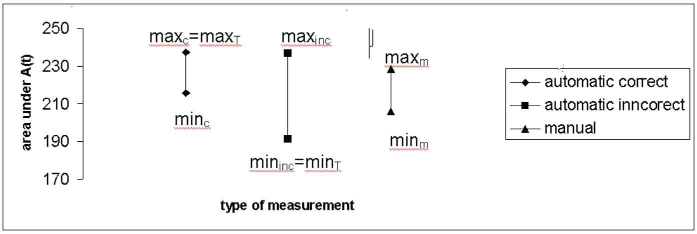


Fig. 15. Ranges of values of the area $\sum A(t)$ for automatic correct, automatic incorrect and manual positioning of the cross-sectional line

5. Conclusion

The algorithm described makes it possible to automatically collect the data used for the coronary blood flow measurement based on the analysis of a diluted indicator; though some occurrences of mismatch happen. The assessed error of the method did not exceed 11%. The algorithm of automatic data collection for the coronary flow estimation significantly accelerates the collection of the data without deteriorating the accuracy of the results obtained. Our method of automatic tracing of the measurement area not only reduces the manual effort of the operator involved in positioning the cross-sectional line in the frame sequence, but also provides a simple method of obtaining a densitometric curve with the time resolution equal to that of the timeline of the sequence investigated. Most of the other approaches to the measurements of coronary flow are based on the analysis of densitometric curves calculated from the frames referring to a particular phase of heart cycle and are later interpolated with a preset fitting curve. The presented algorithm makes it also possible to correct mismatching and incorporation of movement trajectories obtained to analyze the sequence range in which there is no contrast medium in the artery segment investigated (invisible in the frame). This permits perfusion estimation of the part of the myocardium which surrounds the artery segment. For the artery (LCA and RCA) analyzed segments not taking the rotations into consideration has no influence on the accuracy of the collected data due to the relatively small (<45°) rotation movement of the cross-section investigated during the heart cycle and the features of algorithms generating straight lines in the digitized space. It seems that this relatively simple algorithm can be accepted even in routine clinical tests due to a short time (few minutes) of the sequence analysis.

Some other ideas related to the solutions of the problem of effective artery segment tracing can be mentioned here:

- Inclusion of some searching strategy into the matching algorithm to reduce the number of positions of the template for comparison with a selected fragment [Orkisz, 1999].
- Investigation of the efficiency of the Fourier transform [Fei, 1999],
- Use of specialized signal processors [Bogorodzki, 2000],

6. Appendix A

Coronary flow estimation using densitometric measurements

The calculation of the indicator concentration from a radiographic image is based on image brightness measurements. The differences in brightness in X-ray images are caused by spatially different attenuations of the X-ray beam by the tissue, as described by the Lambert-Beer law:

$$I_1 = I_0 e^{-sl} \tag{A1}$$

where I_0 is the intensity of the monochromatic input beam, I_1 is the intensity of the attenuated beam at a distance l [cm], and s is the linear X-ray attenuation coefficient [cm^{-1}]. In angiographic images, morphometric and haemodynamic measurements are performed mainly on some parts of the image in the region of uptake of the contrast medium, i.e. in arteries or the myocardium filled with an X-ray indicator, whose attenuation coefficient s is significantly higher than that of the blood (the blood attenuation coefficient is neglected).

The value of the distance l can be expressed as:

$$\mathrm{l} = -\frac{1}{s}\ln\frac{\mathrm{I}}{\mathrm{I}_0} \tag{A2}$$

Assuming a linear relationship between the X-ray intensities I and I_0 and the corresponding image brightness D and D_0 (image value without the indicator on the same site for the same exposure parameters , i.e. a background value) respectively, the value V_{pi} of the indicator in the volume along the distance l, viewed as an area of one pixel a [cm^2], can be expressed as:

$$\mathrm{V}_{\mathrm{pi}} = -\frac{a}{s}\ln\frac{\mathrm{D}}{\mathrm{D}_0} \tag{A3}$$

For the region of interest (ROI) consisting of P_{ROI} pixels, V_{ROIi}, equation (A4) becomes a summation calculated for each pixel with coordinates $m,n \subset ROI$:

$$\mathrm{V_{ROIi}} = -\frac{\mathrm{a}}{\mathrm{s}} \sum_{\mathrm{m,n \in ROI}} \ln \frac{\mathrm{D(m,n)}}{\mathrm{D}_0(m,n)} \tag{A4}$$

Introducing

$$\mathrm{A_{ROI}(t)} = -\sum_{\mathrm{m,n \in ROI}} \ln \frac{\mathrm{D(m,n,t)}}{\mathrm{D}_0(m,n,t)} \tag{A5}$$

an equation for the densitometric curve in the time domain is obtained. Equations (A4) and (A5) provide a basis for densitometric measurements performed on angiographic images. Using the Stewart-Hamilton equation from the indicator-dilution theory, the coronary blood flow *Q* can be expressed as a ratio of the mass *M* [g] of the injected indicator to the area under the curve of the indicator concentration *c* [g/cm^3] versus time:

$$Q = \frac{M}{\int_0^\infty c(t)dt} \tag{A6}$$

This equation was a basis for different measurement techniques, e.g. thermodilution, used since the early eighties for densitometric purposes. The indicator concentration in the volume V_{ROI} represented in the image by the region of interest *ROI* can be calculated as follows:

$$\mathrm{c(t)} = \frac{\mathrm{m_i(t)}}{\mathrm{V_{ROI}}} = g\frac{\mathrm{V_{ROIi}(t)}}{\mathrm{V_{ROI}}} \tag{A7}$$

where m_i is the mass of the undiluted indicator of volume V_{ROIi}, *g* is the density of the undiluted indicator [g/cm^3]. Using equation (A4) for densitometric measurements, equation (A7) can be rewritten as:

$$\mathrm{c(t)} = \mathrm{g} \cdot \frac{-\frac{\mathrm{a}}{\mathrm{s}} \sum_{m,n \in ROI} \ln \frac{\mathrm{I(m,n,t)}}{\mathrm{I_0(m,n,t)}}}{\mathrm{V_{ROI}}} \tag{A8}$$

and using equation (A5) we obtain

$$\mathrm{c(t)} = \mathrm{g} \cdot \frac{-\frac{\mathrm{a}}{\mathrm{s}} A_{ROI}(t)}{\mathrm{V_{ROI}}} \tag{A9}$$

Denoting

$$\mathrm{k} = -\frac{sV_{ROI}}{ga} \tag{A10}$$

$$Area = \int_0^\infty \mathrm{A_{ROI}(t)}dt \tag{A11}$$

and using equation (A9), equation (A6) can be rearranged as:

$$Q = \frac{M}{\int_0^\infty g \cdot \frac{-\frac{a}{s} A_{ROI}(t)}{V_{ROI}}} = -\frac{sV_{ROI}}{ga} \cdot \frac{M}{\int_0^\infty A_{ROI}(t))dt} = k\frac{M}{Area} \tag{A12}$$

Published in part in Machine Graphics & Vision (2008) 17: 69-90 (with permission).

7. References

Benayoun, S.; Ayache N. (1998). *Dense Non-Rigid Motion Estimation In Sequence of Medical Images Using Differential Constraints*. International Journal of Computer Vision 26(1), Kluwer Academic Publishers, 25-40.

Bogorodzki, P.; Wolak, T.; Piątkowski, A. (2000). *Real Time Implementation of the Parametric Imaging Correlation Algorithms*, 59th ICB Seminar on Recent Development in MR Imaging, Warsaw

Buzug, T. M.; Weese J.; Strasters K.C. (1998). *Motion detection and motion compensation for digital subtraction angiography image enhancement*, Philips J. Res. 51, 203-229

Buzug, T.M.; Weese J. (1998). *Voxel-Based Similarity Measures for Medical Image Registration in Radiological Diagnosis and Image Guided Surgery*, Journal of Computing and Information Technology – CIT 6, 2, 165-179.

Coatrieux, J.-L. et al. (1995). *2D et 3D Motion Analysis in Digital Subtraction Angiography.* Computer Vision, Virtual Reality and Robotics in Medicine, (Ayache N. ed.), Springer, CVRMed

Fei, R.; Gui, L.; Merzkirch, W. (1999). *Comparative study of correlation-based PIV evaluation methods*, Machine Graphics & Vision, 8(4), 572-578.

Gonzalez, R.C.; Wintz P. (1987). *Digital Image Processing*, Addison-Wesley Publishing Company.

Goszczyńska, H. (2004). *Computer analysis of coronarographic images for coronary flow calculation* (in Polish) PhD Thesis, Institute of Biocybernetics and Biomedical Engineering Polish Academy of Sciences, Warsaw

Goszczyńska H. (2008). Method for Densitometric Analysis of Moving Object Tracking in Medical Images Machine Graphics & Vision 17:69-90,

Goszczyńska, H.; Rewicki M. (2008). *Coronary Flow Evaluation by Densitometric Analysis of Sequences of Coronarographic Images*, Med Bio Eng Comput, 46:199-204, DOI 10.1007/s11517-007-0288-5

Goszczyńska, H; Kowalczyk L, Rewicki M.: *Clinical study of the coronary flow measurements method based on coronagrographic image sequences*. Biocybernetics and Biomedical Engineering (Polish Academy of Sciences) 26:63-73

Guest, E.; Fidrich, M.; Kelly, S.; Berr, E. (1999). *Robust surface matching for registration, 3-D Digital Imaging and Modeling*, 1999. Proceedings. Second International Conference on, 169 – 177

Hildreth, E. C. (1987). *The analysis of visual motion: from computational theory to neural mechanism*, Annual Review of Neuroscience, 10, 477-533.

Horn, B. K. P.; Schunck B. G. (1981). *Determining Optical Flow*, Artificial Intelligence 17, 185-203.

Konrad, J. (2000). *Motion Detection and Estimation*, Handbook of Image&Video Processing, ed. Al Bovik, Academic Press, San Diego, 207-225.

Maragos, P.; Pessoa, L.F.C. (2000). *Morphological Filtering for Image Enhancement and Detection*, in. Handbook of Image&Video Processing, ed. Al Bovik, Academic Press, San Diego, 108-110

Meijering, E. (2000). *Image Enhacement in digital X-ray angiography*, PhD thesis, University Medical Center, Utrecht.

Meijering, E.H.W.; Nissen, W.J. & Viergever, M.A. (1999). *Retrospective motion correction in digital subtraction angiography: a review*, IEEE Trans. Med. Imaging, 18(1), 2-21.

Meijering, E.H.W.; Zuiderveld, K.J. & Viergever, M.A. (1999). *Image registration for digital subtraction angiography*, Int. J. Comput. Vision, 31(2), 227-246

Meunier J. et al. (1989). *Analysis of cardiac motion from coronary cineangiograms: Velocity field computation and decomposition*, SPIE Vol.1090, Medical Imaging III, Image Formation

Nissen S.E. & Gurley J.C. (1991). *Application of indicator dilution principles to regional assessment of coronary flow reserve from digital angiography*. In: Reiber JHC and Serruys PW (eds) Quantitative Coronary Angiography. Kluver Academic Publishers 117:245-263

Orkisz, M.; Bourlion, M.; Gimenez, G.; Flam, T. A. (1999). *Real-time target tracking applied to improve fragmentation of renal stones in extra-corporeal lithotripsy*, Machine Vision and Applications, vol.11, 138-144

Rong J.H. (1989). *Estimation et caracterisation du mouvement en coronarographie*. These de doctorat, l'Universite de Rennes I

Rus J.C. (1995). *The image processing handbook*, CRC Press, Florida.

Schrijver, M. (2000). *Angiographic Image Analysis to Assess the Severity of Coronary Stenoses*, Twente University Press, Enschede, Netherlande.

Wang, Y.; Erez Vidan, E.; Bergman, W. (1999). *Cardiac Motion of Coronary Arteries: Variability in the Rest Period and Implications for Coronary MR Angiography*, Radiology, 213:751-758.

Young, A.A.; Hunter, P.J.; Smaill B.H. (1992). *Estimation of epicardial strain using the motions of coronary bifurcations in biplane cineangiography*, IEEE Trans. Biomed. Eng., 39 (5), 526-531