Tracking People across Multiple Cameras in the Presence of Clutter and Noise

Alberto Colombo

A thesis submitted in partial fulfillment
of the requirements of Kingston University
for the degree of DOCTOR OF PHILOSOPHY

May, 2011

Faculty of Computing, Information Systems and Mathematics
Digital Imaging Research Centre

Kingston University London

Abstract

As video surveillance systems become more and more pervasive in our society, it is evident that simply increasing the number of cameras does not guarantee increased security, since each operator can only attend to a limited number of monitors. To overcome this limit, automatic video surveillance systems (AVSS, computer-based surveillance systems that automate some of the most tedious work of security operators) are being developed. One such task is tracking, defined by the end users in this project as "keeping a selected passenger always visible on a surveillance monitor".

The purpose of this work was to develop a single-person, multi-camera tracker that can be used in real time to follow a manually-selected individual. The operation of selecting an individual for tracking is called *tagging*, and therefore this type of tracker is known as a *tag and track* system. The developed system is conceived to be deployed as part of a large surveillance network, consisting of possibly hundreds of cameras, with possibly large blind regions between cameras.

The main contribution of this thesis is a probabilistic framework that can be used to develop a multi-camera tracker by fusing heterogeneous information coming from visual sensors and from prior knowledge about the relative positioning of cameras in the surveillance network. The developed tracker has been demonstrated to work in real time on a standard PC independently of the number of cameras in the network. Quantitative performance evaluation is carried out using realistic tracking scenarios.

Acknowledgments

Firstly, I would like to thank James Orwell and Sergio Velastin, my supervisors, for their invaluable help and support. Not only have they made possible the eventual submission of this thesis (me notwithstanding), but they also helped make these years an exciting and enriching experience, both professionally and personally.

I am grateful to the European Community for the sponsorship of this project. Its Framework Programme 6 may be the key to "strengthening the scientific and technological bases of industry and encourage its international competitiveness", as they put it, but - and they don't say it - it is a unique chance to travel around the continent meeting new people and new organisations. Our partners in the FP6 Caretaker project have provided useful commentary and insight.

I would also like to thank my friends and colleagues, with whom I have long discussed theory and practice of all my problems (academic or otherwise). In particular, and in no particular order, Justin Cobb, Norbert Buch, Jesús Martínez del Rincón, Fei Yin, Maria Valera, Valerie Leung, J.C. Nebel and Dimitrios Makris. Many many more people (friends, flatmates, passers-by...) provided some help, somehow, at some point; to many to quote them all, and too unfair to quote only some: thanks to all of you!

A surprising "thanks" goes to the Surrey Club, that provided a fertile environment for lunch-time conversations covering all topics of human endeavour, from mathematics to linguistics, from reproduction to global warming. They also gave me the recipe of their chocolate fudge pudding.

And to conclude, a heart-felt thank you to my father and my friends far away in Lecco. Thanks to their affection, shining like a distant beacon, I know that wherever I am I will never be lost.

List of Figures

Contents

List of Symbols

AVSS 1) Automatic Video Surveillance Systems. 2) The IEEE International Conference on Advanced Video and Signal Based Surveillance.

CCTV Closed-circuit television, the use of cameras to transmit a signal to a specific set of monitors.

CIF Common Intermediate Format, a format used to standardize the horizontal and vertical resolutions in pixels of video sequences. CIF corresponds to a resolution of 384x288, 2CIF to 704x288, and 4CIF to 704x576.

DFT Discrete Fourier Transform.

FPR False Positive Rate of a binary classifier. The ratio between the number of True Negatives, over the total number of negative samples. Also known as specificity.

GMM Gaussian Mixture Model, a motion detection algorithm.

GTT Gruppo Torinese Trasporti (the public transportation agency in Turin, Italy). The acronym also refers to the dataset collected by GTT for use within the CARETAKER project.

HMM Hidden Markov Model.

ROC Receiver operating characteristic. It is a graphical plot of FPR versus TPR, for a binary classifier as its discrimination threshold is varied.

RTP Real-time Transport Protocol, a standard internet protocol for the delivery of audio and video data streams.

TnT Tag and Track, i.e. a manually initialised tracker.

TPR True Positive Rate of a binary classifier. It is computed as the ratio between the True Positives over the total number of positive samples. Also known as sensitivity.

Chapter 1

Introduction

Video surveillance systems are becoming increasingly pervasive in our society [40], with private and public companies making extensive use of them to improve safety and security of their premises. However, simply increasing the number of cameras does not guarantee more effective surveillance, since a security operator can only attend to a limited number of monitors, and increasing the number of operators is often too expensive, as opposed to cameras, which, after an initial capital expenditure for installation, only require small maintenance costs. At the same time, the increase in computational power available on cheap hardware has made possible the development of affordable *Automatic Video Surveillance Systems* (AVSS), computer systems that perform video analysis and automate some of the most tedious work of security operators, thus improving operators' efficiency.

Working closely with transportation and security operators, one of the needs that constantly emerged was to have a system that helps operators keep specified individuals in view on a monitor. It is often the case that an operator wants to "keep an eye" on a person who looks suspicious or vulnerable, such as a woman alone late at night, or a troublemaker already known to security officers. Having this person always visible on a surveillance monitor would allow faster intervention if a crime is committed, thus increasing the chances of stopping the offender and/or promptly succouring the victim.

However, following a person on surveillance monitors is a difficult task that requires the full attention of specialised personnel, and for economic reasons it is not possible to divert so many resources on this task in the absence of immediate danger. Hence, the request for a software that is able to follow, or *track*, an operator-selected person across different surveillance cameras, even

when this person goes out of view for a short time, and in the presence of crowd. The act of selecting a target (either manually or by an external system) is called *tagging*, and therefore the system described in this thesis is called *tag and track* (TnT). The expression *tag and track* is also used outside of the Computer Vision community and may refer to any kind of user-initiated tracking (radar, RFID, etc.) [29, 81, 80].

To the authors' best knowledge, there are no TnT systems ready for sale from any surveillance software vendor, and the very topic of tracking people across cameras is fairly recent in video surveillance, as shown by the literature review in Chapter 2.

The development of a TnT system proved to be a challenging task. Functional requirements include: robustness with respect to occlusions, moderate crowding and poor video quality, and real-time operation. Occlusions include both dynamic occlusions (caused by interposition of other people between the camera and the target) and static ones (caused by environment features - such as pillars - or by non-overlapping camera fields of view). Some AVSSs, such as the one described in this thesis, are meant to be deployed on an already established surveillance network, and should therefore work with whatever video quality is available on-site; this implies very different light conditions, colour responses, frame rates and resolutions across different cameras. Real-time operation means that the output of the tracker (i.e. the current position of the person being followed) should be available to the operator after a fixed, short delay with respect to the live video stream. In principle, the operator should have the impression that the tracking is happening instantly: in practice, during informal conversations, security operators we spoke with reported that delays up to 5 seconds are acceptable.

In order to tackle such a difficult problem, some form of modularisation had to be devised from the very beginning. The author's studies focused on a probabilistic multi-camera tracking framework that permits integrating information from a variable number of heterogeneous modules in real time. This information comes from tracking modules, working on single, calibrated cameras, and from people re-identification modules, working across cameras. A study on motion detection and colour correction was also carried out, since these are useful pre-processing stages in an AVSS.

Chapter 3 describes the probabilistic framework and the tracking and re-identification modules, as well as single and multiple camera calibration, that form the main contribution of this work. This framework allows heterogeneous information about a target (e.g. appearance and position) and a surveillance site (e.g. the camera topology) to be fused in order to enable multi-camera track-

ing. The framework was designed to enable real-time tracking, which means that the target's new position has to be computed in a fixed amount of time at each frame, regardless of crowdedness, number of cameras, and length of the current track. This poses significant challenges and, as often a compromise must be stricken between efficacy and computational complexity. In Chapter 4, two novel algorithms for motion detection and colour calibration, respectively, are described. The motion detection algorithm exploits periodicity of certain background elements (such as flashing lights or escalators) to improve segmentation in those areas of the image. The colour calibration algorithm can compensate for the different colour responses of different cameras and to illumination changes across different scenes; it is completely unsupervised, and it exploits the fact that, in a surveillance scenario, the same foreground data (i.e. moving people) is observed by a large number of cameras. Together, these algorithms form a secondary contribution of this thesis. Chapter 5 presents practical implementation issues and experimental results on real surveillance videos, describing all metadata needed to run the tracker and the additional tools developed to collect it. With the tracker set up, performance evaluation is carried out on a dataset consisting of real surveillance video footage from over 20 cameras. This leads naturally to Chapter 6, where a critical discussion of the tracking framework is give, and some ideas for future developments are presented.

The multi-camera tracking framework exploits commonalities between the problem of tracking and other problems that are commonly studied in Computer Vision, namely single-camera tracking and people re-identification. In fact, the tracking method could be summarised as *single-camera tracking* in one video stream, combined with *people re-identification* in streams from adjacent cameras. How these two algorithms are combined is detailed in Section 3.4, and the state of the art our solution is based upon is presented in sections 2.1.1 and 2.1.5.

This project was funded by the EUvia the CARETAKER [102], a 30-month research project within the Sixth Framework Programme (FP6) that was completed in September 2008. It aimed at studying, developing and assessing multimedia knowledge-based content analysis, knowledge extraction components, and metadata management sub-systems in the context of automated situation awareness, diagnosis and decision support. More precisely, CARETAKER focused on the extraction of a structured knowledge from large multimedia collections, recorded over networks of cameras and microphones or acquired in real-time, deployed in real sites. CARETAKER included 9 partners: 2 commercial companies (Thales Communications France and Solid Information Technology

- now part of IBM), 5 research institutes (Multitel, INRIA[1] Sophia Antipolis, Kingston University, Idiap, and Brno University of Technology), and 2 public transportation companies (Agenzia per la Mobilità del Comune di Roma and GTT - Gruppo Torinese Trasporti).

[1] Institut National de Recherche en Informatique et Automatique.

Chapter 2

Literature Review

This chapter presents existing solutions to the problem of tracking (both in general terms, and specifically for multi-camera visual tracking) and to ancillary problems that are often encountered while tackling visual tracking, such as camera calibration and motion detection. Solutions to the tracking problem are presented from a historical perspective, starting with the original definition of tracking, to arrive to the current task of visually tracking people across multiple cameras. After that, in the second half of the chapter, the state of the art on performance evaluation, pre-processing, camera calibration, and metadata representation is presented. These topics were selected because they all emerged, during this research project, as sub-problems of tracking.

2.1 A Historical Perspective

The Free Online dictionary defines *tracking* as the pursuit of a person or animal by following tracks or marks that they have left behind [103]. With the invention of the radar in the 1930s, and its strategic deployment by the Royal Air Force during World War II [75], the semantics of "tracking" had to be extended to include ships and airplanes. The tracks they would leave behind are radio waves bouncing off their metallic hulls. Tracking, applied to radar signals, consists of associating targets (radar detections with distance, bearing and size) to actual flying or floating objects. If during WWII this process was entirely manual, the ever increasing use and sophistication of radars, both military and civilian, in subsequent years, has generated interest in algorithms capable of tracking large number of objects using detection data from many and possibly diverse

sensors [9] (not necessarily radars). In this new context, *tracking* can be defined as the processing of measurements obtained from a target in order to maintain an estimate of its current state, which typically consists of:

- kinematic components - position, speed, acceleration, ...
- feature components - appearance of the objects, as seen by the sensor (reflected signal strength for a radar).

2.1.1 Tracking with an Imaging Device

Measurements are noise-corrupted observations related to the state of a target. For instance, radar measurements are based on round-trip time and return strength of a radio signal. A radar produces an array of "points", or *blips*, each corresponding to at least one target.

An imaging device such as a video camera can detect the intensity and colour of light emitted or reflected by objects placed in front of it. Thanks to video cameras, security operators can track large objects (mostly humans and vehicles) moving within an environment, by following the tracks left by photons that bounce off the targets and hit the camera sensors. A camera sensor, however, consists of a grid of light-sensitive elements, each continuously providing information about the colour and intensity of the light it is hit by. A camera does not directly indicate the presence or the absence of a target, therefore its presence must be detected indirectly using the information provided by the imaging sensor, for example using one of the motion detection algorithms reviewed in Section 2.4.1.

Target tracking can be divided up in three stages: detection, initialisation, and track maintenance (precision tracking or continuation). A sensor can be characterised by its detection probability p_D and false alarm probability p_{FA}. An ideal sensor has $p_D = 1$ and $p_{FA} = 0$; in this case, a track can be initiated as soon as there is a detection, and each detection measurement can be used to estimate the target state. However, a realistic sensor has $p_D < 1$ and $p_{FA} > 0$, and several techniques have been developed to discard inconsistent detections and to work with missing ones [17, 8]. Additionally, some tracking scenarios, such as tag-and-track, require manual initialisation, in which case it is a human (or an external process) that indicates to the tracker which detection should be tracked. Once a track is initialised, the video feed and the detection output can be processed by an image-based tracker, explained in Section 2.2, which uses certain features of the image, to estimate the position (on the image) of the target in each new frame.

2.1.2 Track Maintenance

The general goal of track maintenance is to increase the accuracy of measurements obtained from the sensor, and to obtain estimates of variables not measured by the sensor (i.e. estimating a target speed given several measurements of its position). If the target has a linear motion model, and the sensor measurements are affected by white Gaussian noise with known moments, the Kalman Filter (KF) [112] is the optimal tracker, and it has been used successfully in many surveillance applications, including scenes with partial static and dynamic occlusions [117].

If the motion model of the target is non-linear, the Extended Kalman Filter (EKF) can be used where a continually updated linearisation around the previous state estimate is calculated [52]. This approach is a simple way of dealing with non-linearities, but only produces a reasonable estimate if the linearisation sufficiently approximates the non-linear system, and if the initial estimate is sufficiently close to the true solution.

The Unscented Kalman Filter (UKF) is an alternative to the EKF. In the EKF, the state distribution is approximated by a Gaussian random variable (GRV), which is then propagated analytically through the first-order linearisation of the non-linear system [111]. On the other hand, the UKF uses deterministic sampling and propagates the samples through the system dynamics, the result of which is approximated by a GRV. The UKF has the advantage that it is accurate to the third order in a Taylor series expansion for any non-linearity, as compared to a first order approximation using the EKF.

While the LKF, EKF and the UKF all assume that the process and measurement errors of the system can be modelled by a Gaussian, the particle filter [6] (also known as iCONDENSATION) approximates any probability distribution with a large set of particles. These particles are propagated through time using importance sampling, allowing any arbitrary process model to be used, thus offering flexibility. The disadvantage is its computational complexity.

2.1.3 Data Association and Multi-Object Tracking

Due to the presence of multiple targets within the detection threshold, false detections, and missed detections, there will not always be a trivial, one-to-one mapping between measurements and targets. The process of deciding which measurement to feed into which tracker is called *data association*. A comprehensive survey of data association techniques is outside the scope of this thesis. However, some of the best known techniques are briefly presented here.

One of the most basic forms of data association is the nearest neighbour association [9]. At each time step, this algorithm simply associates each observation with the nearest target, according to some domain-specific measurement of distance. Nearest neighbour, however, tends to perform poorly in cluttered environments.

The multi-hypothesis tracker (MHT) can be used to assign a batch of measurements to a set of tracks [51]. The MHT finds the best association by enumerating all possible associations and keeping the one that minimises some cost function. In the probabilistic multi-hypothesis tracker (PMHT), data association and state estimation are both performed in a single estimation process of two sets of random variables: target states and target-measurement associations [88].

Schikora *et al.* [90] propose an implementation of MHT based on optical flow [93] (for tracking) and finite state statistics (for data association). Optical flow can be computed with any of the may existing algorithms, such as [93]. Finite set statistics [66] is a Bayesian filtering technique that uses sets, instead of vectors, to represent states and observations, which makes it particularly effective for multi-object tracking. The algorithm is evaluated on a publicly available dataset, but only qualitative performance results are reported.

Berclaz *et al.* [12] propose a new algorithm for data association, that can be used with any detection algorithm to create an effective and efficient multi-object tracker. In order to find the optimal set of tracks for multiple objects, a data association algorithms should enumerate all possible associations, which is an NP-complete problem. A faster algorithm can find a solution in polynomial time, by using heuristics or a probabilistic approach, but it may miss the actual optimum. The authors propose to cast the problem into one of constrained flow optimisation, which can be solved in polynomial time yielding quasi real-time performance on realistic datasets. The algorithm was evaluated on the PETS dataset [82], outperforming all trackers participating the PETS-2009 tracking workshop.

2.1.4 Tracking across Multiple Cameras

When there are multiple objects to be tracked across multiple cameras, it may be desirable to split the computational load into a number of independent processes. This can allow better scalability, fault tolerance, or both. There are three main types of supporting architectures [78]: a centralised architecture where all the information is processed at a central point, a distributed architecture where some low level processing is performed at each node before communication with

the central processing unit, and a decentralised architecture where there is no centralised facility: each node performs its own processing and communicates high-level metadata with the other nodes over the network on a peer-to-peer basis. Regardless of the chosen architecture, the layout of the camera network (i.e. the connections between cameras) must be known to the tracker: this is discussed in Section 2.5.

Each architecture has its pros and cons. The centralised option has the obvious disadvantages of a single point of failure and a possible communication bottleneck; moreover it is not scalable. A distributed architecture partially ameliorates the problems of a centralised architecture. A decentralised architecture on the other hand does not suffer from the problems above; it is scalable and robust, meaning that nodes can join or split from a network easily. However, from a group decision-making point of view, the centralised architecture means that decisions are made at one location, requiring less negotiation and bargaining. The decentralised architecture, on the other hand, needs to employ multiple rounds of bargaining to reach a globally optimal solution.

Another point to consider is the end-user application. For example, for visual surveillance, where existing systems usually already have a central monitoring facility, a decentralised architecture might not be necessary.

Regardless of the supporting architecture, data fusion (track and identity fusion) can be performed across a network efficiently using the concept of decentralised data fusion (DDF) [69]. In this scheme, only new information is passed around, and each message can summarise information from many new measurements, making communication efficient. DDF can be achieved using a Kalman filter framework (or more precisely, the inverse of it, i.e. the information filter framework).

2.1.5 Person re-identification

In a video surveillance scenario, sometimes operators do not need to know a target's position at any instant in time, but they only need to determine whether a target (the *query*) has already been observed in a camera network. This problem is similar enough to multi-camera tracking that they can be cast into one another; in particular, multi-camera tracking is a form of person re-identification when the tracker attempts to reacquire a target that has gone out of view.

Bak *et al.* [7] propose two methods, one based on haar-like features [61], and the other on dominant colour descriptors (an MPEG-7 descriptor presented in Section .3.7.2). In both cases, the video stream from one camera is processed by

a human detector, which provides foreground "human blobs" for the signature generation module (which is either haar-like or dominant colour-based). Each blob is also tracked using a single-camera tracker, so that signatures can be generated using several views of the same person. Experimental results on the CAVIAR [21] dataset show a precision of 80% at equal error rate, while on the (more challenging) i-LIDS [44] dataset this figure is reduced to about 40%.

Satta *et al.* [89] provide a theoretical framework for person re-identification that they call *Multiple Component Matching* (MCM). As in the previous paper, it assumes that a human detection algorithm extracts foreground "blobs" and supplies them to the MCM re-identification module. Each blob is split between upper and lower body, and each half is further subdivided in a set of random patches. Every patch is described by its colour histogram in HSV space and its position along the y axis. In order to improve robustness to lighting changes, some patches are modified by adding or subtracting a fixed value to the RGB colour of each pixel in the patch, as this will simulate its appearance after a change in illumination. The modified patch is then added to the set, in place of the original one. The sets of patches can be generated by a single blob or, if a tracker is present, by multiple blobs corresponding to the same individual. These two sets of patches, called a *template* in the paper, can be compared for similarity, by providing two distance functions D (between templates) and d (between sets). D is the average of the distances between sets. To compute d, the Bhattachaaryya distance between each pair of patches is computed; then, d is computed as the *Haussdorf distance* between the two sets, defined as the maximum of the minimum distances between each element of one set and each element of the other [86]. TODO performance evaluation.

People re-identification systems use only appearance to determine equality of two individuals, whereas multi-camera trackers, reviewed in the following section, also use topological information about camera layout. Clearly, because it does not need topological information, a re-identification system is much easier to set up than a full-fledged tracker, however this also means that matching will be less reliable. For this reason, the output of a re-identification system is usually a set of candidate individuals, ranked by likelihood of matching the query individual. The user is left to decide whether a real match is within the provided set, or if a larger set should be provided.

2.1.6 Appearance Description

A large volume of literature connected to tracking or re-identification is devoted to methods for describing a target's appearance. An appearance descriptor can

be applied to a person's image (or sequence of images), and the output is a compact representation, usually called *description, observation* or *signature*. Two descriptions can be compared for (dis)similarity, and this measure is connected to how likely it is that the two descriptions describe the same person. A more in-depth theoretical analysis on this topic is presented in Section 3.7; in this section, we shall give a brief overview on existing appearance descriptors.

Truong *et al.* [CITE People Re-identification by Means of a Camera Network Using a Graph-based Approach] developed a simple yet effective descriptor, which Section 3.7.3 analyses in detail. Given the silhouette of a target, it is divided vertically in n parts, and for each part the average colour is computed. The resulting array of n RGB values can be used as a description of the target. Its simplicity notwithstanding, the descriptor performs remarkably well, correctly identifying almost 100% of the individuals in a realistic (but private) dataset. The experiments performed by the author of this thesis, with a custom implementation of this descriptor, also show that it outperforms many more complicated approaches [27].

Hamdoun *et al.* [41] propose to generate descriptions using short sequences of frames where the target is visible, therefore requiring a single-camera tracker. In each frame, a number of key-points are detected using a variant of SIFT features[63]. To measure the similarity between two description, the *sum of absolute differences* (SAD) is used. Even though it is not clear from the paper, presumably the difference is calculated for each pair of key-points from the two descriptions being compared, and all differences are added together. Evaluation is carried out on the publicly available CAVIAR dataset [21], yielding a precision of 80% at equal error rate. However, the fact that this descriptor requires a tracker makes it undesirable for a highly modular system such as the one described in this thesis, since it is desirable that appearance descriptors work independently of tracking in order to minimise coupling between modules.

2.1.7 Existing Tag-and-Track Systems

As it will become evident, the existing literature on tracking across multiple non overlapped cameras is scarce. For this reason, some of the papers included in this review are not strictly relevant, for example because they assume overlapping camera views, or because the presented algorithm is not real-time and can only be used in an off-line, "forensic" mode after all relevant video is made available. However, the large number of multi-camera tracking papers makes a comprehensive review unfeasible; therefore, the author of this thesis chose a subset that, it is hoped, is representative of the current state of the art.

Black and Ellis [14, 15] describe a multi-camera tracker. It deals only with overlapping cameras (as it uses an epipolar approach to reconstruct the 3d position of a target), and it is not realtime (observations are collected over a period of time, and associated into tracks in a subsequent stage). The problem of non-overlapping cameras is dealt with in a subsequent paper [16], but the method remains non-realtime.

Mittal and Davis [72] developed M_2Tracker, a multi-camera tracker capable of segmenting and tracking people in a cluttered scene. M_2Tracker uses a wide-baseline stereo algorithm to segment individuals in a crowd, and therefore it is not suitable for a large surveillance network, where camera views rarely overlap. Detection and tracking are based on an iterative algorithm, therefore the method is not suitable for real-time tracking. It uses a colour descriptor of each person based on subdividing the target image into a number of horizontal stripes, similarly to what was done for this thesis (based on [106]).

Thirde *et al.* [104] present a multi-camera tracking component of a complete surveillance system that can recognise activities around a parked aircraft to improve efficiency, safety and security of the servicing operation. The system operates in real-time and can resolve merging, dynamic occlusion, fragmentation and complex object interaction in the congested area around an aircraft during servicing. The system, however, was designed specifically for the partly controlled scenario of aircraft servicing, and requires several cameras with overlapping fields of view, all calibrated with respect to the same ground plane, in order to operate. The system is composed of a single-camera tracking module (based on motion detection), and a multi-camera module that fuses information coming from the single-camera trackers.

Madden and Piccardi [65] developed a framework for multi-camera tracking. This framework can deal with non-overlapping cameras and long occlusions. However, it relies mostly on appearance and it has an explicit "matching phase" where tracks from all cameras are associated: tracking is performed by single-camera trackers in real-time, and there is no attempt to preserve identity across cameras at this stage. After a number of tracks has been collected, target appearances are used to find the globally optimal association between tracks; the method is not, therefore, realtime.

Ning and Tan [79] present a novel approach for tracking a moving target in a large, heterogeneous network of fixed and moving cameras. The system is targeted at city-wide surveillance, where bus-mounted cameras could be used to aid fixed surveillance cameras for tracking people over a vast area. A map of the city is divided up into a large number of discrete cells, small enough to contain no more than one target (the paper suggests square cells of 1 metre,

as a compromise between too large a number of cells and too large cells). All cameras, fixed and mobile, are geo-located and fully calibrated, so that their pose and footprint is known with respect to the city map (buses need to be equipped with a GPS receiver); calibration error has to be small compared to cell size. All videos need to be time-stamped using a common clock signal, possibly derived from the GPS. Any appearance-based descriptor can be used to match two observations and generate a probability of them belonging to the same individual. Even though the approach is theoretically sound, all performance evaluation was done on simulated data.

Zhang *et al.* [120] of ObjectVideo present a multi-camera tracker and Intelligent Video Surveillance (IVS) system. The work presented in their paper is similar in scope to the on-line part of the CARETAKER system, although the system design is less modular, and all stages of the pipeline (motion detection, tracking, classification, and event detection) are predefined: tracking and classification produce metadata that is fed into a rule-based system in order to produce events. In ObjectVideo's IVS, all video processing is performed by smart cameras, which send the resulting metadata (and no video) to a "fusion sensor", where information fusion and tracking are carried out. In contrast, the CARETAKER processing components (tracker and event detectors) are independent modules, that can be used stand-alone or in conjunction with a rule-based system for high-level analysis. Performing video analysis "on the edge", i.e. next to the video source, has many benefits. The most computationally intensive tasks are distributed (and therefore parallelised) among the single camera sensors, and communication bandwidth requirements are low due to the transmission of the meta-data only. On the other hand, retro-fitting smart cameras on a large legacy system can be extremely expensive, and it may be preferable to have a centralised tracker that only analyses the subset of video streams it requires at any given time. The biggest drawback of ObjectVideo's system is that it requires overlapped cameras. Two examples in the paper show their system used for perimeter defence (with 48 cameras daisy-chained along a fence) and laboratory monitoring (with a few wide-angle cameras mounted on the ceiling). In neither case is the system expected to deal with long occlusions. Also, disambiguating a target after an occlusion is handled using only spatial information, while the system could benefit by also using appearance information.

Snidaro *et al.* in [94] employ multiple video sensors to enhance tracking. They also develop a quality function to assess the performance of each sensor for each target, which is based on foreground blob connectivity and contrast between foreground pixels and expected background colour. Target information from each sensor, weighted by the quality function, is then fused by a central process.

In their paper, information is fused coming from heterogeneous sensors, namely colour and infra-red cameras. Experimental results show how fusing tracks weighted by their quality gives better results than "blind" (unweighted) fusion. However, this method is only applicable for overlapped sensors.

In [73], Monari *et al.* present a system for human tracking across multiple, non-overlapped cameras. Monaro's tracker is similar to the one presented in this thesis, in that it combines spatial and appearance information about targets and known camera topology to track people across possibly long occlusions, e.g. on a lift. They do not, however, model the uncertainty of transition times across cameras. The main difference between the tracker presented in this thesis, and Monaro's tracker, is in system design. Monaro's tracker uses a distributed architecture, with peripheral smart sensors performing segmentation and single-camera tracking, and a remote multi-camera tracker that uses only high-level information coming from the smart sensors. In their experimental setup, 5 personal computers were used to simulate 10 smart sensors, and another 5 PCs were used to run several multi-camera trackers. By contrast, the centralised approach used for this thesis needs only one PC to track across 20 cameras, at the cost of higher bandwidth usage. The upside of a distributed design, however, is that multiple trackers can be started at very little additional computational cost, whereas with a centralised architecture a PC is required for each instance of the multi-camera tracker. No quantitative results about the performance of their tracker are provided.

Montcalm *et al.* [74] also developed a multi-camera tracker that works across non-overlapping views. For intra-camera tracking, they use target location, velocity, size, colour histogram and shape descriptor; all of these are recomputed on each frame, and compared with the previous frame to re-acquire the target. Additionally, all colour histograms and shape descriptors generated from one target in one camera are aggregated, in order to create a more robust descriptor that is used for multi-camera tracking. For each pair of cameras, a *feature transfer function* is learned (using ground truth) that can transform a feature vector acquired in one camera, into the feature vector expected in the other camera. However, experiments are shown using only one pair of cameras, and no quantitative results are given.

2.1.8 Conclusion

A review of the state of the art of multi-camera tracking was given, from a historical perspective. The review started from the origins of the concept of "tracking", went through the difficulties of tracking with an imaging device, and

eventually arrived to the current state of multi-camera tracking. The remainder of this chapter will review state of the art techniques for the lower levels of a multi-camera tracker pipeline. Firstly, a review of single-camera, image-based tracking techniques is given in Section 2.2. Then, the main techniques used to evaluate the performance of a tracker will be reviewed in Section 2.3. The subsequent two sections present a review of low-level tasks that are often performed prior to tracking, namely motion detection (Section 2.4) and camera calibration (Section 2.5). The chapter concludes with a review of metadata representation formats in Section 2.6.

2.2 Tracking With Object Model

Tracking with object model means that the tracker does not use a motion model for the target, and therefore it must rely exclusively on the target's appearance. Essentially, this means that certain features extracted from the image are reacquired from frame to frame. The common types of features used include SIFT [63] and the Mean Shift representation [23]. The latter forms the basis of the CamShift Algorithm [47]. Block-based tracking (also known as block-matching), where the features are essentially spatial templates, are also used [110, 2, 22, 49].

SIFT features are scale and rotation invariant, and provide robust matching across affine distortion, change in illumination and change in 3D viewpoint. In a tracking application, SIFT features provide reliable point correspondences between sequential frames.

The Mean Shift algorithm [28] uses a histogram (colour or greyscale) to represent the target. This representation has the advantage that it is scale invariant, and can overcome partial occlusions. It is possible to initialise the tracker using motion detection, however it is more common to allow user input for selecting the target (or region) of interest to be tracked. When a new frame arrives, the occurrence of the target histogram in the new frame is found using a gradient search. The Bhattacharyya distance is used to evaluate the closeness of match. However it is possible that this search becomes stuck at a local maximum, causing the algorithm to stagnate. In this case, a new initialisation is required. The CamShift algorithm extends the Mean Shift algorithm by allowing the histogram to adapt to the changes temporally. An open-source implementation of CamShift is available in the OpenCV library [47].

Block-matching is the process of defining a block (neighbourhood) of interest in an image, and identifying its occurrence in a subsequent frame using a systematic search, minimising a given cost function. This process is often used in

motion estimation in video encoding techniques such as MPEG-2 [25]. Although only effective in low crowding, low occlusion situations, tracking using MPEG-2 motion vectors has the advantage of very low computational cost, since most digital video is already encoded in MPEG format (or hardware encoders can be used). A prototypical system was tracking all pedestrians in a scene at 15fps using non-optimised code on an old Pentium 1GHz.

Another possible approach to tracking with object model is to use a Swarm Intelligence metaphor [5], such as in *Swarmtrack* by Antón-Canalís *et al.* Swarmtrack is based on a prey-predator scheme with a swarm of predator particles defined to track a herd of prey pixels, following a pixel intensity the way a predator would follow a prey scent track. The method includes the definition of predator particles' behaviour as a set of rules in a boids fashion (a *boid*, as defined by Reynolds [85], is an independent actor that belongs to a virtual swarm and navigates according to its own perception of the dynamic environment [62]). Object tracking behaviour emerges from the interaction of individual particles. The algorithm is efficient enough to be used for real-time vision based tasks on a general purpose computer. Unfortunately, they do not provide an implementation of their algorithm.

In their 1994 seminal paper, Shi and Tomasi [93] derive a method to extract optimal features for tracking. Each feature detected in one frame is searched for in the following frame in a neighbourhood of the feature itself. This method can be used to generate a discrete optical flow, but because it does not enforce a spatial relationship between the features, it cannot be directly used to track moving objects, as the single features will tend to be stuck onto background elements or other moving objects and spread throughout the image. The OpenCV library [47] provides an open-source implementation of this method.

2.3 Performance Evaluation

Performance evaluation of tracking is important for comparison and further development of algorithms both in academia and in industry. Performance evaluation is complicated by the fact that several flavours of tracking exist: single or multiple targets, single or multiple cameras (with or without overlap), automatic or manual initialisation, with or without operator interaction (such as restart or disambiguation), etc. Some methods stem from performance evaluation of motion detection [76, 58, 10], and are not particularly effective in evaluating tracking algorithms. Other methods, like the *cumulative matching characteristic* (CMC) curve [39], are widely used in person re-identification scenarios, and

can be used to evaluate a multi-camera tracker (in its ability to re-acquire an individual that has gone out of view), but do not evaluate the tracker's precision in locating the individual in the image. Most other methods are suitable for multi-target, automatically initialised, with single [77] or multiple overlapping [119] cameras.

Some of the most significant efforts towards the standardisation of datasets, metadata and evaluation metrics specific to tag and track (single person, manually initialised, multi-camera tracking), are the PETS [82], CLEAR [24], and TRECVid [105] programmes.

Following up these efforts, the UK Home Office started the i-LIDS programme, which includes over 30 hours of annotated, synchronised videos suitable for tag and track evaluation; to the author's knowledge, this is the largest dataset publicly available. Annotation is stored in XML, using a dialect of Viper, a metadata representation format discussed in Section 2.6.1. The programme includes a low-level network protocol for communication between the tracker and a "virtual" operation room. It does not include, however, a public performance evaluation metric.

This shortcoming has been addressed by the American National Institute for Standards and Technologies (NIST), which issued a multi-camera, single person tracking challenge [1]. The challenge is based on a subset of the i-LIDS videos, annotated by NIST (in a different dialect of Viper), and on a publicly defined set of evaluation metrics based on CLEAR [53]. These metrics, explained in detail in the following section, will be used to evaluate the tracker developed in this thesis.

2.3.1 NIST Evaluation Metrics

The performance evaluation metrics adopted by NIST for the AVSS tracking challenge were originally developed to evaluate multi-target, auto-initialised trackers. This is reflected in the metrics by including measurements of false positives (FP) and missed detections (false negatives, FN), and by the inclusion of an algorithm that computes the best mapping between ground truth objects G_i and tracker output objects D_i, that caters for false positives and missed detections (i.e. when the numbers of ground truth objects N_G and tracked objects N_D differ). The mapping can be computed on each frame, or on the whole sequence: in the following paragraphs, $N^{(t)}_{mapped}$ is the number of mapped objects in frame t, and N_{mapped} is the number of mapped objects in the sequence. All metrics presented in this section return a number between $-\infty$ and 1, where 1 means perfect tracking, 0 means that there are as many erroneous tracking

instances as there are correct ones, and a negative number means that erroneous tracking instances outnumber the correct ones.

The performance evaluation is designed for *multi-target* tracking, but the challenge was for a *single-person* tracker. In order to use this pre-existing metrics, the single-person tracking problem was cast into a multi-target tracking problem, by collecting the tracks of many single-person trackers, and considering them as the output of one multi-target tracker.

The collection of metrics described in the following subsections are explained in detail in [13]. It is to be noted that those authors seem to use *accuracy* and *precision* with a slightly different meaning from the usual. Normally *accuracy* refers to how close a measurement is to its actual (true) value, and *precision* to the degree to which further measurements show the same or similar results [101]. In [13], however, the authors take *precision* to mean how close the estimated position of the target is to the real one, and accuracy to mean how good the tracker is at preserving and individual's identity throughout the track.

2.3.1.1 Sequence Frame Detection Accuracy

The Sequence Frame Detection Accuracy (SFDA) is the average over all frames of the overlap ratio between ground truth objects and tracker output objects. The overlap ratio is defined as

$$\text{Overlap\_Ratio} = \sum_{i=1}^{N_{mapped}^{(t)}} \frac{\left|G_i^{(t)} \cap D_i^{(t)}\right|}{\left|G_i^{(t)} \cup D_i^{(t)}\right|} \tag{2.1}$$

where $N_{mapped}^{(t)}$ is the number of ground truthed objects in frame t with a corresponding tracker output object. Normalising Equation 2.1 over the number of objects we obtain the Frame Detection Accuracy (FDA):

$$\text{FDA}\,(t) = \frac{\text{Overlap\_Ratio}}{\frac{N_G^{(t)} + N_D^{(t)}}{2}}. \tag{2.2}$$

In order to measure the detection performance over the entire sequence, the FDA is computed for all frames, and normalised by the number of frames where there is at least one object:

$$\text{SFDA} = \frac{\sum_{t=1}^{N_{frames}} \text{FDA}\,(t)}{\sum_{t=1}^{N_{frames}} \exists \left(N_G^{(t)} \vee N_D^{(t)}\right)}. \tag{2.3}$$

To forgive minor localisation errors, the overlap ratio is thresholded at 20%: a tracker bounding box overlapping the ground truth by 20% or more is assigned a score of 100%. The threshold was determined empirically by having part of the data annotated by more than one human.

2.3.1.2 Multiple Object Detection Accuracy

The Multiple Object Detection Accuracy (MODA) of a frame t is computed as

$$\text{MODA}(t) = 1 - \frac{c_m(m_t) + c_f(fp_t)}{N_G^{(t)}}, \tag{2.4}$$

where m_t and fp_t are respectively the number of missed detections and false positives at frame t, and c_m and c_f are their weight functions (whose value depends on the relative importance of missed detections and false positives, but it was chosen to be 0.5 for the AVSS tracking challenge). If a measure of the accuracy over the entire sequence is required, the Normalised MODA (N-MODA) can be used. It is defined as

$$\text{N-MODA} = 1 - \frac{\sum_{t=1}^{N_{frames}} (c_m(m_t) + c_f(fp_t))}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}. \tag{2.5}$$

2.3.1.3 Multiple Object Detection Precision

The Multiple Object Detection Precision (MODP) is similar to the FDA (Eq. 2.2), but it normalises the overlap ratio (Equation 2.1) by the number of mapped objects in each frame:

$$\text{MODP}(t) = \frac{\text{Overlap\_Ratio}}{N_{mapped}^{(t)}}. \tag{2.6}$$

This gives the precision of detection in any given frame t. If $N_{mapped}^{(t)} = 0$, then the MODP is forced to zero.

2.3.1.4 Multiple Object Tracking Accuracy and Precision

The Multiple Object Tracking Accuracy (MOTA), and the Multiple Object Tracking Precision (MOTP), were developed in 2006 for the CLEAR tracking challenge [24], in order to offer a general framework for the evaluation of multi-body trackers in all domains and for all modalities (visual, radar, acoustic, etc...) [100]. The two metrics are used to calculate the two basic types of errors

made by multi-object trackers: imprecision in the estimated object locations, and failure to keep a consistent identification of tracked objects (ID swaps). A detailed explanation of how MOTA and MOTP are calculated was made available on-line to all participants of the NIST multi-camera tracking challenge at AVSS 2008, but that page is no longer available.

Multiple Object Tracking Accuracy The Multiple Object Tracking Accuracy (MOTA) is used to extract the accuracy of the tracker. It is defined as

$$\text{MOTA} = 1 - \frac{\sum_{t=1}^{N_{frames}} (m_t + fp_t + \log_{10} \text{IDC})}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}, \tag{2.7}$$

where m_t is the number of missed detections, fp_t is the number of false positives, and IDC is the number of ID changes, calculated considering the difference between the mapping for frame t and the mapping for frame $(t-1)$. The authors have not explained why a logarithm is applied to the number of ID changes.

Because MOTA is the measure of choice used by NIST for the multi-camera tracking challenge, it is also the main measure used in this thesis to evaluate the tracker in Chapter 5. Therefore, a detailed explanation of this metric will be given in Section 2.3.1.4.

Multiple Object Tracking Precision The Multiple Object Tracking Precision (MOTP) is based on the spatio-temporal overlap between the ground truth tracks and the tracker output tracks. It uses the overlap ratio (Equation 2.1), and is defined as

$$\text{MOTP} = \frac{\sum_{i=1}^{N_{mapped}} \text{Overlap\_Ratio}}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}}, \tag{2.8}$$

where N_{mapped} is the global (optimal) mapping, and $N_{mapped}^{(t)}$ is the local mapping of frame t.

2.4 Pre-Processing

All the reviewed papers that present experiments on real video data perform some pre-processing of the same before feeding them to the tracking stage, the most typical of which is motion detection. Even though the tracking stage itself is conceptually independent from the pre-processing stage, the performance of

the system is influenced by both. This work therefore had to review some of the most common techniques used to process the video data before the tracking stage.

2.4.1 Motion Detection

For advanced video surveillance systems, background subtraction is a useful tool that can allow the detection of moving objects in the scene. It requires a sufficiently accurate model of the background to enable them to be distinguished from the foreground. One of the most common methods is the Gaussian Mixture Model (GMM) [96, 97], which models each pixel as a mixture of Gaussians and uses an on-line approximation to update the model (see Figure 2.2b). This approach deals robustly with repetitive, irregular motions of background objects, such as swaying trees or waving flags. For the special case of background exhibiting regular (i.e. periodic) variations, a novel algorithm is presented in Section 4.1. One drawback of the Gaussian Mixture Model is that slow-moving objects can be mistakenly incorporated into the background. There is one parameter, the *learning constant*, that allows to specify this trade-off: high values will make the algorithm adapt quickly to background changes, at the cost of quickly incorporating slow-moving objects, whereas low values will make the algorithm more conservative, slower to adapt to background changes but also less likely to lose a slow-moving object.

There is still a growing literature on better alternatives or improvements of GMM for background estimation, e.g. the algorithms shown in Figure 2.2. However, for the purpose of this thesis, we will assume that GMM is sufficiently good to provide appropriate foreground data to a tracker.

2.4.2 Human Detection

Using motion detection for background/foreground separation assumes that anything that "moves" (i.e. any variation in pixel colour over time) is an object of interest, and vice versa that pixels whose colour never changes are not relevant for tracking. While this assumption may hold in some surveillance scenarios (with people walking against a static background), most scenes will actually feature moving objects that are not of interest (e.g. scrolling advertisements), and non-moving people (e.g. queueing at a gate or a ticket machine). Human detection algorithms have been developed to overcome this problem. A human detection system takes as input an image, and returns as output the location and size of all humans appearing in that image. While, in principle, human

detection is superior to motion detection for people tracking, in practice the computational cost of these algorithms is still too high, by at least one order of magnitude, for real-time usage.

Dalal and Triggs [33] compare several feature sets for human detection. Experimentally, they show that *histograms of oriented gradients* (HOG) outperform all previously used feature sets. HOGs are computed by subdividing an image into a large numbed of small (typically 8×8 or 16×16) cells, computing the histogram of gradient orientation for each cell (separate by channel and weighted by gradient intensity). The set of HOGs over a detection window forms the descriptor that can be used for detection. A *support vector machine* (SVM) [31] is trained with positive and negative samples, and it is used to classify the HOGs inside a detection window as either human or non-human. The algorithm was evaluated against the publicly available INRIA dataset[1], and performance is reported to be between 84% and 89% true positives at 10^{-4} FPPW (false positives per detection window). The author of this thesis implemented in MATLAB a HOG detector based on this paper, but the best performance achieved was of several seconds per frame.

Schwartz *et al.* [91] augment widely used edge-based detectors (such as the HOG-based detector reviewed above) to include colour and texture information. The descriptor so obtained has a very high dimensionality (in the order of 10^5), which makes it intractable for any standard machine learning algorithm such as SVM. To overcome this problem, a dimensionality reduction algorithm called *partial least squares* (PLS) [87] is applied to the descriptors, reducing the number of dimensions from over 100,000 to 20. Performance evaluation on the same INRIA dataset show a true positive rate of 94% at 10^{-5} FPPW. Even though some thought is also given to computational costs, the reported speed is still too slow for real-time usage.

2.4.3 Shadow Removal

Shadows and reflections cause many false positives in foreground detection. Shadow removal techniques have been developed to limit this problem, even though deep shadows are very difficult to remove. Some of the most commonly used algorithms are Hoprasert [45] (Figure 2.2d), Cucchiara [32] and double mixture model [71] (Figure 2.2c). All these methods exploit the property that cast shadows only change the intensity of a pixel, not its chromaticity, while a

[1] This dataset can be downloaded from `http://lear.inrialpes.fr/data` and used for research purposes.

Figure 2.1 – *Example of shadow detection and removal (taken from [45]).* Upper left: *background model;* upper right: *input frame;* lower left: *output from background subtraction (foreground is blue, shadows are red, highlights are green and background pixel are kept in the original colour);* lower right: *foreground region after shadow removal is performed.*

real foreground object usually changes the chromaticity of a pixel as well as its intensity.

Detecting reflections is a much harder task, since they also modify pixels chromaticity. Several methods exist, such as [92], but they are too computationally intensive to be used in real-time video analysis without dedicated hardware.

2.5 Camera Calibration

Camera calibration is the process of finding the parameters of the perspective projection that maps a point from the world reference frame to the image reference frame. The projection can also be inverted to get a mapping from a point in the image reference frame to a line in the world reference frame. Camera calibration can improve single-camera tracking by providing hints on expected target size and speed as a function of their position on the image.

2.5.1 Single Cameras

A perspective projection has 11 or more parameters [34]: 6 are the extrinsic parameters, describing a rigid transformation from the world to the camera reference frame; 5 are the intrinsic parameters, describing the perspective projection from the 3D camera reference frame to the 2D image reference frame. Figure 2.3 shows a schematic representation of the extrinsic parameters (the transformation from world to camera reference frame) and the intrinsic parameters (the projection onto the image plane). More parameters can be used to

(a) *Input frame*

(b) *Mixture of Gaussian*

(c) *Double mixture model*

(d) *Horprasert*

(e) *Li and Huangs [60]*

Figure 2.2 – *Motion detection algorithms. All the algorithms were run with the default parameters suggested in their respective papers.*

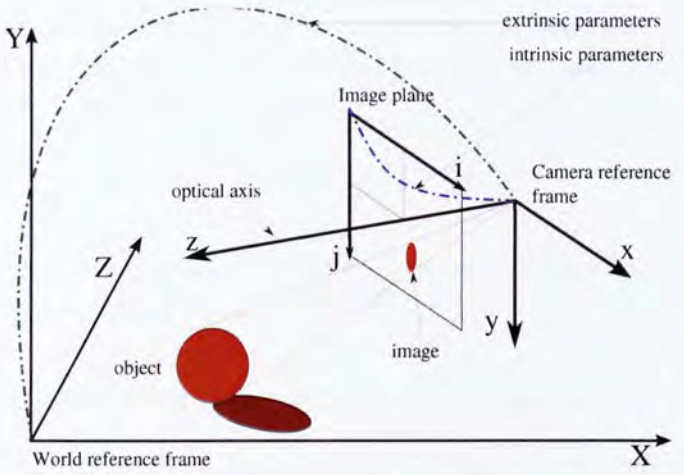


Figure 2.3 – *Camera calibration parameters. The green dash-dotted line ("extrinsic parameters") represents the position of the camera reference frame in the world reference frame. The blue dash-dotted line (intrinsic parameters) represents the position of the image plane in the camera reference frame.*

compensate for non-linear distortions (i.e. radial distortion, not shown in the figure).

Usually, video-surveillance systems do no require all these parameters. The 6 extrinsic parameters do not need to be fully estimated for a single camera system, since in this case only the equation of the ground plane in the camera reference frame is needed, thus leaving 3 parameters. Of the 5 intrinsic parameters, skew and aspect ratio can safely be assumed to be 0 and 1 respectively, leaving only the position of the principal point and the focal length to be estimated. This leaves a total of 6 free parameters, which can be estimated with any of the manual methods presented in [34], or using the (semi-)automated methods described below.

Camera Calibration from Detected Tracks

Renno *et al.* [84] developed an auto-calibration procedure to recover the image-to-ground plane homography by accumulating tracks. A homography is a transformation that maps points lying on a plane from one camera viewpoint to another, or to and from the ground plane. The parameters of this transformation are the optical centre (i_0, j_0), the focal lengths (α_x^f, α_y^f), and the pitch angle θ. A 2D histogram $H[i, \mu]$, where i is the vertical coordinate of the target in the image, and μ is the target height in pixels, is created from a large number of detected moving regions. This histogram shows that the two variables relate linearly to each other. Two key positions of a projected object are at the

horizon i_h and at the height of the optical centre i_0. The horizon i_h can be extrapolated as the position where all targets are $0px$ high, while i_0 has to be computed from the optical flow while the camera is zooming. The pitch angle can then be computed as

$$\tan^{-1}\theta = \alpha_y^f(i_h - i_0).$$

An estimate of a target's height in metres requires at least one target of known height to be used as a reference.

Camera Calibration from Head-to-Feet Homography

In this method developed by Krahnstoever and Mendonça [55], people are modelled as parallel segments of approximately the same length, all perpendicular to the ground plane. These parallel segments intersect at a vanishing point. All lines connecting the upper points of pairs of segments and lower points of the same pair are parallel and intersect at the horizon. The vanishing point and the horizon define the camera intrinsic parameters. However, there are so many sources of noise (i.e. people do not have all the same height) that a careful Bayesian formulation of the problem is required. This is the only calibration method, among the ones reviewed here, that can deal with non-zero roll angles.

Camera Calibration from Optical Flow

Camera calibration from optical flow was developed by Velastin *et al.* [109]. It assumes that people move at similar speeds independently of their distance from the camera, and it estimates the camera focal length f, height L from the ground plane, and tilt angle θ. Let (x, y, z) be the camera reference frame (centred in the image plane, with the z-axis pointing outwards), and (x'', y'', z'') the world reference frame, with y'' being the vertical axis. All moving objects in the scene are expected to move parallel to the (x'', z'') plane (the ground plane), thus having null vertical speed ($v_{y''} = 0$). The y component of an object's position (in the camera reference frame) can therefore be written as a function of the z component:

$$y = (z - z_0)\tan\theta = \frac{z\sin\theta - L}{\cos\theta}.$$

Thus, the projection y' on the image plane of the y component is given by

$$y' = \frac{fy}{f - z} = \frac{f\cos\theta}{\sin\theta} - \frac{fL}{(f - z)\sin\theta}.$$

The measured motion on the image plane $(v_{x'}, v_{y'})$ represents the projection of scene motion $(v_{x''}, v_{y''}, v_{z''})$ onto the camera reference frame $(x, y\, z)$ *prior* to being projected onto the image plane. it can be shown that $v_x = v_{x''}$ and $v_y = v_{y''} \cos\theta + v_{z''} \sin\theta$. By applying a perspective projection of focal length f to these equations we obtain

$$(v_{x'}, v_{y'}) = \left(\frac{f v_{x''}}{f - z}, \frac{f\left(v_{y''} \cos\theta + v_{z''} \sin\theta\right)}{f - z} \right),$$

but since we assumed that $v_{y''} = 0$ this equation can be simplified as

$$(v_{x'}, v_{y'}) = \left(\frac{f v_{x''}}{f - z}, \frac{f v_{z''} \sin\theta}{f - z} \right).$$

Assuming $z \gg f$ (i.e. the distances of a person's head and feet from the camera are roughly the same), the velocity of the object on the image plane will be inversely proportional to the object depth (z-coordinate). If optical flow is computed and averaged for a sufficiently long period of time, then the value of its horizontal component will be proportional to the y_{img} coordinate at which it is computed. This proportionality can be used to estimate the image-to-ground plane homography. However, the assumption that $z \gg f$ is too constraining for an indoor surveillance scenario, especially in underground stations, where the height of the ceiling severely limits the height at which cameras can be placed. Also, the algorithm uses large quantities of noisy data to estimate the calibration, and while the authors recognise that a good stochastic model is required, none is proposed.

2.5.2 Camera Networks

The purpose of multi-camera calibration is to establish a geometric or topological relation between multiple cameras in a surveillance network.

A topological relation is an oriented graph of connections between cameras. Nodes of the graph represent cameras, and edges represent a relation of adjacency between cameras. In this context, two cameras a and b are adjacent either if their views are overlapped, or if they are not overlapped and an individual can move from the view of a to the view of b without having to pass through the view of any other cameras (but the fact that two cameras are adjacent does not preclude the existence of a longer, indirect path from a to b, see Figure 2.4). Often, the edges are extended with average transition times, transition probabilities, and transition regions (the area of the image where a transition occurs).

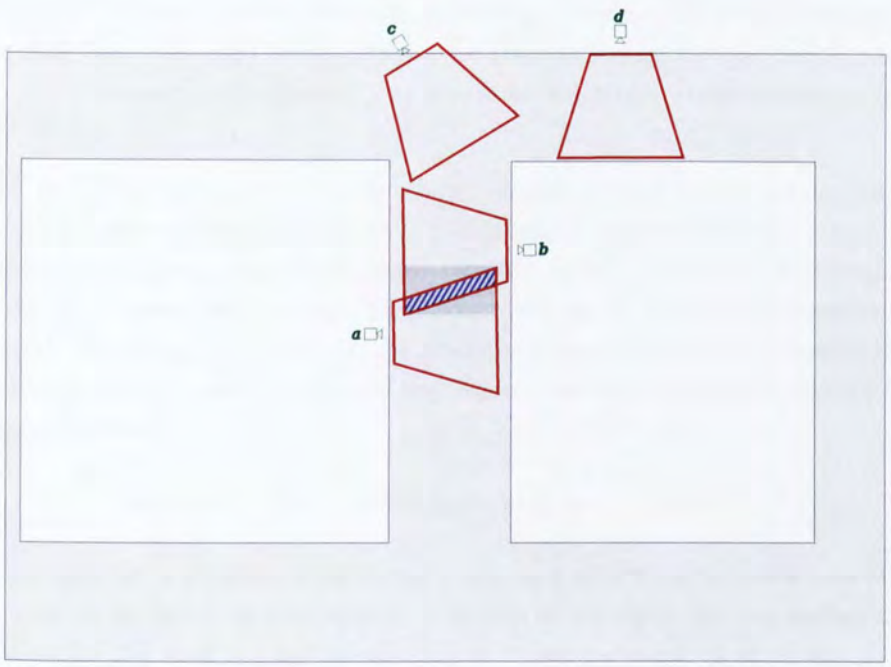


Figure 2.4 – *Different types of adjacencies in a camera network of a simplified surveillance scenario (corridors are in gray). Cameras* a *and* b *are adjacent and overlapped (overlapping area is marked with blue stripes); cameras* b *and* c *are adjacent but not overlapped; cameras* b *and* d *are not adjacent, because any passenger walking between these two cameras will have to pass by the field of view of* a *or* c*. Cameras* a *and* d *are adjacent, even though between the two of them there is a long occlusion.*

A geometric relation is a mapping from each camera reference frame to one or more other camera reference frames, or to a common world reference frame. If two adjacent cameras are close enough, geometric calibration allows a tracker to predict a target's motion from the first to the second, and, if the views of the two cameras overlap, targets in the overlapping zone can be tracked more easily, as information coming from more than one sensor can be combined.

Geometric Calibration

Geometric relations can be inferred by collecting a large number of single-camera tracking sequences, and then searching for pairs of sequences that correspond up to an unknown homography. This is called a *tracking correspondence model* (TCM).

The first fully automatic TCM for large camera networks with overlapping fields of view was introduced by [98]. Given only a large set of video streams from a surveillance area, it can automatically build a model of the camera network. A tracking sequence S_i is comprised of N_i tracking observations $\{s_i(t_0), \ldots, s_i(t_{N_i})\}$, indexed by the absolute times they occurred. Each observation includes a description of the object from a particular camera at a particular time:

$$s_i(t) = \{x(t),\ y(t),\ d_x(t),\ d_y(t),\ s(t),\ \text{image}(t),\ \ldots\}\ .$$

Any two tracking sequences that belong to the same object over the same interval of time are in *direct* correspondence. The goal of correspondence modelling is to estimate the ideal correspondence matrix Γ^*, each element of which is

$$\gamma_{ij}^* = \begin{cases} 1 & \text{if } S_i \text{ and } S_j \text{ correpond to the same object,} \\ 0 & \text{otherwise.} \end{cases}$$

This is accomplished by finding the maximum likelihood correspondence assignment given the observations and the TCM. The probability that two sequences are equivalent given the observations and the TCM is

$$p\left(\gamma_{ij}^* = 1 \mid S_i,\ S_j,\ \text{TCM}\right)$$

and can be determined by maximising the likelihood of

$$p\left(\gamma_{ij}^* = 1 \mid S_1, \ldots,\ S_j,\ \text{TCM}\right) = \prod_{S_i, S_j \in \mathbf{S}} p\left(\gamma_{ij}^* = 1 \mid S_i,\ S_j,\ \text{TCM}_{ab}\right)^{\gamma_{ij}^*}\ .$$

This model can be used to track objects through the environment. Assuming that all objects move on a common ground-plane, the position of an object that is visible in two cameras a and b is related by a 3×3 homography H_{ab}:

$$H_{ab}p_a(t) \cong \hat{p}_b(t)$$

where $p_a(t) = (x, y, 1)$ is the location of the object, in homogeneous coordinates, in camera a at time t, and $\hat{p}_b(t)$ is the interpolated position of the tracked object in camera b at the same time. The TCM of a pair of cameras includes the homography and a visual occlusion model R_{ab} which is an estimate of the region of overlap between the cameras:

$$\text{TCM}_{ab} = \{H_{ab}, R_{ab}\} .$$

To estimate the homography, pairs of co-occurring tracks are randomly sampled whose likelihood of belonging to the same object is given by

$$L_{ij} \propto P_c P_t P_m$$

where P_c is the probability of this particular pair being valid (given the number of other objects that were visible at the same time), P_t is the probability of a corresponding track lasting a particular time interval, and P_m is the probability of matching S_i to S_j directly. For each pair of tracks, a homography is estimated, and a score is computed based on the number of corresponding tracking sequences; the one with the maximum score is chosen.

In [15], the Least Median of Squares (LMS) method presented in [99] is used to determine the homography that aligns the object centroids detected in each camera view. The coefficients of the homography can be computed with 4 correspondence points (see [30]); singular value decomposition (SVD) can be used to compute the LMS.

Topological Multi-Camera Calibration

All methods reviewed here work by collecting a list of "object" entry and exit events in the images (where the objects are typically pedestrians or vehicles, depending on the application). This list is used to find a correlation between the entry time of an object in a scene and the exit time of all other objects in all other scenes: if a correlation is found between a pair of objects, they are assumed to be the same, and a topological link is assumed to exist between the two scenes. Such methods are capable of autonomously creating the topological

network of a surveillance system. Also, overlapping views can be thought of a special case of non-overlapping views, having negative transition times.

In [70] the problem of topology is formalised in terms of the inference of a weighted directed graph which captures the connectivity relationships between the positions of the cameras. The notion of multiple agents moving asynchronously through a camera network can be modelled as a Markov process. The network is described as a directed graph $G = \langle V, E \rangle$, where the vertices $V = v_i$ represent the sensor locations, and the edges $E = e_{ij}$ represent the connectivity between them. The motion of N agents in this graph can be modelled as transitions across edges. Let $O = \{o_1, \ldots, o_{N_O}\}$ be a set of events detected at times $t = 1 \ldots N_O$ from the various sensors (vertices of the graph) , which indicate the likely presence of one of the N agents in that position at that time. Given the observations O and the number of agents N, the authors [70] propose a Monte Carlo Expectation Maximisation algorithm for estimating E.

In [67] a similar method is proposed, with the addition of automatically learned set of entry and exit zones in each camera view. Connections are established between entry and exit zones of cameras, instead of between the cameras themselves. The method was used to accurately reconstruct the topology of a 6-cameras network using 13 hours of video.

2.5.3 Colour Calibration and Correction

To track targets as they move through a network of cameras, appearance descriptors are used to model their appearances. Descriptors can be compared to calculate a probability that two targets represent the same individual. For this comparison to be as effective as possible, any systematic difference in descriptions obtained from two cameras should be identified and removed. Such variations may be caused by differences in the cameras (different hardware and configuration), or by differences in the scene (different illumination).

Colour correction methods can be divided in two classes: supervised (manual) or unsupervised (automatic). Supervised methods such as [46] require the user to select, from pairs of cameras, pixels or areas which are known to have the same colour. The advantage of supervised methods is that they are generally more reliable and accurate (assuming the user inputs correct data), but they are not practical when the number of cameras is large.

Unsupervised methods, on the other hand, make some hypotheses about illumination in the scene, and use them to automatically normalise colours across cameras. Typical hypotheses include assuming that the average reflectance of

all scene elements is grey (the *grey world* hypothesis [19]) or that the brightest pixel of every image represents white. The advantage of an unsupervised method is that it can be applied to an arbitrarily large number of cameras; the main disadvantage is that if the hypothesis is not met, the method will fail to provide a good calibration and may actually make colours less consistent across cameras.

2.6 Meta-data Representation

The word "metadata" is a compound of the Greek μετά (*meta*, meaning "besides", "over" and "above"), and Latin *data* (plural of *datum*, meaning "given"). Loosely, it can be defined as data about data. For example, the traditional card catalogue of a library is metadata of the library (where books are considered the actual data). In Computer Vision, algorithms work on video data, and metadata is anything that describes a video stream (for instance, it could be the time and date the video was captured, or a textual description of the weather, or the presence of an object of interest in a given location at a given time).

Metadata can be used as additional input for a Computer Vision algorithm, or be produced as output by it. For example, the "alert" status of an anomaly detector, or the target position for a tracker, are metadata produced by a detection or by a tracking algorithm. The initial position of a target is metadata provided as input to a tracker, in order to initialise the tracking process. Metadata is usually much more compact that the corresponding video (*raw*) data, and can be stored in files or databases. It can be compared against a ground truth (which is also a form of metadata) in order to get a performance measure of the algorithm generating it.

By far the most common transport format for metadata is the ***E**xtensible **M**ark-up **L**anguage* (XML) [113]. XML is a general-purpose *specification* for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to help information systems share structured data, and it is used both to encode documents and to serialise data. It is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. There are two levels of correctness of an XML document:

Well-formed A well-formed document conforms to all of XML syntax rules (e.g. if a start tag appears without a corresponding end tag, it is not well-formed). A document that is not well-formed is not considered to be XML and a conforming parser is not allowed to process it.

Valid A valid document additionally conforms to some semantic rules. These rules are either user-defined, or included as an *XML schema*. For example, if a document contains an undefined element, then it is not valid; a *validating parser* is not allowed to process it.

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type. An XML schema provides a view of the document type at a relatively high level of abstraction. XML is an international, widely adopted, fee-free standard recommended by the World Wide Web Consortium (W3C). A large number of XML-related libraries, tools and documents is available on the Internet.

XML subsets can be defined using the *xsd schema* specification, which is itself an XML file. Schemas allow programmers to declare the structure of an XML document, i.e. to formally specify which nodes (elements, attributes or text) are allowed in which part of the document, whether they are optional, mandatory or forbidden, and what values they can be assigned.

Based on a human readable text format, supported by many high quality libraries and tools, and backed by vast industrial support, it comes as no surprise that XML has been adopted by many members of the Computer Vision community for metadata representation. The following sections review some of the Computer Vision-specific schemas that have been developed.

2.6.1 Viper

The Viper schema was designed for the ViperGT ground truthing tool [56] in order to represent metadata for any Computer Vision task. The complete schema is split in two files: one describes the overall structure of a Viper file, while the other one defines basic data-types (e.g. bounding boxes, polygons, etc.). The schema is extensible, i.e. it is designed so that user-defined data-types can be added in a backwards compatible manner. The trade-off of this flexibility is an increased complexity in the usage of Viper files. A "quirk" of the Viper schema is that it is object-based rather than frame- or time-based. This means that the XML cannot be output "live" as the algorithm runs, but it has to be stored in memory until the algorithm completes before it can be dumped to file.

Its quirks notwithstanding, the Viper format was adopted by the UK Home Office and by the American National Institute for Standards and Technologies (NIST) for the ground truthing of the i-LIDS multi-camera tracking dataset.

2.6.2 Serket

Serket was specifically designed by the Serket EU Project Consortium to describe metadata for Computer Vision algorithms [4]. The Serket schema cannot be extended with user-defined data-types, thus sacrificing some flexibility to favour ease of use. Even though it cannot be extended, it seems to be complete enough for most video-surveillance metadata. It is frame-based, has support for multiple cameras, it defines bounding boxes for targets, regions in the image- or ground-plane, events, etc. However, the author of this thesis was unable to locate on the Internet the actual schema definition, and no recent publication was found that uses Serket for metadata interchange.

2.6.3 SMAF

The Surveillance Media Application Format (SMAF) [3] is a proposed *restriction* to the MPEG-7 standard for video surveillance applications. MPEG-7 provides a general purpose framework for associating metadata to multimedia data [48], and allows application-specific *restrictions* to be defined for specific domains. Any such restriction is referred to as a Multimedia Application Format (MAF).

The restriction defined by SMAF covers a description of the surveillance system and of the activity in the scene. In addition to this set, appropriate descriptions for the relation between camera and scene are also considered. To improve interoperability between systems and between components of a system, two types of restrictions are proposed. The first proposal is a restricted subset of the MPEG-7 elements that are applicable to the surveillance domain. The second proposal is to use the MPEG-7 tools to include domain-specific taxonomies to restrict the names of elements used in the semantic descriptions.

At the time of writing, however, SMAF has not yet been officially adopted in the MPEG standard.

2.6.4 PETS

In 2001, the second International Workshop on Performance Evaluation for Tracking and Surveillance (PETS) [82] designed a schema for the representation of tracking and high-level behaviour analysis. In contrast to Viper and SMAF, the PETS schema focuses on tracking and behaviour analysis only, and is not designed to represent any possible type of surveillance metadata. This lack of extensibility makes the format easier to use, but also limits its scope to

the two tasks it was designed for. To the best of the author's knowledge, the PETS format has not been used outside of the PETS workshop.

2.6.5 Agent-Based

This system was developed at the Multitel research centre in Mons, Belgium, to address the need for a generic, context-independent and adaptive system for storing and managing video analysis results [20]. The system is based on a schema-independent data warehouse backed by a multi-agent system. Each agent is either a communication agent, representing a data-flow between the processing algorithm and the data warehouse, or a data agent, representing the knowledge contained in the XML produced by a processing algorithm. A data agent can represent any XML document, and therefore the data warehouse is not linked to any specific schema (although a schema must be provided at system initialisation).

Schema-independence makes this format suitable for any Computer Vision application (and, indeed, it is not limited to Computer Vision at all), and at the same time keep it simple, since an ad-hoc schema can be specified for any application. Graphical tools are provided for querying the data warehouse, collecting long-term statistics and discovering trends. The downside is that this format requires a server running 24/7 to host the data warehouse, which is a single point of failure and, depending on network load and the amount of metadata produced, may become a bottleneck. The agent-based format is the metadata format adopted for the CARETAKER project, where running a server 24/7 was not an issue, and where the amount of metadata produced by the processing algorithms proved to be manageable.

2.7 Conclusion

This chapter reviewed the state of the art of tracking, starting from a historical perspective, with particular focus on visual tracking over multiple cameras. The section on existing tag-and-track systems, sadly small, shows that there is plenty of space to research a real-time, single-target, multi-camera tracker, since no reviewed system fulfils these criteria (especially with respect to the real-time constraint). The review also covered the state of the art of motion detection, colour correction, camera calibration, and metadata representation. These topics were analysed since they often play a role in various stages of a TNT pipeline. In fact, the number of options available for each stage of the TNT pipeline suggests to keep the tracker design modular, so that new algorithms and formats

can be easily added at a later stage of development if it becomes desirable. The next chapter presents the main contribution of this thesis, namely a probabilistic framework that allows the techniques reviewed in this chapter to be combined in order to produce a tag-and-track system for multiple non-overlapped cameras. Two new pre-processing algorithms, one for motion detection, and one for colour calibration, are presented in Chapter 4. The effectiveness of the framework will be assessed in Chapter 5.

Chapter 3

Tracking using Multiple Cameras

This chapter presents the multi-camera tracking framework, which is the main contribution of this thesis. Starting with an overview of the problem and with some definitions that disambiguate the meaning of common terms, the chapter will then introduce the theoretical foundation of the proposed framework. Subsequent sections will analyse in more detail how it is suggested that the framework be modularised, and will provide several proposed implementations for each module.

3.1 Overview

Within the scope of this thesis, a *tracker* is defined as an algorithm that accepts as input a target i_{t-1} (observed at time $t-1$) and a video frame captured at time t, and gives as output a target i_t (observed at time t) representing the same individual. It can also be noted that a tracker may produce an output even if the target is completely occluded or out of view. In this case, the output is a prediction of the target's position.

Although a multitude of approaches to tracking exist, this definition was chosen based on the end-user's requirement of a single-target, real-time tracker with minimal latency. Therefore, each frame is processed as it becomes available, and an output is produced before the following frame arrives. It should be noted that this decision does not exclude the possibility of using other, more sophisticated trackers, for instance capable of handling multiple tracking hypotheses and to provide a confidence estimate for each hypothesis. The definition of *tracker* given above was chosen because it is the simplest definition that can capture

the user requirements. Moreover, because the goal of this thesis is to develop a multi-camera tracking *framework*, it should always be possible to incorporate e.g. a multi-hypotheses tracker, as long as it can be cast to a single-hypothesis tracker (by considering only the most likely hypothesis) and as long the real-time constraint is respected.

In the case of *single-camera* tracking, there is only one input video and a target is completely defined by a binary mask specifying what pixels in an image belong to it. A review of existing, single-camera trackers is available in Section 2.2. In the case of *multi-camera* tracking, there are multiple videos and a target needs to be associated with the one(s) in which it was observed. A multi-camera tracker may be implemented by combining several single-camera trackers and fusing their output, but that is not the only option. In particular, in this work, it was decided to have only one single-camera tracker running at a time, because of the following practical considerations. Firstly, overlapping cameras in the available data-sets are rare, and the overlapping area is small, therefore the possibilities of fusing outputs from more trackers would be rare as well; small to no camera overlap is common in video-surveillance scenarios, where the main concern is to maximise coverage of the area. Secondly, camera calibration is often good only in a limited area of a camera view, therefore even if there is some overlap between two cameras, this area may be poorly calibrated in one of them (*e.g.* because it is far away): it is better to use only information from only one (well calibrated) area than including noisy data from another tracker. Thirdly, since processing multiple video streams in real-time can be computationally expensive (the final implementation can process about 6 streams at 5 FPS on a dual-Athlon 2.4GHz), it was decided to save CPU power by not running more than one single-camera tracker at a time.

Within this work, the multi-camera tracking framework combines information from one single-camera tracker, one or more descriptors, and prior information coming from the camera network layout. Before proceeding any further, precise definitions of some of the terms introduced above (i.e. target, individual, descriptor, etc.) will be given.

3.2 Definitions

The layout of a network of cameras is a weighted, directed graph where each node corresponds to a camera, and there is a link from node a to node b if either a and b have overlapping views, or they do not and an individual viewed in camera a can transition to camera b without passing by any other camera. When two

Figure 3.1 – *Targets and individuals. The map of the underground station in the background represents the real world, where several individuals are present (to avoid cluttering the image, only one individual $\hat{i}$ is shown, wearing a yellow top). The real world is imaged by three cameras (inset left, centre and right). In each camera, there is a target $i_{1,2,3}$ (highlighted with a read bounding box) corresponding to this individual, plus other targets corresponding to other individuals. Because all targets i_n correspond to the same individual, we can write $i_1 \equiv i_2 \equiv i_3$.*

nodes in the topology graph are linked, we say that their corresponding cameras are *adjacent* (see Section 2.5.2). The weights on the edges are the minimum time required for the transition (this time could be negative if the cameras have overlapping views).

The graph is directed because not all adjacencies are symmetric. A typical case of an asymmetric adjacency arises when two cameras are placed at the two ends of an escalator. In this case, people moving in the direction of the escalator will complete the transition in a shorter time than people moving in the opposite direction (indeed, this latter case will be extremely rare, but it is physically possible and therefore must be accounted for by the topology). An adjacency may even be unidirectional: this is the case, for example, for cameras placed along the platform in an underground station. Topologically, these cameras are adjacent to similarly placed cameras on the next station along the line, but a transition may only happen in the direction travelled by the train. Finally, there is the case of two cameras placed nearby, such that the difference between average transition times in the two directions is not statistically significant, in which case we say that the adjacency is symmetric. Even though the latter case is the most common, it is considered here only as a special case.

A *target* i is a region of an image corresponding to an *individual* $\hat{i}$ in the scene. Two different targets i and j may correspond to the same individual seen from different cameras or at different times (in this case, we write $i \equiv j$, meaning that the two targets have the same *identity*, see Figure 3.1), and two different

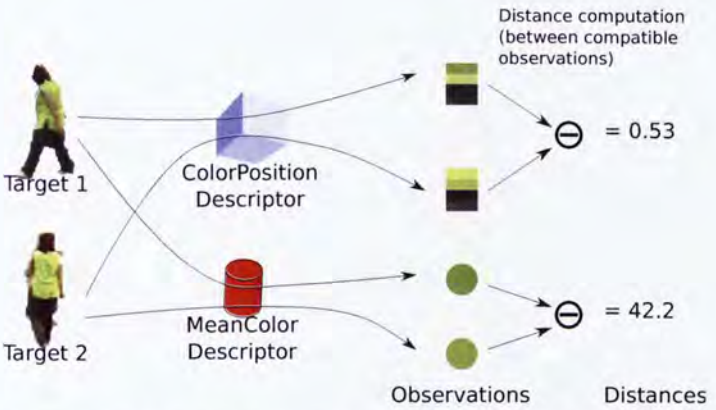


Figure 3.2 – *The figure shows two targets and two appearance descriptors (in this instance, the targets represent the same individual, but this need not be the case). The descriptors are applied to the targets to obtain observations. Observations produced by the same descriptor are comparable, in that a comparison function (here represented with a circled "minus") can be applied to them. The result of this application is a real number (shown here are two arbitrary values). N.B.: the difference in colours between observations is subtle, and may become unnoticeable in low-quality printing.*

individuals may appear as a single target, for example if they are very close to each other. *Tagging* is the action performed by the operator when they choose a target to track by selecting a rectangular region on a frame of a surveillance video. A *tracking error* happens when a tracker outputs a target j which does not correspond to the same individual as its input i. If a tracking error can be detected, the erroneous tracker can be reinitialised, either manually or automatically, to track a different target. When this process is performed automatically, and the new target is in a different camera, we say that a *handover* is performed, since, from an end-user perspective, the target is handed over to another camera. A handover may also take place when, even in the absence of tracking errors, another target j' is found that corresponds to the tagged individual and that, according to some criteria, is more suitable for tracking (these criteria are analysed in detail in Section 3.4).

A *descriptor* D can be applied to a target i to generate an *observation* D_i (see Figure 3.2). An observation is a compact representation of some features of the target (appearance, motion, position, etc.). If two observations D_i and D_j are generated by the same descriptor they are said to be of the same type, and a comparison function $c(D_i, D_j)$ can be applied to them. The result of this comparison is a quantity D_{ij} that represents a *similarity* between the two observations. In more rigorous terms, the comparison function c defines a semi-metric on the set of observations generated by the same descriptor. For a function to define a semi-metric, the following conditions must be satisfied [54]:

1. Positive definiteness, *i.e.* $c(D_i, D_j)$ is non-negative, and it is zero if and only if two observations are identical:
   - $c(D_i, D_j) \geq 0$,
   - $c(D_i, D_j) = 0 \iff D_i = D_j$.
2. Symmetry: $c(D_i, D_j) = c(D_j, D_i)$.

For some descriptors, it is easy to provide a comparison function that also satisfies a third condition:

3. Triangle inequality, also called *subadditivity*: given three descriptors D_i, D_j and D_k, $c(D_i, D_k) \leq c(D_i, D_j) + c(D_j, D_k)$;

thus making the descriptor space a metric space, where the intuitive notions about the concept of distance are valid. For example, the triangle inequality means that the distance "traversed" directly between D_i and D_k is not larger than the distance to traverse in going first from D_i to D_j, and then from D_j to D_k. However, because subadditivity will never be used in subsequent parts of this thesis, the comparison function c is only required to satisfy the conditions of a semi-metric.

Note that when the distance between two observation is zero, it does not necessarily mean that the two targets have exactly the same on-screen appearance, nor that the two observed targets represent the same individual. It just means that the descriptor generating the observations is unable to distinguish one target from the other:

$$c(D_i, D_j) = 0 \not\Rightarrow i \equiv j\,.$$

3.3 Theoretical Basis

Tracking can be thought of as the process of estimating the probability that two targets, acquired at different instants in time, correspond to the same individual. Typically, the two targets may be acquired diachronically at two consecutive frames of the same video stream, or isochronically on a set of streams. *Bayesian Recursive estimation* is a general probabilistic approach for estimating an unknown *probability density function* (pdf) recursively over time using incoming measurements and a mathematical process model, when the unknown pdf is the unobserved state $\mathbf{x}$ of a Markov process:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots, \mathbf{x}_0) = p(\mathbf{x}_t|\mathbf{x}_{t-1})\,. \tag{3.1}$$

In Equation 3.1, the equality holds because of the Markov assumption. Some measurements $\mathbf{z}$ of the Markov process are available, and the measurement at time t depends only upon the current state:

$$p(z_t|x_{t-1}, x_{t-2}, \ldots, x_0) = p(z_t|x_{t-1}) \ . \tag{3.2}$$

The following sections will explain how the Bayesian Recursive estimation model can be applied to a tracker in order to get an estimate of the probability that two targets correspond to the same individual (the Markov process state), given a set of observations (the measurements):

$$p\left(i \equiv j|D_{ij}^{(i\ldots n)}\right) \ , \tag{3.3}$$

where D_{ij} is the distance between the two observations D_i and D_j. In the context of this work, measurements correspond to distances between observations of different targets. Observations are generated by descriptors, which can be divided up in two sets: appearance and motion. Appearance descriptors encode a target's appearance in a compact way, and will be described in Section 3.7. Motions descriptors estimate the dynamic state of an individual (typically, position and velocity) and will be described in Section 3.8. A corollary of this difference is that motion descriptors need to be continuously fed with their target position in order to update the model, so that they can generate valid observations; on the other hand, appearance descriptors assume that an individual has constant appearance, and therefore do not need to be given any information other than the current target's pixels. The assumption of constant appearance may not hold in all cases, for example when a person clothes have radically different colours on the front and back sides. However, for most real cases, the difference in appearance due to the point of view is small compared to the appearance of different people. The following section explains how appearance and motion descriptors can be combined in order to maximise the probability that the current tracked target corresponds to the same individual as the tagged target.

3.4 Combining Cues

Tagging identifies a reference target r at time t_0 for which an appearance observation A_r is immediately generated, and starts a single-camera tracker to track r so that a motion descriptor M_r is available. The appearance observation A_r will be used throughout the tracking process as a reference appearance, since r is selected by the operator and can therefore be safely assumed to be correct.

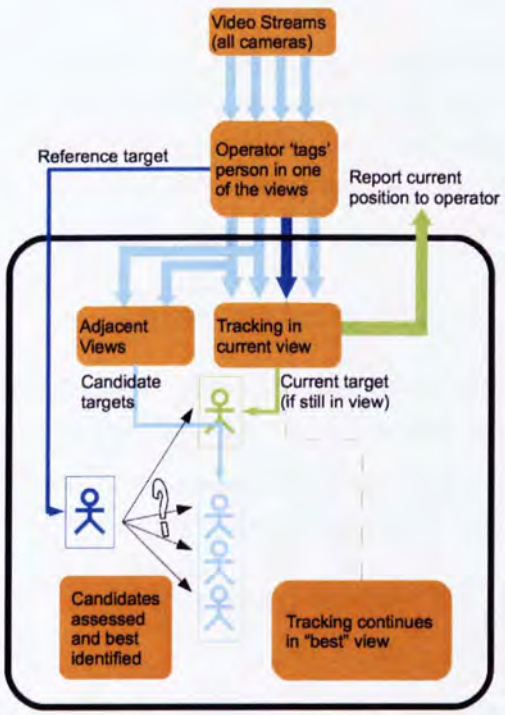


Figure 3.3 – *Tracker workflow. A security operator initialises the tracker by tagging a target in a video stream (known as the* current *stream). A single-camera tracker is started to follow the target in this view. At the same time, targets in adjacent views are assessed for handover, by comparing their motion observations with the current target's, and their appearance observations with the reference target's (as detailed by equations 3.4, 3.5, and 3.6).*

In each subsequent frame, the single camera tracker will identify a new target i that (ideally) represents the same individual as r. Also, in each subsequent frame *all* targets other than i are taken into account as candidates for *handover*, i.e. for resetting the single camera tracker and assigning a new target to it (see Figure 3.3). Let $\mathbf{C} = \{j \,|\, j \neq i\}$ be the set of candidate targets, i.e. all targets other than i. At some point in time, the individual being tracked is likely to go out of view, or to be better visible in a different camera; in this case, the tracker should perform a handover so as to get the best view of the individual, and avoid losing it. If there exists a j such that it has the same identity as r and appears to be a better choice for tracking, a handover is performed and j becomes the new current target.

A target j is considered a better choice than the current target i when all the following conditions are true:

1. j is close to i:

$$p(i \equiv j|M_{ij}) > \tau_M \tag{3.4}$$

where M_{ij} is the distance between the observations of the current target i and the candidate j generated by the motion descriptor M, and τ_M the threshold on motion matching, i.e. the minimum probability required to consider i and j the same person using motion only;

2. j is more similar to r than i is:

$$p(j \equiv r|A_{jr}) > p(i \equiv r|A_{ir}) \tag{3.5}$$

where A_{jr} is the distance between the observations of the candidate j and the reference target r generated by the appearance descriptor A;

3. the joint probability of j corresponding to the same individual as r (given their appearance) and as i (given their position) is high enough:

$$p(j \equiv r|A_{jr})\, p(j \equiv i|M_{ji}) > \tau \tag{3.6}$$

where τ is a probabilistic threshold, i.e. the minimum joint probability required in order to consider j, i and r the same individual, according to both appearance and motion.

If more targets are found that satisfy all conditions, the handover is performed using the one that maximises condition 3. Note that appearance observations are always compared with the reference observation, A_r, since it is the only one known to be correct. Colour constancy techniques should be used to accommodate for colour changes due to different lighting conditions in different camera views. Motion information cannot be compared with a reference observation, since motion prediction cannot yield meaningful results for an arbitrarily long time. Motion descriptors have a motion model that can be used to predict the state the reference target r (acquired at time t_r) will have in a future time t_i (when target i is acquired). As the difference between t_r and t_i grows, the uncertainty associated with the model will grow as well. At some point, the uncertainty will become so large as to span the entire environment, therefore making the descriptor completely useless. In our experiments it became apparent that, using a Kalman filter with linear motion model, predictions become unreliable after just a few seconds, or approximately 10-20 time steps. This was due mostly to the unpredictability of people's motion, and only to a lesser extent to errors in the estimate of the state (which would become negligible after 5-10 time steps).

The fact that appearance observations are always compared to a reference, while motion observations are compared to the latest observation available, is also due to a fundamentally different theoretical basis behind descriptor generation. Motion descriptors assume a model (constant velocity) and a dynamic state for the target (the Kalman filter state, comprising position and speed in the world reference frame); with each new measurement (*i.e.* the position in the world reference frame), the state is updated according to the model and the measurement, and a new descriptor is generated that reflect these changes. In the case of appearance descriptors, on the other hand, the model is assumed to be constant (no variation of a target's appearance over time), and therefore there is no need to keep an up-to-date appearance model. While this assumption makes tracking difficult in the case of drastic changes of appearance, such as when a person removes their coat, it also means that the appearance descriptor is never contaminated with erroneous updates. This limitation is considered a minor issue by security operator, who, during informal conversations, conceded that it is a reasonable tradeoff (offering better tracking in the common case of constant appearance, and requiring manual intervention in the rare case of a drastic change).

3.5 Obtaining Target Observations

In the context of tracking in a wide surveillance area, there are three sources of information that contribute to an overall probability that any given target corresponds to the individual the operator tagged.

Firstly there is the coarse-scale temporal information that allows observations from different stations or different areas of the same station (*e.g.* different floors) to be assessed. Under the proposed framework, this is considered to be the probability of correct association, given only the time-stamps of the two observations (and the camera network layout). This is written as $p(i \equiv j | S_{ij})$, where S is a descriptor that allows matching targets based only on their timestamp and camera location (a more detailed explanation will be given in Section 3.8).

Secondly, there is the fine-scale spatio-temporal information available about the observations from overlapping, adjacent or nearby cameras. This conditional probability is written as $p(i \equiv j | M_{ij})$, where M is a motion descriptor as defined in the previous section. In any given case only one of M or S will be available, depending on the relationship between the cameras: if the camera views are overlapping, adjacent or nearby, and they are on the same ground plane, then the motion descriptor M can be used. If, on the other hand, cameras are in

different stations or in different areas of the same station, then only the spatio-temporal descriptor S can be used. When M can be used, it is preferred to S, because it is much more informative (as it encodes the target velocity and its position on the ground plane, whereas S only encodes information about the start and end cameras of the transition, and the transition time). Section 3.8 will clarify where the two different descriptors are applicable.

Thirdly, there is the appearance information from the colour descriptors A, and hence there is available a third conditional probability $p(i \equiv j | A_{ij})$. In order to convert a descriptor-specific similarity measure D_{ij} into a cross-descriptor probability, Bayes' theorem is used as explained in the following paragraphs.

A large data set with N targets has to be ground-truthed so that the individual corresponding to each target is known. A descriptor D is then applied to every target to generate a set of observations $D_T = \{D_{i_1}, D_{i_2}, \ldots D_{i_N}\}$. All observations in D_T are then compared pairwise to generate an $N \times N$ matrix of similarity measures, D_{TT}, also called a similarity matrix. The similarity matrix is symmetric, since $D_{ij} = D_{ji}$, and has all zeroes on the leading diagonal, since $D_{ii} = 0$. Some of these distances are between observations of the *same* individual, while others originate from *different* individuals. Using only the upper triangular part of D_{TT} without the leading diagonal, the distances are collected in three normalised histograms:

1. $\mathcal{H}_{\texttt{same}}[D]$, distances between pairs of observations generated by the same individual.
2. $\mathcal{H}_{\texttt{diff}}[D]$, distances between pairs of observations generated by different individuals.
3. $\mathcal{H}_{\texttt{aggr}}[D]$, all distances.

The histograms represent the relative frequencies of distance values for the three cases. They are also probability density functions, corresponding to the probability of obtaining a certain distance from a comparison of two observations, provided that the two observations are

- from the *same* individual,

$$p(D_{ij} | i \equiv j) \tag{3.7}$$

- from *different* individuals,

$$p(D_{ij} | i \not\equiv j) \tag{3.8}$$

- from either the same or different individuals,

$$p(D_{ij}) \ . \tag{3.9}$$

Applying Bayes' theorem to 3.7 and 3.9, we obtain

$$p(i \equiv j | D_{ij}) = \frac{p(D_{ij} | i \equiv j)\, p(i \equiv j)}{p(D_{ij})} \ , \tag{3.10}$$

where $p(i \equiv j)$ is the prior and is set to 0.5. Assigning equal probabilities to the "same" ($i \equiv j$) and "different" ($i \not\equiv j$) cases ensures that whatever is chosen depends only on the distance D_{ij}, and to on the prior (the following subsection gives the formal proof). Figure 3.4 shows a graphical representation of Bayes' theorem used to transform the histograms of the appearance descriptor (same, different and aggregate) into a probability that two targets represent the same individual.

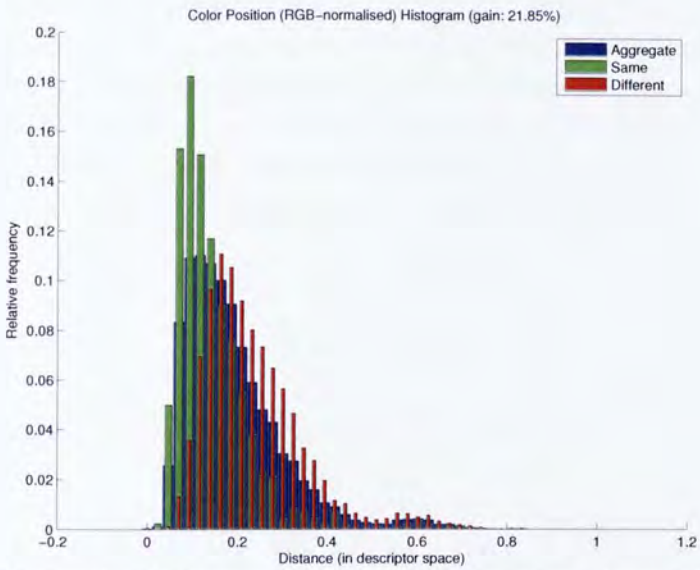
Equation 3.10 is the probability that two targets i and j represent the same individual according to descriptor D. It also permits to combine similarities obtained from descriptors of different types by converting them into probabilities. Once these probability density functions are defined for a descriptor, the concept of *entropy* can be used to quantify its efficacy, namely how better than random it is at distinguishing individuals. By combining three independent sources of information, target matching is much more reliable than by using appearance alone, as a native approach could suggest. If, for instance, the best candidate was chosen by selecting the most similar one to the reference target, the tracker wouldn't be able to compensate to appearance changes (due to lighting or point-of-view) that cause a different individual to be "more similar" to the reference target than the same individual. By also using motion and topological information, such an error is less likely, as the erroneous target needs to be similar and close to the correct target for it to be mis-associated.

3.5.1 Entropy

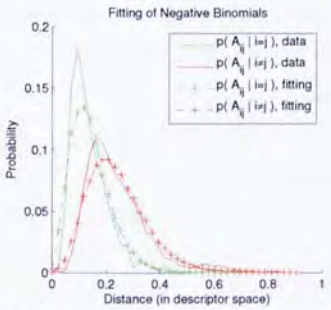
In information theory, entropy is a measure of the uncertainty associated with a random variable. By definition, the entropy of a random variable Y is

$$\mathrm{H}(Y) = -\sum_{y \in Y} p(y) \log p(y) \ . \tag{3.11}$$

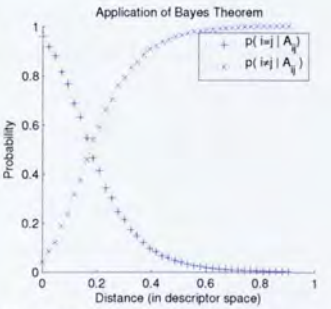
The base of the logarithm is irrelevant, as long as the same base is used throughout. If base 2 is used, then the entropy is expressed in bits (as is the case for



(a) *Normalised histograms of distances between descriptions. Green histogram: distances between pairs of descriptions generated by the same individual (Equation 3.7). Red histogram: distances between pairs of descriptions generated by different individuals (Equation 3.8). Blue histogram: aggregate distribution (Equation 3.9).*



(b) *Using two negative binomial functions (cross-dashed lines) to approximate the histograms in Figure 3.4a (continuous lines). The negative binomial distribution was chosen because, among the common univariate probability distributions, it was the one that gave the best fit regardless of the descriptor used.*



(c) *Probability of two targets being the same or different individuals given the distance between their appearance descriptions estimated using Bayes' theorem (Eq. 3.10). The value of 0.2 for the crossing point, in this example, is specific to the descriptor and dataset used to generate the histograms; other descriptors and datasets are likely to have their crossing point at different locations on the x-axis.*

Figure 3.4 – *Application of Bayes' theorem to derive $p(i \equiv j | A_{ij})$ given the histograms of distances between observations generated by an appearance descriptor A (the example data shown was generated with the Colour Position descriptor).*

the remainder of this thesis). If $p(y) = 0$ for some y, the value of the summand $0 \log 0$ is taken to be 0, which is consistent with the limit $\lim_{p \to 0} p \log p = 0$.

When used with descriptors, the random variable Y is $\{i \not\equiv j, i \equiv j\}$, that is, whether or not two descriptors represent the same individual prior to using an observation. For brevity, we shall use a pseudo-boolean notation, and write $Y = \{0, 1\}$ where 0 means "not the same" and 1 means "the same". Assuming equal probability for the two cases of the prior, $p(0) = p(1) = 1/2$. Since we are using base-2 logarithms, we can calculate a value for Equation 3.11:

$$\begin{aligned} \mathrm{H}(Y) &= -\sum_{y \in Y} p(y) \log_2 p(y) \\ &= -\sum_{y \in Y} 1/2 \log_2 1/2 \\ &= -1/2 \log_2 1/2 - 1/2 \log_2 1/2 \\ &= -\log_2 1/2 \\ &= \log_2 2 \\ &= 1 \end{aligned}$$

which is the entropy of a completely random choice between "same" and different". Any descriptor is expected to be better than random at discriminating between the two cases, and therefore should have entropy smaller than 1. An ideal descriptor, that can always discriminate correctly between the two cases, would have an entropy of 0. Note that the assumption that the "same" and "different" cases have the same prior probability is not valid in many real-world scenarios. Indeed, in a busy underground station, two observations are much more likely to correspond to different individuals than to the same one. However, the purpose of using entropy is to determine how much a descriptor improves the chances of correct identification, and for this purpose the actual value of the prior probability is not important. If, in order to represent the actual probability of a match, the prior were biased towards the "different" case, then a dummy descriptor that always classifies two targets as "different" would have a low entropy (possibly close to 0, depending on how biased the prior is), and this is of course undesirable.

Conditional entropy $\mathrm{H}(Y|X)$ represents the remaining entropy (i.e. uncertainty) of a random variable Y given that the value of a second random variable X is known. To estimate $\mathrm{H}(Y|X)$ the starting point is the two set of samples, *same* and *different*, shown above. These can be used to generate estimates of the probability $p(D_{ij}|i \equiv j)$ and $p(D_{ij}|i \not\equiv j)$, respectively, by taking the normalised histograms $\mathcal{H}_{\mathtt{same}}[D]$ and $\mathcal{H}_{\mathtt{diff}}[D]$. The choice of histogram number of bins

will affect the final estimate, a balance needs to be struck between the number of samples and the number of bins to avoid under-sampling or over-sampling. Alternatively, a parametric estimate of the distribution can be generated from the samples; in the case of a multi-dimensional histogram, a parametric representation also solves the problem of scarcity of samples due to the *curse of dimensionality* [114]. From these distributions, the prior distribution $p(D_{ij})$ can be constructed, and then Bayes' theorem can be used to invert the expression into the required form, $p(i \equiv j | D_{ij})$. The conditional entropy is calculated as the expected value of $\log p(i \equiv j | D_{ij})$.

3.6 Single Camera Tracking

The definition of a single camera tracker, given at the beginning of this chapter, described it as an algorithm that accepts as input a target i_{t-1} (observed at time $t-1$) and a video frame captured at time t, and gives as output a target i_t (observed at time t) representing the same individual, where all observations come from the same video stream.

To fulfil this role, any of the single-camera trackers reviewed in Section 2.2 can be used. These trackers require an explicit initialisation step, where they acquire the initial model of the object to be tracked. In the following frame, they search a neighbourhood of the initial location for a best-match to the initial model, report the new position, and possibly update the model. Within the multi-camera tracking framework, the first initialisation step is performed by the operator by tagging the reference target. Subsequently, if a better candidate is found and the multi-camera tracker needs to switch camera view, the bounding box of the candidate will be used to re-initialise the single-camera tracker. At each frame, the multi-camera tracker projects the position of the target in the image onto the ground plane, which is then used to generate a motion descriptor (see Section 3.8.1).

A single-camera tracker is not expected to be robust to occlusions, since the multi-camera tracker may resolve them; however, a single-camera tracker could signal when it is no longer able to follow a target (for example because of an occlusion or because the target is out of view). This could be done by checking the foreground detection mask corresponding to the target area: if it is empty or underpopulated (e.g. the number of foreground pixels is lower than a threshold), the tracker can signal a "target lost" event.

3.7 Appearance Descriptors

As their name implies, appearance descriptors represent a target's appearance. The following sections enumerate and explain the appearance descriptors developed as part of this project. Each explanation includes how observations are generated, what data they contain, and how they are matched. Where the descriptor was not developed by the author, proper credit is given. When referring to appearance descriptors, without specifying which one, they are represented with the letter A.

3.7.1 Mean Colour Descriptor E

One of the simplest ways of describing a target's appearance is by using its mean colour. Indeed, the Mean Colour descriptor E is easy to implement and it is computationally efficient, both to generate observations and to compare them. Given the set of pixels (p) representing a target i in an image using an arbitrary colour-space (for the sake of simplicity, RGB will be used in this example), the Mean Colour Descriptor is defined as the mean of the pixels colours:

$$E_i = \frac{\sum_{p \in i} [p_r, p_g, p_b]}{|i|} \tag{3.12}$$

where $|i|$ is the number of pixels comprised by the target i. Because E_i is a vector in the colour-space used by the image, the distance between two observations E_i, E_j is simply $||E_i - E_j||$. Clearly, many variations of this descriptor can be defined using different colour spaces.

3.7.2 MPEG-7 Descriptors

At the other end of the complexity spectrum, there is a host of MPEG-7 Visual Descriptors [68]. The main goal of the MPEG-7 visual standard is to provide standardised descriptions of images and videos. These descriptors can be used to compare, filter or browse multimedia content without the need for text-based queries. Although there are a number of colour, textures, shape and motion descriptors defined in the standard aimed at different user domains (multimedia catalogues, media selection, media authoring, etc...), no descriptors were specifically designed for people re-identification in surveillance videos. Colour descriptors represent different aspects of the colour feature, including colour distribution, spatial colour layout and spatial colour structure. There are 6 descriptors defined by the standard, two of which are relevant to tracking and

will be explained in the following section. The author wishes to thank James Annesley for providing the implementation of the Dominant Colour descriptor.

3.7.2.1 Colour Spaces and Dominant Colour Descriptors

The MPEG-7 standard specifies the following colour-spaces: RGB, YCbCr, HSV, HMMD, Linear, and Monochrome. RGB is one of the more popular spaces, and it is defined as the unit cube in the Cartesian coordinate system. The Linear colour-space is a linear transformation from the RGB space. The YCbCr space, used extensively in MPEG-1/2/4, is actually a Linear space whose transformation matrix has been defined by the standard. The Monochrome space is the Y component of the YCbCr space. HSV and HMMD are nonlinear transformations from the RGB space that are more perceptually uniform.

The standard also specifies colour-space components as continuous-value entities, that need to be quantised for discrete representation. The Colour-Space Descriptor specifies the colour-space an image is encoded in, and how that space is quantised.

The Dominant Colour Descriptor provides a compact description of the representative colours in an image or image region. Its main target applications are similarity retrieval in image databases and browsing of image databases based on single or several colour values. The Dominant Colour Descriptor T of a target i is defined to be

$$T_i = \{(\mathbf{c}_k, p_k, v_k), s\}, \ (k = 1, 2, \ldots, N)$$

where N is the number of dominant colours. Each dominant colour value $\mathbf{c}_k$ is a vector in the corresponding colour-space (described by the Colour Space Descriptor), p_k is the proportion of pixels in the target corresponding to colour $\mathbf{c}_k$, such that $\sum_{k=1}^{N} p_k = 1$. The optional colour variance v_i describes the variation of the colour values of the pixels in a cluster around the corresponding representative colour. The last component, s, is the spatial coherency, a number that represents the overall spatial homogeneity of the dominant colours in the image. The number of dominant colours N can vary from image to image, up to a maximum of 8 (as defined by the standard).

One comparison function between two Dominant Colour observations $T_i = \{(\mathbf{c}_{i,a}, p_{i,a}, v_{i,a}), s_i\}, \ (a = 1 \ldots N_i)$, and $T_j = \{(\mathbf{c}_{j,b}, p_{j,b}, v_{j,b})\}, \ (b = 1 \ldots N_j)$, is defined by the MPEG-7 standard as

$$c(T_i, T_j) = \sum_{a=1}^{N_i} p_{i,a}^2 + \sum_{b=1}^{N_j} p_{j,b}^2 - \sum_{a=1}^{N_i} \sum_{b=1}^{N_j} 2\alpha(\mathbf{c}_{i,a}, \mathbf{c}_{j,b}) p_{i,a} p_{j,b}.$$

In the above equation, $\alpha(\cdot,\cdot)$ is a similarity function between two colours:

$$\alpha(\mathbf{c}_1,\mathbf{c}_2) = \begin{cases} 1 - \frac{|\mathbf{c}_1-\mathbf{c}_2|}{d_{\max}} & |\mathbf{c}_1 - \mathbf{c}_2| \leq \tau_d \\ 0 & \text{otherwise} \end{cases}$$

where τ_d is the maximum distance for two colours to be considered similar and $d_{\max} = \kappa\tau_d$. This means that two dominant colours from one single observation are at least τ_d distance apart. The standard gives recommended values for τ_d and κ in the CIE-LUV colour-space.

3.7.3 Colour Position Descriptor

As it will be shown in Section 5.2.2, the best results were obtained with a Colour Position descriptor developed at INRETS[1] [106]. Colour Position descriptors P divide a target i into n equally spaced horizontal bands (typically 8). The mean colour of the pixels in each band and in each colour channel is computed, and the three components of each are used as a $3n$-dimensional vector. For example, a Colour Position Descriptor using n bands and the RGB colour-space is defined as

$$P_i = [r_1, g_1, b_1, \ldots, r_n, g_n, b_n]^{\mathrm{T}} ,$$

where r_1 is the mean on the red channel of the first band, g_1 is the mean on the green channel in the first band, and so on.

The distance between two Colour Position observations is the distance between these vectors: $P_{ij} = ||P_i - P_j||$.

In their paper, the authors also suggest two colour-spaces that improve the descriptor performance by removing most of the luminosity information: the normalised RGB space and the UV space. The former is a standard RGB space where all pixels have been normalised so that they have unitary norm, while the latter is simply a YUV space without the Y component (details on both the latter colour spaces can be found in any image processing book, *e.g.* [38]).

3.8 Spatio-Temporal Descriptors

Spatio-Temporal descriptors represent a target by its location in the surveillance network and by the time when it was observed. The two descriptors in this

[1] *Institut National de Recherche sur les Transports et leur Sécurité*, the French National Institute for Research on Transport and its Safety.

category, Motion and Spatial, were developed to complement each other, as the observations they generate are effective for targets that are, respectively, close to each other (in space and time) or far apart. The exact conditions under which the two descriptors are more effective will be detailed in their respective sections.

In a surveillance scenario, it would be useful to have all cameras calibrated with respect to the same world reference frame. However, if the camera network is very large (*e.g.* spanning an entire city, as in one of Caretaker's test-sites), ensuring that the reference frame is the same may require using GPS and geographical data (which is difficult to obtain) and would make calibration more difficult (since 3D information must be considered for multi-level stations). It was therefore decided to group cameras according to the station and level they belonged to, and to calibrate each group independently from the others. We call *coterie* a group of cameras positioned on the same level (in the same station) and calibrated according to the same ground plane. In every coterie, there are some special cameras that are connected to other coteries. All inter-coterie connections in the surveillance network are assumed to be known. Moreover, the average transition time of each connection is also assumed to be known. A problem with this design, however, is that it cannot model the presence of multiple ground planes in the same view, e.g. a stairway connecting two floors. A possible solution is proposed by Yin *et al.* [118], but it is outside the scope of this thesis.

3.8.1 Motion Descriptor M

The Motion descriptor$M$ represents the value and accuracy of a target's position and velocity:

$$M_i = [\mathbf{x}_i, \dot{\mathbf{x}}_i, \sigma_{\mathbf{x}_i}, \sigma_{\dot{\mathbf{x}}_i}]^{\mathrm{T}} \tag{3.13}$$

where $\sigma_{\mathbf{x}_i}$ and $\sigma_{\dot{\mathbf{x}}_i}$ are covariance matrices. Spatio-temporal descriptors are particularly effective when comparing targets from the same camera, or from different cameras in the same coterie.

Using a Kalman filter to track the targets, their position and velocity are known at every frame, along with an uncertainty ellipse. Other filters may be used as well, *e.g.* the particle filter, but their use was not investigated due to time constraints. Candidate targets from different cameras within the same coterie as the current camera can be compared to the current target using the Mahalanobis distance.

The Mahalanobis distance is a similarity measure between a known sample set

to an unknown one. In contrast to Euclidean distance, it takes into account the correlations of the data set and is scale-invariant, i.e. it does not depend on the scale of measurements. The Mahalanobis distance of two random vectors $\mathbf{x}$ and $\mathbf{y}$ extracted from the same probability distribution with covariance matrix σ is

$$\mathcal{M}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} \sigma (\mathbf{x} - \mathbf{y})}$$

When the Mahalanobis distance is applied to spatio-temporal observations, however, the two random vectors $\mathbf{x}$ and $\mathbf{y}$ do not come from the same distribution, since they correspond to the motion features of two different targets with different covariance matrices σ_x and σ_y. For example, the Mahalanobis distance between two spatio-temporal observations S_i and S_j is

$$\mathcal{M}(S_i, S_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \sigma_{\mathbf{x}_i} (\mathbf{x}_i - \mathbf{x}_j)}. \tag{3.14}$$

In this case, the Mahalanobis distance is not symmetric, because $\mathbf{x}_i$ and $\mathbf{x}_j$ are not sampled from the same distribution, and therefore they have different covariance matrices. Moreover, only the position of the target is used, because some targets are observed only on one frame, and therefore their velocity is undefined. However, because symmetry is required to build the similarity matrix, the comparison function for two spatio-temporal observations uses a symmetric variation of Equation 3.14:

$$c(M_i, M_j) = \frac{\mathcal{M}(M_i, M_j) + \mathcal{M}(M_j, M_i)}{2} \tag{3.15}$$

which yields a value between 0 and ∞. Indeed, the comparison function for spatio-temporal observations is not applicable if the involved targets were observed in different coteries. If two targets i and j were observed in the same coterie, but at different times (t_i and t_j), the comparison may still be applied, but the time difference has to be catered for. This can be done by using the motion model of the filter to predict the future position of the target whose observation is the oldest. Assuming without loss of generality that $t_i > t_j$, let $\Delta t = t_i - t_j$. The position $\mathbf{x}_{j'}$ of target j at time t_i can be estimated by predicting the motion of target j. Assuming a simple linear model, the prediction is

$$\mathbf{x}_{j'} = \mathbf{x}_j + \dot{\mathbf{x}}_j \Delta t$$

and the filter equations can be used to predict the uncertainty on position and velocity. This defines a new "virtual" observation, $M_{j'}$, that can be compared with S_i using Equation 3.15.

3.8.2 Coarse-Scale Spatial Descriptor S

The Spatial descriptor is a non-appearance descriptor suitable for comparing targets acquired at times or places far apart from each other. Indeed, if two targets are "close enough" in space and time, a Spatio-Temporal descriptor should be able to discriminate between the two, by using the motion model to extrapolate the position of one of the targets. However, if two targets are too far apart in time, the predicted position of the oldest target will have too much uncertainty to be reliably used for a match. Moreover, if the two targets have been acquired from cameras situated in different coteries, it is not possible to use motion prediction (by definition of coterie).

Spatial descriptors allow targets to be matched in the above scenarios without using appearance. Because the physical relation between coteries is assumed not to be known (apart from the cross-coterie connections and their average transition times), only timestamps (t) and camera names (id) may be used to estimate this probability: additional spatio-temporal information such as velocity and position (with respect to the ground plane) is not applicable. The definition of a spatial descriptor therefore is

$$S_i = \{\mathrm{id}_i, t_i\} \ . \tag{3.16}$$

Given two targets i, j and their corresponding spatial descriptors, S_i, S_j, the distance S_{ij} between them is a function of the two targets' timestamps (t_i,t_j), and the minimum time required for an individual to transit from one camera (id_i) to the other (id_j). It is computed as follows:

1. Use a shortest-path algorithm to transverse the network layout graph and find a minimum path between id_i and id_j.
2. Let $t_{\min}$ be the sum of all transition times of each edge of the path.
3. $S_{ij} = \begin{cases} |t_i - t_j| - t_{\min} & \text{if } |t_i - t_j| > t_{\min} \\ \infty & \text{otherwise.} \end{cases}$, expressed in seconds.

Minimum transition times between pairs of topologically adjacent cameras have to be estimated from training data. The minimum transition time between two non-adjacent cameras a and b can be inferred from the topology by finding the shortest path between a and b, and summing the transition times of all connections along the path. Where $|t_i - t_j|$ us the transition time between the two cameras. This formula ensures that, if the actual transition time is less than the minimum transition time, the two targets are never identified as the

same individual. The distance is minimal if the actual transition time is the same as the minimum transition time, and increases linearly as the two times diverge. The choice of a linear relation, and the parameters of this relation, are not particularly important in this case, since the distance will be non-linearly mapped to a probability density function as explained in Section 3.5 .

3.9 Conclusion

In this chapter, a theoretical model for tracking across multiple non-overlapped cameras was described. This model, based on a probabilistic framework, allows a tracker to fuse information regarding a target's appearance, its position, and a prior model of the camera network, in order to estimate the target's whereabouts in real time. The next chapter, of a more practical nature, will describe the motion detection and colour correction algorithm that supply data to the modules described here (the appearance descriptor and the single camera tracker), and each analysed module is evaluated independently. In Chapter 5 the effectiveness of the TNT system as a whole is evaluated, leading to the conclusions in Chapter 6.

Chapter 4

Pre-Processing

This chapter presents one novel motion detection algorithm and one novel colour normalisation algorithm. Motion detection and colour normalisation are the very first stage of processing in the proposed tracking framework, as shown in Figure 4.1. Motion detection can be performed using any of the techniques reviewed in Section 2.4.1, but a novel method was also developed that exploits periodicity in the background to deterministically predict what colour a colour-changing pixel is going to assume on subsequent frames [59].

Colour normalisation can be used to compensate for the different colour responses of different cameras and to illumination changes across different scenes; colour normalisation techniques were reviewed in Section 2.5.3. However, since target appearance is only used to compare appearance descriptions across cameras, if the appearance descriptor used is robust to illumination changes and cameras colour responses are not too dissimilar, then the colour normalisation stage can be skipped, leaving more computational resources available for the rest of the tracking system.

4.1 Foreground Detection

For advanced video surveillance systems, background subtraction tools (such as those reviewed in Section 2.4.1) can allow the detection of the moving objects in the scene. Background subtraction requires a sufficiently accurate model of the background to enable foreground objects to be distinguished from the background. This section considers the case in which the background is moving according to some repeating and predictable pattern. In data captured at

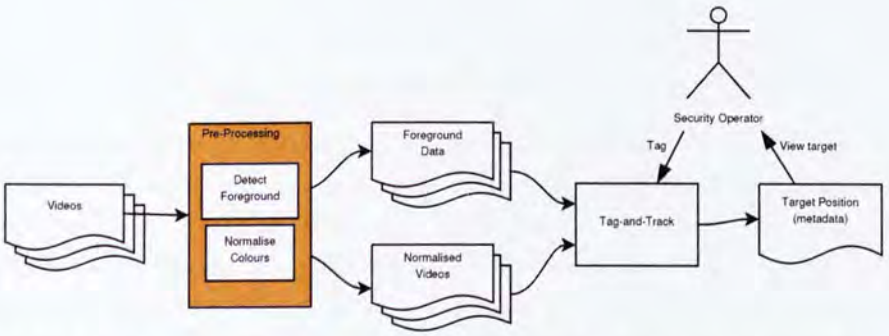


Figure 4.1 – *Overview of the tracking framework . From left to right: video streams are pre-processed to produce a foreground-data stream (see sections 2.4.1 and 2.4.2) and, optionally, colour-normalised video streams (see Section 2.5.3). These streams are passed to the tracker, described in Chapter 3, that, after having been initialised with the initial "tag", produces metadata (position of the target) to be provided to operators in the control room.*

underground metro stations, three elements of the background were exhibiting this property: escalators, flashing warning lights (Figure 4.3a) and scrolling advertisements (Figure 4.3c).

The work is motivated by a requirement to accurately and automatically model a background which includes elements that are periodically varying, with an unknown frequency (up to an arbitrary limit). This model can then be used to predict the appearance of these elements in future frames, and thereby to distinguish this changing background from foreground elements (such as moving people), allowing foreground objects on the periodic background to be more accurately estimated. Therefore, a common metric to assess the efficacy of the background modelling technique is the foreground detection performance that it can provide.

The simplest type of natural background is that observed by a fixed camera with constant illumination. In this case, each pixel of the background image can be modelled with a Gaussian random variable. More complex variations in the background signal can be modelled with a Gaussian Mixture Model (GMM, [97]). This can account both for abrupt signal changes caused by small camera motions and small movements in elements of the scene, e.g. windblown trees. This approach has also been used to combat step-changes in appearance, e.g. changes in illumination caused by moving clouds, street lights, headlights, and other sources of light and shadow. For the case of significant camera motion, this motion along with the background would need to be modelled. The GMM model is also applicable to periodically varying backgrounds: for each pixel, each component of the period is modelled by the most appropriate element in

the mixture. Indeed, the regular cycle through these components ensures that the relative priors for the mixture elements can be accurately estimated. (This is in contrast to less predictable variations such as alternation between cloudy and sun-lit illumination, a situation this algorithm is not design to cater for.)

Alternatively, periodic variation in appearance can be considered as a special case of *dynamic texture*, and techniques for modelling this process have already been proposed. Soatto *et al.* [95] used a Kalman filter to model the evolutionary process of the dynamic texture, and determined the parameters using an iterative technique similar to Expectation-Maximisation (EM). This approach was adopted in [121], and segmentation of foreground objects from a dynamic background was achieved.

The experiments are performed on two data sources collected as part of the CARETAKER Project. One sequence shows a platform and escalator in a station that forms part of the Torino Metro system. The periodic background elements are the escalator (with a period of approximately one second) and a flashing warning light (approximately two seconds). The other sequence shows the top of an escalator that is part of the Roma Metro system. It includes an advertising board that scrolls every ten seconds between four adverts, giving an overall period of about forty seconds. Foreground detection experiments on the Torino Metro data set show a significant improvement over the technique of Gaussian Mixture Models.

4.1.1 System Overview

The system operation is divided into training and update phases, as illustrated in Figure 4.2. In the training phase, the video sequence is used to generate a time series for each pixel. As explained in the following section, the corresponding set of Fourier coefficients provides the data to distinguish the periodic from the non-periodic elements. The Fourier analysis also gives an estimation of the number of states required in the Markov model. Neighbouring pixels of the same periodicity are grouped into regions and can be processed together in the subsequent steps of the system. The next steps in the training phase are to initialise the values of the states, and calculate the matrix of transition probabilities. These processes are described in Section 4.1.3.

Once the training is complete, the system moves into the update phase. Here, a predicted state based on the current state is compared to a posterior state based on a Bayes update using the current state and a measurement (from the video data). Depending on whether the two states agree, the next state is determined,

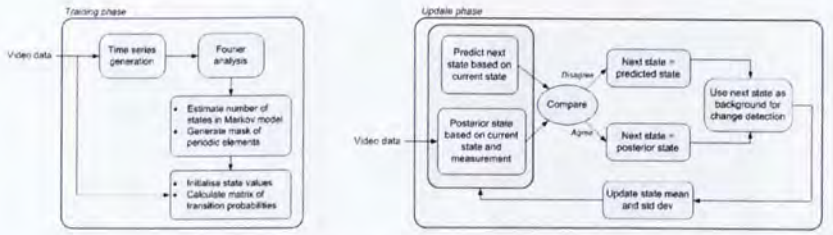


Figure 4.2 – *System diagram of the foreground detection algorithm.*

and the corresponding state values are used to provide a background for change detection. The mean and standard deviation of the state are then updated.

4.1.2 Detection of Periodic Scene Elements

In this section, a method is described for detecting the pixels which exhibit significant periodic characteristics and estimating this period for each of them. The effectiveness of the method is then evaluated by comparing its output with a hand-labeled map of periodic elements, and results are shown in Section 4.1.4.

The analysis begins by creating the time series of the intensity value of each pixel over an appropriate length of training window in time domain. Colour information is not used, because as it is extremely unlikely that an object in the scene changes colour without changing intensity, processing all three channels would add to the computational cost without providing any additional value. The duration of the training phase is dependent on the maximum length of the period to be modelled, and was chosen to span at least five complete periods of the background signal. It should be noted that the analysis is not limited to using pixel intensity; other pixel features can also be used. Figure 4.3 shows the scenarios in the Torino and Roma stations respectively. A typical time series (of the escalator region) is also shown in Figure 4.3a. All pixels in the sequences were included in the generation of the time series (no spatial subsampling). For the Torino dataset the temporal sampling frequency was 5 frames per second. For the Roma dataset, however, the temporal sampling frequency was much coarser at 4 frames every 10 seconds, since in this case the true period is very long (40 seconds) and the inclusion of every frame would impose an excessive computational demand (processing 200 frames at a resolution of 720×576 requires over 1.7GB of RAM with the widely used FFTW[1] library).

Next the Discrete Fourier Transform (DFT) [18] over each time series is computed. The absolute (modulus) values are used to discriminate between peri-

[1] An open-source subroutine library for computing the discrete Fourier transform, available from http://www.fftw.org/

(a) (b) (c)

Figure 4.3 – *Datasets: (a) The escalator scene from Torino underground. (b) Time series of 12 pixels sampled from the escalator for 100 frames. (c) The scrolling advertisement scene from Rome Underground. The advertisement presents three different posters, A, B, and C, repeated as A-B-C-B. Each poster remains visible for approximately 10 seconds (including scrolling time), yielding a main period of about 40 seconds.*

odic and non-periodic background elements, and for the former, to estimate the main frequency of the periodic motion. Naturally, the periodic elements have a distinct peak at their fundamental frequency, and also at associated harmonics. The task is to discriminate between these peaks and the highest values produced by the stochastic non-periodic signals. The detection of peaks is band-limited to a set range: from above, by the Nyquist limit (or else the shortest period to be detected); from below, by the sample period (or else the longest period to be detected). In general, the spectra of natural scenes (as opposed to synthesised images) follow a $1/f$ distribution [13] (this distribution also applies for natural scenes spectra in the spatial frequency domain). The pre-normalised spectra for the Torino data are illustrated in Figure 4.4. In order to simplify subsequent calculations, we compensate for the $1/f$ distribution by normalising each spectrum multiplying each element by its corresponding frequency. The next step is to compute the mean μ of each normalised spectrum. Periodic spectra will show a peak whose value is much higher than μ, while non-periodic spectra will resemble white noise. The two can therefore be discriminated by thresholding each spectrum with $K\mu$, where K is a constant, and checking whether any component of the spectrum survives. If there is at least a surviving component, the spectrum is periodic and that component should be the main frequency (although, rarely, harmonics are detected instead of the main frequency). The actual range of periodicity that can be detected depends on available memory (since processing a longer time series requires more RAM) and on the spacial and temporal resolution desired, since videos can be down-sampled spatially and/or temporally to allow longer periods to be detected.

The output of the thresholding procedure is a binary image where each pixel is classified either as periodic or non-periodic, and for periodic pixels, a frequency estimate is provided (corresponding to the first peak above the threshold). This

output was compared to a ground truth, generated by manually annotating the Torino and Roma videos. The ground truth consists of a binary image where pixels corresponding to the periodic elements (i.e. the escalator and the flashing light) are white and all other non-periodic pixels are black. Even though small errors in ground-truthing may bias evaluation, and a thorough evaluation should have several humans annotate the same frames in order to estimate ground truthing errors, this bias was estimated to be too small (compared to actual segmentation performance) to justify the additional cost in human resources. Instead, priority was given to annotate as long a video segment as possible.

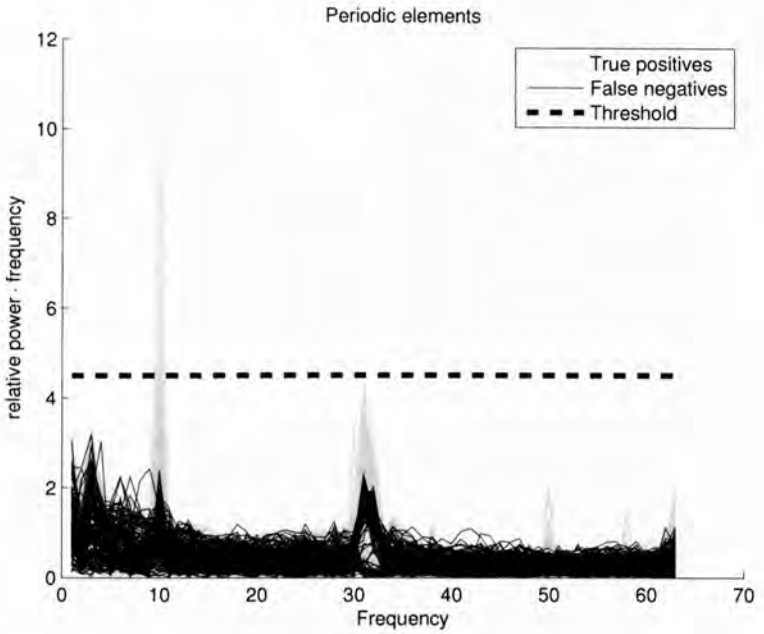
The detector algorithm was run with varying threshold values, and the output was compared against the ground truth in order to generate a ROC (Receiver Operating Characteristic) curve. A ROC curve is a plot of True Positive Rate (TPR) against False Positive Rate (FPR) and these are defined as:

$$\begin{aligned} \text{TPR} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ \text{FPR} &= \frac{\text{FP}}{\text{FP}+\text{TN}} \end{aligned} \tag{4.1}$$

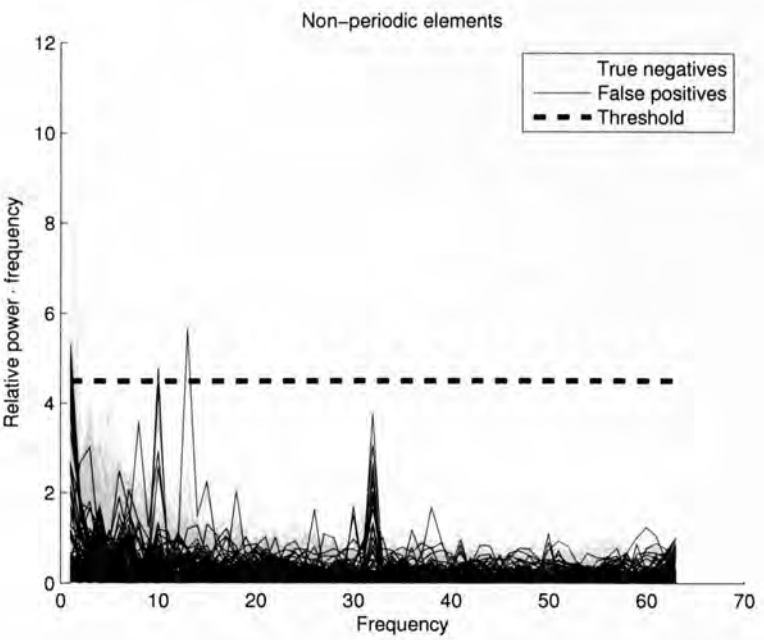
where TP, FP, TN, FN are the number of true positives, false positives, true negatives, and false negatives respectively [35]; more specifically,

- *true positives* are periodic pixels correctly detected as periodic;
- *true negatives* are non-periodic pixels correctly not detected as periodic;
- *false positives* are non-periodic pixels erroneously detected as periodic;
- *false negatives* are periodic pixels erroneously not detected as periodic.

Lower values of K make the algorithm more sensitive, and values between 4 and 5 have been found to give the best results. All the results shown in the following pages were achieved using $K = 4.5$ (suggesting, therefore, that the best value of k is scene-independent, since the two capture scenarios differed significantly in resolution, frame rate, and periodicity of the background). In a previous version of this framework, the threshold was fixed and therefore it had to be changed for the algorithm to work with different time sampling intervals [26]. In the version discussed in [59], however, since the threshold changes with the mean μ of the spectrum, the same parameter K can be used regardless of the sampling interval (*e.g.* the Rome and Torino sequences were sampled at 0.4Hz and 5.0Hz respectively). In Figure 4.5 the output of the detection process is illustrated alongside the hand-labelled periodic pixels.



(a) *Periodic elements.*



(b) *Non-periodic elements.*

Figure 4.4 – *Normalised spectra of the pixels from the escalator scene (Figure 4.3b). For display purposes, the DC component has been removed and the amplitude has been normalised by the mean μ, so that k could be plotted (since now all spectra have $\mu = 1$). For a definition of* true positives, etc. . . , *see text after Equation 4.1. Frequency is expressed in Hz. Error rates are reported in Section 4.6.*

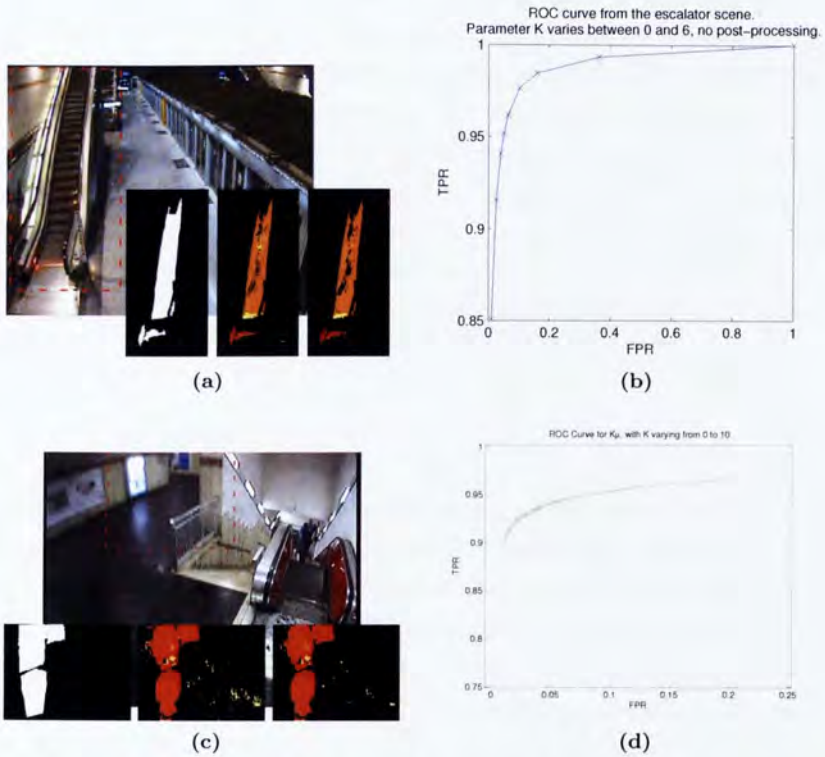


Figure 4.5 – *Comparison between the ground truth and the detection output of the periodic background detection algorithm applied to the escalator and scrolling advertisement scenes. Subfigures (a) and (c) show the scene and, inset, the ground truth, the detection output, and the output after the application of a dilate/erosion operation (coloured pixels are periodic, and different colours indicate different frequencies). Subfigures (b) and (d) are the corresponding ROC curves.*

The output of the algorithm serves as a mask to identify periodic background elements. The detected frequency of each element, measured in *frames per period* (fpp) is used to choose the number of states used to model the periodic background, as explained in the following section.

4.1.3 Markov Model

Once the pixels of the image displaying periodicity have been detected, adjacent pixels having the same periodicity are grouped into regions (either manually or using a clustering algorithm such as k-means [64]), and a Markov model for each region is constructed. The number s of states in the model is first estimated according to the DFT analysis, and it should be as close as possible to the number of frames per period (FPP) of the background variation. However,

because the number FPP is not necessarily an integer, it is approximated by two "tentative" models, one with ceil (FPP) states, and one with floor (FPP) states, where ceil is the function that maps a number to the smallest following integer and floor the one that maps a number to the largest previous integer, and the decision is finalised by assessing the performance of each model in the learning phase. For instance, in the Torino dataset, the detected frequency of the escalator pixels is 5.3 frames per period, hence both a 5-state and a 6-state model are tested. If more than 6 states were used, there would be redundancy in the model; if fewer than 5 states were used, the different phases in the period would not all be distinguishable (as a state may have to cover a number of frames).

4.1.3.1 Learning Phase

The learning of the transition probabilities of the Markov model is performed using the classic Baum-Welch algorithm [11] available in MATLAB. It is assumed that the first N measurements correspond directly to the N states in the model. Subsequent measurements are assigned a state whose values are closest to the measurement in a Euclidean sense. Once the learning phase is completed (this was achieved in just over 100 frames of real data, *i.e.* about 20 seconds at 5fps), the transition matrix can be used in the update phase.

As mentioned, both a 5-state and a 6-state model have been tested, where the pixels on the escalator, detected as periodic, were used for the purpose of learning and validation. It was found that a 5-state model is more accurate, compared to a 6-state model, in predicting the subsequent state correctly; consequently this has been used in the subsequent stages of the system.

4.1.3.2 Update Phase

Once the learning phase is completed, the algorithm in run in the update phase, where the goal is to determine the next state that the system will be in at each time step. There are two estimates that can be determined: a prediction that is based solely on the transition probabilities estimated in the learning phase, $\hat{s}$; and an *a posteriori* estimate that incorporates a measurement, $\tilde{s}$. These two quantities are related as follows:

$$p(\tilde{s}_i|m) = \frac{p(m|\hat{s}_i)\,p(\hat{s}_i)}{p(m)} \tag{4.2}$$

where $i \in I$ is the set of possible next states, $\hat{s}$ denotes the prior state, $\tilde{s}$ denotes the posterior state, and m is the measurement. The measurement likelihood is

calculated as:

$$p(m|\hat{s}_i) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(m - v_{s_i})^{\mathrm{T}} \Sigma^{-1} (m - v_{s_i})\right\} \tag{4.3}$$

where v_{s_i} is the value of state i.

The predicted *a priori* state $\hat{s}$ (estimated without taking the measurement into account) is compared with $\tilde{s}$, the state with the highest posterior probability given the measurement. If the predicted state and the posterior state agree (to within 1 state), then the posterior state is accepted as the next state $\overline{s}$. Otherwise, the predicted state is accepted as the next state (since the mismatch of $\hat{s}$ and $\tilde{s}$ is probably due to the presence of foreground pixels).

The corresponding mean and standard deviation of the next state $\overline{s}$ can then be updated, using a similar scheme to that of Stauffer and Grimson's Gaussian mixture model [96]. An adaptive scheme is preferred over a static approach where the state values are not updated because the Markov model is only an approximation to the periodic process (in this case, a 5-state model for 5.3 frames per period). Moreover, by calculating a standard deviation on the mean, a decision threshold for segmenting the foreground can be defined as a scalar multiple of the standard deviation. The update equations are as follows:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho m_t \tag{4.4}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(m_t - \mu_t)^{\mathrm{T}}(m_t - \mu_t) \tag{4.5}$$

where

$$\rho = \alpha p(m_t|\mu_t, \sigma_t)\,. \tag{4.6}$$

Here, t denotes time, and α is the learning rate of the algorithm. The conditional probability in Equation 4.6 is equivalent to the one in Equation 4.3 since the state can be represented by a mean and a standard deviation.

The technique was applied to the real data collected at a Torino underground station. Since the objective of this technique is to provide a reliable periodic background from which foreground segmentation can be generated, results of foreground mask generation are given in Section 4.1.4 to demonstrate the efficacy of this approach.

It should be pointed out that the Markov model described here takes the structure of a Hidden Markov Model (HMM), and it is possible to use a series of measurements (instead of only the previous one) to estimate the current state. However, a preliminary evaluation phase showed that using a series of measurements would increase the computational cost without improving the final result,

since the current state and the previous state are sufficient to determine where in the periodic cycle the system is.

Since different camera views would capture different scene elements with different periodicities, training has to be performed for each camera view in a networked system. The time required for training is dependent on the length of the maximum period one wishes to detect, the computational complexity of FFTW being $O(n \log_2 n)$, and at least five or six periods being required for the periodically varying pixels to be detected. For the escalator here, the training process is fast (just over 100 frames, *i.e.* 20 seconds at 5fps); therefore if elements of similar periodicities are to be detected, the training can be achieved in a short time. Results are presented in Section 4.1.4.

4.1.4 Results

The real data used to test the foreground detecting method described in Section 4.1 is the escalator sequence with people in the foreground (i.e. on the escalator), a sample frame of which is shown in Figure 4.6a. Only the area of interest has been displayed here for clarity - the section of the image with the platform has not been shown.

This sequence has approximately 300 frames at 5fps and involves 9 people using the escalator; a rectangular area of 150 × 280 pixels centred on the escalator was defined and used for evaluation, providing a total of 12.6 megapixels (150 × 280 × 300 frames) to test the algorithm on. The periodicity was determined using the periodic background detector described in Section 4.1.2, and the 5-state Markov model was constructed following the approach described above. Figure 4.6b shows the foreground mask of the proposed approach for one frame, and Figure 4.6c shows the corresponding output of the GMM. It should be noted that only the escalator region defined by the mask generated by the periodic background detector is being processed. It can be seen that the proposed approach seems to produce fewer false positives and the output regions are more contiguous in this frame. For a quantitative comparison of the two algorithms, a ROC analysis is carried out and the results are presented below.

Ultimately, the efficacy of the background modelling technique is assessed by the foreground detection performance that it can provide; therefore the segmentation output of the proposed method is compared against that of the Gaussian Mixture Model (GMM). The parameters for both methods have been varied over a wide range to obtain the best performance in each case (see Table 4.1). It is believed that the GMM provides the best baseline for comparison because it has the ability to model multi-modal distributions, and is adaptive. In fact,

Algorithm	Parameter	Description
Markov Chain	α	learning rate
	k	scalar determining the threshold for the detection of periodic pixels (see Figure 4.4)
	initVar	initial variance of the state model
GMM	α	learning rate
	numModels	number of Gaussians
	initVar	initial variance of Gaussians
	initWeight	initial weight of new model
	numStdDev	scalar determining the threshold between background and foreground pixels
	T	proportion attributable to background

Table 4.1 – *Parameters of the Gaussian Mixture Model and Periodic Background segmentation algorithms.*

the proposed method can be viewed as a GMM with a prior on the transition probabilities. Using these outputs, a ROC curve is generated, plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) as defined in Equation 4.1 on page 65.

The ground truth values have been generated by manually labelling the pixels as foreground or background. Foreground pixels here correspond to pixels where a person is present on the escalator. The resulting ROC curve of a pixel-based analysis is shown in Figure 4.6d. The blue square points correspond to the proposed algorithm, while the red triangular points are the outputs of the GMM. It can be seen that over the parameter space, the proposed method performs better than the GMM. For 80% TPR, the FPR is reduced relatively by 40%. At 20% FPR, the number of missed detections is reduced relatively by 47%.

The implications of this improvement are not straightforward to assess. For detection and tracking of pedestrians, the increased reliability of foreground detection can be expected to lead to improved track reliability. The effect will be restricted to specific areas of the scene such as escalators and advertising boards, which in some sequences cover over 20% of the image area. If these areas have a key role in the detection of important events, then the improvement will have a significant impact. Also, there is a significant amount of manual intervention needed at the moment: clustering of periodic pixels, selection of a subsampling frequency, and selection of a maximum period to be detected. The fact that the algorithm needs re-training after major lighting changes is also a deterrent to wider adoption.

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

Figure 4.6 – *Comparison of foreground detection accuracy of the Periodic Background Detector compared to the Gaussian Mixture Model, applied to a camera view from the Torino dataset, without noticeable lighting changes. (a) A sample data frame where the humans on the escalator are the foreground components that we wish to detect. Note that here only the escalator region is processed. (b) Foreground map using the proposed method. (c) Foreground map using GMM. (d) ROC curve of the pixel-wise.*

4.2 Colour Correction: Normalisation of Chromaticity and Intensity

Systematic variations between cameras of the input signals may be caused by differences in hardware, configuration, illumination and setting. The purpose of colour normalisation is to minimise colour differences between cameras due to different illumination and colour responses (as opposed to differences due to uneven illumination *within* one camera view). In this section the input data is represented in the YCbCr colour-space, each channel having 8 bits *i.e.* a range of between 0 and 255.

Several colour constancy techniques exist, such as [83] and [50], that require a set of reference colours to be given as input. However, given the high number of cameras in a typical surveillance scenario (e.g. in the GTT test site there are 22 cameras per station, and 12 stations), a completely unsupervised method has been preferred. The approach presented in this chapter and published in [27] is an adaptation of the "grey world" algorithm [36], employed in the context of foreground data in common between cameras [37] (but in our approach there is no requirement for camera overlap). It assumes that the set of foreground data from each camera is a fair sample from an underlying distribution of object appearances which need to be rendered as invariant as possible with respect to which camera they are observed from. The assumption that all cameras observe, *on average*, the same set of foreground data is justified in a surveillance scenario, where passengers transiting by the station will be seen by several cameras. Over a long enough period of time, random variations due to differently coloured passengers taking different routes should average out.

4.2.1 First-Order

To normalise the data from two or more cameras, it is assumed that a training set of observations is available, drawn from the same distribution of passengers and poses. This training set is generated by running a foreground detection algorithm (such as the one presented in the first half of this chapter) on all cameras, and retaining all foreground pixels. Writing each (Y_c, Cb_c, Cr_c) foreground pixel value from camera c as $p_c(x, y)$, and the mean value of *all* foreground data from this camera as $\bar{p}_c$, the first-order corrected values are calculated as

$$p'_c(x, y) = \frac{128}{\bar{p}_c} p_c(x, y) \tag{4.7}$$

This enforces a mean of 128 for each channel of the signal, *i.e.* mid-scale luminance and neutral chromaticity.

4.2.2 Second Order

It is also possible to consider the covariance, when equating the per-camera distributions of foreground colour data. If a particular value of colour signal $p_c(x, y)$ from camera c is represented as a different value $p_d(x, y)$ in camera d, then the general relationship between these spaces may be written as an affine transform

$$p_d = R_{cd} p_c + \mathbf{t}_{cd} \tag{4.8}$$

where R_{cd} and $\mathbf{t}_{cd}$ are respectively a generalised rotation and a translation between the input spaces of cameras c and d. If there is no mixing between the luminance and two chromaticity channels, then only the diagonal elements of R_{cd} will be nonzero. More generally, there may be mixing between the colour channels, corresponding to some small generalised rotation between the axes of the colour space for each camera. The normalisation process is intended to identify and correct for these differences between the signals from various cameras that presents all input signals in a common representation (the pseudo-code for this operation can be seen in Algorithm 4.1). If the affine model is a valid representation of the differences between the colour response from any two cameras, and there is sufficient structure to the covariance structure of the input statistics, then it is possible to estimate R_{cd} and $\mathbf{t}_{cd}$ from an unlabelled training set of foreground data. This is described in the following section.

Algorithm 4.1 Colour normalisation: takes a set of video streams as input, and normalises them using second order statistics on foreground pixels.

```
Require: cameras: a set of video streams
  for all c in cameras do
    count ⇐ 0
    for all f in frames(c) do
      for all p in pixels(f) do
        if foreground(p) then
          sum[c] ⇐ sum[c] + p
          sumsqr[c] ⇐ sumsqr[c] + p^2
          count[c] ⇐ count[c] + 1
        end if
      end for
    end for
    mean[c] ⇐ sum[c] / count
    globalMean ⇐ globalMean + mean[c]
  end for
  globalMean ⇐ globalMean / length(cameras)
  for all c in cameras do
    for all f in frames do
      f ⇐ f · global / mean[c]
    end for
  end for
```

4.2.2.1 Required Covariance Properties

If two multivariate random variables, X and Y, are related via a linear transformation $\mathbf{y} = T_{XY}\mathbf{x}$, then it may be possible to estimate T_{XY} from the means μ_X, μ_Y and covariances S_X, S_Y, generated from n samples of both X and Y. Simulations can be used to demonstrate the accuracy of the estimate, by generating a set of random samples from X, transforming them into Y using values of T, and then measuring how closely the two sets of vectors are aligned in some standard common co-ordinate system.

It is more convenient to transform them both onto the same "whitened" co-ordinate system with zero mean and unit diagonal covariance, as this will simplify subsequent formulas. If S_X diagonalises into $E_X \Lambda_X E_X^{\mathrm{T}}$, then the transform to the whitened version $\mathbf{w}_X$ of the vector $\mathbf{x}$ is

$$\mathbf{w}_X = \Lambda_X^{-1/2} E_X^{\mathrm{T}} (\mathbf{x} - \mu_X) \tag{4.9}$$

If the equivalent process is also applied to the variable Y, to obtain a sample of whitened vectors $\mathbf{w}_Y$, then the accuracy of the estimate can be measured as the expected L_2 distance between the whitened samples, *i.e.* $E\left[|\mathbf{w}_Y - \mathbf{w}_X|^2\right]$.

The accuracy of the estimate depends weakly on the number of samples n and

strongly on the relative magnitudes of the ellipsoid radii (*i.e.* eigenvalues) of the covariance structure associated with X. For the two-dimensional ellipse, the standard term used to describe these relative magnitudes is *eccentricity*, which varies between 0 (circle) and 1 (a line). In this work, an alternative term is used that is better suited to the investigation and is not limited to two dimensions: the *ellipticality*, ϵ, is defined as the smallest ratio between successive eigenvalues, ordered by size, and can vary between 1 (a sphere) and ∞ (an ellipsoid with at least one degenerate dimension).

Experiments were conducted to measure the reconstruction accuracy, using several test sets consisting of between 200 and 5,000 samples. Values of ϵ from between 1 and 5 were used, creating covariance matrices with the following form:

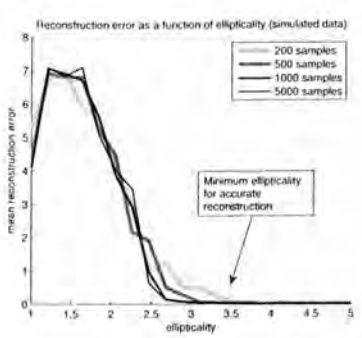
$$S = \begin{bmatrix} \epsilon^2 & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Figure 4.7a shows how the accuracy of the estimate of S is a function of ϵ for four different sample set sizes, and for $\epsilon = 1 \ldots 5$. The graphs show that accuracy increases with increasing values of ϵ, and that the increase in speed is higher with bigger sample-set sizes. Below values of $\epsilon = 2.5$, the alignment of the two sets of data failed completely and the accuracy was no better than random. Above values of $\epsilon = 3.0$, the alignment worked very well and the mean squared error between the two sets of whitened vectors rapidly approached zero. In the range $2.5 < \epsilon < 3.0$, the alignment process required a large number of samples (around 5,000) to obtain an accurate estimate. In the next section, the values of ϵ are estimated for the real video signals encountered in a surveillance system.

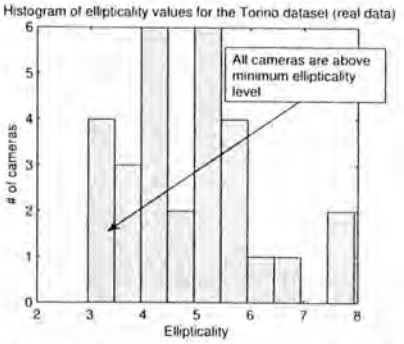
4.2.3 Analysis of Input Signal Covariance

The statistics of 29 surveillance video feeds from the Torino Metro system were analysed. Approximately 10 minutes per camera were used; only foreground pixels (extracted using a GMM method) were included. If the eigenvalues of the covariance of these data are sufficiently different, then the simulations suggest that the procedure outlined in the previous section may be applied to improve the alignment of these colour signals.

As shown in Figure 4.7b, the values of ϵ are all between 3 and 8 (we remind the reader that because of how ϵ is derived, its value is defined up to a scale factor, and should only be compared to other ϵ values obtained with the same scale factor - in this case, 1). This ensures that the cameras covariances can



(a) *Accuracy in the reconstruction of the mean μ and covariance S of a normal distribution, as a function of the ϵ of S, for 4 sample-set sizes.*



(b) *Histogram showing all ϵ values of the Torino dataset. In total 29 videos were used, giving an average ϵ value of* 4.8 ± 1.6.

Figure 4.7 – *Ellipticality estimates of a real surveillance dataset.*

be reconstructed with good accuracy. For the experimental validation of these colour correction methods, see Section 4.2.4.

4.2.4 Results

The experimental validation of colour correction methods is carried out by comparing the performance of appearance descriptors applied to the same set of targets, but using differently colour corrected video streams. The baseline is given by the original, non-colour corrected set of streams. The performance of the descriptors is measured using their *entropy* (see Section 3.5.1).

Figure 4.8 shows the histograms of three appearance descriptors, with and without colour correction. As evidenced by the figure, a descriptor's performance does not always benefit from first-order colour correction. However, second-order colour correction noticeably improves descriptors ability of discriminating between individuals, increasing the entropy from 13% to 16% for the Colour Position descriptor, and from 6% to 12% for the Mean Colour descriptor.

4.3 Conclusion

This chapter introduced two novel algorithms for the pre-processing stages of the TNT pipeline: foreground detection and colour correction. For the foreground detection stage, an algorithm was developed that can detect periodic variations in a scene background, and use them to better model the background by predicting the colour a pixel is going to assume in a subsequent frame. For

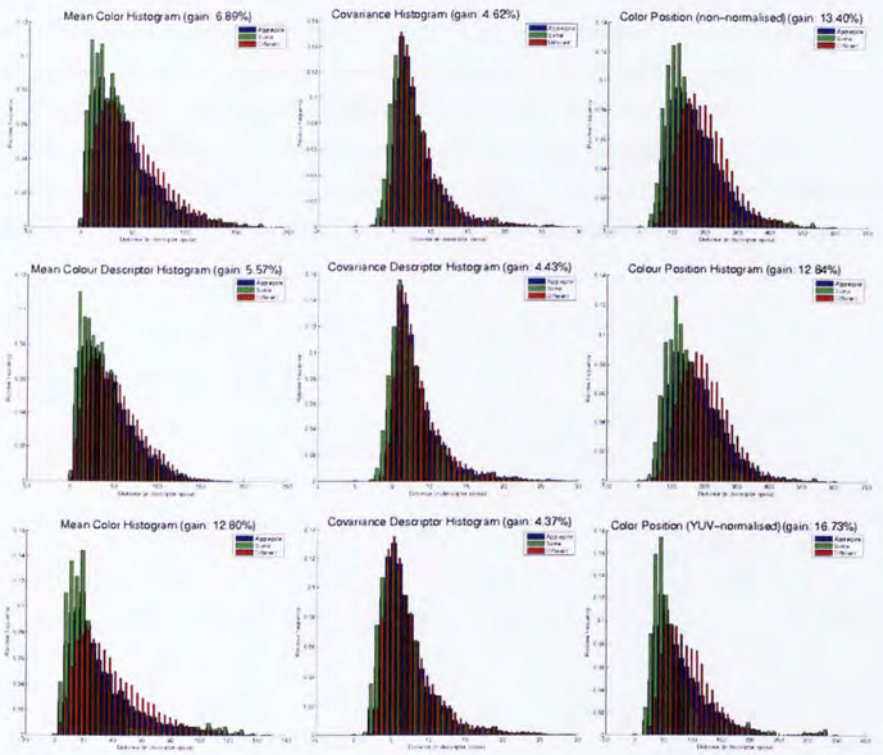


Figure 4.8 – *Variation of descriptors performances as different colour correction techniques are applied to the video streams. Each column corresponds to an appearance descriptor (*left*: mean colour descriptor;* centre*: covariance descriptor;* right*: colour position descriptor). Each row corresponds to a colour normalisation method (*top row*: no normalisation;* middle row*: first order normalisation;* bottom row*: second order normalisation). Above each histogram, the entropy provided by that specific combination of appearance descriptor and colour normalisation is displayed (a higher entropy corresponds to a better performing descriptor, see Section 3.5.1). The histograms were generated using over 100 observations of 12 individuals.*

the colour normalisation stage, an unsupervised method was developed that can learn systematic variations in colour responses between cameras using the motion of pedestrians in the surveillance network. Each algorithm has been evaluated on real data, showing an improvement in performance over the state of the art. However, both proposed algorithms have a downside, in that they require an explicit "training" phase before they can be used.

This is especially a problem for the motion detection algorithm, since most state-of-the-art algorithms (including the Gaussian Mixture Model used for the evaluation) do not require an separate training stage, and they can simply adapt to lighting and background changes as they run. It is for this reason that the period background detection algorithm has not been developed any further and has not been used for evaluation of the tracker. The next chapter presents the performance evaluation of the TNT system as a whole, using the GTT dataset.

Chapter 5

Experiments and Results

In this chapter, the experimental framework and the datasets used to test the tracker are presented (Section 5.1), followed by tracking results in Section 5.2. Since the main contribution of this thesis is a multi-camera, single-target tracker that can incorporate information from heterogeneous data sources, the evaluation will focus on comparing the relative performance of the multi-camera tracker with all possible combinations of data sources.

Before presenting the experimental results, an overview of the experimental process is given here. A data-flow diagram of the process is shown in Figure 5.1. The diagram is divided in four sections: prior knowledge, video processing, format conversion, and evaluation. The *Prior Knowledge* section covers the collection of all data and metadata that needs to be available before the tracker can be run, and is presented in detail in Section 5.1.

Once all data has been collected, and all metadata is made available, it is possible to run the tracker, as shown in the *Video Processing* section of the diagram. In a real scenario, the tracker would require a human operator to select a reference target and start the tracking; however, for performance evaluation purposes, a set of ground truth tracks are needed. These can be created with a purpose-built application, *KanAnnotate*, created by the CARETAKER group in Kingston University and described in Section 5.1.4. Both KanAnnotate and the tracker can be seen in the *Video Processing* layer of Figure 5.1 (the tracker is indicated as *cvplay*, since this is the name of the executable that runs it).

The next step after tracking is *format conversion*, where metadata produced by the tracker and the annotation tool is converted to a format suitable for the evaluation tool. There are three reasons why a separate data conversion

format is preferable: firstly, for ease of implementation, the output format of the tracker was chosen to match as closely as possible (within the limits of the *viper* format) the internal data structures of the tracker itself. Secondly, an external conversion tool can be used both with the ground truth and the tracking results, which is more cost-effective than writing several output modules for the tracker and the annotation tool (in fact, while this would be possible for the tracker, the annotation tool is hardcoded to use the *viper* format). Thirdly, a separate conversion step allows one to run the tracker once, and use the same data with multiple evaluation tools. The evaluation step, which is at the end of this data flow diagram, is explained in Section 5.1.5.

5.1 Prior Knowledge and Data Acquisition

Two datasets have been used for performance evaluation: the Torino dataset, collected for the CARETAKER project, and the ILIDS dataset, produced by the Home Office Scientific Development Branch (HOSDB). A third dataset, from Roma, was also collected during the CARETAKER project, but it has not been used for performance evaluation since is was found to have too poor video quality. Specifically, all the videos were blurred and desaturated, and two of the cameras (out of 12) had problems with brightness adjustment, causing their videos to be almost completely white.

During the 2009 Advanced Video and Signal-based Surveillance (AVSS) conference in Genova, Italy, the American National Institute for Standards and Technologies (NIST) organised a multi-camera tracking challenge, using the ILIDS dataset. The tracker presented in this thesis was one of the only two participants. Both trackers performed poorly, yielding negative MOTA values (see Section 2.3.1.4). It became apparent to the contest organisers, as well as to the other conference attendees, that the ILIDS dataset is extremely challenging. The main difficulties, as it was noted at the conference, were the high level of crowdedness and the low placement of cameras, which caused a high level of dynamic occlusions between the targets. Because of all these reasons, the ILIDS dataset was not used for performance evaluation in this chapter.

5.1.1 Video Sources

A realistic use-case requires the tracker to work in real-time with video streaming live from the surveillance cameras. The standard protocol for video streaming is RTP (real-time transport protocol, see Nomenclature). Within the CARE-

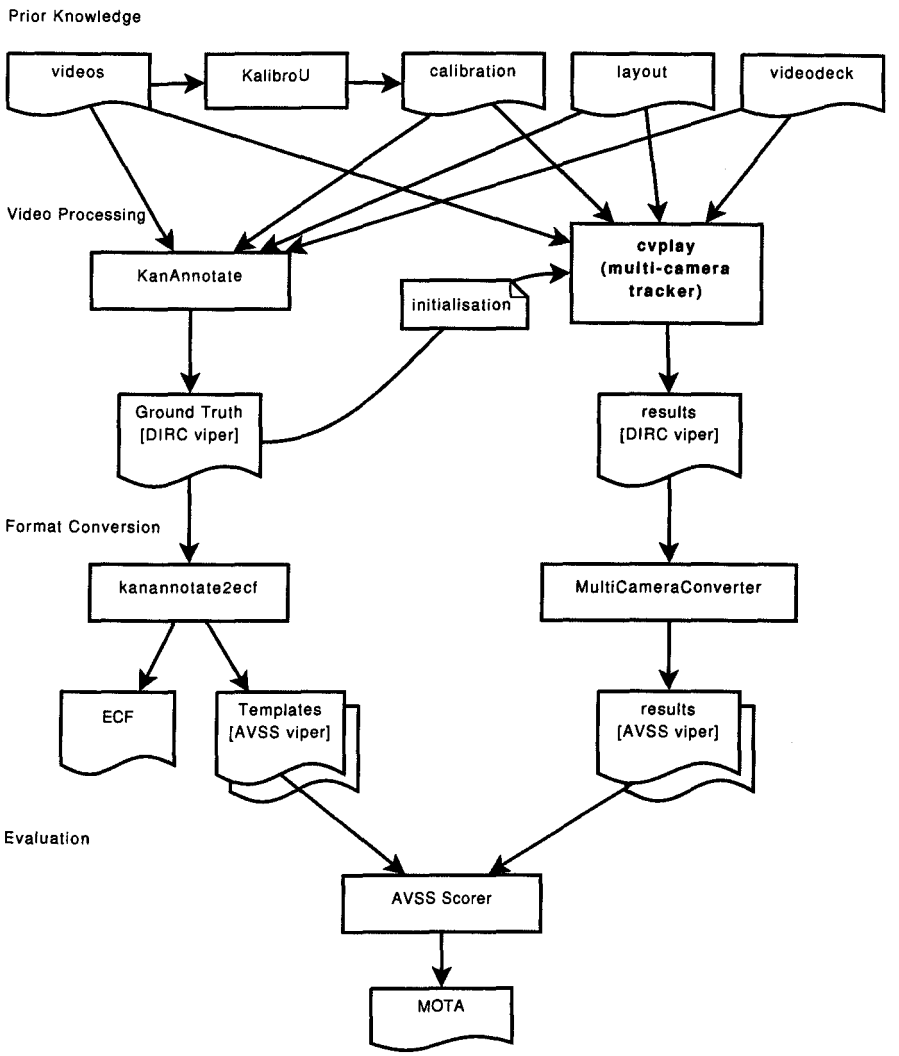


Figure 5.1 – *Data-flow diagram of the multi-camera tracking system, including setup and evaluation. Square boxes represent processing, and "wavy" boxes represent data.*

TAKER project, RTP reception functionalities were implemented using a library provided by Thales[1], a member of the CARETAKER consortium.

Testing and debugging use-cases require the tracker to be run on stored data, so that experiments may be repeatable. Also, a possible use-case of off-line tracking (e.g. after an incident) requires the tracker to work on stored videos. The CARETAKER system uses a data base (DB) to store video, in order to facilitate random search within large archives of data (several[2] days of videos from hundreds of cameras): DB access was therefore required, and was implemented with the collaboration of Solid Information Technology (a database company, now acquired by IBM, that was a member of the CARETAKER consortium). Finally, considering that file access is a very common way of storing and exchanging video data, file access was also implemented.

In order to keep the software modular and simple, all three video sources use the same interface, namely the OpenCV `CvCapture` pseudo-object. Standard objects for file access are already provided by OpenCV, while custom ones were implemented for RTP and DB access.

An XML schema, called *videodeck*, was designed to describe the heterogeneous set of resources that represent a data capture session (including a live session as a special case). This schema allows authors to associate a camera (uniquely identified by a number) with one or more data sources. Each data source can be either file, database, or live stream. In the case of file and database sources, the optional attributes *start*, *end* and *sync* are also available.The *start* and *end* attributes can be used to specify which subset of the data source contains valid video data, in the case that corrupted video was recorded (possibly due to a network error or to a camera malfunction); if they are not specified, the data source is assumed to contain only valid video. The sync attribute allows the media time of each source to be synchronised with a global timeline. If videos from all data sources are recorded at the same time, then the *sync* value is zero (the default). However, non-zero *sync* values are used extensively in the GTT dataset, since all files in each dataset start at a different time (approximately within 30 seconds of each other). A simple OCR program was developed by the author of this thesis to read the timestamp overlaid on the video and compute *sync* values.

A Videodeck file does not contain any information about the cameras themselves, such as calibration or adjacency: this is covered by the *topology* schema, presented in Section 5.1.3. An example of a Videodeck file is shown in Figure 5.2.

[1] `http://www.thalesgroup.com`

[2] Italian laws require surveillance data to be erased after 7 days, unless an event of interest has happened and permission to retain data has been granted by the police.

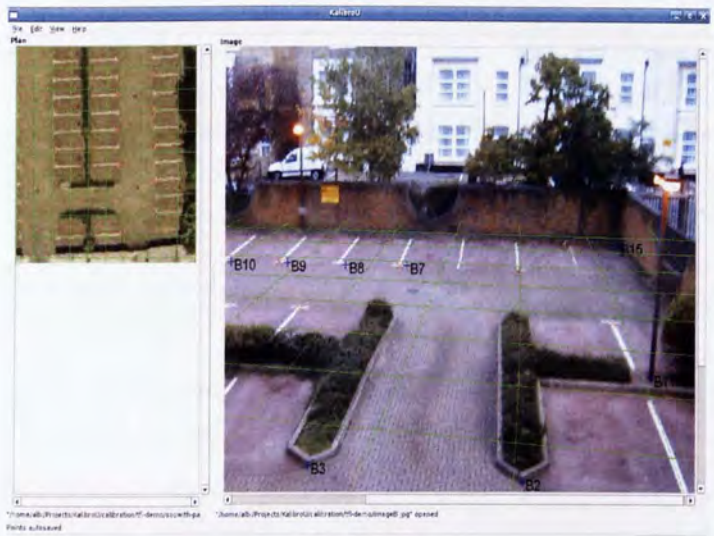
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<ka:deck xmlns:ka="uk.ac.kingston.dirc.kanannotate">
  <!-- This is a modified version of solid-video.xml, synchronised to the frame -->
  <db-default password="[redacted]" server="[redacted].202.123" sync="0" username="[redacted]"/>
  <view camera="sen-300">
    <db sync="-9.399999999994179"/>
  </view>
  <view camera="sen-301"/>
  <view camera="sen-302"/>
  <view camera="sen-617"/>
  <view camera="sen-618"/>
  <view camera="sen-902">
    <db sync="-4.699997999996413"/>
  </view>
  <view camera="sen-903">
    <db sync="-12.199995999995735"/>
  </view>
  <view camera="sen-906">
    <db sync="-8.199995999995735"/>
  </view>
  <view camera="sen-907">
    <db sync="-5.866663999993762"/>
  </view>
  <view camera="sen-908">
    <db sync="-12.533331999999064"/>
  </view>
  <view camera="sen-909">
    <db sync="[illegible]"/>
```

Figure 5.2 – *A Videodeck file. This file uses database sources; cameras may not have an associated data source to indicate that there is no video available.*

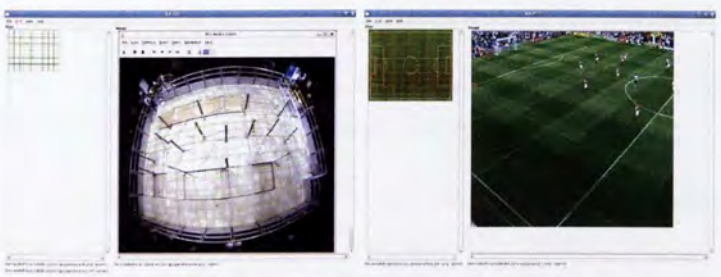
5.1.2 Calibration

Even though many semi-automatic camera calibration methods exist in the scientific literature (see e.g. Section 2.5), it was decided that research should focus on tracking, and therefore a traditional manual calibration method was adopted. A simple camera calibration software named *KalibroU* (see Figure 5.3) was developed the Kingston University CARETAKER team that uses the Tsai coplanar method [107, 108] to calibrate cameras.

KalibroU can be used to calibrate a surveillance camera with respect to a map of the surveilled area, if an image of the map and a frame of the camera view are available. The software allows the user to click on corresponding points in the images and provides a grid to be used as a guideline. When at least five non-collinear, coplanar points are selected, a grid is displayed on both images to let the user judge the quality of the calibration, and as the mouse hovers in one image, its corresponding position is displayed in the other one. If the user is not satisfied with the calibration, points can be added or moved about, until the calibration is deemed satisfactory. When calibration quality is visually good enough, the estimated camera parameters can be saved as an XML file, that can be loaded by an external application. Even though the procedure is completely manual, a camera can be calibrated in 5-10 minutes, which made it possible to calibrate the 7 cameras used in the Rome test-site and the 54 used in Turin (note that because all stations in Torino underground are alike, only



(a) *Car park*



(b) *PAMELA (University College London) simulation/recording facility*

(c) *Football pitch*

Figure 5.3 – *Three screenshots of KalibroU running under Linux.*

18 cameras have actually been calibrated, and the calibration has been used for three different stations).

5.1.3 Topology and Layout

The Kanannotate schema is used to represent the network layout as a graph, where each node represents a camera and each arc represents a connection between cameras, as described in Section 2.5.2. The graphs representing the topologies for the three test sites (Roma, Torino and the i-Lids dataset) were created by hand, with approximate times estimated by looking at the videos. Automatic graph-inference methods were discarded for the same reasons why camera self-calibration methods were discarded (see Section 5.1.2, namely that

research should focus on tracking, not on topology calibration, as errors in topology estimation must not be brought forward to the tracking stage).

The above representation of the network layout has to be stored in a structured file for easy human and programmatic access. XML is the grammar we chose for the implementation. There exists an unofficial W3C syntax recommendation for serialising graphs into XML [57] that we used to design the *Topology Schema* shown in Figure 5.4. A Topology XML file consists of four sections:

image whose elements are simply links to image files containing the map of a level of a station.

camera whose elements represent the cameras of the network, and link to a file containing the calibration data for that camera.

coterie that is a collection of cameras sharing the same ground plane. Each coterie contains two or more cameras and a reference to the **image** element that contains the ground plane of that coterie; additional attributes of the **coterie** tag are used for display purposes.

connection whose elements link pairs of cameras and specify the average transit time in seconds using the **occluded_for** attribute.

Figure 5.5 shows the Torino layout file as an example. Both the schema design, and all layout files (for Roma, Torino, and the i-Lids datasets) were created by the author of this thesis for the tag-and-track project.

5.1.4 Ground Truthing

Because performance evaluation of tracking is based on comparing the tracker results with a manually generated ground truth, annotations had to be made for the Torino dataset (whereas the i-Lids dataset is shipped with ground truth files). Ground-truthing a large dataset such as the Torino one is a task that can take a person many months to complete, as it requires navigating through half an hour of video over 200 cameras, looking for suitable targets in various crowding conditions, and tagging their locations.

Therefore, a new annotation software, named *KanAnnotate*, was developed by the Kingston University CARETAKER team[3] specifically for this purpose. KanAnnotate uses the topology file format defined above to construct a visual representation of the connections between cameras, that can be navigated using

[3]Most of the credit and gratitude, though, go to my colleague Justin Cobb.

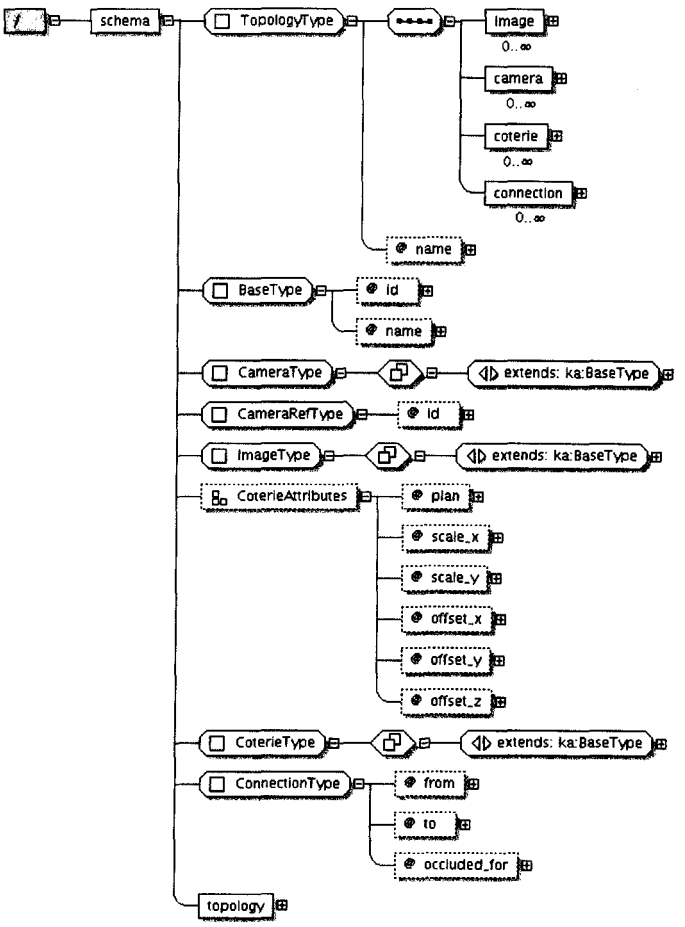


Figure 5.4 – *A graphical depiction of the* Topology *XML Schema, representing the layout of a camera network. The term* KanAnnotate *comes from the name of the first software tool that used it.*

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<ka:topology xmlns:ka="uk.ac.kingston.dirc.kanannotate" name="Torino Network Layout">

    <camera_defaults width="704" height="288"/>

    <image file="XVIII_Dicembre__atrio__01.png" id="img-01" name="DOD Upper Level"
     offset_x="0.00" offset_y="0.00"
     scale_x="1.00" scale_y="1.00"
     pixels_per_metre="10.58"/>
    <image file="Racconigi__banchina__1.png" id="img-02" name="Platform Level (all stations)"
     offset_x="0.00" offset_y="0.00"
     scale_x="1.00" scale_y="1.00"
     pixels_per_metre="17.4"/>

    <!--DOD-->
    <!-- calibrated cameras -->
    <camera calibration="Acc-Stazione-DOD.xml" id="sen-1001" name="Entrance, Station Side"/>
    <camera calibration="Acc-Cernaia-DOD.xml" id="sen-1000" name="Entrance, Via Cernaia Side"/>
    <camera calibration="Ticket-1-DOD.xml" id="sen-1006" name="Ticket Machine 1"/>
    <camera calibration="Ticket-2-DOD.xml" id="sen-1007" name="Ticket Machine 2"/>
    <camera calibration="Ascensore-atrio-1-DOD.xml" id="sen-1002" name="Hall Lift 1"/>
    <camera calibration="Ascensore-atrio-2-DOD.xml" id="sen-1003" name="Hall Lift 2"/>
    <camera calibration="Tornelli-DOD.xml" id="sen-1012" name="Turnstyles"/>

    <!-- Coteries -->
    <coterie id="cot-dod-01" plan="img-01" name="DOD Hall" level="0.0">
        <camera id="sen-1001"/>
        <camera id="sen-1000"/>
        <camera id="sen-1006"/>
        <camera id="sen-1007"/>
        <camera id="sen-1002"/>
        <camera id="sen-1003"/>
        <camera id="sen-1012"/>
        <camera id="sen-1008"/>
        <camera id="sen-1009"/>
    </coterie>

    <coterie id="cot-dod-02" plan="img-02" name="DOD Platform 1" level="-1.0">
        <camera id="sen-1013"/>

    <connection from="sen-1000" occluded_for="0.00" to="sen-1006"/>
    <connection from="sen-1006" occluded_for="0.00" to="sen-1000"/>
    <connection from="sen-1001" occluded_for="0.00" to="sen-1007"/>
    <connection from="sen-1007" occluded_for="0.00" to="sen-1001"/>
    <connection from="sen-1006" occluded_for="0.00" to="sen-1012"/>
    <connection from="sen-1012" occluded_for="0.00" to="sen-1006"/>
    <connection from="sen-1007" occluded_for="0.00" to="sen-1012"/>
    <connection from="sen-1012" occluded_for="0.00" to="sen-1007"/>
    <connection from="sen-1002" occluded_for="1.00" to="sen-1007"/>
    <connection from="sen-1007" occluded_for="1.00" to="sen-1002"/>
    <connection from="sen-1003" occluded_for="1.00" to="sen-1006"/>
    <connection from="sen-1006" occluded_for="1.00" to="sen-1003"/>
    <connection from="sen-1011" occluded_for="4.00" to="sen-1003"/>
    <connection from="sen-1002" occluded_for="4.00" to="sen-1010"/>
    <connection from="sen-1003" occluded_for="4.00" to="sen-1011"/>
    <connection from="sen-1003" to="sen-1012"/>
```

Figure 5.5 – *A topology XML file. This example shows part of the layout definition for the Torino dataset (the complete file would be too long to include). The top section defines the list of coteries (with associated image files, see Section 5.1.3) and the list of cameras. The middle section associates cameras with coteries, and the bottom section defines connections between pairs of cameras.*

the mouse. It can also read videodeck files, that define the video sources, thus presenting the user with the same scenario (data sources and topology) that the tracker uses for tracking. The user can then use the mouse to select an individual and mark their positions in the videos.

5.1.5 Evaluation Methodology

Evaluation of the tracker performance was carried out using the MOTA metric, which was presented in Section 2.3.1.4 but is repeated here for easier reference:

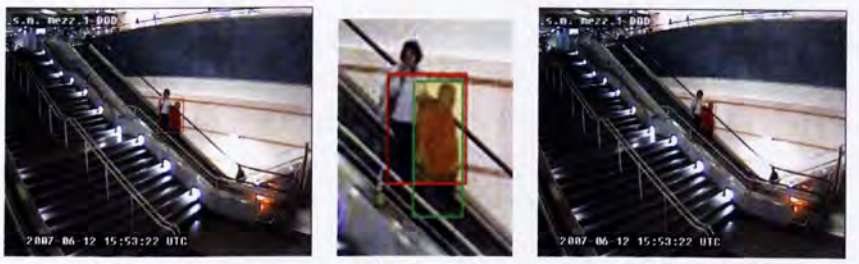
$$\text{MOTA} = 1 - \frac{\sum_{t=1}^{N_{frames}} (m_t + fp_t + \log_{10} \text{IDC})}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}, \tag{5.1}$$

where m_t is the number of missed detections, fp_t is the number of false positives, and IDC is the number of ID changes, calculated considering the difference between the mapping for frame t and the mapping for frame $(t-1)$.

The other evaluation measures reviewed in Section 2.3 are available only for the results submitted to the AVSS multi-camera tracking challenge, since the publicly available evaluation tool only outputs MOTA values. Possible values for the MOTA range from $-\infty$ to 1. A MOTA value of 1 represents perfect accuracy, i.e. all output bounding boxes overlap to the ground truth bounding boxes by more than a threshold, and there are no unmatched ground truth bounding boxes; a MOTA value of 0 means that there are as many correct bounding boxes as erroneous ones. Finally, values below zero mean that erroneous bounding boxes outnumber the correct ones.

When implementing a MOTA-based tool for evaluating the multi-camera tracker described in this thesis, the algorithm has been slightly simplified with the additional constraint of having a single-person tracker. In particular, tracking precision (MOTP) ceases to be meaningful, and therefore only MOTA was used for evaluation. Moreover, in the MOTA formula shown in Equation 2.7, $\log_{10}$ IDC is always 0 and can be ignored.

The major difference, however, lies on how false positives are counted. The original formulation of the MOTA assumes that ground truth is available for all frames where the target is visible in the video; therefore, if the tracker produces a bounding box for a frame where there is no ground truth defined, it is counted as a false positive. Early in the development of this project, however, it became apparent that producing frame-by-frame annotations of individuals would not have been feasible in the available time, especially if we wanted a significant number of individuals annotated. To make the tasks manageable by the author



(a) *Tracker output* (b) *Intersection* (c) *Ground truth*

Figure 5.6 – *Example frame of comparison between ground truth and tracker output to evaluate the MOTA. The intersection to union ratio is greater than 20%, and therefore this frame counts as a true positive.*

alone, ground truth has been collected only at selected frames (*ground truthed frames*), typically the entry, middle, and exit point of each individual in each camera. The first and last frames of the ground truth are used to define a temporal window where the target is considered visible; if the tracker produces bounding boxes outside of this window, they are counted as false positives, but inside this window the tracker is evaluated only in those frames where the ground truth is defined, and other bounding boxes are simply ignored.

The last difference from the original formulation of the MOTA consisted of running a linear interpolation algorithm to the tracker output. Interpolation is done on each camera independently; the algorithm looks for discontinuities (i.e. missing frames) in the output, and fills them in by linearly interpolating the available bounding boxes. This phase was added for the following reason: if there are overlapped cameras, the ground truth is typically available for all cameras in the overlap area, but the tracker can only produce results for one camera at a time (this design decision is explained in Section 3.3). On the other hand, if there is a good view of the target in more than one camera, the tracker is likely to perform frequent hand-overs (see Section 3.4); in this case, it may happen that the tracker does not provide a bounding box for a ground truthed frame, even though it provides bounding boxes for the frames before and after. In order not to penalise the tracker for a fortuitous hit-or-miss of a ground truthed frame, the interpolation phase was added.

Figure 5.6 shows an example of comparing ground truth and tracking results to compute the MOTA for one video frame.

5.2 Tracking Performance

The tracker was demonstrated live at the final review of the Caretaker project, showing tracking on both recorded and live video streams. However, the demonstration was not intended to provide a quantitative evaluation of the tracking algorithms. This section provides quantitative results for the Torino dataset, allowing to assess performance of each module of the TnT system and to determine the best combination of modules.

5.2.1 System Parameters

The tracker performance was evaluated varying a number of parameters, shown in Table 5.1.

5.2.2 Comparison of Components

These experiments aim to compare the performance of the three main modules of the multi-camera tracker (foreground detectors, appearance descriptors and single-camera trackers). The experiments were performed by keeping one of the components fixed, while changing all the others, and running the multi-camera tracker on all of our 11 ground-truthed individuals. Histograms of the resulting MOTAs are provided.

Statistical significance tests are performed for each module being evaluated, in order to find out which differences in MOTA are due to a better component, and which are just noise. For this, we assume that the average MOTA produced by an experiment is normally distributed with unknown variance (the Central Limit Theorem [115] ensures that this assumption is valid for large enough datasets).

A two-tailed *Student's t-test* [116] is then applied to the collected data, comparing the hypothesis "two components have different performances" against the null hypothesis that the "two components have the same performance". We reject the null hypothesis if the confidence is greater than 95%. For each module, we provide a summary table that compares each pair of components; cells where confidence is greater than 95% are highlighted with a green background (text is bold if confidence is greater than 99%).

Individuals The first experiment was run to compare the average MOTAs produced by different individuals. Although individuals are not a component of the multi-camera tracker, it is useful to provide a visual representation of how

Parameter	Description	Values or range
Motion detection	the algorithm used for foreground segmentation (see Section 2.4.1 and 2.4.3)	Double mixture model (GMMHSV), Horprasert, Mixture of Gaussians as provided by OpenCV (std-mog)
Tracker	The algorithm used for single-camera tracking (see Section 2.2)	Dense optical flow (DenseFlowTracker), template-based (DescriptorTracker), OpenCV camshift (HueShift), Lukas-Kanade (OpticFlowTracker), combination of template-based and dense optical flow (CombiningTracker)
Appearance descriptor	(See Section 3.7)	MeanColor, ColorPosition
measurement noise	Kalman filter measurement noise	$[0..\infty]$, typically [50..5000]
process noise	Kalman filter process noise	$[0..\infty]$, typically $[10^{-3}..10^{-5}]$
Minimum $p(i \equiv j \mid S_{ij})$	Probability of a match given two spatio-temporal observations (minimum probability to accept identity)	[0..1], typically [0.9..0.95]

Table 5.1 – *Tracker parameters.*

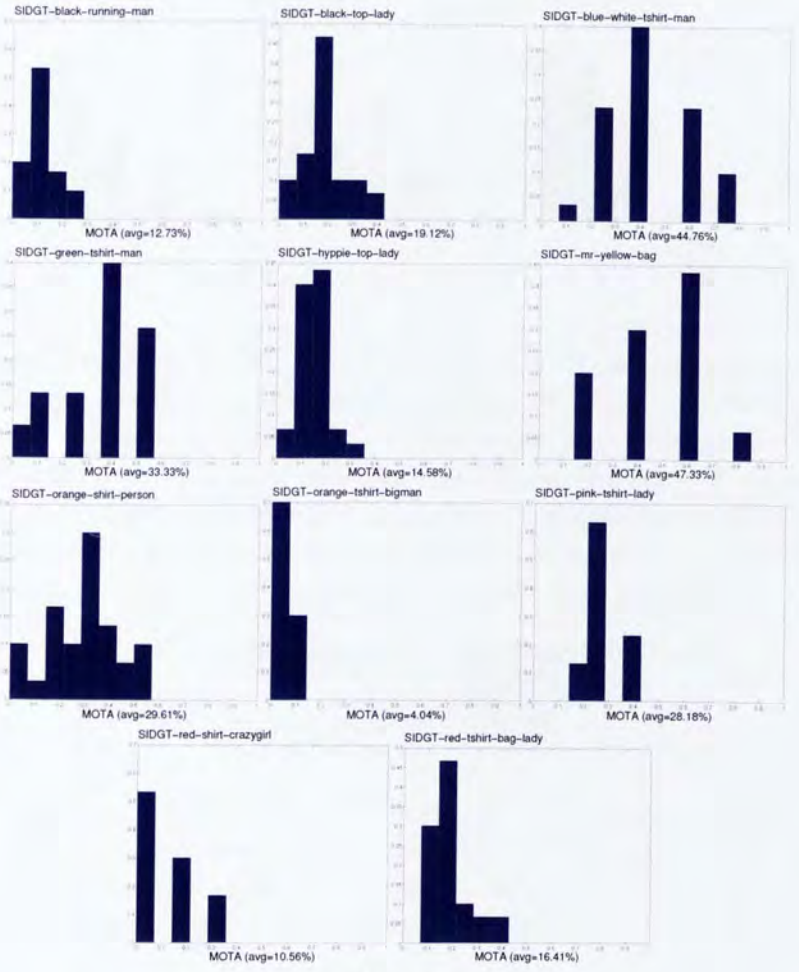


Figure 5.7 – *MOTA by individual.*

much variation in performance is due by the individual being tracked, and how much by the system tracking it.

As it can be seen from Figure 5.7, different individuals tend to produce quite different MOTAs; this is due to different level of crowdedness, different paths taken by the individuals, and other factors that make tracking specific individuals more or less difficult. For some individuals (i.e. "SIDGT-black-running-man"), all tracker configurations produce similar results. For other individuals, changing the tracker configuration has drastic effects on tracking performance.

Motion Detectors In this experiment, the three motion detectors are compared. For each motion detectors, MOTA values are generated by running

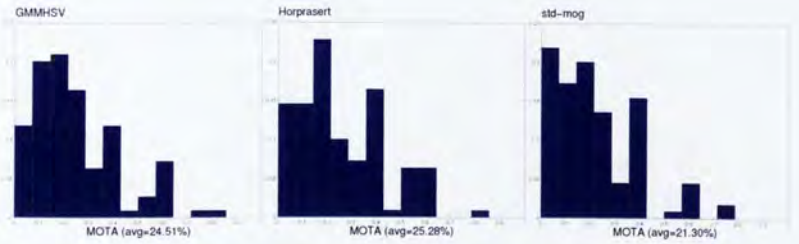


Figure 5.8 – *Comparison of motion detectors.*

the multi-camera tracker with all combinations of appearance descriptors and single-camera trackers. Results are shown in Figure 5.8. In the table below, and in subsequent tables in the remainder of this chapter, each cell corresponds to the confidence that two tracking modules have inherently different performance, rather than differing by chance. This confidence is expressed in percentage and is calculated using a *Student's t-test*, as explained in Section 5.2.2. The two modules a cell refers to are below and to the left of the cell itself.

GMMHSV	43.24	98.41
Horprasert		**99.37**
std-mog		

This experiment indicates that the best motion detectors are the Gaussian Mixture Model with HSV-based shadow removal (GMMHSV) and the Horprasert method, as there is no significant difference in the results produced by them. Both are significantly better than the Mixture of Gaussians method.

Trackers In this experiment, the five single-camera trackers are compared. For each single-camera tracker, MOTA values are generated by running the multi-camera tracker with all combinations of appearance descriptors and motion detectors. Results are shown in Figure 5.9.

CombiningTracker	19.88	96.32	**99.62**	**99.75**
DenseFlowTracker		95.35	**99.79**	**99.88**
DescriptorTracker			86.14	59.39
HueShift				43.98
OpticFlowTracker				

The Student's t-tests shown above seem to split single-camera trackers into two groups of distinct performances: on the lower end, the Combining Tracker and the Dense Flow Tracker, producing a MOTA of approximately 20%, and, on the

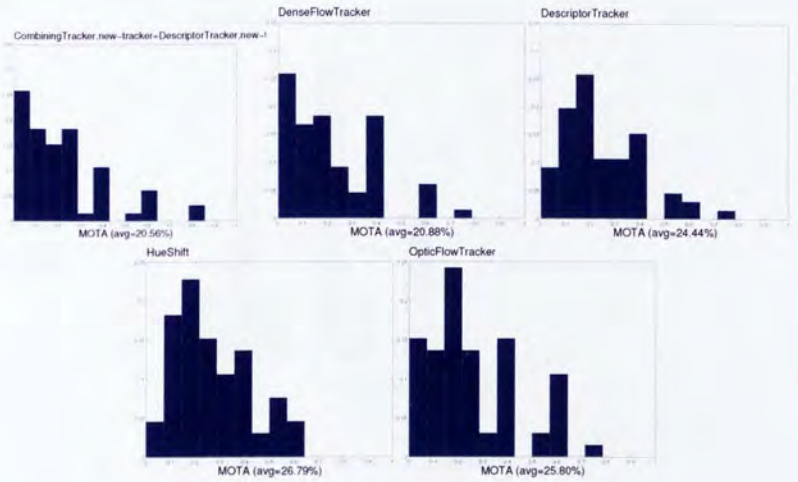


Figure 5.9 – *Comparison of trackers.*

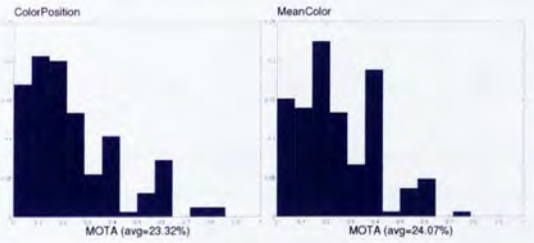


Figure 5.10 – *Comparison of appearance descriptors.*

upper end, the Descriptor Tracker, the Hue Shift and the Optic Flow Tracker, all providing a MOTA of around 25%. Performance differences within these groups are not statistically significant.

Appearance Descriptors In this experiment, the two appearance descriptors are compared. For each appearance descriptor, MOTA values are generated by running the multi-camera tracker with all combinations of single-camera trackers and motion detectors. Results are shown in Figure 5.10.

ColorPosition	44.99
MeanColor	

Although the MeanColour descriptor yields a slightly higher MOTA, the difference between the two descriptors is not statistically significant.

Best of Breed The histogram in Figure 5.11 shows the results of running the tracker on all 11 individuals using the best single-camera tracker, appearance

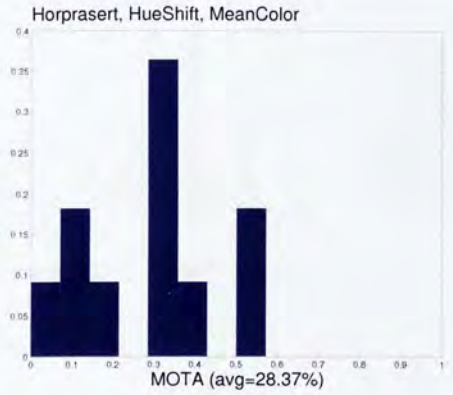


Figure 5.11 – *MOTA obtained running the multi-camera tracker with the best segmentation, appearance description and single-camera tracking modules.*

descriptor and motion detector, as determined by the experiments above.

5.2.3 Direct Comparison Using a Best-of-Breed System

In this section, the three main modules of the multi-camera tracker are evaluated again, but this time experiments evaluating one module will not use all combinations of the other modules. Instead, based on the results from the previous section, a "best" configuration is created using the best-performing modules.

Motion Detectors In this experiment, the three motion detector modules are compared by running the multi-camera tracker on all 11 ground-truthed individuals. The appearance descriptor and the single-camera tracker are MeanColor and HueShift respectively. Results are shown in Figure 5.12.

GMMHSV	43.24	98.41
Horprasert		**99.37**
std-mog		

Again, the best-performing modules are GMMHSV and Horprasert, with no statistically significant difference in performance between the two.

Trackers In this experiment, the three single-camera tracker modules are compared by running the multi-camera tracker on all 11 ground-truthed individuals. The appearance descriptor and the motion detector are MeanColor and Horprasert respectively. Results are shown in Figure 5.13.

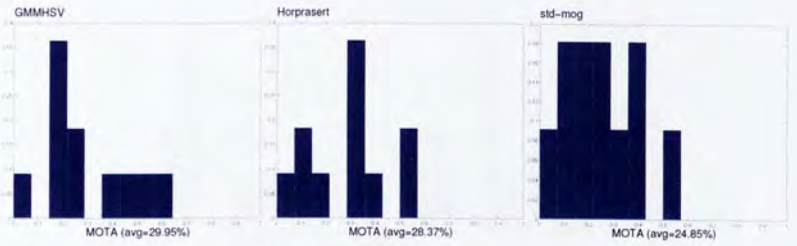


Figure 5.12 – *Comparison of motion detectors.*

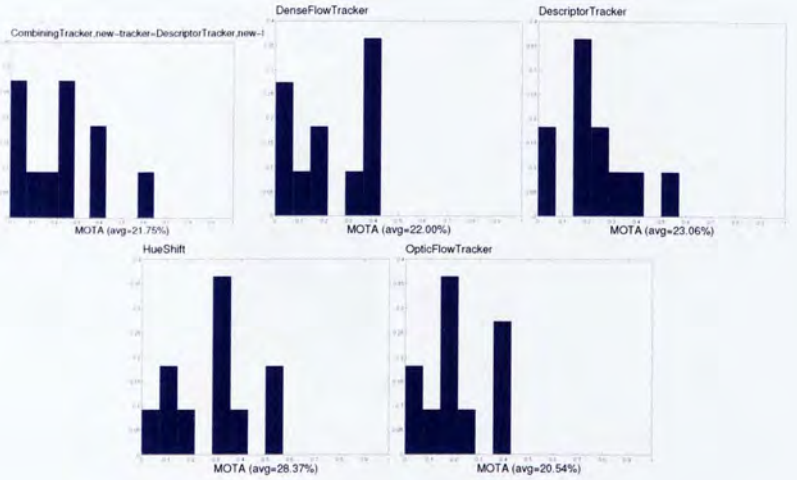


Figure 5.13 – *Comparison of trackers.*

CombiningTracker	6.00	22.32	63.53	31.06
DenseFlowTracker		20.72	73.32	42.44
DescriptorTracker			69.42	52.16
HueShift				84.66
OpticFlowTracker				

While overall MOTA values have increased, and the highest one is still provided by HueShift, it can be noticed that now no single-camera tracker is significantly better than any other.

Appearance Descriptors In this experiment, the two appearance descriptor modules are compared by running the multi-camera tracker on all 11 ground-truthed individuals. The single-camera tracker and the motion detector are HueShift and Horprasert respectively. Results are shown in Figure 5.14.

ColorPosition	44.99
MeanColor	

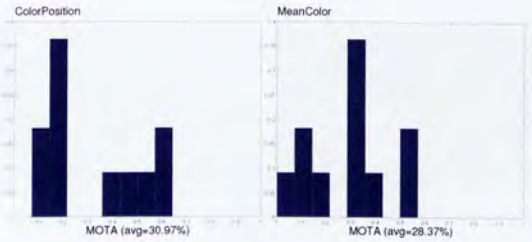


Figure 5.14 – *Comparison of appearance descriptors.*

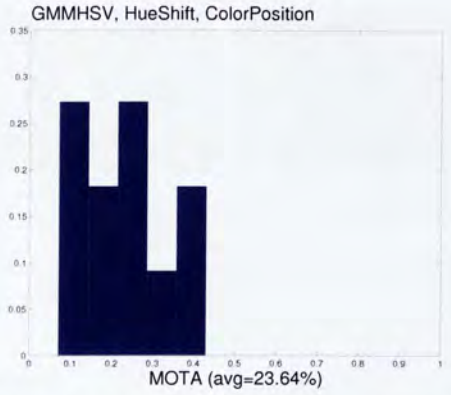


Figure 5.15 – *Best-of-breed multi-camera tracker using the modules that provide the highest MOTA as measured in Section 5.2.3.*

As with the previous experiment, there is no significant difference in performance between the two appearance descriptors.

5.2.4 Best Configuration

The histogram in Figure 5.15 shows the results of running the tracker on all 11 individuals using the components that have performed the best in the experiments of Section 5.2.3. We note that the MOTA provided by this configuration is noticeably lower than the one shown in Figure 5.11 (23% versus 28%). This result suggests that the interactions between modules change the performance of the multi-camera tracker in unintuitive ways, and combining the supposedly best-performing modules does not necessarily provide the best-performing tracker. While this is unexpected results that will be discussed in the next chapter, from a practical point of view one can still find the tracker with the best performance by comparing all possible combinations, as it was done in Section 5.2.2.

(a) *Tracker initialisation.* (b) *The target enters a highly crowded area.*

Figure 5.16 – *Tracking error: the target is lost in the crowd.*

(a) *Tracker initialisation.* (b) *The target operates a ticket machine for an extended period of time.*

Figure 5.17 – *Segmentation error: a target remaining motionless for a long time is incorporated into the background.*

5.3 Discussion

This chapter presented the results of performance evaluation on the multi-camera tracker, as well as presenting the datasets used for evaluation and development, and the additional tools used to set up the testing framework. The following paragraphs will provide examples of tracking errors, in order to highlight the difficulties of the task and to provide some "visual feedback" to the MOTA figures presented above. These will lead to the conclusion of this thesis in Chapter 6.

In the first example (Figure 5.16), the tracker is initialised to follow the gentleman in a green t-shirt entering the turnstiles (5.16a). As a small crowd concentrates around that area, the target becomes almost completely occluded (only his right shoulder is visible in Figure 5.16b). The tracker is unable to identify the target with only such a small portion visible.

The second example, shown in Figure 5.17, shows the tracker failing to track due to a segmentation error. In this example, the target spends a long time (approximately one minute) buying a ticket from the ticket machine (Figure 5.17b), and remaining almost completely motionless all the time. The motion detector, therefore, incorporates the target's appearance into the background model (a well known flaw of the GMM algorithm), thus making the target "disappear".

In the third, and final, example (Figure 5.18), the tracker is initialised to follow a

(a) *Tracker initialisation.*

(b) *A new target is acquired, with clothes similar to the original target's.*

(c) *The tracker re-identifies the initial target.*

(d) *The tracker settles on the wrong target.*

Figure 5.18 – *Tracking error: two targets of similar colours cannot be distinguished.*

person wearing a blue t-shirt (the *reference* target, highlighted in Figure 5.18a). The target is isolated and is wearing distinctively coloured clothes, thus creating an apparently simple tracking scenario. Please note the group of people standing on the left of the reference target. In Figure 5.18b, it becomes apparent that one of the persons who were standing on the left of the reference target is actually wearing clothes of the same colour as the reference target. Also, a minor segmentation error causes this person to appear much taller and nearer. Given the similarity and the proximity between the two targets, the tracker is unable to distinguish between them (Figure 5.18c). Eventually, the tracker settles on the wrong target (Figure 5.18d).

With these examples, some of the many ways in which tracking can fail were presented. It is clear that many improvements need to be made before real-time tracking on multiple, non-overlapping cameras becomes of any practical use. Suggested improvements, and the conclusion of this thesis, will be presented in the next chapter.

Chapter 6

Conclusions and Future Work

This thesis addresses computer vision algorithms for multi-camera tracking of pedestrians. The aim is to improve safety and security in public places, especially underground stations, by providing a tool that allows security operators to easily "keep an eye" on individuals deemed vulnerable or potentially dangerous. Multi-camera tracking in a large CCTV network is currently a difficult job, requiring the full attention of a specialised operator. The tracker presented in this thesis was developed as part of CARETAKER [102], an international EU-funded project to investigate the problem of content analysis and knowledge extraction on massive recordings. For ease of reading, a short summary of the work is provided in the next section. The following section will critically discuss the work, and an outlook for further developments will be provided in Section 6.3.

6.1 Summary of Work

As stated in the introduction, the main contribution of this thesis is a probabilistic framework for the construction of a real-time tag-and-track (TNT) system. This framework allows heterogeneous information about a target and a surveillance site to be fused in order to enable tracking over multiple, non-overlapping cameras. TNT systems allow a security operator to keep selected individuals always visible on a monitor as they move through a surveillance network consisting of any number (possibly hundreds) of fixed CCTV cameras with non-overlapping views.

The problem was decomposed in a number of modules for detecting moving

foreground, tracking individuals in single cameras, and comparing targets appearances, and for each module a state of the art survey was performed. Also, two novel methods were proposed for foreground detection and colour correction. The proposed foreground detection method exploits the presence of many periodically-changing background elements in indoor scenes (escalators, scrolling advertisements, flashing lights, ...); it detects and models pixels exhibiting periodic changes in colour, and uses this information to predict the pixel colour in subsequent frames in order to improve on foreground detection; experimental results show that the algorithm performs better than state-of-the-art Gaussian Mixture Model on scenes containing periodically changing pixels.

For the problem of comparing target appearances, a novel colour correction method was proposed that can learn differences in camera colour responses up to second-order statistics; the algorithm is completely unsupervised, and can automatically detect whether it has gathered enough information for a reliable colour correction; experimental results show a noticeable increase in the separability of appearance descriptions when second-order colour correction is used.

A probabilistic approach was chosen to integrate heterogeneous information coming from the different modules. Single-camera tracking, appearance descriptors, and prior knowledge on the camera network all contribute information about the identity of targets. The framework can combine this information and find a target that is globally most likely to be the one intended by the operator. The solution is re-computed at each frame, using only the state of the system at the previous frame, thus avoiding the computational overhead of optimising a long track and allowing the multi-camera tracker to work in real-time.

6.2 Discussion

The work started from state-of-the-art, single-camera, real-time tracking algorithms. A common, high-level interface was extracted from existing algorithms, so that a multi-camera framework could be developed independently from the chosen single-camera tracker. After extensive analysis of the state of the art of multi-camera trackers, it was found that none covered the user requirements of this project (real-time usage on a network of hundreds of non-overlapping cameras).

As noted in the literature review (Section 2.1.4), there are several approaches to multi-camera tracker architectures: centralised, distributed, and decentralised. A centralised architecture was chosen because it allows to increase the number of cameras without requiring additional processing power. The downside of this

architecture is that it requires additional computational power if more targets need to be tracked simultaneously, and it was deemed the best compromise, allowing the development of a demonstration system that can track a pedestrian on a network of hundreds of cameras using a standard desktop PC.

Geometric camera calibration can help tracking by constraining a target's size as a function of its position in the image, and vice versa. Cameras can be calibrated with respect to a virtual ground plane, defined in the camera reference frame, or with respect to a predefined ground plane which defines its own reference frame, such as the map of the station. The latter approach has several advantages, including the possibility of plotting the position of a target on the map of the station. The downsides of this approach are that detailed station maps are required, and no full-automatic calibration is possible. It seemed reasonable that good maps would have been readily available, but it became apparent that that was not always the case. Eventually, map acquisition, manual calibration of each camera, and redaction of the camera layout description file proved to be much more time-consuming than initially expected. At the end of the project, it became clear that, in a production system, manual camera calibration would not be a realistic option.

Tracking across cameras is carried out by combining heterogeneous information from a number of sources (single-camera trackers, appearance descriptors, and prior knowledge on the camera network topology). A probabilistic approach is used to fuse these information sources using the model exposed in Chapter 3, which is the main contribution of this thesis. One disadvantage of this probabilistic approach, however, is that not all information sources can directly output probability values: most appearance descriptors, for example, output a distance measure in a descriptor-specific space. In order to convert this distance into a probability value, annotated training data is required. Collecting and annotating training data is a time consuming process, whose cost adds to the tracker setup cost. On the other hand, other information sources (such as the Kalman filter modelling a target's dynamic state) directly provide a probability value without need for training. It would be desirable if all sources had this property.

Evaluation of the proposed algorithms is less exhaustive than we had originally envisioned, as it is often the case for surveillance applications. Longer video data for testing is desirable, but it comes with the costs of ground truthing and setup (camera calibration, layout generation, etc...). Because of these costs, eventually only two data sets were used for evaluation: the standard i-LIDS data set (used for the NIST multi-camera tracking challenge), and the GTT data set that was collected as part of the CARETAKER project. The i-LIDS data set proved extremely difficult, so much so that only two universities

participated the challenge, both with disappointing results. The GTT data set was collected in order to provide a simpler, but still realistic, tracking scenario for evaluation. Overall, a good attempt has been made to evaluate the tracker on realistic, comprehensive data, and the relatively small number of ground-truthed individuals (only 11) is compensated by the length of their tracks, that include a complete trip in the underground network, spanning two stations and over a dozen cameras each.

An interesting consideration is the comparison of performance between the tracker and a human operator. It is clear that an adequately trained operator, familiar with the surveillance area, can outperform the most advanced computer vision algorithms. Nevertheless, there is space for automatic tracking as a support role in addition to human operators. As explained in the introduction, a multi-camera tracker can improve safety and security by allowing operators to follow potentially dangerous but low-priority targets, which are currently ignored, without requiring additional human resources.

6.3 Future Work

Thanks to the modular nature of the tracker presented in this thesis, there is significant space for expansion and improvement by implementing new modules to plug into the existing framework. The single-camera tracker and the appearance descriptors use the output of the segmentation module to track and describe targets respectively. Because segmentation is based on background modelling, there is an implicit assumption that large foreground blobs correspond to images of people. This assumption, unfortunately, is not valid in many circumstances, for example when a target remains static for a long time or when there is a crowd. In the former case, the target will merge with the background, and will not be detected; in the latter case, a single foreground blob will correspond to several targets, that cannot be separated from each other. This problem could be solved by implementing the segmentation module with a person detection algorithm, whose output is the position and size of each person appearing in the frame, even if they are static or partially overlapped. At the time this research project was begun, however, no person detection algorithm was found that could work on CCTV footage in real-time with a sufficiently high accuracy. The Histogram of Oriented Gradients (HOG) method, however, described in Section 2.4.2, looks promising both in terms of detection accuracy and computational cost; the author of this thesis expects a real-time implementation of this algorithm to be available within a few years.

The modular nature of the framework also means that it is highly suitable for parallelisation, both on a single system (in order to exploit multi-core CPUs) and as a distributed system. The former case has already been partially implemented in the prototype used for the experiments in Chapter 5. However, parallelisation is limited to motion detection, and the rest of the tracking process happens sequentially. With more development time, it would be possible to separate *each* module of the tag-and-track framework (motion detection, colour correction, single-camera tracking, appearance description generation and comparison, etc...) into a separate thread, and exploit parallelism to improve performance on a multi-core or multi-CPU system. Given even more development time, the modules could be separated and deployed on different systems that communicate over a network. Even though module communication over a network is much slower than within one system, a distributed tracker could optimise module placement by exploiting locality, for example by having motion detection modules directly connected to video sources. Such an arrangement could optimise network load and improve overall performances, especially if more than one tracker is to be used simultaneously.

The multi-camera tracker requires much information about the network of cameras, namely geometric calibration of each camera and topological connections between cameras. This information is either provided manually, or learned from the scene. The choice for this project was to input the information manually, in order to focus on the specific problem of tracking; however, providing the complete topology and calibration of even a subset of the CARETAKER test sites proved to be extremely time-consuming. In the future, consideration should be given to automating this process, so that more or larger test sites can be added to the data set.

In this thesis, only visual sensors were considered as data sources. However, there are many more technologies that could provide additional information to help discriminate between targets. The most obvious, for example, would be radio-frequency identification (RFID) tags. RFID readers could be installed on gates at every access point in a station, and they could provide highly reliable identity information of a target, albeit only at a limited set of locations.

In the near future, as the cost of high definition (HD) cameras and 3D sensors diminishes, it seems likely that these technologies will be used for surveillance. When this happens, it will be possible to integrate them in the existing framework as additional sources of information. With HD cameras, for example, it should be possible to extract reliable appearance descriptors from targets' faces, effectively enabling face recognition algorithms to be used for tracking.

3D sensors, such as time-of-flight (TOF) cameras, can be installed next to a

standard camera to provide depth information for each pixel. There are several ways in which this can improve tracking, for example by helping discriminate between partially overlapped targets that are far away along the camera axis, and by incorporating pixels depth into the background model for motion detection [42].

In addition to algorithmic and technological improvements, further work should be done on human-machine interaction (i.e. user interface). For this project, only a rudimentary graphical user interface (GUI) was developed, using OpenCV primitives and keyboard. This minimalist GUI allows the user to select the active camera, either by directly entering its ID, or by navigating the network layout. When a target is visible on the active camera, the user can "tag" it with a single mouse click (camera calibration is used to convert the click point into a bounding box of the appropriate size). If the data source supports it, video playback can be paused, resumed, and advanced frame by frame while tracking; additionally, the video can be rewound and fast-forwarded (but not during tracking). Deciding on a GUI design that is suitable for deployment in control rooms, however, will require close collaboration with security operators, in order to understand their specific needs.

6.4 Publications

V Leung, A. Colombo, J. Orwell and S.A. Velastin - **Modelling periodic scene elements for visual surveillance** in IET Computer Vision, IET Proceedings 2(2) IET, June, pp. 88-98. DOI http://dx.doi.org/10.1049/iet-cvi:20070070 (2008).

A. Colombo, V Leung, J. Orwell and S.A. Velastin - Chapter **Consistent detection and identification of individuals in a large camera network** in Unmanned/Unattended Sensors and Sensor Networks IV, Proceedings of the SPIE Edited by Carapezza, E. M., SPIE, September, Florence, Italy, (2007).

A. Colombo, J. Orwell and S.A. Velastin - **Colour constancy techniques for re-recognition of pedestrians from multiple surveillance cameras** in Workshop on Multi-camera and Multi-modal Sensor Fusion, 18 Oct 2008, Marseille, France, DOI http://www.elec.qmul.ac.uk/staffinfo/andrea/M2SFA2_program.html (2008).

J. Annesley, A. Colombo, J. Orwell and S.A. Velastin - **A profile of MPEG-7 for visual surveillance** in IEEE Conference on Advanced Video and Signal Based Surveillance, Institute of Electrical and Electronics Engineers, 05 - 07 Sep

2007, London, U.K., pp. 482-487. DOI http://dx.doi.org/10.1109/AVSS.2007.4425358 (2007).

A. Colombo, V Leung, J. Orwell and S.A. Velastin - **Markov Models of Periodically Varying Backgrounds for Change Detection** in Visual Information Engineering 2007, IET, July, London, (2007).

J. Annesley, V Leung, A. Colombo, J. Orwell and S.A. Velastin - **Fusion of multiple features for identity estimation** in The Institution of Engineering and Technology Conference on Crime and Security, Visual Information Engineering IET, 13 June - 14 June 2006, London, U.K., pp. 534-539. DOI http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4123728 (2006).

Bibliography

[1] 2009 AVSS Multiple Camera Person Tracking (MCPT) Evaluation Plan. Technical report, National Institute for Standards and Technologies (NIST), 17 June 2009.

[2] José Miguel Aguilera, Horst Wildernauer, Martin Kampel, Mark Borg, David Thirde, and James Ferryman. Evaluation of Motion Segmentation Quality for Aircraft Activity Surveillance. *2nd Joint IEEE International Workshop on Visual Surveillance - Performance Evaluation of Tracking and Surveillance (PETS)*, pages 293–300, October 2005.

[3] James Annesley, Alberto Colombo, James Orwell, and Sergio A. Velastin. A profile of MPEG-7 for visual surveillance. *Advanced Video and Signal Based Surveillance (AVSS2007)*, September 2007.

[4] Sophia Antipolis. Intelligent environments for problem solving by autonomous systems. Activity Report, 2007.

[5] Luis Antón-canalís, Elena Sánchez-nielsen, and Mario Hernández-tejera. Swarmtrack: a particle swarm approach to visual tracking. *International Conference on Computer Vision Theory and Applications (VISAPP)*.

[6] M. Sanjeev Arulampalam, Simon Maskell, Neil A. Gordon, and Timothy G. Clapp. A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[7] Slawomir Bak, Etienne Corvee, François Bremond, and Monique Thonnat. Person Re-identification Using Haar-based and DCD-based Signature. In *2nd Workshop on Activity Monitoring by Multi-Camera Surveillance Systems, AMMCSS 2010, in conjunction with 7th IEEE International Con-*

ference on Advanced Video and Signal-Based Surveillance, AVSS - 2010, Boston, United States, August 2010.

[8] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and data association.* Mathematics in science and engineering. Academic Press, 1988.

[9] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS Publishing, 1995.

[10] Faisal Bashir and Fatih M. Porikli. Performance evaluation of object detection and tracking systems. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, June 2006.

[11] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[12] Jerome Berclaz, Engin Turetken, Francois Fleuret, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[13] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, page 10, 2008.

[14] James Black and Tim Ellis. Multi Camera Image Tracking. *Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, December 2001.

[15] James Black, Tim Ellis, and Paul Rosin. Multi View Image Surveillance and Tracking. *Proceedings of the Workshop on Motion and Video Computnig*, December 2002.

[16] James Black, Dimitrios Makris, and Tim Ellis. Validation of blind region learning and tracking. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 0:9–16, 2005.

[17] Samuel S. Blackman. *Multiple-target tracking with radar applications.* The Artech House radar library. Artech House, 1986.

[18] Ronald N. Bracewell. *The Fourier transform and its applications.* electrical and electronic engineering series. McGraw-Hill, New York, 2nd edition, 1978.

[19] Gershon Buchsbaum. A Spatial Processor Model for Object Colour Perception. *Journal of the Franklin Institute*, 310:1–26, 1980.

[20] Cyril Carincotte, Xavier Desurmont, and Arnaud Bastide. Adaptive metadata management system for distributed video content analysis. *Advanced Concepts for Intelligent Vision Systems (ACIVS)*, October 2007.

[21] CAVIAR. CAVIAR Test Case Scenarios. Web site, July 2003. http://homepages.inf.ed.ac.uk/rbf/CAVIAR.

[22] Jau-Ling Chen and Pei-Yin Chin. An efficient gray search algorithm for the estimation of motion vectors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(2):242–248, May 2001.

[23] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 17:790–799, 1995.

[24] CLEAR. Evaluation Protocol. (on web), 2007.

[25] Miguel Coimbra, Michael Davies, and Sergio A. Velastin. Pedestrian detection using MPEG-2 motion vectors. In *4th European Workshop Image Analysis Multimedia Interactive Services*, London, UK, April 2003.

[26] Alberto Colombo, Valerie Leung, James Orwell, and Sergio A. Velastin. Markov Models of Periodically Varying Backgrounds for Change Detection. *IET International Conference on Visual Information Engineering (VIE)*, September 2007.

[27] Alberto Colombo, James Orwell, and Sergio A. Velastin. Colour Constancy Techniques for Re-Recognition of Pedestrians from Multiple Surveillance Cameras. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2)*, October 2008.

[28] Dorin Comaniciu, Venkataraman Ramesh, and Peter Meer. Kernel-Based Object Tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 25, 2003.

[29] Donna Cox, Volodymyr Kindratenko, and David Pointer. IntelliBadge: Towards Providing Location-Aware Value-Added Services at Academic Conferences. *UbiComp 2003: Ubiquitous Computing*, pages 264–280, 2003.

[30] Antonio Criminisi, Ian Reid, and Andrew Zisserman. A Plane Measuring Device. *Proceedings of the British Machine Vision Conference*, September 1997.

[31] Nello Cristianini. *An Introduction to Support Vector Machines.* Cambridge University Press, Cambridge, 2000.

[32] Rita Cucchiara, Costantino Grana, Giovanni Neri, Massimo Piccardi, and Andrea Prati. The Sakbot System for Moving Object Detection and Tracking. *Video Based Surveillance Systems - Computer Vision and Distributed Processing*, pages 145–157, 2001.

[33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[34] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1994.

[35] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[36] Brian V. Funt. Color constancy in digital imagery. In *International conference on image processing*, volume 3, pages 55–59, October 1999.

[37] Andrew Gilbert and Richard Bowden. *Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity*, pages 125–136. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006.

[38] Rafael C. Gonzales. *Digital Image Processing.* Addison Wesley, 3rd edition, 28 April 1994.

[39] Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 262–275, Berlin, Heidelberg, 2008. Springer-Verlag.

[40] Niels Haering, Péter L. Venetianer, and Alan Lipton. The evolution of video surveillance: an overview. *Machine Vision and Applications*, 19(5-6):279–290, 19 June 2008.

[41] Omar Hamdoun, Fabien Moutarde, Bogdan Stanciulescu, and Bruno Steux. Person re-identification in multi-camera system by signature based on interest point descriptors collected on short video sequences. In *2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-08)*, pages –, Stanford, Palo Alto, États-Unis, 2008.

[42] Dan Witzner Hansen, Mads Syska Hansen, Martin Kirschmeyer, Rasmus Larsen, Davide Silvestre, and Davide Silvestre. Cluster tracking with time-of-flight cameras. *Computer Vision and Pattern Recognition Workshop*, 0:1–6, 2008.

[43] J. Hans Van Hateren and Arjen Van Der Schaaf. Temporal properties of natural scenes. In *In Proceedings of the IS&T/SPIE Conference on Electronic Imaging: Science & Technology*, volume 2657, pages 139–143, 1996.

[44] Home Office Scientific Development Branch. Evaluating 'intelligent' CCTV - iLIDS: imagery library for intelligent detection systems. http://scienceandresearch.homeoffice.gov.uk/hosdb/news-events/270405, 2005.

[45] Thanarat Horprasert, David Harwood, and Larry S. Davis. A Statistical Approach for Real Time Robust Background Subtraction and Shadow Detection. In *IEEE International Conference on Computer Vision 99 - FRAME RATE Workshop*, 1999.

[46] Adrian Ilie and Greg Welch. Ensuring color consistency across multiple cameras. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1268–1275, Washington, DC, USA, 2005. IEEE Computer Society.

[47] Intel Corporation. Open Source Computer Vision Library Reference Manual. http://opencvlibrary.sourceforge.net/, 2001.

[48] ISO/IEC JTC1/SC29/WG11. Introduction to Low-level Visual Description Tools. Retrieved from http://www.chiariglione.org, July 2005.

[49] Saket Jamkar, Swapnil Belhe, Sankalp Dravid, and Mukul Sharad Sutaone. A comparison of block-matching search algorithms in motion estimation. In *Proceedings of the 15th International Conference on Computer communication*, pages 730–739, 2002.

[50] Omar Javed, Zeeshan Rasheed, Khurram Shafique, and Mubarak Shah. Tracking across multiple cameras with disjoint views. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 952, Washington, DC, USA, 2003. IEEE Computer Society.

[51] Seong-Wook Joo and Rama Chellappa. A multiple-hypothesis approach for multiobject visual tracking. *IEEE Transactions on Image Processing*, 16(11):2849–2854, 2007.

[52] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL*, 1997.

[53] Rangachar Kasturi, Dmitry Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John Garofolo, Rachel Bowers, Matthew Boonstra, Valentina Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336, 2009.

[54] Mohamed A. Khamsi and William A. Kirk. *Introduction to Metric Spaces and Fixed Point Theory*. Wiley-Interscience, 1 edition, 3 2001.

[55] Nils Krahnstoever and Paulo R. S. Mendonça. Bayesian autocalibration for surveillance. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1858–1865, Washington, DC, USA, 2005. IEEE Computer Society.

[56] University of Maryland Language and Media Processing Laboratory. ViPER: The Video Performance Evaluation Resource. Available at http://viper-toolkit.sourceforge.net/. Retrieved 2-June-2007.

[57] Andrew Layman. XML Syntax Recommendation for Serializing Graphs of Data. Technical report, World Wide Web Consortium (W3C), 2 December 1998.

[58] Neda Lazarevic-McManus, J.P. Renno, Dimitrios Makris, and Graeme A. Jones. An object-based comparative methodology for motion detection based on the f-measure. *Computer Vision and Image Understanding*, 111(1):74–85, 2008.

[59] Valerie Leung, Alberto Colombo, James Orwell, and Sergio A. Velastin. Modelling Periodic Scene Elements for Visual Surveillance. *IET Computer Vision Journal, Special Issue on Visual Information Engineering*, 2(2):88–98, June 2008.

[60] Liyuan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian. Foreground object detection from videos containing complex background. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10, New York, NY, USA, 2003. ACM.

[61] Rainer Lienhart and Jochen Maydt. *An extended set of Haar-like features for rapid object detection*. volume 1, pages 900–903. Ieee, 2002.

[62] Yang Liu and Kevin M. Passino. Swarm intelligence: Literature overview, 2000.

[63] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[64] David Mackay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, Cambridge, 2004.

[65] Christopher Madden and Massimo Piccardi. A framework for track matching across disjoint cameras using robust shape and appearance features. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 188–193, 2007.

[66] Ron P. S. Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions On Aerospace And Electronic Systems*, 39(4):1152–1178, 2003.

[67] Dimitrios Makris, Tim Ellis, and James Black. Bridging the gaps between Cameras. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2004.

[68] B.S. Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7.* John Wiley & Sons, 2002.

[69] James Manyika and Hugh Durrant-Whyte. *Data Fusion and Sensor Management.* Prentice Hall, 1994.

[70] Dimitri Marinakis and Gregory Dudek. Topology Inference for a Vision-Based Sensor Network. *ICRA*, April 2005.

[71] Nicolas Martel-Besson and André Zaccarin. Learning and Removing Cast Shadows through a Multidistribution Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1133–1146, July 2007.

[72] Anurag Mittal and Larry S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *International Journal of Computer Vision*, pages 189–203, 2002.

[73] Eduardo Monari, Jochen Maerker, and Kristian Kroschel. A robust and efficient approach for human tracking in multi-camera systems. In *Proceedings of the 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, AVSS '09, pages 134–139, Washington, DC, USA, 2009. IEEE Computer Society.

[74] Trevor Montcalm and Bubaker Boufama. Object inter-camera tracking with non-overlapping views: A new dynamic approach. In *Proceedings of the 2010 Canadian Conference on Computer and Robot Vision*, CRV '10, pages 354–361, Washington, DC, USA, 2010. IEEE Computer Society.

[75] Robert Morris. *The Origin of Radar*. Doubleday Anchor, New York, 1962.

[76] Jacinto Nascimento and Jorge S. Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Transactions on Multimedia*, 8(4):761–774, 2006.

[77] Chris J. Needham and Roger D. Boyle. Performance evaluation metrics and statistics for positional tracker evaluation. In *Proceedings of the Computer Vision Systems: Third International Conference (ICVS)*, pages 278–289, 2003.

[78] Eric Nettleton, Hugh F. Durrant-Whyte, and Samer Abdallah. A robust architecture for decentralised data fusion. In *International Conference on Advanced Robotics*, Portugal, July 2003.

[79] Nan Ning and Tele Tan. A Framework for Tracking Moving Target in a Heterogeneous Camera Suite. *International Conference on Control, Automation, Robotics and Vision*, 2006.

[80] Kusha Panta, Ba-Ngu-Vo, and Daniel E. Clark. An efficient track management scheme for the gaussian-mixture probability hypothesis density tracker. In *Intelligent Sensing and Information Processing*, 2006.

[81] Judy K. Partin, Mark L. Stone, John Slater, and James R. Davidson. Infrared tag and track technique. United States Patent 7304300, 2007.

[82] PETS. Performance Evaluation of Tracking and Surveillance. http://www.cvg.rdg.ac.uk/slides/pets.html.

[83] Fatih M. Porikli. Inter-camera color calibration by cross-correlation model function. In *International conference on image processing*, volume 2, pages 133–136, 2003.

[84] J.P. Renno, James Orwell, and Graeme A. Jones. Learning Surveillance Tracking Models for the Self-Calibrated Ground Plane. *Proceedings of the 13h British Machine Vision Conference*, 2002.

[85] Craig W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. In *SIGGRAPH '87*, volume 21, pages 25–34, 1987.

[86] R. Tyrrell Rockafellar. *Variational Analysis*. Springer, Berlin, 1998.

[87] Roman Rosipal and Nicole Kra mer. Overview and recent advances in partial least squares. In *Lecture Notes in Computer Science*, pages 34–51. Springer, 2006.

[88] Yanhua Ruan, Peter K. Willett, and Roy L. Streit. A Comparison of the PMHT and PDAF Tracking Algorithms Based on their Model CRLBs. In *Proceedings of SPIE, the International Society for Optical Engineering*, Orlando, Florida, April 1999.

[89] Riccardo Satta, Giorgio Fumera, Fabio Roli, Marco Cristani, and Vittorio Murino. A multiple component matching framework for person re-identification. *Proceedings of the International Conference on Image Analysis and Processing*, abs/1105.2491, 2011.

[90] Marek Schikora, Wolfgang Koch, and Daniel Cremers. Multi-object tracking via high accuracy optical flowand finite set statistics. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1409–1412, 2011.

[91] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S. Davis. Human detection using partial least squares analysis. *Proceedings of the 12th IEEE International Conference on Computer Vision*, 2009.

[92] Hui-Liang Shen, Hong-Gang Zhang, Si-Jie Shao, and John H. Xin. Chromaticity-based separation of reflection components in a single image. *Pattern Recognition*, 41(8):2461–2469, 2008.

[93] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[94] Lauro Snidaro, Ruixin Niu, Gian Luca Foresti, and Pramod K. Varshney. Quality-based fusion of multiple video sensors for video surveillance. 37(4):1044–1051, August 2007.

[95] Stefano Soatto, Gianfranco Doretto, and Ying Nian Wu. Dynamic textures. In *International Journal of Computer Vision*, pages 439–446, 2001.

[96] Chris Stauffer and W. Eric L. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. *Conference on Computer Vision and Pattern Recognition*, 1998.

[97] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[98] Chris Stauffer and Kinh Tieu. Automated multi-camera planar tracking correspondence modeling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2003.

[99] Gideon P. Stein. Tracking from Multiple View Points: Self-calibration of Space and Time. *DARPA IU Workshop*, pages 1037–1042, 1998.

[100] Rainer Stiefelhagen, Keni Bernardin, Rachel Bowers, R. Travis Rose, Martial Michel, and John Garofolo. The clear 2007 evaluation. pages 3–34, May 2008.

[101] John Taylor. *An Introduction to Error Analysis.* University Science Books, Sausalito, 1997.

[102] The CARETAKER Consortium. Content Analysis and Retrieval Technologies to Apply Knowledge Extraction to massive Recordings. Available at: http://sceptre.kingston.ac.uk/caretaker. Retrieved 15-March-2010.

[103] The Free Online Dictionary. Definition of "track". Available at: http://www.thefreedictionary.com/track. Retrieved 4-April-2009.

[104] David Thirde, Mark Borg, James Ferryman, Joseph Aguilera, Martin Kampbel, and Gustavo Fernandez. Multi-Camera Tracking for Airport Surveillance Applications. In *11th Computer Vision Winter Workshop 2006*, February 2006.

[105] TRECVid. Event Detection Evaluation Plan. http://http://www.nist.gov/speech/tests/trecvid/2008/doc/EventDet08-EvalPlan-v07.htm.

[106] Cong Dung Nghi Truong, Louahdi Khoudour, Catherine Achard, and C. Meurie. Comparaison de séquences d'images pour le suivi d'objets déformables dans des séquences d'images couleur. Application à la surveillance des sites de transport. In *Workshop Surveillance, Sureté et Sécurité des Grands Systèmes (3SGS)*, Troyes, June 2008.

[107] Roger Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[108] Roger Y. Tsai. Metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987.

[109] Sergio A. Velastin, Boghos A. Boghossian, Benny Ping Lai Lo, Jie Sun, and Maria Alicia Vicencio-Silva. PRISMATICA: Toward Ambient Intelligence in Public Transport Environments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, January 2005.

[110] Sergio A. Velastin, Anthony C. Davies, Jia Hong Yin, Maria Alicia Vicencio-Silva, Richard A. Allsop, and A. Penn. Analysis of crowd movements and densities in built-up environments using image processing. In *IEE Colloquium on Image Processing for Transport Applications*, volume 236, pages 8/1–8/6, London, UK, 1993.

[111] Eric A. Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control*, pages 153–158, Lake Louise, Alta., Canada, October 2000. IEEE.

[112] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR 95-041, 2004.

[113] Wikipedia. Xml — wikipedia, the free encyclopedia, 2008. [Online; accessed 3-November-2008].

[114] Wikipedia. Curse of dimensionality — wikipedia, the free encyclopedia, 2009. [Online; accessed 19-October-2009].

[115] Wikipedia. Central limit theorem — wikipedia, the free encyclopedia, 2010. [Online; accessed 9-August-2010].

[116] Wikipedia. Student's t-test — wikipedia, the free encyclopedia, 2010. [Online; accessed 9-August-2010].

[117] Ming Xu and Tim Ellis. Partial Observation vs Blind Tracking through Occlusion. In *Proceedings of the 13th British Machine Vision Conference*, pages 777–786, Cardiff, UK, September 2002.

[118] Fei Yin, Dimitrios Makris, James Orwell, and Sergio A. Velastin. Learning non-coplanar scene models by exploring the height variation of tracked objects. In *Proceedings of the 10th Asian conference on Computer vision - Volume Part III*, ACCV'10, pages 262–275, Berlin, Heidelberg, 2011. Springer-Verlag.

[119] Fei Yin, Dimitrios Makris, and Sergio A. Velastin. Performance evaluation of object tracking algorithms. In *Proceedings of the 10th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2007)*, October 2007.

[120] Zhong Zhang, Andrew Scanlon, Weihong Yin, Li Yu, and Peter Venetianer. Video Surveillance using a Multi-Camera Tracking and Fusion System. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2)*, 18 October 2008.

[121] Jing Zhong and Stan Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust kalman filter, 2003.

PAGE/PAGES
EXCLUDED
UNDER
INSTRUCTION
FROM
UNIVERSITY