Video analytics algorithms and distributed solutions for smart video surveillance

Gaetano Di Caterina

A thesis submitted for the degree of
Doctor of Philosophy

Centre for excellence in Signal and Image Processing
Department of Electronic and Electrical Engineering
University of Strathclyde
Glasgow

2013

Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed:
Date:

Acknowledgement

First of all, I would like to express my deepest gratitude to my PhD supervisor Prof John J. Soraghan, for his wise guidance throughout my whole PhD. His help, support and advise in academic and personal matters since my undergraduate studies have been invaluable. At times, Prof Soraghan was more of a parental figure to me, rather than a supervisor. He believed in me and in this PhD, even before I had finished my MEng. As one of his many successful PhD students said: "One could not ask for a better person as a supervisor". I definitely agree.

I would like to thank my family, for their constant love and encouragement, despite the distance. They always have a nice word for me and their enthusiasm is like a guiding light. I would like to thank especially my parents, as it is their love and guidance that have made me the person I have become.

I would also like to remember and thank my friend Magnus Milner, who is no longer with us. He helped me discover my new life in Scotland.

Finally, I would like to thank my wonderful wife Michelle and my step-daughter Megan, for the love and support they give me every day. They should be awarded a prize for dealing with a very moody Gaetano during the thesis write-up period. Michelle in particular is always by my side and I do not think I could have achieved anything without her.

I dedicate this thesis to my parents, my brother, my two sisters, to Michelle and Megan, and to Magnus' memory.

Abstract

The growth in the number of surveillance cameras deployed and the progress in digital technologies in recent years have steered the video surveillance market towards the usage of computer systems to automatically analyse video feeds in a collaborative and distributive fashion. The semantic analysis and interpretation of video surveillance data through signal and image processing techniques is called Video Analytics (VA). In this thesis new video analytics methods are presented that are shown to be effective and efficient when compared to existing methods.

A novel adaptive template matching algorithm for robust target tracking based on a modified Sum of Absolute Differences (SAD) called Sum of Weighted Absolute Differences (SWAD) is developed. A Gaussian weighting kernel is employed to reduce the effects of partial occlusion, while the target template is updated using an Infinite Impulse Response (IIR) filter. Experimental results demonstrate that the SWAD-based tracker outperforms conventional SAD in terms of efficiency and accuracy, and its performance is comparable to more complex trackers. Moreover, a novel technique for complete occlusion handling in the context of such a SWAD-based tracker is presented that is shown to preserve the template and recover the target after complete occlusion. A DSP embedded implementation of the SWAD-based tracker is then described, showing that such an algorithm is ideal for real-time implementations on devices with low computational capabilities, as in the case of fixed-point embedded DSP platforms.

When colour is selected as target feature to track, the mean shift (MS) tracker can be used. Although it has been shown to be fast, effective and robust in many scenarios, it fails in case of severe and complete occlusion or fast moving targets. A new improved MS tracker is presented which incorporates a failure recovery strategy. The improved MS is simple and fast, and experimental results show that it can effectively recover a target after complete occlusion or loss, to successfully track target in complex scenarios, such as crowd scenes.

Although many methods have been proposed in the literature to detect abandoned and removed objects, they are not really designed to be able to trigger alerts within a time interval defined by the user. It is actually the background model updating procedure that dictates when the alerts are triggered. A novel algorithm for abandoned and removed object detection in real-time is presented. A detection time can be directly specified and the background is "healed" only after a new event has been detected. Moreover the actual detection time and the background model updating rate are computed in an adaptive way with respect to the algorithm frame processing rate, so that even on different machines the detection time is generally the same. This is in contrast with other algorithms, where either the frame rate or the background updating rate is considered to be fixed. The algorithm is employed in the context of a reactive smart surveillance system, which notifies the occurrence of events of interest to registered users, within seconds, through SMS alerts.

In the context of multi-camera systems, spatio-temporal information extracted from a set of semantically clustered cameras can be fused together and exploited, to achieve a better understanding of the environment surrounding the cameras and monitor areas wider than a single camera FOV. A highly flexible decentralised system software architecture is presented, for decentralised multi-view target tracking, where synchronisation constraints among processes can be relaxed. The improved MS tracker is extended to a collaborative multi-camera environment, wherein algorithm parameters are set automatically in separate views, upon colour characteristics of the target. Such a decentralised multi-camera tracking system does not rely on camera positional information to initialise the trackers or handle camera hand-off events. Tracking in separate camera views is performed solely on the visible characteristics of the target, reducing the system setup phase to the minimum. Such a system can automatically select from a set of views, the one that gives the best visualisation of the target. Moreover, camera overlapping information can be exploited to overcome target occlusion.

Contents

List of acronyms

CCTV	Closed Circuit Television
DA	Detection Agent
DS	Data Server
DSP	Digital Signal Processor
EVM	Evaluation Module
FOV	Field of view
FN	False Negative
FP	False Positive
fps	frames per second
IIR	Infinite impulse response
IP	Internet Protocol
MS	Mean shift
NCC	Normalised Cross-Correlation
PTZ	Pan-Tilt-Zoom
RGB	Red, Green, Blue
ROI	Region of interest
SAD	Sum of Absolute Differences
SMS	Short Message Service
SS	Status Server
SSD	Sum of Squared Differences
SWAD	Sum of Weighted Absolute Differences
TA	Tracking Agent
TN	True Negative
TP	True Positive
VA	Video Analytics
YCbCr	Luminance, blue chrominance, red chrominance

List of figures

List of tables

Chapter 1

Introduction

Video surveillance refers to the monitoring through video cameras of an area of interest. As this video data is usually streamed over small private video networks for local access, video surveillance systems are sometimes referred to as Closed Circuit Television (CCTV). The semantic analysis of video surveillance data through signal and image processing techniques on computer systems is called Video Analytics (VA) [2–5]. In this thesis new video analytics methods are presented that are shown to be effective and efficient when compared to existing methods.

In recent years the number of CCTV cameras deployed has grown exponentially and so the very large number of surveillance devices in place raises a serious challenge on how to analyse quickly and effectively such an enormous amount of raw video data. In traditional surveillance systems a human operator constantly looks at the surveillance monitors to spot events of interest. The drawbacks of this approach are obvious, as persons can get easily distracted and so important events can go undetected [6]. Moreover employing more surveillance personnel is expensive, when compared to the fixed costs of the system hardware and software. For this reason the video surveillance market is moving towards the usage of computer systems to automatically analyse video feeds in a collaborative and distributive fashion. Therefore video analytics is becoming more and more popular in video surveillance, to help improve both efficiency and effectiveness.

1.1 Research motivation

Visual tracking is the process of inferring information between detections of a selected target in consecutive frames, and it is an important task in many video processing systems. Many algorithms with medium-high complexity have been proposed in the literature, as reported in [1]. The Sum of Absolute Differences is a well-known metric and for its simplicity it is often used in video coding and stereo imaging. However it is not very common in tracking applications, as its performance can be compromised by partial occlusion and target changes. In adaptive template tracking algorithms the risk is to include portions of background or occluding objects in the target model during the updating step. In the long run, the target template becomes corrupt and the tracker may lose the correct target in favour of an incorrect one, which has a higher match with the corrupt template.

Colour is a strong cue in visual systems and it is invariant to scaling and rotation. The mean shift (MS) tracker [7–11] is a very commonly used colour tracking algorithm. However its weakness is in the assumption of overlapping between tracking detections in consecutive frames. In case of fast moving targets and complete occlusion, such a condition does not hold and, as the MS cannot detect occlusion or fast target loss, the tracker may start following an incorrect target and so it fails.

In video analytics event detection refers to the task of recognising when situations of interest take place. Abandoned and removed object event detection is a typical example and generally it is performed by adopting a background subtraction approach [12, 13]. However the methods proposed in the literature are not usually designed to be able to trigger alerts within a time interval defined by the user, after new abandoned and removed object events have occurred. On the contrary, alerts are indirectly triggered by background updates. Moreover such algorithms require manual tuning of their parameters.

To monitor areas larger than a single camera field of view (FOV), a multi-camera system can be deployed. The spatio-temporal information extracted from multiple cameras can then be collated. However, it is not straightforward to extend single camera tracking algorithms to a multi-camera setup, due to issues, for example, time synchronisation constraints among processes and accurate calibration of camera positions within a common 3D coordinate system.

1.2 Summary of original contributions

The main research contributions of this thesis are described below.

1. The first contribution is represented by a novel adaptive template matching algorithm for robust target tracking, based on a novel Sum of Weighted Absolute Differences (SWAD), which uses a Gaussian kernel to reduce the

effects of partial occlusion. Furthermore a novel complete occlusion handling technique for the SWAD-based tracker is presented. Such a technique can detect when occlusion is taking place and it updates parts of the target template at different rates, to preserve its overall structure (Chapter 4).

2. The second contribution is represented by a novel improved mean shift tracker with failure detection and recovery strategy. This algorithm is able to detect when the target is lost, possibly due to fast motion or complete occlusion, and it can automatically recover it. The improved MS is fast and can effectively track a target in complex scenarios. (Chapter 5).

3. The third contribution is represented by a novel adaptive algorithm for abandoned and removed object detection in real-time, where the detection time can be directly specified and the background updating rate is a function of it. In this context also a novel technique is presented, to distinguish between abandoned and removed objects solely on the gradient direction on the object contour, without any user-defined threshold (Chapter 6).

4. The last contribution is represented by a highly flexible distributed system architecture, for decentralised multi-view target tracking, with relaxed synchronisation constraints among processes. Such a system exploits a novel extension of the single camera improved MS tracker presented in Chapter 5, to a multi-camera setup (Chapter 7).

1.3 Organisation of the thesis

The reminder of this thesis is organised as follows.

In Chapter 2 modern video surveillance and video analytics are introduced. Some image and video processing techniques commonly used in VA systems are then described. After that examples of VA tasks are given, with particular emphasis on abandoned and removed object detection, to put in context the novel algorithm described in Chapter 6.

In Chapter 3, the visual tracking problem is analysed and two approaches for target tracking proposed in the literature are described in detail, namely template matching and mean shift tracker. These two methods are the starting points for the two novel tracking algorithms presented respectively in Chapter 4 and Chapter 5.

In Chapter 4 the novel adaptive template matching algorithm based on SWAD is presented. A complete occlusion handling technique is also presented for the

SWAD-based tracker. Moreover a real-time DSP embedded implementation of this tracker is described in the context of a smart surveillance system.

In Chapter 5 the novel improved MS tracker with fast failure recovery strategy is presented.

In Chapter 6 the novel adaptive algorithm for abandoned and removed object detection is presented. This algorithm is employed in the context of a reactive smart surveillance system, which notifies the occurrence of events of interests to registered users through SMS.

In Chapter 7 the highly flexible distributed system for decentralised multi-view target tracking is presented. In this system the improved MS from Chapter 5 is used to track the target in single FOVs, while the SWAD-based tracker from Chapter 4 is used to stabilise tracking detections between consecutive frames.

Overall conclusions for this thesis are reported in Chapter 8, along with suggestions for possible future work.

Chapter 2

Video surveillance and video analytics

2.1 Introduction

Video surveillance is the monitoring of a confined area of interest through video cameras. The acquired video feeds are streamed out to a set of monitors, so that human operators can watch and analyse them in real-time. The video feeds are also stored on one or more video recorders. Video surveillance systems are usually referred to as Closed Circuit Television (CCTV) and their basic functionalities are real-time viewing, recording and reviewing of recorded videos.

In the last few years, the number of CCTV cameras employed for surveillance purposes has grown exponentially and market trend research indicates that the surveillance market is still expanding [14, 15], with the UK being one of the leading countries [16, 17]. Nowadays CCTV cameras are employed in the most diverse scenarios, such as airports, stations, banks, schools, offices, retailer shops and private premises. Nonetheless, as most of these devices are still analogue, four major issues can affect CCTV usage and reduce its effectiveness:

1. Video quality: high quality analogue cameras are expensive; therefore the usage of multiple medium/low quality cameras is preferred, to achieve a good trade-off among video quality, system flexibility and costs. However, important details such as people faces and car plate numbers may be not recognisable, due to poor image quality [18].

2. Storage space: the amount of space required to store a 24 hour continuous video stream from an analogue camera is prohibitive, so that analogue video

streams are usually recorded at low frame rates (4-5 fps), to save storage space.

3. System installation: each analogue camera requires a point-to-point connection to the nearest control centre. This solution is not always feasible, as in the case of large areas to cover or rigid structural constraints.

4. Human supervision: the very large number of CCTV cameras in place raises serious challenges on how to analyse and use the enormous amount of raw video data acquired through them. Traditional surveillance systems rely on human operators, constantly looking at the surveillance monitors, to analyse the video feeds in real-time. The drawbacks of this approach are obvious, as constant human attention cannot be guaranteed [6, 19].

Such issues are slowly leading the market of surveillance products towards the usage of digital systems [20]:

- IP cameras are digital CCTV cameras that rely on the Internet infrastructure, hence the name, as their video output is streamed over the network and it is viewable remotely via a browser. Such cameras usually have high resolution and can use various compression formats, such as MPEG, H.264 and MJPEG.

- Video Servers can acquire analogue video signals and make them available in digital format over the network, so that old analogue video cameras can be accessed as IP cameras.

- Network Video Recorders (NVR) can be operated remotely and can record real-time video streams from a large number of IP cameras simultaneously.

These digital surveillance devices can be interconnected to create a complete network of sensors. They offer clear advantages over their analogue counterparts, for example:

- High video quality: digital cameras have resolutions in the order of megapixels, i.e. millions of pixels.

- Reduced memory requirements: digital videos can be compressed with ratios of 10:1 and higher. Moreover, the cost of digital memory decreases constantly.

- Easy installation and high flexibility: IP cameras can be directly plugged into any Ethernet network, avoiding cumbersome point-to-point connections. Moreover, with Power-on-Ethernet (PoE) technology, also power supply cables are eliminated. IP cameras can also be connected to the Internet via wireless extensions, for complete system flexibility.

Nonetheless, the most important advantage of using digital devices is the possibility to perform video content analysis on the surveillance feeds automatically via software. This technology is called video analytics and is becoming more and more popular in video surveillance systems, as it can help improve efficiency and effectiveness of the most expensive element and yet the one most likely to fail, i.e. human personnel [21].

In the rest of this chapter, video content analysis for video surveillance, i.e. video analytics, is introduced in Section 2.2. In Section 2.3 some image and video processing techniques widely used in video analytics algorithms are described. Section 2.4 introduces some common video analytics tasks, while Section 2.5 describes in more details the abandoned and removed object detection task. Finally Section 2.6 concludes the chapter.

2.2 Video analytics

Video content analysis is the semantic analysis and interpretation of video data through signal and image processing techniques. It can be successfully applied to many different contexts, such as health care, social sciences, industrial production and sport, where the common characteristic is the partial or total absence of human operators in the video analysis process. For example, video content analysis can be used in health care treatments to assess physical conditions of patients, such as walking and limb movement, facial paralysis gradation [22], or detection of falling in the elderly [23]. In sport, it can be used to automatically track players [24–26]. In transport, it can be used to detect, and avoid, surrounding obstacles [27].

In video surveillance, video analytics (VA) is used to automatically analyse the surveillance feeds and detect situations and events of interest, or to highlight relevant data and bring it to the attention of the surveillance personnel [2–5]. This second approach can significantly increase the efficiency of human operators in real-time monitoring operations, as they can focus their attention on a reduced set of cases. Their effectiveness can also be improved, as the number of events left undetected decreases [28]. A video surveillance system can be defined as “smart”, or “intelligent”, when it is enhanced with VA algorithms, as they add intelligence capabilities to the underlying passive CCTV infrastructure.

Historically surveillance systems have been divided into three generations [3, 29]. The first generation mainly consisted of passive analogue CCTV systems constantly monitored by human operators. The second generation of surveillance systems included digital devices, and algorithms for detection and tracking. These were developed to help the real-time monitoring of the surveillance feeds. The third generation refers to distributed systems for automated surveillance of wide areas, with a high number of cameras and sensors of different kind. Here the challenge is to effectively, and coherently, fuse together large amounts of information. In this context such distributed systems rely on a network of heterogeneous surveillance sensors, including fixed and moving cameras [30–32]. The latter are referred to as active cameras or PTZ, as they can pan, tilt and zoom upon command of a remote operator.

Video analytics algorithms can run on general purpose computers or on specialised hardware, such as Digital Signal Processors (DSPs), Field-Programmable Gate Arrays (FPGAs) or Graphics Processing Units (GPUs), to achieve faster processing [33–35]. When CCTV cameras are equipped with embedded processing units, it is possible to process the acquired data on-board and discard unimportant information. Such an approach is useful, especially in the case of surveillance systems with a large number of sensors, as it can significantly reduce the amount of data to transmit or store, and therefore save valuable memory, bandwidth and time. Surveillance sensors enhanced with “in-camera” processing are referred to as “smart sensors”.

2.3 Image and video processing for video analytics

At the core of a smart surveillance system there are video analytics algorithms which process the incoming video feeds and perform VA tasks, for example object detection and tracking. Usually a digital video stream is broken up into separate images, called “frames”, which represent discrete instants in time. The number of frames per second (fps) is the frame rate of the video stream, while the number of frames processed per second for a VA task is the frame rate of the VA algorithm. A frame acquired at time instant i, with $i \in [0, +\infty]$, can be seen as a 2-dimensional discrete function $F_i(x, y)$ with $x \in [0, N_x - 1]$ and $y \in [0, N_y - 1]$, or as a 2D N_x-by-N_y matrix $\mathbf{F}_i$. The point with coordinates (x, y) in function $F_i(x, y)$, or similarly the element (x, y) in matrix $\mathbf{F}_i$, is called a "pixel" and the product $N_x \times N_y$ defines the resolution, i.e. the number of pixels, of the frame and of the video at the same time.

Pixel values in a frame are usually integers in the range $[0, 255]$, and are therefore represented with 8 bits (1 byte), or floating point numbers in the range $[0, 1]$ with 32 bits (4 bytes). In digital devices, colour is represented as the mixture of multiple colour components, as for example in RGB, YCbCr, HSV and other colour representations [36, 37]. Therefore a colour frame is actually a 3-dimensional function, or matrix, where the third dimension identifies a particular colour plane depending on the colour representation chosen, and pixel values are triplets in the range $[0, 255] \times [0, 255] \times [0, 255]$, or $[0, 1] \times [0, 1] \times [0, 1]$. When a frame has a single colour plane, i.e. it is a 2D discrete function, it is a grey scale image. In a binary image pixels can assume only two values, either 0 and 1. Binary images are used to indicate where within a frame a given predicate is "true" (1) or "false" (0).

Image and video processing operations are then performed on video frames to extract relevant information [36, 38, 39]. Typical operations in video analytics are: segmentation, filtering, feature extraction and matching. Many techniques have been proposed in the literature to perform such operations and some of them are reported in the following subsections.

2.3.1 Segmentation

Segmentation is the process of dividing an image into two of more groups of pixels, which are referred to as regions, blobs or patches. For example segmentation can be used to separate within a frame foreground moving objects from a static background; or it can be used to extract regions with a characteristic colour, as for example skin colour.

Thresholding

Given a grey scale image $\mathbf{I}$, thresholding refers to the process of creating a binary image $\mathbf{M}$ of the same size as $\mathbf{I}$, and with pixel values being either 0 or 1, depending on whether their corresponding pixels in $\mathbf{I}$ are smaller or greater than a certain threshold ϑ, as in (2.1):

$$M(x, y) = \begin{cases} 1, & \text{if } I(x, y) \geq \vartheta \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Although methods are available to automatically determine a value for ϑ [40], in general in VA such a value is application dependant and thresholding is used in combination with other techniques. Thresholding can be generalised to colour images by thresholding each colour plane in an image separately.

Colour segmentation

Colour segmentation is based on thresholding and it uses a lower threshold ϑ_L and an upper one ϑ_U. Pixels are selected depending on whether their values fall in the interval $[\vartheta_L, \vartheta_U]$. If $\mathbf{I}$ is a grey scale frame, a binary mask can be obtained according to (2.2):

$$M(x,y) = \begin{cases} 1, & \text{if } I(x,y) > \vartheta_L \wedge I(x,y) < \vartheta_U \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

If $\mathbf{I}$ is a colour frame, a 1D interval is defined for each colour plane and then pixels are segmented depending on whether their values fall within such a multi-dimensional interval.

Background subtraction

In situations where the background in a video is static or changes slowly, background subtraction can be used to segment new and moving objects in the scene [12, 13]. If $\mathbf{B}$ is the background image and $\mathbf{F}_i$ is the i^{th} frame, a difference image $\mathbf{D}$ is computed as $\mathbf{D} = |\mathbf{F}_i - \mathbf{B}|$. Where pixel values have not changed significantly between $\mathbf{B}$ and $\mathbf{F}_i$, $\mathbf{D}$ has low values; while for significant changes, pixel values in $\mathbf{D}$ are high. Usually the difference image $\mathbf{D}$ is thresholded, to obtain a binary mask of new objects in the scene.

Other segmentation techniques

Other segmentation techniques commonly used in video analytics are:

- clustering: the whole image is divided into contiguous regions, where pixels share a chosen characteristic, for example spatial or colour proximity. Clustering techniques include: mean shift [7, 10], k-means [41], graph cuts [42].

- image differencing: two consecutive frames $\mathbf{F}_{i-1}$ and $\mathbf{F}_i$ are subtracted from each other, to obtain a difference image $\mathbf{D} = |\mathbf{F}_i - \mathbf{F}_{i-1}|$. As for background subtraction, image differencing is used to segment moving objects in the scene [43].

- active contours: large closed contours are superimposed to an image and shrunk iteratively, to adapt them to the underlying image structure [44]. A separate contour is needed for each object to segment in the scene.

- optical flow: the motion of real objects in space can be represented with a 3D vector. In a video sequence, such 3D motion is projected on the image plane and therefore it is associated with a 2D vector. The estimation of these projected 2D motion vectors in the image is called optical flow field. Such a vector field should have a high density, i.e. motion vectors should be computed with pixel accuracy, and many techniques have been presented in the literature to estimate the optical flow in video frames, as reported in [45–47]. Moving objects with high magnitude motion flow can be easily segmented from the rest of a static scene.

2.3.2 Feature extraction

Feature extraction refers to the process of computing characteristic properties of an image. Such features can apply to single points, to groups of pixels or to the whole image, and they provide means of comparison within the same image or with other ones [37, 39].

Colour histogram

Colour histogram is probably the easiest and most common feature to compute. It can apply to the entire image or to parts of it. In simple terms, a colour histogram is a function and it associates to each value in the image colour range the number of pixels within that value. For a grey scale image the colour histogram is a 1D function, while it is multi-dimensional in the case of colour images. As a histogram is discrete, it can be represented as a vector or a matrix, and its elements are called "bins". Moreover, the colour range can be quantised, so that each bin itself refers to a range of values, rather than a single value [36].

Gradient and edge detection

When images are seen as 2D functions, it would be obvious to identify edges with the discontinuity points arising from the transition between two different colour regions. It is then possible to apply gradient operators in the form of templates of various size and aspect, such as Sobel, Prewitt and others [36, 38], to detect intensity changes in such an image. Usually these operators are applied separately in the horizontal and vertical directions. The output consists of two images representing, at each location, the values of the gradient components I_x and I_y for the pixel in the original image. Such information can be used to describe local characteristics of a region in an image.

However, as digital images are discrete functions, discontinuities are present between any two pixels which do not have exactly the same intensity values. As a consequence, edge detection in a digital image usually includes thresholding, where only edges with strength, i.e. gradient magnitude, above a certain value are considered to be "real edges". The value of the threshold can be selected manually or automatically. The Canny edge detector [48] uses a low threshold and a high threshold on the gradient magnitude and orientation images, which are computed with one of the aforementioned gradient operators.

Other feature extraction techniques

Other features commonly used in video analytics algorithms are:

- local binary patterns (LBPs) [49];
- corner detection [38, 50];
- histogram of oriented gradients (HOGs) [51];
- Haar features [52, 53];
- edgelets [54];
- optical flow [45–47].

2.3.3 Matching

Matching is the process of comparing two images or regions and assessing their similarity. Usually one of them acts as a model or reference, while the other is a candidate match for it. The most common techniques for matching are histogram matching and template matching.

Histogram matching

In histogram matching, two histograms are compared and their degree of similarity is evaluated. As suggested in literature, such comparison can be performed using several different metrics:

- histogram intersection [55]
- Bhattacharyya distance [9, 11]
- Euclidean distance [56]
- Manhattan distance [24]

The Manhattan distance is also referred to as Sum of Absolute Differences (SAD), while the Euclidean distance is referred to as Sum of Squared Differences (SSD). Both Manhattan and Euclidean distance are particular cases of the Minkowski distance [57].

As it cannot be proved that a particular metric is better than all the others, different researchers have experimentally shown in their work that each metric gives better performance in the specific context where they have been used.

Template matching

In template matching the model is represented by a rectangular patch, which includes the actual appearance of the reference object. Matching is then performed by convolving the template with an image and computing a chosen metric at each step [36, 38]. Pixel values in the template are usually intensities and in general template matching is used for motion estimation, target tracking and object detection within an image.

2.3.4 Filtering and enhancement

As in the case of 1D signal filtering, in image filtering unwanted or unnecessary characteristics of the input signal, i.e. the image, are removed. As the quality of the output image after filtering is in some sense “better” than in the original image, filtering is sometimes identified with image enhancement. Depending on the noise being removed in the image, and on the information in it that one is interested in, different image processing techniques for filtering and enhancement can be applied [36, 38].

Convolution-based filtering

In general, image filtering is performed by convolving an image with a small rectangular patch referred to as a “kernel”. Such operation is linear and the shape of the kernel determines the type of filtering, i.e. low-pass, high-pass, and so on [36, 38].

For example, if the intensity values in the kernel are all the same, simple averaging takes place. If the kernel has the shape of a 2D Gaussian function, the filter smooths the original image, and the degree of smoothing is controlled by the value of σ. Also template matching and edge detection are both based on kernel convolution. Convolution-based filtering is also used to perform kernel density estimation, as explained in [58].

Histogram equalisation

When pixel intensities in an image do not span over the whole available range, but they are instead localised within a portion of it, it is possible to stretch the colour histogram of the image, to make use of the entire range. Such a technique is called histogram equalisation and it is used to enhance the contrast in images that look "flat" [36, 38].

Morphology

In image processing, mathematical morphology refers to a set of non-linear geometrical transformations for image filtering [59]. Typical morphological operations are: erosion, dilation, opening (an erosion followed by a dilation) and closing (a dilation followed by an erosion). Morphology is widely used in video analytics to remove small noisy regions in an image and to fill gaps.

Gray World Assumption

The Gray World Assumption states that in natural images the pixel values in each colour plane should average to a common grey value [60]. Therefore, by rescaling the image intensities in each colour plane so that the three averages are roughly the same, it is possible to eliminate possible colour distortion in the image introduced by artificial light sources.

Median filter

A very common filtering technique is represented by the median filter, which is non-linear but very useful in removing salt and pepper noise from images, while preserving much of the original detail. For each point in the image, pixels from a neighbourhood are selected and sorted based on their intensities; the value of the reference pixel is then replaced in the output image with the value in the middle of the sorted list of neighbouring intensities [36, 38].

2.4 Video analytics tasks

In this section five video analytics tasks common to many VA systems are described.

2.4.1 Object detection

Object detection is the process of locating a target object within an image and isolating it from the rest of the scene. Such an operation is usually the first processing step in a VA system [3]. Typical objects to detect are: people [51, 54, 61, 62]; faces [63–66]; abandoned objects [67–70]; cars [27, 71, 72]. A very large variety of approaches have been proposed in the literature and two short surveys about this topic can be found in [1, 3]. As a very coarse classification, object detection methods can be divided into two categories:

- Segmentation-based: in these approaches segmentation is used to locate and isolate the object from the rest of the image. No visual characteristics of the object are taken into account, but regions within the frame are selected depending on the structure and characteristics of the image itself. Segmentation-based object detection methods include: background subtraction [67–70], image differencing [43] and clustering [73].

- Feature-based: such methods assume that some visual characteristics of the object to be detected are known. These features are then searched for within the whole image and the object is deemed as detected if a set of features in the image matches a known pattern to a certain agreed level of confidence. In general these methods exhaustively search for features within the whole frame and then machine learning techniques are used to perform pattern recognition, if an exact match between object model and features is not possible. Feature-based methods include: Haar-like features [63], histogram of oriented gradients [51], local binary patterns [74], integral histogram [75], template matching [76].

2.4.2 Tracking

Tracking is the detection and recognition of the same target object in consecutive frames in a video feed [1, 37]. Such a task is central to any type of high level analysis of videos, as for example action recognition and behaviour analysis. A more detailed review of tracking methods in the literature is given in Chapter 3.

2.4.3 Event detection

In event detection an alert is triggered when a particular event of interest takes place [77]. Events to detect usually have known patterns, as for example abandoned and removed object detection [67–70], door access detection [78], falling persons [23], and others [79]. In these cases it is possible to define a set of visual and temporal rules, to describe the event. A brief review of methods described in the literature for abandoned and removed object detection is given in Section 2.5.

Very different is the case when a situation not previously thought of arises, so the event is unknown and rules are not available to identify it. The particular characteristic of unknown events is that they present unusual patterns, compared to the normal flow in the video feed. For example if in a motorway video feed cars are static, as in the case of an accident or a long queue, such situation should be considered unusual, and an event alert should be triggered. Unusual event detection usually relies on machine learning and pattern recognition techniques, to learn usual patterns in the video feeds and recognise unusual situations [80].

2.4.4 Behaviour analysis

Behaviour analysis tries to recognise a single action, or set of actions, of individuals in a video feed to understand their behaviour [2]. Such a task implies a high level of abstraction and it usually uses machine learning, as human actions are difficult to describe and identify exactly. In this case patterns are identified and learnt by example, and therefore pattern recognition is used rather than pattern matching.

2.4.5 Crowd analysis

Crowd analysis refers to processing and interpretation of video feeds in which a large number of individuals is present. In these scenarios it is not possible to exactly, and entirely, segment each single person, therefore groups of people have to be treated as an entity, i.e. a crowd [81]. In recent years much attention has been given to this VA task as crowded places inherently have higher sensitivity with respect to both safety and security [82]. Typical crowd analysis operations are: crowd density estimation, flow estimation and counter flow detection within a crowd, detection of crowd formation and dispersion.

2.5 Abandoned and removed object detection

Being able to recognise when an object is left unattended, or abandoned, is an important task in video surveillance since such an object may contain harmful devices endangering nearby people. Similarly it is important to detect when an object is taken away, or removed, from a scene, as for example in the case of valuable pieces of art in a museum. Therefore it is not surprising that both tasks always raise high interest in the video analytics community, as proved by the abundant amount of research carried out on these topics [67–70, 83–85].

As the detection of abandoned and removed objects in a video sequence can be seen as the problem of locating and identifying differences between pairs of images, i.e. one with and one without the object of interest, it is easy to understand why all the work in this field uses background subtraction as a general approach [12, 13]. In fact, while image differencing can only highlight moving objects in the scene, background subtraction can also highlight new static objects. Such an approach can then be extended to the case of removed objects if the background left uncovered is interpreted as a new object. This is why both abandoned object and removed object detections are dealt with jointly.

2.5.1 Background subtraction

In the background subtraction approach for abandoned and removed object detection, the difference image $\mathbf{D}_n$ between each new frame $\mathbf{I}_n$ and the background image $\mathbf{B}_n$ updated for every n is computed as:

$$\mathbf{D}_n = |\mathbf{I}_n - \mathbf{B}_n| \tag{2.3}$$

High pixel values in $\mathbf{D}_n$ are due to significant differences between $\mathbf{I}_n$ and $\mathbf{B}_n$, so that high valued regions in $\mathbf{D}_n$ are likely to correspond to new objects in the scene. To reinforce such an assumption, the difference image $\mathbf{D}_n$ is thresholded as in (2.4):

$$\mathbf{M}_n = \begin{cases} 1, & \text{if } \mathbf{D}_n > \mathbf{T}_n \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

where $\mathbf{M}_n$ is a binary mask. Therefore unconnected regions in $\mathbf{M}_n$ refer to possible new static objects in $\mathbf{I}_n$, with a degree of confidence related to threshold $\mathbf{T}_n$. In Fig. 2.1, a visual example of such an approach is given for grey scale images.

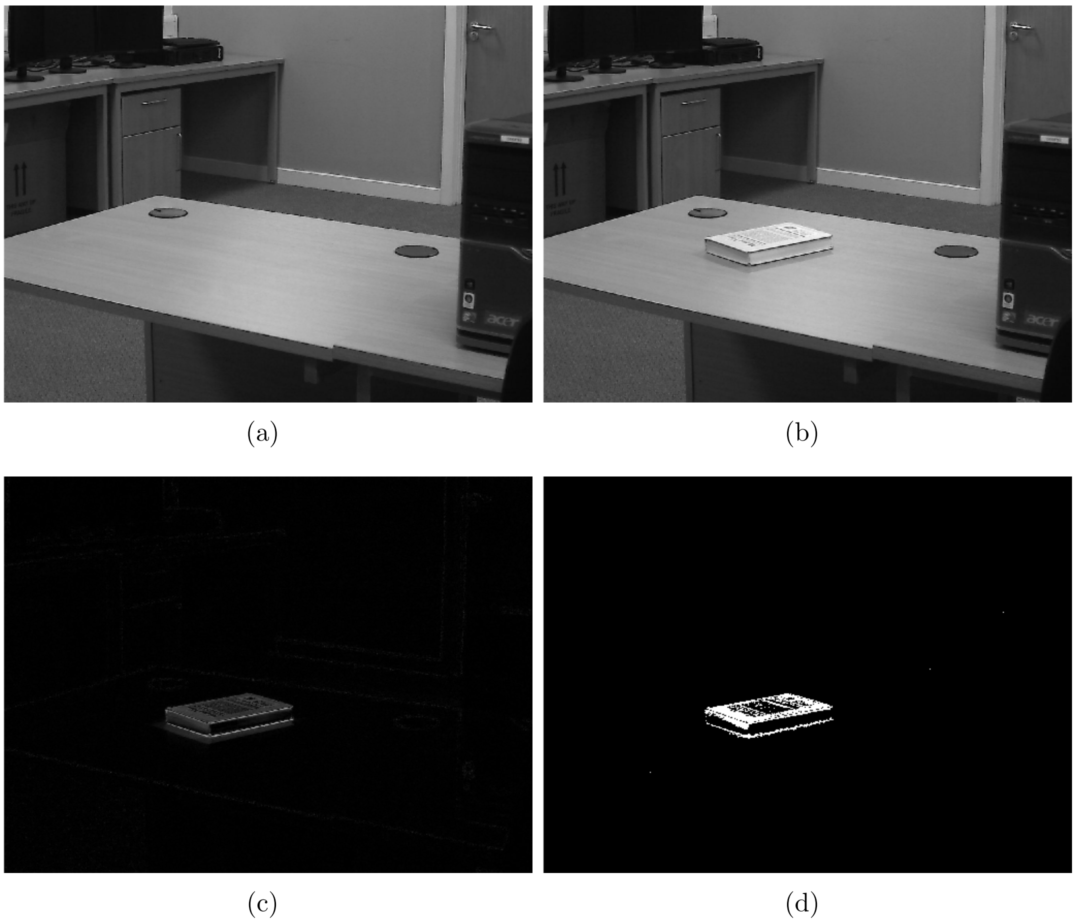


Fig. 2.1: A visual example of background subtraction to detect new static objects in the scene: (a) background $\mathbf{B}_n$; (b) new frame $\mathbf{I}_n$; (c) difference image $\mathbf{D}_n$; (d) binary mask $\mathbf{M}_n$ obtained with $T_n = 44$.

2.5.2 Issues

Although the general formulation of background subtraction is simple and effective, there exist some practical issues that make its application to real, generic video sequences challenging. These issues are: background modelling and updating; threshold value setting; and classification between abandoned and removed objects.

Background modelling and updating

Background modelling refers to the way in which the background is represented and how temporal changes are incorporated in it. From the background model it is then possible at any time n to obtain the image $\mathbf{B}_n$, which is the synthesised view of the static background to be used in (2.3).

As reported in [12, 13], many background models have been proposed in the literature. The most commonly used ones include:

- Gaussian Mixture Model (GMM) [86]: the background model is represented by a series of K Gaussian distributions for each pixel in the image. Every new pixel value in the new frame $\mathbf{I}_n$ is sought to match one of the K distributions. If it does, a weight for the matching distribution is increased. If it does not, a new distribution is added to the set and the one with the lowest weight is discarded. GMM performs well in scenarios where background objects have recurrent motion, such as tree branches and leaves. However the updating rate cannot be controlled directly and the number, K, of distributions is set arbitrarily.

- Median filter: the authors in [87] adopt a separate median filter associated with each pixel in the image, over a sampled set of the last $n\Delta t$ images, with typically $\Delta t = 10$. The value of each pixel in $\mathbf{B}_n$ converges to its real background value and outliers due to moving objects are discarded automatically by the median filtering operation. A clear drawback is that the algorithm has to store n images and performing median filtering on each pixel is computationally expensive. To solve this problem, McFarlane and Schofield [88] use an approximate median, where the background model is represented by a single image: if a new pixel value is higher than in the background image, the latter is incremented by one; instead it is decremented by one if lower. Nonetheless, in both approaches the updating rate strictly depends on the video data itself.

- Running average: the authors in [89] use a running average for each pixel in the image, assuming that each background pixel value has a Gaussian distribution. Instead Jung [90] proposes a metrically trimmed mean, to save computation time. Once again, the updating rate cannot be controlled directly and also the weights for the averaging operation are set arbitrarily.

Although these methods perform well in tracking applications and in the detection of abandoned and removed objects, they are not really designed to be able to trigger an alert within a specific time interval defined by the user, after new abandoned or removed object events have occurred. On the contrary, it is the background model updating procedure that implicitly dictates when the alerts are triggered. In other words, it is the background update that causes the alert, rather than the alert itself to cause the background update.

Threshold setting

Good segmentation depends on both the background image $\mathbf{B}_n$ and the threshold $\mathbf{T}_n$. In fact, if the value chosen for $\mathbf{T}_n$ is too low, high values in $\mathbf{D}_n$ due to noise introduced by the acquisition device may appear in $\mathbf{M}_n$ as false positives; while if $\mathbf{T}_n$ is too high, portions of real new objects in the scene may be erroneously detected as background. This can be seen for example in Fig. 2.1(d), where the new object, i.e. the book, is segmented into two parts, since its middle section is considered as background, due to a high threshold in this case.

Therefore it is clear that the selection of a value for the threshold $\mathbf{T}_n$ is also an important factor in background subtraction. Unfortunately no general rule exists and usually $\mathbf{T}_n$ is left to the user to set arbitrarily, as in [49, 86, 87, 89, 90], to best suit the current application and video sequence under analysis.

Abandoned and removed object classification

Although abandoned and removed objects can be both detected using background subtraction, it is important in some applications to be able to distinguish between the two cases. While such a task is trivial for a human observer, it is more difficult for a computer application, since computer systems cannot easily extract contextual information from an image and infer high level meaning from it. Even though it would be possible to train a classifier, and apply object recognition to distinguish between abandoned and removed objects, it is not a practical solution in a video surveillance system, mainly because such an approach requires time consuming training and it is not generalisable to any object. Moreover the decision process is normally required to be fast as well as effective.

Not many methods have been proposed in the literature to solve this issue and generally the approach given by Connell *et al.* [91] is used, where the energy in $\mathbf{I}_n$ on the contour of regions detected in $\mathbf{M}_n$ is computed. If this energy is higher than a certain threshold, it means that a contour corresponds to real edges, which in turn indicates the presence of a real new object in the scene. Although being conceptually sound, the drawback of this approach is that a threshold must still be defined. Instead, in [70], morphology and region growing are used to detect texture discontinuities in $\mathbf{I}_n$ between the detected regions in $\mathbf{M}_n$ and the surrounding background. This approach is more heuristic than [91] and also slower, as the iterative region growing process is computationally more expensive.

2.6 Conclusion

In this chapter an overview has been given of modern video surveillance technologies and trends, and video analytics, which is the automatic video content analysis of surveillance feeds, has been introduced. Some common techniques for image and video processing usually implemented in VA algorithms have been described. Finally a general description has been given of typical VA tasks and related techniques proposed in literature, with a particular emphasis on abandoned and removed object detection.

Chapter 3

Visual target tracking

3.1 Introduction

Central to many video surveillance systems is target tracking, which is the detection and identification of an object and computation of its motion in consecutive frames. Such a video analytics task is the first step to any higher level analysis, as for example action recognition and behaviour analysis. Due to the importance of this topic, very many approaches have been proposed in the literature. As the novel tracking algorithms described in this thesis are not tailored to any specific type of objects, this chapter reviews general tracking algorithms, as opposed to specific ones, for example for human body tracking [92–95] or crowd [81].

Short reviews on general target tracking can be found in [2–4], while an extensive survey is given in [1]. A more theoretical overview of the video tracking problem is provided in [37].

In the rest of this chapter, Section 3.2 explains the main characteristics of a video tracker. Section 3.3 reviews tracking algorithms based on template matching, while Section 3.4 reviews methods based on the mean shift tracker. These two sections are preliminary to put in context the two novel tracking algorithms described in Chapter 4 and Chapter 5 respectively. Section 3.5 discusses some metrics commonly used to assess the performance of tracking algorithms. Section 3.6 concludes the chapter.

3.2 Tracker characterisation

Although video trackers are usually application-specific, it is possible to identify three main aspects to characterise a tracker in a general sense:

- visual features used;

- target representation adopted;
- localisation method employed.

3.2.1 Visual features

As reported in Section 2.3.2, several different features can be selected in a tracking application, as for example colour, gradients, edges, texture and orientation. Nonetheless, it should be noted that a selected feature, or set of features, must be discriminative for the chosen target to track. In other words, such features must be characteristic for the target, so that it can be distinguished from the rest of the image with relatively high confidence. For example colour is a discriminative feature to track a white football on a green pitch, while it is certainly not discriminative if one wants to track different football players from the same team wearing the same jersey.

3.2.2 Target representation

The target representation, or model, encloses target information useful for the tracking application. Maggio and Cavallaro [37] distinguish between shape representation and appearance representation.

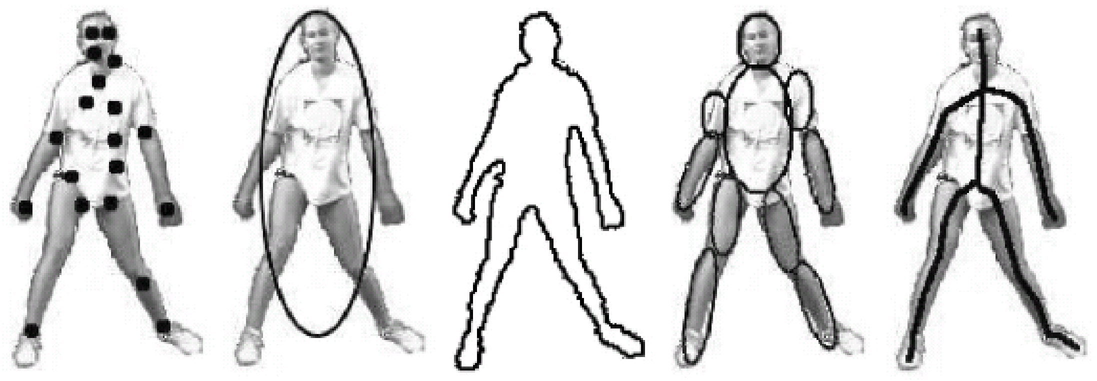
Shape representation

Such a representation simply defines the shape of the target in an image and it can be divided into three main groups:

1. basic models: the target shape is approximated with simple geometric figures, such as single points, rectangles or ellipses, cuboids or ellipsoids.
2. articulated models: the target object is represented as a set of points forming a skeleton, or as a set of primitive shapes linked together.
3. deformable models: in this case an object undergoes deformations which cannot be modelled with joint-like approximations. Deformable models include contours and object silhouettes, or point distribution models in the form of 2D mesh grids [96].

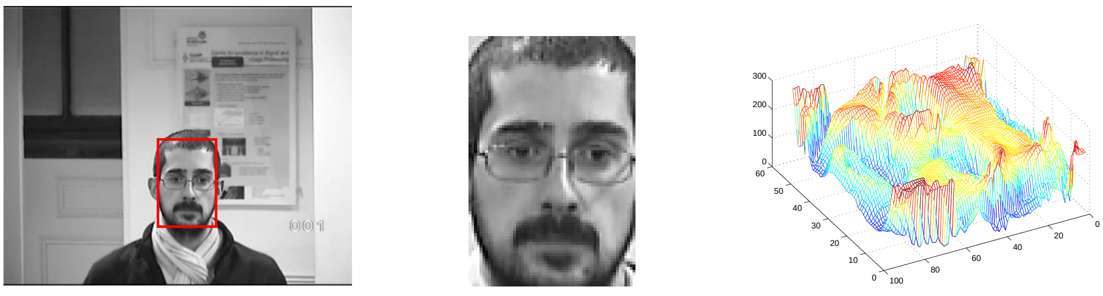
Similar to [37], Yilmaz *et al.* [1] divide shape representations into:

- points;



(a) (b) (c) (d) (e)

Fig. 3.1: Possible shape representations as suggested in [1]: (a) points; (b) primitive shape; (c) contour; (d) articulated model; (e) skeleton.



(a) (b) (c)

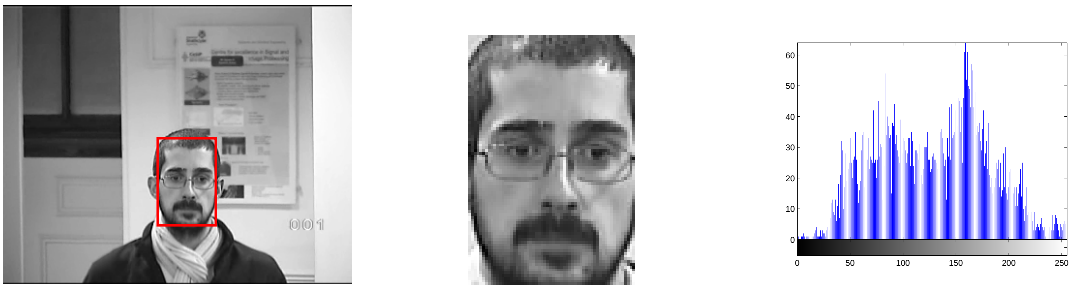
Fig. 3.2: Template representation: (a) target selected within frame; (b) close-up of the target; (c) target template, seen as a 2D discrete function.

- primitive geometric shapes;
- silhouettes and contours;
- articulated shape models;
- skeletal models.

as illustrated in Fig. 3.1. These five different representations are selected depending on the type of object to track: small objects are best represented with single points; rigid objects can be enclosed in primitive shapes, while non rigid objects are best represented with contours, articulated and skeletal models.

Appearance representation

As stated in [37], the appearance representation is the "model of the expected projection of the object appearance onto the image plane". So the appearance



(a) (b) (c)

Fig. 3.3: Histogram representation: (a) target selected within frame; (b) close-up of the target; (c) histogram of the target.

representation models how the target should look in terms of 2D characteristics in a video frame.

Commonly used appearance representations are:

- template: the value of a chosen target feature in a pixel location is stored along with its positional information. For example, if the selected feature is colour, the template is simply an image of the target and it can be obtained by manual or automatic selection in one of the video frames. As such, the target template can be seen as a 2D discrete function or matrix (Fig. 3.2).

- histogram: the value of a target feature in a pixel is stored without its positional information, in the form of a statistical distribution, i.e. a histogram (Fig. 3.3). Although some of the original target information is lost, histogram representations are more robust than a simple template with respect to scaling, rotation and partial occlusion.

3.2.3 Localisation

Localisation, or object tracking, is strictly speaking, the process of finding the position of the target in consecutive video frames in order to track the target motion over time.

Maggio and Cavallaro [37] suggest a division of tracking methods into two major classes, depending on whether they deal with only one track candidate at any time (single-hypothesis localisation, SHL); or they manage multiple track candidates simultaneously (multi-hypothesis localisation, MHL).

Differently from [37], Yilmaz *et al.* [1] classify object tracking methods from another perspective, which depends on their shape representation, as they argue that the chosen target representation, and in particular the shape, greatly affects the motion model assumed for the target, and its possible deformations in a image. Therefore they divide object tracking into three categories:

- Point tracking: the current frame is processed first, regardless of the target object appearance, to extract feature points, and then correspondence between such points and target feature points extracted from the previous frame is performed. Such an operation can be deterministic, i.e. based on some predefined motion constraints, or it can be probabilistic, i.e. based on object state estimates, as for example using Kalman and particle filters. This extract-and-associate approach can generalise to entire blobs and regions, which are initially extracted from a frame using segmentation methods, as reported in Section 2.3.1.

- Kernel tracking: the target object is represented by primitive regions, whose motion is expressed as a parametric transformation, such as translation, rotation and affine, and it is sought between consecutive frames. The motion computation method is strictly related to the appearance representation used, while the computed motion actually depends on the object appearance within the region. Yilmaz *et al.* further divide kernel tracking into template-based and density-based, depending on the appearance representation. The most common template-based approach is template matching, where the object model is represented by a template, which is then searched within the current frame. In the density-based approach, the target model is represented by some form of probability density function (PDF); usually its colour histogram. A widely used density-based approach is the mean shift tracker [11], where the model is encoded in a weighted colour histogram.

- Silhouette tracking: it refers to situations where the target object cannot be well represented by a primitive shape, for example in people tracking and, more generally, for non rigid objects. In this case the target is accurately described by its silhouette or contour. Yilmaz *et al.* divide silhouette tracking into shape matching, where the object shape is searched for within the next frame to find possible matches, and contour tracking, where the initial target object contour evolves from frame to frame, usually by minimising some form of energy constraint function.

3.3 Template matching

In template matching the target model is a template, which stores feature value and positional information for every pixel within the selected primitive shape. Generally speaking, template matching has many applications in image and video processing:

- registration and disparity computation in stereo images [97];
- motion estimation for video coding [98];
- object detection within an image [76];
- optical flow computation [46].

In tracking applications, template matching is used to find the best match for the given template in consecutive frames, i.e. tracking a target in a video sequence [99, 100].

Usually pixel intensities are used as the feature and therefore the template $\mathbf{T}$ of $M_T \times N_T$ pixels reduces to an image of the target. A match for $\mathbf{T}$ is then searched for within the frame, by minimising or maximising, a chosen metric. The two main characteristics of a tracking algorithm based on template matching are the search method used and the matching metric adopted.

3.3.1 Search methods

Concerning the search method, one can use a gradient descent method, as in [101], to minimise the error between the target and its best estimate within the image. As an alternative, one can use a direct search approach, i.e. template $\mathbf{T}$ is compared with the given frame $\mathbf{I}$ at a series of locations within $\mathbf{I}$, as done in motion estimation for video coding [98]. The easiest approach is full search, where template $\mathbf{T}$ is shifted horizontally and vertically in every position within $\mathbf{I}$ and the chosen metric is computed, to evaluate how similar $\mathbf{T}$ is with the portion of $\mathbf{I}$ located at each position. As searching within an entire image is computationally expensive, a smaller search area can be defined as a region of interest (ROI) around the previous target position. Searching in every pixel location is sometimes referred to as a "brute force" approach. Alternative search methods can be found again in motion estimation techniques and the most common ones are:

- hierarchical search: searching is performed on a downsampled version of the image **I**. Once a best match has been found, the resolution of the downsampled image is refined and the best match position is propagated to it. Searching is then performed again, to find a "more accurate" best match. Usually a 2–3 level hierarchy is used.

- N-step search: starting from the centre of the search area, 8 points are taken at a given step size and the one with highest match is chosen as next starting point. The step size is then halved and a second new best match is found. This procedure is repeated N times.

3.3.2 Matching metrics

Concerning the matching metric, a key factor for its selection is computational complexity. In fact, as video applications need to be able to process an entire frame in a few tenths of a millisecond (more precisely ≤ 40 ms, giving ≥ 25 fps, to be considered "real-time"), a matching metric needs to be computationally fast to evaluate. For this reason only few metrics are commonly used in template matching [76]. They include:

- Normalised Cross-Correlation (NCC) [102]: computed as in (3.1), it generally gives good matching performance, but it is computationally expensive, due to the multiplications and square root operations involved in it.

$$NCC(x,y) = \frac{\sum_{m=0}^{M_T-1}\sum_{n=0}^{N_T-1}\left(I'(m,n)-\overline{I}(x,y)\right)\left(T(m,n)-\overline{T}\right)}{\sqrt{\sum_{m=0}^{M_T-1}\sum_{n=0}^{N_T-1}\left(I'(m,n)-\overline{I}(x,y)\right)^2\sum_{m=0}^{M_T-1}\sum_{n=0}^{N_T-1}\left(T(m,n)-\overline{T}\right)^2}} \tag{3.1}$$

where $I'(m,n) = I(x+m, y+n)$ is the block within **I** located at (x,y). $\overline{T}$ and $\overline{I}(x,y)$ are the mean values respectively of **T** and $I'(m,n)$, computed as:

$$\overline{I}(x,y) = \frac{\sum_{m=0}^{M_T-1}\sum_{n=0}^{N_T-1} I(x+m,y+n)}{M_T N_T} \tag{3.2}$$

$$\overline{T} = \frac{\sum_{m=0}^{M_T-1}\sum_{n=0}^{N_T-1} T(m,n)}{M_T N_T} \tag{3.3}$$

- Sum of Absolute Differences (SAD): computed as in (3.4), it has slightly worse performance than NCC, since the sign of the error is not taken into account; however it is easier to implement especially in hardware, as it uses only integers.

$$SAD(x, y) = \sum_{m=0}^{M_T-1} \sum_{n=0}^{N_T-1} |I(x+m, y+n) - T(m, n)| \tag{3.4}$$

- Sum of Squared Differences (SSD): computed as in (3.5), it generally has the same matching performance as SAD, but it is clearly more expensive computationally.

$$SSD(x, y) = \sum_{m=0}^{M_T-1} \sum_{n=0}^{N_T-1} (I(x+m, y+n) - T(m, n))^2 \tag{3.5}$$

3.3.3 Issues

Despite being very popular in disparity computation, optical flow and motion estimation, template matching is not so common in tracking applications, due to two well-known issues: visual changes and occlusion.

Visual changes

It is clear that a tracker using the same template for every frame would not be robust, as normally in a video sequence a target experiences visual changes, due to deformation, rotation, scaling and light changes. For this reason the authors in [99] update the entire template using an infinite impulse response (IIR) filter. However, in case of temporary partial occlusion or when portion of the background are present in the updating patch, parts of the occluding object or of the background are incorporated in the template, which may then become corrupted and irrecoverable.

Partial and complete occlusion

As template matching simply minimises, or maximises, a matching metric, it "always finds" a best match within a frame, even though the real target is completely occluded or not visible. In this case the best match refers to the wrong object and therefore the tracker starts following an incorrect target. The authors in [100] divide the target template into overlapping fragments to overcome partial occlusion; NCC template matching is then replicated to all the fragments and these results are then averaged. Nonetheless, the tracker might still lose the correct target in case the best match is selected in correspondence of another object. A possible solution to this problem is to set an arbitrary threshold, above which the match is considered to be incorrect. Although this approach is sound, it is difficult to manually tune such a threshold. Moreover, not all the metrics are "normalised" (SAD and SSD), and therefore the threshold value cannot be expressed as a percentage. In these cases the threshold value strictly depends on the visual characteristics of the chosen target, reducing the generality of this approach.

3.4 Mean shift tracker

In visual systems, colour is a very strong cue, as it is invariant to scaling and rotation. Therefore colour has been used as discriminative feature for tracking by many researchers. For example, McKenna *et al.* [103] use an adaptive mixture of Gaussian distributions in a hue-saturation space. Nummiaro *et al.* [104] integrate colour distributions into a particle filtering framework. Porikli [75] introduces an integral histogram, to compute the histograms of all possible sub-regions in a Cartesian space. However, in more recent years, a very commonly used algorithm for colour tracking has been the mean shift (MS) tracker, by Comaniciu *et al.* [7–11]. The MS tracker adopts a weighted colour histogram distribution as a representation of the target, and tracking is performed by maximising the Bhattacharyya distance from this distribution.

3.4.1 Formulation

In the conventional MS tracker the target model has a rectangular shape and its appearance is represented by the target colour histogram $\mathbf{Q}$ [11]. The selected target is initially identified as a set of pixels $\mathbf{x}_j \in \Gamma_1$ in a colour frame $\mathbf{F}_0$ of size $H \times W$, as illustrated in Fig. 3.4. A rectangular target area of $h_0 \times w_0 = N_0$ pixels $\mathbf{x}_n \in \Gamma_2$ with centroid $\mathbf{y}_0$ can be defined, to enclose the target pixels $\mathbf{x}_j$, i.e. $\Gamma_1 \subseteq \Gamma_2$, where $\mathbf{x}_j$, $\mathbf{x}_n$ and $\mathbf{y}_0$ are 2-dimensional indices such that $\mathbf{x}_j, \mathbf{x}_n, \mathbf{y}_0 \in \Gamma_2 \subseteq [1, H] \times [1, W] \subseteq \Re^2$.

For each pixel $\mathbf{x}_n$, a difference vector $\mathbf{x}_n^0 = \mathbf{y}_0 - \mathbf{x}_n$ that goes from pixel $\mathbf{x}_n$ to the target area centroid $\mathbf{y}_0$ is computed. Moreover, the function $g(h_0, w_0, \mathbf{x}_n^0)$ is defined as:

$$g(h_0, w_0, \mathbf{x}_n^0) = \frac{h_0 w_0}{\sqrt{(h_0 \cos(\angle \mathbf{x}_n^0))^2 + (w_0 \sin(\angle \mathbf{x}_n^0))^2}} \tag{3.6}$$

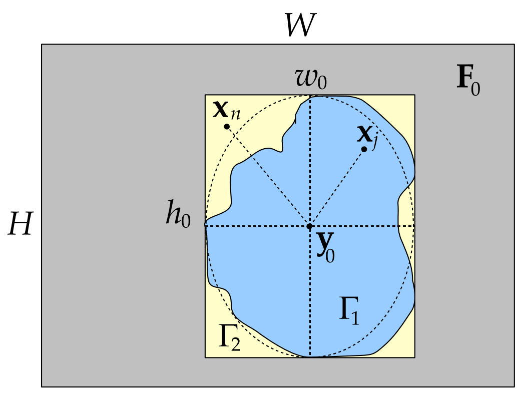


Fig. 3.4: Target area Γ_2 enclosing target pixels $\mathbf{x}_j \in \Gamma_1$, within the frame $\mathbf{F}_0$.

In (3.6), $g(\cdot)$ gives the length of a vector that goes from the target area centroid $\mathbf{y}_0$, to the ellipse with axes h_0 and w_0, inscribed in the target area Γ_2, as illustrated in Fig. 3.4.

In an RGB representation, $\mathbf{F}_0$ has three colour planes and the target histogram $\mathbf{Q}$ is 3-dimensional, with β bins for each dimension. The value of the histogram bin q_{rgb} in $\mathbf{Q}$, with $\mathbf{u} = [r, g, b]$ and $r, g, b \in [1, \beta]$, is computed as:

$$q_{rgb} = C_0 \sum_{n=1}^{N_0} k\left(\frac{\|\mathbf{x}_n^0\|^2}{g^2(h_0, w_0, \mathbf{x}_n^0)}\right) \delta[b(\mathbf{x}_n) - \mathbf{u}] \tag{3.7}$$

where $b(\cdot)$ is a function that retrieves the 3-dimensional bin index in the histogram $\mathbf{Q}$ for the colour in $\mathbf{x}_n$, and $\delta[\cdot]$ is a function defined as:

$$\delta[\mathbf{v}] = \begin{cases} 1, & \mathbf{v} = \mathbf{0} \\ 0, & \mathbf{v} \neq \mathbf{0} \end{cases} \tag{3.8}$$

In (3.7), $k(\cdot)$ is the Epanechnikov weighting kernel [58] defined as:

$$k(x) = \begin{cases} \frac{2}{\pi}(1 - x) & \text{if } x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

and C_0 is a normalising factor computed as in (3.10), so that the sum of all the bins q_{rgb} equals 1.

$$C_0 = \left[\sum_{n=1}^{N_0} k\left(\frac{\|\mathbf{x}_n^0\|^2}{g^2(h_0, w_0, \mathbf{x}_n^0)}\right)\right]^{-1} \tag{3.10}$$

The magnitude $g(h_0, w_0, \mathbf{x}_n^0)$ in (3.7) and (3.10) allows one to define an ellipsoidal smoothing kernel $k(\cdot)$, with an extension given by the ellipse inscribed in the rectangular target area Γ_2. Note that when $h_0 = w_0$, Γ_2 is a square and

$g(h_0, w_0, \mathbf{x}_n^0)$ describes a circle.

For every new frame $\mathbf{F}_i$, the target position $\mathbf{y}_{i-1}$ in the previous frame $\mathbf{F}_{i-1}$ is used as initial estimated target position in $\mathbf{F}_i$. The histogram bin $p_{rgb}(\mathbf{y}_{i-1})$ of the weighted colour distribution $\mathbf{P}(\mathbf{y}_{i-1})$ in $\mathbf{F}_i$ is computed as:

$$p_{rgb}(\mathbf{y}_{i-1}) = C \sum_{n=1}^{N_{i-1}} k \left(\frac{\|\mathbf{x}_n^{i-1}\|^2}{g^2(h_{i-1}, w_{i-1}, \mathbf{x}_n^{i-1})} \right) \delta[b(\mathbf{x}_n) - \mathbf{u}] \tag{3.11}$$

where $\mathbf{x}_n^{i-1} = \mathbf{y}_{i-1} - \mathbf{x}_n$, C is a normalising factor defined as:

$$C = \left[\sum_{n=1}^{N_{i-1}} k \left(\frac{\|\mathbf{x}_n^{i-1}\|^2}{g^2(h_{i-1}, w_{i-1}, \mathbf{x}_n^{i-1})} \right) \right]^{-1} \tag{3.12}$$

and h_{i-1} and w_{i-1} are height and width of the target area of $h_{i-1} \times w_{i-1} = N_{i-1}$ pixels. The target candidate distribution $\mathbf{P}(\mathbf{y}_{i-1})$ is used to evaluate the weights a_n:

$$a_n = \sum_{r=1}^{\beta} \sum_{g=1}^{\beta} \sum_{b=1}^{\beta} \sqrt{\frac{q_{rgb}}{p_{rgb}(\mathbf{y}_{i-1})}} \delta[b(\mathbf{x}_n) - \mathbf{u}] \tag{3.13}$$

Such weights allow one to compute the mean shift vector [10, 11] and therefore the new estimated target position $\mathbf{y}_i$:

$$\mathbf{y}_i = \frac{\sum_{n=1}^{N_i} \mathbf{x}_n a_n}{\sum_{n=1}^{N_i} a_n} \tag{3.14}$$

This procedure iteratively maximises the Bhattacharyya coefficient $\rho(\mathbf{y}_i)$:

$$\rho(\mathbf{y}_i) = \sum_{r=1}^{\beta} \sum_{g=1}^{\beta} \sum_{b=1}^{\beta} \sqrt{p_{rgb}(\mathbf{y}_i) q_{rgb}} \tag{3.15}$$

and consequently it minimises the distance $d(\mathbf{y}_i)$:

$$d(\mathbf{y}_i) = \sqrt{1 - \rho(\mathbf{y}_i)} \tag{3.16}$$

The final target position $\mathbf{y}_i$ in $\mathbf{F}_i$ is then used as initial estimated target position in the next frame $\mathbf{F}_{i+1}$.

3.4.2 Issues

The conventional MS tracker has been shown to be fast, effective and robust in many scenarios, and many extensions have been proposed to further improve its performance. Leichter *et al.* [105] propose the combination of multiple histograms of the target, to capture its appearance from different views. Wang *et al.* [106] propose a fragment-based representation of the target, with a separate tracker associated with each fragment. In [107], SIFT features are used along with the MS tracker, while in [108] the target histogram is fitted around a simple target structure representation. In [109, 110], MS tracking is integrated into particle filtering, while in [111] it is used to carry out the measurement in a Kalman filter framework. In [112], the MS is integrated with edge information and is applied to infrared images. Edge orientations are used also in [110]. In [113], the MS is used in the context of a multi-target tracking system, where re-initialisation is executed periodically, over foreground detections obtained with background subtraction.

Nonetheless, the conventional MS tracker suffers from some well-known issues, which are: asymmetrical kernel shape; scaling and rotation; occlusion; fast target loss.

Asymmetrical kernel shape

Comaniciu *et al.* [7–11] use in their works a symmetrical Epanechnikov kernel, as it is convex and monotonic and such characteristics ensures convergence of the MS gradient descent iteration, as demonstrated in [8, 9]. Although it is easy to generalise the formulation from circular to ellipsoidal, the shape of real targets does not fit well such a precise geometrical representation. To obviate this problem, Yilmaz [114] proposes the use of asymmetrical kernels based on a level set function, so that the tracking accuracy for arbitrary-shaped objects can be greatly improved. However, as Yilmaz states in his work, the constancy of the kernel shape remains a problem. Also the authors in [115] propose the use of asymmetrical kernels, but their mathematical formulation is not as rigorous as in [114] and the results shown are not exhaustive.

Scaling and rotation

In the conventional MS tracker [7–11] the basic iteration is executed three times for each frame, adopting three different kernel bandwidths, i.e. the previous bandwidth h_{prev}, a larger one $h^{+} = 1.1h_{prev}$ and a smaller one $h^{-} = 0.9h_{prev}$. Of the three bandwidths, the one that gives the highest match for the same target model is chosen as new kernel bandwidth. For rotation, no suggestion is given on how to deal with such scenario and target orientation is implicitly assumed to be always the same. A very interesting method to deal with both scaling and rotation is described by Bradski [116] in his Continuously Adaptive Mean Shift (CAMShift). This algorithm is mainly focused on face tracking and it aims at providing a perceptual user interface based on user head movements. Scale and rotation of the target face are computed through colour moments in a probability backprojected image, and they are continuously adapted, hence the name of the algorithm. A similar approach is described in [117], where the authors use the distributions of both target model and target candidate to estimate scale and orientation. A very different approach is proposed by Collins [118] to deal with target scaling. In this work a scale space is created by convolving the image with a filter bank of Difference-of-Gaussian spatial kernels and then the correct scale is sought within this space.

Occlusion

Although being robust to small partial occlusion, the MS tracker may fail when the target undergoes severe or complete occlusion: in the former case the MS gradient descent procedure might get trapped in a temporary local maximum; in the latter, the estimated density function has no maximum associated with the correct target and all the other local maxima are therefore incorrect by definition. Nonetheless, if the temporal length of the occlusion is short and, most importantly, the correct target reappears in a neighbourhood of its last tracked position, the MS tracker might be able to recover and continue to track the correct target. In [7–11], Comaniciu *et al.* suggest employment of a motion filter, for example a Kalman filter, when occlusion is present. In the context of MS tracking, a few approaches have been proposed in literature to deal with severe and complete (short) occlusion. In [119] the authors use SIFT features to find the next target position first and then apply MS to increase the precision of the detection. In this way, local maxima are avoided and the tracker can recover after short occlusion. Particle filtering is integrated with MS in [110, 120], to predict the next target position after occlusion. In [111], a Kalman filter is used instead. In [121] Li *et al.* deal with a simple partial occlusion scenario, i.e. single pedestrians occluding each other, by tracking both occluded and occluding objects. Similarly, the approach described in [122] is for a single target undergoing short occlusion. In [123], the MS tracker is applied to football video sequences.

In this case, occlusion is dealt with specifically, depending on whether it concerns players from the same or different teams, and the next position is computed with a simple prediction scheme.

Fast target loss

As pointed out in [124], a fast target can either be an object experiencing large spatial displacement in a high frame rate video, or a slow moving object in a low frame rate video. As the conventional MS tracker uses a support kernel, a so-called "basin of attraction" results automatically defined for the MS gradient descent procedure to converge, as explained in [11]. This means that, for the MS tracker to work, detections of the target, i.e. the target areas around the tracked target centroids, must overlap in consecutive frames. It is thus clear why the conventional MS algorithm fails in the case of fast moving targets. In [11], Comaniciu *et al.* suggest the use of a kernel bandwidth larger than the actual target size, to have a wider basin of attraction. To compensate for background colours now present in the larger kernel, the target histograms are normalised with respect to the histogram of the background (as proposed in [125]) in a region around the target. Although this approach is sound and is shown to be effective, it is not always practical to apply. In fact, in the case of a small target with fast motion, i.e. moving from one side of the image to the other over a single frame, one would have to use a kernel bandwidth almost as large as the entire frame. In [124], Porikli and Tuzel apply the basic MS procedure to multiple candidate locations in the frame, to recover the target in low frame rate videos. For this purpose a statistical background model is maintained and the candidate locations are selected from a foreground mask. The drawback in this approach is that a background model is required. Moreover, it is not always possible to have a foreground mask where the intended target is the only moving object, or is, at least, well separated from other moving objects. For example, in highly crowded video sequences, it is difficult to construct a background model and foreground masks present large unsegmented blobs. In [126], to predict the kernel position, a backprojection image is created, from the target colour distribution, and then it is divided into blocks. Of these blocks, the one that maximises the zero-th moment (i.e. it maximises the sum of all the pixel values in the block) is chosen as the new kernel position in the new frame. Although the authors in [126] claim this approach to be effective, they do not analyse cases where a zero-th moment maximum does not correspond to the correct target.

3.5 Evaluation of tracking performance

In tracking applications it is important to be able to evaluate the performance of a tracking algorithm. In general this is done by comparing, for a given reference video sequence, the tracking results obtained with the algorithm under examination and the ground truth, which is a set of pre-computed reference results for that particular video sequence.

There are different ways and metrics to assess tracking performance, depending on the particular characteristics of an algorithm one wants to evaluate. A simple approach is to compute in each frame how close the target position computed by the tracker is, with respect to the reference position available in the ground truth. If $\mathbf{p}_n$ and $\mathbf{r}_n$ are respectively computed and reference target positions in the n^{th} frame $\mathbf{F}_n$, the absolute error is computed as Euclidean distance e_n:

$$e_n = |\mathbf{p}_n - \mathbf{r}_n| \tag{3.17}$$

The error can be computed for every frame with $n \in [0, N-1]$ in the video sequence, and mean value and standard deviation are:

$$\mu_e = \frac{1}{N} \sum_{i=0}^{N-1} e_n \tag{3.18}$$

$$\sigma_e = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (e_n - \mu_e)^2} \tag{3.19}$$

In this case, the mean value μ_e represents the accuracy of the tracker, while the standard deviation σ_e represents its precision.

Another approach is to determine if the position retrieved by the tracker in a frame corresponds to the correct target ("true positive", TP), or to an incorrect object ("false positive", FP). On the other hand a target can be correctly detected as not present (or visible) in the frame ("true negative", TN), or incorrectly undetected ("false negative", FN), i.e. the target is actually present and visible in the frame, but the tracker has not been able to find it. With TP, TN, FP and FN four metrics are defined [37]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.20}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.21}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.22}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.23}$$

Such metrics are useful to evaluate the ability of a tracker to handle severe and complete occlusion, as in some frames a target may be covered by other objects in the scene and therefore be not visible. Similarly they can evaluate the ability of a tracker to detect when a target has left the field of view (FOV) (and therefore it is not present in the frame) and when it is back in the FOV.

3.6 Conclusion

This chapter has reviewed tracking techniques for video applications. The three main aspects of a video tracker are: the features used to characterise the target; the mathematical model adopted to describe the target appearance; the localisation method employed to compute the target motion between consecutive frames. A detailed overview of template matching and mean shift tracking has been given, as these techniques are the basis for the two novel tracking algorithms presented respectively in Chapter 4 and Chapter 5. Some performance metrics have been introduced to assess the tracking performance of such novel algorithms.

Chapter 4

Adaptive tracker based on SWAD minimisation

4.1 Introduction

The Sum of Absolute Differences is a well-known metric and it is often used in video coding and disparity computation for stereo images, due to its simplicity. However SAD is not very common in tracking applications, as issues such as partial occlusion and target template changes can dramatically affect its performance. In this chapter, a novel adaptive template matching algorithm for robust target tracking based on a modified SAD called Sum of Weighted Absolute Differences (SWAD) is presented, where a Gaussian weighting kernel is employed to reduce the effects of partial occlusion [127]. Simulation results demonstrate that the presented SWAD-based tracker outperforms conventional SAD in terms of efficiency and accuracy.

When occlusion takes place, there exists the risk for adaptive template tracking algorithms to include portions of the occluding objects in the target detection. In the next template the updating step would then incorporate visual characteristics of such occluding objects. Eventually the target template would become completely corrupt and irrecoverable as the tracker starts following an incorrect target. To solve this issue in the SWAD-based tracker described in this chapter, an occlusion handling technique is presented to update portions of the target template at different rates and so preserve its overall structure [128]. Experimental results show that such an approach can further improve the robustness of the SWAD-based adaptive template matching tracker, by allowing it to successfully recover the correct target after severe and complete occlusion.

The rest of this chapter is organised as follows. Section 4.2 describes in detail

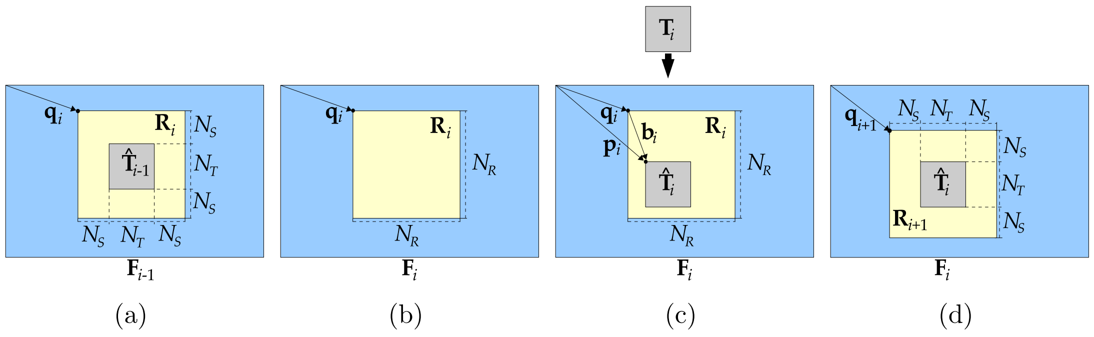


Fig. 4.1: Layout of the current frame $\mathbf{F}_i$ showing the best match $\hat{\mathbf{T}}_i$, the region of interest $\mathbf{R}_i$ and their positions within $\mathbf{F}_i$.

the novel SWAD-based tracking algorithm, while the occlusion handling procedure is described in Section 4.3. Section 4.4 reports on performance evaluation of the tracker, including the description of a DSP embedded implementation of it, to demonstrate that its simplicity and algorithmic structure make it suitable for real-time implementation in embedded systems with fixed-point architecture. Section 4.5 concludes the chapter.

4.2 Adaptive template matching target tracking

The novel tracking algorithm presented in this chapter is based on template matching, and in particular on the minimisation of a Sum of Weighted Absolute Differences (SWAD) in a region of interest (ROI) $\mathbf{R}_i$ within the current frame $\mathbf{F}_i$, as shown in Fig. 4.1(b). The target model is represented by a template $\mathbf{T}_i$ of $N_T \times N_T$ pixels, computed as:

$$\mathbf{T}_i = (1-\alpha)\mathbf{T}_{i-1} + \alpha\hat{\mathbf{T}}_{i-1} \tag{4.1}$$

where $\alpha \in [0,1]$ is a blending factor, and $\mathbf{T}_{i-1}$ and $\hat{\mathbf{T}}_{i-1}$ are respectively target template and best match in the previous frame $\mathbf{F}_{i-1}$ (Fig. 4.1(a)). In the first frame $\mathbf{F}_0$ the target template $\mathbf{T}_0$ is initialised as the portion of $\mathbf{F}_0$ containing the selected target. The ROI $\mathbf{R}_i$ in position $\mathbf{q}_i$ is selected as the neighbouring area around $\hat{\mathbf{T}}_{i-1}$ in the previous frame $\mathbf{F}_{i-1}$, with size of $N_R \times N_R$ pixels, where $N_R = N_T + 2N_S$ and N_S is a spatial offset around $\hat{\mathbf{T}}_{i-1}$. The proposed algorithm requires single colour plane images. It can therefore be applied to both grey scale and colour video sequences. In the case of colour images, $\mathbf{F}_i$ can be the luminance component. The absolute differences in the SWAD metric are weighted by a kernel $\mathbf{K}$ of $N_T \times N_T$ pixels. In the current implementation a Gaussian weighting kernel

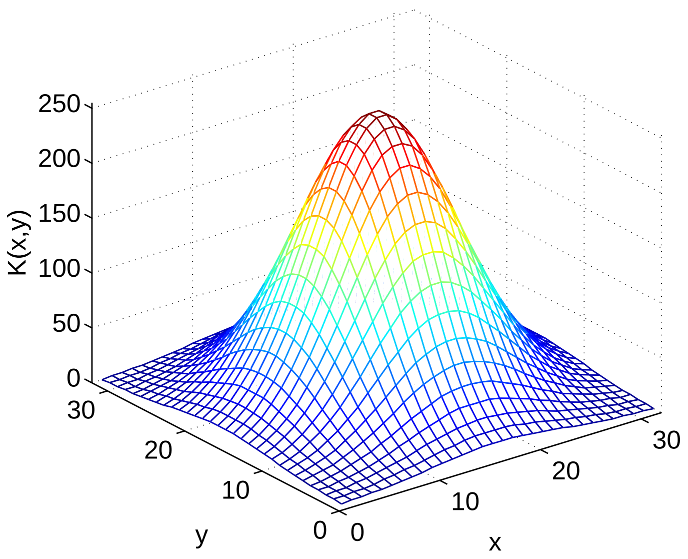


Fig. 4.2: Weighting kernel $\mathbf{K}$ of $N_T \times N_T$ pixels as in (4.2), with $N_T = 32$.

has been employed, to assign high weights to central pixels and low weights to peripheral ones, as these pixels might belong to background or even occluding objects. Assuming integer pixel values in the range $[0, 255]$, the kernel $\mathbf{K}$ is:

$$K(x, y) = \left\lfloor 255 \times \frac{g(x, y)}{g(\lfloor\mu\rfloor, \lfloor\mu\rfloor)} \right\rfloor \tag{4.2}$$

where $x, y \in [0, N_T - 1]$ and $g(x, y)$ is 2-dimensional Gaussian function with mean $\mu = (N_T - 1)/2$ and experimentally selected standard deviation $\sigma = N_T/5$, defined as:

$$g(x, y) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} - \frac{(y-\mu)^2}{2\sigma^2}\right) \tag{4.3}$$

The kernel $\mathbf{K}$ with $N_T = 32$ is illustrated in Fig. 4.2.

Given the current template $\mathbf{T}_i$, its best match $\hat{\mathbf{T}}_i$ in $\mathbf{F}_i$ is searched within $\mathbf{R}_i$ (Fig. 4.1(c)). For this purpose the SWAD coefficient $\psi(x, y)$ at location (x, y) within $\mathbf{R}_i$ is computed as:

$$\psi(x, y) = \sum_{m=0}^{N_T-1} \sum_{n=0}^{N_T-1} K(m, n)\Delta(x, y, m, n) \tag{4.4}$$

where $\mathbf{K}$ is the Gaussian weighting kernel and $\Delta(x, y, m, n)$ is the pixel difference:

$$\Delta(x, y, m, n) = |R_i(x + m, y + n) - T_i(m, n)| \tag{4.5}$$

The position of the best match $\hat{\mathbf{T}}_i$ within $\mathbf{R}_i$ for the given template $\mathbf{T}_i$ is represented by $\mathbf{b}_i$, which is obtained by minimising the SWAD coefficient as in (4.6):

$$\mathbf{b}_i = \underset{x,y\in[0,N_R-N_T]}{\arg\max} \; \psi(x,y) \tag{4.6}$$

The index $\mathbf{b}_i$ refers to the top-left pixel of the best matching block $\hat{\mathbf{T}}_i$. The position of $\hat{\mathbf{T}}_i$ within $\mathbf{F}_i$ is computed as in (4.7):

$$\mathbf{p}_i = \mathbf{q}_i + \mathbf{b}_i \tag{4.7}$$

as illustrated in Fig. 4.1(c). Once the best match $\hat{\mathbf{T}}_i$ in the current frame has been found, the new target model $\mathbf{T}_{i+1}$ for the next frame $\mathbf{F}_{i+1}$ is computed as in (4.1), which describes an IIR filter and it ensures that the target template is updated and adapts to changes, while guaranteeing tracker stability by reducing the effects of partial occlusion. Moreover, updating the template as in (4.1) implicitly deals with possible rescaling and rotation of the target. The value of the blending factor α mainly depends on the approach one wants to follow to update the target template. In a high frame rate implementation, α can be set to 0.5, giving equal weight to current and past templates, so that $\mathbf{T}_i$ can adapt to changes fairly quickly. For a more conservative approach, i.e. to preserve the template and let it adapt slowly, the blending factor can be $\alpha < 0.5$. The optimal selection of α is application dependant. However, α can be set adaptively as explained in Section 4.3 to deal with complete occlusion. Finally a new region of interest $\mathbf{R}_{i+1}$ for the next frame $\mathbf{F}_{i+1}$ is defined around the position $\mathbf{p}_i$ of $\hat{\mathbf{T}}_i$, within $\mathbf{F}_i$ (Fig. 4.1(d)). In particular the position $\mathbf{q}_{i+1}$ of $\mathbf{R}_{i+1}$ in $\mathbf{F}_{i+1}$ is computed as:

$$\begin{aligned} q_{i+1}(0) &= \min\left(\max\left(p_i(0) - N_S, 0\right), W - N_R\right) \\ q_{i+1}(1) &= \min\left(\max\left(p_i(1) - N_S, 0\right), H - N_R\right) \end{aligned} \tag{4.8}$$

where W and H are respectively width and height of the frame. This ensures that $\mathbf{R}_{i+1}$ is entirely within the frame $\mathbf{F}_{i+1}$.

A block diagram of the proposed algorithm is shown in Fig. 4.3, while its steps for each frame $\mathbf{F}_i$ are summarised as follows:

1. Given $\mathbf{F}_i$, $\mathbf{q}_i$ and $\mathbf{T}_i$, select $\mathbf{R}_i$ from $\mathbf{F}_i$ and minimise the SWAD coefficient $\psi(x,y)$ as in (4.4)–(4.6), to obtain the position $\mathbf{b}_i$ of the best match $\hat{\mathbf{T}}_i$ in $\mathbf{R}_i$.

2. Given $\mathbf{b}_i$ and the position $\mathbf{q}_i$ of $\mathbf{R}_i$ within $\mathbf{F}_i$, compute the target position

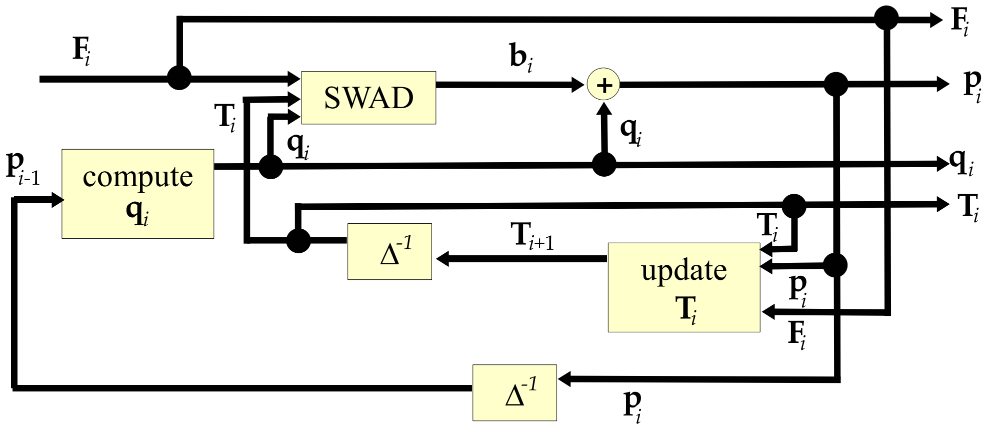


Fig. 4.3: Block diagram of the tracking algorithm described in Section 4.2.

$\mathbf{p}_i$ as in (4.7).

3. Given $\mathbf{F}_i$, $\mathbf{p}_i$ and $\mathbf{T}_i$, select the best match $\hat{\mathbf{T}}_i$ from $\mathbf{F}_i$ and update the target model using (4.1).
4. Compute the position $\mathbf{q}_{i+1}$ of the ROI $\mathbf{R}_{i+1}$ in the next frame $\mathbf{F}_{i+1}$ from (4.8).

In Fig. 4.3, Δ^{-1} represents a delay of one frame.

4.3 Complete occlusion handling

In the adaptive template matching target tracking algorithm described in the previous section, the target template $\mathbf{T}_i$ of $N_T \times N_T$ pixels at the i^{th} iteration is computed as a weighted sum of the previous target template $\mathbf{T}_{i-1}$ and its best match $\hat{\mathbf{T}}_{i-1}$ in the current frame $\mathbf{F}_i$ as in (4.1), where the blending factor $\alpha \in [0, 1]$ is set empirically and is the same for every pixel in $\mathbf{T}_i$. In order to deal with severe and complete occlusion, this section describes the use of a matrix $\mathbf{A}_i$ of blending factors α_i^n, with $\mathbf{A}_i$ having the same size of $\mathbf{T}_i$ and its n^{th} element α_i^n referring to the n^{th} pixel $\mathbf{x}_i^n \in [0, N_T-1]\times[0, N_T-1]$ in $\mathbf{T}_i$, where $n \in [0, {N_T}^2-1]$.

By taking a conservative approach, one can assume that in a good match the difference d_{i-1}^n between pairs of pixels in previous template $\mathbf{T}_{i-1}$ and its best match $\hat{\mathbf{T}}_{i-1}$ obtained as:

$$d_{i-1}^n = \left| T_{i-1}(\mathbf{x}_{i-1}^n) - \hat{T}_{i-1}(\mathbf{x}_{i-1}^n) \right| \tag{4.9}$$

should be less than a given maximum value ϵ_{max}. The main idea behind the occlusion handling technique described here is that for a given pixel $\mathbf{x}_{i-1}^n$, a low value of d^n_{i-1} ($d^n_{i-1} < \epsilon_{max}$) indicates a good match, while a large d^n_{i-1} ($d^n_{i-1} \geq \epsilon_{max}$) indicates a poor match. As a consequence, for poor matching pixels the corresponding blending factors in $\mathbf{A}_{i-1}$ should be set to less than 0.5, while a perfect matching pixel ($d^n_{i-1} = 0$) should have $\alpha^n_{i-1} = 0.5$.

The chosen intensity difference range $[0, \epsilon_{max})$ is divided into L levels of width $\Delta\epsilon = \epsilon_{max}/L$. Similarly the blending range $[0, 0.5]$ is divided into L levels of width $\Delta\alpha = 0.5/L$. For pixel $\mathbf{x}^n_{i-1}$ with difference d^n_{i-1}, the corresponding blending factor α^n_{i-1} in matrix $\mathbf{A}_{i-1}$ is computed as:

$$\alpha^n_{i-1} = \begin{cases} \left(L - \lfloor d^n_{i-1}/\Delta\epsilon \rfloor\right) \Delta\alpha, & d^n_{i-1} < \epsilon_{max} \\ 0, & d^n_{i-1} \geq \epsilon_{max} \end{cases} \tag{4.10}$$

and so the value of the new template $\mathbf{T}_i$ in position $\mathbf{x}^n_{i-1}$ is:

$$T_i(\mathbf{x}^n_{i-1}) = (1 - a^n_{i-1})T_{i-1}(\mathbf{x}^n_{i-1}) + a^n_{i-1}\hat{T}_{i-1}(\mathbf{x}^n_{i-1}) \tag{4.11}$$

In other words, if the best match $\hat{\mathbf{T}}_{i-1}$ obtained by minimising the SWAD coefficient has a small difference from $\mathbf{T}_{i-1}$, the pixel intensity $\hat{T}_{i-1}(\mathbf{x}^n_{i-1})$ is likely to be due to the correct target. When the difference is (very) high, this pixel intensity is likely to be due to an occluding object. In the case $\alpha^n_{i-1} = 0$, the template pixel intensity $T_{i-1}(\mathbf{x}^n_{i-1})$ is retained in the updating formula in (4.11).

It is clear how extending this approach to every pixel in $\mathbf{T}_{i-1}$ allows one to preserve the correct template in case of severe and complete occlusion, as occluded pixels in $\mathbf{T}_{i-1}$ are not influenced by the occluding pixels that might be present in in $\hat{\mathbf{T}}_{i-1}$.

Ultimately, a threshold $\vartheta = \Delta\alpha {N_T}^2$ can be defined, so that when the majority of the pixels in $\hat{\mathbf{T}}_i$ are poor matches and pixels in $\mathbf{T}_i$ are not being updated, the target can be deemed to be completely occluded and therefore not visible. For example:

$$\begin{cases} \text{if } \displaystyle\sum_{n=0}^{{N_T}^2-1} a^n_{i-1} < \vartheta & \Rightarrow \hat{\mathbf{T}}_{i-1} \text{ is not the correct target} \\ \text{otherwise} & \Rightarrow \hat{\mathbf{T}}_{i-1} \text{ is the correct target} \end{cases} \tag{4.12}$$

It should be noted that, in case of misdetection, i.e. the target is occluded but $\hat{\mathbf{T}}_{i-1}$ is still considered as the correct target, the values in $\mathbf{A}_i$ are usually much less than 0.5, so the majority of the pixels in $\mathbf{T}_i$ are not updated and the overall structure of the template is preserved. In this way, the tracker can successfully recover the correct target after complete occlusion.

4.4 Performance evaluation

4.4.1 Tracking performance

Bearing in mind the evaluation criteria for trackers reported in Section 3.5, the performance evaluation of the tracker presented in this chapter comprises two parts: in the first one, the intra-frame tracking capability of the SWAD-based algorithm is assessed, in terms of absolute error; in the second part, the occlusion handling procedure of the tracker is evaluated.

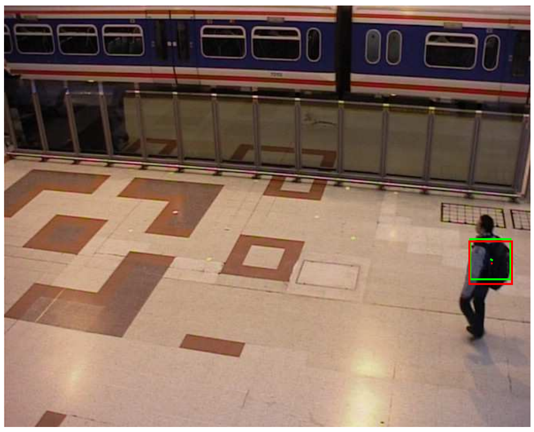
Absolute error distance

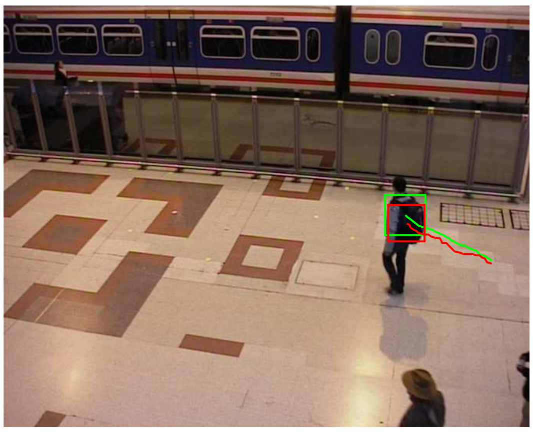
For this experiment, Matlab implementations of SAD-based, NCC-based, conventional MS [11] and SWAD-based trackers have been tested. For the SWAD-based tracker, both versions of the algorithm with occlusion handling (referred to as "SWADOCC") and without (referred to as "SWAD") have been tested. For this purpose three publicly available test sequences have been used: S1-T1-C/3 from PETS2006 dataset [83], S06_2/1 from PETS2007 dataset [84], and Dudek face sequence [129]. As the Dudek sequence is in grey scale, the MS has not been applied to it. The ground truth for the Dudek sequence is already available, while the two PETS sequences have been manually labelled. For each frame $\mathbf{F}_i$, an error e_i is computed in terms of Euclidean distance between the ground truth and the target position returned by the trackers. Visual results are illustrated in Fig. 4.4–4.8 for the PETS2006 sequence, while results for PETS2007 and Dudek sequences are illustrated in Fig. A.1–A.5 and in A.6–A.9 respectively, in Appendix A. In all the figures, boxes highlight the target, while lines represent target tracks. With respect to this, the manually labelled ground truth is shown in green, while the tracker result is shown in red.

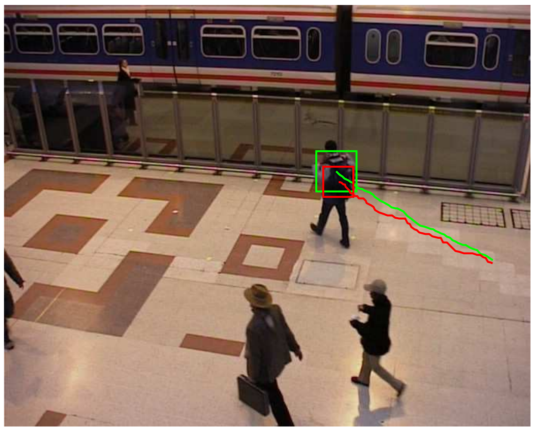
It can be seen that both SAD and NCC lose the target in the two PETS sequences (Fig. 4.5(f), 4.6(f), A.2(f) and A.3(f)), while the MS can track the target correctly (Fig. 4.4 and A.1). On the contrary, in the Dudek sequence, SAD and NCC can correctly track the selected target (Fig. A.6 and A.7), while the MS tracker cannot be used in this case.

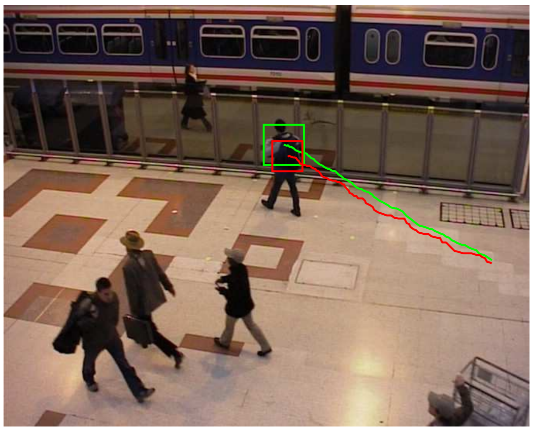
Instead it can be seen that SWAD and SWADOCC can successfully track the selected targets in all three sequences (Fig. 4.7, 4.8, A.4, A.5, A.8 and A.9) and their results are very close to the ground truth in all cases.

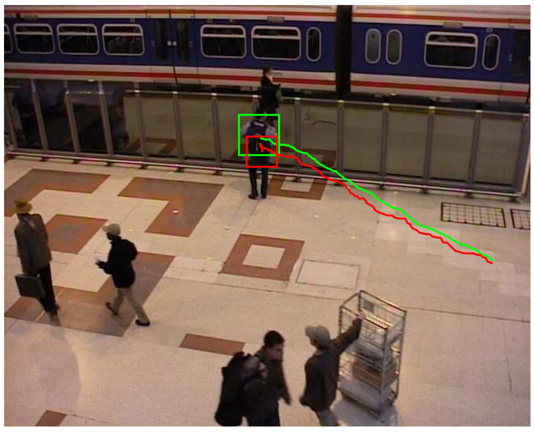
From the three graphs of the absolute error e_i illustrated in Fig. 4.9, 4.10 and 4.11 for the three test sequences, it can be seen that in general the error for SWAD and SWADOCC is significantly lower with respect to the other three


(a) frame #968


(b) frame #996


(c) frame #1026


(d) frame #1056


(e) frame #1086

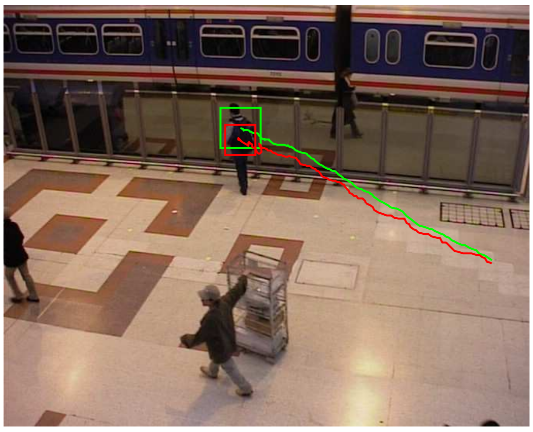

(f) frame #1116

Fig. 4.4: MS results for the S1-T1-C/3 sequence from PETS2006 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.

trackers. In particular, it can be seen that the error for SAD and NCC diverges in Fig. 4.9 and 4.10, as in both cases SAD and NCC lose the target. In Fig. 4.11 it can be noticed that SWADOCC has a spike around frame #130; this single

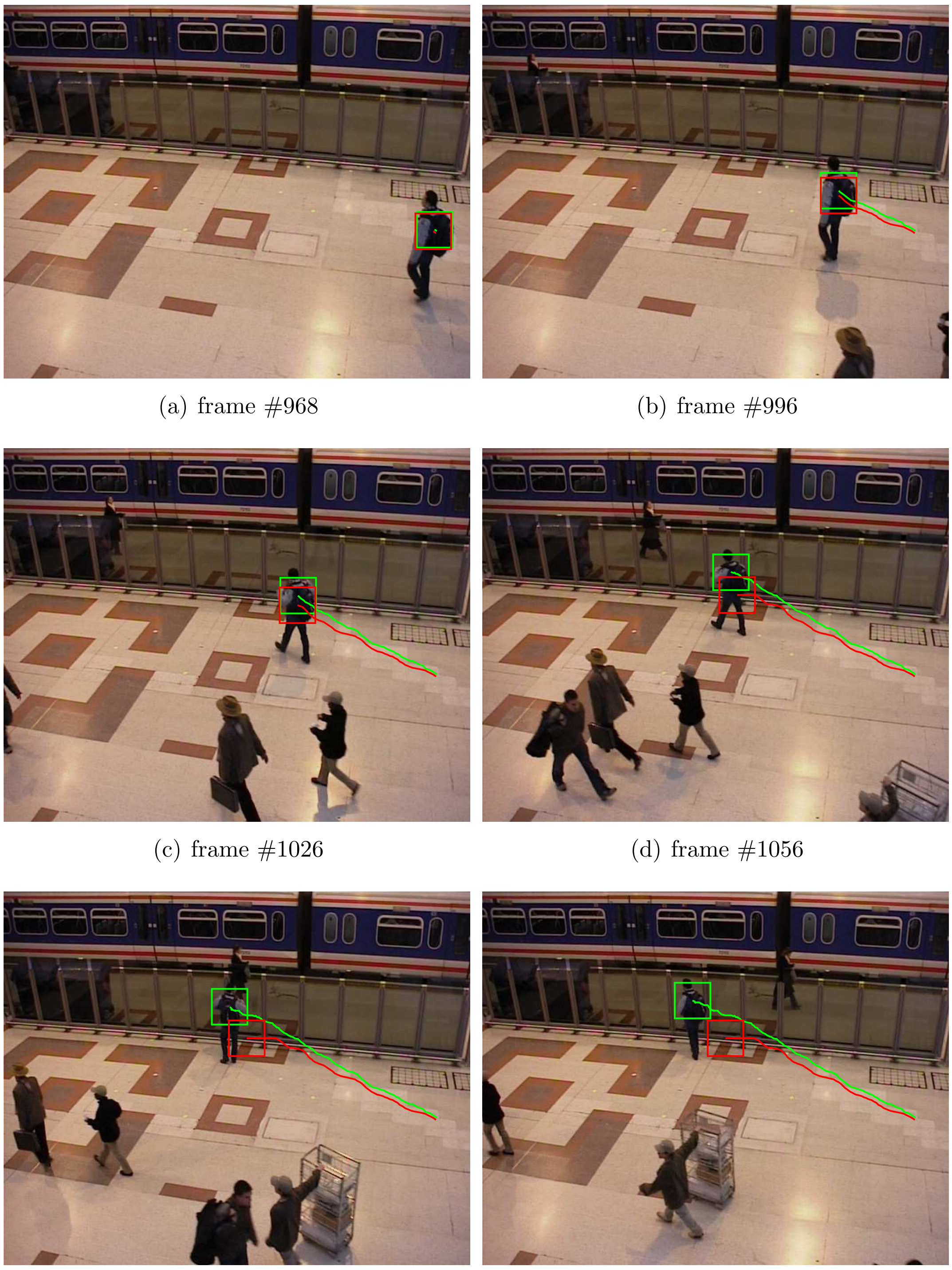
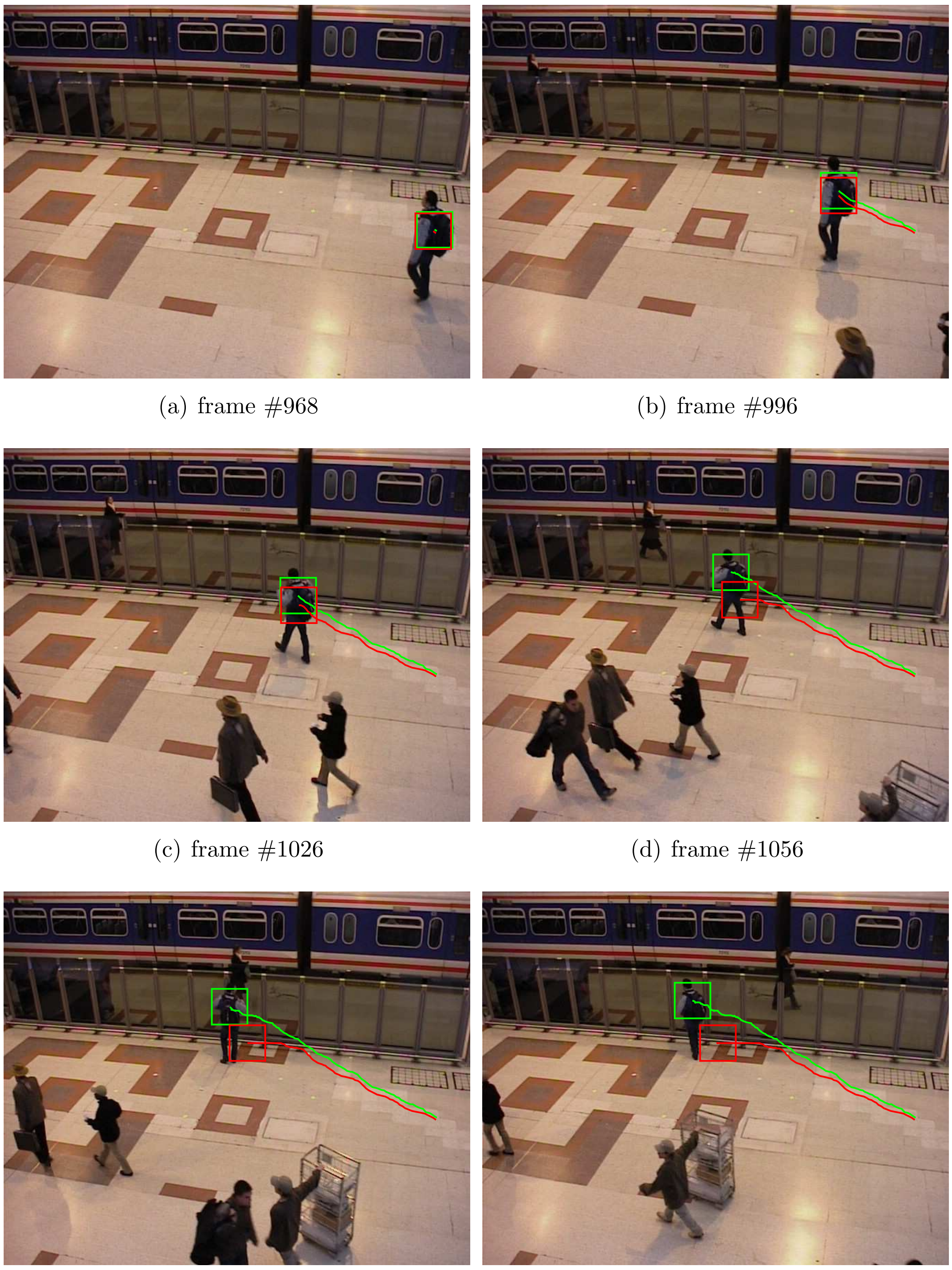

(a) frame #968
(b) frame #996
(c) frame #1026
(d) frame #1056
(e) frame #1086
(f) frame #1116

Fig. 4.5: SAD results for the S1-T1-C/3 sequence from PETS2006 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.

misdetection corresponds to the frame in the Dudek sequence in which the target moves his hand close to his face. Nonetheless, the tracking results for SWADOCC in this particular sequence are the best ones, among the four trackers.

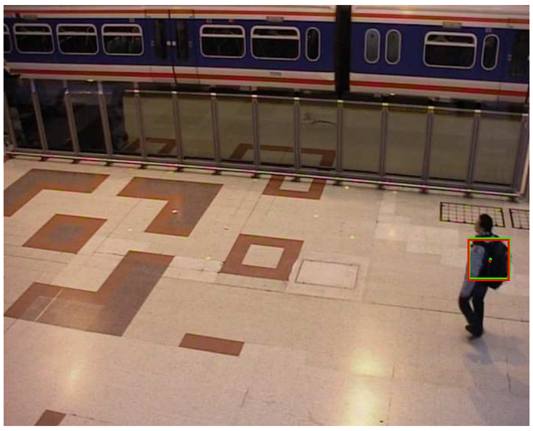

(a) frame #968 (b) frame #996

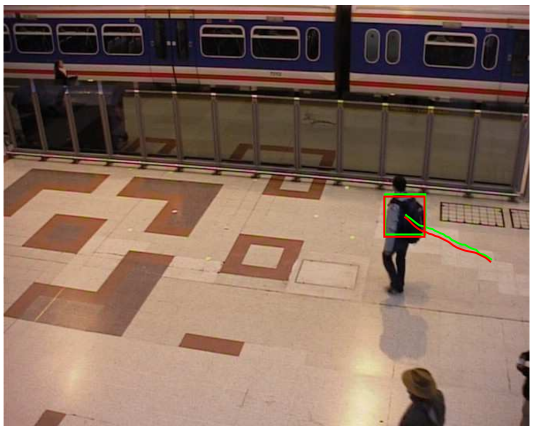
(c) frame #1026 (d) frame #1056

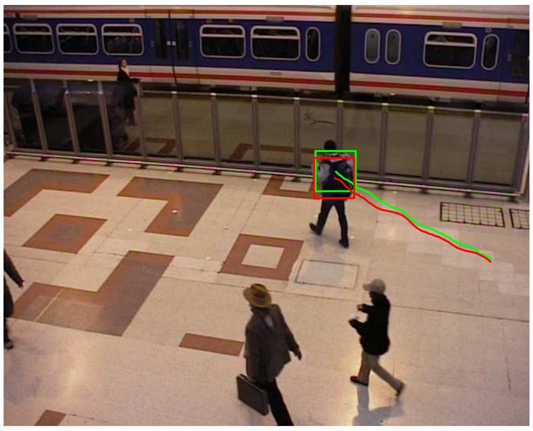
(e) frame #1086 (f) frame #1116

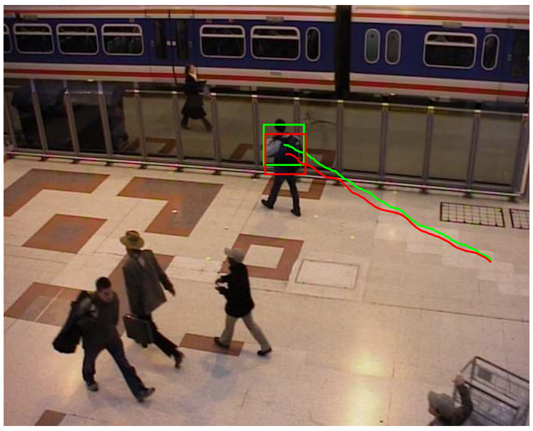
Fig. 4.6: NCC results for the S1-T1-C/3 sequence from PETS2006 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.

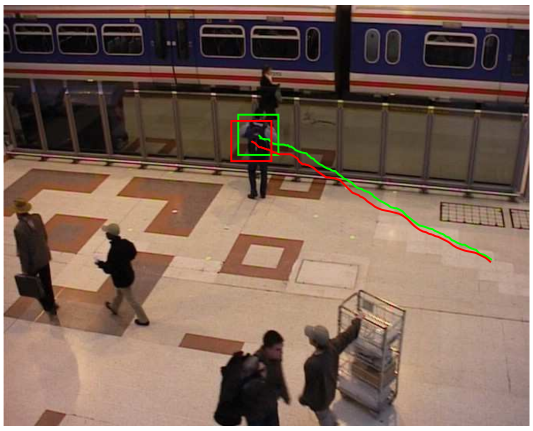
Numerical results reported in Tab. 4.1, in terms of mean error μ_e and standard deviation σ_e, show once more the better performance of SWAD and SWADOCC, compared to the other trackers. In the PETS2006 sequence, SWADOCC and


(a) frame #968


(b) frame #996


(c) frame #1026


(d) frame #1056


(e) frame #1086

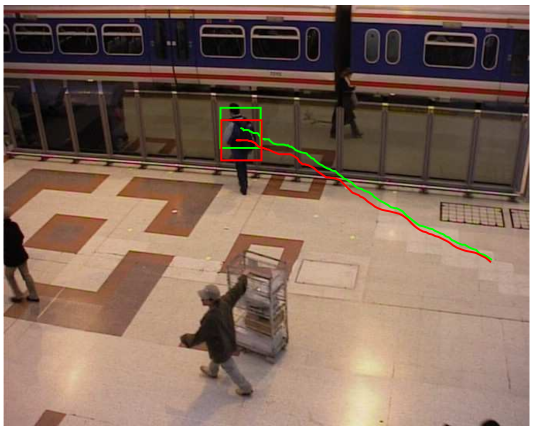

(f) frame #1116

Fig. 4.7: SWAD results for the S1-T1-C/3 sequence from PETS2006 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.

SWAD show similar performance. For the PETS2007 sequence, SWADOCC performs worse than SWAD; this is due to a very small vertical drift in the SWADOCC tracker, as it can be appreciated in Fig. A.5(d)–(f), where the red and

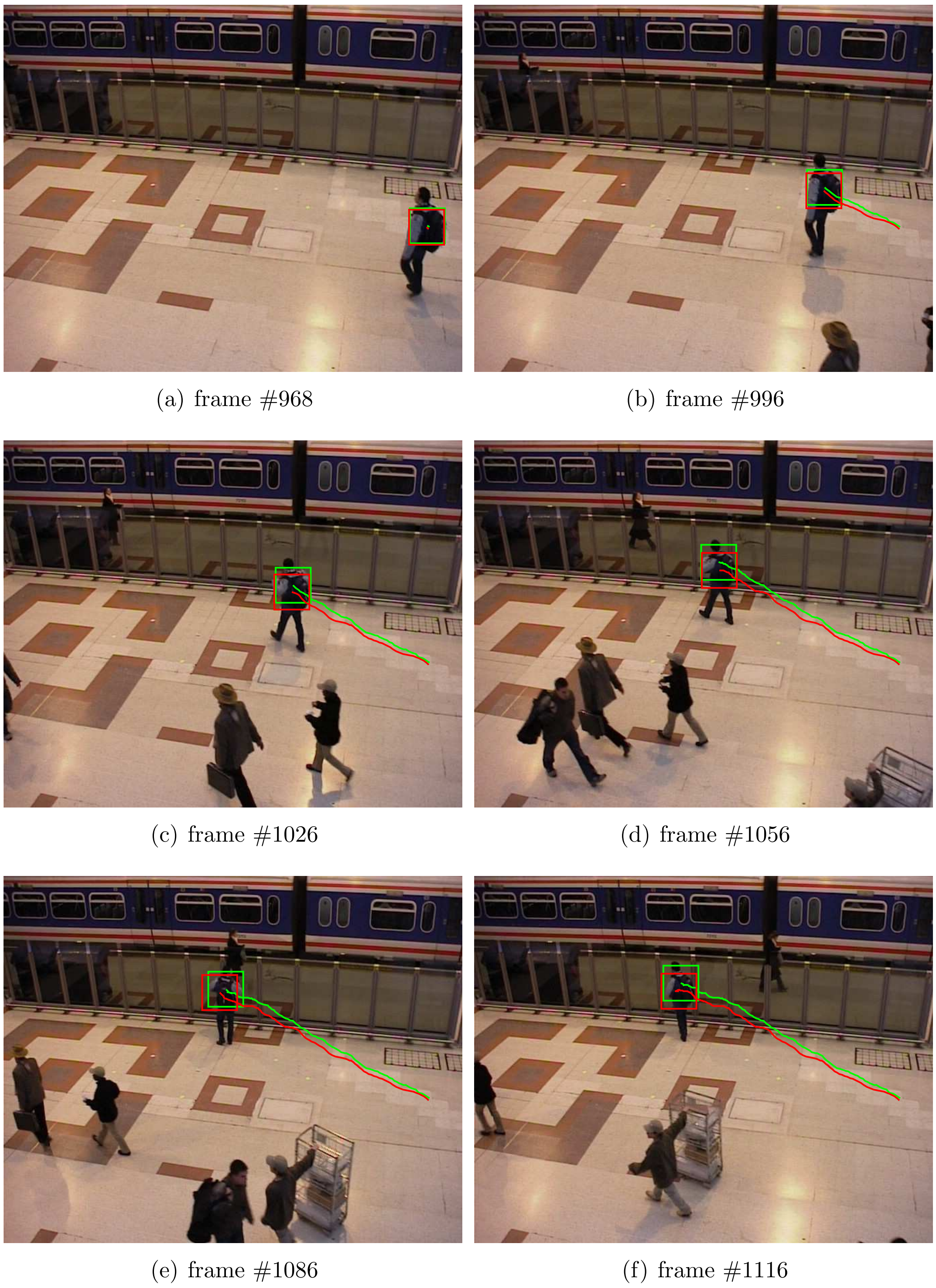

(a) frame #968
(b) frame #996
(c) frame #1026
(d) frame #1056
(e) frame #1086
(f) frame #1116

Fig. 4.8: SWADOCC results for the S1-T1-C/3 sequence from PETS2006 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.

green tracks are further part compared to the SWAD case (Fig. A.4). Nonetheless, SWADOCC can correctly track the target throughout the sequence, unlike SAD and NCC which lose the target (Fig. A.2 and A.3).

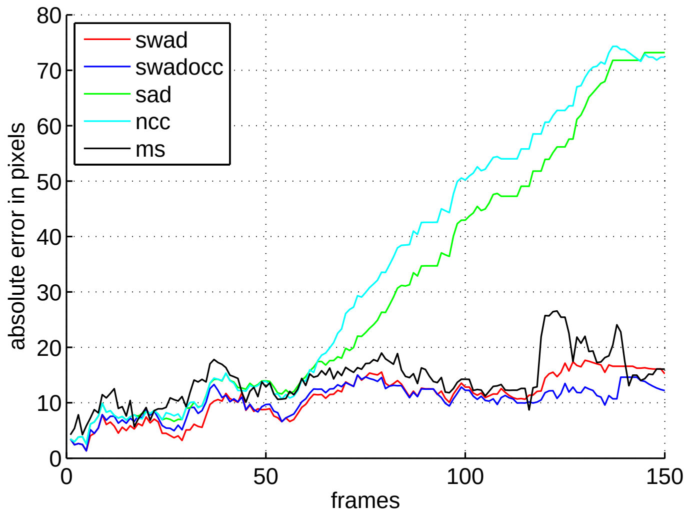


Fig. 4.9: Absolute error e_i for the S1-T1-C/3 sequence from PETS2006 dataset.

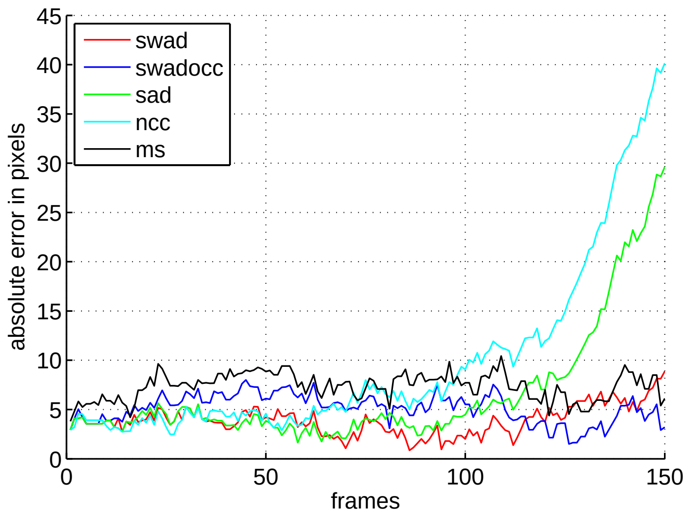


Fig. 4.10: Absolute error e_i for the S06_2/1 sequence from PETS2007 dataset.

Tab. 4.1: Mean value μ_e and standard deviation σ_e of absolute error e_i in pixels.

	SWAD		SWADOCC		SAD		NCC		MS	
	μ_e	σ_e	μ_e	σ_e	μ_e	σ_e	μ_e	σ_e	μ_e	σ_e
PETS2006	10.91	4.14	10.38	2.89	31.00	22.56	34.13	24.20	14.19	4.44
PETS2007	3.89	1.50	5.13	1.43	6.67	6.22	9.93	9.24	7.34	1.35
Dudek	3.67	2.13	0.98	1.83	4.25	2.54	4.29	2.36	–	–

Occlusion handling performance

To demonstrate the ability of the occlusion handling technique described in Section 4.3 to overcome complete occlusion and successfully recover the correct target, comparisons between SWAD and SWADOCC are carried out on three publicly available sequences. For the first experiment, the Laboratory sequence from

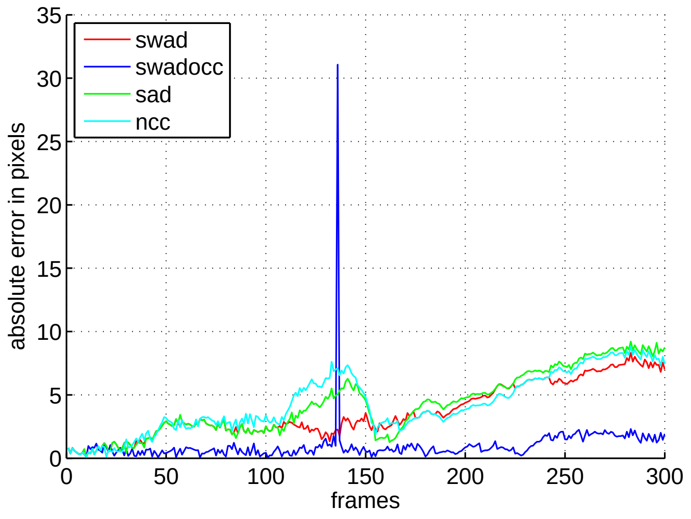
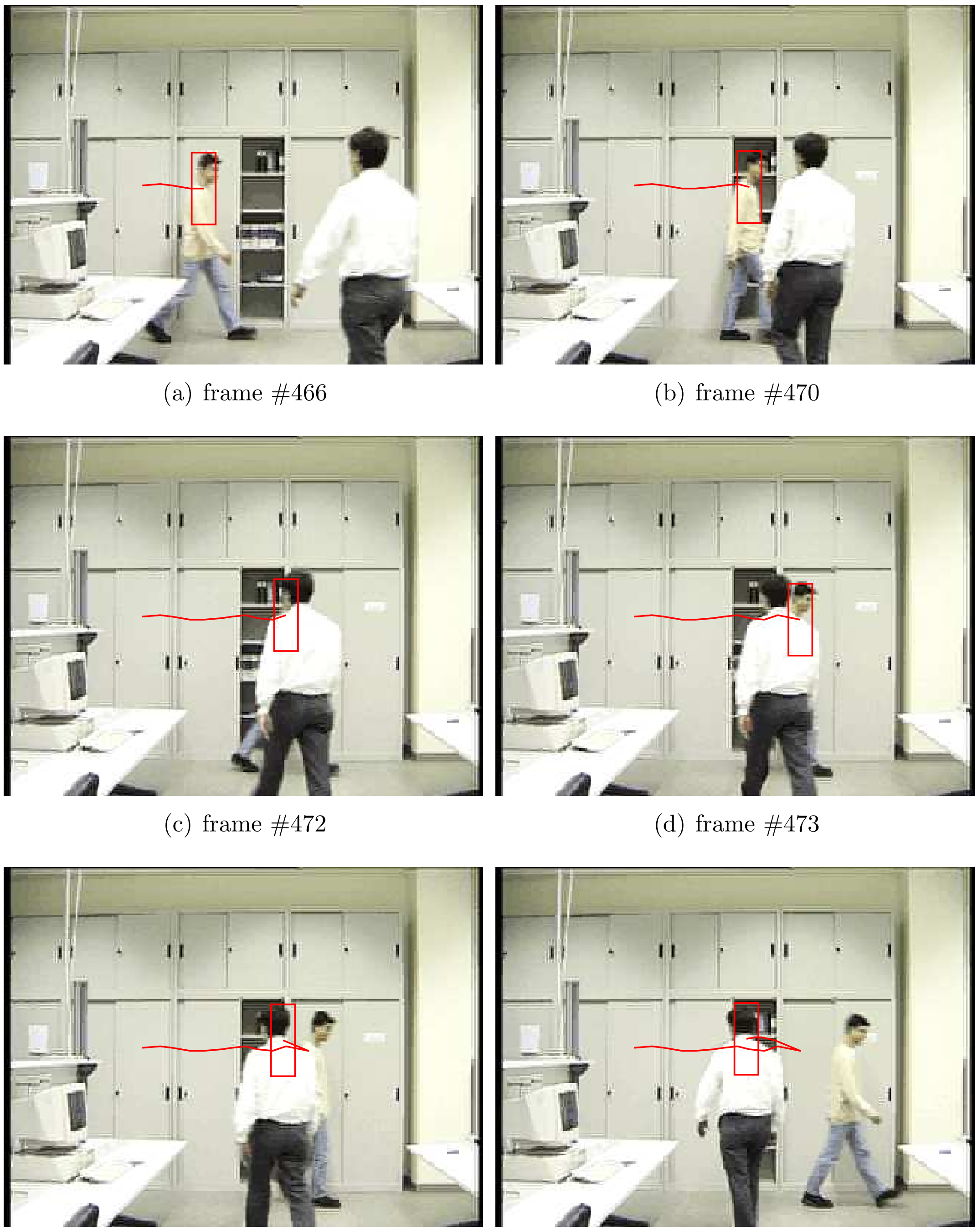


Fig. 4.11: Absolute error e_i for the Dudek face sequence.

the ATON dataset is used [130]. Fig. 4.12 and 4.13 illustrate six frames from this sequence respectively for SWAD and SWADOCC. The trackers are initialised on the person entering the scene from the left (Fig. 4.12(a) and 4.13(a)). When the second person entering from the right occludes the target in the frame centre, SWADOCC correctly detects the target as occluded, i.e. not visible, in Fig. 4.13(c) and 4.13(d), while SWAD selects the occluding person as the correct target in Fig. 4.12(e). In Fig. 4.13(f), SWADOCC still follows the correct target, while SWAD gets distracted by the occlusion event and starts following the second person in Fig. 4.12(f).

In the second experiment, the video sequence S2-L1-12.34-007 from the PETS-2009 dataset is used [131]. Both trackers are initialised on the person in the centre as illustrated in Fig. A.10(a) and A.11(a) in Appendix A. When occlusion takes place (Fig. A.10(c) and A.11(c)), the correct target is not visible, but both SWAD and SWADOCC erroneously select the occluding object, i.e. person with the red jacket, as the target. However, after occlusion, SWADOCC successfully recovers the correct target (Fig. A.11(d)), while SWAD keeps following the wrong target (Fig. A.10(d)). This is due to the fact that, even though SWADOCC selects an incorrect target as in Fig. A.11(c), the blending factors are small (≤ 0.1), so the overall template structure is preserved. Instead, SWAD includes the incorrect target in the template, so after occlusion the best match for the template is actually the person with the red jacket. To prevent misdetections in SWADOCC, the threshold in (4.12) can be increased, as $\vartheta = \lambda \Delta \alpha {N_T}^2$, with $1 < \lambda < L$.

In the third experiment, a synthetic sequence is generated using the Dudek face sequence [129]. In this synthetic video, a square block of random values

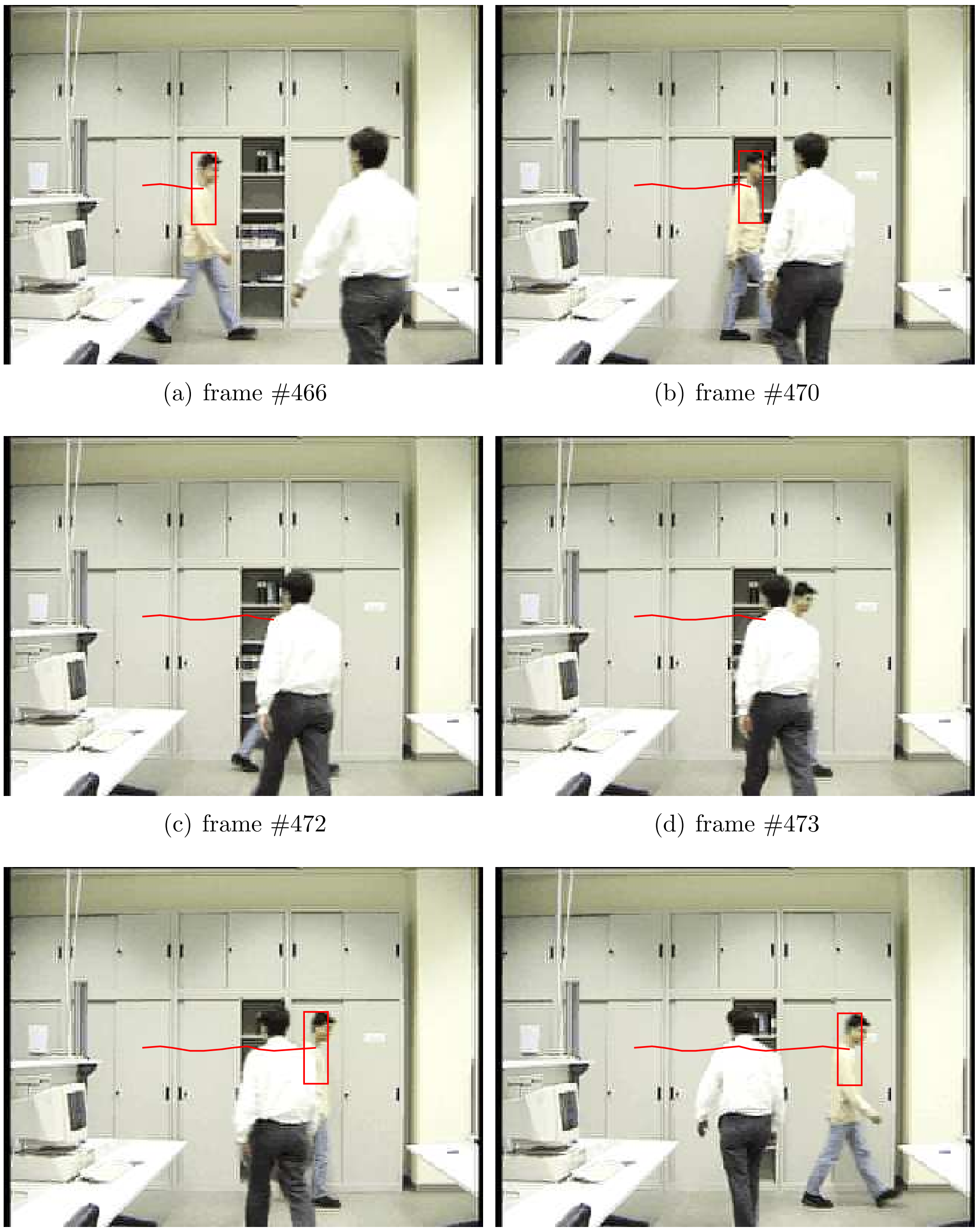

(a) frame #466 (b) frame #470

(c) frame #472 (d) frame #473

(e) frame #474 (f) frame #477

Fig. 4.12: SWAD results for Laboratory sequence from ATON dataset.

in the range $[0, 255]$ moves horizontally, also occluding the person's face, which is the selected target, as illustrated in Fig. A.12(a) and A.13(a) in Appendix A. Moreover, salt and pepper noise is added to each frame. It can be seen in Fig. A.12(d) and A.12(e) that the target detected by SWAD moves along with the occluding block and, in Fig. A.12(f), SWAD definitely loses the target.


(a) frame #466 (b) frame #470

(c) frame #472 (d) frame #473

(e) frame #474 (f) frame #477

Fig. 4.13: SWADOCC results for Laboratory sequence from ATON dataset.

Instead SWADOCC deems the target as occluded in Fig. A.13(c) and A.13(d) and successfully recovers the correct target in Fig. A.13(e) after occlusion.

In the three experiments, a value of $\epsilon_{max} = 26$ was used, which is roughly 10% of the pixel intensity range $[0, 255]$. This means that pixel intensities in the template $\mathbf{T}_i$ can have a variation of ± 26 levels, before being deemed as poor matches. Also, the number of levels used was $L = 5$.

As a final remark, it can be said that both SWAD and SWADOCC trackers are not very sensitive to initialisation, provided that the visible portion of the target selected as template contains texture or visual structure which are characteristic enough of the target.

4.4.2 DSP embedded implementation

The algorithm described in Section 4.2 has been implemented on a DSP, in the context of an embedded smart surveillance sensor for target tracking with a PTZ camera [132, 133]. The novel Sum of Weighted Absolute Differences is implemented in real-time on the multimedia board DM6437 Evaluation Module from Spectrum Digital [134], equipped with a single core TMS320DM6437 fixed-point DSP from Texas Instruments [135]. Experimental results reported in [132, 133] show that, when compared to other tracking methods, such as SAD and NCC, the SWAD-based algorithm shows better performance in the context of an embedded implementation, as it exploits the DM6437 fixed-point architecture.

System overview

The PTZ camera is an ACTi IP Speed Dome CAM-6510, with 360° panning, 180° tilting and a maximum angular speed of 400° per second. A composite analogue video signal is also available as output from the PTZ and is fed into the EVM's video-in port. Both PTZ and EVM are connected to a local area network (LAN) through their Ethernet interfaces, so that they can communicate with each other via TCP/IP. The video analytics algorithm running on the EVM automatically controls the PTZ camera, to follow the target and keep it central to the FOV, when it moves towards the frame boundaries. Moreover the tracker can be activated and deactivated remotely, through on/off TCP-based messages. The PTZ enhanced with the DSP embedded video analytics becomes a smart surveillance sensor. An illustration of such a system is shown in Fig. 4.14.

The system software is implemented in C and runs in real-time at more than 30 frames per second on the EVM. The PTZ camera hosts a proprietary web server and hence no software has been developed for it. Commands for the PTZ are encoded in HTTP requests to the camera web server. The video analytics algorithm on the EVM controls the PTZ by issuing such HTTP-based commands over the network. The EVM also runs a simple TCP server, so that remote on/off signals can be sent to the EVM, to activate the tracking algorithm.

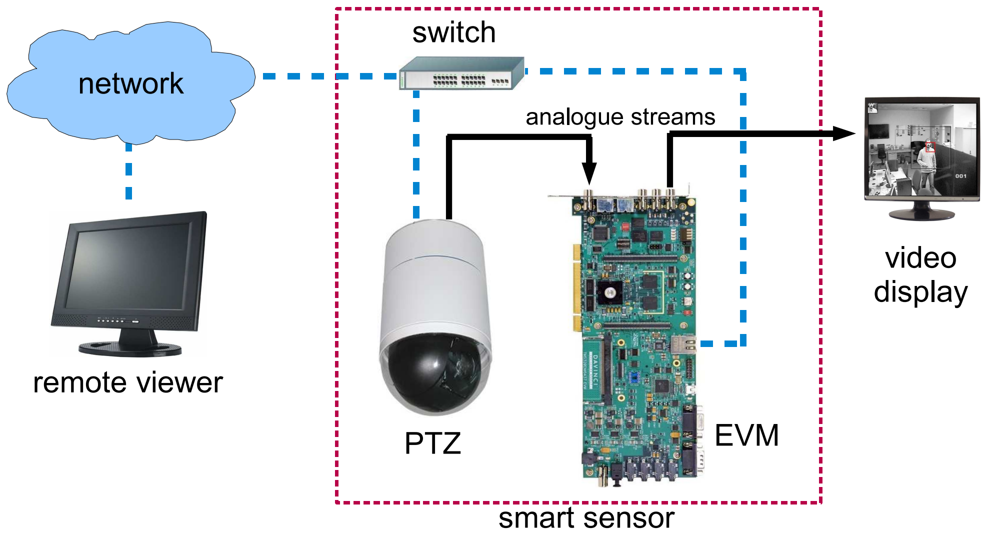


Fig. 4.14: Overview of the DSP embedded system described in Section 4.4.2, with the EVM and PTZ coupled together to form a smart surveillance sensor.

When in stand-by mode, the system does not track any target and the PTZ can be moved freely. When the algorithm is activated, it starts tracking what is in the middle of the FOV at that exact moment. It is straightforward to integrate the proposed DSP embedded smart sensor with other event-based surveillance systems: for example an external smart system [136] can detect an event, compute the 3D position of the target, control the PTZ to point on the target, and then activate the SWAD-based tracking algorithm on the EVM, to control the PTZ and automatically follow the designated target.

Video processing implementation

The composite analogue video output from the PTZ is fed into the EVM's video-in port. This video stream is digitised into 8-bit interleaved YCbCr 4:2:2 frames of 576×1440 pixels, with CbYCrY packed format and video resolution of 576×720 pixels, by the video decoder present on the EVM board. A deinterleaving operation separates individual YCbCr 4:2:2 frames into luminance ($\mathbf{Y}$), blue ($\mathbf{C_b}$) and red ($\mathbf{C_r}$) chrominance components, of which the tracking algorithm requires only $\mathbf{Y}$ to perform its task. The luminance component is then downsampled by a factor of 2 both vertically and horizontally, achieving a frame size of 288×360 pixels.

For speed optimisation on the EVM, deinterleaving and decimation are performed at the same time, by simply extracting only the required samples from the 4:2:2 YCbCr frames. If $\mathbf{J}_i$ is the i^{th} interleaved YCbCr 4:2:2 frame of 576×1440 pixels, the corresponding decimated luminance component $\mathbf{F}_i$ of 288×360 pixels is computed as:

$$F_i(x,y) = J_i(2x, 4y+1) : x \in [0, H-1], y \in [0, W-1] \tag{4.13}$$

where $H = 288$ and $W = 360$ are height and width of $\mathbf{F}_i$.

In the template matching block, pixel values in $\mathbf{F}_i$, kernel values in $\mathbf{K}$ and absolute differences in the SWAD metric are all integers in the range [0,255], and therefore represented with 8 bits (1 byte). The kernel values are computed offline and stored in a look-up table for faster access, as they remain constant for each iteration. Moreover, by setting the blending factor as $\alpha = 0.5 = 1/2$, it is possible to compute the template update in (4.1) as:

$$\mathbf{T}_i = \left[\mathbf{T}_{i-1} + \hat{\mathbf{T}}_{i-1}\right] \gg 1 \tag{4.14}$$

where the division by 2 is performed by the bitwise right-shift operator $\gg$. Using 8-bit integers and bitwise shift operators allows one to optimally exploit the fixed-point architecture of the DM6437 DSP.

The high resolution digital video output from the PTZ can be accessed over the network. For display purposes only, an analogue video signal is available from the EVM's video-out port. This signal contains the interleaved 4:2:2 YCbCr version $\mathbf{S}_i$ of the frame $\mathbf{F}_i$, with the values of the chrominance pixels set to 127. The interleaved frame $\mathbf{S}_i$ is therefore a single plane matrix, with size equal to $H \times 2W = 288 \times 720$ pixels. The interleaving process is performed similarly to the deinterleaving process described above. For every new incoming frame $\mathbf{F}_i$, only the pixels of $\mathbf{S}_i$ corresponding to the luminance component are modified and set equal to $\mathbf{F}_i$, while all the other pixels remains set to 127, as:

$$S_i(x,y) = \begin{cases} F_i(x,m), & y = 2m+1 \\ 127, & otherwise \end{cases} \tag{4.15}$$

where $x \in [0, H-1]$, $y \in [0, 2W-1]$ and $m \in [0, W-1]$. Four frames from the video analytics algorithm running on the EVM are shown in Fig. 4.15. Note that the template $\mathbf{T}_{i-1}$ is shown on the top-left corner of each video frame, while the best match $\hat{\mathbf{T}}_{i-1}$ is highlighted by the red box.

Execution time optimisation

After manual profiling and code optimisation, the total running time of the DSP embedded SWAD-based adaptive template matching target tracking algorithm

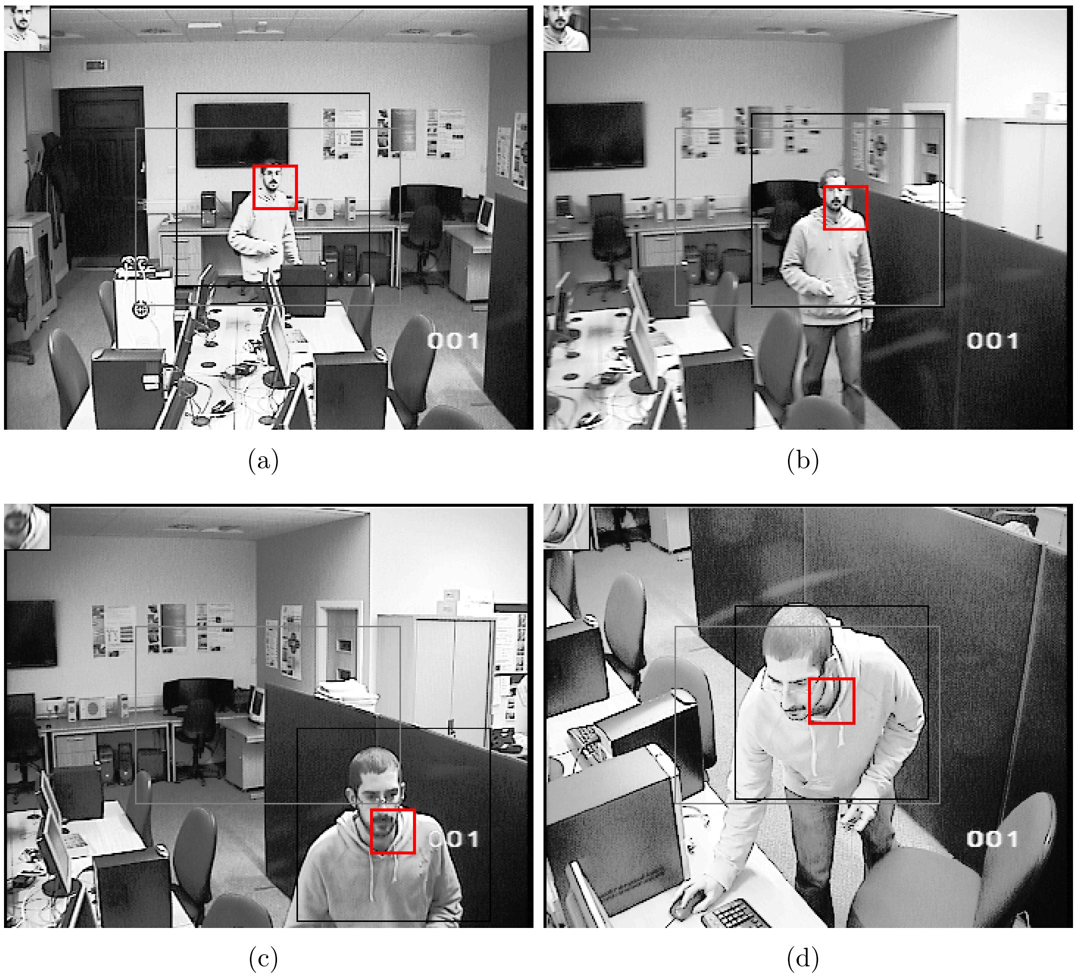


Fig. 4.15: Four images from the tracking algorithm running on the EVM.

is 15 ms per frame, giving a processing frame rate comfortably higher than the real-time requirement of $25 - 30$ fps. The SWAD matching block with $N_T = 32$ and $N_S = 50$ takes 7 ms. Optimisation for the SWAD matching is achieved by exploiting the C code "intrinsics", which are specific functions for the C6000 architecture of the DM6437 DSP [137]. Each C-level intrinsic function is mapped to a single assembly instruction and it executes additions, multiplications and absolute subtractions on groups of four 8-bit integers. For example, the intrinsic function `_MEM4` reads 4 pixel values from memory; `_SUBABS4` computes the absolute difference between two groups of 4 pixels; and `_DOTPU4` computes the dot product between two vectors of 4 pixels. This approach reduces the number of operations for each row of pixels in the template $\mathbf{T}_i$ by a factor of 4. A plain implementation of the SWAD matching without intrinsics takes 63 ms, so being 9 times slower than optimised SWAD. This demonstrates that the presented SWAD-based adaptive template matching algorithm exploits the fixed-point architecture of the DSP on the EVM.

An optimised version of the SAD matching using intrinsic functions takes 5 ms. Nonetheless, even though SWAD is slightly slower than SAD, its better tracking performance reported in Section 4.4.1 entirely justifies its usage over conventional SAD. Concerning an implementation of NCC on the DM6437, it can be said that extra care must be taken to simulate floating point operations, as for example square root, in integer arithmetic. It takes about 9 ms just to compute the mean values of the template $\mathbf{T}_i$ and of each $N_T \times N_T$ subregion in the ROI $\mathbf{R}_i$. Thus it is clear that the execution of a complete implementation of NCC matching on the DM6437 DSP would definitely take longer than 7 ms, and therefore NCC would be slower than SWAD.

4.5 Conclusion

This chapter has presented a novel adaptive template matching algorithm for tracking, based on the minimisation of the Sum of Weighted Absolute Differences. A Gaussian weighting kernel is used to assign different weights to pixels distant from the target centroid, as these pixels may experience partial occlusion. The target template is updated using an IIR filter to adapt to template changes, such as rescaling and rotation.

To overcome severe and complete occlusion, a novel technique based on a matrix of blending factors has been described, so that separate parts of the template are updated at different rates and the overall structure of the target template is significantly preserved.

Despite its simplicity, experimental results have shown the robustness of the SWAD-based tracker and its higher accuracy with respect to SAD-based tracking. Moreover the presented technique for complete occlusion handling allows the tracker to detect the target as occluded and recover it, after the complete occlusion is over.

Finally a DSP embedded implementation of the SWAD-based tracker has been described, in the context of a smart surveillance application. The SWAD-based tracker is suitable for real-time implementations on devices with low computational capabilities, as in the case of fixed-point embedded DSP platforms.

Chapter 5

Improved mean shift tracker

5.1 Introduction

As reported in Section 3.4, a basic assumption of the conventional MS tracker is that there exists overlaps between detections of the target in consecutive frames. This condition does not always hold, particularly in the case of fast moving targets, low frame rate sequences and in occurrence of complete occlusion, such as crowd scenes, where a target may only be partially visible at random intervals and in random positions. Many approaches have been proposed in the literature [105–115, 117–126], to improve the tracking performance of the conventional MS tracker. However such approaches are not free from drawbacks and are not generally applicable.

This chapter presents a novel improved MS tracker which incorporates a failure recovery strategy. The improved MS tracker is simple and fast, and experimental results show that it can effectively recover a target after complete occlusion or loss, to successfully track targets in complex scenarios, such as crowd scenes [138]. The presented algorithm can be easily combined with the numerous extensions for the conventional MS tracker proposed in literature.

The remainder of this chapter is organised as follows. In Section 5.2 an overview of the novel improved MS tracker is given, while the failure detection approach is introduced in Section 5.3. Section 5.4 describes the failure recovery procedure. Details of the tracker initialisation are given in Section 5.5. Experimental results reported in Section 5.6 demonstrate the superiority of the new tracker, compared to the conventional MS algorithm. Section 5.7 concludes the chapter.

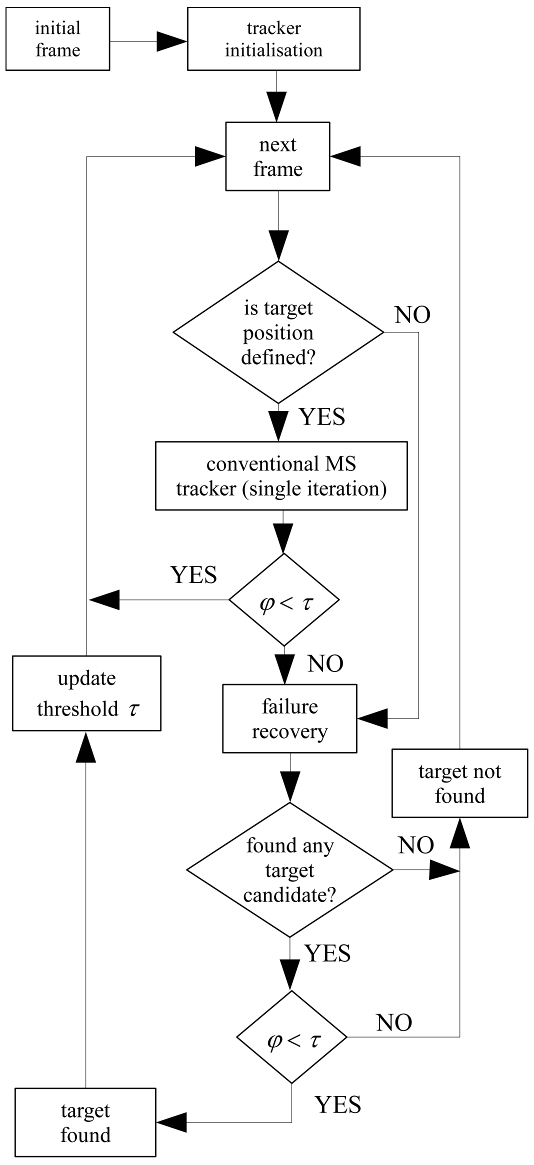


Fig. 5.1: Block diagram of the improved MS tracking algorithm.

5.2 Algorithm overview

A block diagram of the proposed tracking algorithm is shown in Fig. 5.1. The target to track is directly selected in the initial frame $\mathbf{F}_0$, in position $\mathbf{y}_0$. Then the initialisation step computes from $\mathbf{F}_0$ the target model, represented by the target histogram $\mathbf{Q}$, and an initial value for a threshold τ, which is the smallest distance between the target histogram $\mathbf{Q}$ and any other object in the frame other than the target itself.

After the acquisition of a new frame $\mathbf{F}_i$, the algorithm checks if the position $\mathbf{y}_{i-1}$ of the target in the previous frame $\mathbf{F}_{i-1}$ is defined, i.e. the target has been found in the previous frame. Although this check is trivial for $i = 1$, it is indeed necessary when $i > 1$. In fact, if the target has not been found in $\mathbf{F}_{i-1}$, the algorithm proceeds directly to the failure recovery block, to detect the presence of the target in the current frame $\mathbf{F}_i$. If the position in $\mathbf{F}_{i-1}$ is defined, a single step of the conventional MS tracker is executed, which computes the new possible target position $\mathbf{y}_i$ in the current frame and the measure of similarity $\varphi = d(\mathbf{y}_i)$ between the target model $\mathbf{Q}$ and the target candidate found in $\mathbf{y}_i$. The distance φ is then compared with the threshold τ computed in the initialisation step.

If $\varphi < \tau$, the correct target is found. Otherwise, if $\varphi \geq \tau$, one of the three following cases may have occurred:

1. the target found by the conventional MS tracker is the correct one, but its appearance has changed, possibly due to light changes or partial occlusion;
2. the correct target has moved substantially from the previous position, so the conventional MS tracker has lost the target and the returned position $\mathbf{y}_i$ belongs to the background;
3. the correct target is completely occluded, i.e. not visible, and $\mathbf{y}_i$ belongs to the occluding object.

Therefore, when $\varphi \geq \tau$, the failure recovery step is executed and it searches for the correct target in the whole frame $\mathbf{F}_i$. If a candidate is available and it is $\varphi < \tau$, the target is found and the threshold τ is updated, before proceeding to the next frame $\mathbf{F}_{i+1}$. If a candidate is not available or $\varphi \geq \tau$, the target cannot be found, possibly due to complete occlusion; therefore the target position for the current frame remains undefined and the algorithm proceeds directly to the next frame $\mathbf{F}_{i+1}$.

From the implementation of the conventional MS tracker reported in Section 3.4, it is possible to isolate a set of operations into a self-contained procedure referred to as the "MS loop", as shown in Tab. 5.1. Its inputs are the current frame $\mathbf{F}_i$, the position $\mathbf{y}_{i-1}$ of the target in the previous frame and the target model distribution $\mathbf{Q}$. The MS loop also has as input a maximum number of iterations M_{max} for the MS procedure to converge. The outputs of the algorithm are the position of convergence $\mathbf{y}_i$ in the current frame of the MS distance minimisation procedure, and the actual distance $d(\mathbf{y}_i)$. The MS loop is central to the initialisation, MS tracking and failure recovery blocks, and it computes the distance $d(\mathbf{y}_i)$ between the target model $\mathbf{Q}$ and a target candidate $\mathbf{P}(\mathbf{y}_i)$ in position $\mathbf{y}_i$, which is found by following the mean shift vector from an input position $\mathbf{y}_{i-1}$.

Tab. 5.1: MS loop.

```
Algorithm 1: MS loop
INPUTS: $\mathbf{F}_i, \mathbf{y}_{i-1}, \mathbf{Q}, M_{max}$
OUTPUT: $\mathbf{y}_i, d(\mathbf{y}_i)$
BODY:
    $\epsilon = 1, M = 0$
    $\mathbf{y}_b = \mathbf{y}_{i-1}, \mathbf{y}_a = \mathbf{y}_b + \epsilon$
    while $(|\mathbf{y}_a - \mathbf{y}_b| \geq \epsilon)$ AND $(M < M_{max})$ do
        $\mathbf{y}_a = \mathbf{y}_b, M = M + 1$
        compute distribution $\mathbf{P}(\mathbf{y}_a)$ as in (3.11)
        compute weights $a_n$ as in (3.13)
        compute next position $\mathbf{y}_i$ as in (3.14)
        $\mathbf{y}_b = \mathbf{y}_i$
    end
    compute $\rho(\mathbf{y}_i)$ as in (3.15)
    compute $d(\mathbf{y}_i)$ as in (3.16)
```

5.3 Failure detection

In the conventional MS tracker, the only indication of the actual tracking performance in each frame $\mathbf{F}_i$ is given by the similarity measure $\varphi = d(\mathbf{y}_i)$ between the target model distribution $\mathbf{Q}$ and the target candidate distribution $\mathbf{P}(\mathbf{y}_i)$. When $\mathbf{Q}$ and $\mathbf{P}(\mathbf{y}_i)$ are similar, the value of φ is low, while if $\mathbf{Q}$ and $\mathbf{P}(\mathbf{y}_i)$ are different, φ is high. Therefore it is reasonable to consider that a very high value of φ occurs due to occlusion or target loss. Both occlusion and target loss can be treated in the same way at this stage, as in both cases $\mathbf{P}(\mathbf{y}_i)$ does not correspond to the correct target and so it is different from $\mathbf{Q}$. Failure in the MS tracker can be detected by defining a threshold τ, so that the target is considered to be occluded or lost when φ is greater than τ, as follows:

$$\begin{cases} \varphi \geq \tau & \Rightarrow \text{ incorrect target} \\ \varphi < \tau & \Rightarrow \text{ correct target} \end{cases} \tag{5.1}$$

For every new frame $\mathbf{F}_i$, starting from the target position $\mathbf{y}_{i-1}$ in the previous frame and its kernel size $\mathbf{s}_{i-1} = [h_{i-1}, w_{i-1}]$, the MS loop is run three times, with three different values of kernel size $\mathbf{s}$, namely (i) $\mathbf{s} = \mathbf{s}_{i-1}$, (ii) $\mathbf{s} = (1+\alpha)\mathbf{s}_{i-1}$ and (iii) $\mathbf{s} = (1-\alpha)\mathbf{s}_{i-1}$, as suggested in [11]. This is to deal with possible changes in scale of the target, where α is a target rescaling percentage with typical value of $\alpha = 0.1$. Of these three executions, the one that returns the minimum distance $\varphi = d(\mathbf{y}_i)$ is chosen. Then φ is compared with the threshold τ, as in (5.1). If

$\varphi < \tau$, the position $\mathbf{y}_i$ is retained and used as starting point in the next frame $\mathbf{F}_{i+1}$. If $\varphi \geq \tau$, the failure recovery procedure is run to detect whether the high value of φ is due to either target loss or occlusion.

The initial value for τ is computed when the tracker is first initialised, as described in Section 5.5, and is based on the characteristics of the whole image with respect to the target colour distribution $\mathbf{Q}$. Subsequently the threshold τ is updated every time the failure recovery procedure is run, to take into account possible new objects within the frame, with a colour distribution similar to the target histogram, as explained in Section 5.4.

The failure recovery procedure returns the position $\mathbf{o}_m$ of the minimum distance $\varphi = d(\mathbf{o}_m)$ from $\mathbf{Q}$ within the whole frame $\mathbf{F}_i$; it also returns a possible update value τ' for the threshold τ. The distance φ is again compared to τ as in (5.1). If $\varphi \geq \tau$, a good match for $\mathbf{Q}$ cannot be found and the target is considered to be occluded. In this case, no position is defined for the target in the current frame and consequently the improved MS tracker will start directly from the failure recovery procedure in the next frame $\mathbf{F}_{i+1}$.

If $\varphi < \tau$, $\mathbf{o}_m$ is considered to be the target position in the current frame and it is used as starting point $\mathbf{y}_{i+1} = \mathbf{o}_m$ for the improved MS in the next frame $\mathbf{F}_{i+1}$. Moreover, the threshold τ is updated as follows:

$$\tau = \min(\tau, \tau') \tag{5.2}$$

5.4 Failure recovery

In the first step of the failure recovery procedure in the improved MS tracker, the current frame $\mathbf{F}_i$ is segmented based on the colour range of the target. Since the colour distribution $\mathbf{Q}$ of the target is available, it is possible to extract for each colour plane an associated colour range. This procedure is explained in detail in Section 5.5. Such colour ranges are computed only once when the target is first selected. For an RGB representation, the colour ranges are $[R_1, R_2]$, $[G_1, G_2]$ and $[B_1, B_2]$. The 3-dimensional values in each pixel $\mathbf{x}_n$ in $\mathbf{F}_i$ are compared with the aforementioned colour ranges, to produce a binary mask $\mathbf{T}$ defined as:

$$\mathbf{T}(\mathbf{x}_n) = \begin{cases} 1, & \mathbf{F}_i(\mathbf{x}_n) \in \Psi \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

where $\Psi = [R_1, R_2] \times [G_1, G_2] \times [B_1, B_2]$. In $\mathbf{T}$, regions with less than θ pixels are discarded. This operation helps prevent noise elements in $\mathbf{T}$ from being considered as candidate restarting points for the tracker. Although the exact value for θ is application-dependent, θ must not be too high, otherwise small targets might never be found by the failure recovery procedure. As general guideline, $1 < \theta < 10$ has been found experimentally to provide acceptable results. For the L remaining regions in $\mathbf{T}$, their centroids $\mathbf{c}_l$ with $l \in [1, L]$ are computed. Then the MS loop is executed for each of these centroids, yielding L output points (modes) $\mathbf{o}_l$, with associated distances $d(\mathbf{o}_l)$. The L points $\mathbf{o}_l$ are the candidate restarting points for the improved MS tracker. The point $\mathbf{o}_m \in \{\mathbf{o}_l\}_{l=1..L}$ with the associated smallest distance $\varphi = d(\mathbf{o}_m)$ is retained as the possible restarting point for the tracker.

From the $L - 1$ remaining points $\{\mathbf{o}_l\}_{l=1..L} - \{\mathbf{o}_m\}$, the smallest distance $d(\mathbf{o}_s)$ is retained as the new possible threshold τ', i.e. $\tau' = d(\mathbf{o}_s)$ with the point $\mathbf{o}_s \in \{\mathbf{o}_l\}_{l=1..L} - \{\mathbf{o}_m\}$ satisfying the condition

$$|\mathbf{o}_m - \mathbf{o}_s| \geq 2|\mathbf{s}| \tag{5.4}$$

and $\mathbf{s}$ is the size of the kernel used in the last correct detection. The condition in (5.4) prevents the threshold from being updated with $d(\mathbf{o}_s)$, and $\mathbf{o}_s$ belonging to the target. Updating the threshold as in (5.2) defines τ as the separation value between the highest value of $d(\cdot)$ for the correct target, and the smallest value of $d(\cdot)$ for the wrong target, i.e. for a generic background point $\mathbf{y}_b$. In this way the conditions $\tau \leq d(\mathbf{y}_b)$ is always satisfied for every point $\mathbf{y}_b$ in the frame $\mathbf{F}_i$ not belonging to the target, so that tracking failure can be reliably detected as described in Section 5.3. If $L = 0$, φ and τ' are both set equal to 1.

The failure recovery procedure explained above allows the improved MS tracker to find the target after complete occlusion or target loss. Since the MS loop converges to the mode of the estimated density function of a given set of points, the proposed failure recovery procedure converges, for a set of points in $\mathbf{F}_i$ around the centroid $\mathbf{c}_l$, to the local mode $\mathbf{o}_l$ of the estimated density function.

In Fig. 5.2, an example of failure recovery is illustrated, with both frames $\mathbf{F}_i$ and binary masks $\mathbf{T}$. The target is the blue ball selected in Fig. 5.2(a). The ball moves towards the centre of the FOV and it experiences significant partial occlusion in Fig. 5.2(c), while in Fig. 5.2(d) the ball is completely hidden behind the jar in the centre of the frame and no target is detected. In Fig. 5.2(e), the ball is visible again and correctly detected, so normal target tracking is carried out in the next frame (Fig. 5.2(f)). In the binary mask of Fig. 5.2(e), the restarting point for the tracker after occlusion is marked with a red X.

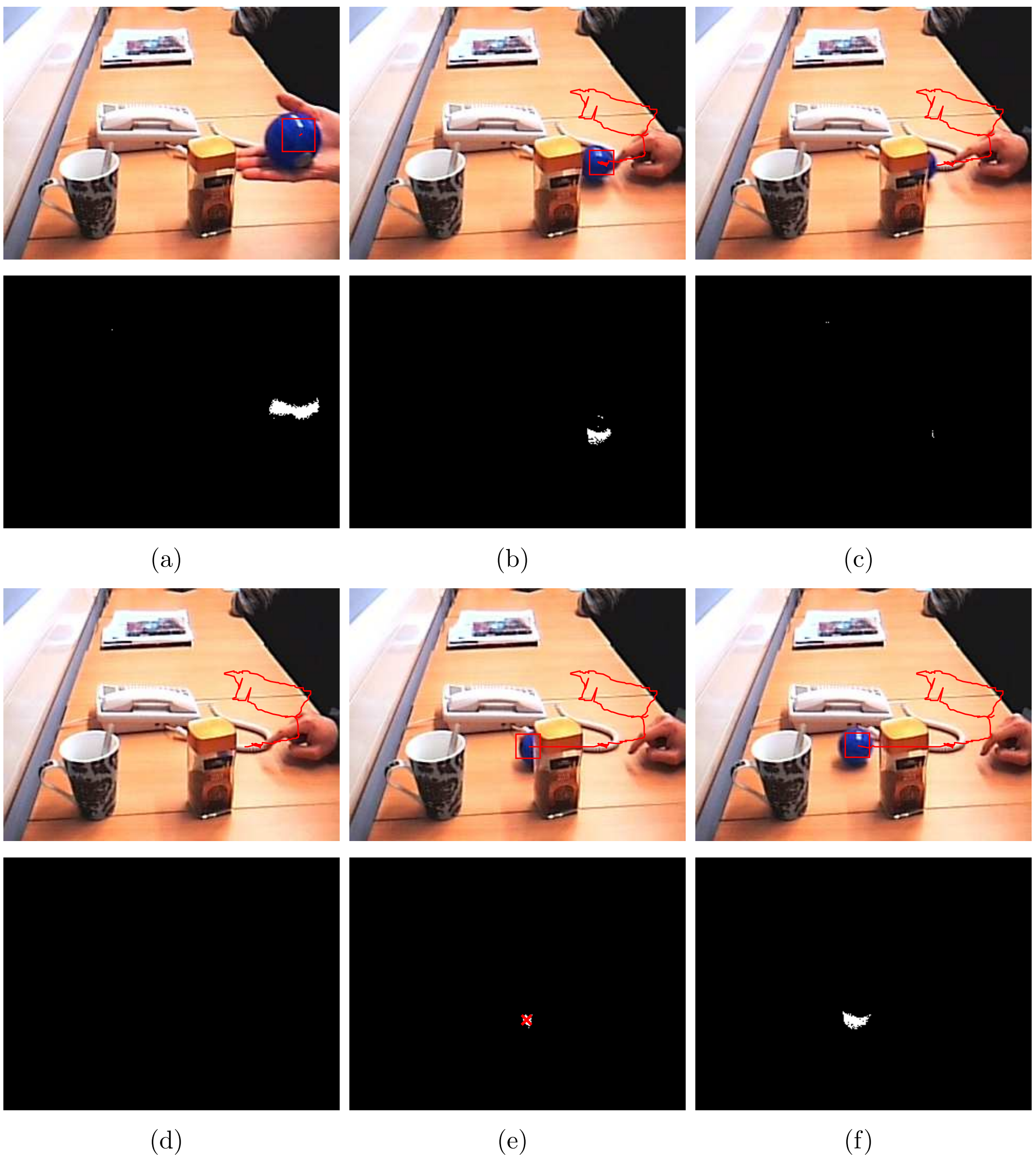


Fig. 5.2: Failure recovery: sequence frames with associated binary masks illustrated underneath. (a) Selected target; (b) target tracking; (c)-(d) target experiencing occlusion; (e) target recovery, with candidate restarting point selected in the corresponding binary mask; (f) target tracking after complete occlusion.

5.5 Tracker initialisation

When the tracker is first initialised in frame $\mathbf{F}_i|_{i=0}$, the colour distribution $\mathbf{Q}$ of the selected target with initial position $\mathbf{y}_0$ and size $\mathbf{s}_0$, as illustrated in Fig. 5.3(a), is computed from (3.7), along with an initial value for the threshold τ and the target colour ranges $[R_1, R_2]$, $[G_1, G_2]$ and $[B_1, B_2]$ introduced in Section 5.4. Such colour ranges are computed as follows.

In an RGB representation, image $\mathbf{F}_0$ has three colour planes and the amplitude of a pixel in each plane is in the range $[0, A-1]$. If β is the number of bins for each dimension in the colour histogram $\mathbf{Q}$, the quantised version $\mathbf{I}$ of $\mathbf{F}_0$ is computed as:

$$\mathbf{I} = \left\lfloor \frac{\mathbf{F}_0}{\Delta} \right\rfloor \tag{5.5}$$

with the bin amplitude Δ defined as:

$$\Delta = \left\lfloor \frac{A}{\beta} \right\rfloor \tag{5.6}$$

From the quantised colour image $\mathbf{I}$, the R, G and B planes are indicated with $\mathbf{I}_r$, $\mathbf{I}_g$ and $\mathbf{I}_b$ respectively. From $\mathbf{Q}$, the 3-dimensional index corresponding to the maximum q_{rgb} is computed as:

$$[r_m, g_m, b_m] = \underset{r,g,b \in [0,\beta-1]}{\arg\max} \left(q_{rgb}\right) \tag{5.7}$$

The computation of the colour ranges $[R_1, R_2]$, $[G_1, G_2]$ and $[B_1, B_2]$ is similar, so for simplicity only the computation of $[R_1, R_2]$ is described here. The N target pixels $\mathbf{x}_n$ are indicated with:

$$\mathbf{X} = [\mathbf{x}_0, \cdots, \mathbf{x}_{N-1}] \tag{5.8}$$

and $\mathbf{I}_r(\mathbf{X})$ represents the values of the N pixels $\mathbf{x}_n$ in $\mathbf{I}_r$. The standard deviation σ_r of all the target pixels $\mathbf{X}$ in $\mathbf{I}_r$ is:

$$\sigma_r = \mathrm{stdev}(\mathbf{I}_r(\mathbf{X})) \tag{5.9}$$

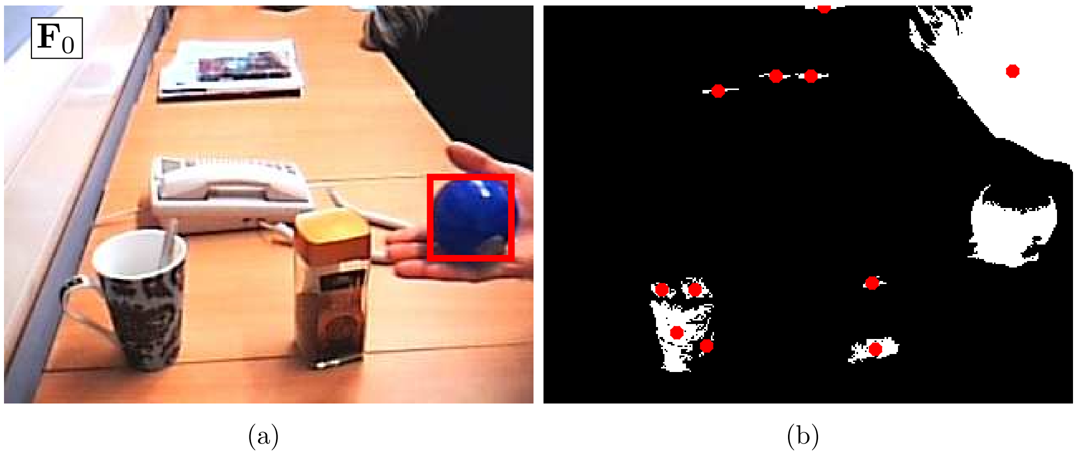
The value of σ_r is used as an indication for colour range $[R_1, R_2]$ centred around r_m. From σ_r, the coefficient ω_r computed as:

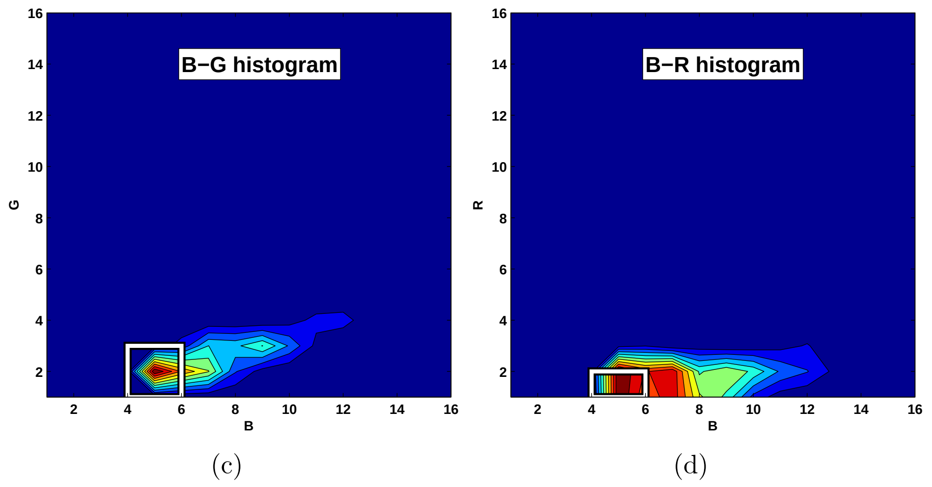
$$\omega_r = \max\left(\left\lfloor \frac{\sigma_r - 1}{2} \right\rfloor, 1 \right) \tag{5.10}$$

is roughly half of the standard deviation σ_r and at least equal to 1. This coefficient is used to compute R_1 and R_2 as follows:

$$\begin{aligned} R_1 &= \max(0, (r_m - \omega_r)\Delta) \\ R_2 &= \min(A-1, (r_m + \omega_r + 1)\Delta - 1) \end{aligned} \tag{5.11}$$

The coefficient ω_r defines the number of bins in the histogram $\mathbf{Q}$, to the right and to the left of r_m, to compute the colour range $[R_1, R_2]$. The values G_1, G_2, B_1





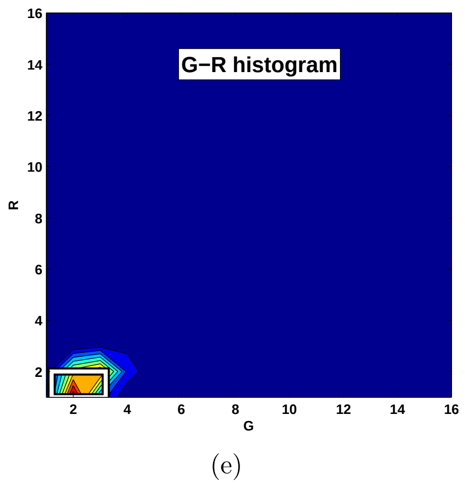


Fig. 5.3: Tracker initialisation: (a) initial target selection in $\mathbf{F}_0$; (b) selection of points $\mathbf{o}_l$ to compute the initial value of τ; (c) B-G histogram of the selected target; (d) B-R target histogram; (e) G-R target histogram.

and B_2 are computed using $\mathbf{I}_g$ and $\mathbf{I}_b$ instead of $\mathbf{I}_r$ in (5.9). In Fig. 5.3(c), 5.3(d) and 5.3(e) the colour ranges quantised with $\beta = 16$ for the target selected in Fig.

5.3(a) are illustrated: the target model $\mathbf{Q}$ is represented as three 2D histograms, while the set of three 2D colour ranges $[B_1, B_2] \times [G_1, G_2]$, $[B_1, B_2] \times [R_1, R_2]$ and $[G_1, G_2] \times [R_1, R_2]$ are represented by the white boxes in Fig. 5.3(c), 5.3(d) and 5.3(e) respectively

The initial value of τ is obtained in a similar fashion to the process of obtaining the smallest distance φ described in Section 5.4. With the target colour ranges computed as described above, the image $\mathbf{F}_0$ may be segmented as in (5.3), wherein L centroids $\mathbf{c}_l$ of regions with at least θ pixels are obtained in the binary mask $\mathbf{T}$. The L candidate points $\mathbf{o}_l$ are computed from $\mathbf{c}_l$ using the MS loop and are illustrated in Fig. 5.3(b). The smallest distance $\varphi = d(\mathbf{o}_m)$, for which:

$$|\mathbf{o}_m - \mathbf{y}_0| \geq 2|\mathbf{s}_0| \tag{5.12}$$

is set as the initial value for τ, i.e. $\tau = \varphi$. In (5.12), $\mathbf{y}_0$ and $\mathbf{s}_0$ are respectively the initial centroid and size of the target.

5.6 Experimental results

To evaluate the performance of the improved MS tracker presented in this chapter, representative tests have been carried out. The video data used for this include:

- a low frame rate sequence recorded in an office environment with an IP camera;
- the Table Tennis test sequence [139];
- two sequences from the PETS2009 dataset for crowded environments [131].

The improved MS tracker is implemented in Maltab, with both RGB and YCbCr colour representations. For RGB, it is $\beta = 32$, while for YCbCr only the chrominance components are used, with $\beta = 64$. The computer used to run the experiments is an Intel Core 2 Quad CPU at 3GHz, with 3.00GB of RAM.

First, both conventional MS and improved MS have been tested on a short low frame rate sequence recorded in an office environment, illustrated in Fig. 5.4 and 5.5. The target in the sequence is the person in red highlighted in Fig. 5.4(a) and 5.5(a). In this sequence, as in all the other ones illustrated in this section, the ground truth is represented by the green track, while tracker result is shown in red. As the frame rate is low, the target displacement in consecutive frames is considerable, so consecutive target detections are outside of the MS basin of attraction. As expected, the conventional MS loses the target (Fig. 5.4), while

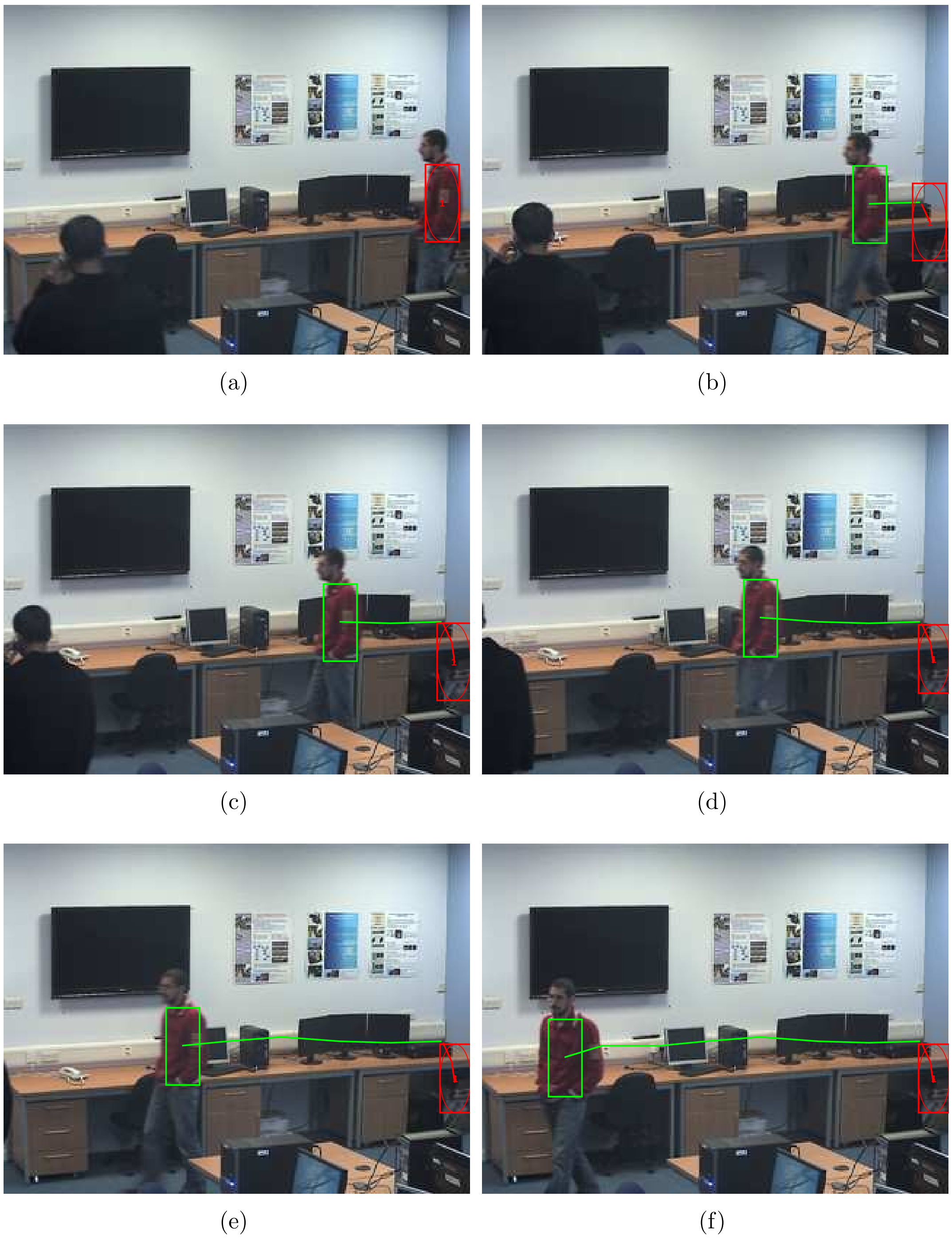

(a) (b) (c) (d) (e) (f)

Fig. 5.4: Conventional MS results for low frame rate indoor video sequence. In each image, the green track is the ground truth, while the red one is the tracker result.

the improved MS can correctly track the target in every frame (Fig. 5.5). The absolute error for both trackers is illustrated in the graph in Fig. 5.6, where it can be seen that the error in the conventional MS diverges, while it is low for the improved MS.

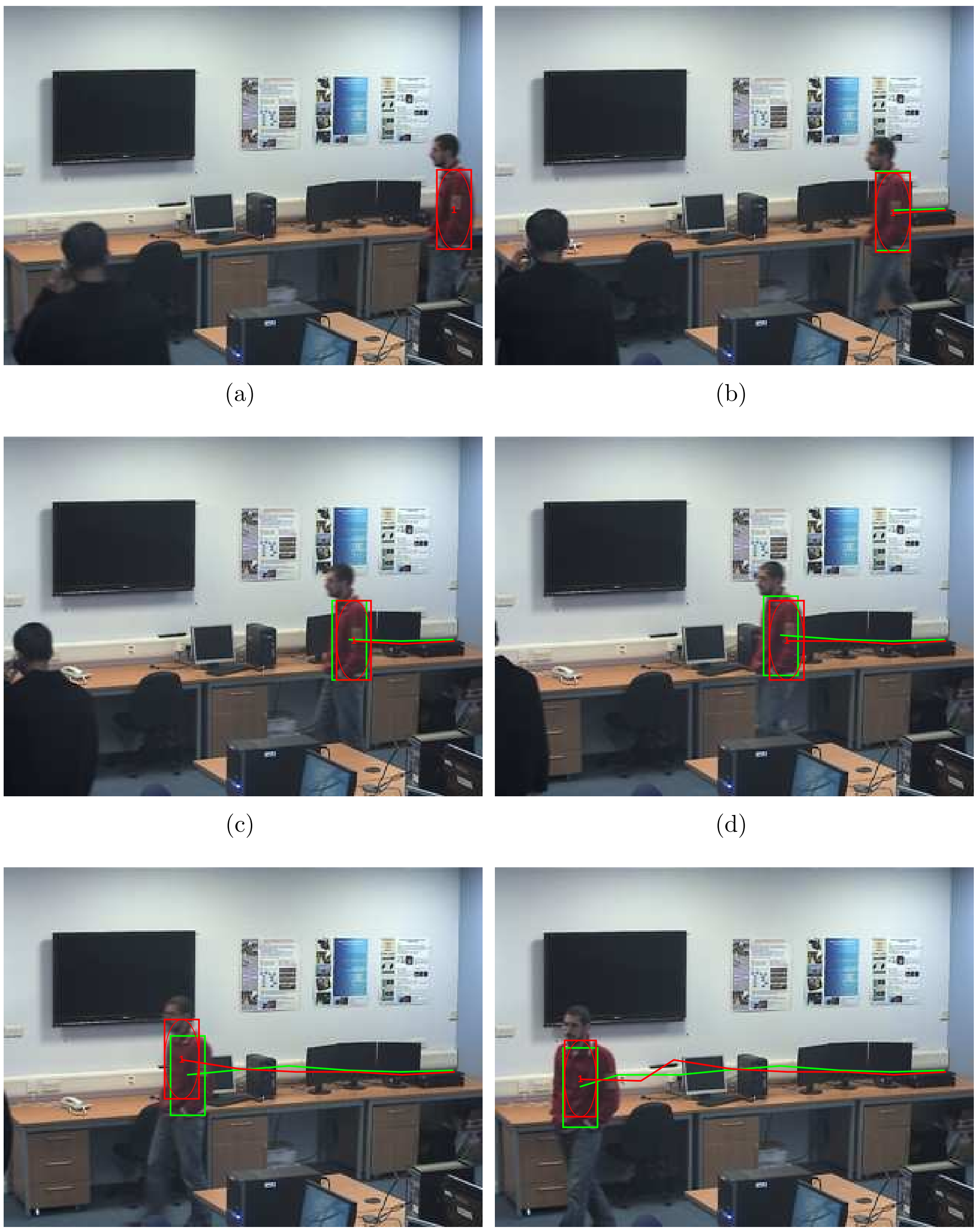

(a) (b) (c) (d) (e) (f)

Fig. 5.5: Improved MS results for low frame rate indoor video sequence. In each image, the green track is the ground truth, while the red one is the tracker result.

A similar behaviour for both trackers is obtained in the Table Tennis sequence illustrated in Fig. 5.7 and 5.8, where the target – the white ball – has large displacement from frame to frame. As it can be seen, the conventional MS soon loses the target (Fig. 5.7(b)), while the improved MS correctly follows the target

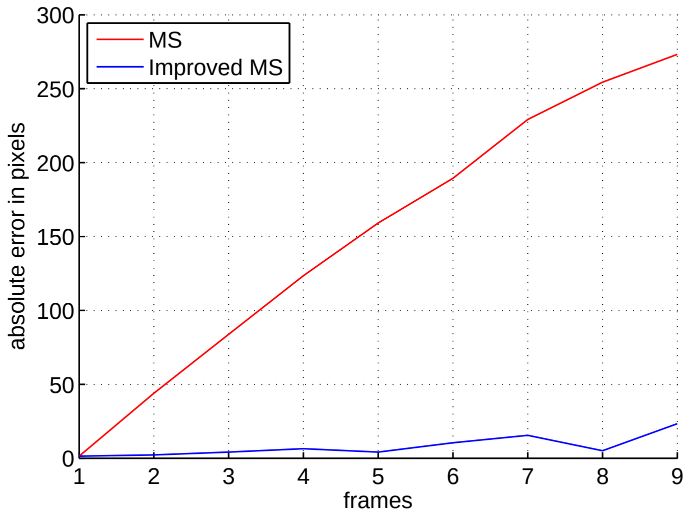
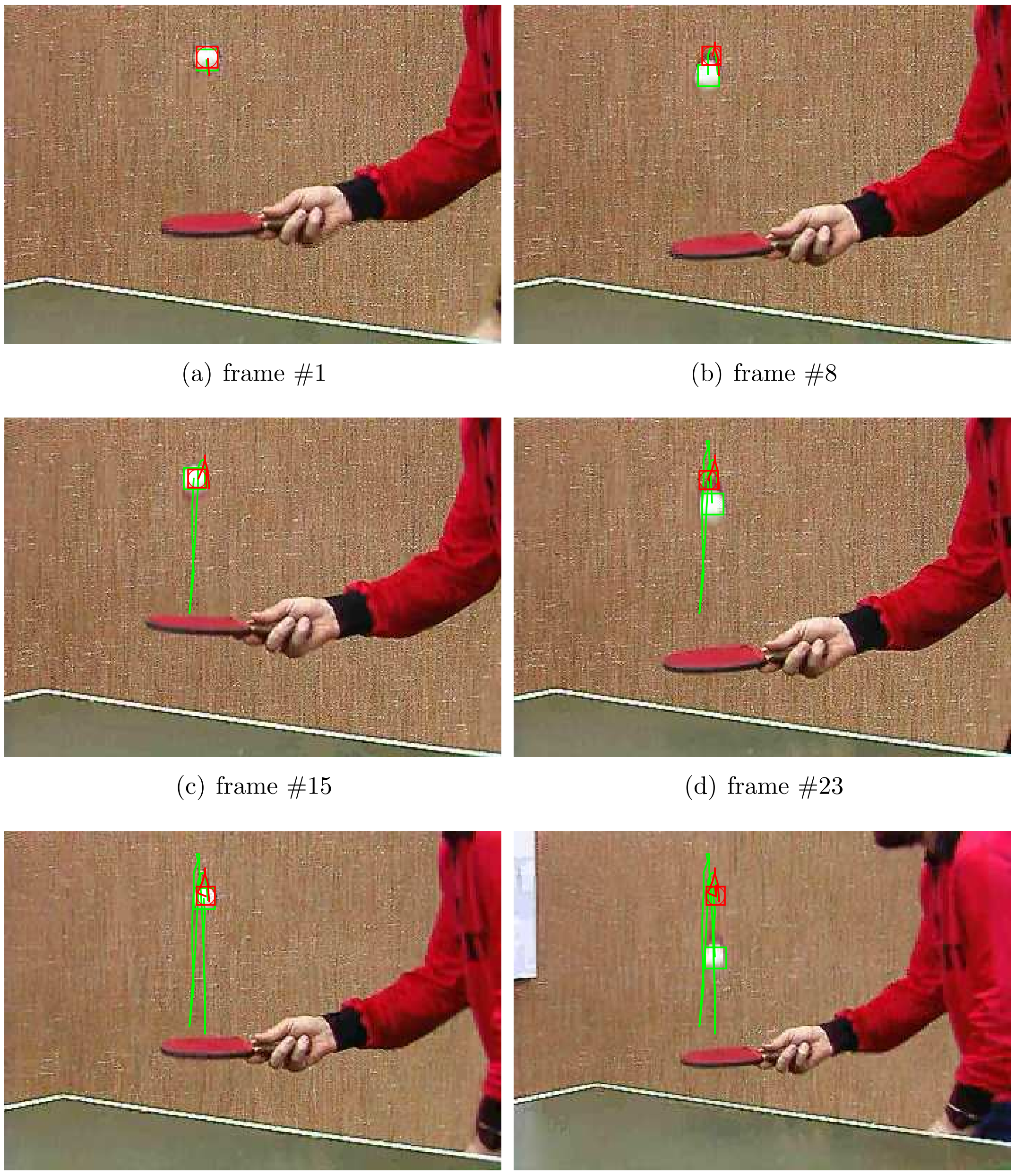


Fig. 5.6: Absolute error for low frame rate indoor video sequence.

throughout the whole sequence (Fig. 5.8). The absolute error for both trackers is illustrated in Fig. 5.9, with the error for the conventional MS being significantly higher than for the improved MS. In particular, the "periodic" behaviour of the error for the conventional MS is due to the oscillating movement of the target in the sequence: the target position returned by the conventional MS algorithm is incorrect, as the target has been lost, and it does not move significantly from the initial target position; the real target instead moves up and down and therefore alternatively reducing and increasing its gap from the tracker incorrect position.

To assess the performance of the improved MS in complete occlusion scenarios, both conventional MS and improved MS have been tested on two sequences from the PETS2009 dataset, namely (i) S3-Multiple_flow-time_14.46-001 and (ii) S3-Multiple_flow-time_14.52-001. Both sequences show a crowd of people moving from right to left in the camera FOV. In both sequences the target is the woman with the pink backpack, as highlighted in Fig. A.14(a), A.15(a), A.16(a) and A.17(a) in Appendix A. Throughout the sequences, the target undergoes severe and complete occlusion, due to the other people in the crowd. It can be seen that the conventional MS tracker soon loses the target in both sequences (Fig. A.14(b) and A.16(b)), while the improved MS can correctly track the target in both sequences, by successfully recovering the target after complete occlusion (Fig. A.15(d) and A.17(d)). The absolute error graphs are reported in Fig. A.18 and A.19, in which the error for the conventional MS again diverges, while it is significantly lower for the improved MS.

Tab. 5.2 reports the mean error μ_e and standard deviation σ_e for both conventional MS and improved MS, with respect to the aforementioned four sequences.


(a) frame #1 (b) frame #8

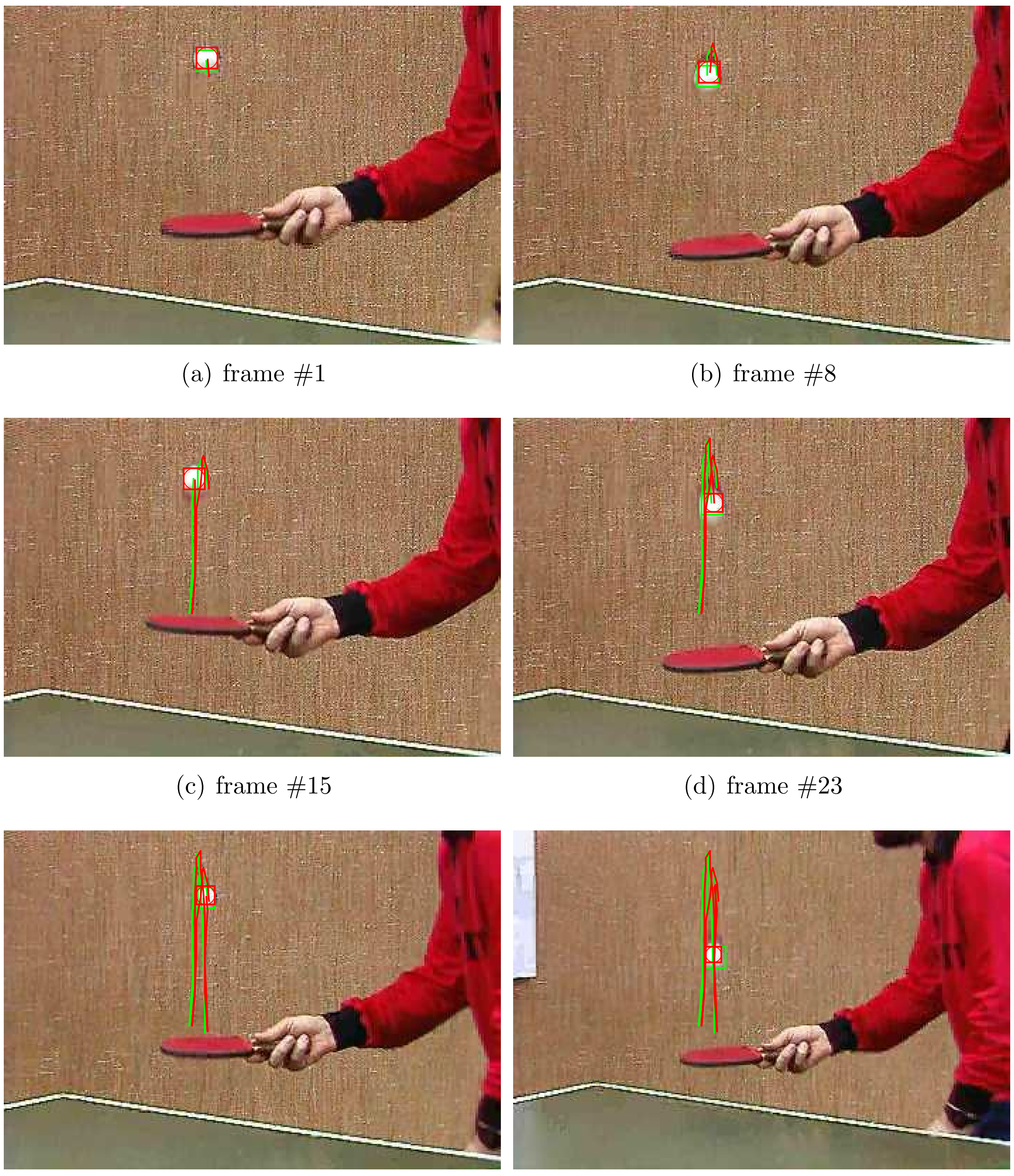
(c) frame #15 (d) frame #23

(e) frame #31 (f) frame #38

Fig. 5.7: Conventional MS results for Table Tennis sequence. In each image, the green track is the ground truth, while the red one is the tracker result.

μ_e and σ_e are considerably lower for the improved MS in all four sequences, demonstrating its better tracking performance in the case of occlusion and fast moving targets.

In terms of computational workload, the average processing rates of the improved MS tracker is approximately 14 fps for the two PETS2009 sequences, with 768×576 resolution, and 30 fps for the other sequences, with 352×240 resolution for the Table Tennis sequence and 320×240 for the indoor sequence. This is to be



(a) frame #1 (b) frame #8

(c) frame #15 (d) frame #23

(e) frame #31 (f) frame #38

Fig. 5.8: Improved MS results for Table Tennis sequence. In each image, the green track is the ground truth, while the red one is the tracker result.

compared to an average processing rate of approximately 16 fps for the PETS2009 sequences and 32 fps for the other two, for the conventional MS tracker. It is clear that, at the cost of a small increase in processing time, the improved MS tracker is able to recover the target after complete occlusion and therefore it has better tracking performance.

The processing time in the improved MS tracker depends on the number of candidate points required to be evaluated in order to restart the algorithm, as

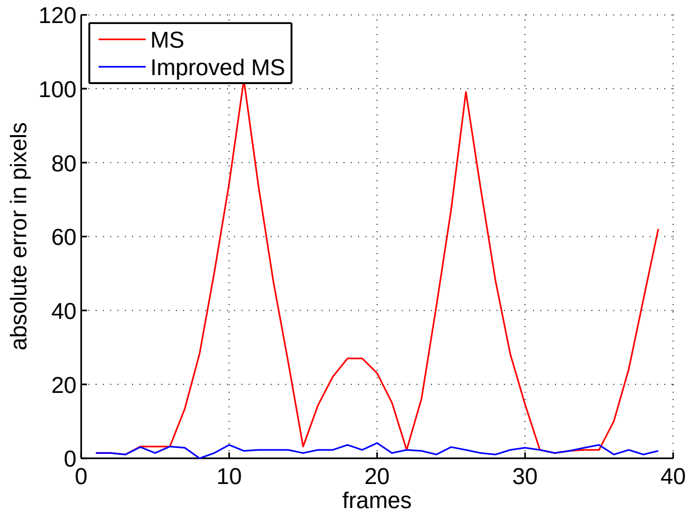


Fig. 5.9: Absolute error for Table Tennis sequence.

Tab. 5.2: Mean value μ_e and standard deviation σ_e of absolute error e_i in pixels, for conventional MS and improved MS.

	MS		Improved MS	
	μ_e	σ_e	μ_e	σ_e
Indoor sequence	143.30	101.19	12.70	17.67
Table Tennis sequence	243.24	138.80	6.23	5.80
PETS2009 S3-Multiple_flow-time_14.46-001	28.18	28.62	2.09	0.88
PETS2009 S3-Multiple_flow-time_14.52-001	150.89	95.13	8.08	7.21

explained in Section 5.4. The maximum number of iterations for a single execution of the MS loop can be controlled by setting the parameter M_{max}. Therefore, for a high number of candidate points, M_{max} can be lowered, to achieve a predefined processing frame rate. It must be noticed that the proposed algorithm is currently implemented in Matlab, so a C implementation of the same algorithm would give significantly higher processing rates. However a frame rate of 14 fps is already acceptable for many video surveillance applications.

5.7 Conclusion

In this chapter an improved MS tracker with failure recovery strategy after complete occlusion has been presented. Experimental results show that the proposed algorithm can track targets in complex scenarios in a fast and effective fashion. The improved MS tracker can successfully recover the target after complete occlusion and target loss. The proposed algorithm can be easily integrated with the numerous extensions to the conventional MS tracker, to incorporate further tracking features in the target model.

Chapter 6

Adaptive algorithm for abandoned/removed object detection

6.1 Introduction

Although the methods proposed in the literature and reported in Section 2.5 perform quite well in detecting moving and static objects, they are not really designed to be able to trigger alerts within a time interval defined by the user, after new abandoned and removed object events have occurred. It is actually the background model updating procedure that dictates when the alerts are triggered.

This chapter presents a novel algorithm for abandoned and removed object detection in real-time, in the context of a reactive smart surveillance system, which notifies the occurrence of such events of interest to registered users, within seconds, through SMS alerts.

In the presented algorithm, a detection time can be directly specified and the background is "healed" only after an abandoned or removed object event has been detected. Moreover the actual detection time and the background model updating rate are computed in an adaptive way with respect to the algorithm frame processing rate, so that even on different machines the detection time is generally the same, in contrast with other algorithms, where either the fps or the background updating rate is considered to be fixed. The algorithm includes also a novel technique to classify detected static regions into abandoned or removed objects, based on the gradient projections in the object boundary pixels [140].

The remainder of this chapter is organised as follows. Section 6.2 describes the novel algorithm for abandoned and removed object detection, while Section

6.3 describes the reactive smart surveillance system, which includes the aforementioned detection algorithm. Experimental results are reported in Section 6.4, while Section 6.5 concludes the chapter.

6.2 Adaptive detection algorithm

6.2.1 Acquisition and object extraction

The images acquired by the proposed algorithm are initially rescaled to 320×240 pixels for faster processing and then converted to grey scale. The algorithm maintains an adaptive model of the background $\mathbf{B}_n$ and, for each new frame $\mathbf{I}_n$, the absolute difference $\mathbf{D}_n = |\mathbf{I}_n - \mathbf{B}_n|$ is computed. For both $\mathbf{I}_n$ and $\mathbf{B}_n$, pixel values are integers in the range $[0, 255]$. If the noise for each pixel in the acquired image is assumed to be Gaussian and $\mathbf{B}_n$ is computed as exponentially weighted moving average, as explained in Section 6.2.4, the difference image $\mathbf{D}_n$ is thresholded to compute a binary mask $\mathbf{M}_n$ expressed as:

$$\mathbf{M}_n = \begin{cases} 1, & \mathbf{D}_n > 4\sqrt{\mathbf{V}_n} \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

where $\mathbf{V}_n$ is a 320×240 matrix containing the exponentially weighted moving variance of the previous n difference images, as described in Section 6.2.4. Starting from the "three-sigma" rule, which states that in a normal distribution 99.73% of the samples fall within three standard deviations, the value 4 is chosen as coefficient for $\mathbf{V}_n$ in (6.1), since four standard deviations contain 99.994% of the samples. Therefore, if the noise in the acquired image is supposed to be Gaussian, all the pixels for which $\mathbf{M}_n$ is 1 are very likely to belong to new object regions in the image. This approach allows to automatically compute an appropriate threshold value for each pixel in $\mathbf{D}_n$, based on real statistics of the video sequence.

Morphological opening and closing are applied to $\mathbf{M}_n$ respectively to reduce noise and fill gaps. Regions with less than ϑ pixels are then removed and holes are filled. The specific value of ϑ depends on the size in pixels of the smallest object to detect. In the binary mask $\mathbf{M}_n$, remaining blobs $\mathbf{b}_i$, with $i \in [0, I-1]$, are groups of R_i connected pixels and they can result from:

1. new real objects in the scene;
2. portions of background left uncovered due to objects being removed;

3. false positives (FP), such as light changes and unconnected cast shadows.

To distinguish between true positives (TP), i.e. new and removed objects, and false ones, each blob region in $\mathbf{I}_n$ is compared with the corresponding region in $\mathbf{B}_n$. If $\mathbf{p}_r$ is a 2D index representing the r^{th} pixel in blob $\mathbf{b}_i$, with $r \in [0, R_i - 1]$, the similarity between all pixels in $\mathbf{I}_n$ and $\mathbf{B}_n$ belonging to blob $\mathbf{b}_i$ is computed as:

$$\rho_i = \frac{\sum_{r=0}^{R_i-1} \mathbf{I}_n(\mathbf{p}_r)\mathbf{B}_n(\mathbf{p}_r)}{\sqrt{\sum_{r=0}^{R_i-1} \mathbf{I}_n^2(\mathbf{p}_r) \sum_{r=0}^{R_i-1} \mathbf{B}_n^2(\mathbf{p}_r)}} \tag{6.2}$$

where ρ_i is the normalised cross-coefficient at lag 0. As suggested in [141], NCC is invariant to linear and affine illumination changes, so that it can compare the object texture for blob $\mathbf{b}_i$, in $\mathbf{I}_n(\mathbf{p}_r)$ and $\mathbf{B}_n(\mathbf{p}_r)$. Moreover, since only static objects are to be detected, the comparison is computed only for lag 0, i.e. with no horizontal or vertical NCC displacement. For true positives, $\mathbf{I}_n(\mathbf{p}_r)$ and $\mathbf{B}_n(\mathbf{p}_r)$ are very dissimilar and ρ_i is low. For false positives, frame and background are similar and ρ_i is high. Therefore ρ_i is compared to a threshold φ: if $\rho_i < \varphi$, $\mathbf{b}_i$ is a true positive; if $\rho_i \geq \varphi$, $\mathbf{b}_i$ is a false positive. At this stage, false positives are discarded, while TP regions $\mathbf{o}_k$ with $k \in K \subseteq [0, I - 1]$ are further analysed. The pixels $\mathbf{p}_r$ of a discarded blob are removed from the binary mask $\mathbf{M}_n$, i.e. they are set to 0, as:

$$\mathbf{M}_n(\mathbf{p}_r) = \begin{cases} \mathbf{M}_n(\mathbf{p}_r), & \text{if } \rho_i \geq \varphi \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

A visual example of the object extraction process is illustrated in Fig. 6.1, where Fig. 6.1(a) shows the background $\mathbf{B}_n$ and Fig. 6.1(b) shows the acquired frame $\mathbf{I}_n$, with an abandoned object in it – a camera – and an object removed – the black box – with respect to $\mathbf{B}_n$. Fig. 6.1(c) shows the difference image $\mathbf{D}_n$. Fig. 6.1(d) shows the "threshold" image $4\sqrt{\mathbf{V}_n}$ as in (6.1), while the binary mask obtained by thresholding $\mathbf{D}_n$ with $4\sqrt{\mathbf{V}_n}$ is illustrated in Fig. 6.1(e). The detected abandoned and removed objects are correctly highlighted in Fig. 6.1(f).

6.2.2 Event detection

To verify whether the k^{th} true positive $\mathbf{o}_k$ detected in Section 6.2.1 is a static or a moving object, a list of previously detected objects is maintained. Each list entry $\mathbf{e}_j$ refers to a detected object $\mathbf{o}_j$ and it contains:

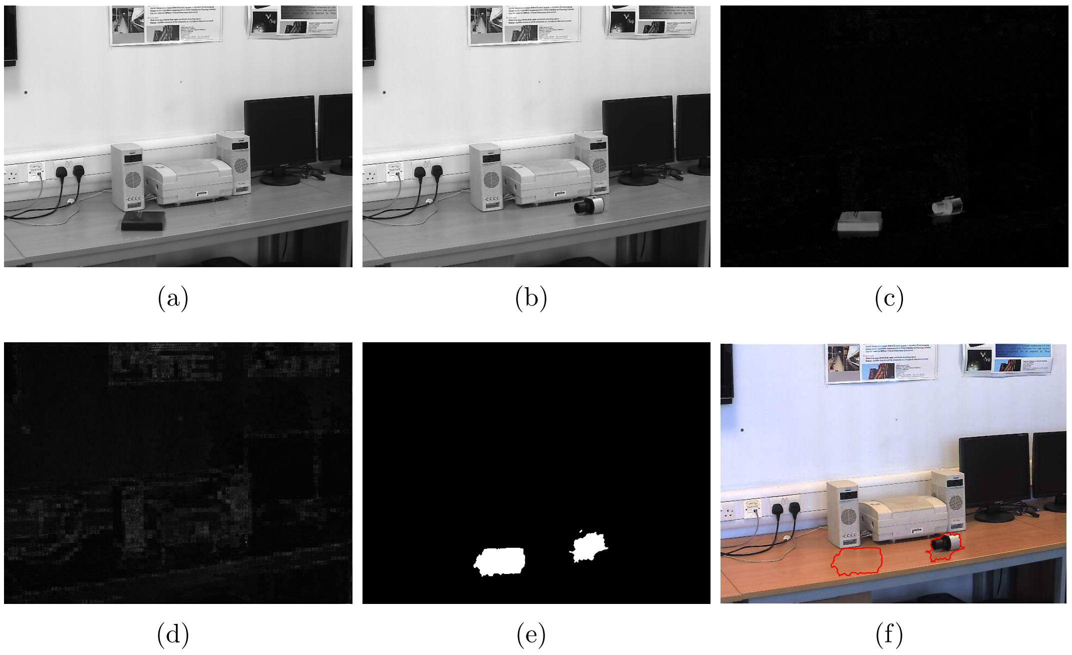


Fig. 6.1: Object extraction process: (a) background $\mathbf{B}_n$; (b) new frame $\mathbf{I}_n$; (c) difference image $\mathbf{D}_n$; (d) "threshold" image $4\sqrt{\mathbf{V}_n}$; (e) binary mask $\mathbf{M}_n$; (f) abandoned object (camera) and removed object (box) detected.

- a feature vector $\mathbf{f}_j = [\beta_j, \zeta_j, \mu_j, \sigma_j]$;
- the object centroid coordinates $\mathbf{c}_j$;
- a frame counter ψ_j;
- a time-to-live counter ν_j.

The elements in the feature vector $\mathbf{f}_j$ are:

- β_j – ratio between the number of pixels in $\mathbf{o}_j$ and the total number of pixels in the frame;
- ζ_j – ratio between the perimeter length λ_j and the blob area;
- μ_j – mean value normalised by its possible maximum number 255;
- σ_j – standard deviation normalised by 255.

The frame counter ψ_j indicates in how many frames object $\mathbf{o}_j$ has been detected, while the time-to-live counter ν_j records the number of frames for which $\mathbf{o}_j$ should be retained in the list.

Each detected object $\mathbf{o}_k$ is compared to every entry $\mathbf{e}_j$ in the object list. If $\mathbf{c}_k$ and $\mathbf{f}_k$ are respectively the centroid coordinates and feature vector of object $\mathbf{o}_k$, a match for $\mathbf{o}_k$ is found in the list if $\mathbf{c}_k$ and $\mathbf{c}_j$ are within a neighbourhood, and if $\mathbf{f}_k$ and $\mathbf{f}_j$ are close enough in the feature space, as defined in (6.4):

$$|\mathbf{c}_k - \mathbf{c}_j| \leq \eta \;\; AND \;\; |\mathbf{f}_k - \mathbf{f}_j| \leq \delta \;\; \Rightarrow \text{match} \tag{6.4}$$

If a match is not found, a new entry $\mathbf{e}_k$ in the object list is created for $\mathbf{o}_k$. The initial value of the frame counter ψ_k for $\mathbf{e}_k$ is set to zero. For the time-to-live counter ν_k, if the user defines a minimum detection time of τ_d seconds, a detection time limit of $\tau_{lim} = 2\tau_d$ seconds is automatically identified and the time-to-live counter ν_k is set to $\nu_k = \tau_{lim} \times fps_{n-1}$, where fps_{n-1} is the current frame rate. If a match is found, then $\mathbf{o}_k \equiv \mathbf{o}_j$ and the frame counter ψ_j associated with entry $\mathbf{e}_j$ is incremented by 1. This counter records the number of frames in which $\mathbf{o}_j$ has been found. The length of time for which $\mathbf{o}_j$ has been detected as static is computed as $\tau_s = \psi_j/fps_{n-1}$. An event is triggered if $\tau_s \geq \tau_d$. If $\mathbf{b}_j$ is the pixel blob in $\mathbf{M}_n$ corresponding to object $\mathbf{o}_j$, $\mathbf{b}_j$ is removed from $\mathbf{M}_n$ and copied to $\mathbf{M}^{static}$, which is a binary mask containing blobs of static objects. Finally entry $\mathbf{e}_j$ is removed from the object list.

The time-to-live counter ν_j ensures that detected objects are not retained in the object list for too long. This prevents moving objects from being considered static after a long time. For example, if an object in the scene had a periodic movement and at regular intervals it was in position $\mathbf{p}_0$, a blob associated with this object would also be detected in position $\mathbf{p}_0$. After a sufficient number of detections, even though the object is clearly moving, the frame counter ψ_0 would reach the detection condition $\psi_0/fps_0 > \tau_d$, and the moving object would be marked as static. This situation has to be avoided and therefore, when the time-to-live counter ν_j reaches zero, the entry associated with the object is removed from the object list.

6.2.3 Abandoned and removed object detection

To determine whether the object $\mathbf{o}_j$ that triggered the event is actually a new object in the scene or an object removed, the boundary of the object in $\mathbf{I}_n$, $\mathbf{B}_n$ and $\mathbf{M}^{static}$ is analysed. For this, the horizontal and vertical Sobel 3×3 operators $\mathbf{S}_x$ and $\mathbf{S}_y$ [38] are applied to $\mathbf{I}_n$, $\mathbf{B}_n$ and $\mathbf{M}^{static}$ as:

$$\begin{aligned} \mathbf{I}_x &= \mathbf{I}_n \otimes \mathbf{S}_x, & \mathbf{I}_y &= \mathbf{I}_n \otimes \mathbf{S}_y \\ \mathbf{B}_x &= \mathbf{B}_n \otimes \mathbf{S}_x, & \mathbf{B}_y &= \mathbf{B}_n \otimes \mathbf{S}_y \\ \mathbf{M}_x &= \mathbf{M}^{static} \otimes \mathbf{S}_x, & \mathbf{M}_y &= \mathbf{M}^{static} \otimes \mathbf{S}_y \end{aligned} \tag{6.5}$$

where $\mathbf{I}_x$ and $\mathbf{I}_y$ are the horizontal and vertical gradient components for $\mathbf{I}_n$; $\mathbf{B}_x$ and $\mathbf{B}_y$ are the gradient components for $\mathbf{B}_n$; and $\mathbf{M}_x$ and $\mathbf{M}_y$ are the gradient components for the mask $\mathbf{M}^{static}$. In (6.5), the symbol $\otimes$ represents the 2D convolution operator. For each boundary pixel $\mathbf{p}_b$ of object $\mathbf{o}_j$, with $b \in [0, \lambda_j - 1]$, the similarity measures $m_1(\mathbf{p}_b)$ and $m_2(\mathbf{p}_b)$ are computed as:

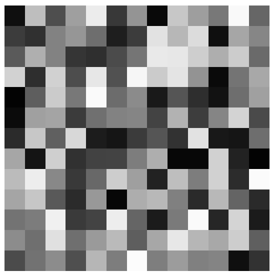
$$\begin{aligned} m_1(\mathbf{p}_b) &= \left| [\mathbf{I}_x(\mathbf{p}_b), \mathbf{I}_y(\mathbf{p}_b)] \, [\mathbf{M}_x(\mathbf{p}_b), \mathbf{M}_y(\mathbf{p}_b)]^T \right| \\ m_2(\mathbf{p}_b) &= \left| [\mathbf{B}_x(\mathbf{p}_b), \mathbf{B}_y(\mathbf{p}_b)] \, [\mathbf{M}_x(\mathbf{p}_b), \mathbf{M}_y(\mathbf{p}_b)]^T \right| \end{aligned} \tag{6.6}$$

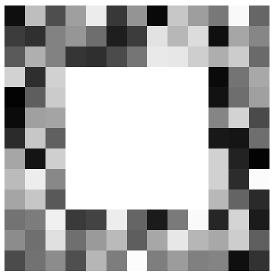
In (6.6), $m_1(\mathbf{p}_b)$ and $m_2(\mathbf{p}_b)$ are the absolute values of the dot products between the gradients of $\mathbf{I}_n$ and $\mathbf{M}^{static}$, and the gradients of $\mathbf{B}_n$ and $\mathbf{M}^{static}$, computed in pixel $\mathbf{p}_b$. Also, $m_1(\mathbf{p}_b)$ and $m_2(\mathbf{p}_b)$ are the projections of the gradients in $\mathbf{I}_n$ and $\mathbf{B}_n$ over the gradient in $\mathbf{M}^{static}$. Therefore, to decide if $\mathbf{o}_j$ is an abandoned or a removed object, two counters γ_1 and γ_2 are defined: for a given boundary pixel $\mathbf{p}_b$, γ_1 is incremented if $m_1(\mathbf{p}_b) > m_2(\mathbf{p}_b)$, otherwise γ_2 is incremented. Having examined all the border pixels, $\mathbf{o}_j$ is classified as an abandoned object if $\gamma_1 > \gamma_2$, while if $\gamma_2 > \gamma_1$, $\mathbf{o}_j$ is a removed object. It is easy to appreciate how, in the proposed method, the classification between abandoned and removed objects is based solely on the characteristics of the image, and no threshold needs to be set arbitrarily as in other works [70, 91]. Moreover, a percentage of confidence ξ_j for this classification can be computed as:

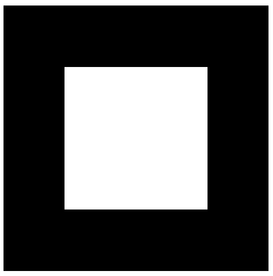
$$\xi_j = \frac{\max(\gamma_1, \gamma_2)}{\lambda_j} \tag{6.7}$$

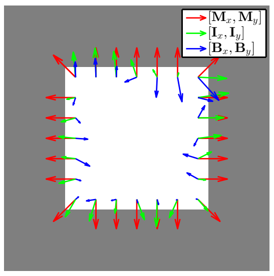
A visual example for the proposed approach is illustrated in Fig. 6.2, where the background $\mathbf{B}_n$ is a 13×13 matrix of random values in the range [0,255] (Fig. 6.2(a)). In the new frame $\mathbf{I}_n$ (Fig. 6.2(b)), the abandoned object is represented by the white square in the middle. The border projections computed as in (6.5) are illustrated in Fig. 6.2(d), while the values of m_1 and m_2 for each border pixel are plotted in Fig. 6.2(e). It can be seen that it is $m_1 > m_2$ in 20 border pixels out of 25; therefore the white blob in the binary mask $\mathbf{M}^{static}$ illustrated in Fig. 6.2(c) is correctly classified as an abandoned object, with $\xi_j = 20/25 = 80\%$.

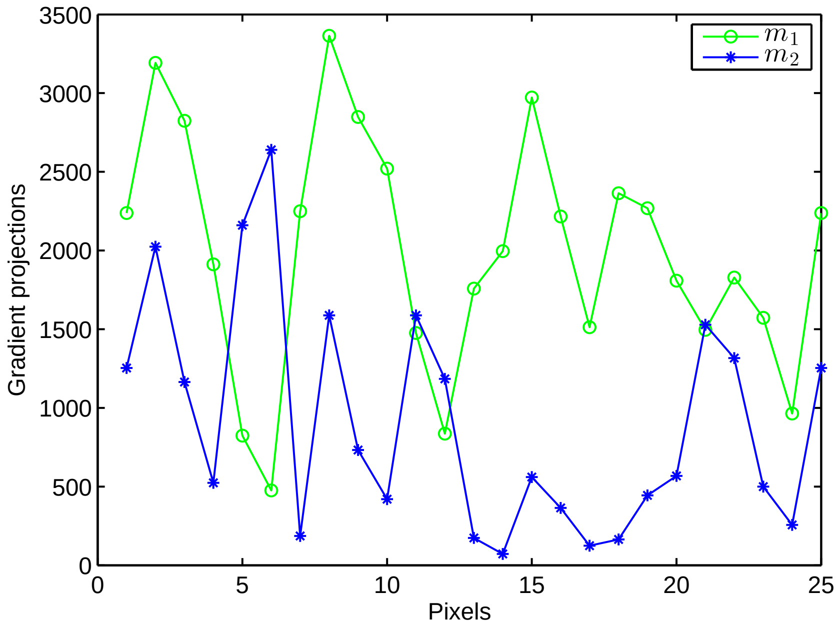
Similarly, an example of object removal is illustrated in Fig. 6.3. In this case the background image $\mathbf{B}_n$ is again a matrix of random value, with the object represented by the white square in the middle (Fig. 6.3(a)). In the new frame $\mathbf{I}_n$ (Fig. 6.3(b)), the object has been removed. By computing the border projections (Fig. 6.3(d)) and analysing the graph in Fig. 6.3(e), it can be seen that it is


(a)


(b)


(c)


[$\mathbf{M}_x, \mathbf{M}_y$]
[$\mathbf{I}_x, \mathbf{I}_y$]
[$\mathbf{B}_x, \mathbf{B}_y$]
(d)



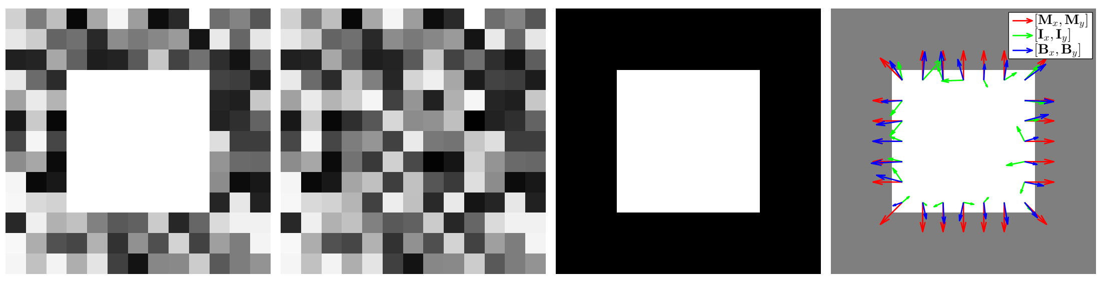
(e)

Fig. 6.2: Abandoned object detection case: (a) background image $\mathbf{B}_n$; (b) frame $\mathbf{I}_n$, with the new object represented by the white square in the middle; (c) binary mask $\mathbf{M}^{static}$; (d) boundary pixel projections, with $[\mathbf{M}_x, \mathbf{M}_y]$ illustrated in red, $[\mathbf{I}_x, \mathbf{I}_y]$ in green, and $[\mathbf{B}_x, \mathbf{B}_y]$ in blue; (e) plot of m_1 and m_2 over the object border. In 20 border pixels out of 25, it is $m_1 > m_2$; therefore the white blob in (c) is correctly classified as an abandoned object.

$m_2 > m_1$ in 24 pixels out of 25; therefore the detected object is classified as removed, with $\xi_j = 24/25 = 96\%$.

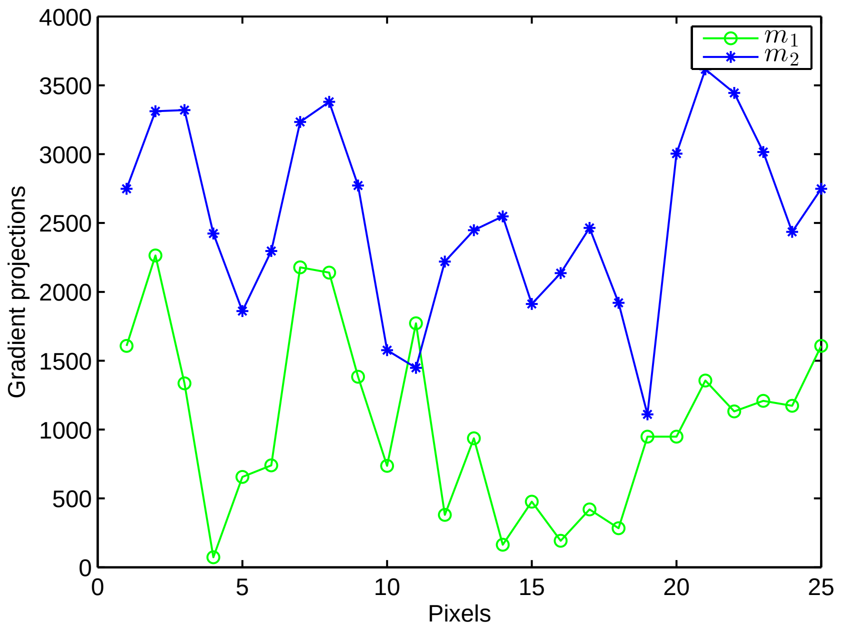
6.2.4 Object list and background update

Once all the detected objects $\mathbf{o}_k$ have been examined, the object list is updated: for each entry $\mathbf{e}_j$ in the list, the time-to-live counter ν_j is decreased by 1. If ν_j reaches zero, the entry $\mathbf{e}_j$ is discarded. The frame rate fps_n for the n^{th} frame is computed. For this purpose, the processing time τ_n for the n^{th} frame is saved as the first element in a queue containing the processing times τ_q of the last Q



$[\mathbf{M}_x, \mathbf{M}_y]$
$[\mathbf{I}_x, \mathbf{I}_y]$
$[\mathbf{B}_x, \mathbf{B}_y]$

(a) (b) (c) (d)



(e)

Fig. 6.3: Removed object detection case: (a) background image $\mathbf{B}_n$ with the object represented by the white square in the middle; (b) frame $\mathbf{I}_n$; (c) binary mask $\mathbf{M}^{static}$; (d) boundary pixel projections, with $[\mathbf{M}_x, \mathbf{M}_y]$ illustrated in red, $[\mathbf{I}_x, \mathbf{I}_y]$ in green, and $[\mathbf{B}_x, \mathbf{B}_y]$ in blue; (e) plot of m_1 and m_2 over the object border, with $m_2 > m_1$ in 20 border pixels out of 25. The detected object is correctly classified as removed.

frames, assuming a maximum achievable frame rate of Q fps. Frame rate fps_n is computed as in (6.8).

$$fps_n = \underset{b \in [0, Q-1]}{\arg\min} \left(\sum_{q=0}^{b-1} t_q \geq 1 \right) \tag{6.8}$$

The final step updates the background model $\mathbf{B}_{n+1}$ and threshold matrix $\mathbf{V}_{n+1}$

as follows:

$$\mathbf{B}_{n+1} = \begin{cases} \mathbf{I}_n, & \text{where } \mathbf{M}^{static} = 1 \\ \mathbf{B}_n, & \text{where } \mathbf{M}_n = 1 \\ (1-\alpha_n)\mathbf{B}_n + \alpha_n \mathbf{I}_n, & \text{otherwise} \end{cases} \tag{6.9}$$

$$\mathbf{V}_{n+1} = \begin{cases} 1, & \text{where } \mathbf{M}^{static} = 1 \\ \mathbf{V}_n, & \text{where } \mathbf{M}_n = 1 \\ \max(1, \mathbf{Z}), & \text{otherwise} \end{cases} \tag{6.10}$$

$$\mathbf{Z} = (1-\alpha_n)\,\mathbf{V}_n + \alpha_n \mathbf{D}_n^2 \tag{6.11}$$

The equations (6.9)–(6.11) indicate how the background is updated depending on whether a portion of the acquired image is a new static object ($\mathbf{M}^{static} = 1$), a new moving region in the scene ($\mathbf{M}_n = 1$), or just background (otherwise). The coefficient α_n is an adaptive weighting factor empirically defined as:

$$\alpha_n = \frac{1}{fps_n + 1} \tag{6.12}$$

The background $\mathbf{B}_n$ is updated using the weighting factor α_n, which in turn is updated as in (6.12) using the new frame processing rate of the algorithm from (6.8). In this way the updating rate for the background is directly related to the execution speed of the algorithm. For a fast execution, the acquisition frame rate is high and so the background updating rate must be low. For a slow execution, the acquisition frame rate is also slow, so the background needs to be updated more quickly. In the proposed algorithm, both event detection and background update adapt their "speed" accordingly to the current processing frame rate as computed in (6.8). Therefore, the complete algorithm is essentially independent of the computational capability of the machine on which it is running. Also, multiple instances of the same algorithm can simultaneously run on the same computer for different cameras.

6.3 Reactive smart surveillance system

A block diagram of the overall reactive smart surveillance system is depicted in Fig. 6.4. With respect to this, the proposed detection algorithm described in Section 6.2 is implemented in the camera processors, which can run on single or multiple machines, at different frame rates. The proposed system is designed to sit on top of any pre-existing video surveillance system, and enhance it with video

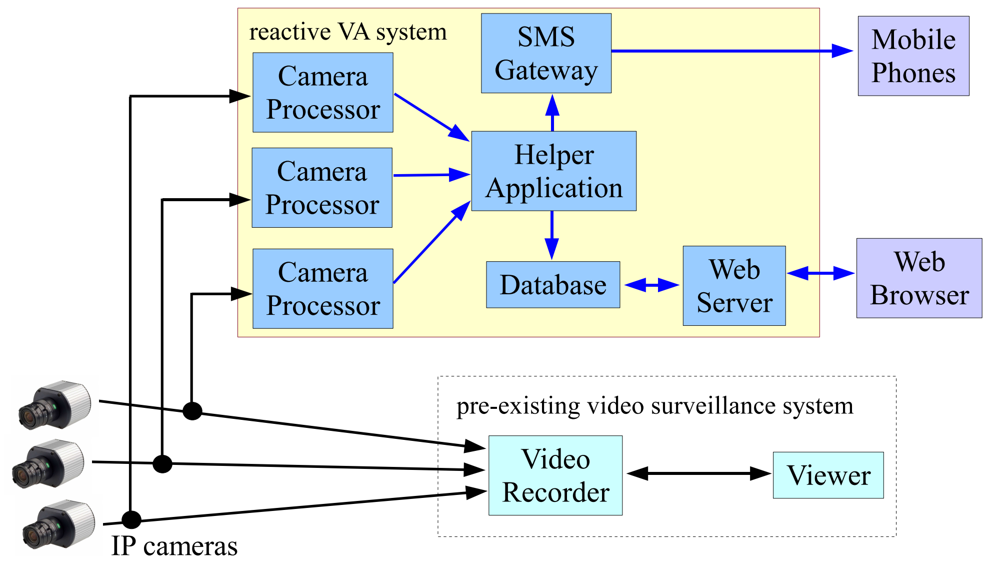
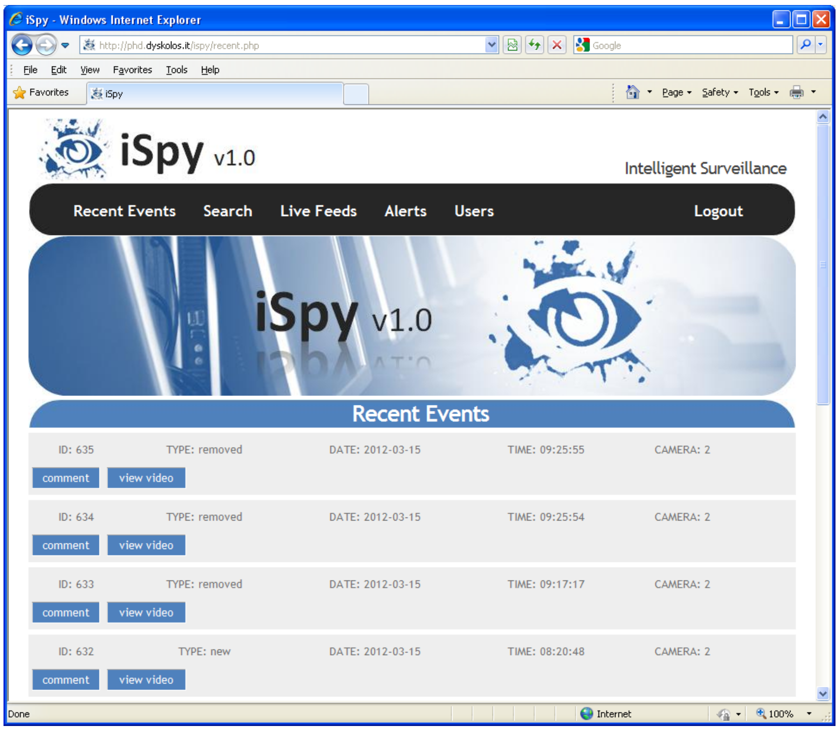


Fig. 6.4: Overview of the reactive smart surveillance system described in Section 6.3. The detection algorithm presented in Section 6.2 is implemented in the camera processors.

analytic functionalities, such as automatic detection of abandoned and removed objects. The proposed system can also be used as a standalone system, since the video feeds are acquired directly from the cameras and independently from the video recorder. The main system components are: one or more camera processors, which analyse the input video feeds; a web server with database, to store details of the detected events; a helper application, which saves event data received from the camera processors, into the database. The helper application also sends SMS alerts to registered users.

The subdivision of the camera processors and helper application ensures decoupling between video analytics and data management. In a typical scenario, an event is detected by the camera processors and the event data is sent to the helper application, which saves it into the database and notifies the registered user. By accessing the web-based user interface illustrated in Fig. 6.5, the user can review the event details and its short video clip, to take immediate counteractions. The system ensures modularity, since all the components are loosely coupled through TCP/IP communication; scalability, as multiple instances of camera processors and helper application can run simultaneously. It can also be extended since new video analytics algorithms can be added to the systems as new camera processor implementations. Finally it is a cost effective system, as all the system software can run on single or multiple desktop computers with no special hardware requirements.


iSpy - Windows Internet Explorer
http://phd.dyskolos.it/ispy/recent.php
File Edit View Favorites Tools Help
iSpy v1.0
Intelligent Surveillance
Recent Events Search Live Feeds Alerts Users Logout
Recent Events
comment
view video
Done
Internet
100%

ID: 635	TYPE: removed	DATE: 2012-03-15	TIME: 09:25:55	CAMERA: 2
ID: 634	TYPE: removed	DATE: 2012-03-15	TIME: 09:25:54	CAMERA: 2
ID: 633	TYPE: removed	DATE: 2012-03-15	TIME: 09:17:17	CAMERA: 2
ID: 632	TYPE: new	DATE: 2012-03-15	TIME: 08:20:48	CAMERA: 2

Fig. 6.5: Web-based user interface for the reactive system described in Section 6.3.

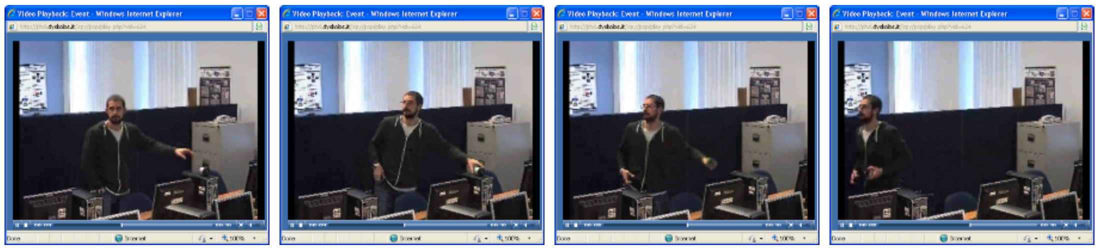


Fig. 6.6: Four frames from a removed object event video clip.

6.3.1 Event notification

Once all the event details are available, the camera processors create a low frame rate (2 – 5 fps) video clip of the past τ_{lim} seconds and asynchronously send it to the helper application described in Section 6.3, along with event details, such as time, date, camera ID and event type. Four frames of the recorded video of a removed object event are shown in Fig. 6.6. The helper application saves the event data in the web server database and issues an SMS alert to a list of pre-registered users, who can access the remote interface, to review event details and short video clip in real-time.

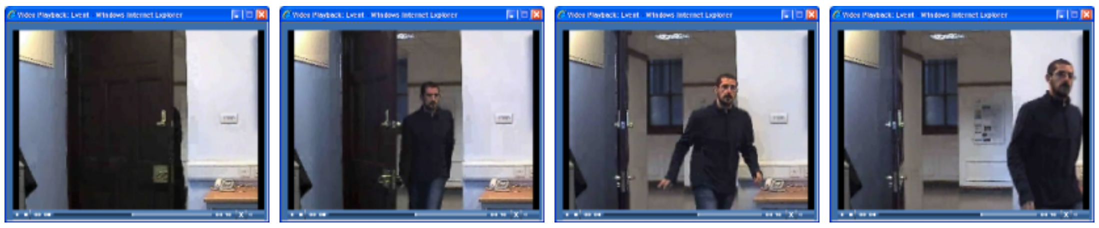


Fig. 6.7: Four frames from a door access event video clip.

6.3.2 System extension

In [142], the proposed reactive smart surveillance system is extended to incorporate door access event detection. Such a system can automatically detect whether a person enters a room and it can control a high resolution PTZ camera to zoom in on the person's face. As in the case of the abandoned and removed object detection algorithm in Section 6.2, also in the system described in [142] a low frame rate video clip of the door access event is recorded and a notification is issued to the user. More details about the door access event detection algorithm and the overall system can be found in [136, 142]. Four frames of the recorded video of a door access event are shown in Fig. 6.7.

6.4 Experimental results

In the current implementation, two Arecont Vision AV1300 IP cameras are used as acquisition devices. The video analytics part of the camera processors is implemented in Matlab.

6.4.1 Execution speed

The frame rate for the camera processor running on an Intel Core 2 Duo at 2.99 GHz is 11 fps, roughly giving a total processing time of $\tau_{total} = 91$ ms for each frame. After profiling the Matlab code, it was seen that the operation to read a new frame from the IP camera over the Ethernet takes about $\tau_{imread} = 53$ ms, which is 58% of the total processing time for each frame. This means that the pure processing time of the algorithm is $\tau_{pure} = \tau_{total} - \tau_{imread} = 38$ ms. Therefore, the algorithm is able to work in real-time and a C implementation can decrease τ_{pure} dramatically. Moreover there are two methods to reduce τ_{imread}: (i) using different acquisition devices, and (ii) directly connecting the IP cameras to the machine on which the camera processors are running. These considerations suggest that τ_{pure} and τ_{imread} could be simultaneously reduced by an implementation of the same

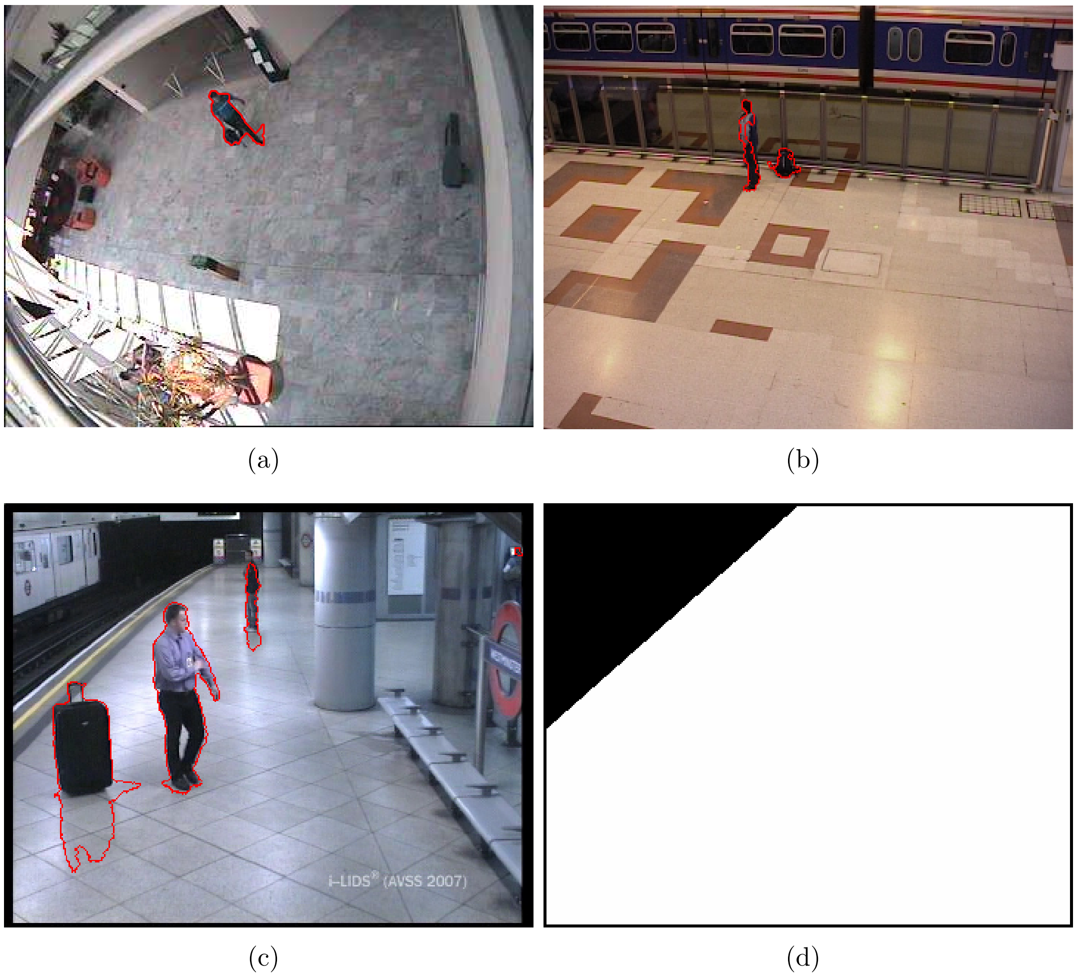


Fig. 6.8: Frames from sequences from (a) CAVIAR, (b) PETS2006 and (c) AVSS2007 datasets. In (d), it is illustrated the binary mask used for AVSS2007 sequences to select the platform has region of interest.

algorithm on a dedicated multimedia DSP, directly connected to the IP cameras.

6.4.2 Detection rate

For the detection capabilities of the system, in an office-like setup, true positive rate is 95%, while the false positive rate is 10%. The proposed system has also been tested on the CAVIAR dataset for left objects [143], on the subway sequences in the AVSS2007 dataset [85], and on the PETS2006 dataset for left-luggage scenarios [83]. Visual results are illustrated in Fig. 6.8.

In the CAVIAR sequences (Fig. 6.8(a)) "LeftBag", "LeftBag_PickedUp", the algorithm has correctly detected the object of interest, as "abandoned" first, and then as "removed". In "LeftBox", the object has been correctly detected as "abandoned". In the "LeftBag_AtChair" sequence, the object has mistakenly been detected as “removed” in both cases. However, the confidence for the incorrect case was only of 58%. No false positives have been detected.

For the PETS2006 dataset (Fig. 6.8(b)), camera 3 sequences for S1, S2, S5, S6 and S7 have been used. For all the sequences, the left item has been detected as “abandoned”. Moreover, in S2, removed object events are also triggered for the purple bins initially included in the background, and then removed from the scene.

In the AVSS2007 abandoned baggage scenario (Fig. 6.8(c)), the left item has been correctly detected as “abandoned” in all the three sequences (“AVSS AB Easy”, “AVSS AB Medium” and “AVSS AB Hard”). However, since no person classifier is implemented at this stage in the proposed algorithm, some false positives are detected for people standing or sitting motionlessly for a long period of time. In these three sequences, a binary mask as illustrated in Fig. 6.8(d) has been used to select a region of interest, i.e. the platform, and avoid false positives from the trains in the top left corner of the scene.

For PETS2006 and AVSS2007, the detection time has been set to 30 seconds, while for CAVIAR it has been set to 5 seconds, which is also the default value for the algorithm.

6.4.3 Abandoned and removed classification

For all the true positives, the rate of correct decision between an object being abandoned or removed is approximately 85%. To prove that the proposed method for abandoned and removed object classification works also with textured backgrounds, the images of a cup and of its non-homogeneous background are analysed. Using the method described in Section 6.2.3, the cup in Fig. 6.9(a) is classified as an abandoned object, with a confidence of $\xi_j = 64\%$ computed as in (6.7). The output of the segmentation process is shown in Fig. 6.9(c). The synthetic image shown in Fig. 6.9(d) is created by overlapping the detected object from Fig. 6.9(c) on the highly textured background shown in Fig. 6.9(e). The result of the segmentation process for the synthetic image is shown in Fig. 6.9(f). In this case the proposed method has correctly classified the detected object as abandoned, with a confidence of $\xi_j = 57\%$.

6.4.4 Discussion

The proposed algorithm for real-time detection of abandoned and removed objects has been proven to work in real scenarios and on standard test sequences.

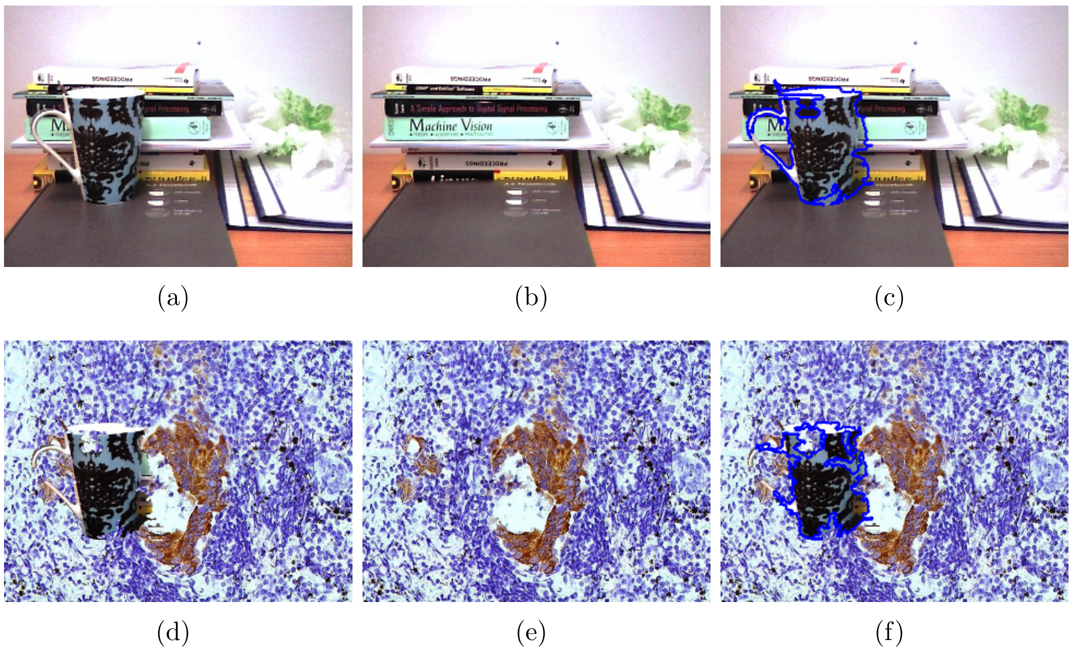


Fig. 6.9: Abandoned and removed object classification: (a) abandoned object image; (b) background; (c) detected object; (d) synthetic image; (e) synthetic background; (f) detected object in the synthetic image.

The novelty of the algorithm is not in the detection rate, as it is similar to other algorithms in literature, but rather in its computational efficiency and adaptive behaviour. The algorithm works on machines with different computational capabilities, where the detection time and background update rate are independent from the underlying architecture. This characteristic allows the integration of the presented algorithm in a reactive real-time surveillance system, for quick detection and immediate notification of events of interest to registered users, as described in Section 6.3.

6.5 Conclusion

This chapter has presented a novel adaptive algorithm for automatic detection of abandoned and removed objects, in the context of a reactive smart surveillance system. A detection time can be directly specified by the user and the background is "healed" only after an abandoned or removed object event has been detected. The actual detection time and the background model updating rate are computed adaptively with respect to the algorithm frame processing rate, so that even on different machines the detection time is generally the same. A novel technique based on the gradient projections in the object boundary pixels has also been described, to correctly classify detected static objects as either abandoned or removed.

The overall system makes extensive use of IP technologies, to ensure communication among components and remote availability of the system resources, such as IP cameras, event database and user front-end. Despite its simplicity, the proposed algorithm performs well and it is suitable for real-time detection.

Chapter 7

Decentralised multi-camera tracking system

7.1 Introduction

In the context of multi-camera systems, spatio-temporal information extracted from a set of semantically clustered cameras can be fused together and exploited, to achieve a better understanding of the surrounding environment and monitor areas wider than a single camera FOV [3]. Each sensor can be associated with one or more VA processing tasks [32, 144], to distribute the surveillance workload among cameras and decentralise it towards the edges of the network. This approach produces a collaborative, or co-operative, network of smart surveillance sensors [145–147]. To fully exploit the information gathered by a smart sensor network, both topological and geographical layouts of the network are required. While the former defines which cameras have overlapping FOVs, the latter specifies the position in space of each sensor, with respect to a common coordinate frame.

From the network topology it is possible to know, at any time, which other cameras should be "seeing" a specific target in the FOV of a given camera. If overlapping cameras can simultaneously detect the same target, information from multiple FOVs can be merged together to obtain a better representation of the target. A smart surveillance system can also automatically select, from a set of views, the one that gives the best visualisation of the target [148, 149]. Moreover camera overlapping can be exploited to overcome target occlusion [62, 150, 151]. Many algorithms have been proposed in literature for multi-camera system calibration [152–154].

When colour is assumed as a discriminative feature of the target, it is possible to use the MS tracker [11]. Although it has been shown to be fast, effective and robust in many scenarios, it fails in case of occlusion or fast moving targets as reported in Section 3.4. In the improved MS tracker introduced in Chapter 5, tracking performance is evaluated with respect to a threshold computed empirically based on the characteristics of the image, when the target is first selected. This threshold is automatically updated during tracking. The candidate points to restart the tracker after complete occlusion or target loss are selected based on colour similarity with the target. These are validated by the MS procedure itself.

Assuming that colour is the target detection feature, this chapter describes a novel decentralised multi-camera system for collaborative tracking, exploiting the improved MS algorithm described in Chapter 5. For this purpose, at the initialisation step only knowledge of the topological layout of the network is required, simply in terms of occurrence or not of overlapping between FOVs, without the actual degree of overlapping being important. The novel contribution of the work presented in this chapter is twofold: first, a highly flexible decentralised system software architecture is introduced, for decentralised multi-view target tracking, where synchronisation constraints among processes can be relaxed. Secondly, the improved MS tracker from Chapter 5 is extended to a collaborative multi-camera environment, wherein algorithm parameters are set automatically in separate views, upon colour characteristics of the target. Unlike the approaches in [155, 156], the decentralised multi-camera tracking system in this chapter does not rely on camera positional information to initialise the trackers or handle camera hand-off events. Tracking in separate camera views is performed solely on the visible characteristics of the target, reducing the system setup phase to the minimum.

The remainder of the chapter is organised as follows. Section 7.2 gives a description of the multi-camera system architecture, while Section 7.3 reports on the target detection approach adopted in the system. Details about the target information storage are given in Section 7.4, while Section 7.5 describes initialisation and operation of the tracking algorithm implemented for each camera view. Details of the data collection and collation performed to achieve multi-camera target tracking are provided in Section 7.6. Experimental results are reported in Section 7.7, while Section 7.8 comments on the scalability aspects of the system. Finally Section 7.9 concludes the chapter.

7.2 Multi-camera system architecture

From a conceptual point of view, multi-camera target tracking can be divided into four tasks:

1. initial target detection upon occurrence of an event of interest;
2. target status storage and broadcasting;
3. target tracking in each separate camera view;
4. data collection from all the trackers and collation.

Such tasks can be mapped into separate processes, which share information with each other to achieve the higher level goal of the system, i.e. target tracking across multiple views. Therefore four types of processing entities are defined:

- a detection agent (DA);
- a tracking agent (TA);
- a status server (SS);
- a data server (DS).

Fig. 7.1(a) shows the layout of a simple implementation of the system with two cameras $\mathcal{C}^1$ and $\mathcal{C}^2$, two TAs T_1 and T_2, two DAs D_1 and D_2, a SS and a DS.

As the task of the TA is to track a target in a single camera view, a separate TA must be associated with each camera in the system. Similarly, since it is desirable to be able to detect events of interest in each view, a separate DA can be associated with each camera. On the other hand, it is reasonable to have a single SS acting as a central hub that receives detections from single DAs and broadcasts these to all the TAs. Concerning data collection, a single DS can act as a sink for the tracking information produced by all the TAs. Such a DS has an overall view of the trackers and can merge all their data to compute a multi-camera track of a given target.

In general, for a multi-camera tracking system with N cameras, there is an equal number of TAs and DAs. Tracking agent T_n and detection agent D_n are associated with the n^{th} camera $\mathcal{C}^n$, with $n \in [1, N]$, and $\mathbf{F}_i^n$ is the i^{th} frame in the n^{th} camera view. For all the agents, a single SS and DS are available. As illustrated in Fig. 7.1(b), target information is first sent by a DA – D_1 in this case – to the SS, upon occurrence of an event of interest. The SS broadcasts this information to T_1 and T_2 (Fig. 7.1(c)). The TAs send their tracking results to

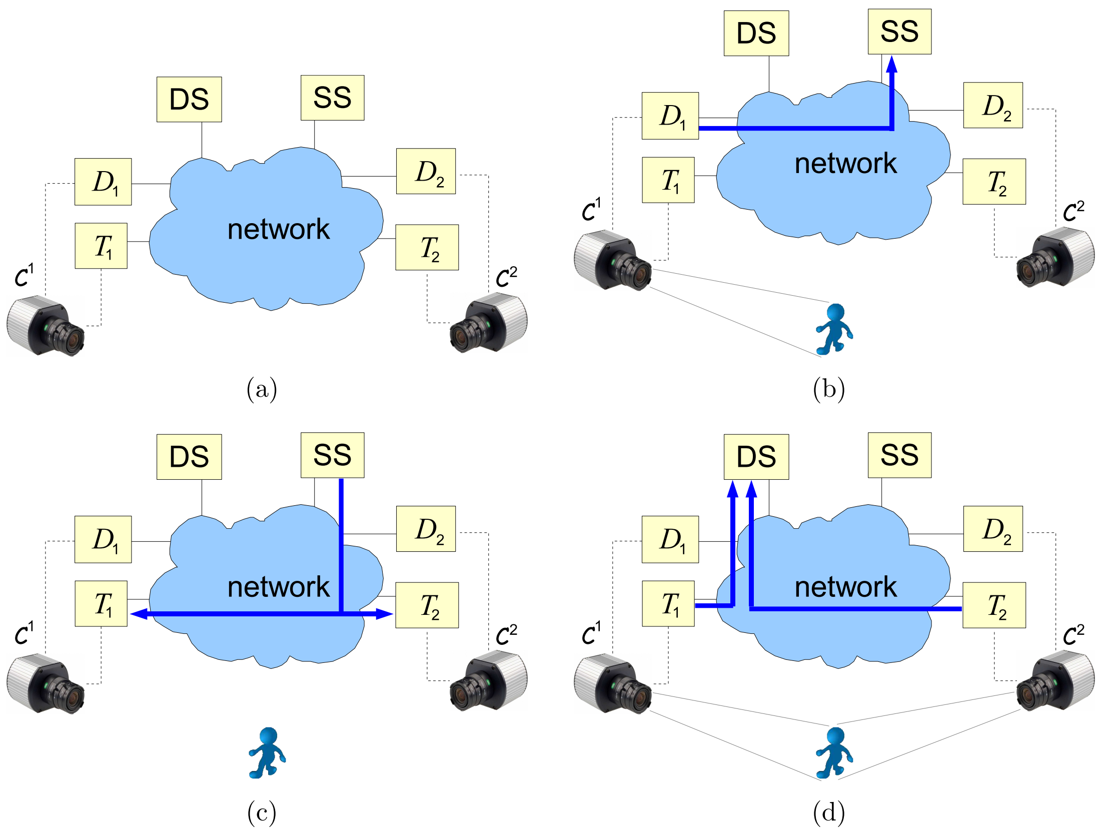


Fig. 7.1: System implementation with two cameras $\mathcal{C}^1$ and $\mathcal{C}^2$ (a). The DA D_1 detects a target and sends data to the SS (b). The SS broadcasts target information to both TAs T_1 and T_2 (c). The TAs track the target and send their results to the DS (d).

the DS (Fig. 7.1(d)), which merges them in an attempt to resolve possible inconsistencies and produce a unique multi-camera track of the target, as explained in Section 7.6.

All agents and servers are loosely coupled, as they share their information through simple messages over the network. The system architecture is therefore highly flexible and multiple configurations are possible to deploy the processing entities on physical processing units. For example:

- all entities can run on the same machine; or
- SS and DS can run on the same machine, while each set of TA and DA associated with the same camera runs on a different machine; or
- each entity runs on a dedicated machine.

Without loss of generality, this chapter deals with a single target, as multi-target tracking can be achieved by replicating the TA at each camera, for every new target to track.

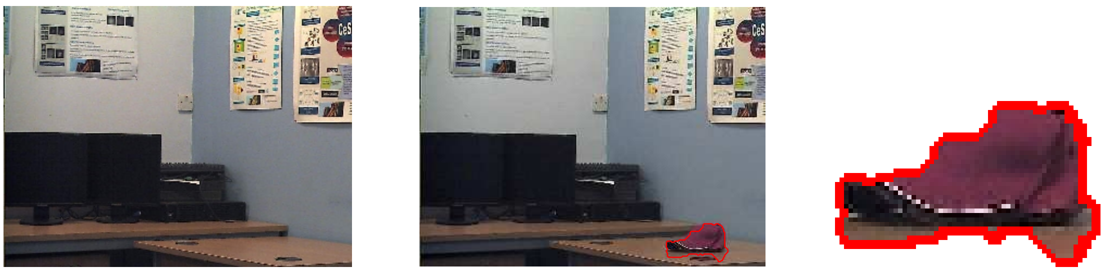

(a) (b) (c)

Fig. 7.2: DA target detection: (a) background image; (b) new frame $\mathbf{F}_i^n$ with the selected target in it; (c) close-up of the portion of $\mathbf{F}_i^n$ sent to the SS.

7.3 DA – Target detection

As explained in the previous section, any DA in the system can select a target within its FOV, upon occurrence of a predefined event of interest. The actual detection algorithm implemented in the DA is application-dependent, and one could either manually select the target, or apply one of the many automatic event detection algorithms proposed in the literature, as for example the ones described in [13, 79, 157].

In the context of this thesis, the adaptive algorithm for the detection of abandoned and removed objects presented in Section 6.2 has been implemented in the DAs. However, differently from the reactive system described in Section 6.3, the detection algorithm in the DA D_n sends to the SS only the camera number n and the portion of frame $\mathbf{F}_i^n$ corresponding to the selected target, as illustrated in Fig. 7.2. In particular, Fig. 7.2(a) shows the background image; Fig. 7.2(b) shows the frame $\mathbf{F}_i^n$ with the selected target in it, while Fig. 7.2(c) shows a close-up of the portion of $\mathbf{F}_i^n$ sent to the SS.

7.4 SS – Target information storage

The SS receives data from a DA, i.e. the portion of frame $\mathbf{F}_i^n$ representing the detected object as illustrated in Fig. 7.2, and stores it along with a timestamp, the camera number n and a unique identification number ξ for the target.

The SS acts as a sink for all the DA detections, while it follows a publisher-subscriber pattern with respect to the TAs. More specifically, the SS is always running and the TAs subscribe to it at setup time. When the SS receives target data from a DA, it broadcasts such information to all its subscribers. The TAs can then look for the new target in their FOVs, and track it if present, as described in Section 7.5. The SS is the point of connection between DAs and TAs and its role is merely to store and forward target data.

7.5 TA – Single view target tracking

The tracking algorithm implemented in the TAs is based on the improved MS tracker for single camera view described in Chapter 5, although the initialisation step has been modified, to accommodate the multi-camera nature of the system.

7.5.1 Initialisation

For multi-view colour tracking, colour calibration among all the cameras is required. In the proposed multi-camera system, the Gray World Assumption [60] is used to colour-normalise all camera views. More accurate techniques for multi-camera colour calibration are described in [62, 158–160]. Also, to make the system more resilient to different lighting conditions in separate views, the luminance component Y is removed and only the red and blue chrominance Cb and Cr are used from the YCbCr colour space, to compute the colour distributions **Q** and **P**. The number of histogram bins is $\beta = 64$. A difference in the system in this chapter with respect to the improved MS tracker initialisation described in Section 5.5 is in the coefficient ω_r defining the colour range width, which is computed as:

$$\omega_r = \lfloor \sigma_r \rfloor - 1 \tag{7.1}$$

This allows a wider colour range, i.e. $\approx 2\sigma_r$, for both Cb and Cr colour planes, with σ_r being the standard deviation of the quantised values of the selected target pixels.

Knowledge of the camera topology is required for a correct setup of each TA. For this purpose the system only needs to know which cameras have overlapping FOVs, while the exact 3D position of the cameras and the degree of view overlapping are not necessary. Therefore the required information can be easily encoded with a look-up table stating whether two cameras overlap or not. Such information can be manually provided to each TA at setup time.

For a target ξ selected by DA D_n in camera $\mathcal{C}^n$, the initialisation of the TA T_n is the same as the one described in Section 5.5. For the other $N-1$ trackers T_m with $m \neq n$, their initialisation varies depending on whether the FOVs of cameras $\mathcal{C}^n$ and $\mathcal{C}^m$ overlap or not.

Non-overlapping camera views

If cameras $\mathcal{C}^n$ and $\mathcal{C}^m$ do not overlap, target ξ selected at a given time instant in $\mathcal{C}^n$ is certainly not present in $\mathcal{C}^m$. Clearly the position in $\mathcal{C}^m$ of the best match for the given target model $\mathbf{Q}$ refers to an object which is not the target. Therefore the threshold τ_m for tracker T_m is computed as:

$$\tau_m = \arg\min_{\mathbf{o}_l^m} [d(\mathbf{o}_l^m)] \tag{7.2}$$

where $\mathbf{o}_l^m$ are the candidate points selected as in the tracker initialisation step in Section 5.5, but with no spatial constraints on their position in the initial frame $\mathbf{F}_0^m$, i.e. Eq. (5.12) is not reinforced.

After initialisation, tracker T_m proceeds to the failure recovery step described in Section 5.4, as by definition target ξ was not present in $\mathbf{F}_0^m$, so the initial target position in $\mathbf{F}_1^m$ is undefined. Here it is assumed that, when the target ξ enters the FOV of camera $\mathcal{C}^m$, its colour distribution will have a distance $d(\cdot)$ from the target model $\mathbf{Q}$ smaller than any other object in the frame $\mathbf{F}_0^m$ and therefore smaller than τ_m. So the tracker T_m can successfully start to track the correct target ξ in $\mathcal{C}^m$.

Overlapping camera views

If cameras $\mathcal{C}^n$ and $\mathcal{C}^m$ have overlapping FOVs, it means that target ξ should be present in both $\mathbf{F}_0^m$ and $\mathbf{F}_0^n$. The failure recovery procedure applied to $\mathbf{F}_0^m$ gives the position of the best match for $\mathbf{Q}$ in the frame. This best match should be the correct target ξ, assuming that its distribution minimises the distance $d(\cdot)$ from $\mathbf{Q}$. So tracker T_m finds the correct target ξ in $\mathbf{F}_0^m$ and then proceeds to the next frame $\mathbf{F}_1^m$.

If the target ξ is hidden or not visible in $\mathbf{F}_0^m$, the best match found in the frame by the failure recovery procedure will refer to an incorrect target ξ'. This misdetection is automatically corrected by the tracker as soon as target ξ is visible again in camera $\mathcal{C}^m$, as ξ gives a higher match than ξ' and therefore ξ is selected as target to track. However, an initial value for the threshold τ_m is computed anyway and it is updated by the tracker T_m in the next frames of $\mathcal{C}^m$.

This approach allows an initial value of the threshold for each camera view to be defined. Such threshold values are soon tuned by the trackers, so potential initial misdetections are confined to a small number of frames. Moreover possible tracking inconsistencies across different views can be resolved by the DS at data collation time as explained in Section 7.6.

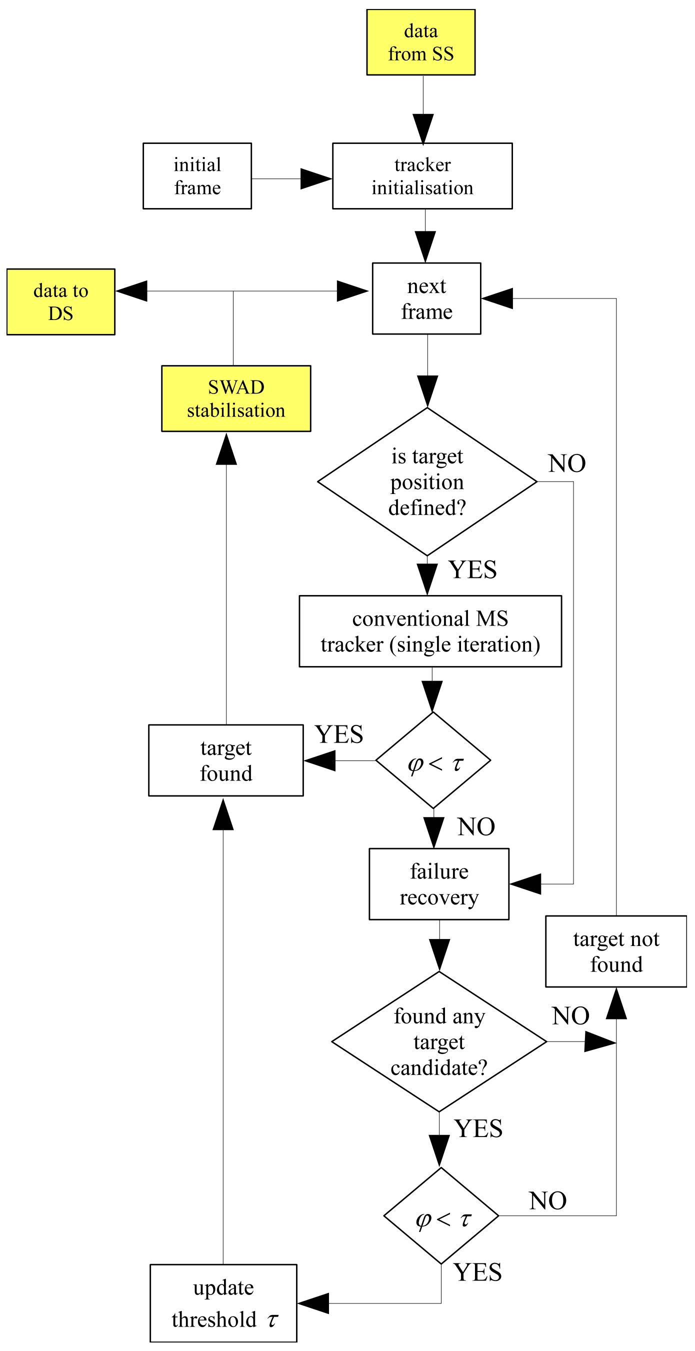


Fig. 7.3: Block diagram of the tracking algorithm implemented in the TAs.

7.5.2 Tracking

A block diagram of the tracking algorithm implemented in the TAs is illustrated in Fig. 7.3. With respect to the one in Fig. 5.1, the block diagram in Fig. 7.3 has three additional blocks:

- “data from SS” which represents the message sent by the SS to the TAs once an object has been detected;

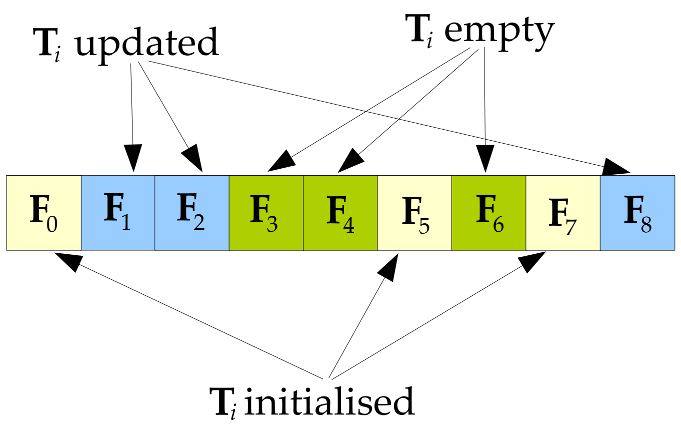


Fig. 7.4: A set of 9 consecutive frames. Frames $\mathbf{F}_3$–$\mathbf{F}_4$, $\mathbf{F}_6$ are U-frames, i.e. the target is not found. Frames $\mathbf{F}_0$–$\mathbf{F}_2$, $\mathbf{F}_5$, $\mathbf{F}_7$–$\mathbf{F}_8$ are D-frames, i.e. the position $\mathbf{y}_i$ of the target is defined.

- "data to DS" which represents the message sent by a TA to the DS;
- "SWAD stabilisation" which uses the SWAD-based tracker described in Section 4.2, to improve the precision of the single view tracker, over a set of consecutive frames wherein the target has been found.

For this purpose, the algorithm keeps a target template $\mathbf{T}_i$. However, as the stabilisation step is performed only when the target has been found in the current frame $\mathbf{F}_i$ and the target position $\mathbf{y}_i$ is available, $\mathbf{T}_i$ is left undefined when $\mathbf{y}_i$ is not available.

For a set of consecutive frames in which $\mathbf{y}_i$ is defined and therefore the target is visible, the template $\mathbf{T}_i$ is re-initialised in the first frame of the set. For the following frames in the set the SWAD matching is carried out and $\mathbf{T}_i$ is updated using (4.1) from Chapter 4.

A visual representation of such an approach is given in Fig. 7.4, where a set of 9 consecutive frames is depicted. Frames where the target is not found are referred to as U-frames ($\mathbf{F}_3$–$\mathbf{F}_4$, $\mathbf{F}_6$), while frames where $\mathbf{y}_i$ is defined are referred to as D-frames ($\mathbf{F}_0$–$\mathbf{F}_2$, $\mathbf{F}_5$, $\mathbf{F}_7$–$\mathbf{F}_8$). As it can be seen, the target template $\mathbf{T}_i$ is:

- empty – in all the U-frames ($\mathbf{F}_3$–$\mathbf{F}_4$, $\mathbf{F}_6$);
- re-initialised – in the first D-frame after a U-frame ($\mathbf{F}_0$, $\mathbf{F}_5$, $\mathbf{F}_7$);
- updated – in the consecutive D-frames ($\mathbf{F}_1$–$\mathbf{F}_2$, $\mathbf{F}_8$).

When the template $\mathbf{T}_i$ is re-initialised, it is set equal to a portion of $\mathbf{F}_i$ around $\mathbf{y}_i$. The dimensions $h_T \times w_T$ of $\mathbf{T}_i$ are equal to the current target size $h_i \times w_i$.

The template dimensions stay the same for all the following D-frames, wherein the SWAD matching is performed, until $\mathbf{T}_i$ is re-initialised.

The position $\mathbf{p}_i$ of the best match $\hat{\mathbf{T}}_i$ for $\mathbf{T}_i$ in $\mathbf{F}_i$ obtained by minimising the SWAD coefficient $\psi(x, y)$ becomes the final target position in the current frame $\mathbf{F}_i$, i.e. $\mathbf{y}_i = \mathbf{p}_i$. After this, the template is updated as mentioned above. As in this case one is mainly interested in stabilising the position of the target across single pairs of D-frames, the target template should adapt quickly and so a higher weight in (4.1) should be given to the best match $\hat{\mathbf{T}}_i$, rather than the previous template $\mathbf{T}_i$. Therefore the blending factor α is assigned a value higher than 0.5. In the current implementation of the tracker, it is $\alpha = 0.8$.

7.6 DS – Data collation and multi-view target tracking

In the presented multi-camera system, the DS collects tracking results from all the TAs and collates them, to remove inconsistencies and create a unique coherent multi-view track of the selected target.

A multi-camera tracking system requires some form of synchronisation among all the trackers, to be able to reliably collate target tracks from different views. Although it is possible to synchronise multiple machines using the Network Time Protocol (NTP), it is difficult to ensure temporal synchronisation at a frame level among multiple trackers. Therefore the presented system adopts a notion of temporal synchronisation in terms of tracker detections falling in the same time interval defined by the DS. This means that the DS defines a time line of consecutive time slots t_s^{DS} of given temporal length Δt^{DS}. The TAs run at their own specific pace and regularly send their tracking results to the DS. Tracking detections sent from different TAs and received by the DS within the same time slot t_s^{DS} are considered to be synchronous.

Similarly, each TA defines a time line of consecutive time slots t_s^{TA} of temporal length Δt^{TA}, and tracking detections in separate frames falling within the same time slot t_s^{TA} are accounted for as a unique tracking detection. At the end of each time slot t_s^{TA}, the TA sends its tracking results to the DS if a target was detected in such a time interval; otherwise no transmission takes place. The information sent by the TA T_n includes the camera number, the target number ξ, a timestamp and the value of the highest match, computed as $\lambda_n^{\xi} = 1 - d(\cdot)$, for all the tracking detections in time slot t_s^{TA}. In this context, timing constraints for the overall system are relaxed and Δt^{DS} and Δt^{TA} can be set to accommodate the application requirements. As a guideline, assuming that the DS processes the TA notifications sequentially, the lengths of Δt^{DS} and Δt^{TA} can be set according to:

$$\begin{aligned} \Delta t^{DS} &> N\Delta t^{TX} + \Delta t^{TA} \\ \Delta t^{TA} &> \Delta t^{TX} \\ \Delta t^{TA} &> \tfrac{1}{fps^{TA}} \end{aligned} \tag{7.3}$$

where N is the number of TAs sending information to the DS, Δt^{TX} is the transmission time from TA to DS and fps^{TA} is the TA processing rate. The "frame rate" for the DS is $1/\Delta t^{DS}$. For example, assuming $N = 10$, $\Delta t^{TX} = 2$ ms and $fps^{TA} = 10$, one can cautiously choose $\Delta t^{TA} = 150$ ms and $\Delta t^{DS} = 200$ ms; in this case the DS frame rate is 5. It is clear however that a performance-driven implementation of the system would allow a much higher frame rate for the DS.

At the end of each time slot t_s^{DS}, the DS knows which TAs have detected the target ξ in their FOV and which have not. Moreover it can use the match value of each detection as a level of confidence for it. When any two TAs, T_n and T_m, send a tracking detection to the DS for the same target ξ, in the same time slot t_s^{DS}, two cases are possible:

- the two TAs have overlapping FOVs and their tracking results are consistent;
- the two FOVs do not overlap and therefore the results are inconsistent.

When inconsistencies are present, different application-dependent strategies can be applied to solve them. One possible solution would be to select as "correct" the tracking result with the highest match associated, between T_n and T_m. In this approach, the DS selects as best view for the time slot Δt^{DS} the view associated with the tracker that returned the highest match value; therefore the DS can generate a single continuous video stream made up of the portions of video feeds coming from the selected best views, at each Δt^{DS}. This approach is the one currently implemented in the described multi-camera system. Such a simple strategy is also useful in overlapping cameras, to be able to select the best view of the target in case of occlusion, as shown by the example reported in Section 7.7.2.

A drawback of this approach is the possible rapid back-and-forth switching between FOVs. This might be the case of correct detections in overlapping cameras or single misdetections in general. Again, various application-dependant strategies are available. For example, in case of overlapping views, back-and-forth

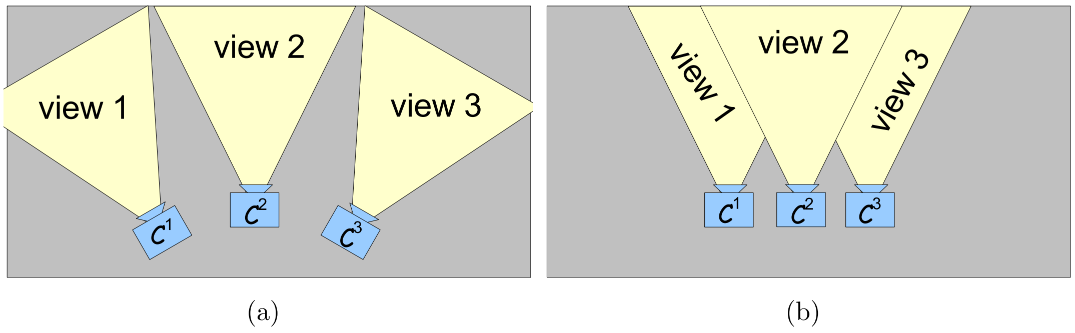


Fig. 7.5: Camera topologies for first experiment: (a) no overlap; (b) overlap.

switching between two cameras can simply be skipped to preserve fluidity of the generated multi-camera video stream. For non-overlapping cameras, time-spatial constraints can be taken into account to eliminate inconsistencies, by checking whether it is possible or not for a detected target to sequentially appear within two FOVs which are physically far apart from each other.

Although in the presented system the actual tracking of the target takes place in the TAs in a decentralised fashion using the improved MS described in Chapter 5, the DS then gathers all the tracking detections and collates them. Therefore it can be argued that the DS indirectly performs target tracking across multiple camera views.

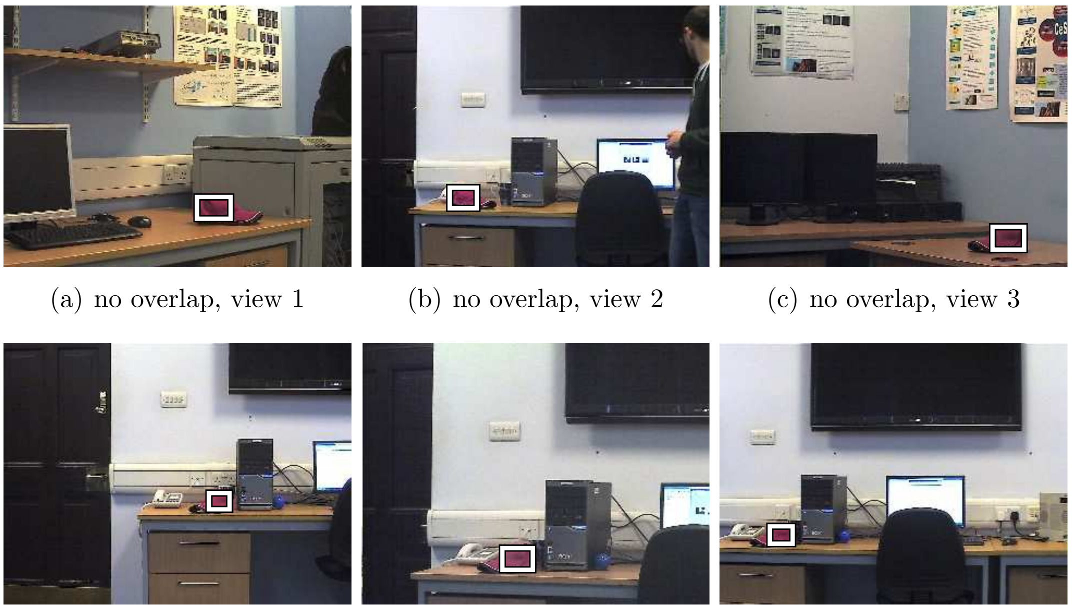
7.7 Experimental results

To evaluate the multi-camera tracking system described in this chapter, its tracking performance are assessed numerically first, and then visual results of its best view selection capability are given. The system software is implemented in Matlab and Java. In particular, the tracking block of the TA components is implemented in Matlab, while the communication blocks, the SS and the DS are implemented in Java. The computer used to run the experiments is an Intel Core 2 Quad CPU at 3 GHz, with 3.00 GB of RAM.

7.7.1 Tracking performance

The tracking performance of the system is numerically evaluated in terms of precision, recall, specificity and accuracy as in (3.20)–(3.23) reported in Chapter 3.

As a first experiment, two indoor multi-camera video sequences, acquired from three IP cameras, have been recorded, respectively one with three overlap-



(a) no overlap, view 1 (b) no overlap, view 2 (c) no overlap, view 3

(d) overlap, view 1 (e) overlap, view 2 (f) overlap, view 3

Fig. 7.6: Indoor testing: non-overlapping views in top row; overlapping views in bottom row.

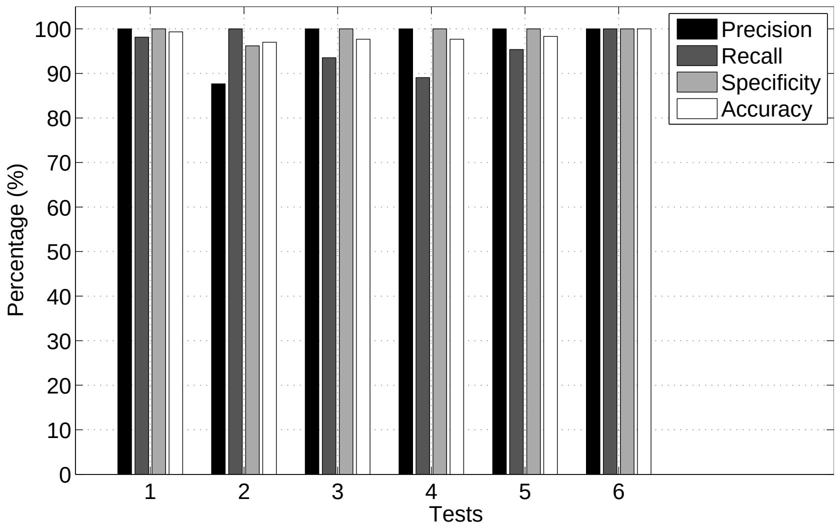


Fig. 7.7: Results for indoor testing with non-overlapping views.

ping views and one with three non-overlapping views as illustrated in Fig. 7.5, to assess the initialisation step described in Section 7.5.1. In these sequences, the target is a pink cap being carried from one camera view to the other and experiencing complete occlusion. Three images of the detected target for the non-overlapping and overlapping cases are shown in Fig. 7.6, in the top and bottom row respectively. In this experiment, for both overlapping and non-overlapping sequences, the target has been manually selected in one of the views and then

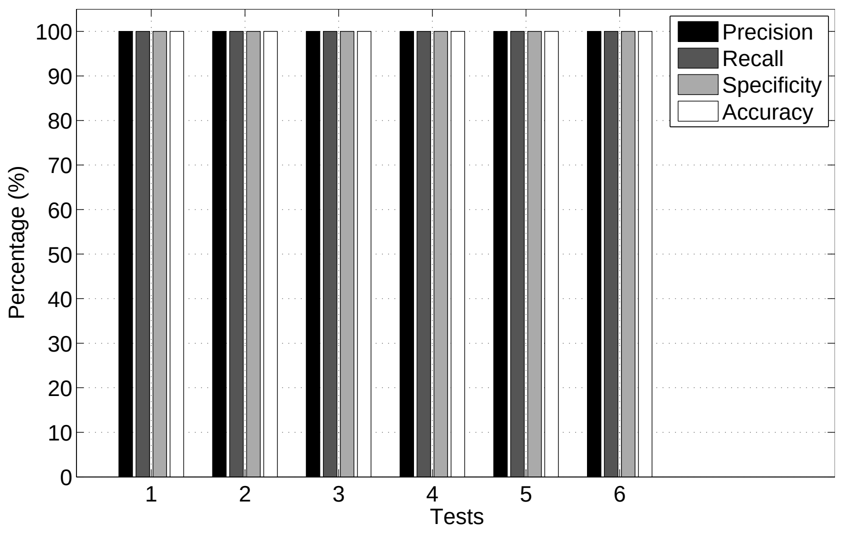


Fig. 7.8: Results for indoor testing with overlapping views.

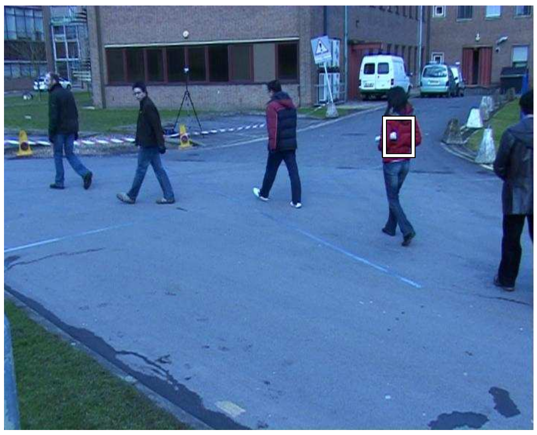
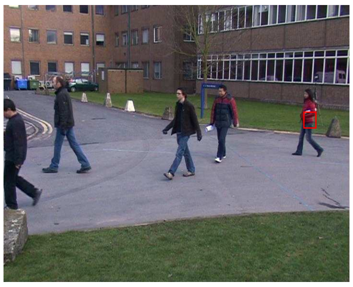
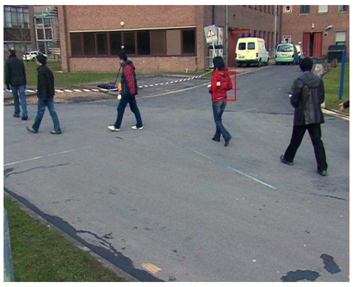


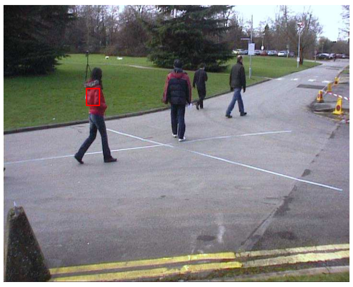
Fig. 7.9: Target selected in PETS2009 S3-Multiple-12.43-06 (sequence 2), frame #72.

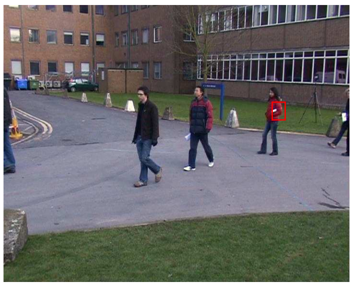
tracked in the other two views. This is repeated for all views. Therefore, a total of six tests (1-6) have been obtained for both overlapping and non-overlapping cases. Numerical values of precision, recall, specificity and accuracy of such tests compared with the manually labelled ground truth are illustrated in Fig. 7.7 for the non-overlapping case, and in Fig. 7.8 for the overlapping case. It can be seen that in general the system has high precision, recall, specificity and accuracy, i.e. more than 85% in all 12 tests. In the overlapping case in particular, neither FP nor FN have been detected (Fig. 7.8). These results indicate that the system has very good tracking performance in an indoor environment.

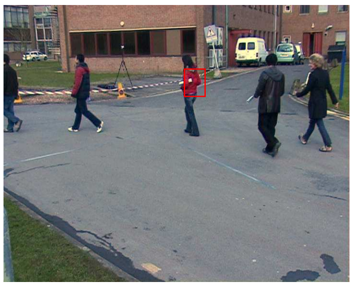
As a second experiment, six entire video sequences from the PETS2009 dataset have been used: (1) S3-Multiple-12.43-05, (2) S3-Multiple-12.43-06, (3) S3-Multiple-12.43-07, (4) S2-L1-12.34-05, (5) S2-L1-12.34-06 and (6) S2-L1-12.34-07. The camera topology for these videos is available at [131]. The target is the woman

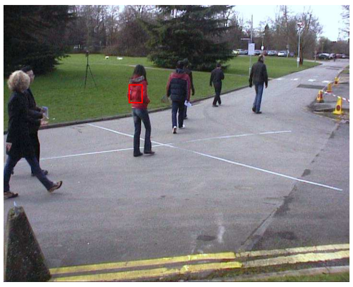

(a) seq. 1, frame #80

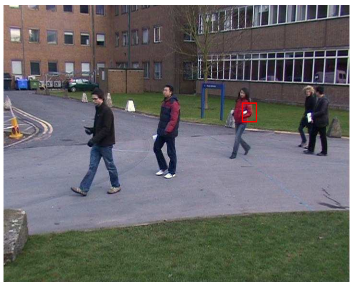

(b) seq. 2, frame #80

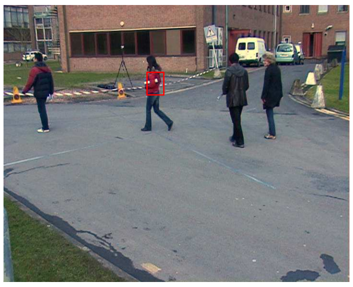

(c) seq. 3, frame #80

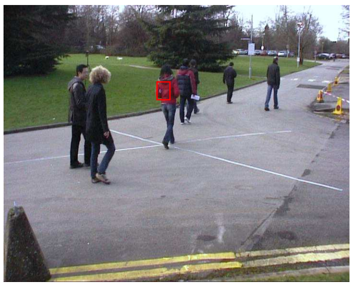

(d) seq. 1, frame #86

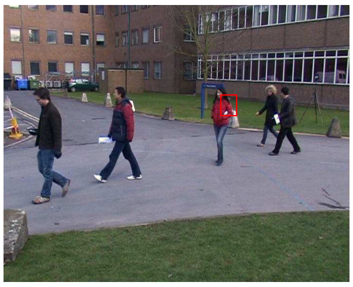

(e) seq. 2, frame #86

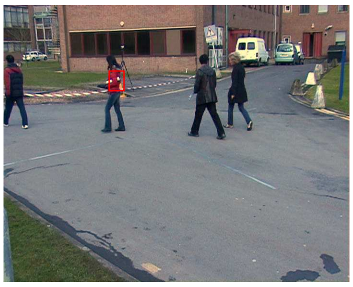

(f) seq. 3, frame #86


(g) seq. 1, frame #92


(h) seq. 2, frame #92


(i) seq. 3, frame #92


(j) seq. 1, frame #98


(k) seq. 2, frame #98

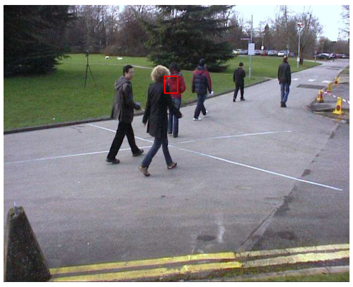
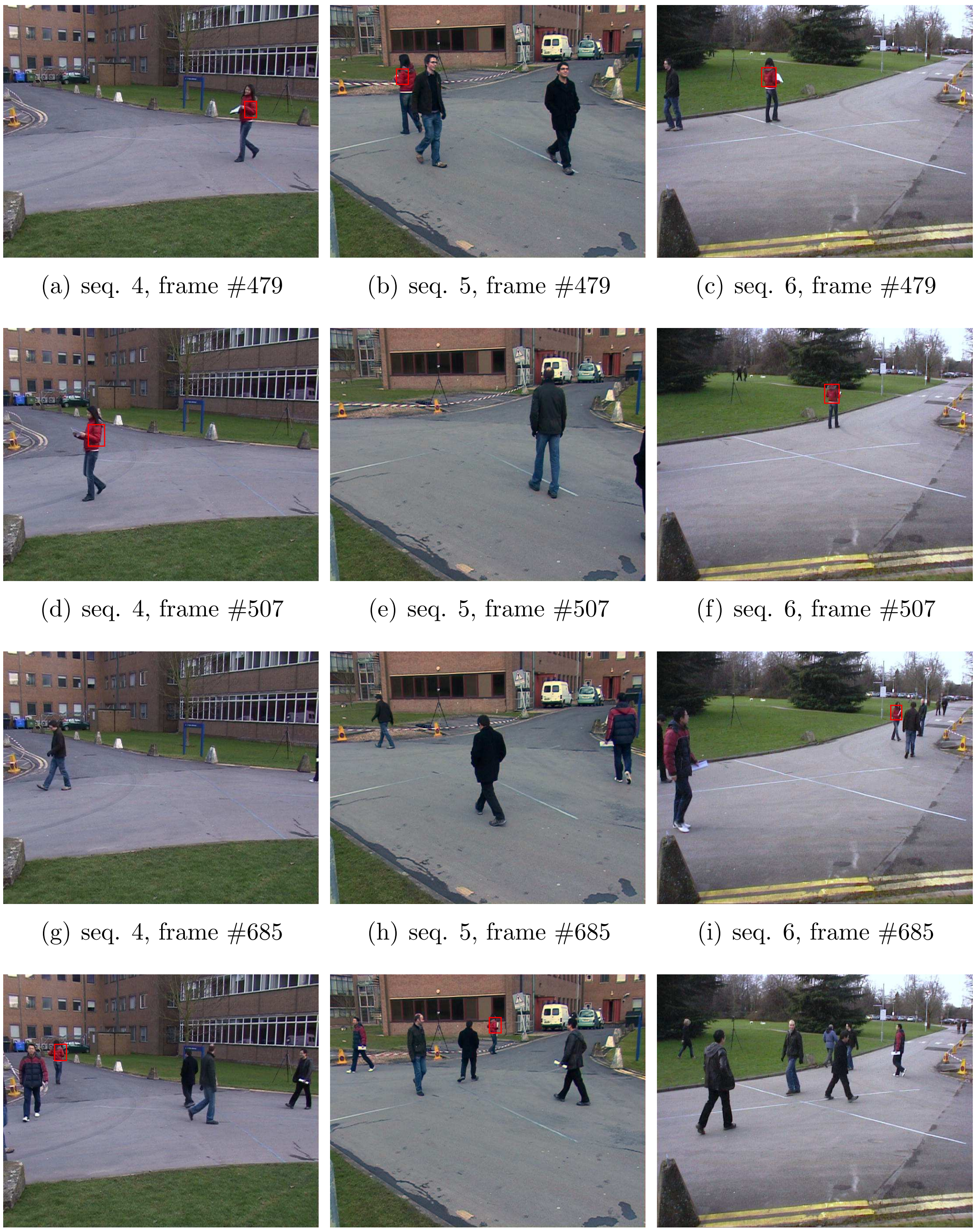

(l) seq. 3, frame #98

Fig. 7.10: Synchronised images from PETS2009 sequences 1, 2 and 3 with correct target highlighted.

in red jacket manually selected in frame #72 of sequence 2, as illustrated in Fig. 7.9, and it is used to compute the target model **Q**. This model is then tracked in all six mentioned sequences as it moves across camera views, going out of the FOVs and experiencing complete occlusion by other pedestrians in the scene. Of these six video sequences, the first three (1, 2 and 3) are synchronised with each



(a) seq. 4, frame #479 (b) seq. 5, frame #479 (c) seq. 6, frame #479

(d) seq. 4, frame #507 (e) seq. 5, frame #507 (f) seq. 6, frame #507

(g) seq. 4, frame #685 (h) seq. 5, frame #685 (i) seq. 6, frame #685

(j) seq. 4, frame #716 (k) seq. 5, frame #716 (l) seq. 6, frame #716

Fig. 7.11: Synchronised images from PETS2009 sequences 4, 5 and 6 with correct target highlighted.

other, which means that the n^{th} frame in each sequence refers to the same instant in time and it has a counterpart in the other two sequences. The same is true for sequences 4, 5 and 6. This can be seen in Fig. 7.10 and 7.11, where four synchronised frames from sequences 1, 2 and 3 are reported in Fig. 7.10, while four synchronised frames from sequences 4, 5 and 6 are reported in Fig. 7.11. It

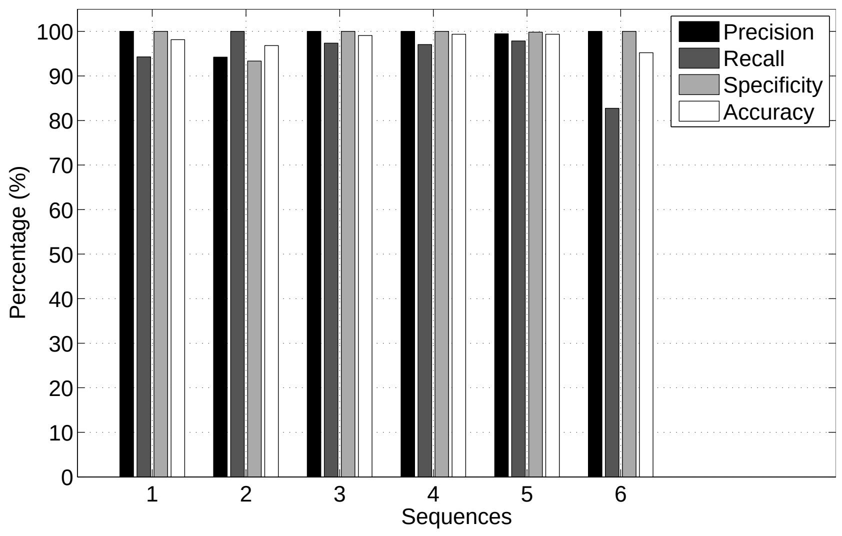
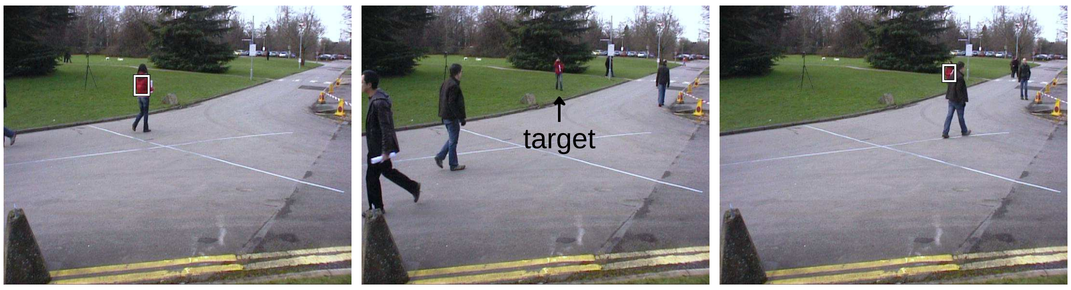


Fig. 7.12: Precision, recall, specificity and accuracy for PETS2009 sequences.

can be appreciated from Fig. 7.11 how the target may be not visible at times in some of the views (Fig. 7.11(e)), while it can still be tracked at the same time instant in other views (Fig. 7.11(d) and 7.11(f)). This aspect is fundamental for selecting at any time the best view of a target, as reported in Section 7.7.2. Values of precision, recall, specificity and accuracy for the six sequences are illustrated in Fig. 7.12. It can be seen that the system has very high precision and specificity also in an outdoor setup, with only 4 FP erroneously detected in all six sequences. In general also recall and accuracy are high ($> 90\%$), apart from sequence 6, where the lower value of recall (83%) is due to the fact that in this sequence the target walks aways from the camera and its size decreases to less than half of its initial one, as can be seen in the centre image in Fig. 7.13. In this case, the histogram of the target changes significantly, resulting in a poor match and therefore a high number of FN. Nonetheless, when the target moves back closer to the camera, the tracker successfully locates the target. Regarding camera colour calibration, from Fig. 7.9 it is noticed that view 6 (sequences 2 and 5) in PETS2009 dataset is slightly blue-ish. Nonetheless, by applying the Gray World Assumption, one is able to normalise all the sequences, as it can be seen in Fig. 7.10 and 7.11, so that the colour model $\mathbf{Q}$ is valid for all the views, as demonstrated by this experiment.

7.7.2 Best view selection and occlusion handling

In the proposed system, the DS receives target information, i.e. distance $d(\cdot)$ and position within the frame, from each TA associated with a different camera

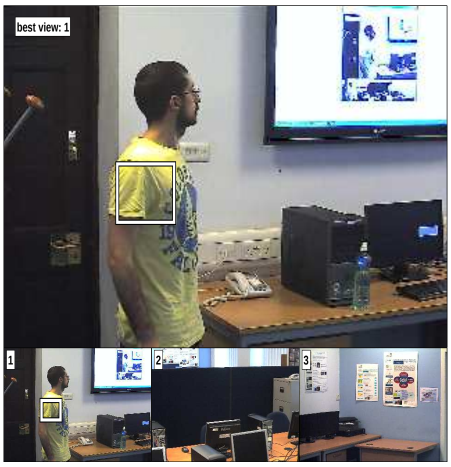


(a) frame #481 (b) frame #599 (c) frame #624

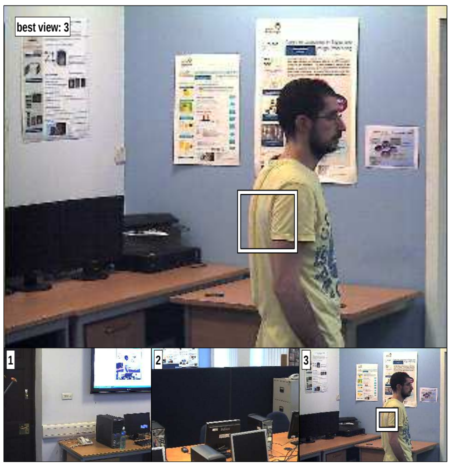
Fig. 7.13: Target moving towards the far field, in PETS2009 S2-L1-12.34-07 (sequence 6).

and, from this data, it can select the best view of the target. Therefore, it can produce a continuous video stream from multiple views, when the selected target appears in the different FOVs of non-overlapping cameras, as illustrated in Fig. 7.14, where six snapshots of the DS video output are shown. The current system implementation uses three IP cameras and, for testing purposes only, the DS shows the best view of the target, at the top of its video output, and also the current feeds from the three cameras, at the bottom. In this video the target is the person in yellow moving from one side of an indoor environment to the other, and back. In the six sets of camera feed subimages in Fig. 7.14, it can be seen that the target is sequentially detected in camera 1 (Fig. 7.14(a)), camera 3 (Fig. 7.14(b)), camera 2 (Fig. 7.14(c) and 7.14(d)), and then again in camera 3 (Fig. 7.14(e)), and finally in camera 1 (Fig. 7.14(f)). The DS is able to switch between camera FOVs to select always the best view of the target and produce a continuous video of it. Such a video is represented by the top subimages in Fig. 7.14.

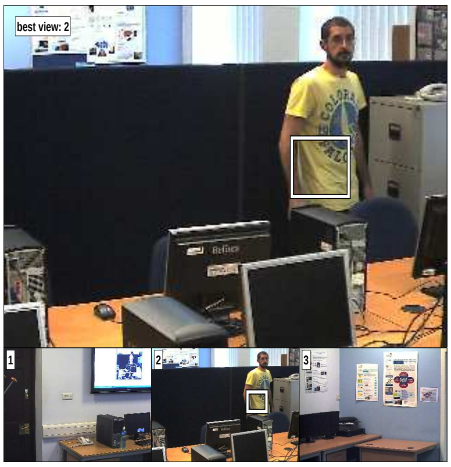
Moreover, the DS can use the tracking results from the TAs associated with overlapping cameras, to select at any time the view with the highest TA match λ_n^ξ. Therefore the DS is able to easily solve the occlusion problem in overlapping cameras, when the target is occluded in one of the views. As illustrated in Fig. 7.15, the target is occluded in view 3, partially occluded in view 2 and completely visible in view 1. The target is correctly tracked in views 1 ($\lambda_1^\xi = 76.4\%$) and 2 ($\lambda_2^\xi = 44.5\%$). As view 1 gives a higher match between target histogram $\mathbf{P}$ and target model $\mathbf{Q}$, it is selected as best view among the three available ones, and occlusion is implicitly solved.



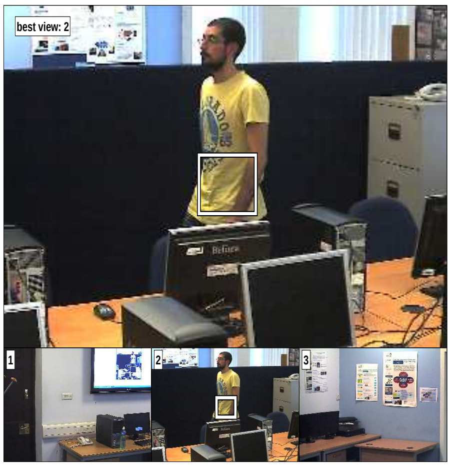
(a)



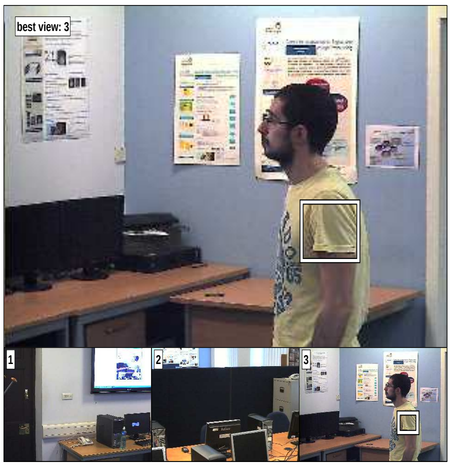
(b)



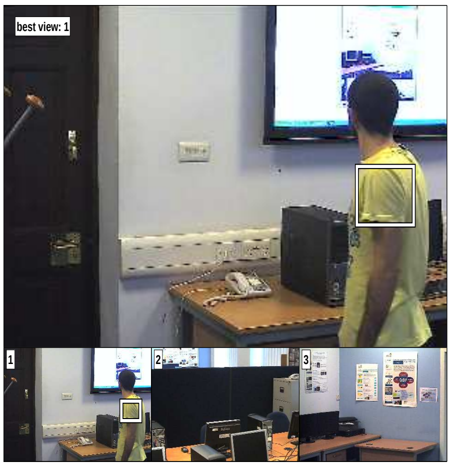
(c)



(d)



(e)



(f)

Fig. 7.14: Six images from the DS: the system is able to switch between camera views wherein the target is detected, to create a continuous video stream of the tracked target.

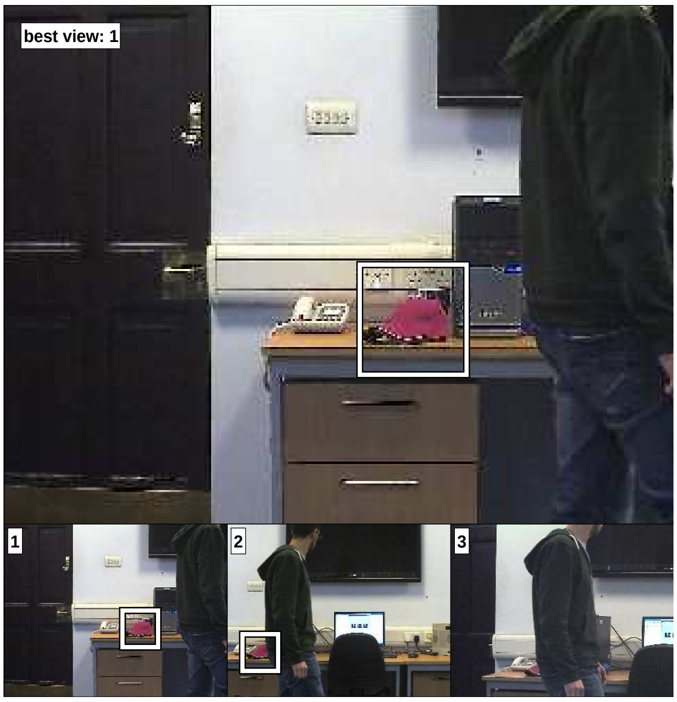


Fig. 7.15: DS with best view selection: the target is occluded in view 3; partially occluded in view 2; completely visible in view 1.

7.8 System scalability

Although only three cameras have been used in the reported experiments, the system allows scalability to a larger number of sensors. Concerning the DAs and TAs, they run independently from each other, so they can be easily replicated, to accommodate more cameras in the system. On the other hand, the SS and DS would have to handle a higher number of connections. Nonetheless, the data exchanged among DAs, SS and TAs is of few hundred bytes and with no strict time constraint on it. Similarly, the tracking data transmitted from the TAs to the DS is less than a hundred bytes, so a multi-threading implementation of the DS would easily handle connections from a large number of TAs. In case of a very large surveillance network, it is possible to organise the TAs in clusters, with a DS and a SS associated to each. To ensure communication among clusters, a two-tier hierarchy of DSs and SSs could be introduced.

7.9 Conclusion

In this chapter a decentralised multi-camera system for collaborative tracking has been presented. The system architecture is highly flexible and synchronisation constraints on software and hardware can be relaxed. The tracking algorithm adopted is based on the improved MS described in Chapter 5, where colour is the only information used by the tracking agents to find the selected target. Moreover, the SWAD-based tracker from Chapter 4 is used to stabilise tracking detections from the improved MS, over consecutive frames. No positional information is used to localise the target in multiple views, while only the knowledge of occurrence of overlapping between views is required to initialise the trackers. As a consequence, the system setup phase is simplified. Nonetheless, experimental results demonstrate the effectiveness of the presented system in both indoor and outdoor environments, in tracking a colour target over multiple overlapping and non-overlapping views. The system is ideal in situations where video synchronisation and detailed knowledge of camera setup are neither required nor available.

Chapter 8

Conclusions and future work

8.1 Discussion

This thesis has presented novel video analytics algorithms for target tracking in single and multi-camera setup, and for automatic detection of abandoned and removed objects.

Starting from the well-known Sum of Absolute Differences (SAD), a novel adaptive template matching algorithm for robust target tracking based on the minimisation of a Sum of Weighted Absolute Differences (SWAD) has been presented. This algorithm uses a Gaussian weighting kernel to assign lower weights to peripheral pixels in the target template, in order to reduce the effects of partial occlusion. Moreover target changes are incorporated in the template using an IIR filter. A novel technique to detect complete occlusion and recover from it has also been presented in the context of the SWAD-based tracker. Experimental results have shown the robustness of the proposed tracker and its higher accuracy with respect to SAD-based tracking. A DSP embedded implementation of this algorithm has been described, to demonstrate that it can successfully exploit devices with low computational capabilities.

To overcome issues present in the conventional MS tracker, such as fast target loss and failure due to complete occlusion, a novel improved MS tracker with failure recovery strategy has been presented. Experimental results show that the proposed algorithm can effectively recover the target after complete occlusion and therefore track it also in complex scenarios, as for example crowd scenes.

As methods proposed in the literature are not designed to be able to trigger alerts within a time interval defined by the user, after new abandoned and removed object events have occurred, a novel algorithm for abandoned and removed object detection in real-time has been presented. The user can specify a detection time interval, while the background is updated only after new events are detected. Algorithm parameters are adaptive with respect to the frame processing rate, so that even on different machines the detection time is generally the same. Also a novel technique based on the gradient projections in the object boundary pixels has been described, to correctly classify detected static objects as either abandoned or removed. The overall novel algorithm is included in the context of a reactive smart surveillance system, which notifies the occurrence of such events of interest to registered users, within seconds, through SMS alerts.

Finally a decentralised multi-camera system for collaborative tracking has been presented. The system architecture is highly flexible and synchronisation constraints on software and hardware can be relaxed. The tracking algorithm for single view is based on the novel improved MS, with colour being the only information used to track the selected target. No positional information is used for hand-off between cameras. Experimental results demonstrate the effectiveness of the presented system in both indoor and outdoor environments, in tracking a colour target over multiple overlapping and non-overlapping views. The system is ideal in situations where video synchronisation and detailed knowledge of camera setup are neither required nor available.

8.2 Future work

There are possible extensions arising from the work in this thesis:

- In the adaptive tracker based on SWAD minimisation presented in Chapter 4, a resizing mechanism for the target template could be added. Such a feature would be beneficial in situations where the target moves very far away or closely to the camera and, as a consequence, its size within the FOV changes considerably.

- Concerning the improved mean shift tracker presented in Chapter 5, additional target features could be incorporated in the target model, to further discriminate the correct target from other objects with similar colour within the FOV. For example one could take into account the shape of the object or the texture within it. Alternatively one could infer a relatively simple structure to the target, for example a torso-legs partitioning in case of a pedestrian, with the overall histogram being a weighted combination of the histograms of each partition.

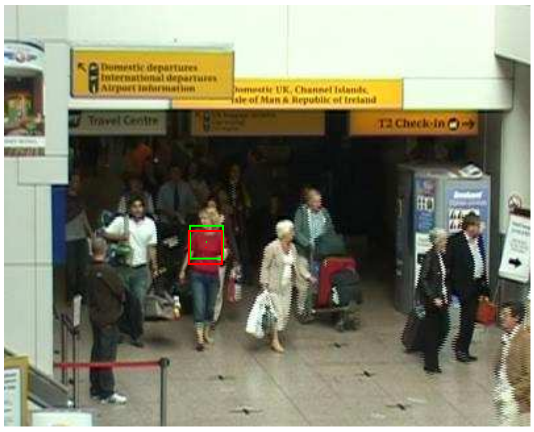
- A natural extension for the abandoned/removed object detection algorithm described in Chapter 6 would be to apply object analysis/recognition after an abandoned object alert has been triggered. On the same line, a person classifier could be incorporated, so that people being motionless in the FOV for a relatively long period of time would not be considered as abandoned objects.

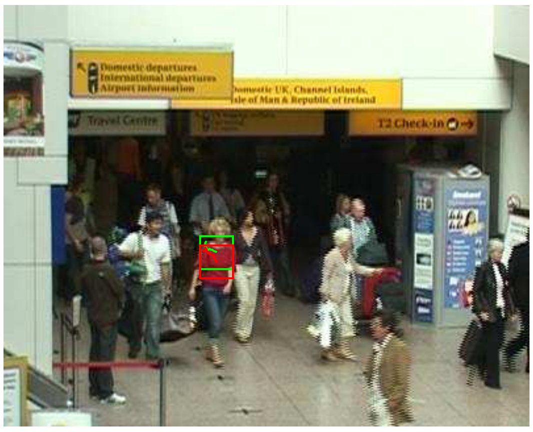
- Further extensions for the multi-camera tracking system described in Chapter 7 would be to incorporate additional target features, as in the case of the improved MS of Chapter 5, and to develop new strategies for the DS to collate the tracking detections from several TAs.

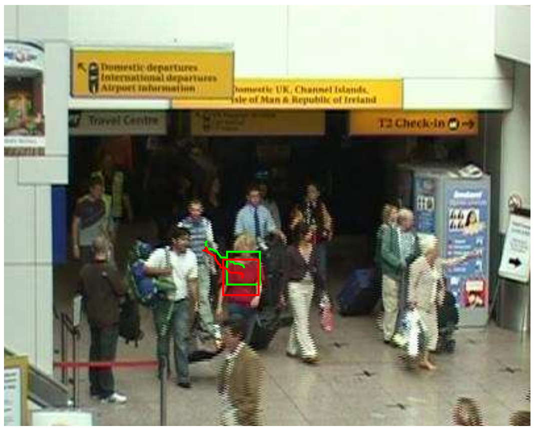
Appendix A

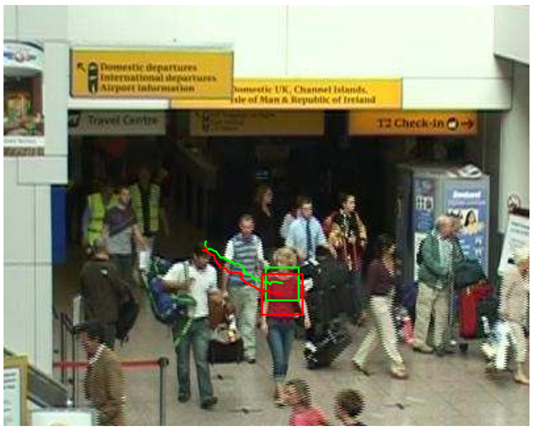
Visual results

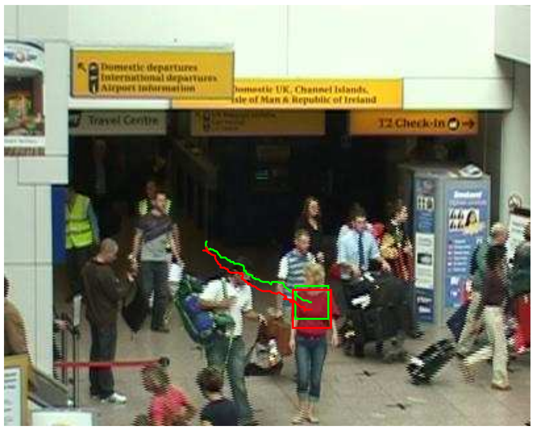
This appendix contains further visual results for the experiments reported in Chapter 4 and 5.


(a) frame #177


(b) frame #204


(c) frame #234


(d) frame #264


(e) frame #294

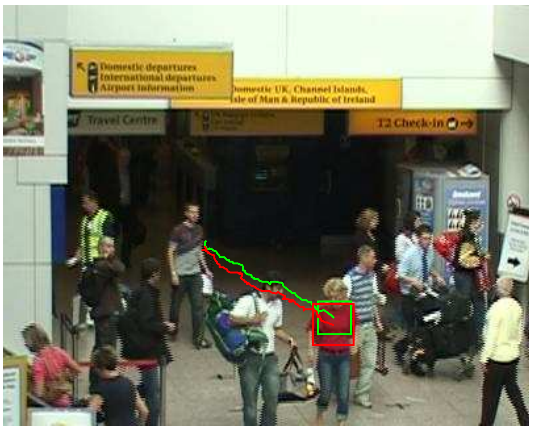
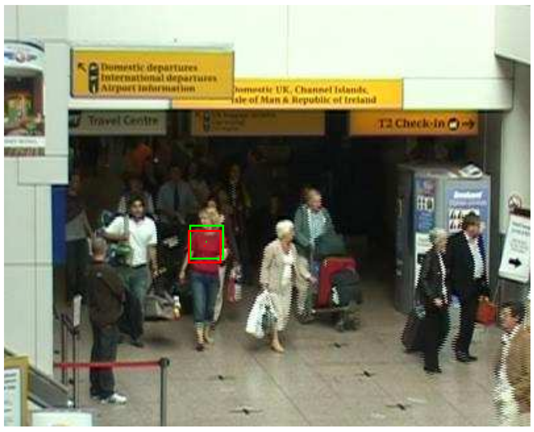
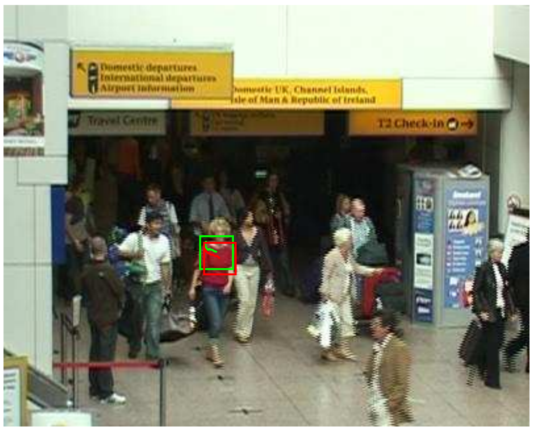
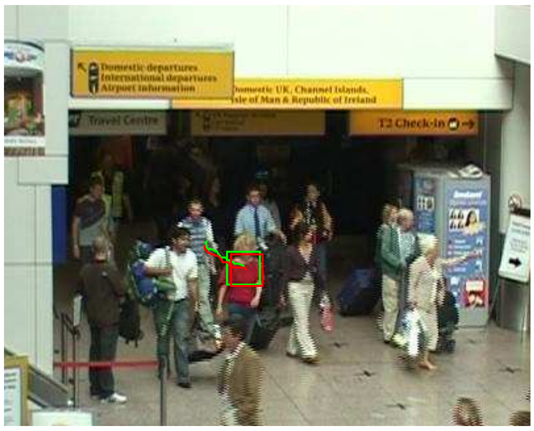
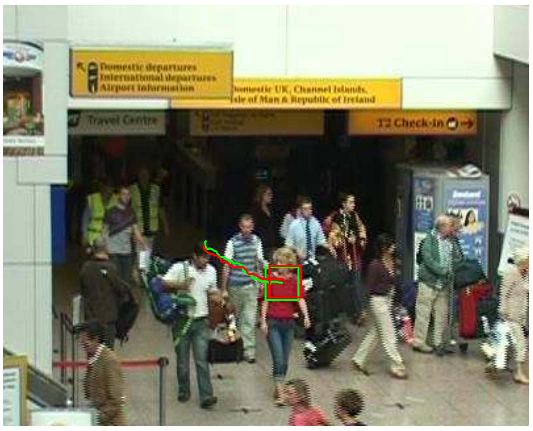

(f) frame #324

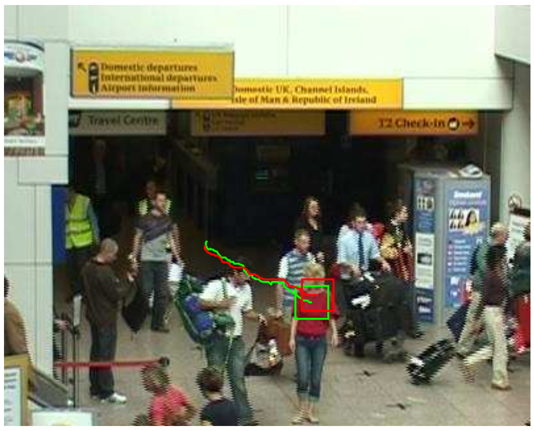
Fig. A.1: MS results for the S06_2/1 sequence from PETS2007 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #177


(b) frame #204


(c) frame #234


(d) frame #264


(e) frame #294

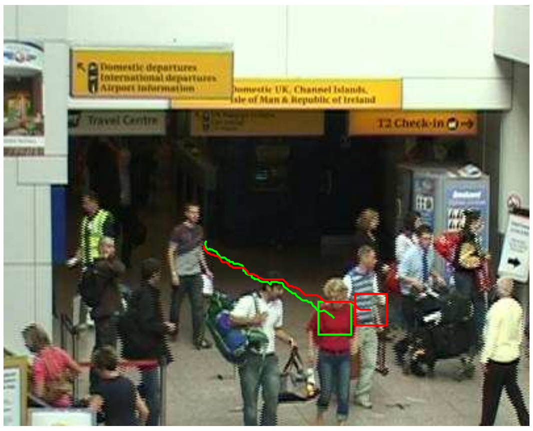
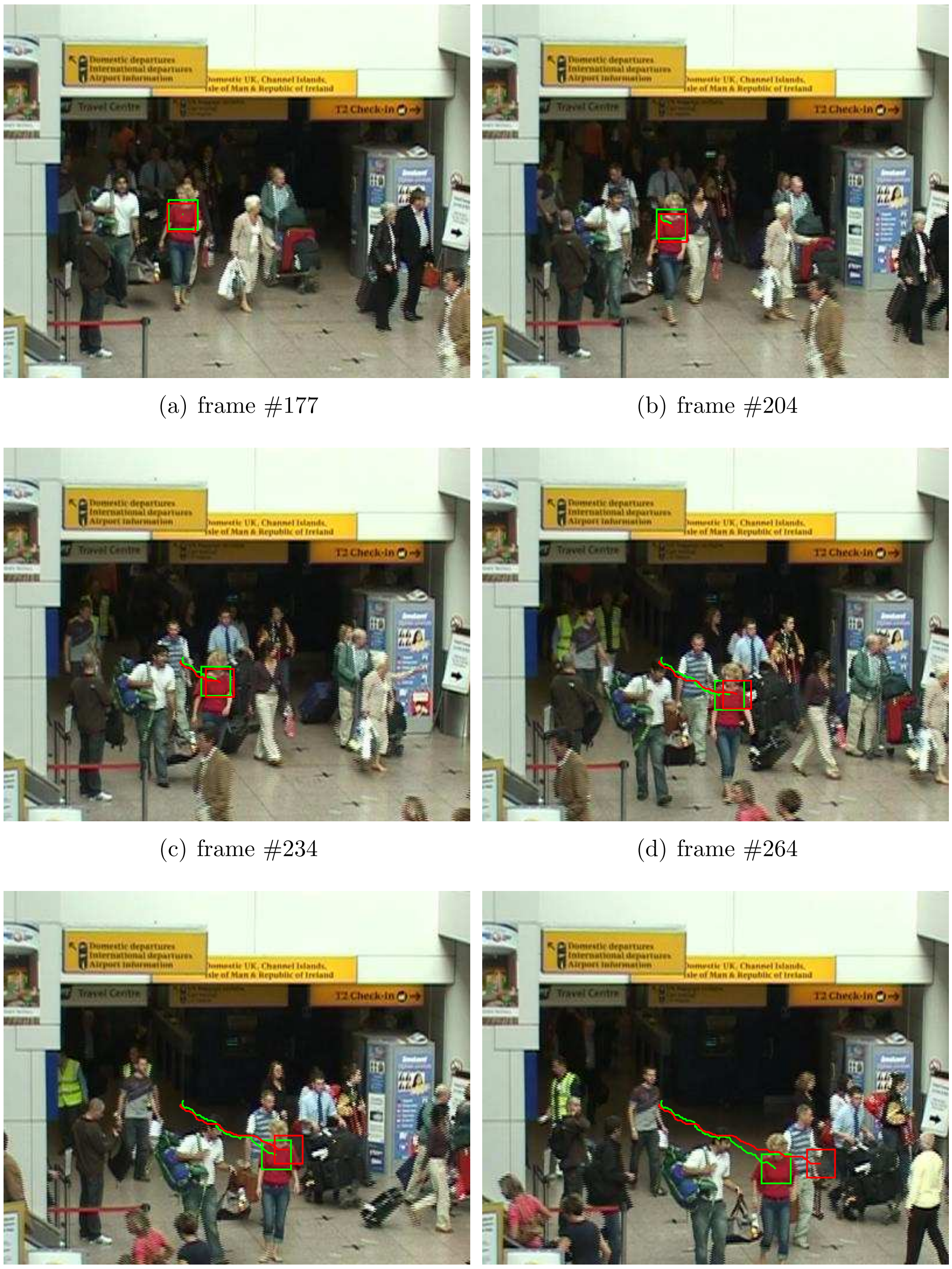

(f) frame #324

Fig. A.2: SAD results for the S06_2/1 sequence from PETS2007 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.



(a) frame #177

(b) frame #204

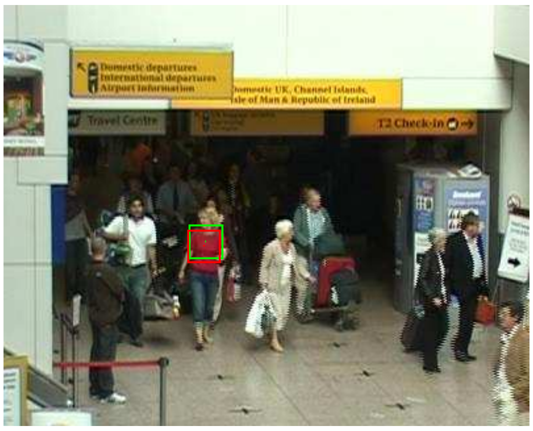
(c) frame #234

(d) frame #264

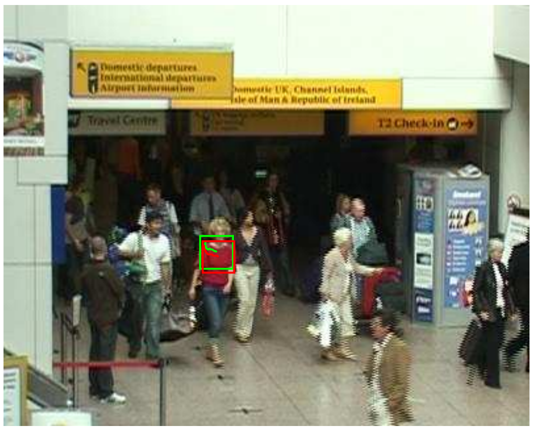
(e) frame #294

(f) frame #324

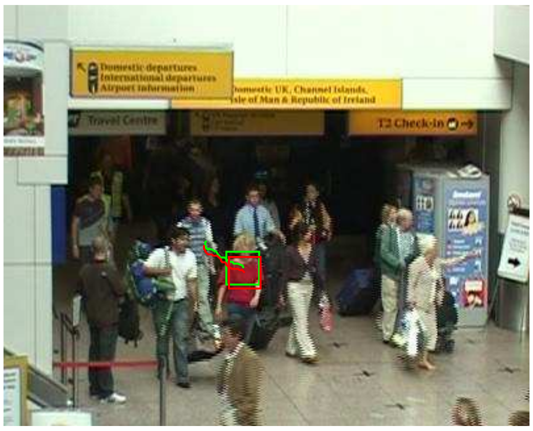
Fig. A.3: NCC results for the S06_2/1 sequence from PETS2007 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.



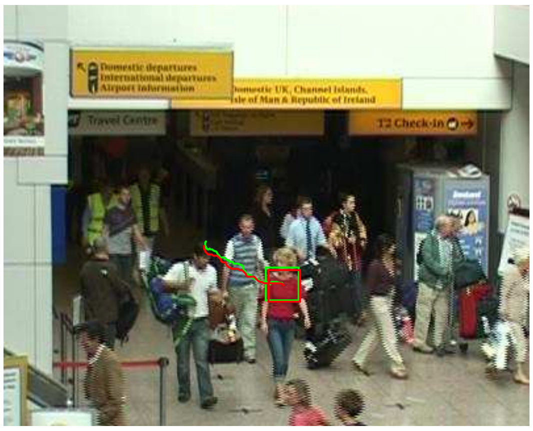
(a) frame #177



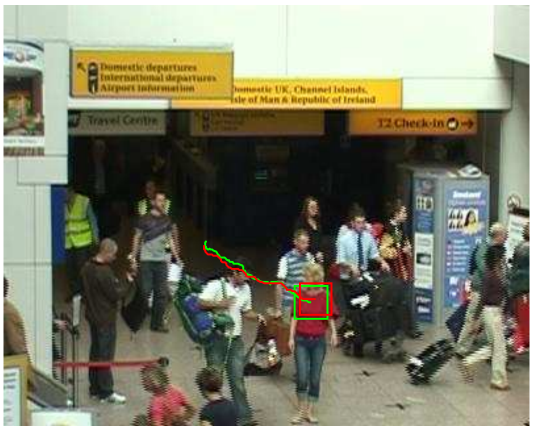
(b) frame #204



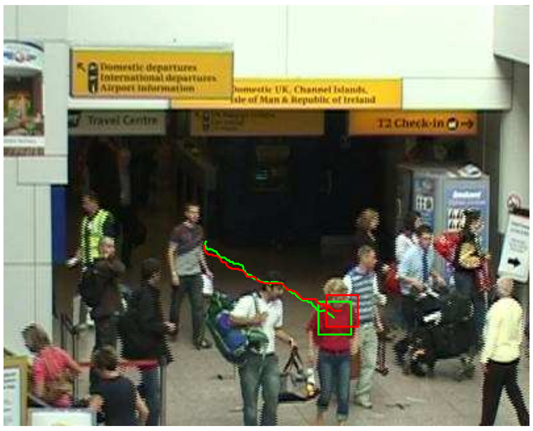
(c) frame #234



(d) frame #264

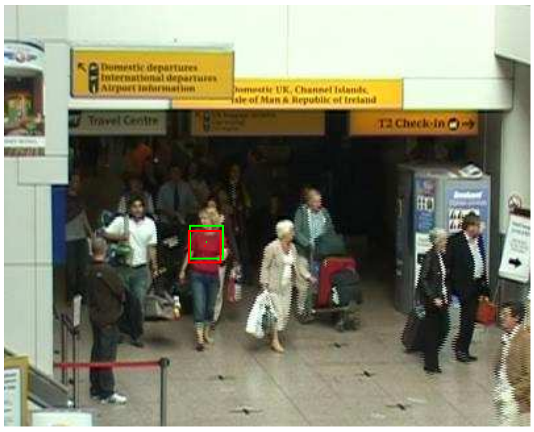


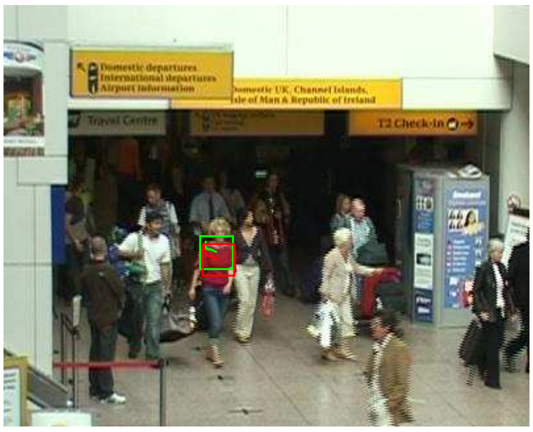
(e) frame #294

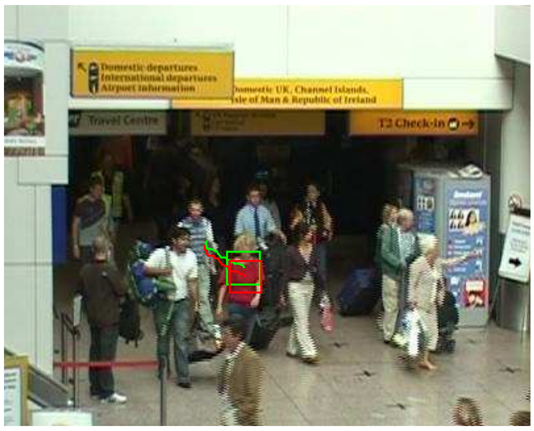


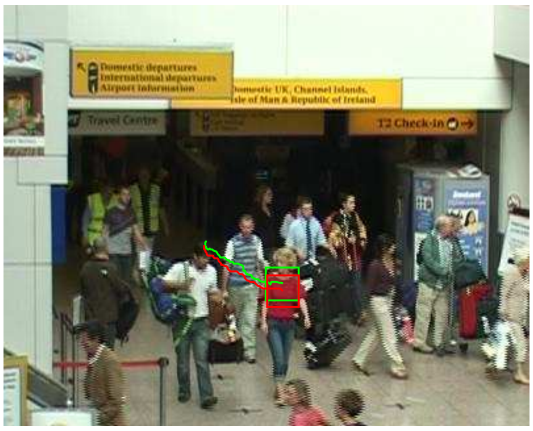
(f) frame #324

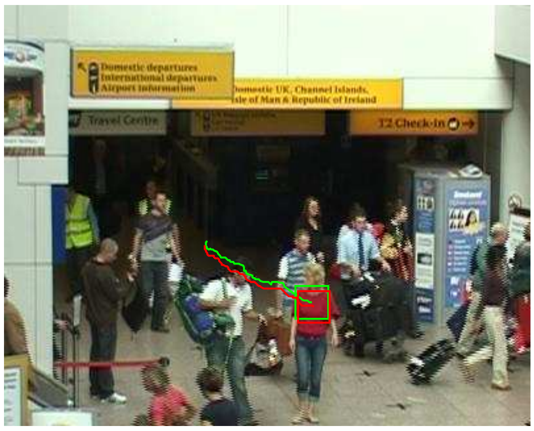
Fig. A.4: SWAD results for the S06_2/1 sequence from PETS2007 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #177


(b) frame #204


(c) frame #234


(d) frame #264


(e) frame #294

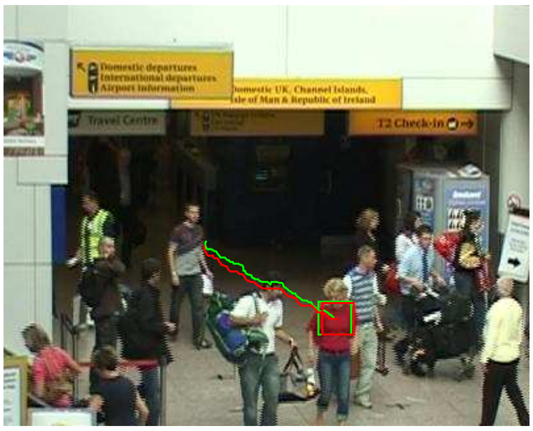
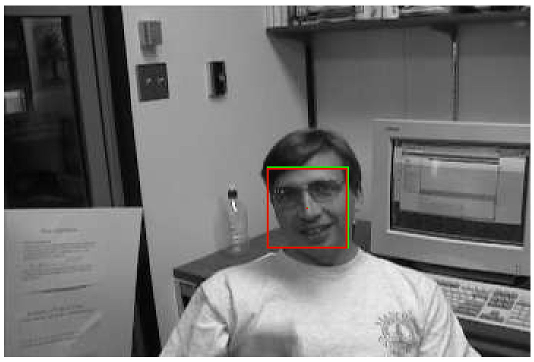
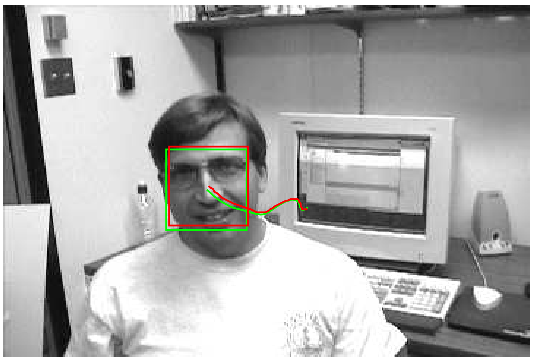
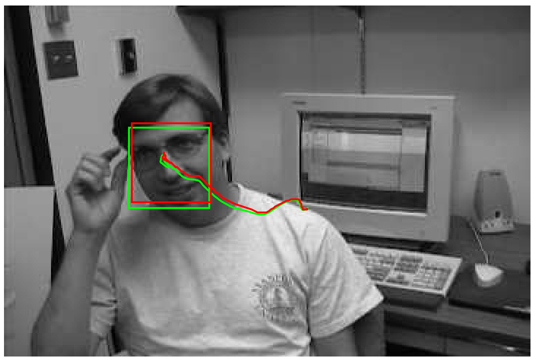
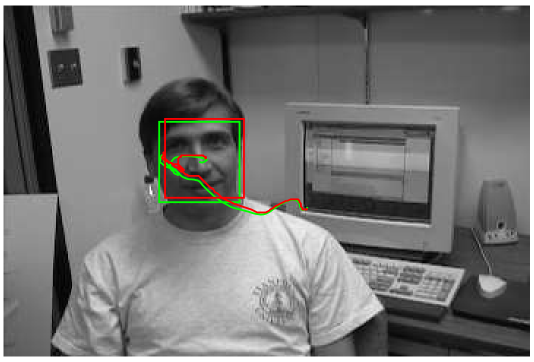

(f) frame #324

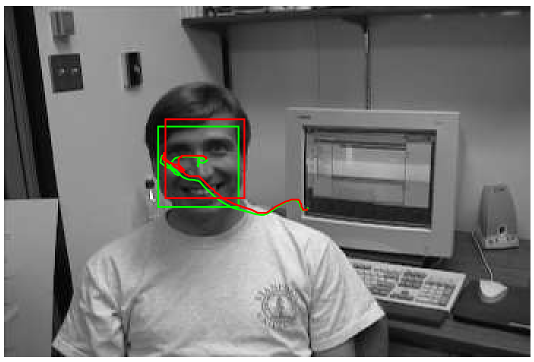
Fig. A.5: SWADOCC results for the S06_2/1 sequence from PETS2007 dataset. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #232


(b) frame #286


(c) frame #349


(d) frame #412


(e) frame #475

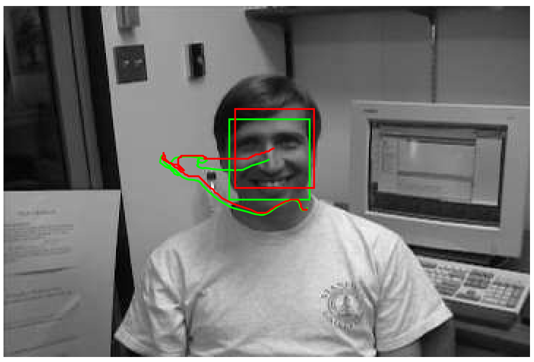
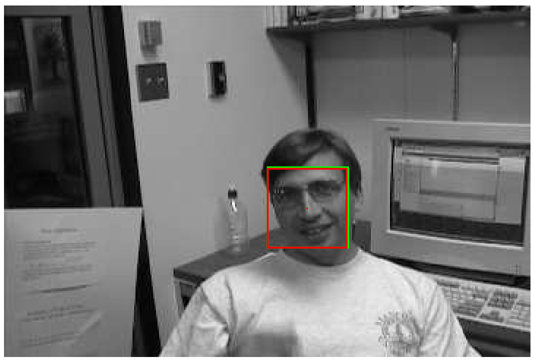
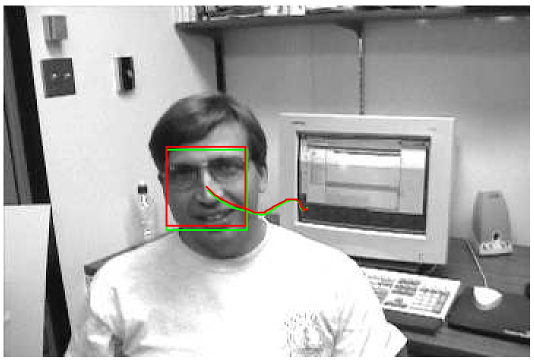
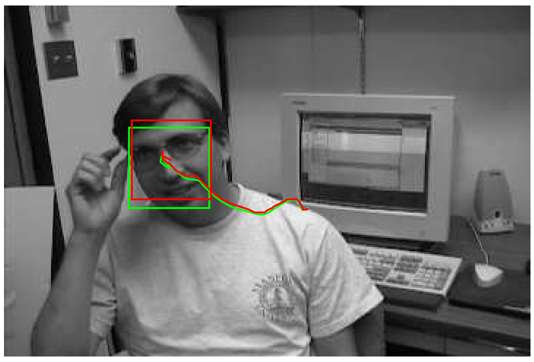
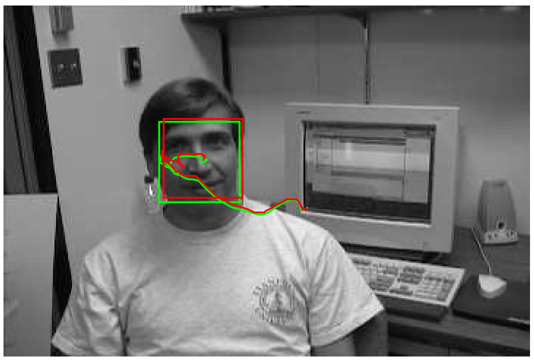

(f) frame #539

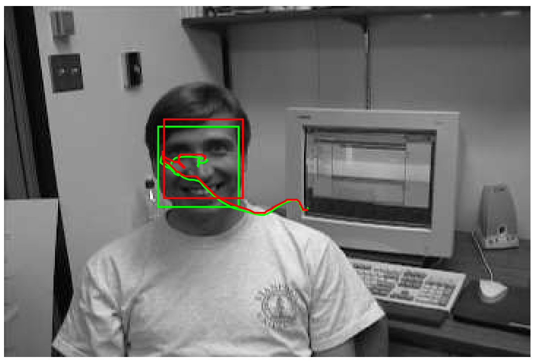
Fig. A.6: SAD results for the Dudek face sequence. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #232


(b) frame #286


(c) frame #349


(d) frame #412


(e) frame #475

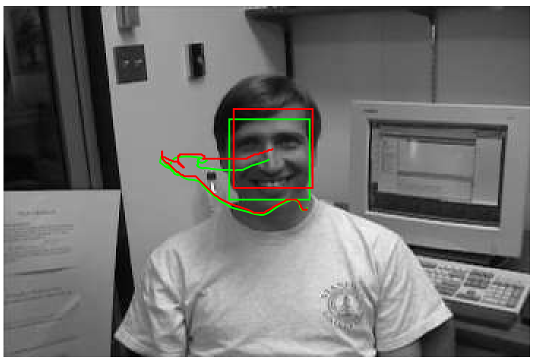
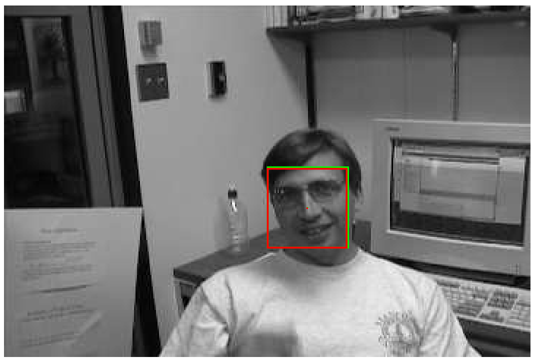
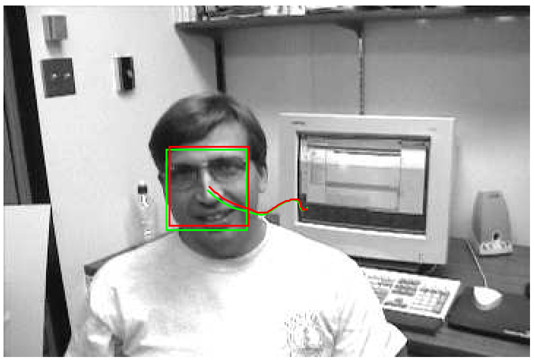
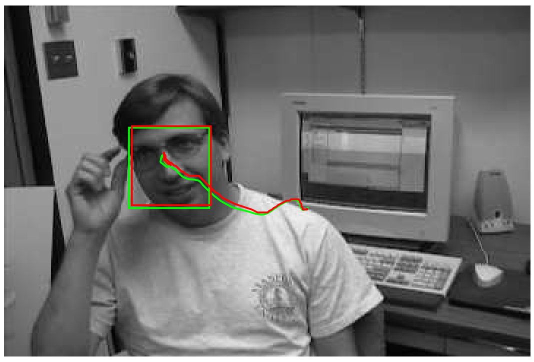
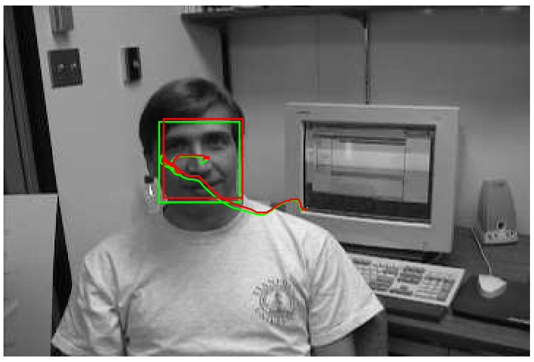

(f) frame #539

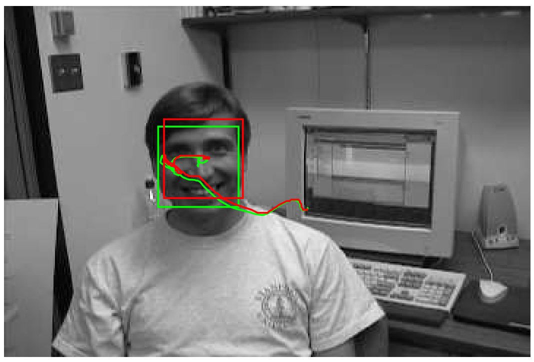
Fig. A.7: NCC results for the Dudek face sequence. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #232


(b) frame #286


(c) frame #349


(d) frame #412


(e) frame #475

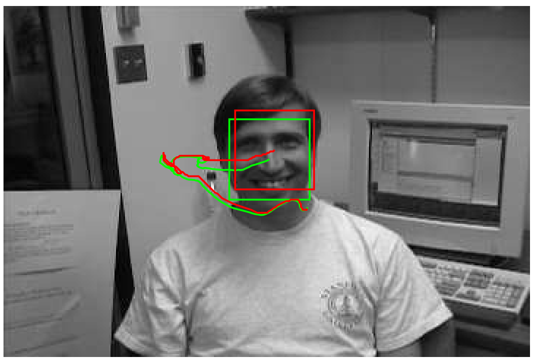
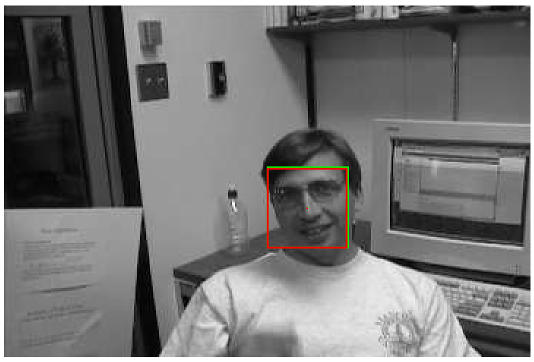
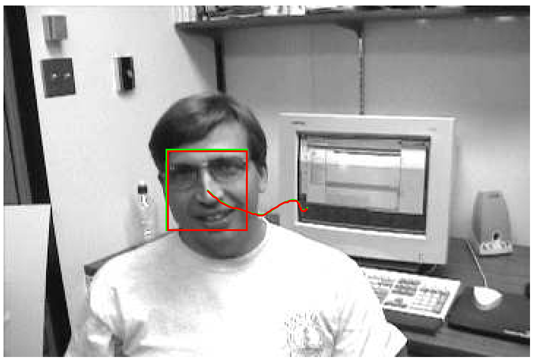
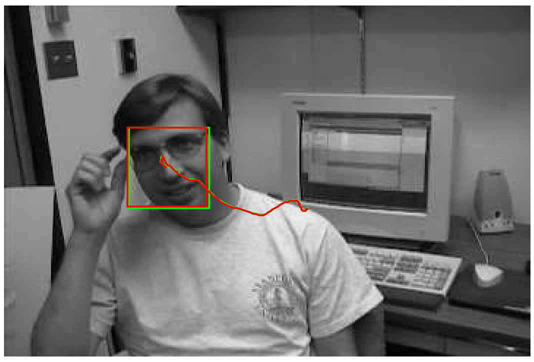
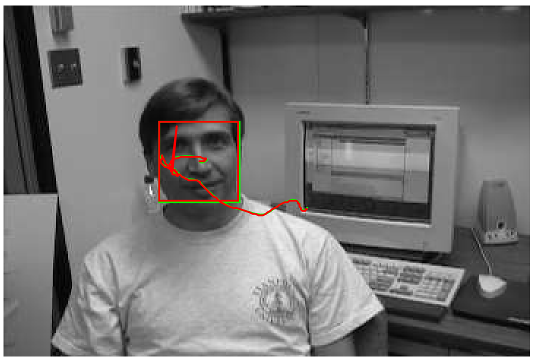

(f) frame #539

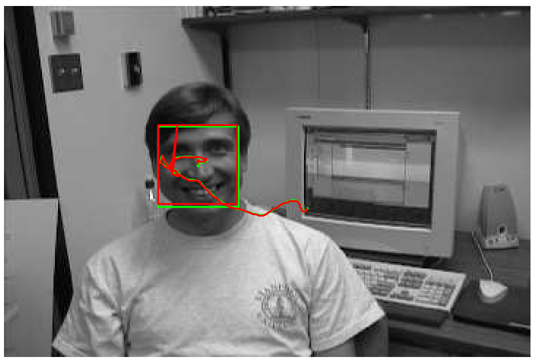
Fig. A.8: SWAD results for the Dudek face sequence. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #232


(b) frame #286


(c) frame #349


(d) frame #412


(e) frame #475

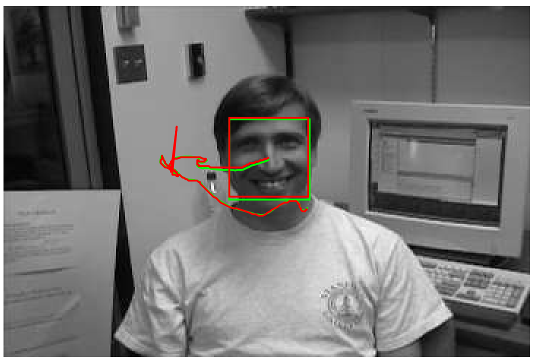
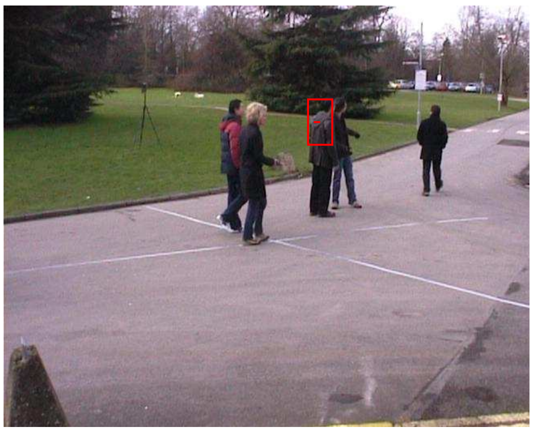
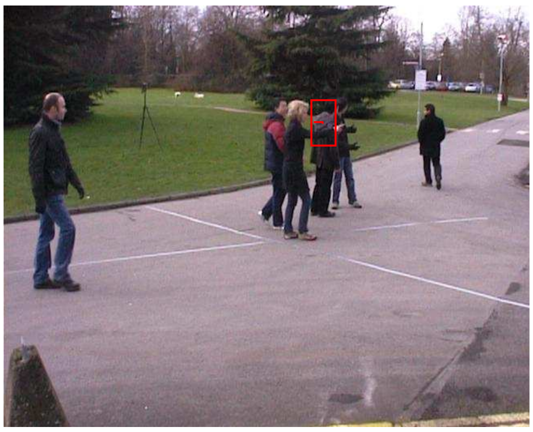
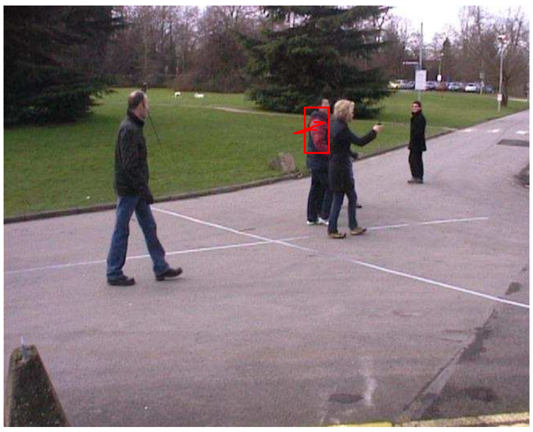
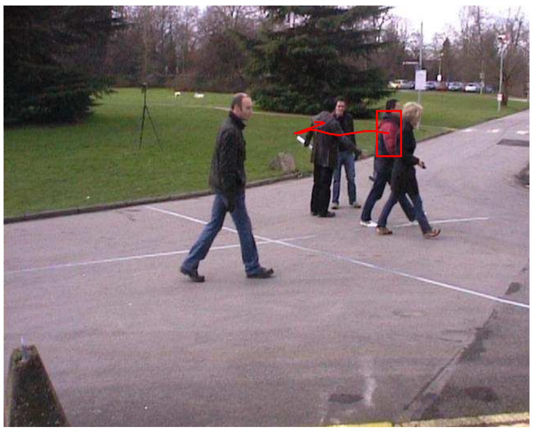

(f) frame #539

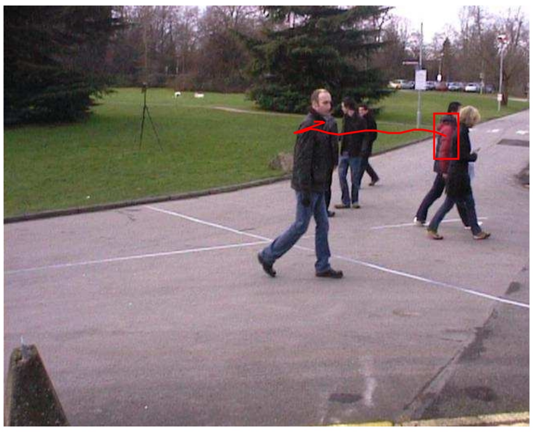
Fig. A.9: SWADOCC results for the Dudek face sequence. In each image, the green track is the ground truth, while the red one is the tracker result.


(a) frame #65


(b) frame #70


(c) frame #75


(d) frame #81


(e) frame #86

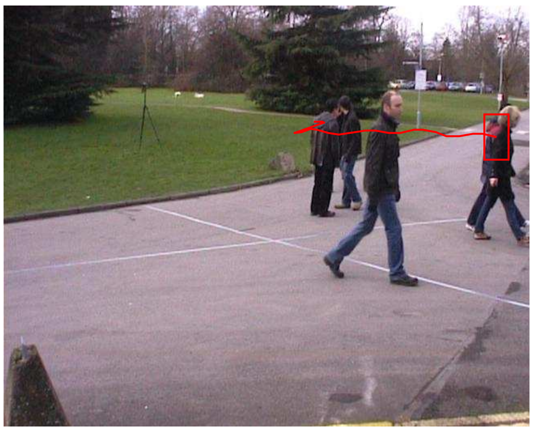
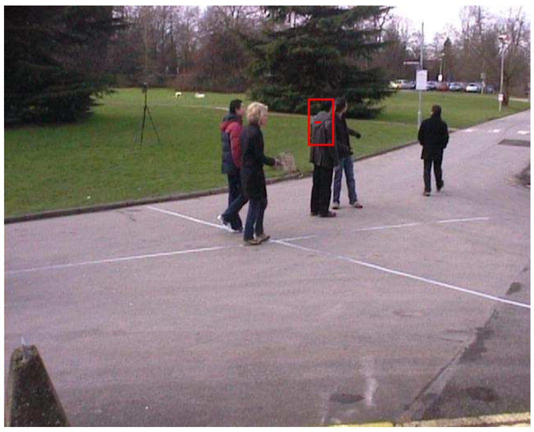
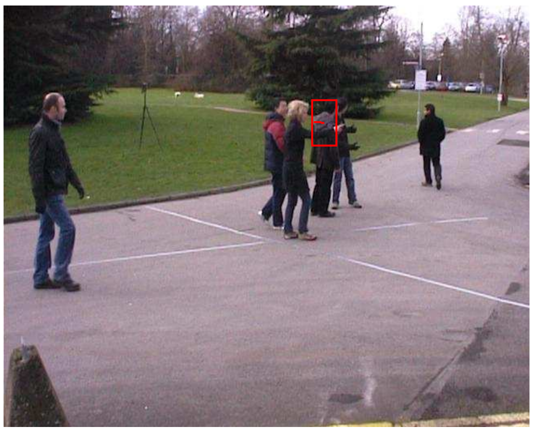
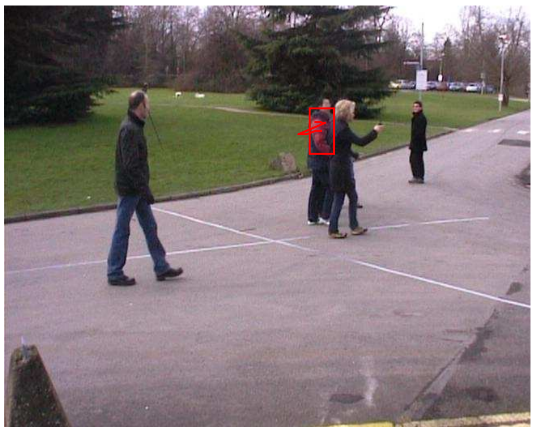
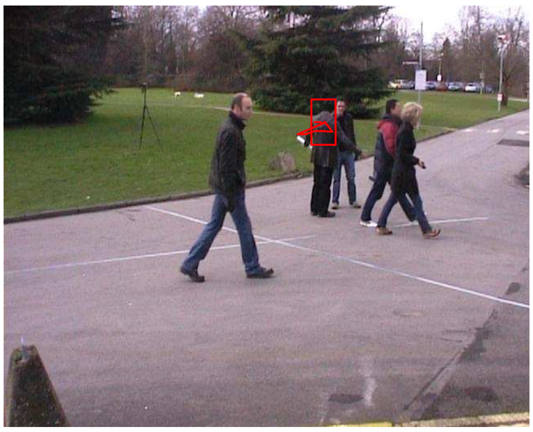

(f) frame #90

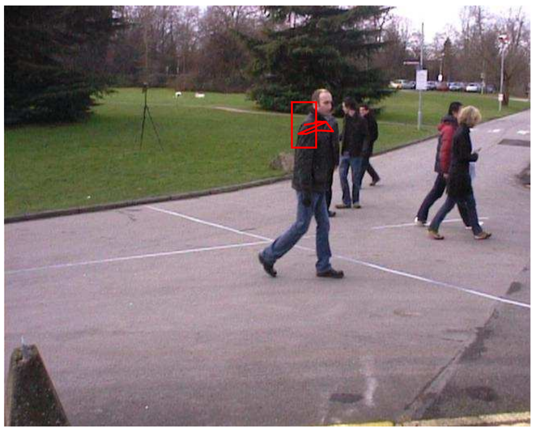
Fig. A.10: SWAD results for S2-L1-12.34-007 sequence from PETS2009 dataset.


(a) frame #65


(b) frame #70


(c) frame #75


(d) frame #81


(e) frame #86

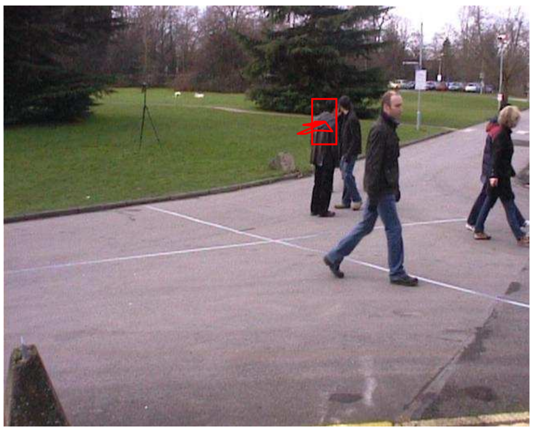

(f) frame #90

Fig. A.11: SWADOCC results for S2-L1-12.34-007 sequence from PETS2009 dataset.

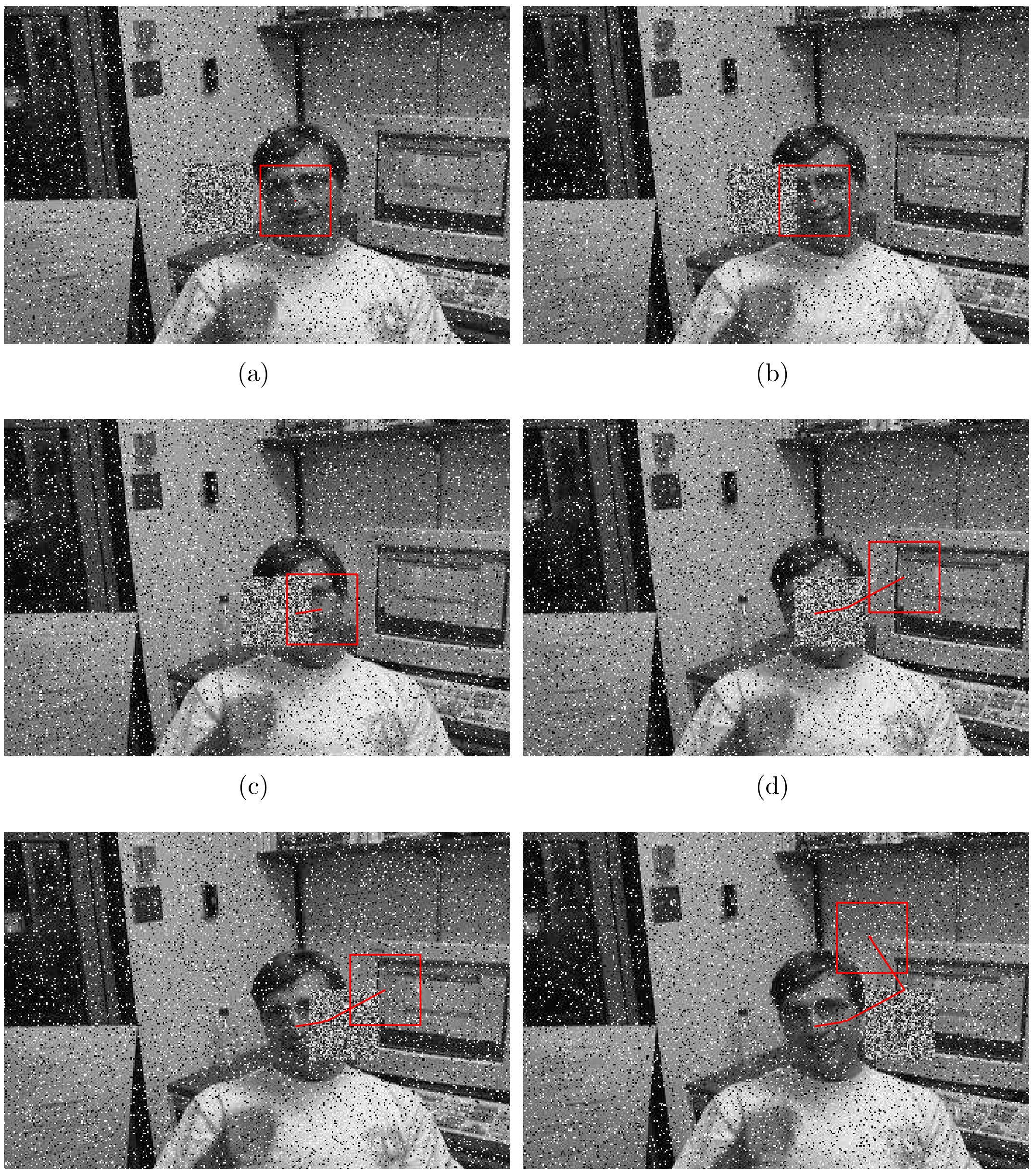
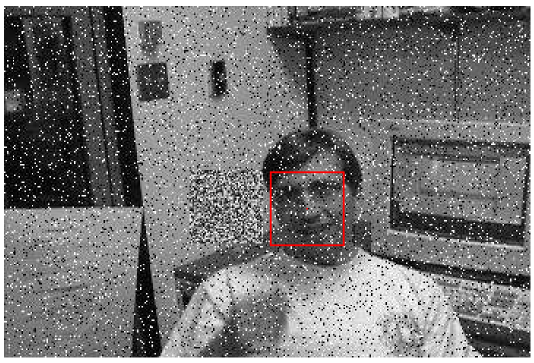
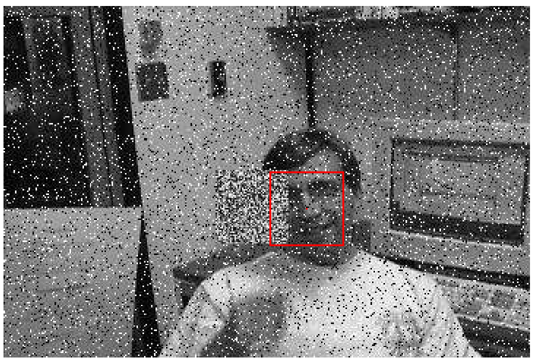
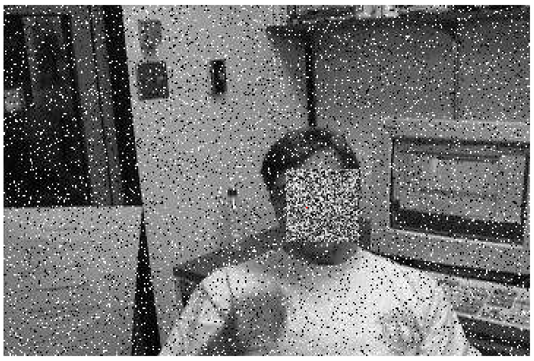

(a) (b) (c) (d) (e) (f)

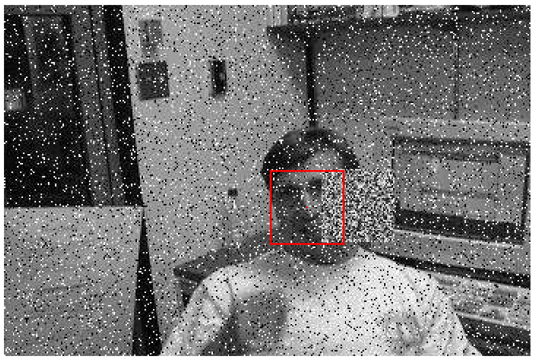
Fig. A.12: SWAD results with synthetic sequence.


(a)


(b)

(c)


(d)


(e)

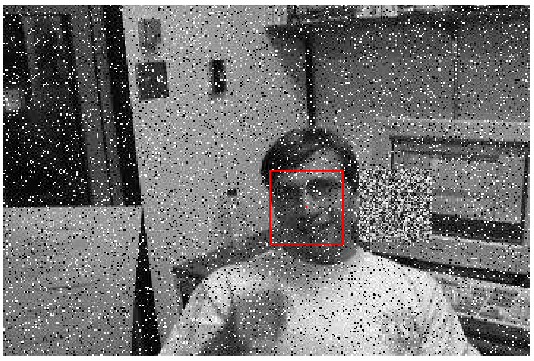
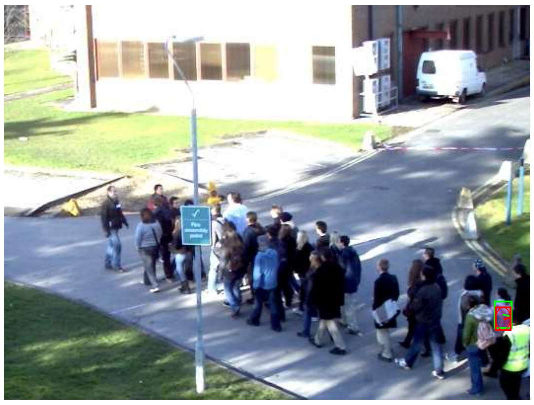
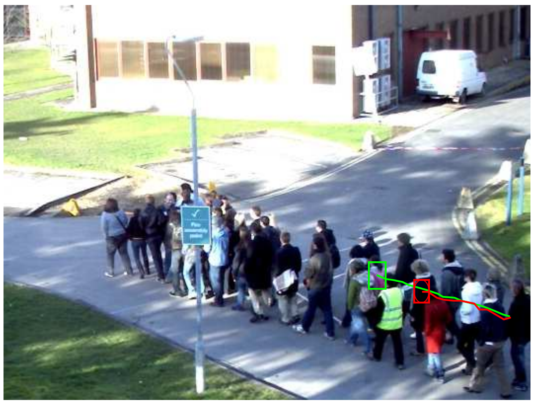
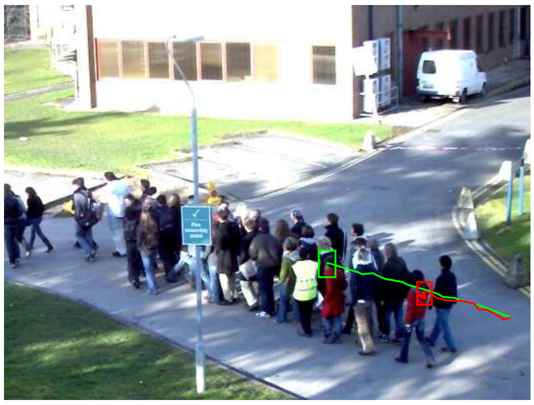
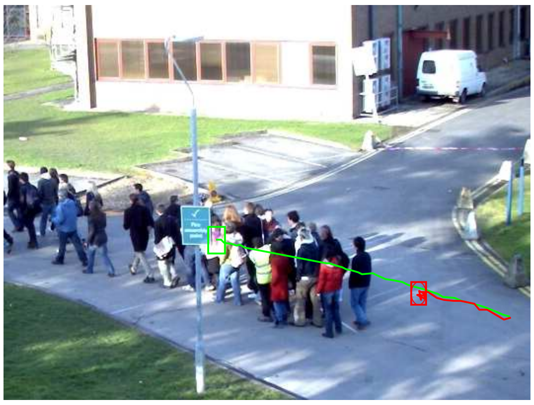

(f)

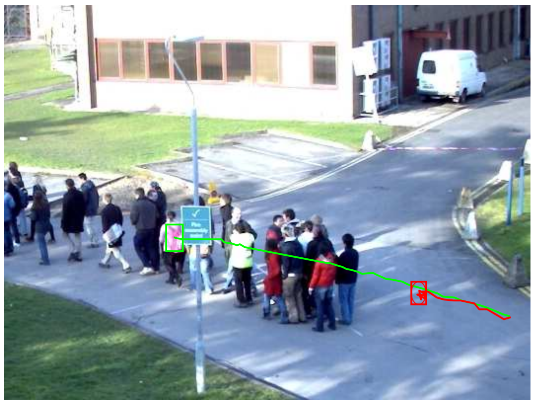
Fig. A.13: SWADOCC results with synthetic sequence.


(a) frame #22


(b) frame #38


(c) frame #56


(d) frame #85


(e) frame #94

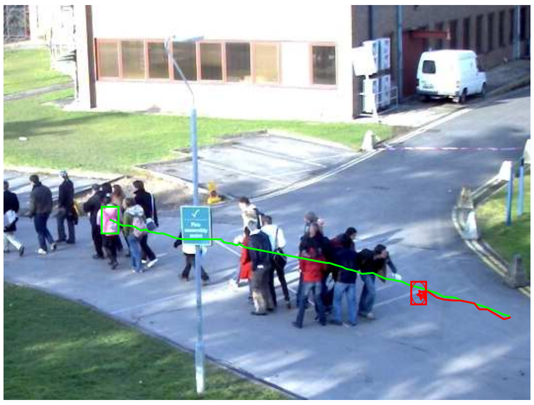
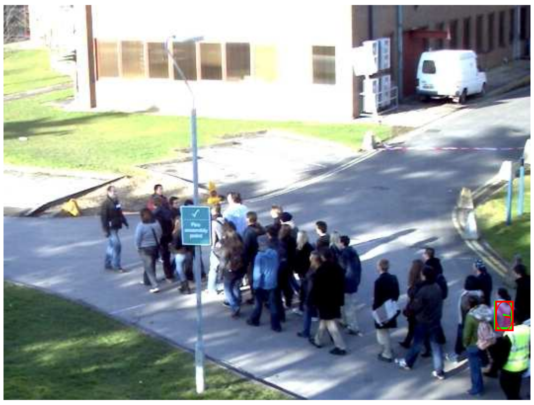
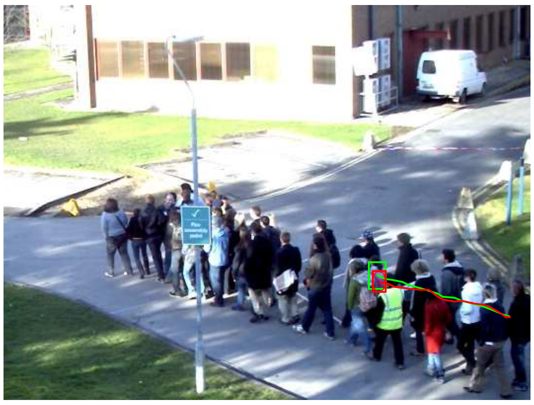

(f) frame #109

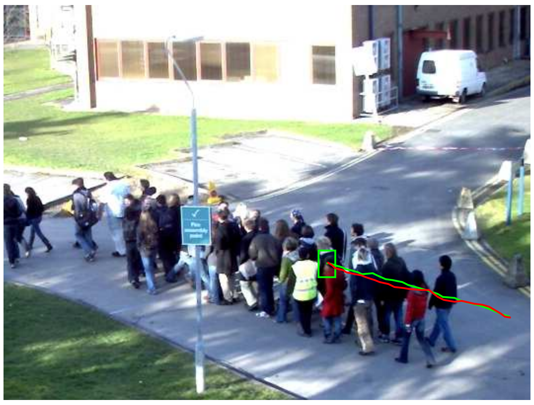
Fig. A.14: Conventional MS results for PETS2009 S3-Multiple_flow-time_14.46-001 sequence. In each image, the green track is the ground truth, while the red one is the tracker result.



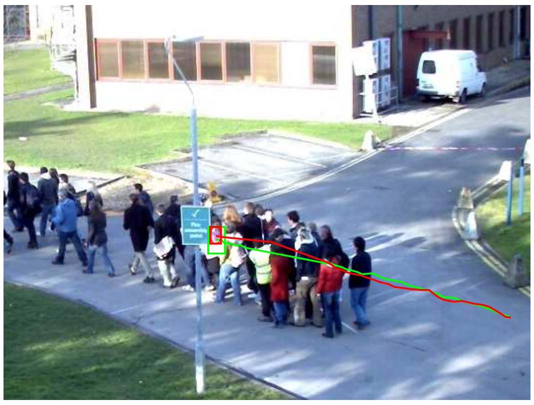
(a) frame #22



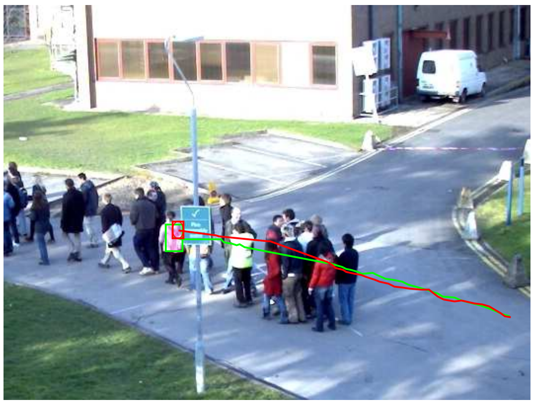
(b) frame #38



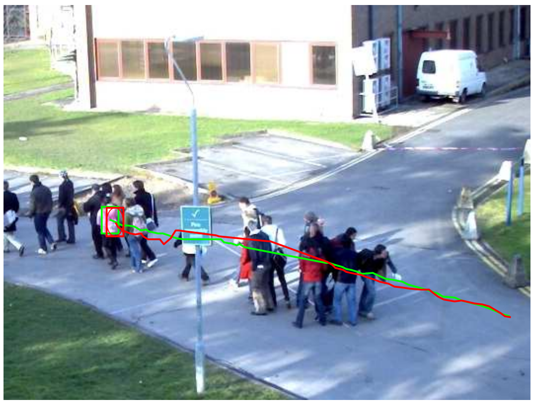
(c) frame #56



(d) frame #85

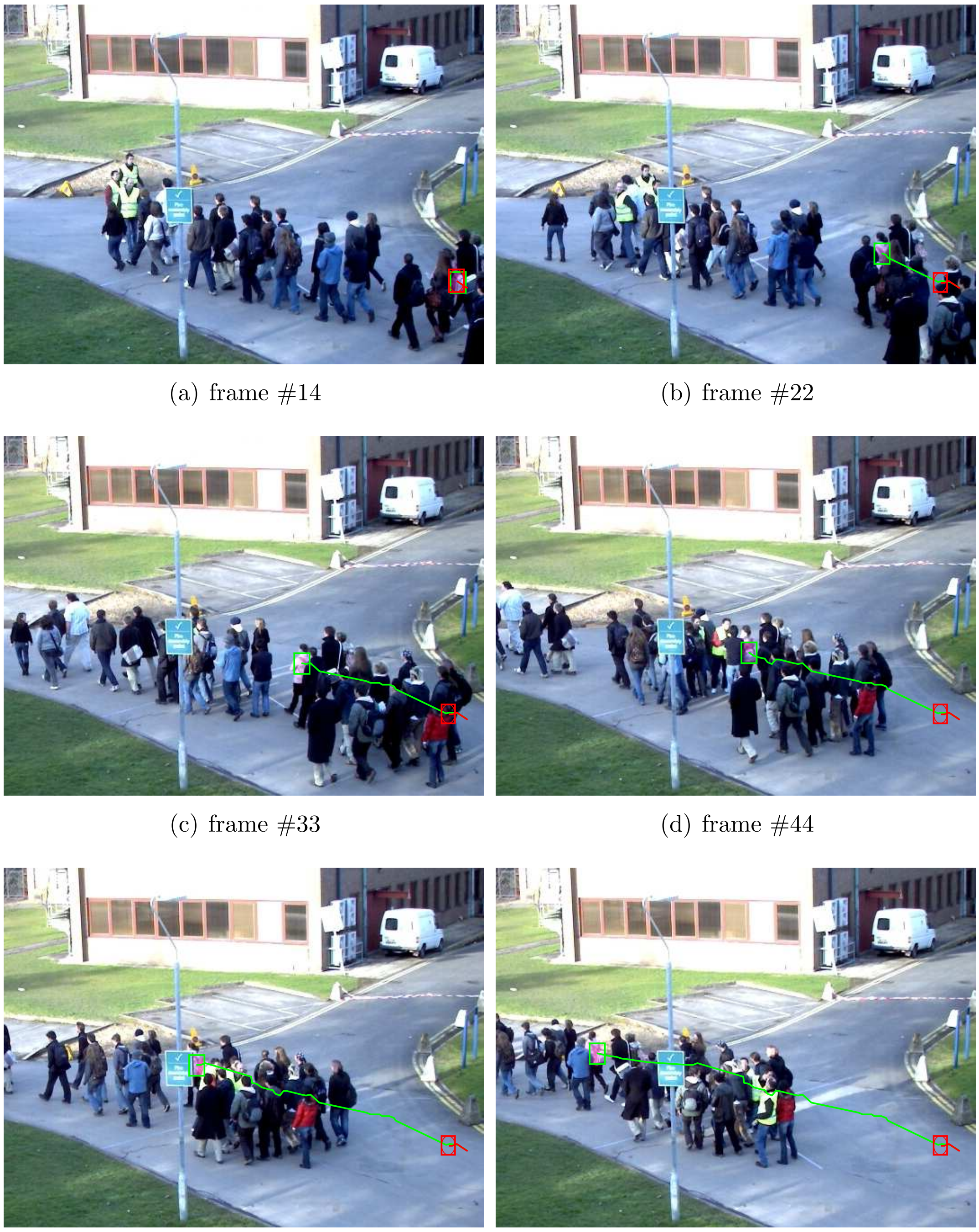


(e) frame #94



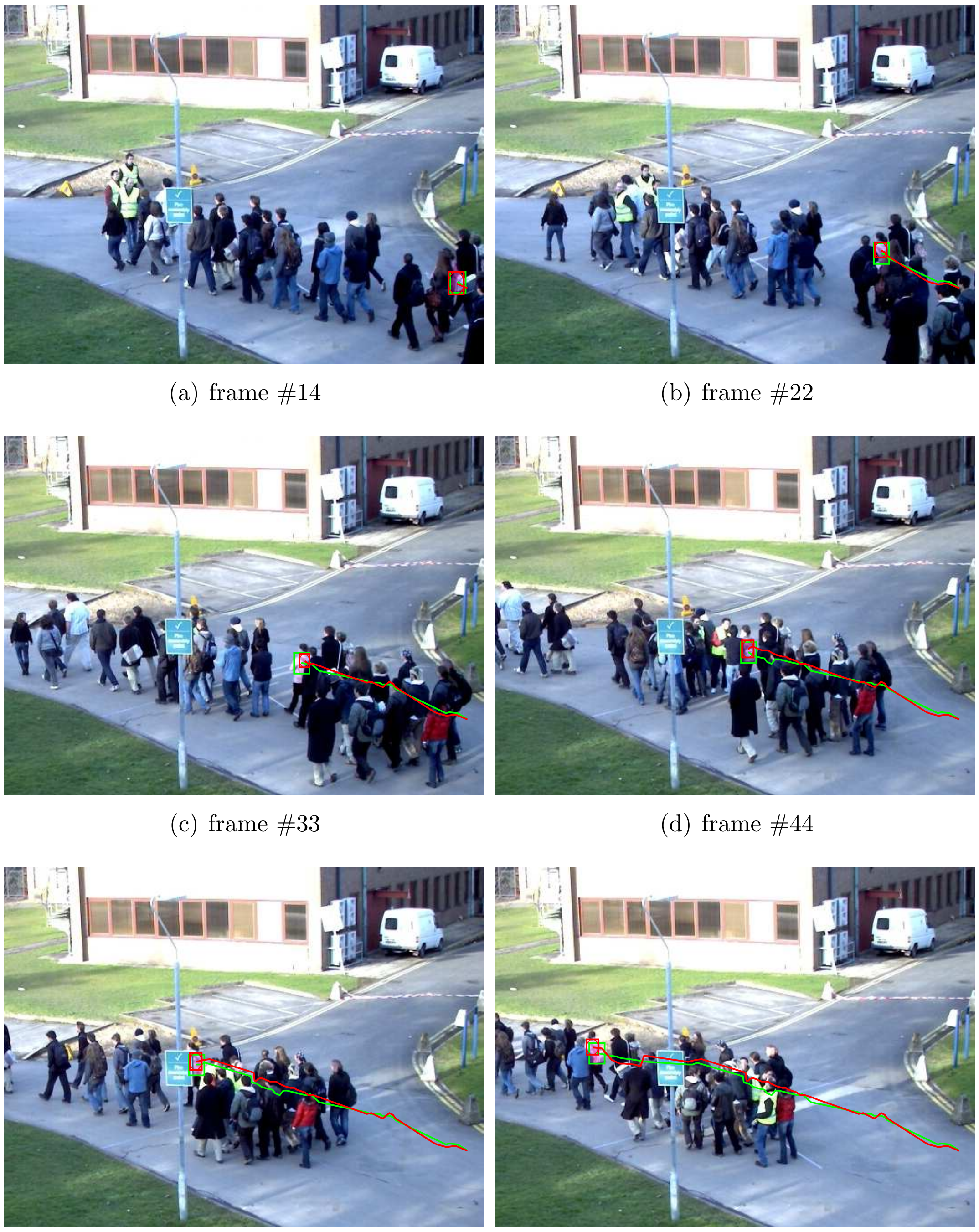
(f) frame #109

Fig. A.15: Improved MS results for PETS2009 S3-Multiple_flow-time_14.46-001 sequence. In each image, the green track is the ground truth, while the red one is the tracker result.



(a) frame #14
(b) frame #22
(c) frame #33
(d) frame #44
(e) frame #55
(f) frame #67

Fig. A.16: Conventional MS results for PETS2009 S3-Multiple_flow-time_14.52-001 sequence. In each image, the green track is the ground truth, while the red one is the tracker result.



(a) frame #14 (b) frame #22

(c) frame #33 (d) frame #44

(e) frame #55 (f) frame #67

Fig. A.17: Improved MS results for PETS2009 S3-Multiple_flow-time_14.52-001 sequence. In each image, the green track is the ground truth, while the red one is the tracker result.

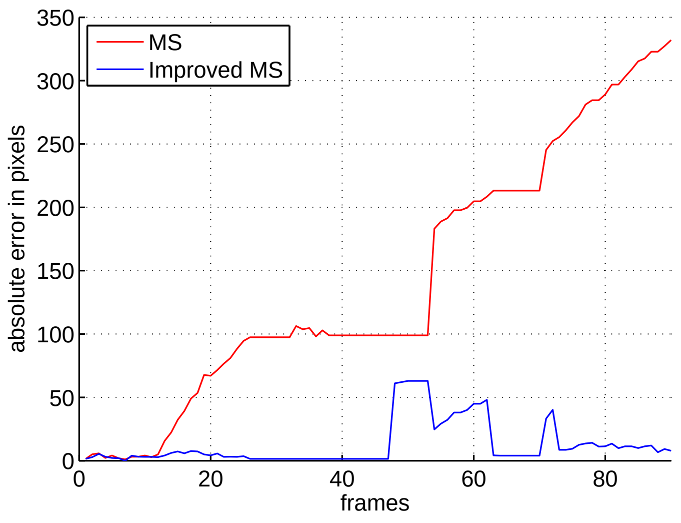


Fig. A.18: Absolute error for PETS2009 S3-Multiple_flow-time_14.46-001 sequence.

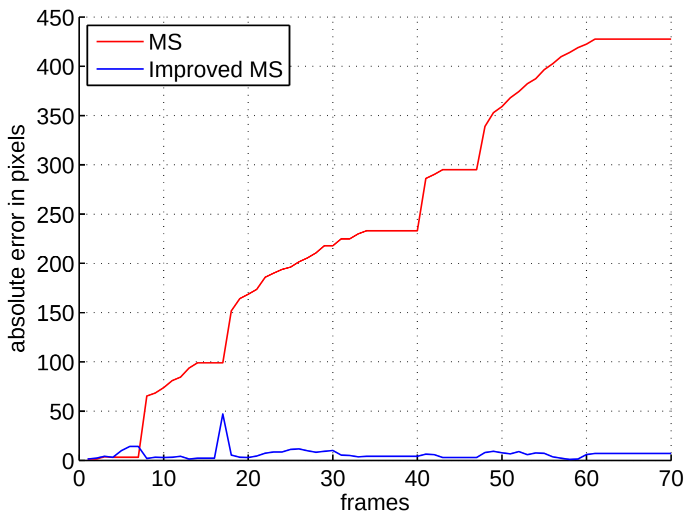


Fig. A.19: Absolute error for PETS2009 S3-Multiple_flow-time_14.52-001 sequence.

Author's publications

1. G. Di Caterina, I. Hunter, and J. J. Soraghan, "DSP embedded smart surveillance sensor with robust SWAD-based tracker," in *Advanced Concepts for Intelligent Vision Systems* (J. Blanc-Talon, W. Philips, D. Popescu, P. Scheunders, and P. Zemcik, eds.), vol. 7517 of *Lecture Notes in Computer Science*, pp. 48–58, Springer Berlin / Heidelberg, 2012.

2. P. Lipinski, S. McCabe, A. Stark, G. Di Caterina, C. Clemente and J. J. Soraghan, "RoboKinect - A low-cost mobile vision system for 2.5D object detection," in *European DSP Education and Research Conference*, pp. 276–280, 2012.

3. J. Zabalza, J. Ren, C. Clemente, G. Di Caterina and J. J. Soraghan, "Embedded SVM on TMS320C6713 for signal prediction in classification and regression applications," in *European DSP Education and Research Conference*, pp. 90–94, 2012.

4. G. Di Caterina and J. J. Soraghan, "Robust complete occlusion handling in adaptive template matching target tracking," *IET Electronic Letters*, vol. 48, no. 14, pp. 831–832, 2012.

5. G. Di Caterina, N. A. Manap, M. M. Ibrahim, and J. J. Soraghan, "Real time door access event detection and notification in a reactive smart surveillance system," in *Image and Signal Processing* (A. Elmoataz, D. Mammass, O. Lezoray, F. Nouboud, and D. Aboutajdine, eds.), vol. 7340 of *Lecture Notes in Computer Science*, pp. 459–467, Springer Berlin / Heidelberg, 2012.

6. G. Di Caterina and J. J. Soraghan, "Adaptive template matching algorithm based on SWAD for robust target tracking," *IET Electronic Letters*, vol. 48, no. 5, pp. 261–262, 2012.

7. G. Di Caterina and J. J. Soraghan, "An improved mean shift tracker with fast failure recovery strategy after complete occlusion," in *IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pp. 130–135, 2011.

8. N. A. Manap, G. Di Caterina, M. M. Ibrahim, and J. J. Soraghan, "Co-operative surveillance cameras for high quality face acquisition in a real-time door monitoring system," in *European Workshop on Visual Information Processing*, pp. 99–104, 2011.

9. G. Di Caterina and J. J. Soraghan, "An abandoned and removed object detection algorithm in a reactive smart surveillance system," in *International Conference on Digital Signal Processing*, pp. 1–6, 2011.

10. G. Di Caterina, I. Hunter, and J. J. Soraghan, "An embedded smart surveillance system for target tracking using a PTZ camera," in *European DSP Education and Research Conference*, pp. 165–169, 2010.

11. N. A. Manap, G. Di Caterina, J. J. Soraghan, V. Sidharth, and H. Yao, "Face detection and stereo matching algorithms for smart surveillance system with IP cameras," in *European Workshop on Visual Information Processing*, pp. 77–81, 2010.

12. N. A. Manap, G. Di Caterina, J. J. Soraghan, V. Sidharth, and H. Yao, "Smart surveillance system based on stereo matching algorithms with IP and PTZ cameras," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pp. 1–4, 2010.

13. N. A. Manap, G. Di Caterina, and J. J. Soraghan, "Low cost multi-view video system for wireless channel," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pp. 1–4, 2009.

References

[1] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: a survey,” *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006. xi, 2, 15, 22, 23, 24, 25

[2] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, vol. 34, no. 3, pp. 334–352, 2004. 1, 7, 16, 22

[3] A. Valera and S. A. Velastin, “Intelligent distributed surveillance systems: a review,” *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192–204, 2005. 8, 15, 92

[4] H. Dee and S. A. Velastin, “How close are we to solving the problem of automated visual surveillance? A review of real-world surveillance, scientific progress and evaluative mechanisms,” *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 329–343, 2008. 22

[5] S. A. Velastin, “CCTV video analytics: recent advances and limitations.” Springer-Verlag Berlin Heidelberg, 2009. 1, 7

[6] “Police ‘not using CCTV properly’.” http://news.bbc.co.uk/1/hi/uk/8158942.stm, Jul 2009. Accessed: 15/05/2012. 1, 6

[7] D. Comaniciu and P. Meer, “Robust analysis of feature spaces: color image segmentation,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 750 –755, 1997. 2, 10, 30, 33, 34

[8] D. Comaniciu and P. Meer, “Mean shift analysis and applications,” in *IEEE International Conference Computer Vision*, vol. 2, pp. 1197–1203, 1999. 33

[9] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” in *IEEE Conference Computer Vision and Pattern Recognition*, vol. 2, pp. 142–149, 2000. 13, 33

[10] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002. 10, 32

[11] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003. 2, 13, 26, 30, 32, 33, 34, 35, 44, 62, 93

[12] M. Piccardi, “Background subtraction techniques: a review,” *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3099–3104, 2004. 2, 10, 17, 18

[13] A. Bayona, J. C. San Miguel, and J. M. Martinez, “Comparative evaluation of stationary foreground object detection algorithms based on background subtraction techniques,” in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 25–30, 2009. 2, 10, 17, 18, 96

[14] Key Note Limited, "Closed-Circuit Television Update Market Report 2012," Jan 2012. 5

[15] RNCOS, "Global CCTV Market Forecast to 2014," Mar 2012. 5

[16] P. Lewis, "Every step you take: UK underground centre that is spy capital of the world." http://www.guardian.co.uk/uk/2009/mar/02/westminster-cctv-system-privacy, Mar 2009. Accessed: 20/04/2012. 5

[17] P. Lewis, "You're being watched: there's one CCTV camera for every 32 people in UK." http://www.guardian.co.uk/uk/2011/mar/02/cctv-cameras-watching-surveillance, Mar 2011. Accessed: 15/05/2012. 5

[18] "'Worthless' CCTV camera footage is not good enough to fight crime, leading QC warns." http://www.telegraph.co.uk/news/politics/6088086/Worthless-CCTV-camera-footage-is-not-good-enough-to-fight-crime-leading-QC-warns.html, Aug 2009. Accessed: 25/07/2012. 5

[19] E. Wallace and C. Diffley, "CCTV control room ergonomics," Tech. Rep. 14/98, Police Scientific Development Branch (PSDB), UK Home Office, 1988. 6

[20] N. Gagvani, G. Groenemann, and A. Smoot, "The Economics of Monitored Video in the Commercial Market," White Paper, Cernium Corporation, Nov 2008. 6

[21] N. Hewitson, "Video Content Analysis, what is it and why would I want it?," White Paper, Smart CCTV Limited, Nov 2005. 7

[22] S. He, J. Soraghan, B. O'Reilly, and D. Xing, "Quantitative analysis of facial paralysis using local binary patterns in biomedical videos," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 7, pp. 1864–1870, 2009. 7

[23] M. Yu, S. Naqvi, A. Rhuma, and J. Chambers, "One class boundary method classifiers for application in a video-based fall detection system," *IET Computer Vision*, vol. 6, no. 2, pp. 90–100, 2012. 7, 16

[24] P. L. Mazzeo, P. Spagnolo, M. Leo, and T. D'Orazio, "Visual players detection and tracking in soccer matches," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 326–333, 2008. 7, 13

[25] J. Xing, H. Ai, L. Liu, and S. Lao, "Multiple player tracking in sports video: a dual-mode two-way bayesian inference approach with progressive observation modeling," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1652–1667, 2011.

[26] M. C. Hu, M. H. Chang, J. L. Wu, and L. Chi, "Robust camera calibration and player tracking in broadcast basketball video," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 266–279, 2011. 7

[27] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 694–711, 2006. 7, 15

[28] O. Vellacott, "Video Analytics Understanding their Limitations." http://www.indigovision.com/documents/public/articles/Analytics Article US.pdf, 2007. Accessed: 07/06/2012. 7

[29] T. D. Raty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 5, pp. 493–515, 2010. 8

[30] G. L. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, and T. Ellis, "Active video-based surveillance system: the low-level image and video processing techniques needed for implementation," *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 25–37, 2005. 8

[31] C. Micheloni, B. Rinner, and G. L. Foresti, "Video analysis in pan-tilt-zoom camera networks," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 78–90, 2010.

[32] R. Tron and R. Vidal, "Distributed computer vision algorithms," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 32–45, 2011. 8, 92

[33] B. Kisacanin, "Examples of low-level computer vision on media processors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 135–140, 2005. 8

[34] I. Hunter, "Overview of embedded DSP design," in *European Signal Processing Conference*, pp. 475–479, 2009.

[35] B. Kisacanin and Z. Nikolic, "Algorithmic and software techniques for embedded vision on programmable processors," *Signal Processing: Image Communication, ELSEVIER*, vol. 25, no. 5, pp. 352–362, 2010. 8

[36] R. C. Gonzalez and R. E. Woods, *Digital image processing.* Prentice Hall, 3rd ed., 2008. 9, 11, 12, 13, 14

[37] E. Maggio and A. Cavallaro, *Video Tracking: Theory and Practise.* John Wiley & Sons, 1st ed., 2011. 9, 11, 15, 22, 23, 24, 25, 36

[38] E. R. Davies, *Machine Vision: Theory, Algorithms and Practicalities.* Academic Press, 1990. 9, 12, 13, 14, 80

[39] M. S. Nixon and A. S. Aguado, *Feature extraction and image processing.* Oxford Academic, 2nd ed., 2007. 9, 11

[40] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. 10

[41] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Addison-Wesley, 1974. 10

[42] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. 10

[43] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson, "A system for video surveillance and monitoring," tech. rep., Carnegie Mellon University, 2000. 11, 15

[44] A. Blake and M. Isard, *Active contours.* Springer, 2000. 11

[45] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994. 11, 12

[46] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, no. 3, pp. 433–467, 1995. 27

[47] D. J. Fleet and Y. Weiss, "Optical flow estimation," in *Mathematical Model in Computer Vision: the handbook* (N. Paragios, Y. Chen, and O. Faugeras, eds.), ch. 15, pp. 239–258, Springer, 2005. 11, 12

[48] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986. 12

[49] M. Heikkila and M. Pietikainen, “A texture-based method for modeling the background and detecting moving objects,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 657–662, 2006. 12, 20

[50] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of Alvey Vision Conference*, pp. 147–151, 1988. 12

[51] N. Dalal and B. Triggs, “Histogram of oriented gradients for human detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005. 12, 15

[52] P. Viola and M. J. Jones, “Robust real-time object detection,” in *International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*, pp. 1–25, 2001. 12

[53] P. Viola and M. J. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, 2001. 12

[54] B. Wu and R. Nevatia, “Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors,” *International Journal of Computer Vision*, vol. 75, pp. 247–266, 2007. 12, 15

[55] M. J. Swain and D. H. Ballard, “Indexing via color histograms,” in *IEEE International Conference on Computer Vision*, pp. 390–393, 1990. 13

[56] I. Rano, E. Lazkano, and B. Sierra, “On the application of colour histograms for mobile robot localization,” in *European Conference on Mobile Robotics*, pp. 189–193, 2005. 13

[57] P. Suhasini, K. Sri Rama Krishna, and I. V. Murali Krishna, “CBIR using color histogram processing,” *Journal of Theoretical and Applied Information Technology*, vol. 16, no. 1, pp. 116–122, 2009. 13

[58] B. Silverman, *Density estimation for statistics and data analysis.* Chapman and Hall, 1986. 14, 31

[59] J. Serra, *Image Analysis and Mathematical Morphology*, vol. 1. Academic Press, 1982. 14

[60] G. Buchsbaum, “A spatial processor model for object colour perception,” *Journal of The Franklin Institute*, vol. 310, no. 1, pp. 1–26, 1980. 14, 97

[61] I. Haritaoglu, D. Harwood, and L. S. Davis, “W4: Real-time surveillance of people and their activities,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000. 15

[62] A. Mittal and L. S. Davis, “M2tracker: a multi-view approach to segmenting and tracking people in a cluttered scene,” *International Journal of Computer Vision*, vol. 51, no. 3, pp. 189–203, 2003. 15, 92, 97

[63] P. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004. 15

[64] H. Li, K. N. Ngan, and Q. Liu, “Faceseg: automatic face segmentation for real-time video,” *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 77–88, 2009.

[65] M. Yang, J. Crenshaw, B. Augustine, R. Mareachen, and Y. Wud, “Adaboost-based face detection for embedded systems,” *Computer Vision and Image Understanding, ELSEVIER*, vol. 114, no. 11, pp. 1116–1125, 2010.

[66] M. Bauml, K. Bernardin, M. Fischer, H. K. Ekenel, and R. Stiefelhagen, "Multi-pose face recognition for person retrieval in camera networks," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 441–447, 2010. 15

[67] M. D. Beynon, D. J. Van Hook, M. Seibert, A. Peacock, and D. Dudgeon, "Detecting abandoned packages in a multi-camera video surveillance system," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2003. 15, 16, 17

[68] M. Bhargava, C. Chen, M. S. Ryoo, and J. K. Aggarwal, "Detection of abandoned objects in crowded environments," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 271–276, 2007.

[69] A. Singh, S. Sawan, M. Hanmandlu, V. K. Mdasu, and B. C. Lovell, "An abandoned object detection system based on dual background subtraction," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 352–357, 2009.

[70] Y. Tian, R. Feris, H. Liu, A. Hampapur, and M. Sun, "Robust detection of abandoned and removed objects in complex surveillance videos," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. PP, no. 99, pp. 1–12, 2010. 15, 16, 17, 20, 81

[71] F. Lamosa, Z. Hu, and K. Uchimura, "Vehicle detection using multi-level probability fusion maps generated by a multi-camera system," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 10–17, 2008. 15

[72] L. M. Brown, "Example-based color vehicle retrieval for surveillance," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 91–96, 2010. 15

[73] B. Heisele, U. Kressel, and W. Ritter, "Tracking non-rigid, moving objects based on color cluster flow," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 257–260, 1997. 15

[74] L. Snidaro, I. Visentini, and G. L. Foresti, "Dynamic models for people detection and tracking," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 29–53, 2008. 15

[75] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 829–836, 2005. 15, 30

[76] F. Tombari, L. Di Stefano, and S. Mattoccia, "A robust measure for visual correspondence," in *International Conference on Image Analysis and Processing*, pp. 376–381, 2007. 15, 27, 28

[77] G. Lavee, E. Rivlin, and M. Rudzsky, "Understanding video events: a survey of methods for automatic interpretation of semantic occurrences in video," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 39, no. 5, pp. 489–504, 2009. 16

[78] H. K. Ekenel, J. Stallkamp, and R. Stiefelhagen, "A video-based door monitoring system using local appearance-based face models," *Computer Vision and Image Understanding, ELSEVIER*, vol. 114, no. 5, pp. 596–608, 2010. 16

[79] G. L. Foresti, L. Marcenaro, and C. S. Regazzoni, "Automatic detection and indexing of video-event shots for surveillance applications," *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 459–471, 2002. 16, 96

[80] C. Piciarelli, C. Micheloni, and G. Foresti, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1544–1554, 2008. 16

[81] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L. Xu, "Crowd analysis: a survey," *Machine Vision and Applications, Springer*, vol. 19, pp. 345–357, 2008. 16, 22

[82] J. Ferryman and A. Shahrokni, "PETS2009: Dataset and challenge," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pp. 1–6, 2009. 16

[83] PETS2006, "Benchmark data." http://www.cvg.rdg.ac.uk/PETS2006/data.html. 17, 44, 88

[84] PETS2007, "Benchmark data." http://www.cvg.rdg.ac.uk/PETS2007/data.html. 44

[85] i-LIDS, "Dataset for AVSS 2007." http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html. 17, 88

[86] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 246–252, 1999. 19, 20

[87] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003. 19, 20

[88] N. J. B. McFarlane and C. P. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications, Springer-Verlag*, vol. 8, pp. 187–193, 1995. 19

[89] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997. 19, 20

[90] C. R. Jung, "Efficient background subtraction and shadow removal for monochromatic video sequences," *IEEE Transactions on Multimedia*, vol. 11, no. 3, pp. 571–577, 2009. 19, 20

[91] J. Connell, A. W. Senior, A. Hampapur, Y. L. Tian, L. Brown, and S. Pankanti, "Detection and tracking in the IBM Peoplevision system," in *IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 1403–1406, 2004. 20, 81

[92] J. K. Aggarwal and Q. Cai, "Human motion analysis: a review," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 428–440, 1999. 22

[93] D. M. Gavrila, "The visual analysis of human movement: a survey," *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82–98, 1999.

[94] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, 2001.

[95] T. B. Moeslund, A. Hilton, and V. Kruger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 23, pp. 90–126, 2006. 22

[96] J. Xiao, S. Baker, M. I., and T. Kanade, "Real-time combined 2D+3D active appearance models," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 535–542, 2004. 23

[97] A. Bovik, *Handbook of image and video processing.* Elsevier, Academic Press, 2nd ed., 2005. 27

[98] V. Bhaskaran and K. Konstantinides, *Image and video compression standards : algorithms and architectures.* Kluwer Academic Publishers, 2nd ed., 1997. 27

[99] S. Wong, "Advanced correlation tracking of objects in cluttered imagery," *Proceedings of SPIE: Acquisition, Tracking and Pointing*, vol. 5810, pp. 158–169, 2005. 27, 29

[100] M. I. Khan, J. Ahmed, A. Ali, and A. Masood, "Robust edge-enhanced fragment based normalized correlation tracking in cluttered and occluded imagery," *International Journal of Advanced Science and Technology*, vol. 12, pp. 25–34, 2009. 27, 29

[101] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981. 27

[102] J. P. Lewis, "Fast Normalized Cross-Correlation." http://scribblethink.org/Work/nvisionInterface/nip.html. Accessed: 31/08/2012. 28

[103] S. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image and Vision Computing, ELSEVIER*, vol. 17, no. 3-4, pp. 225–231, 1999. 30

[104] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99–110, 2003. 30

[105] I. Leichter, M. Lindenbaum, and E. Rivlin, "Mean shift tracking with multiple reference color histograms," *Computer Vision and Image Understanding, Elsevier*, vol. 114, pp. 400–408, 2010. 33, 59

[106] F. Wang, S. Yu, and J. Yang, "Robust and efficient fragments-based tracking using mean shift," *International Journal of Electronics and Commununication, AEU*, vol. 64, pp. 614–623, 2010. 33

[107] H. Zhou, G. Schaefer, Y. Yuan, and M. E. Celebi, "Can mean shift trackers perform better?," in *International Conference Signal-Image Technology and Internet Based Systems*, pp. 2405–2408, 2010. 33

[108] S. H. Lee and M. G. Kang, "Motion tracking based on area and level set weighted centroid shifting," *IET Computer Vision*, vol. 4, no. 2, pp. 73–84, 2010. 33

[109] K. Chen, D. Li, Q. Huang, and L. E. Banta, "Video motion tracking using enhanced particle filtering with mean-shift," in *International Congress Image and Signal Processing*, pp. 387–391, 2010. 33

[110] Y. S. Chia, W. Y. Kow, W. L. Khong, A. Kiring, and K. T. K. Teo, "Kernel-based object tracking via particle filter and mean shift algorithm," in *International Conference on Hybrid Intelligent Systems*, pp. 522 –527, 2011. 33, 34

[111] J. Zhao, W. Qiao, and G. Z. Men, "An approach based on mean shift and kalman filter for target tracking under occlusion," in *International Conference Machine Learning and Cybernetics*, pp. 2058–2062, 2009. 33, 34

[112] X. Wang, L. Liu, and Z. Tang, "Infrared human tracking with improved mean shift algorithm based on multicue fusion," *Applied Optics*, vol. 48, no. 21, pp. 4201–4212, 2009. 33

[113] C. Beyan and A. Temizel, “Adaptive mean-shift for automated multi object tracking,” *IET Computer Vision*, vol. 6, no. 1, pp. 1–12, 2012. 33

[114] A. Yilmaz, “Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–6, 2007. 33

[115] X. Chen, S. Yu, and Z. Ma, “An improved mean shift algorithm for moving object tracking,” in *World Congress on Intelligent Control and Automation*, pp. 5111–5114, 2008. 33, 59

[116] G. R. Bradski, “Computer vision face tracking for use in a perceptual user interface,” *Intel Technology Journal*, vol. 2, no. 2, pp. 1–15, 1998. 34

[117] J. Ning, L. Zhang, D. Zhang, and C. Wu, “Scale and orientation adaptive mean shift tracking,” *IET Computer Vision*, vol. 6, no. 1, pp. 52 –61, 2012. 34, 59

[118] R. T. Collins, “Mean-shift blob tracking through scale space,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. II–234–40, 2003. 34

[119] Y. Lu, C. Guo, and T. Ikenaga, “Estimation-correction scheme based articulated object tracking using sift features and mean shift algorithm,” in *International Conference New Trends in Information Science and Service Science*, pp. 275–280, 2010. 34

[120] K. Deguchi, O. Kawanaka, and T. Okatani, “Object tracking by the mean-shift of regional color distribution combined with the particle-filter algorithms,” in *International Conference Pattern Recognition*, vol. 3, pp. 506–509, 2004. 34

[121] Z. Li, Q. L. Tang, and N. Sang, “Improved mean shift algorithm for occlusion pedestrian tracking,” *Electronics Letters*, vol. 44, no. 10, pp. 622–623, 2008. 34

[122] T. Liu and X. P. Cheng, “Improved mean shift algorithm for moving object tracking,” in *International Conference Computer Engineering and Technology*, vol. 1, pp. 575–578, 2010. 34

[123] T. Chiang, J. Leou, and C. Lin, “An improved mean shift algorithm based tracking system for soccer game analysis,” in *APSIPA Annual Summit and Conference*, pp. 380–385, 2009. 34

[124] F. Porikli and O. Tuzel, “Multi-kernel object tracking,” in *IEEE International Conference on Multimedia and Expo*, pp. 1234–1237, 2005. 35

[125] J. Ning, L. Zhang, D. Zhang, and C. Wu, “Robust mean-shift tracking with corrected background-weighted histogram,” *IET Computer Vision*, vol. 6, no. 1, pp. 62 –69, 2012. 35

[126] Z. Li, J. Chen, and N. N. Schraudolph, “An improved mean-shift tracker with kernel prediction and scale optimisation targeting for low-frame-rate video tracking,” in *International Conference on Pattern Recognition*, pp. 1–4, 2008. 35, 59

[127] G. Di Caterina and J. J. Soraghan, “Adaptive template matching algorithm based on SWAD for robust target tracking,” *IET Electronics Letters*, vol. 48, no. 5, pp. 261–262, 2012. 38

[128] G. Di Caterina and J. J. Soraghan, “Robust complete occlusion handling in adaptive template matching target tracking,” *IET Electronics Letters*, vol. 48, no. 14, pp. 831–832, 2012. 38

[129] Visual Tracking Benchmark, "Dudek face sequence." http://www.cs.toronto.edu/vis/projects/adaptiveAppearance.html. 44, 51

[130] CVRR, "ATON – Testbed Data." http://cvrr.ucsd.edu/aton/shadow/. 51

[131] PETS2009, "Benchmark data." http://www.cvg.rdg.ac.uk/PETS2009/a.html. 51, 68, 105

[132] G. Di Caterina, I. Hunter, and J. Soraghan, "An embedded smart surveillance system for target tracking using a PTZ camera," in *European DSP Education and Research Conference*, pp. 165–169, 2010. 54

[133] G. Di Caterina, I. Hunter, and J. J. Soraghan, "DSP embedded smart surveillance sensor with robust SWAD-based tracker," in *Advanced Concepts for Intelligent Vision Systems* (J. Blanc-Talon, W. Philips, D. Popescu, P. Scheunders, and P. Zemcik, eds.), vol. 7517 of *Lecture Notes in Computer Science*, pp. 48–58, Springer Berlin / Heidelberg, 2012. 54

[134] Spectrum Digital Inc., *TMS320DM6437 Evaluation Module Technical Reference – 509105 -0001 Rev. C*, December 2006. 54

[135] Texas Instruments, *TMS320DM6437 Digital Media Processor – SPRS345D*, June 2008. 54

[136] N. A. Manap, G. Di Caterina, M. M. Ibrahim, and J. J. Soraghan, "Co-operative surveillance cameras for high quality face acquisition in a real-time door monitoring system," in *European Workshop on Visual Information Processing*, pp. 99–104, 2011. 55, 87

[137] Texas Instruments, *TMS320C6000 Programmer's Guide – SPRU198I*, March 2006. 57

[138] G. Di Caterina and J. J. Soraghan, "An improved mean shift tracker with fast failure recovery strategy after complete occlusion," in *IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pp. 130–135, 2011. 59

[139] "Xiph.org – derf's test media collection." http://media.xiph.org/video/derf/. 68

[140] G. Di Caterina and J. J. Soraghan, "An abandoned and removed object detection algorithm in a reactive smart surveillance system," in *International Conference on Digital Signal Processing*, pp. 1–6, 2011. 76

[141] M. Magno, F. Tombari, D. Brunelli, L. Di Stefano, and L. Benini, "Multimodal abandoned/removed object detection for low power video surveillance systems," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 188–193, 2009. 78

[142] G. Di Caterina, N. Manap, M. Ibrahim, and J. Soraghan, "Real time door access event detection and notification in a reactive smart surveillance system," in *Image and Signal Processing* (A. Elmoataz, D. Mammass, O. Lezoray, F. Nouboud, and D. Aboutajdine, eds.), vol. 7340 of *Lecture Notes in Computer Science*, pp. 459–467, Springer Berlin / Heidelberg, 2012. 87

[143] CAVIAR, "Test case scenarios." http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1. 88

[144] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001. 92

[145] P. Remagnino, A. I. Shihab, and G. A. Jones, "Distributed intelligence for multi-camera visual surveillance," *Pattern Recognition, Elsevier*, vol. 37, no. 4, pp. 675–689, 2004. 92

[146] C. Micheloni, G. L. Foresti, and L. Snidaro, “A network of co-operative cameras for visual surveillance,” *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 205–212, 2005.

[147] M. Taj and A. Cavallaro, “Distributed and decentralized multicamera tracking,” *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 46–58, 2011. 92

[148] Q. Cai and J. K. Aggarwal, “Tracking human motion in structured environments using a distributed-camera system,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1241–1247, 1999. 92

[149] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. Van Gool, “Color-based object tracking in multi-camera environments,” in *Pattern Recognition Symposium of the German-Association-for-Pattern-Recognition*, 2003. 92

[150] S. L. Dockstader and A. M. Tekalp, “Multiple camera tracking of interacting and occluded human motion,” *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1441–1455, 2001. 92

[151] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multicamera people tracking with a probabilistic occupancy map,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 267–282, 2008. 92

[152] J. Orwell, P. Remagnino, and G. A. Jones, “Multi-camera colour tracking,” in *IEEE Workshop on Visual Surveillance*, pp. 14–21, 1999. 92

[153] O. Javed, K. Shafique, Z. Rasheed, and M. Shah, “Modeling inter-camera spacetime and appearance relationships for tracking across non-overlapping views,” *Computer Vision and Image Understanding*, vol. 109, no. 2, pp. 146–162, 2008.

[154] E. Lobaton, R. Vasudevan, R. Bajcsy, and S. Sastry, “A distributed topological camera network representation for tracking applications,” *IEEE Transactions on Image processing*, vol. 19, no. 10, pp. 2516–2529, 2010. 92

[155] F. Porikli and A. Divakaran, “Multi-camera calibration, object tracking and query generation,” in *International Conference on Multimedia and Expo*, vol. 1, pp. I–653–6, 2003. 93

[156] B. Moller, T. Plotz, and G. A. Fink, “Calibration-free camera hand-over for fast and reliable person tracking in multi-camera setups,” in *International Conference on Pattern Recognition*, pp. 1–4, 2008. 93

[157] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, “Robust real-time unusual event detection using multiple fixed-location monitors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 555 –560, 2008. 96

[158] T. D’Orazio, P. Mazzeo, and P. Spagnolo, “Color brightness transfer function evaluation for non overlapping multi camera tracking,” in *ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 365–370, 2009. 97

[159] P. Doubek, T. Svoboda, and L. Van Gool, “Monkeys - a software architecture for viroom - low-cost multicamera system,” in *International Conference on Computer Vision Systems*, 2003.

[160] F. Porikli, “Inter-camera color calibration by correlation model function,” in *IEEE International Conference on Image Processing*, vol. 2, pp. 133–136, 2003. 97