Long-term tracking of multiple interacting pedestrians using a single camera

by

Advice Seiphemo Keaikitse

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Applied Mathematics in the Faculty of Science at Stellenbosch University

Department of Mathematical Sciences,
Stellenbosch University,
Private Bag X1, Matieland, 7602, South Africa.

Supervisors:

Dr W. Brink Ms N. Govender

March 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2014 .....

Abstract

Object detection and tracking are important components of many computer vision applications including automated surveillance. Automated surveillance attempts to solve the challenges associated with closed-circuit camera systems. These include monitoring large numbers of cameras and the associated labour costs, and issues related to targeted surveillance. Object detection is an important step of a surveillance system and must overcome challenges such as changes in object appearance and illumination, dynamic background objects like flickering screens, and shadows. Our system uses Gaussian mixture models, which is a background subtraction method, to detect moving objects. Tracking is challenging because measurements from the object detection stage are not labelled and could be from false targets. We use multiple hypothesis tracking to solve this measurement origin problem. Practical long-term tracking of objects should have re-identification capabilities to deal with challenges arising from tracking failure and occlusions. In our system each tracked object is assigned a one-class support vector machine (OCSVM) which learns the appearance of that object. The OCSVM is trained online using HSV colour features. Therefore, objects that were occluded or left the scene can be re-identified and their tracks extended. Standard, publicly available data sets are used for testing. The performance of the system is measured against ground truth using the Jaccard similarity index, the track length and the normalized mean square error. We find that the system performs well.

Uittreksel

Die opsporing en volging van voorwerpe is belangrike komponente van baie rekenaarvisie toepassings, insluitend outomatiese bewaking. Outomatiese bewaking poog om die uitdagings wat verband hou met geslote kring kamera stelsels op te los. Dit sluit in die monitering van groot hoeveelhede kameras en die gepaardgaande arbeidskoste, en kwessies wat verband hou met toegespitse bewaking. Die opsporing van voorwerpe is 'n belangrike stap in 'n bewakingstelsel en moet uitdagings soos veranderinge in die voorwerp se voorkoms en beligting, dinamiese agtergrondvoorwerpe soos flikkerende skerms, en skaduwees oorkom. Ons stelsel maak gebruik van Gaussiese mengselmodelle, wat 'n agtergrond-aftrek metode is, om bewegende voorwerpe op te spoor. Volging is 'n uitdaging, want afmetings van die voorwerp-opsporing stadium is nie gemerk nie en kan afkomstig wees van valse teikens. Ons gebruik verskeie hipotese volging (*multiple hypothesis tracking*) om hierdie meting-oorsprong probleem op te los. Praktiese langtermynvolging van voorwerpe moet heridentifiseringsvermoëns besit, om die uitdagings wat voortspruit uit mislukte volging en okklusies te kan hanteer. In ons stelsel word elke gevolgde voorwerp 'n een-klas ondersteuningsvektormasjien (*one-class support vector machine*, OCSVM) toegeken, wat die voorkoms van daardie voorwerp leer. Die OCSVM word aanlyn afgerig met die gebruik van HSV kleurkenmerke. Daarom kan voorwerpe wat verdwyn later her-identifiseer word en hul spore kan verleng word. Standaard, openbaar-beskikbare datastelle word vir toetse gebruik. Die prestasie van die stelsel word gemeet teen korrekte afvoer, met behulp van die Jaccard ooreenkoms-indeks, die spoorlengte en die genormaliseerde gemiddelde kwadraatfout. Ons vind dat die stelsel goed presteer.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Dr Willie Brink and Ms Natasha Govender for the patient guidance and direction they provided throughout my studies. I could not have hoped for a better introduction to the big scary world of research.

My deepest gratitude is to Dr Daniel Withey for the numerous discussions and thoughtful comments. You were always prepared to read those two or more papers just to provide insightful suggestions to my questions.

I am deeply indebted to Belinda Matebese and Gabriel Magalakwe for, literally, getting me started on this rewarding journey. To indicate what this has meant to me and my family; you are the first people my granny asks about whenever we talk!

Finally, thanks are due to the Mobile Intelligent Autonomous Systems (MIAS) unit within the Council for Scientific and Industrial Research (CSIR) for this opportunity. I could never have accomplished this feat without the support and resource, financial and otherwise, you put at my disposal. I look forward to contributing positively to the organisation.

Dedication

To the One that got away!

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

BS	Background subtraction
DA	Data association
FPR	False positive rate
GMM	Gaussian mixture model
GNN	Global nearest neighbour
HMM	Hidden Markov model
HSV	Hue-saturation-value
i.i.d.	Independent, identically distributed
IPP	Integer programming problem
IPPSolver	Integer programming problem solver
JPTA	Joint probabilistic data association
JSC	Jaccard similarity coefficient
MAP	Maximum *a posteriori*
MDAP	Multi-dimensional assignment problem
MHT	Multiple hypothesis tracking
NMSE	Normalized mean square error
OC-SVM	One class support vector machine
OpenCV	Open source computer vision library
PMHT	Probabilistic multiple hypothesis tracking
RBF	Radial basis function

RGB	Red-green-blue
ROC	Receiver operating characteristic
SCS	Single camera system
SVM	Support vector machine
TPR	True positive rate
TTCT	Time to confirm a track

Variables: Background subtraction

α_t	The learning rate at time t
$\mathbf{X}$	The random process that measures the pixel values
$\mathcal{X}$	The data set used to determine the parameters of the Gaussian mixture model
$\mu_{k,t}$	The mean vector of the k-th component of the Gaussian mixture model at time t (t may be left out to simplify the notation)
$\omega_{k,t}$	The prior of the k-th component of the Gaussian mixture mode at time t (t may be left out to simplify the notation)
$\Sigma_{k,t}$	The covariance matrix of the k-th component of the Gaussian mixture model at time t (t may be left out to simplify the notation)
$\sigma_{k,t}$	The scalar standard deviation of the k-th component of the Gaussian mixture model at time t such that $\Sigma_{k,t} = \sigma_{k,t}^2\mathbf{I}$ (t may be left out to simplify the notation)
Θ_t	The total parameter set of a Gaussian mixture model at time t (t may be left out to simplify the notation). It includes the parameters of each component as well as their priors
$\theta_{k,t}$	The parameter set $\{\mu_{k,t}, \Sigma_{k,t}\}$ of the k-th component of the Gaussian mixture model at time t (t may be left out to simplify the notation)
K	The number of components in a Gaussian mixture model
k	The index of the components of Gaussian mixture model
N	The number of training samples in $\mathcal{X}$

T	The minimum fraction of the training samples that should accounted for the background

Variables: Data association

λ_N	The average number of false measurements per unit time per unit area of the region under surveillance
λ_o	The average number of new objects per unit time per unit area of the region under surveillance
λ_s	The average rate parameter of a Poisson process that models the number of observations on a single object
$\mathbf{z}_{ik}$	The k-th measurement associated with track i
ν_i	The last time a measurement associated with track i was received
A	The cost of initializing a new track
B	The cost of terminating a track
N	The number of consecutive missed detections required to conclude that a tracked object left the scene or was a false measurement
T	The current time
W	The number of frames used to solve the multiple hypothesis tracking problem

Variables: Filtering

$\hat{\mathbf{P}}_{k+1,k}$	The predicted state error covariance matrix at time $k+1$ given information up to time k
$\hat{\mathbf{P}}_{k,k}$	The updated state error covariance matrix at time k
$\hat{\mathbf{x}}_{k+1,k}$	The predicted state estimate given the information up to time k
$\hat{\mathbf{x}}_{k,k}$	The estimate of the state at time k
$\mathbf{F}_k$	The transition matrix at time k
$\mathbf{H}_k$	The measurement matrix at time k
$\mathbf{u}_k$	The input external to the system

$\mathbf{v}_k$	The zero mean white measurement noise with known covariance $\mathbb{E}[\mathbf{w}_k\mathbf{w}_k^T] = \mathbf{Q}_k$
$\mathbf{w}_k$	The zero mean white process noise with known covariance $\mathbb{E}[\mathbf{w}_k\mathbf{w}_k^T] = \mathbf{R}_k$
$\mathbf{X}_k$	The history of the state vector up to time k
$\mathbf{x}_k$	The exact state of the system at time k
$\mathbf{Z}_k$	The set of measurements on the system up to time k
$\mathbf{z}_k$	The measurement on the system at time k

Variables: Integer programming problem

ν_{k_i}	The unique index of the i-th node on track k
τ	A many-to-one mapping from I_n to I_m
I_m	A set of unique measurement indices
I_n	A set of unique node indices
K	The number of tracks in a given multiple hypothesis tracking window
M	The number of measurements from a given frame
M_W	The total number of measurements used in the multiple hypothesis tracking window
n_{k_i}	The log-likelihood ratio of the i-th node on track k
T_p	The number of measurements that have already been processed in a given multiple hypothesis tracking window

Variables: Learning object appearances

α_i	The i-th Lagrangian parameter associated with a constraint that must be satisfied by sample i
η_n	The stochastic gradient descent learning rate at time n
$\mathbf{w}$	The normal of the support vector machine hyperplane
$\mathbf{x}_i$	The input of the i-th labelled sample
μ_i	The i-th Lagrangian parameter associated with the misclassification cost ξ_i

ν	The upper bound on the ratio of outliers, $\nu \in (0, 1)$
$\phi(\cdot)$	Maps the input space into a high-dimensional space within which the data is linearly separable
ξ_i	The slack variable that punishes the misclassification of the i-th training sample
b	The displacement of the support vector machine hyperplane
$k(\mathbf{x}_i, \mathbf{x}_j)$	The kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$
N	The number of labelled training samples
y_i	The label of the i-th training sample

Variables: Experiments

$J(S, G)$	The Jaccard coefficient that measures the similarity between the system output S and ground truth G bounding boxes

Chapter 1

Introduction

Judging by the growing number of closed circuit cameras spread throughout cities and towns in South Africa, it is clear that surveillance is an important issue. This increase in closed circuit camera systems is not only driven by commercial institutions like banks and airports, but also by governments through law enforcement departments. Although the price of purchasing and installation of the hardware decreases daily, the cost of the labour required to monitor these systems is increasing rapidly [19]. Meanwhile, the immense volume of video recordings generated by these systems makes it impossible to monitor every frame. In fact, most of the video recordings are mainly used as forensic evidence, being called upon to verify the facts after an event has occurred [19]. Moreover, there are issues related to targeted monitoring where operators decide to pay close attention to a camera based on the appearances of pedestrians, rather than their behaviours [26], [55].

The monitoring of surveillance systems calls for a scientific solution, which is offered by computer vision in the form of active surveillance. Active surveillance "attempts to detect, recognise and track certain objects from image sequences, and more generally, to understand and describe object behaviour" [38]. Thus, the ultimate goal is to automate the entire surveillance process. This technology has applications in diverse areas including access control; flux statistics and congestion analysis; and anomaly detection and alerting of personnel. These are high level functions which involve the description and understanding of object behaviours. The low level functions are modelling of environments; object detection, classification, recognition and tracking; and retrieval and fusion of data from multiple cameras.

1.1 Background

Collins et al. [19] have implemented one of the most complete automated surveillance systems. It uses multiple, different sensors such as video and

thermal cameras to achieve cooperative tracking. Moreover, their system distinguishes different types of objects like people, groups of people and cars. The system also gives the position of tracked objects in terms of global position system (GPS) coordinates. Another state-of-the-art system is the off-the-shelf Knight system by Shah et al. [76] which can detect, track and categorise objects in the scene covered by multiple cameras. Some of the object categories that it can recognise are people, groups of people, vehicles, animals and bicycles. It also flags abnormal events such as the presence of a person on a track while a train is coming and presents a summary in terms of key frames and a textual description of observed activities. This summary is presented to a human operator for final analysis and decision making.

We now consider systems that use a single camera to track multiple interacting objects. Yang et al. [90] use background subtraction to detect moving objects and the global nearest neighbour algorithm for data association. However, they do not model the motion of the objects. The decision to associate a track and a measurement is based on the Euclidean distance and deciding on the threshold might be arbitrary. Merge and split events are detected and handled explicitly. However, the assumption in the re-identification stage is that the appearance of an object before a merge event is similar to the appearance immediately after the split event. This is practical in the case of short-lived interaction.

Gilbert and Bowden [33] point out that in crowded scenes with overlapping people techniques such as background subtraction cannot be used. As a result the authors train a generic object detector to detect the outline of the head and shoulder. This is due to the assumption that the camera is overlooking the pedestrians and thus the head and shoulders should always be visible. The detected objects are tacked using the mean-shift tracker [20]. The authors note that the mean-shift tracker can fail and result in fragmented tracks. These are joined together using dynamic programming methods.

Benfold and Reid [9] use a generic head detector to detect pedestrians which are then tracked using the Kanade-Lucas-Tomasi (TLK) tracker [54] to track up to four corner features. They note that the TLK tracker is more robust than the mean-shift tracker. They also note that tracking multiple points provides redundancy against tracking failures. The assignment of measurements to tracks is performed using Markov chain Monte Carlo data association.

Our goal is to implement a system that can detect and track multiple interacting pedestrians using a single static camera. The short review of complete systems pointed out challenges that we must solve and points that we must take into account in order to realize our system. Firstly, background subtraction does not work in crowded scenes. Therefore, we either use background

subtraction and consider sparsely crowded scenes, or consider crowded scenes and use generic object detectors. Secondly, tracking algorithms can fail and result in fragmented tracks. Therefore, a method must be devised to connect the fragments into complete tracks. Thirdly, a data association method may be required to assign measurements to tracks. Finally, we should explicitly detect and handle merge and split events.

Our pedestrian detection and tracking system uses a single camera because a multiple camera system can be implemented as a super-system that associates and fuses tracks from multiple single camera systems [45], [7]. In the process, we demonstrate the ability to systematically select components that work together to achieve our goal. We also use machine learning methods to overcome the challenges such as track fragmentation due to occlusions and interacting pedestrians. Track fragmentation occurs when a tracking algorithm fails to track an object and a new track is generated for that object. We use standard computer vision components but their combination into a system is unique.

1.1.1 Object detection

Object detection is an important first stage of a surveillance system because it focuses the attention of subsequent stages such as tracking and classification on dynamic regions of the image and scene. Techniques for object detection may be classified as either background subtraction [18], [31], [61], [79], optical flow [8], [54] or machine learning [24], [34], [56], [86]. Background subtraction and optical flow methods rely on the motion of objects to detect them. The goal of background subtraction is maintenance of an image that is representative of the scene covered by a camera. Optical flow methods, particularly dense flow methods, can be computationally expensive and thus not suitable for real-time systems [19], [93].

Machine learning approaches to object detection learn the generic appearance and shape of objects for them to be detected in images and videos [24], [86]. Most of the methods in this class must be trained off-line using large labelled data sets. They do not adapt to the changes in the appearance of objects as it is not possible to learn all the appearances of all the objects in a class. Moreover, it is especially difficult to make viewpoint- or scale-invariant models. Algorithms have been proposed to learn the appearance of objects online but those rely on robust tracking and/or selective updating of the models [34], [43]. This is a drawback because incorrectly labelled samples can corrupt the learnt model. As a result background subtraction, and in particular Gaussian mixture models (GMM), will be used in this thesis.

1.1.2 Tracking

Tracking is a crucial component for computer vision applications such as automated surveillance and human-computer interaction. It seeks to consistently label objects of interest in every frame of the video sequence. Tracking can be complex due to noise in images, cluttered environments, illumination changes in the scene, object and camera motions, non-rigid and articulated objects, and object occlusions. The requirement for real-time systems is also a challenge because it can disqualify components or combinations thereof that are optimal yet computationally expensive [89]. Tracking may also require the use of multiple cameras either to handle occlusions or to cover large areas. In this case the challenge is reconciling the different identities of an object as seen from the fields of view of different cameras. In our case a single static camera is used.

Approaches to tracking can be divided into two major groups which are filtering and data association, and target representation and localization [20]. Filtering and data association approaches model the dynamics of the object of interest and solve the problem of assigning measurements to tracks. In contrast, target representation and localization approaches model the shape and appearance of objects and thus can cope with changes in the appearances of those objects. These two approaches may be integrated and weighed depending on the tracking problem that is being solved. For example, tracking a face in a crowded environment relies more on target representation and localization. In contrast, aerial surveillance relies more on the dynamics of the target and the camera. Filtering and data association provide a direct answer to the location of the object being tracked. Target representation and localization answer the question of what the object looks like. Only then do they search for that object in the next frame in order to answer the question of where it is. We intend using both approaches in our system.

Target representation refers to the shape and appearance of objects. Models used to represent the shape of objects include points, geometric primitives such as rectangles and ellipses, object contours and silhouettes, and articulated shape and skeletal models [91]. Points are appropriate for tracking objects that occupy small regions in the image. Primitive shapes identify the bounds of the objects and may be used for either rigid or non-rigid objects. The other methods are appropriate for non-rigid objects and imply exact segmentation of objects due to the high level of detail required. We will use rectangles, which are essentially a pair of points on the diagonal, to represent the shape of objects.

As mentioned earlier, target representation also refers to the appearance of objects. Yang et al. [89] identify colour [20], gradient [24] and texture [59], [62] features as appropriate for tracking applications. Colour features are more

sensitive to illumination compared to gradient and texture features [91]. However, colour features can be more discriminative [89] and will be used in this thesis. Tracking algorithms that rely heavily on object representation must search subsequent images for objects that have similar appearances.

The settings used in our experiments contain sparsely crowded environments and there are interactions between pedestrians. Either one of the approaches to tracking may be used but we make a case for using filtering and by extension data association. Yilmaz et al. [91] classify tracking algorithms into point, kernel and silhouette tracking methods. Point tracking methods include the Kalman [87] and particle filters [28]. Our goal is to track multiple interacting objects. Tracking silhouettes is not ideal as they are sensitive to occlusions. Moreover, they provide more detail than is required for our purpose.

Kernel-based methods such as tracking-by-detection [39], [35] and the mean-shift tracker [20] require an external method for initialization. This can be provided by an object detection method such as background subtraction [36], optical flow methods, or generic object detectors [24], [85]. We use background subtraction which can also highlight the regions kernel-based methods may search for matching patterns. The next issue is that of initializing the search. Cominaciu et al. [20] start searching where the pattern was found on the previous time step. However, they suggest incorporating a filtering algorithm to better predict the starting position.

The appearance of objects may change due to changes in illumination and viewpoint, and the non rigidity of objects. Tracking methods, particularly kernel-based tracking methods, must account for these changes. One approach is to adapt the appearance of objects. An example of a highly adaptive tracker is the mean-shift tracker [20] which considers the current appearance of the tracked object as the target appearance. This adapted template could move off the desired object either because of inclusion of background regions in the template or occlusions [10]. Moreover, mean-shift tracking has no memory of any of the past appearance models and may not be able re-identify objects after tracking failure.

Recent approaches use machine learning methods to learn the appearance of objects online [35], [39], [71]. Even in this case, a strategy must be devised to search for regions in the next frame that are confidently explained by the classifiers to find the object of interest in that frame. An alternative approach is to pair online learning methods with particle filter methods to predict prospective object locations [71], [92]. The particle that is best explained by the model can be used as an estimate for the current location of the object. We also note that online learning of object-specific appearances may corrupt the learnt model if incorrectly labelled samples are used.

In conclusion, kernel-based methods require an external method to initialize them. They also must locate the most similar region in the next frame either by searching or by integrating particle methods to predict prospective object locations. Both of these approaches work well when tracking a single object, which is the assumption in most of the methods mentioned above. However, they may be computationally intensive when applied to the tracking of multiple objects. Moreover, they require an external method for initialization. Therefore, we do not use kernel-based tracking.

We have rejected kernel and silhouette tracking and are left with point tracking. We have also rejected all models but points and geometric primitives (rectangles and ellipses) for object shape representation. Note that two points are sufficient to represent either a rectangle or an ellipse. Earlier we noted that filtering methods are examples of point tracking algorithms. As a result we will use rectangles to represent the shape of objects and filtering methods to track those objects.

1.1.3 Tracking multiple targets

Another challenge that we face is tracking of multiple objects. It is particularly challenging because filtering methods assume a one-to-one mapping between the measurements and tracks. This is a data association problem which we will solve using the multiple hypothesis tracker (MHT). MHT provides track initialization, management and termination functions. It uses a number of frames to make track to measurement assignment decisions. We will pose the MHT problem as an integer programming problem and then solve it using a standard solver.

Nevertheless, we note that tracking approaches can fail due to unsuitable models. Most importantly, tracking failure can be due to the assumption that objects of interest are never completely occluded. These methods are likely to fail when objects interact and this issue is not addressed by these methods. Instead, new tracks are initialized after tracking failure or when objects reappear. We are interested in long term tracking of pedestrians which implies consistent labelling of pedestrians whenever they are in the monitored environment. As a result, we will use machine learning algorithms to learn object-specific appearances which are then used to uniquely re-identify objects when they reappear or after tracking failure. To this end, both shape and appearance modelling will be used, respectively, for filtering and learning of pedestrian appearances.

1.1.4 Overview

We have identified object detection, filtering, data association and learning of object appearances as major components of the system. Figure 1.1 shows the high level interactions between these components. The only input to the system is a sequence of frames and the outputs are the frames and a list of tracks that are displayed on the monitor. Of particular interest is the two way flow of data between the components that perform tracking and learning of pedestrians appearances. Tracking results are used to learn the appearance of pedestrians and the learnt appearances are used to pick up tracks during re-identification.

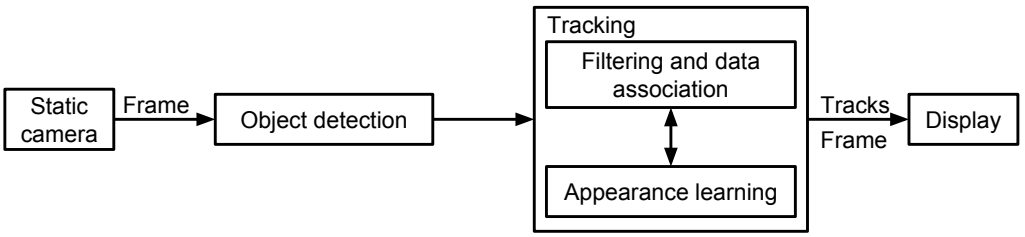


Figure 1.1: High level flow diagram of the system developed.

1.2 Problem statement

The problem we solve is long-term tracking of multiple interacting pedestrians using a single static camera. Such a system must be robust and efficient. Robustness refers to the ability of the system to function under varying conditions. Efficiency refers to the ability of the system to run in real-time. The problem involves both detection and tracking and there are a number of algorithms for each of them, each with its advantages and disadvantages. The problem is to select components that work together to yield a robust and efficient system.

The phrase long-term tracking means that objects are consistently labelled whenever they are in the field of view of the camera. However, standard tracking algorithms can fail due to occlusions and insufficient models. The standard approach is to initialize new tracks whenever tracking failure occurs and this leads to track fragmentation.

1.3 Research objectives

The aim of this thesis is to implement a system that can detect and track multiple interacting pedestrians using a single static camera. Our objectives are

- to systematically select algorithms that implement background subtraction, filtering, data association and online learning,
- and to integrate these algorithms so that they work together to achieve our goal.

Such a system is an important component of an automated surveillance system which aims to understand and describe the behaviour of pedestrians in an environment covered by multiple cameras. Two aspects make our system important to the general automated surveillance system. First, the extension to multiple cameras may be implemented by fusing tracks from multiple single camera systems [45], [7]. Second, the ability to understand and describe the behaviour of pedestrians implies collection of information on those pedestrians. This requires that we know where they are in every frame. In fact, the track itself could help in that process.

1.4 Underlying assumptions

We assume that the system will be used in sparsely crowded environments. This motivated our choice of components which requires that there are periods when tracked objects are not occluded or interacting which will allow the system to build a model of the appearance of the pedestrian which may be used for re-identification. The assumption of sparse crowds also motivated our use of background subtraction.

In addition, it is assumed that all moving objects are pedestrians. Therefore, instead of using a generic pedestrian detector, we define the smallest bounding box that can enclose a pedestrian. Also, this minimises the number of false detections.

We also assume that the camera is static and is placed above the height of pedestrians and a pedestrian in a given frame occupies a small fraction of that frame.

1.5 Thesis outline

This thesis comprises five method chapters and each one addresses the literature specific to it. In Chapter 2 we justify our use of Gaussian mixture models

(GMM), and outline and discuss the GMM equations. In Chapter 3 we introduce filters and motivate our choice of Kalman filters. We then outline the Kalman filter equations and their initialisation. In Chapter 4 we survey the data association literature and conclude that the multiple hypothesis tracker (MHT) is the best for our system. Chapter 5 outlines the steps required to transform the MHT problem into an integer programming problem which may then be solved using off-the-shelf software.

In Chapter 6, the last method chapter, we motivate our use of support vector machines (SVM) to learn the appearance of pedestrians. We then reformulate the SVM optimization problem so that it may be solved using one training sample at a time. This yields an online learning SVM. In Chapter 7 we integrate the components chosen in the above chapters to obtain a complete system. The system is then tested, and the result and discussions are in Chapter 8. We conclude the thesis in Chapter 9.

Chapter 2

Background subtraction

Background subtraction is used in this thesis because it is comparatively computationally efficient. It aims to classify pixels in a video sequence into either foreground (moving objects) or background and relies on the motion of objects to detect them. The idea behind background subtraction is to maintain an image that is representative of the scene monitored by a camera at all times. Although a large number of background subtraction methods exist in the literature, they all follow a simple flow diagram as shown in Figure 2.1. The four major steps in the background subtraction algorithm are pre-processing, background maintenance, foreground detection and post-processing [18]. The pre-processing stage involves simple image processing tasks that transform the raw input video into a format that can be processed by subsequent stages. These may include reducing the frame size and rate, temporal and spatial smoothing and geometric adjustments [18].

The background maintenance stage creates and then maintains a model of the appearance of the background scene as covered by the camera. This stage may be further subdivided into model representation, model initialization and model adaptation. These components expand on the model used to represent the background, how it is initialized and the mechanism used to adapt this model to changes in the background. Various background maintenance methods are available and may be classified as either predictive [83] or non-predictive [79], [88], and recursive [79], [88] or non-recursive [30], [83].

Non-recursive algorithms maintain a buffer of the most recent N frames. The algorithms in this class are highly adaptive as they do not depend on the history

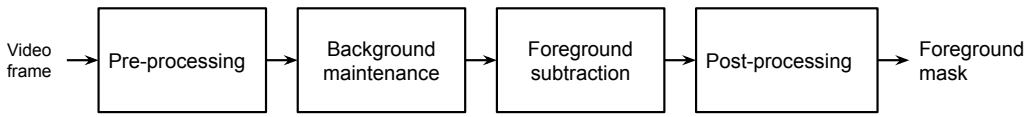


Figure 2.1: Background subtraction flow diagram.

beyond those frames stored in the buffer. However, the storage requirements may be significant [18]. On the contrary, recursive techniques recursively update the background model based on each input frame. Predictive algorithms model the scene as a time series and develop a dynamic model to recover the current input given the past observations. Non-predictive methods neglect the order of the input observations and construct a probability density of the observations at a particular pixel [57].

Foreground detection compares the input video frame with the background model and identifies foreground pixels from the input frame. Absolute differences [83] or statistical [30], [79] techniques may be used to quantify the differences between the input frame and the model. The binary-valued difference map is often obtained by thresholding. Another approach is to use two thresholds with hysteresis [23]. Firstly, pixels with absolute differences that exceed the larger threshold are marked as foreground. Then the foreground region is grown by including neighbouring pixels with absolute differences that exceed the smaller threshold.

The final stage in the pipeline is post-processing. The purpose of this stage is to improve the foreground mask by minimizing the number of false positives and negatives using information external to the background model. Some of the common techniques are median filtering, morphological operations and connected component analysis [18]. When the background model adapts at a slower rate compared to the moving objects, large areas of false objects known as ghosts will appear. These areas can be identified by computing the optical flows at candidate foreground regions because ghosts have no motion [23].

An ideal background subtraction algorithm should adapt to the gradual and sudden changes in illumination, dynamic background objects such as waving trees or escalators, background objects that suddenly start moving and leave holes in the model of the background (ghosts), and background objects that are moved and remain in the foreground forever. It should also handle challenges due to large homogeneously coloured objects where the interior pixels are often undetected, shadows, camouflage and training periods that have foreground objects [83].

2.1 Background maintenance algorithms

The simplest background maintenance model can be obtained in controlled environments, like a movie set, by using a uniformly coloured surface. Even then, the values of a pixel are not fixed in time due to factors such as camera noise and dust particles in the atmosphere. Wren et al. [88] model each pixel with a single Gaussian distribution to allow for these small variations. The

mean and variance are updated using a simple adaptive filter. This approach is similar to the work of Heikkilä and Silvén [37] where the intensity of a pixel is tracked using a Kalman filter. The similarities are due to the Kalman filter assumptions that the dynamic and measurement processes are linear and the noise terms are Gaussian [87]. This type of model cannot handle multi-modal events such as waving trees or flickering monitors which occur in uncontrolled environments [61], [79].

Grimson et al. [79] extend this single Gaussian model by modelling each pixel as a mixture of K Gaussian distributions, where K is fixed and is the same for all pixels. In addition to the mean and variance, each Gaussian distribution is parameterized by a weight that is proportional to its contribution to the mixture. These parameters are adapted using a simple adaptive filter. The algorithm relies on the assumption that the background is visible more frequently than any foreground object and that it has modes with relatively narrow variances [42], [64].

The major drawbacks of this model are the initialization and slow stabilization of the parameters [42], [64]. Moreover, the number of components in a mixture is the same and fixed for all pixels. Zivkovic [94] proposes an improvement to the method that determines at runtime the optimal number of Gaussian distributions required to model the pixel values, in addition to estimating the parameters of each distribution in the mixture. This allows the modes to retain relatively small variances while taking into account that multi-modality is variable both spatially and temporally [15].

In this thesis, moving object detection is performed using the improved mixture of Gaussian distributions algorithm as outlined by KaewTraKulPong and Bowden [42]. Their improvements to the original method by Grimson et al. [79] solve the issues related to the initialization and stabilization of the model parameters. A detailed description of the algorithm is given in Section 2.2. Many other improvements that either attempt to be statistically rigorous or introduce spatial and/or temporal constraints are available, and are surveyed by Bouwmans et al. [15]. Still, all these improvements cannot handle challenges due to substantial illumination changes. The alternative is to detect these changes and then re-initialize the model [64]. For the purpose of completeness we outline other background subtraction methods that aim to solve the challenges associated with mixture models.

One of the challenges to the mixture of Gaussian distributions approach is that the noise in the images is assumed to have a Gaussian distribution [15], [47]. Moreover, the same fixed number of Gaussian distributions in the mixture is used at every pixel in the image. A viable solution to these issues is to model the variations in the intensity of a pixel using adaptive kernel density esti-

mation [30], [57], [82]. Algorithms in this class estimate the density function directly from the data without making any assumptions about the underlying distribution. Elgammal et al. [30], for the purpose of experimentation, assume that the kernel is a Gaussian distribution which results in a generalization of the mixture of Gaussian approach. Note that choosing the Gaussian distribution as a kernel function is different from fitting the distribution to a Gaussian model. Here the Gaussian is only used as a function to weigh the data points [31].

The results of the experiments by Elgammal et al. [30] indicate that the adaptive kernel density estimation approach outperforms the original mixture of Gaussian [79] approach. A comparison with the extension to the mixture of Gaussian distributions by Zivkovic [94] would be interesting because both methods automatically select the number of kernels. Adaptive kernel density estimation methods are computationally and memory intensive [82]. The window size N must also be specified being mindful of the inverse relationship between accuracy and both computation and memory efficiency [82]. The major issue when a finite number of samples is used is the choice of the kernel bandwidth. Too small a bandwidth will result in a ragged density estimate, while too wide a bandwidth will lead to an over-smoothed density estimate [29], [31].

Toyama et al. [83] propose a predictive and non-recursive algorithm that solves most of the background subtraction challenges. It processes frames at a pixel, region and frame level. At a pixel level, each pixel is modelled as a Wiener process using the N most recent pixel intensity values. A pixel that deviates significantly from the predicted intensity value is marked as foreground. This level handles common problems such as gradual illumination changes, dynamic background objects, camouflage and bootstrapping. Region level processing fills in homogeneous regions of foreground by considering inter-pixel relationships. Frame level processing handles global, sudden changes in the frame by switching between alternative models that are kept in memory. The method is computationally expensive because the parameters of the Wiener process for each pixel must be recalculated at every time step. It is also memory intensive because N frames must be kept in memory at all times.

Hidden Markov models (HMMs) may be viewed as a generalization of the mixture of Gaussian distributions if each state of the HMM is modelled using a single Gaussian distribution. The states are hidden (due to unpredictable scene activity) and only indirectly observed through the associated pixel value [64]. This is the approach used by Stenger et al. [80]. More generally, a two state HMM could correspond to an on-off light switch problem where the probability distribution at a state may be a mixture of Gaussians. This would provide statistically rigorous methods to determine when new states may be generated using state splitting. This is in contrast to Toyama et al. [83] where

an arbitrary threshold is used to determine whether to initialize a new model or not. Additionally, HMMs impose a temporal continuity constraint, i.e. if a pixel is part of the foreground it is likely to still be part of the foreground in the next time step [44]. However, real-time computation and topology modification to adapt to dynamic conditions are major limitations of HMMs [80].

2.2 Mixture of Gaussian distributions

The use of a mixture of Gaussian distributions for background subtraction was proposed in Stauffer and Grimson [79] and Grimson et al. [36]. However, the implementation used in this thesis is the improved version of KaewTraKulPong and Bowden [42] that solves the parameter initialization and stabilization problems associated with the original method. This section follows the mathematical derivations of Power and Schoonees [64] and Bilmes [11] while highlighting the assumptions and simplifications that yield the original [79] and improved [42] methods.

2.2.1 Background

Each surface that comes into the view of a given pixel is represented by an element k from the set of states $\{1, 2, \ldots, K\}$ where the number of states K is assumed to be constant. Each state k is associated with an *a priori* probability, $p(k) = \omega_k$, that it will be in the view of the pixel in the next time step, such that $\sum_{k=1}^{K} \omega_k = 1$. The actual state cannot be observed and must be estimated. This is reminiscent of tracking problems where the dynamic and measurement processes are defined. The dynamic model $\mathbf{K}$ generates the state at each time step and the measurement process $\mathbf{X}$ measures the pixel values. The samples of $\mathbf{X}$ may be 1-dimensional (monochrome images) or 3-dimensional (colour).

In this case the pixel value process $\mathbf{X}$ is modelled using a mixture of K Gaussian distribution functions with parameters θ_k for each state k:

$$f_{\mathbf{X}|k}(X|k, \theta_k) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1} (X-\mu_k)}, \tag{2.2.1}$$

where μ_k is the mean vector, Σ_k is the covariance matrix of the kth density and n is the length of the state vector. For computational purposes Stauffer and Grimson [79] assume that the covariance matrix is of the form $\Sigma_k = \sigma_k^2 \mathbf{I}$. This implies that the components have the same statistics [64], and the diagonal matrix implies that the components of $\mathbf{X}$ are independent.

The density parameter set is defined as $\theta_k = \{\mu_k, \sigma_k\}$ for a given state k and the total set of parameters becomes $\Theta = \{\omega_1, \omega_2, \ldots, \omega_K, \theta_1, \theta_2, \ldots, \theta_K\}$. The

state events k are disjoint and thus the distribution of the measurement process $\mathbf{X}$ may be modelled as a mixture of Gaussian distributions:

$$f_{\mathbf{X}}(X|\Theta) = \sum_{k=1}^{K} \omega_k f_{\mathbf{X}|k}(X|k, \theta_k)., \quad \sum_{k=1}^{K} \omega_k = 1. \tag{2.2.2}$$

All the parameters Θ must be estimated from observations of $\mathbf{X}$ in parallel with the estimation of the hidden state.

2.2.2 Estimating the current state

Once the parameters of the Gaussian mixture are known the next step is to estimate which Gaussian distribution gave rise to the current sample X. Given the observation X and the set of parameters Θ, the probability that state k generated this observation, $p(k|X, \Theta)$, may be calculated using Bayes' theorem:

$$p(k|X, \Theta) = \frac{p(k) f_{\mathbf{X}|k}(X|k, \theta_k)}{f_{\mathbf{X}}(X|\Theta)}, \tag{2.2.3}$$

where $p(k|X, \Theta)$ is the posterior probability.

The state k that maximizes the posterior probability, called a match, solves the maximum *a posteriori* (MAP) problem:

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, p(k|X, \Theta) = \underset{k}{\operatorname{argmax}}\, \omega_k f_{\mathbf{X}}(X|k, \theta_k), \tag{2.2.4}$$

where the second equality follows because $f_{\mathbf{X}}(X|\Theta)$ is a normalizing constant in (2.2.3) that is obtained by summing over all values of k, as seen in (2.2.2).

This definition of a match is theoretically correct but does not account for the case where the observation was not generated by any of the components in the mixture. This could be avoided by augmenting the MAP problem with a constraint on the posterior probabilities $p(k|X, \Theta) \geq p_0$, $\forall k$. A better approach is by Stauffer and Grimson [79] and Grimson et al. [36] who consider a component k to have generated the sample X if the distance between this sample and the mean μ_k is less than a constant multiple λ of the standard deviation σ_k. In particular, $\lambda = 2.5$ was used for experimentation. The threshold is a per-pixel-per-distribution threshold which is useful when different regions have different lighting [79]. This is the same matching criterion used by KaewTraKulPong and Bowden [42] and in this thesis.

2.2.3 Estimating the parameters

Given a mixture of Gaussian density functions (2.2.2) governed by a set of parameters Θ, and a data set $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ of size N drawn from

this distribution, the resulting density function for the samples is:

$$p(\mathcal{X}|\Theta) = \prod_{t=1}^{N} f_{\mathbf{X}}(X_t|\Theta) = \prod_{t=1}^{N}\sum_{k=1}^{K} \omega_k f_{\mathbf{X}|k}(X_t|k,\theta_k) \tag{2.2.5}$$

$$= \mathcal{L}(\Theta|\mathcal{X}). \tag{2.2.6}$$

The samples are independent and identically distributed with distribution p. The goal is to find the set of parameters that maximizes the likelihood function $\mathcal{L}(\Theta|\mathcal{X})$.

Analytically, it is easier to estimate the parameters that maximize the log of the likelihood function because it is represented as a sum rather than a product of functions:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \log \mathcal{L}(\Theta|\mathcal{X}) = \underset{\Theta}{\operatorname{argmax}} \sum_{t=1}^{N} \log \left[\sum_{k=1}^{K} \omega_k f_{\mathbf{X}|k}(X_t|k,\theta_k)\right]. \tag{2.2.7}$$

Note that a measurement X_t may be generated by only one Gaussian distribution function $f_{\mathbf{X}|k_t}$, where $k_t \in \{1,2,\ldots,K\}$, at time t. Hence, the sum on which the logarithm operator operates should collapse to a single element and simplify the expression significantly. To this end, an assumption is made that the data set $\mathcal{X}$ is incomplete. In addition, there exists a data set $\mathcal{Y} = \{k_t\}_{t=1}^{N}$ whose values identify the Gaussian pdf that generated X_t at time t.

This leads to the complete-data log-likelihood objective function

$$\log(\mathcal{L}(\Theta|\mathcal{X},\mathcal{Y})) = \sum_{t=1}^{N} \log(\omega_{k_t}\, p_{k_t}(x_t|\theta_{k_t})), \tag{2.2.8}$$

which is optimized iteratively using a set of equations derived from the expectation maximization algorithm. The derivation is found in [11] and the iterative equations are as follows:

$$\hat{\omega}_k = \frac{1}{N}\sum_{t=1}^{N} p(k|X_t,\Theta^g) \tag{2.2.9}$$

$$\hat{\mu}_k = \frac{\sum_{t=1}^{N} X_t\, p(k|X_t,\Theta^g)}{\sum_{t=1}^{N} p(k|X_t,\Theta^g)} \tag{2.2.10}$$

$$\hat{\sigma}_k^2 = \frac{\sum_{t=1}^{N}((X_t-\hat{\mu}_k)\circ(X_t-\hat{\mu}_k))\, p(k|X_t,\Theta^g)}{p(k|X_t,\Theta^g)} \tag{2.2.11}$$

for $k = 1,2,\ldots,K$. Here $\circ$ is the element-wise (Hadamard) multiplication operator and $p(k|X_t,\Theta)$ is given by (2.2.3). Θ^g is the current estimate of the parameters of the mixture of Gaussian distribution functions. The above

equations assume that $\mathbf{K}$ and $\mathbf{X}$ are stationary processes and the number of observations N is fixed. A practical implementation must deal with these assumptions.

An online method must estimate the generating component k and the parameters Θ as new data X_t arrives. It must also adapt to changing scene statistics. Manipulation of (2.2.9) yields the online equivalent:

$$\omega_{k,t} = \frac{1}{t}\, p(k|X_t, \Theta^g) + \left(1 - \frac{1}{t}\right) \omega_{k,t-1} \tag{2.2.12}$$

$$= \alpha_t\, p(k|X_t, \Theta^g) + (1 - \alpha_t)\, \omega_{k,t-1}. \tag{2.2.13}$$

Note the use of t instead of N because the process is online and these two become interchangeable. Also, the time subscript has been added to the parameters.

Substituting $\omega_{k,t}N = \sum_{t=1}^{N} p(k|X_t, \Theta)$ from (2.2.9) into (2.2.10) and (2.2.11) yields, respectively,

$$\mu_{k,t} = (1 - \rho_{k,t})\, \mu_{k,t-1} + \rho_{k,t} X_t \tag{2.2.14}$$

$$\sigma^2_{k,t} = (1 - \rho_{k,t})\, \sigma^2_{k,t-1} + \rho_{k,t}\, ((X_t - \mu_{k,t}) \circ (X_t - \mu_{k,t})) \tag{2.2.15}$$

where

$$\rho_{k,t} = \frac{\alpha_t\, p(k|X_t, \Theta^g)}{\omega_{k,t}}. \tag{2.2.16}$$

The model should be able to adapt to changing illumination by emphasizing more recent samples over older samples [64]. As (2.2.12) stands it integrates all the historical data and becomes more and more insensitive to new data because $\alpha_t = \frac{1}{t} \to 0$ as $t \to \infty$.

Stauffer and Grimson [79] and Grimson et al. [36] fix the value of $\alpha_t = \alpha$ to work around this problem. KaewTraKulPong and Bowden [42] indicate that this leads to poor initialization. The proposed solution is to set a lower bound on the learning rate α_t:

$$\alpha_t = \begin{cases} 1/t, & \text{if } t < L, \\ 1/L, & \text{otherwise,} \end{cases} \tag{2.2.17}$$

where t and L are the number of data points used in estimating the parameters. This definition of α_t highlights the dichotomy that should exist in the behaviour of the iterative equations. If $t < L$ the online implementation of the average should be used. In contrast, if $t \geq L$ the online implementation of the L-window running average should be used. The iterative equations obtained

thus far cater for the first case. The following iterative equations cater for the second case [42]:

$$\omega_{k,t} = \alpha_t\, p(k|X_t, \Theta^g) + (1-\alpha_t)\,\omega_{k,t-1} \tag{2.2.18}$$
$$\mu_{k,t} = (1-\alpha_t)\,\mu_{k,t-1} + \rho_{k,t} X_t \tag{2.2.19}$$
$$\sigma^2_{k,t} = (1-\alpha_t)\,\sigma^2_{k,t-1} + \rho_{k,t}\,((X_t - \mu_{k,t}) \circ (X_t - \mu_{k,t})). \tag{2.2.20}$$

We note that the value of $\rho_{k,t}$ in (2.2.16) differs from the one used by Stauffer and Grimson [79]. Here the posterior probability $p(k|X_t, \Theta^g)$ is used whereas Stauffer and Grimson [79] use the likelihood function $f_{\mathbf{X}|k}(X|k, \theta_k)$. KaewTraKulPong and Bowden [42] point out that the use of the likelihood function means a very small value for $\rho_{k,t}$ which results in the slow adaptation of the parameters. Substituting (2.2.3) into (2.2.16) indicates that Stauffer and Grimson [79] are missing the normalizing constant which could adjust $\rho_{k,t}$ upwards:

$$\rho_{k,t} = \frac{\alpha_t\, f_{\mathbf{X}|k}(X|k, \theta_k)}{f_{\mathbf{X}}(X|\Theta)}. \tag{2.2.21}$$

The posterior $p(k|X_t, \Theta)$ will not be used to find a match. However, it is still used in the iterative equations. Thus, the computational benefit derived from the new definition of a match is lost because the posterior must be calculated [64]. To avoid calculating posterior probabilities Power and Schoonees [64] note that they are either close to 0 or close to 1. Specifically, this probability is close to 1 for one and only one component in the mixture [79], [64]:

$$p(k|X_t, \Theta) \approx M_{k,t} = \begin{cases} 1, & \text{if } k \text{ is a match,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.2.22}$$

In case there is more than one match, the one with the highest supporting data (largest $\omega_{k,t}/\sigma_{k,t}$) is chosen.

The final online equations, with some adjustments, are summarized below for convenience:

$$M_{k,t} = \begin{cases} 1, & \text{if } k \text{ is a match,} \\ 0, & \text{otherwise} \end{cases} \tag{2.2.23}$$
$$\omega_{k,t} = \alpha_t\, M_{k,t} + (1-\alpha_t)\,\omega_{k,t-1} \tag{2.2.24}$$
$$\rho_{k,t} = \frac{\alpha_t}{\omega_{k,t}} M_{k,t}\ , \qquad \eta_{k,t} = \begin{cases} \rho_{k,t}, & \text{if } t < L, \\ \alpha_t\ , & \text{otherwise} \end{cases} \tag{2.2.25}$$
$$\mu_{k,t} = (1-\eta_{k,t})\,\mu_{k,t-1} + \rho_{k,t} X_t \tag{2.2.26}$$
$$\sigma^2_{k,t} = (1-\eta_{k,t})\,\sigma^2_{k,t-1} + \rho_{k,t}\,((X_t - \mu_{k,t}) \circ (X_t - \mu_{k,t})). \tag{2.2.27}$$

2.2.4 Segmenting the foreground

The mixture model does not distinguish between foreground and background surfaces. Once the Gaussian mixture component k that generated the observation X_t has been identified, we need to determine whether it represents the foreground or background surface. Heuristically, background objects will have the most supporting evidence ($\omega_k \to 1$), and the least variance ($\sigma \to 0$) [79]. Consider as an example a static background surface. Each pixel can be modelled with a single Gaussian pdf, $\omega = 1$, and will have a very small variance [88].

The components of the mixture are first sorted using the criterion ω_k/σ_k which integrates the two objectives [79]. If σ_k is n-dimensional the ranking must be done using $\omega_k/\|\sigma_k\|$ or $\omega_k^2/\|\sigma_k\|^2$. Then, the first B components represent the background model, where

$$B = \underset{b}{\operatorname{argmin}} \sum_{k=1}^{b} \omega_k \leq T \tag{2.2.28}$$

and T is the minimum fraction of the data that should accounted for the background. A small value of T forces the model to have a single modality thus only a single surface may represent the background.

Lastly, if a match k is found and $k \leq B$ then the pixel is marked as background, otherwise it is marked as foreground. If a match is not found then the pixel is classified as foreground. In this case the lowest ranked component is replaced with a new Gaussian probability distribution function. The mean of this distribution is the value of the sample X_t, and the variance and weight are set to large and small default values respectively. The parameters may then be updated. Finally the weights are normalized so that they represent probabilities.

2.3 Conclusion

This chapter introduced moving object detection and focused on our choice for segmenting moving objects which is background subtraction. A detailed outline of the use of mixture of Gaussian distributions for background subtraction was given. In particular, the assumptions and simplifications to the theoretic model that are required to implement the version used in the thesis were highlighted.

One advantage of Gaussian mixture models is that the existing background is not destroyed when a new surface becomes part of the background. The existing background remains in the mixture until it becomes the last ranked

surface and is replaced with a new one. Thus, when an object is stationary long enough to become part of the background and then moves, the component that represents the previous background will still be in the mixture and will be recovered quickly [36], [79]. A flow diagram that clarifies the interactions between the different functions is given in Figure 2.2.

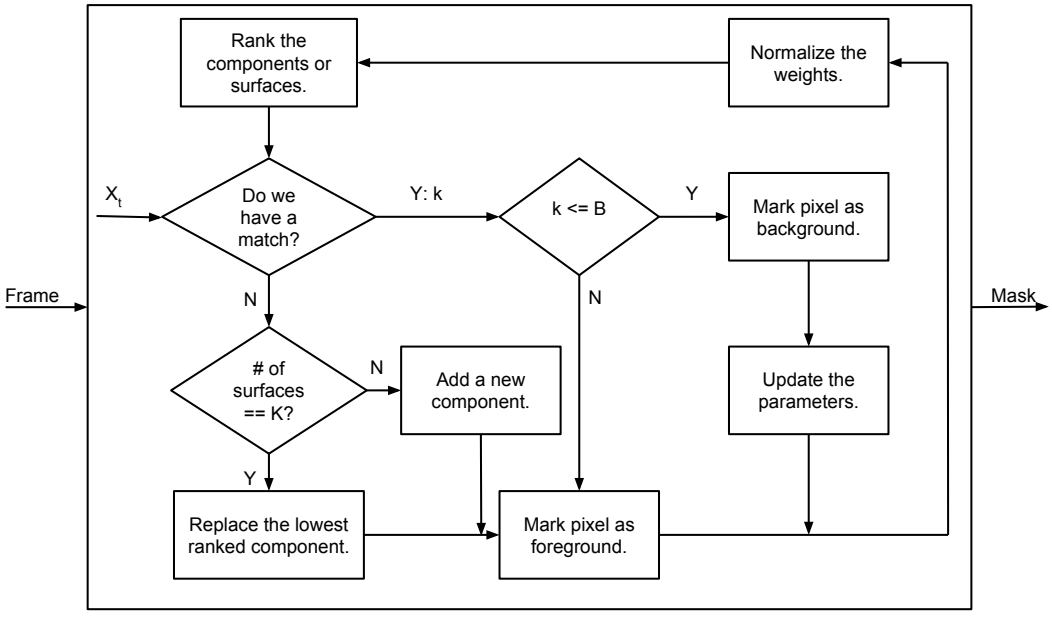


Figure 2.2: Detailed flow diagram of the Gaussian mixture model approach to background subtraction.

Chapter 3

Filtering

Filtering attempts to estimate states of systems. It involves the use of measurements on those systems to obtain better estimates of their states. Filtering is important when the states of systems of interest cannot be measured directly. Moreover, the measurements on the objects are taken at discrete times and may be corrupted with noise.

Formally, the problem that filtering algorithms attempt to solve is to estimate the state $\mathbf{x}_k \in \mathbb{R}^{n_x}$ of the system at all times, k, as the system evolves. The exact state $\mathbf{x}_k$ is not observable and thus at least two models are required to analyze and make inferences about the system. The first is the dynamic process which models the evolution of the state:

$$\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \tag{3.0.1}$$

where $\{\mathbf{w}_k | k = 1, 2, \ldots\}$ is a set of process noise terms that compensates for the inaccuracies in the state vector due to the complexity of the system, insufficient knowledge and unknown environments [73]. The variable $\mathbf{u}_k$ represents the input external to the system. Note that the function can change with time and is possibly nonlinear.

As the system evolves it may be measured at discrete times. This yields the second of the models, referred to as the measurement process, which relates the measurement vector $\mathbf{z}_k \in \mathbb{R}^{n_z}$ to the state vector $\mathbf{x}_k$ at time k:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k), \tag{3.0.2}$$

where $\{\mathbf{v}_k | k = 1, 2, \ldots\}$ is a set of i.i.d. measurement noise terms. This noise takes into account sources of uncertainty such as digitization, backlash and nonlinear response in the sensors. As in (3.0.1), the function may also change with time and is possibly nonlinear.

We note that $\mathbf{X}_k = \{\mathbf{x}_i | i = 1, 2, \ldots, k\}$ is the history of the state vector up to time k. Also, $\mathbf{Z}_k = \{\mathbf{z}_i | i = 1, 2, \ldots, k\}$ is the set of measurements on the system

up to time k. Thus, the models are evaluated in discrete time and formulated within a state-space framework. Moreover, for most problems, an estimate of the state vector $\mathbf{x}_k$ is required every time a new measurement is acquired. This estimation process involves two stages: predicting and updating the state vector. The prediction stage uses (3.0.1) to propagate the state vector forward in time. The updating stage takes into account the measurements as well as their relationship to the state vector as modelled by (3.0.2). This lends itself to the recursive application of the Bayes filter.

3.1 Bayesian framework

The Bayes filter is a theoretic solution to the state estimation problem. Within the Bayesian framework, estimating the state vector $\mathbf{x}_k$ given all sensor measurements $\mathbf{Z}_k$ is equivalent to constructing a posterior probability distribution function $p(\mathbf{x}_k|\mathbf{Z}_k)$. Moreover, Bayes' theorem indicates that

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_{k+1}) = \frac{p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})p(\mathbf{x}_{k+1}|\mathbf{Z}_k)}{p(\mathbf{z}_{k+1}|\mathbf{Z}_k)}, \tag{3.1.1}$$

where the denominator is a normalization constant given by

$$p(\mathbf{z}_{k+1}|\mathbf{Z}_k) = \int p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})p(\mathbf{x}_{k+1}|\mathbf{Z}_k)\, d\mathbf{x}_{k+1}. \tag{3.1.2}$$

In fact, (3.1.1) is the updating equation. It takes into account the likelihood that the hypothetical state $\mathbf{x}_{k+1}$ has a measurement $\mathbf{z}_{k+1}$. The likelihood pdf, $p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})$, is defined by the measurement equation (3.0.2) and the known noise process $\mathbf{w}_{k+1}$.

Equation (3.1.1) also takes into account the prior pdf, $p(\mathbf{x}_{k+1}|\mathbf{Z}_k)$. Supposing that the posterior pdf $p(\mathbf{x}_k|\mathbf{Z}_k)$ at time k is known, the prior is defined as

$$p(\mathbf{x}_{k+1}|\mathbf{Z}_k) = \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Z}_k)\, d\mathbf{x}_k. \tag{3.1.3}$$

The prior takes into account the probabilistic model of the evolution of the system, $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$, which is defined by the dynamic model (3.0.1) as well as the process noise $\mathbf{v}_k$ [2]. This is the prediction equation.

Equations (3.1.1) and (3.1.3) are the theoretic solution of the filtering problem. Analytical solutions of the problem exist for a few cases, in particular when the dynamic and measurement models are linear and their respective noise parameters have Gaussian distributions. These are exactly the assumptions used in deriving the Kalman filter which, in this case, is the optimal filter [70], [87].

If the noise parameters have Gaussian distributions but the models are non-linear, the favoured approaches are the extended and unscented Kalman filters. The extended Kalman filter linearizes the models locally by considering their first order Taylor expansions [87]. In contrast, the unscented Kalman filter approximates the probability distribution of the state using a set of deterministically chosen sample points [40], [41]. The case where at least one of the models is nonlinear and the noise parameters are non-Gaussian is tackled using particle methods, specifically the particle filters.

The choice of which filtering method to use cannot be made in isolation. The data association method which is used to assign measurements to tracks must also be considered. The multiple hypothesis tracker (MHT) is used for data association and the justification is provided in Chapter 4. In short, MHT provides facilities to initialize tracks and quantify their quality. Unfortunately, the computational tractability of the combination of MHT and a particle filter is a major stumbling block.

The linear Kalman filter is used in conjunction with MHT in this thesis. The use of the Kalman filter makes it easier to evaluate the quality of tracks in MHT. One disadvantage of this combination stems from the Kalman filter assumptions that the noise in the measurement and state evolution processes have uni-modal Gaussian distributions. Using MHT may be thought of as introducing multi-modality to these processes. In Morefield [58] and Avitzour [3] the multiple target tracking problem interpretation is such that a measurement is generated by a mixture of density distributions. Also, people tend to walk in straight lines at constant velocities. Thus, the constant velocity Kalman filter model is used in this thesis.

3.2 Linear Kalman filter

3.2.1 Derivation

The Kalman filter may be derived from (3.1.1) and (3.1.3) by assuming that the dynamic and measurement processes are linear and the noise is additive and Gaussian, i.e.

$$\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) = \mathbf{F}_k\mathbf{x}_k + \mathbf{G}_k\mathbf{u}_k + \mathbf{w}_k, \tag{3.2.1}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k, \tag{3.2.2}$$

where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_k)$ is a normal distribution function with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

The system and measurement noise processes $\mathbf{w}_k$ and $\mathbf{v}_k$ are assumed to be uncorrelated, i.e.

$$\mathbb{E}[\mathbf{w}_k\mathbf{w}_l^T] = \begin{cases} \mathbf{Q}_k, & \text{if } k = l, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{3.2.3}$$

$$\mathbb{E}[\mathbf{v}_k\mathbf{v}_l^T] = \begin{cases} \mathbf{R}_k, & \text{if } k = l, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{3.2.4}$$

$$\mathbb{E}[\mathbf{w}_k\mathbf{v}_l^T] = \mathbf{0} \quad \text{for all } k,\, l. \tag{3.2.5}$$

Another assumption is that the initial state of the system $\mathbf{x}_0$ is a random vector that is not correlated to either the system or measurement noise processes. Moreover, the initial state has a known mean $\hat{\mathbf{x}}_{0|0}$ and error covariance matrix $\mathbf{P}_{0,0} = \mathbb{E}[(\hat{\mathbf{x}}_{0,0} - \mathbf{x}_0)(\hat{\mathbf{x}}_{0,0} - \mathbf{x}_0)^T]$.

The linear Kalman filter is an unbiased filter that minimizes the mean square error. Thus, given that the dynamic and measurement processes are linear, and a set of observations $\mathbf{Z}_k = \{\mathbf{z}_i | i = 1, 2, \ldots, k\}$, the Kalman filter yields the optimal estimate of $\mathbf{x}_k$, denoted $\hat{\mathbf{x}}_{k|k}$, that minimizes the expectation of the square-error loss function [69]:

$$\mathbb{E}(\|\mathbf{x}_k - \hat{\mathbf{x}}_{k,k}\|^2) = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_{k,k})^T(\mathbf{x}_k - \hat{\mathbf{x}}_{k,k})] \tag{3.2.6}$$

and is unbiased, i.e. the expected state estimate is the exact expected state:

$$\mathbb{E}(\hat{\mathbf{x}}_{k,k}) = \mathbb{E}(\mathbf{x}_k). \tag{3.2.7}$$

Either of these two approaches yields the Kalman filter which consists of the prediction and update steps.

- The prediction step, which is also known as the time update step, predicts the state and state error covariance matrix at time $k+1$ given the information at time k:

$$\hat{\mathbf{x}}_{k+1,k} = \mathbf{F}_k\hat{\mathbf{x}}_{k,k} + \mathbf{G}_k\mathbf{u}_k, \tag{3.2.8}$$

$$\hat{\mathbf{P}}_{k+1,k} = \mathbf{F}_k\hat{\mathbf{P}}_{k,k}\mathbf{F}_k^T + \mathbf{Q}_k. \tag{3.2.9}$$

- The update step, which is also referred to as the measurement update step, uses the measurement $\mathbf{z}_{k+1}$ and the predicted state to update the state and error covariance matrix:

$$\hat{\mathbf{x}}_{k+1,k+1} = \hat{\mathbf{x}}_{k+1,k} + \mathbf{K}_{k+1}[\mathbf{z}_{k+1} - \mathbf{H}_{k+1}\hat{\mathbf{x}}_{k+1,k}], \tag{3.2.10}$$

$$\mathbf{D} = \mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}, \tag{3.2.11}$$

$$\hat{\mathbf{P}}_{k+1,k+1} = \mathbf{D}\hat{\mathbf{P}}_{k+1|k}\mathbf{D}^T + \mathbf{K}_{k+1}\mathbf{R}_{k+1}\mathbf{K}_{k+1}^T, \tag{3.2.12}$$

where $\mathbf{K}_{k+1}$ is the Kalman gain given by

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1,k}\mathbf{H}_{k+1}^T[\mathbf{H}_{k+1}\mathbf{P}_{k+1,k}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1}]^{-1}. \tag{3.2.13}$$

This, together with the initial conditions on the state and error covariance matrix, yields the Kalman filter which is an iterative algorithm.

3.2.2 Initializing the Kalman filter

One of the Kalman filter assumptions is that the initial state and the error covariance matrix are known. The state of the object at time k is represented as a column vector $\mathbf{x}_k = [x_k, y_k, v_{x,k}, v_{y,k}, w_k, h_k]^T$. The centre of the bounding box is (x_k, y_k) and w_k and h_k are the width and height, respectively. The components of the velocity vector are $(v_{x,k}, v_{y,k})$. We assume that the dimensions of the bounding box remain constant.

Methods for initializing the state and, in particular, the error covariance matrix are available in the literature and include using either one or two measurements. When using a single measurement, the centre and dimensions of the bounding box are known, therefore their variances are set to small values. In contrast, the velocity vector is not known and the variances of its components are set to large values to ensure that their estimates converge quickly and the influence of the initial guess soon is negligible [53]. In the case where two measurements are used for initialization, the object position and velocity vectors can be determined and thus the entries of the posterior state error covariance matrix may be set to small values. In this thesis a single measurement is used to initialize the Kalman filter.

3.3 Conclusion

Filtering methods attempt to estimate the states of objects of interest. This requires modelling of the object dynamic and measurement processes. We outlined the Bayes filter which is the theoretic iterative solution to the filtering problem. Different assumptions on the process and noise terms yield different practical filters. We opted for Kalman filters which can be derived by assuming that the processes are linear and the noise terms are Gaussian and additive. We outlined the Kalman filter equations and the two methods that can be used to initialize filtering. We noted that our interest is the tracking of multiple pedestrians and yet are using a filtering method that assumes a one-to-one relationship between tracks and measurements. This raises data association issues which are addressed in Chapter 4. Strictly speaking, the choice of data association method also informed the choice of filtering method.

Chapter 4

Data association

Filtering involves the use of measurements to obtain better estimates of the state of an object. The measurements are not labelled and could be from valid objects, false alarms or clutter. In the case of multiple target tracking, some of the measurements could be from new targets because the number of targets in the scene is unknown. Moreover, some of the tracked objects may not be detected in a frame because they either exited the surveillance area or were occluded [22]. The goal of data association is to solve this measurement origin problem. Data association enables the use of standard filtering algorithms such as the Kalman filter [87] which assume a one-to-one mapping between tracks and measurements.

Data association is a way to model the interactions between objects particularly when the dynamics of the individual objects are assumed to be independent. Classical approaches to data association may be categorized as either sequential [49] or deferred logic [68], [81] approaches. Sequential logic methods make data association decisions as measurements are received. Deferred logic methods delay making these decisions until more information is available. These methods are preferred because once the data association decisions have been made they may not be changed. Incorrect data association decisions may result in lost or fragmented tracks.

Deferred logic methods may be further classified as enumerative and non-enumerative [65]. Enumerative methods entail the explicit consideration of hypothesis and include the multiple hypothesis tracker [68]. Non-enumerative methods include the probabilistic multiple hypothesis tracker [3], [81] and representing the multiple target tracking problem as a multi-dimensional assignment problem (MDAP) [58], [63]. Note that probabilistic multiple hypothesis tracking (PMHT) and the MDAP are alternative formulations of the multiple target tracking problem that is posed as an incomplete statistical data problem. The state of the tracks represent the known data and the measurement-track associations represent the unknown data. In the case of PMHT, the

measurements may then be assumed to have been generated by a mixture of densities. The MDAP is derived by noting that only one of the components of the mixture may generate a measurement. The probabilistic MHT is derived by converting the problem into a complete data problem. However, it is not clear if they have track initialization capabilities.

Particle filter approaches to multiple target tracking integrate the interaction model with the dynamic and measurement processes [50]. One approach is to use the joint state vector but this results in what is referred to as the curse of dimensionality. That is, the size of the state vector increases exponentially with the increase in the number of tracked objects. An alternative is to use a particle filter to track each object when the objects are far apart. If objects get too close to each other an alternative method is then used to model the interactions. Markov random fields are the preferred method for this purpose [46], [50]. In this thesis each object is assigned a Kalman filter and the interactions are modelled using a classical data association method.

4.1 Data association methods

The simplest data association algorithm is the nearest neighbour tracker which updates a track with the measurement that is closest to the predicted state of the track [49]. Here, the distance is defined as

$$D(\mathbf{z}_k, \hat{\mathbf{x}}_{k,k}) = (\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k,k})^T\mathbf{S}_k^{-1}(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_(k,k)), \tag{4.1.1}$$

where $\mathbf{S}_k = \mathbf{H}_k^T\hat{\mathbf{P}}_k\mathbf{H}_k + \mathbf{R}_k$. $\mathbf{S}_k$ is the covariance of the innovation corresponding to the measurement $\mathbf{z}_k$. All other variables are as defined for the Kalman filter in Section 3.2. Note that distance as defined above is a χ^2 random variable. Thus, given the dimension of the measurement vector (degree of freedom) and the desired probability that a measurement was generated by the target, a threshold λ on the distance may be read off the χ^2 table. The validation region of the state $\mathbf{x}_k$ can then be defined as

$$\mathcal{D}(\mathbf{x}_k, \lambda) = \{\mathbf{z}_k \,|\, D(\mathbf{z}_k, \hat{\mathbf{x}}_k) \leq \lambda^2\}. \tag{4.1.2}$$

The nearest neighbour tracker may result in one track stealing the measurement of another track especially then the targets are close together. This can result in the other track being terminated. An improvement to this method is the global nearest neighbour (GNN) method which minimizes the sum of the distances between predicted states of the tracks and measurements [49]. This can be posed as a 2-dimensional assignment problem and computationally efficient methods for solving such problems are available [14]. GNN works well

when there is no clutter and track contention. However, it cannot handle the appearance and disappearance of objects. T
The joint probabilistic data association (JPDA) filter is more robust to clutter and track contention [7]. It is the extension of the probabilistic data association filter to multiple target tracking. The JPDA filter assumption is that the target may not have generated the closest measurement. That is, the true measurement may be further away. To account for this, each measurement is assigned a weight that is proportional to the probability that it was generated by the track with which it is associated [52]. Hence, the average of the measurements is used to update the state of the target. One drawback of JPDA is that the tracks of closely spaced targets will tend to come together because the same subset of measurements is used to update both targets [12]. Also, the number of tracked objects must be known and fixed. Schulz et al. [75] introduce the sample-based version of JPDA and define a distribution over the number of tracked objects which allows a varying and unknown number of targets to be tracked.

JPDA introduces the concept of the probability of track-measurement association to multiple target tacking. This concept is crucial to multiple hypothesis tracking (MHT) which is a deferred logic method [1], [68]. MHT is An exhaustive method for enumerating all possible track to measurement associations. Ultimately, an optimal set of disjoint tracks, referred to as a hypothesis, must be retained. This process involves the evaluation of probabilities of sequences of measurements having originated from various targets [6]. Two approaches to MHT are the hypothesis-oriented MHT [68] and the track-oriented MHT [6]. The original hypothesis-oriented MHT yields joint probabilities of measurement to track association hypotheses. The probabilities of individual tracks may then be obtained by marginalization. It is a top-down approach and the reverse of track-oriented MHT.

In this thesis multiple hypothesis tracking is used for data association. It implicitly provides facilities for track initialization, continuation and termination [22]. Quantitative meanings are given to track management concepts such as tentative and confirmed tracking [77]. It also explicitly models both spurious measurements and constraints on measurements. The MHT approach is memory and computation intensive but techniques such as gating and track clustering are available to improve the situation. This is explained in more detail in Section 4.4.

4.2 Multiple hypothesis tracking

We consider the assignment problem in Figure 4.1 from Reid [68] to highlight the difference between hypothesis-oriented MHT and track-oriented MHT. The

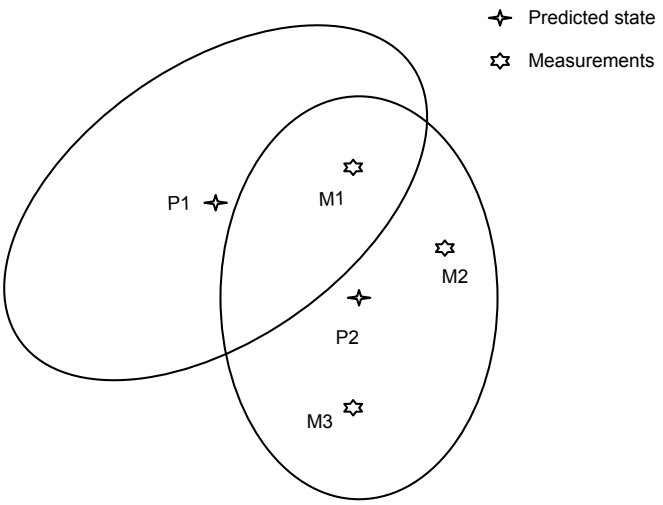


Figure 4.1: Configuration of targets and measurements in example cluster.

current hypothesis contains two tracks with indices 1 and 2. In the case of hypothesis-oriented MHT the new set of hypotheses can be represented in tree form as demonstrated in Figure 4.2. Each path from the root of the tree to one of the leaves corresponds to a hypothesis. In constructing these, it is assumed that a measurement may be associated with a track, a false alarm or the start of a new track. A measurement is a false alarm if it is not associated with at least one track in a hypothesis and this is indicated with a zero. The index of a new track is one more than the number of existing tracks. A track that is associated with a measurement retains its index. Moreover, each track may be associated with at most one measurement.

The path $[0, 0, 0]$ in the tree is the hypothesis obtained by assuming that all measurements are false alarms. This implies that the currently tracked objects were not detected in the current frame. The states of the objects are even more uncertain and this could be reflected by increasing their state error covariance matrices. The path $[1, 0, 0]$ is obtained by updating track 1 with the first measurement. Track 2 was not detected and the other measurements are considered to be false alarms.

An example of an incompatible hypothesis is $[1, 2, 1]$ marked with a star in the tree. This means that track 1 was associated with the first and the third measurements at the same time. Such hypotheses are discarded. The remaining hypotheses are constructed in a similar fashion. This process yields a set of hypotheses and it is repeated for each of these when a new set of measurements is received. The data association problem is solved after a given number of frames has been processed. Cox et al. [22] implement the solution to hypothesis-oriented MHT using an algorithm due to Murty [60] which re-

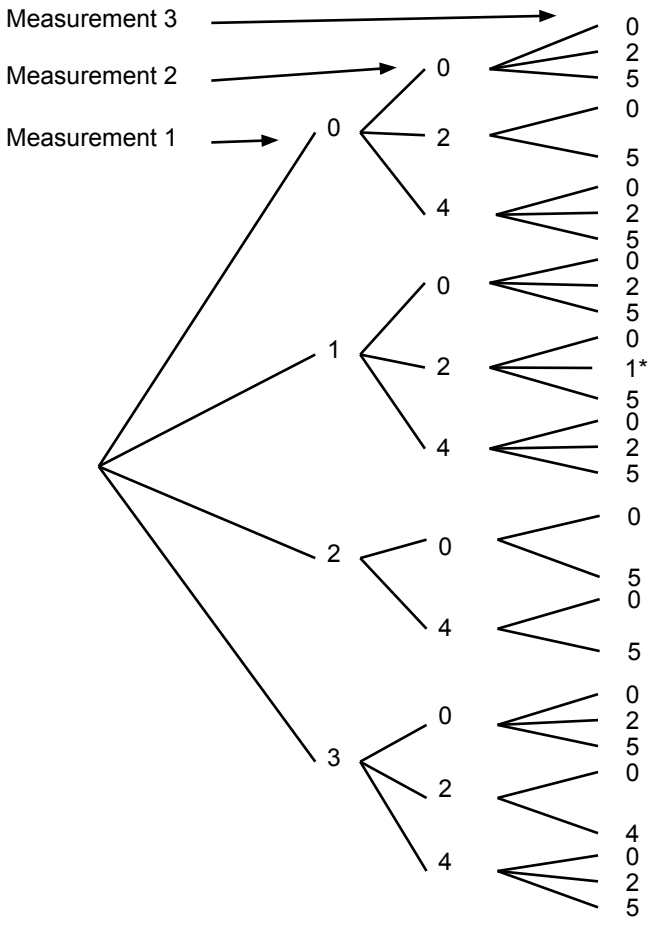


Figure 4.2: Tree representation of the hypotheses in hypothesis-oriented MHT.

turns the k best hypotheses.

The same example in Figure 4.1 is used to explain track-oriented MHT. Each track is represented with a tree as shown in Figure 4.3. Each path down the tree represents a candidate track: a path that the object of interest could have followed. Tracks are updated with all measurements within their validation region with each branch indicating a possible track to measurement association. Moreover, each measurement is used to start a new track. This implies that tracked objects may not be detected in a frame. Missed detections are indicated with a zero and new tracks are assigned indices that are one more than the existing number of tracks.

As more measurements are received with every scan, the depths of the trees increase and new trees are created. As is the case with hypothesis-oriented MHT, the data association problem may be solved after a given number of frames has been processed. In this case, the problem must be augmented with

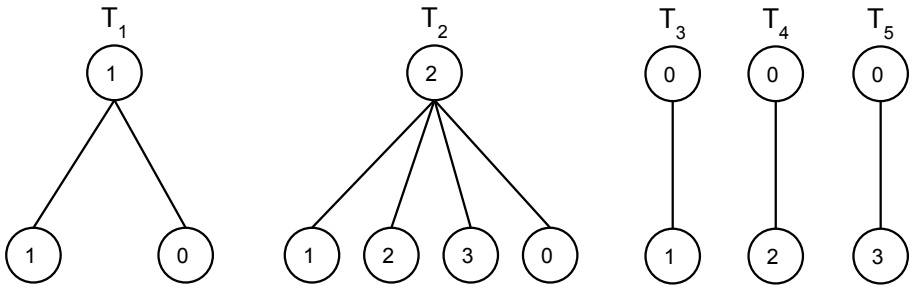


Figure 4.3: Track expansions obtained using track-oriented MHT.

constraints to ensure that each measurement is used only once. At most one path must be selected from a tree. This problem can be posed as an integer programming problem [63].

In constructing hypothesis-oriented MHT trees, false alarms are explicitly taken into consideration. In track-oriented MHT, every measurement is used to initialize a new track. In this case, false alarms become apparent either during the construction of the trees or once the assignment problem has been solved. False alarms initialize false tracks which are identified in solving the assignment problem. Note that a combination of the track hypotheses from the trees in Figure 4.3, taking into account the constraints on measurements, yields the list of hypotheses in Figure 4.2. In the simple single window application of MHT the number of hypotheses far exceeds the number of tracks. Hence, track-oriented multiple hypothesis tracking is used in the thesis. It is the least memory and computation intensive of the two.

4.3 Evaluation of tracks

The major component of track-oriented MHT is the calculation of the probability that a measurement was generated by a track with which it is associated. The preferred approach is to use Bayes' theorem and combinatorial analysis of the data association problem to derive the joint probability of a hypothesis. The result is the probabilistic expression presented by Reid [68] for hypothesis-oriented MHT. The paper also presented a novel method for propagating multiple hypotheses. The track-oriented MHT formulation is then derived from this result using reasonable assumptions [12].

For the purpose of implementation, it is easier to use the mathematically equivalent log-likelihood ratio proposed by Sittler [77]. The likelihood ratio is the ratio of the probability of a measurement having originated from a track to the probability of this measurement having a different origin. The use of log-likelihoods allows the formula to be expressed in terms of sums as opposed

to products. In this section we give the log-likelihood ratio as derived in Sittler [77]. We then discuss what each term of the ratio means and how it is used for track management. This section is based on Sittler [77] and takes into account the MHT framework. For example, every measurement is used to initialize a new track in MHT. In Sittler [77], however, the last measurement of a false track is used to initialize a new track.

The log-likelihood ratio function of track i through time m_i as derived in [77] is as follows:

$$\begin{aligned} L_i = & \ln \frac{\lambda_0 \lambda_s}{\lambda_N(\lambda_s + \frac{1}{\tau_0})} + \sum_{k=2}^{m_i} \ln \frac{\lambda_s e^{-(\lambda_s + 1/\tau_0)\, t_k}}{\lambda_N |2\pi(\epsilon_k + t_k\, \mathbf{S}_{ik})|^{\frac{1}{2}}} \\ & - \frac{1}{2} \sum_{k=2}^{m_i} (\mathbf{z}_{ik} - \mathbf{H}_k \hat{\mathbf{x}}_{ik})' (\epsilon_k + t_k \mathbf{S}_{ik})^{-1} (\mathbf{z}_{ik} - \mathbf{H}_k \hat{\mathbf{x}}_{ik}) \\ & + \ln \left[\frac{\lambda_s e^{-(\lambda_s + 1/\tau_0)(T - v_i)} + \frac{1}{\tau_0}}{\lambda_s + \frac{1}{\tau_0}} \right]. \end{aligned} \tag{4.3.1}$$

where t_k is the time between the $(k-1)$th and the kth measurements (the last received and current measurements). An assumption made in deriving the score is that the numbers of new objects and false alarms may be modelled using Poisson distributions with parameters λ_0 and λ_N, respectively. λ_0 is the average number of new objects per unit time per unit area of the region under surveillance. Similarly, λ_N is the number of false measurements per unit time per unit area. The observations on a single object are assumed to follow a Poisson process with the average rate λ_s.

Each object is assumed to persist independently through a length of time that has an exponential distribution with time constant τ_0. This is the mean track length. The assumption is reasonable because a new track will be created when two or more objects interact. In fact, MHT is expected to correctly handle short interactions and occlusions. Any extended interactions and occlusions will be handled using re-identification of objects. The kth measurement associated with track i is $\mathbf{z}_{ik}$. The predicted state of the track is $\hat{\mathbf{x}}_{ik}$. $\mathbf{S}_{ik}$ is the innovation covariance matrix as defined in (4.1.1). T is the current time and v_i is the last time a measurement associated with track i was received. Note that tracks with larger log-likelihood ratios are preferred.

The first term of (4.3.1) is the score assigned to a new track. When $k = 1$ the summation cannot be evaluated, and $T - v_i = 0$. Hence, all but the first term evaluate to zero. The result is the negative cost of initializing a new track. The second term is evaluated when a measurement is available and it takes into account the uncertainty in the state of the objects. It is the only term that can contribute positively to the score. To ensure that this can happen,

the parameter ϵ_k must be chosen such that

$$\left[\ln \frac{\lambda_s e^{-(\lambda_s+1/\tau_0)\,\delta t}}{\lambda_N |2\pi(\epsilon_k + \delta t\, \mathbf{S}_{ik})|^{\frac{1}{2}}}\right]_{\delta t=0} > 0. \tag{4.3.2}$$

Note that ϵ_k may change with time because the innovation error covariance matrix $\mathbf{S}_{ik}$ changes with time.

The third term in (4.3.1) is also evaluated when a measurement is available. This is the distance between the predicted and the actual measurements. Any disparity between the two affects the score negatively. The best outcome occurs when these two coincide and the term does not contribute to the score. The worst outcome is when the measurement is on the edge of the validation region and the term contributes $-\lambda$. Here λ is the threshold used to define the validation region as shown in (4.1.1) and (4.1.2).

The fourth term in (4.3.1) does not contribute to the log-likelihood ratio if there are no missed detections, i.e. $v_i = T$. Any missed detection contributes negatively to the score. Note that v_i, the last time a measurement was received, is finite. Therefore, letting $T \to \infty$ implies that $T - v_i \to \infty$ which implies that the last measurement that supported the track was received a long time ago. Therefore, this track must be terminated. Evaluating the limit yields the cost of terminating a track:

$$\lim_{T\to\infty} \ln \frac{\lambda_s e^{-(\lambda_s+1/\tau_0)(T-v_i)} + \frac{1}{\tau_0}}{\lambda_s + \frac{1}{\tau_0}} = -\ln(1 + \lambda_s\, \tau_0). \tag{4.3.3}$$

The formula for calculating the log-likelihood ratio of a track is intuitive. The larger the log-likelihood of the track, the more favourable the track. Missed detections should impact the score negatively and this is reflected in the formula. The effect of missed detections is further incorporated in the numerator of the second term. Sittler [77] proposes that it be an increasing function of time. This is reasonable because the longer we wait for supporting measurements the more uncertain we should be about the state of the objects being tracked.

The parameters τ_0, λ_0, λ_s and λ_N impact the ratio in an intuitive manner as well. The ratio is not sensitive to τ_0. This is because it is always used in conjunction with λ_s and for practical systems $\lambda_s \gg \frac{1}{\tau_0}$ [77]. Qualitatively, an object may be tracked for a long time and yet the state estimation may be far off the mark due to drifting problems [43]. So, the expected length of a track should be immaterial in determining the quality of a track. Taking the

quantitative arguments into account, the entrance score may be approximated as

$$\ln \frac{\lambda_0 \lambda_s}{\lambda_N(\lambda_s + \frac{1}{\tau_0})} \approx \ln \frac{\lambda_0}{\lambda_N} = \ln \lambda_0 - \ln \lambda_N. \tag{4.3.4}$$

This indicates that the entrance score is not sensitive to the expected number of observations on an object per unit time λ_s. A larger value of λ_N, the number of false alarms per unit area per unit time, means that a larger penalty must be paid in order to start a track. In contrast, a larger value of λ_0, the number of new objects per unit area per unit time, decreases the penalty that must be paid. The parameter λ_s comes into play when calculating the cost of terminating a track in (4.3.3). In this case, note that $\lambda_s \tau_0$ approximates the expected number of measurements that support a track after the last one was received. Therefore, if the expected number of measurements is very large, then a large price must be paid to terminate that track.

Two important values for the log-likelihood ratio have been mentioned so far. The first is the entrance score of a new track:

$$A = \ln \frac{\lambda_0 \lambda_s}{\lambda_N(\lambda_s + \frac{1}{\tau_0})} < 0. \tag{4.3.5}$$

The second is the cost of terminating a track:

$$B = -\ln(1 + \lambda_s \tau_0) < 0. \tag{4.3.6}$$

A track is considered to be of a valid object if its log-likelihood score eventually exceeds zero. Therefore, the third log-likelihood value that is also important is $C = 0$. These values are important in quantifying the track management concepts which are track initialization, confirmation and termination. We now consider these track management concepts.

4.3.1 Track initialization

Each measurement is used to initialize a new track with a log-likelihood score of A calculated using (4.3.5). However, the measurement may be due to false alarms and clutter. Thus, a single measurement is not sufficient to conclude that a track is correct and further data is required to support the track. If supporting measurements are received, only the second and third terms in (4.3.1) are evaluated, possibly increasing the score. However, if no supporting measurements are received, these terms cannot be evaluated and the last term worsens the score. A track is considered to be from a valid object if its score eventually exceeds $C = 0$ without ever dropping below the value A. Such a track is referred to as an initiated track. Otherwise, the track score drops below A implying that the track was initialized using a false measurement, i.e. it is a false track, and must be deleted.

4.3.2 Track confirmation

The score of an initiated track has another threshold, $-B > 0$, to cross and yield what is referred to as a confirmed track. The score of a confirmed track does not depreciate based on the passage of time. That is, the last term in (4.3.1) is ignored when updating the log-likelihood ratio of a confirmed track. However, missed detections still impact the score negatively through the innovation covariance matrix S_{ik} in the second term. Recall that this covariance matrix gets larger with every missed detection to account for the increasing uncertainty in the state of the tracked object. In contrast, the score of an initiated track, which has not been confirmed yet, depreciates with the passage of time as modelled by the last term in (4.3.1). Missed detections and measurements that are not good enough (the sum of the second and third terms is less than zero) may still drive the track score below A. If this happens, the entire track may be dismissed as a false track and deleted.

4.3.3 Track management

The process of making decisions on confirmed tracks is referred to as track management. At this point, the log-likelihood ratio threshold $A+B<0$ is of interest. The score of a newly confirmed track is reset to zero and this point serves as a reference of the end point of that track. As mentioned earlier, missed detections affect the score through the increasing innovation covariance matrix S_{ik} only, the fourth term in (4.3.1) being ignored. Any new high value in the score serves as a reference to the end point of the confirmed track. A confirmed track must be removed from the set of track hypotheses and saved to disk only if its log-likelihood score drops below $A+B$ indicating that the track terminated some time ago. The track segment after the reference point up to the termination point is first discarded as a false track before the confirmed track is saved. In our system the saved confirmed tracks are candidates for re-identification. This means that the object that generated those tracks may come back into the scene, re-detected, and their tracks extended.

4.4 Additional hypotheses reduction methods

Pruning of trees by deleting unlikely track hypotheses is an intrinsic part of MHT, particularly track-oriented MHT as outlined above. In this section we consider other methods for managing the growth in the number of hypotheses, thus minimizing the computational and memory costs. The simplest of these is to only consider measurements that are within the validation region of a track. A preferred method of defining the validation region when a Kalman filter is used for tracking is defined by (4.1.1) and (4.1.2). A valid measurement can fall outside this region possibly because the object motion does not satisfy the Kalman filter assumptions. This can be minimized by a careful choice of the

threshold λ and the Kalman filter parameters.

The other method is to use a fixed window W where W is the number of frames that must be processed before the data association problem is solved. One extreme is to make data association decisions as data is received, i.e. $W = 1$, which clearly defeats the idea of MHT. The other extreme is batch processing where the data is provided all at once, e.g. as a video recording. Using a small fixed window, say $W = 5$, is amenable to online, real-time surveillance systems. In this case the measurement assignment problem is solved every W frames. Tracks that are not re-detected in a certain number $N < W$ of consecutive frames are assumed to have terminated and are removed. This eliminates tracks that are due to clutter. In our implementation, such tracks are retained if they were confirmed tracks and are candidates for re-identification.

Another method is hypothesis merging where similar tracks are merged [1]. Two tracks may be merged if they share N measurements and are closer than a predefined threshold. A similarity measure between tracks must be defined. Also, the two tracks may be merged if, at a particular instant, one track is associated with a measurement and the other track is associated with a dummy measurement. Lastly, clustering of hypotheses can also be used to reduce the computational cost of track-oriented MHT. It is a divide and conquer strategy [68]. Hypotheses that have at least one measurement in common are grouped together before the data association problem may be solved. Thus, instead of solving one large problem, a number of small sub-problems is solved. This means that the solution may be parallelized at a higher level.

4.5 Conclusion

Tracking multiple targets is complex mainly because measurements from the object detection stage are not labelled. Moreover, filtering methods assume a one-to-one association between the tracks and measurements. Data association methods solve this measurements origin problem. We classified data association methods as either classical or particle filter approaches. Particle filter approaches may suffer from the curse of dimensionality but techniques such as Markov random fields have been proposed to solve this.

In this thesis we use the classical approach to data association, specifically multiple hypothesis tracking (MHT). This method was chosen because it is a deferred logic method and provides track initiation, management and termination facilities. MHT is facilitated by the log-likelihood score which is used to quantify the quality of hypotheses. The higher the log-likelihood score the better the tracker. We presented this formula and our analysis lead to the conclusion that its parameters affect the score in an intuitive manner.

Chapter 5

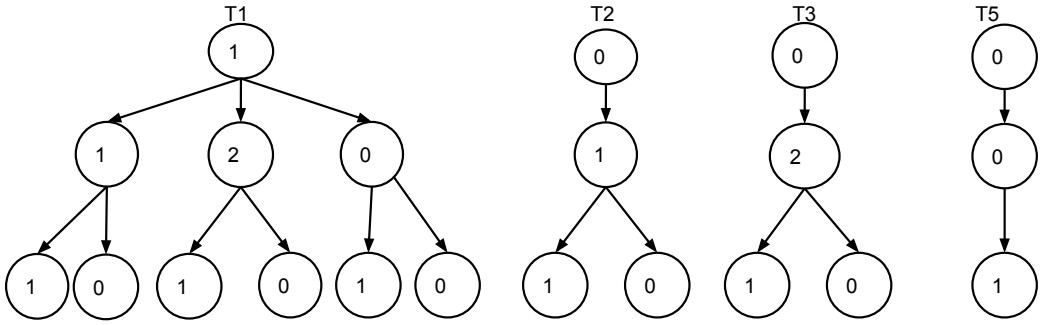
Integer programming problem

A collection of tree structures is used to represent the track-oriented multiple hypothesis (MHT) problem. Each tree is a collection of prospective tracks that could be followed by an object. Therefore, at most one track hypothesis in a tree may be the true track. This is implied by the MHT assumption that there is at most one detection per object. Also, each measurement may be used at most once. Each track hypothesis has a log-likelihood score which quantifies the quality of that track. The track hypothesis with the largest log-likelihood ratio is preferred. This is a combinatorial (assignment) problem that can be posed as an integer programming problem (IPP).

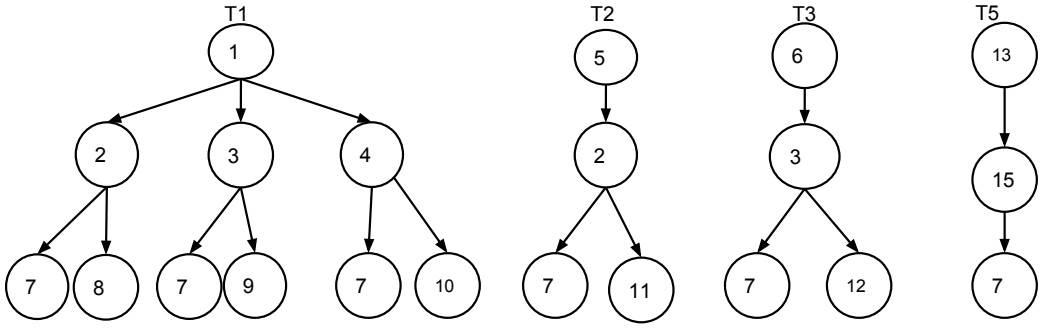
Morefield [58] and Poore et al. [63] represent the multiple target tracking problem as a multi-dimensional assignment problem. One advantage of these approaches is that they do not enumerate the track hypotheses. Moreover, they are presented as alternatives to MHT. In our case, the multi-dimensional assignment problem is derived from MHT. This means that we take advantage of the track management capabilities of MHT. Also, the assignment problem we are solving is smaller because some of the variables are essentially fixed during the construction of the hypotheses. This is due to the condition that a track may only be associated with measurements that are within its validation region. The remainder of this chapter outlines the steps required to pose the MHT problem as an integer programming problem.

5.1 Posing the MHT problem as an IPP

The first step in posing the MHT problem as an integer programming problem is to ensure that all measurements are assigned unique indices. In constructing the track-oriented MHT tree, at each level the measurements are numbered from one to the total number of measurements received at that level. A single dummy measurement, labelled zero, is maintained and is assigned as many times as is necessary. This numbering scheme is appropriate for visualization.



a. Default labelling scheme used in constructing the MHT tree.



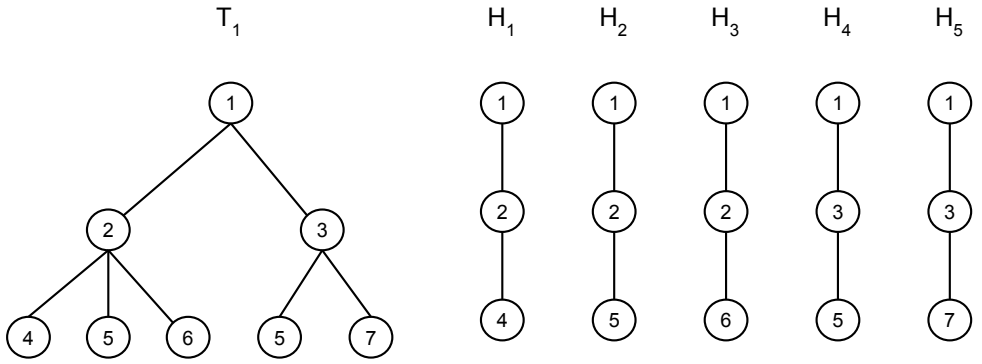
b. New labelling scheme ensures that all measurements, including dummy measurements, have unique indices.

Figure 5.1: Comparison of the different labelling schemes.

We introduce a new numbering scheme that ensures that each measurement in a window is assigned a unique index. This allows us to uniquely identify measurements without having to specify the tree level or frame from which they were obtained.

Suppose T_p measurements have already been processed in the previous scans. The next batch of M measurements will then be re-numbered from $T_p + 1$ to $T_p + M$. Any dummy measurement required is assigned an index that is one more than the total number of measurements. Thus, every dummy measurement is assigned a unique index. This is in contrast to the approaches in Morefield [58] and Poore et al. [63] which maintain a single dummy measurement. This approach is suitable for a formulation that focuses on the presence or absence of nodes rather than edges. It is easier to formulate the problem in terms of nodes because they represent states of objects which are the major concern. See Figure 5.1 for a comparison of the old and new node numbering schemes.

We should point out that the actual tree structure is not used in the implementation. The track hypotheses from the same tree are maintained separately as show in Figure 5.2. Each one of these free standing track hypotheses is a



a. Track-oriented MHT tree. b. Track hypotheses in the implementation.

Figure 5.2: The tree view of the track-oriented MHT vs. the representation used for implementation.

path down the tree from the node to one of the leaf nodes. This representation simplifies the mathematical problem statement and is adopted from here onwards. This clearly increases the memory requirements of the implementation as multiple copies of the same measurement must be maintained but, most importantly, it simplifies the implementation. At this point we can uniquely identify measurements but cannot tell in which one of the free standing track hypotheses they were used.

The final step is to assign unique indices to each of the nodes in each free standing track hypothesis. This is performed within the integer programming problem solver and is simply a counting process where node i is assigned the index i. The order in which the nodes are counted is immaterial. Although these nodes may have unique indices, a subset may refer to essentially the same measurement. Thus, a mapping to keep track of these relations is constructed concurrently:

$$\tau : I_n \rightarrow I_m, \tag{5.1.1}$$

$$I_n = \{1, 2, \ldots, N\} = \bigcup_{k=1}^{K} \{v_{k_i}\}_{i=0}^{W}, \tag{5.1.2}$$

$$I_m = \{1, 2, \ldots, M_W\}, \tag{5.1.3}$$

where I_n and I_m are sets of unique node and measurement indices, respectively. M_W is the total number of measurements used in the current window. K is the number of free standing track hypotheses and W is the window size. The mapping enables the construction of the constraint that a measurement may be used at most once. See Figure 5.3 for a graphical representation of this

process.

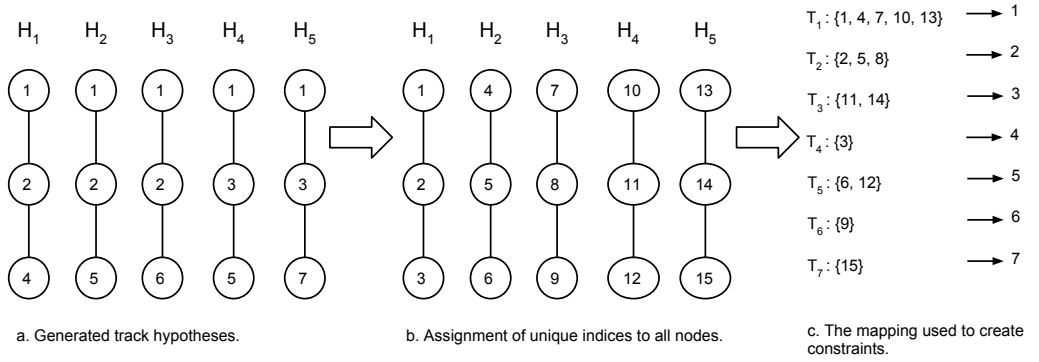


Figure 5.3: Constructing a mapping from the set of unique node indices to the set of unique measurement indices.

Of particular interest in this definition is the notation used to define elements of I_n. An element of $v_{k_i} \in I_n$ encodes the index of a track hypothesis k as well as the level i of the measurement in the tree. A node marks the start of a track if $i = 0$ and the end of a track if $i = W$. Given this definition, the kth track hypothesis, which is a path in a graph, can then be defined as follows:

$$T_k = e_{k_0} e_{k_1} \cdots e_{k_{W-1}}, \tag{5.1.4}$$

$$e_{k_i} = (v_{k_i}, v_{k_{i+1}}) \in E, \quad k = 1, 2, \ldots, K. \tag{5.1.5}$$

Here E is a set of edges (v_i, v_j) in the graph theoretic sense which indicates that there is a link from node v_i to node v_j.

Each node is a placeholder for a measurement. Each measurement may be used at most once. Moreover, each measurement may be associated with more than one node. Therefore, we define an indicator function on node (k_i):

$$\alpha(v_{k_i}) = \alpha_{k_i} = \begin{cases} 1, & \text{if there exists an } m \in I_m \text{ such that } \tau(v_{k_i}) = m \\ & \text{and } m \text{ is assigned to track } k, \\ 0, & \text{otherwise.} \end{cases} \tag{5.1.6}$$

This will allow us to say whether a valid measurement was used to update track k. Track k is a valid track if and only if $\alpha(v_{k_i}) = 1$ for all $i = 0, 1, \ldots, W$. Moreover, each node v_{k_i} has a negative log-likelihood ratio score n_{k_i} which was calculated when the tree was constructed. This log-likelihood score is of interest only if $i = W$ which marks the end point of a track and quantifies the quality of the entire track.

We are now ready to formulate the optimization problem which comprises the objective function and a set of constraints on the variables. The objective function is defined in terms of the negative log-likelihood scores of the tracks:

$$\arg \min_{\{\hat{\alpha}_{k_W}\}_{k=1}^{K}} \sum_{k=1}^{K} \alpha_{k_W} \, n_{k_W} \tag{5.1.7}$$

subject to the constraints that

$$\sum_{k=1}^{K} \sum_{i=0}^{W} \alpha_{k_i} \cdot \delta(\tau(v_{k_i}), m) \leq 1, \qquad \text{for } m = 1, 2, \ldots, M_W, \tag{5.1.8}$$

$$\delta(\tau(v_{k_i}), m) = \begin{cases} 1, & \text{if } \tau(v_{k_i}) = m, \\ 0, & \text{otherwise,} \end{cases} \tag{5.1.9}$$

$$\sum_{i=0}^{W} \alpha_{k_i} - \alpha_{k_W}(W+1) = 0, \qquad \text{for } k = 1, 2, \ldots, K. \tag{5.1.10}$$

The first constraint (5.1.8) means that a measurement may be used at most once. It simply counts the nodes where a measurement m could be used as reflected by $\delta(\tau(v_{k_i}), m)$ while taking into account whether the node is actually used as reflected by α_{k_i}. The second constraint (5.1.10) means that for a track to be valid all the measurements on that track must be used. That is, track k is a valid track if and only if $\alpha(v_{k_i}) = 1$ for all the nodes v_{k_i} in the track. Note that the length of a track, $W + 1$, is multiplied by the indicator function of the leaf node of that track. In this sense the leaf node represents the entire node and if it is used then all the nodes in that track must be used. This has the effect that a track either has length 0 or length $W + 1$. The optimization problem may be solved using an integer programming library. We use the Symphony Callable library [66].

5.2 Conclusion

This chapter outlined the steps required to pose the MHT problem as an integer programming problem. Naturally, an optimization problem comprises of an objective function along with a set of constraints. The objective function is based on the negative log-likelihood scores of tracks. The constraints derive from the MHT assumptions that a measurement may be assigned to at most one track and that there is only one detection per object. The resultant optimization problem is solved using the Symphony Callable library.

Chapter 6

Learning object appearances

Standard tracking approaches like the mean-shift [20] and the Kalman filter (Section 3.2) assume that the object of interest is never completely occluded. This and unsuitable models can result in tracking failure. These methods do not directly address what happens after tracking failure. Instead, new tracks are initialized after tracking failure or when objects reappear. In this thesis machine learning algorithms are used to learn the object-specific appearances which are then used to uniquely re-identify objects when they reappear or after tracking failure.

At least three aspects are essential for learning the appearance model of an object for re-identification purposes. First, the features used to represent the appearance of objects must be discriminative. In the case of recognising people, biometric features such as the face, iris and gait could be used to re-identify people, but most surveillance video have low resolution or are difficult to segment [21]. As a result it is necessary to model the global appearance of each object. This leads to the second aspect which is that models must be learnt online because discriminative appearances of tracked objects cannot be known in advance.

Learning algorithms that have been used to learn object appearance include online versions of support vector machines (SVMs) [92], random forests [24], boosting [35] and density mixture models [39]. Boosting involves using a linear combination of weak classifiers, i.e. classifiers that are accurate at least fifty percent of the time. Online learning methods may be classified as either generative [39], [43] or discriminative [35]. Generative methods learn a model that represents the appearance of an object. Discriminative methods learn a decision boundary that best separates the object from the background.

Last, a strategy must be devised to decide which samples to use to update the model. Each update can introduce errors which can lead to the classifier not learning the appearance of the intended object [89]. The errors may be due to

the inaccuracies in segmenting the object. Moreover, some of the background will be treated as part of the foreground no matter how tight the bounding box.

6.1 Object representation

The subject of object representation was briefly treated in Chapter 1. Colour features are used in this thesis mainly because they are very discriminative. However, colour features lose the shape information. Yang et al. [89] note that no single feature descriptor is robust enough to deal with all kinds of situations. Thus, a combination of features is often used. However, colour features should be sufficient for our goal which is to demonstrate that a global appearance model of an object can be built and subsequently used to re-identify objects.

Several colour features are compared in the work by Van de Sande et al. [84]. Of particular interest is the invariance of colour features to light intensity change and shift. Light intensity change includes shadows and shading. Light intensity shift corresponds to objects being highlighted under a white light source and scattering of a white source. The results indicate that the RGB histogram is not invariant to light intensity change and shift. In contrast, the hue and saturation histograms are invariant to light intensity change and shift. Moreover similar colours are closer together in the hue-saturation-value (HSV) colour space than they are in RGB colour space. As a result we use the HSV colour histogram in the thesis.

The HSV colour histogram is generated by concatenating the histograms of the individual channels. The vote of a pixel is weighed by how far it is from the centre of the bounding box, using the Epanechnikov kernel

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2), & \text{if } |u| \leq 1; \\ 0, & \text{otherwise.} \end{cases} \tag{6.1.1}$$

Here u is the distance from a pixel to the centre of the bounding box, normalized by the distance from the centre to a point on the edge of the bounding box. The point on the edge of the bounding box is chosen such that the pixel lies on the line joining the centre to this point. We experimented with uniform and triangular kernels but the Epanechnikov kernel performed better.

6.2 Selecting training samples

Kalal et al. [43] classify strategies used to decide whether or not to update a model with a training sample as every-frame [34], [51] and selective update [35], [92] strategies. The every-frame update strategy implicitly assumes

that the tracker is always correct. This class implies quick adaptation but can accelerate tracker failure [43]. Selective update strategies use either semi-supervised [35], [92] or unsupervised [39], [43] learning to make this decision. Jepson et al. [39] learn the density mixture model of objects using wavelet-like features. The mixture has three components with the third one representing an outlier. An advantage of this method is that it is a clustering algorithm and as long as samples support a cluster, that cluster will be retained.

Yu et al. [92] use co-training of generative and discriminative classifiers to automatically label training samples. The two classifiers are trained using conditionally different views or features. The generative classifier represents the global object appearance and contains a set of subspaces. Each subspace approximates the local variations in the appearance of the object. The discriminative classifier is an SVM that is trained online using histogram of oriented gradient features. Co-training is accomplished by training one classifier with the samples that were confidently labelled by another classifier. This framework learns the global appearance of objects and approaches tracking as classification. However, in our case we perform filtering and data association which can be understood to provide confident labelling of training samples. Then, the component of co-training that is missing is a generative classifier that models the global appearance of objects.

Kalal et al. [43] implement a novel unsupervised strategy to decide whether to use a sample to train the classifier. A distance measure between the classifier and a given measurement is defined in terms of the normalized cross-correlation between a pair of measurements. The process starts with a measurement that is similar to the classifier (the distance between this measurement and the classifier is less that a given threshold). A sequence of measurements is retained until a measurement that is similar to the classifier is found. In this case all the retained measurements are used to train the classifier. The logic behind this strategy is that if the tracker drifts it is unlikely that it will accidentally return to the object of interest. The increase in the distance measure is then attributed to major changes in the appearance of the object.

The approach in this thesis is similar to the one in Kalal et al. [43] and is influenced by the multiple hypothesis tracker used for data association. During initialization, a sequence of measurements is retained and subsequently used to train the classifier once the track score exceeds zero. During track maintenance, each measurement is used to train the classifier as long as the track score is greater than zero. If the score drops below zero then a sequence of measurements is retained until the score exceeds zero again. In this case all the retained measurements are used to train the classifier. The drop in the score is then attributed to a disagreement between the measurement and dynamic processes, which may be due to object motion dynamics that are inconsistent

with the model used.

6.3 Online learning of the global appearance model

Lee and Kriegman [51] use a manifold, i.e. a set of subspaces, and a transition matrix. Each subspace models the local variations in the appearance of an object. The transition matrix models the probability of moving between two subspaces. Learning in this case involves finding the subspace that best explains the query object and then updating all the subspaces. A drawback of this method is initialization which requires an offline learnt generic manifold.

The density mixture method used by Jepson et al. [39] is closely related to learning a set of manifolds. A disadvantage of this method is the fixed number of components in a mixture, three in this case, that must be specified in advance. As a result, it implicitly assumes that an object may have a known number of stable appearances. Zivkovic [94] presents an improvement to density mixture models which selects the optimal number of components in conjunction with estimating the parameters of those components. To our knowledge this extension has not been applied to learning of object appearance.

Kalal et al. [43] uses random forests to learn the appearance of objects. This class of machine learning algorithms is robust to labelling noise. Moreover, they are computationally efficient both during training and classification [72]. Empirical results by Statnikov et al. [78] indicate that support vector machines (SVMs) outperform random forests at least in microarray-based cancer classification. In addition, SVMs and boosting outperform a number of methods when learning in a high-dimensional space. Thus, we use SVMs to learn the appearance of objects in this thesis.

6.3.1 Support vector machines

Support vector machines are machine learning algorithms that seek to find a hyperplane (or a line in two dimensions) that optimally separates a set of training examples into two classes. Optimality means that the hyperplane is as far as possible from the closest sample of both classes [32]. Given a set of N training examples

$$\{\mathbf{x}_i, y_i\} \quad \text{where} \quad y_i \in \{-1, 1\},\ \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^D \quad i = 1, \ldots, N \tag{6.3.1}$$

the hyperplane is defined as

$$\mathbf{w} \cdot \mathbf{x} + b \ = \ 0 \tag{6.3.2}$$

where $\mathbf{w}$ is the normal of the hyperplane and $\frac{|b|}{\|\mathbf{w}\|}$ is the shortest distance between the hyperplane and the origin.

The hyperplane divides the data set such that each sample $\{\mathbf{x}_i, y_i\}$ satisfies either one of the following constraints:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq \quad 1 - \xi_i \quad \text{for} \quad y_i = \quad 1 \tag{6.3.3}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i \quad \text{for} \quad y_i = -1 \tag{6.3.4}$$

$$\xi_i \geq 0 \quad i = 1, 2, \ldots, N. \tag{6.3.5}$$

The slack variables ξ_i allow for some misclassification, while punishing them, in order to achieve the optimal solution. Training a support vector machine means determining the parameters $\mathbf{w}$ and b that solve the following optimization problem:

$$\min_{\mathbf{w},\xi,b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i, \tag{6.3.6}$$

such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq -\xi_i, \quad \xi_i \geq 0 \quad i = 1, 2, \ldots, N. \tag{6.3.7}$$

The larger the value of C the more the misclassifications are punished.

The Lagrangian formulation of the above optimization is required to yield the simpler convex and quadratic optimization for which optimized solvers exist. This is achieved by combining the objective function and the constraints to form a new objective function:

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{N} \mu_i \xi_i, \tag{6.3.8}$$

where α_i and μ_i are Lagrangian parameters. The objective function must be minimized with respect to $\mathbf{w}$, $\boldsymbol{\xi}$ and b, and maximized with respect to α_i and $\mu_i, i = 1, 2, \ldots, N$. At the optimal, the following must hold:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i, \tag{6.3.9}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \quad \Rightarrow \quad C = \alpha_i + \mu_i, \quad i = 1, 2, \ldots, N, \tag{6.3.10}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{6.3.11}$$

Substituting (6.3.9), (6.3.10) and (6.3.11) into (6.3.8) and simplifying yield the simpler optimization problem given by

$$\max_{\alpha_1,\ldots,\alpha_N} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\,\alpha_j\,y_i\,y_j\,\mathbf{x}_i\cdot\mathbf{x}_j\right], \tag{6.3.12}$$

such that

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \tag{6.3.13}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, N. \tag{6.3.14}$$

A sample $\mathbf{x}_i$ with a Lagrangian multiplier α_i such that $0 < \alpha_i \leq C$ is a support vector. If $0 < \alpha_i < C$ then the support vector is on the margin. A sample $\mathbf{x}_i$ with a weight $\alpha_i = 0$ is correctly classified and does not affect the hyperplane.

The kernel trick is used in order to train an SVM that nonlinearly separates data. The first step of this trick is the assumption that there exists a function $\phi : \mathcal{X} \to \mathcal{F}$ that maps the input space $\mathcal{X}$ into a high-dimensional space $\mathcal{F}$ within which the data is linearly separable. Moreover, the inner product must be defined for objects in $\mathcal{F}$. This is particularly important given that the linear SVM objective function (6.3.12) is defined in terms of inner products. The power of the trick is the realization that it is not necessary to know the functional form of $\phi(\mathbf{x})$. It is only necessary to know a function $k(\mathbf{x}, \mathbf{y})$, referred to as a kernel, defined in $\mathcal{X}$ such that

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{6.3.15}$$

The kernel used when the original data is linearly separable is the linear kernel $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}\cdot\mathbf{y}$. Another common kernel which is used in this thesis is the radial basis function (RBF)

$$k(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{y}\|^2}. \tag{6.3.16}$$

In our case we want to train the support vector machine using samples from the same class. The goal is to obtain a function $f(\mathbf{x})$ that demarcates a region $\mathcal{S} \in \mathcal{X}$ in the input space representative of the training data such that

$$\begin{aligned} f(\mathbf{x}) &\geq 0, \quad \text{if } \mathbf{x} \in \mathcal{S} \\ f(\mathbf{x}) &< 0, \quad \text{otherwise.} \end{aligned} \tag{6.3.17}$$

The function $\phi(\mathbf{x})$ transforms the input data into $\mathcal{F}$ where a hyperplane optimally separates the data from the origin. Thus the origin in $\mathcal{F}$ represents all

inputs that do not belong to the same class as the training data. Schölkopf et al. [74] derive the optimization problem

$$\min_{\mathbf{w},\boldsymbol{\xi},\rho} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - b, \tag{6.3.18}$$

subject to

$$\mathbf{w}\cdot\phi(\mathbf{x}_i) \geq b - \xi_i, \quad \xi_i \geq 0 \quad i = 1, 2, \ldots, N, \tag{6.3.19}$$

where $\boldsymbol{\xi} = \{\xi_1, \xi_2, \ldots, \xi_N\}$ is a set of slack variables, b is the offset of the hyperplane and $\nu \in (0, 1)$ is a parameter chosen by the user. Schölkopf et al. [74] prove that ν is an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors.

We could proceed in the same fashion as for the binary classifier and that would imply batch learning. In our case, however, the object appearance must be learnt online because training samples are provided one at a time. Cauwenberghs and Poggio [17] present an exact solution to incremental training of two-class SVMs. Davy et al. [25] extend this method to online training of the one-class SVM. The framework allows learning and un-learning of a single example at a time. The extension uses a fixed window of training examples in a sense that when a new sample is used for training the oldest sample is removed from the model. The fixed window is a drawback as it means that we assume that earlier appearances will not reappear.

The optimization problem in its current form was derived with batch processing in mind. Kivinen et al. [48] derive an equivalent objective function by considering classification as a risk minimization problem. The resultant optimization problem is solved using stochastic gradient descent, which considers one sample at a time to find the optimal point and essentially results in online learning. This is in contrast to classical gradient descent where all samples are required for one iteration. The latter is not always possible either because the data set is too large to fit in memory or because the data arrives sequentially and decisions must be made at the same time. Stochastic gradient descent is used in this thesis.

6.3.2 Online one-class SVM

In this thesis we use stochastic gradient descent to train the one-class SMV as derived by Kivinen et al. [48]. The first step is to transform the optimization problem (6.3.18)–(6.3.19) into a form equivalent to the one by Kivinen et al. [48]. This is accomplished by multiplying the objective function with a

constant ν:

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \quad \frac{\nu}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}\xi_i - b\nu, \tag{6.3.20}$$

subject to

$$\mathbf{w}\cdot\phi(\mathbf{x}_i) \geq b - \xi_i, \quad \xi_i \geq 0 \quad i = 1, 2, \ldots, N. \tag{6.3.21}$$

This changes the value of the objective function at the optimal point without changing the optimal point. Note that at the optimal all the slack variables ξ_i, $i = 1, 2, \ldots, N$, satisfy

$$\xi_i = \begin{cases} b - \mathbf{w}\cdot\phi(\mathbf{x}_i), & \text{if } \mathbf{x}_i \in \text{NSV}, \\ 0, & \text{otherwise.} \end{cases} \tag{6.3.22}$$

NSV is a set of support vectors that are not on the margin. If the slack variable do not satisfy (6.3.22) the optimal solution has not been found. Given that $\xi_i \geq 0$ for all i, (6.3.22) may be written compactly as

$$\xi_i = \max(0, b - \mathbf{w}\cdot\phi(\mathbf{x}_i)). \tag{6.3.23}$$

Finally, we rewrite the optimization problem in the same form as that in Kivinen et al. [48]:

$$\min_{\mathbf{w},b} \frac{\nu}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}\max(0, b - \mathbf{w}\cdot\phi(\mathbf{x}_i)) - b\nu, \tag{6.3.24}$$

which is equivalent to

$$\min_{\mathbf{w},b} \frac{\nu}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}\left[\max(0, b - \mathbf{w}\cdot\phi(\mathbf{x}_i)) - Nb\nu\right]. \tag{6.3.25}$$

The objective function is referred to as the regularized empirical risk. The first term is the regularization function which is used to measure the complexity of the function $f(\mathbf{x})$. The second term is the empirical risk. Thus, given $\mathbf{w}$ and b, and by extension the decision function $f(\mathbf{x})$, and the set of training examples $\mathcal{S}$, the regularized empirical risk is

$$R_{\text{reg}}[\mathbf{w}, b, \mathcal{S}] = \frac{\nu}{2}\|\mathbf{w}\|^2 + R_{\text{emp}}[\mathbf{w}, b] \tag{6.3.26}$$

$$R_{\text{emp}}[\mathbf{w}, b] = \frac{1}{N}\sum_{i=1}^{N}\left[\max(0, b - \mathbf{w}\cdot\phi(\mathbf{x}))) - Nb\nu\right] \tag{6.3.27}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\ell(\mathbf{w}, b, \mathbf{x}). \tag{6.3.28}$$

The loss function $\ell(\mathbf{w}, b, \mathbf{x})$ punishes the incorrect classification of the sample $\mathbf{x}$.

We are interested in online learning where a single sample $\mathbf{x}$ is presented at a time. The goal is to find a sequence of parameters $\{(\mathbf{w}_n, b_n)\}_{n=1}^{N+1}$, hence a sequence of decision functions $\mathbf{f} = \{f_1, f_2, \ldots, f_{N+1}\}$. Here the initial parameter set $(\mathbf{w}_1, b_1)$ is arbitrary and $(\mathbf{w}_n, b_n)$, $n > 1$, is obtained after observing the $(n-1)$th training sample, that is

$$\mathbf{w}_n = \sum_{i=1}^{n-1} \alpha_i^n \phi(\mathbf{x}_i) \tag{6.3.29}$$

$$f_n(\mathbf{x}) = \mathbf{w}_n \cdot \phi(\mathbf{x}) - b_n. \tag{6.3.30}$$

A superscript is used for the Lagrange multipliers α_i^n, $i = 1, 2, \ldots, n-1$, to emphasize that they evolve as more samples are used to train the SVM.

As a result of the interest in online learning we define the instantaneous approximation of the regularized empirical risk R_{reg}:

$$R_{\text{inst}}[\mathbf{w}_n, f_n, \mathbf{x}_n] = R_{\text{reg}}[\mathbf{w}_n, b_n, \mathbf{x}_n] \tag{6.3.31}$$

$$= \left[\frac{\nu}{2}\|\mathbf{w}\|^2 + \ell(\mathbf{w}, b, \mathbf{x}_n)\right]_{\mathbf{w}=\mathbf{w}_n, b=b_n} \tag{6.3.32}$$

$$= \left[\frac{\nu}{2}\|\mathbf{w}\|^2 + \max(0, b - \mathbf{w}\cdot\phi(\mathbf{x}_n)) - b\nu\right]_{\mathbf{w}=\mathbf{w}_n, b=b_n}. \tag{6.3.33}$$

Thus $\ell(\mathbf{w}_n, b_n, \mathbf{x}_n)$ is the loss the decision function incurs when it tries to predict whether the training sample $\mathbf{x}_n$ belongs to the same class as the historic samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}\}$.

At this point we introduce stochastic gradient descent which enables the approximation of the optimal solution using a single sample. Stochastic gradient descent methods perform gradient descent with respect to the instantaneous regularized risk [48]. In our case the update rules are given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta_n \frac{\partial}{\partial \mathbf{w}} R_{\text{inst}}(\mathbf{w}, b, \mathbf{x}_n)\Big|_{\mathbf{w}=\mathbf{w}_n, b=b_n} \tag{6.3.34}$$

$$b_{n+1} = b_n - \eta_n \frac{\partial}{\partial b} R_{\text{inst}}(\mathbf{w}, b, \mathbf{x}_n)\Big|_{\mathbf{w}=\mathbf{w}_n, b=b_n}, \tag{6.3.35}$$

where η_n is the learning rate which may change with every received training sample $\mathbf{x}_n$ at time n. Kivinen et al. [48] use the dynamic learning rate

$$\eta_n = \frac{1}{n} \tag{6.3.36}$$

which has been shown to achieve the best convergence speed and satisfies the conditions

$$\sum_{i=1}^{\infty} \eta_i^2 < \infty, \qquad \sum_{i=1}^{\infty} \eta_i \to \infty \tag{6.3.37}$$

that are a requirement for convergence [13].

Evaluating the derivative in (6.3.34) and then substituting (6.3.29) into the resultant expression we obtain the update formulae for α_i^n, $i = 1, 2, \ldots, n$, which are given below (we omit the detail of their derivations because they are the same as those obtained by Kivinen et al. [48]):

$$\alpha_i^{n+1} = (1 - \nu\eta_n)\alpha_i^n, \quad i = 1, 2, \ldots, n, \tag{6.3.38}$$

$$\alpha_{n+1}^{n+1} = \begin{cases} \eta_n, & \text{if } f_n(\mathbf{x}_n) < 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6.3.39}$$

The update rule for b_n is different from the one derived in the work by Kivinen et al. [48] and is therefore derived here in detail. We first evaluate the derivative in (6.3.35):

$$\frac{\partial}{\partial b} R_{\text{inst}}[\mathbf{w}, b, \mathbf{x}_n]\Big|_{\mathbf{w}=\mathbf{w}_n, b=b_n} = \begin{cases} 1 - \nu, & \text{if } b_n - \mathbf{w}_n \cdot \phi(\mathbf{x}_n) > 0, \\ -\nu, & \text{otherwise.} \end{cases} \tag{6.3.40}$$

Substituting the above into (6.3.35) and using (6.3.30) yield the update rule for b_n:

$$b_{n+1} = \begin{cases} b_n - (1 - \nu)\eta_n, & \text{if } f_n(\mathbf{x}_n) < 0, \\ b_n + \nu\eta_n, & \text{otherwise.} \end{cases} \tag{6.3.41}$$

In summary, the update rules (6.3.38), (6.3.39) and (6.3.41) yield the stochastic gradient descent solution to the one-class SVM. The use of stochastic gradient descent converts a batch algorithm into an online one. Note that ν and η_n must be chosen such that $1 - \nu\eta_n > 0$. This can be inferred from (6.3.38) by noting that all Lagrange multipliers $\alpha_i^n, i = 1, 2, \ldots, n$, must be non-negative.

6.4 Conclusion

Standard tracking algorithms like the Kalman filter do not address what should happen after tracking failure. This chapter proposed online learning of the global appearances of objects which can then be used to re-identify objects after tracking failure. The chapter addressed three important aspects of online learning of object appearances. The first one is the discriminative power of the features used to represent objects. The HSV colour space is used mainly because it is invariant to changes and shifts in light intensity. However, we note that a combination of features is often more effective but conclude that the HSV features are sufficient for our task.

The next aspect is deciding whether to use a sample for training or not. This is because incorrectly labelled or segmented samples can impact the existing

model negatively. Our choice relies heavily on the multiple hypothesis tracker. If the score of a track exceeds zero the sample is used for training. This essentially assumes that Kalman filtering yields the correct object bounding box. The last aspect is the online machine learning algorithm. An online single class support vector machine is used. The conversion to an online method was possible because the stochastic gradient descent is used to solve the optimization problem.

Chapter 7

System integration

The main goal of the thesis is to design a system that can detect and track multiple interacting people. We have identified four components required for the system which are object detection, filtering, data association and online learning of object appearances. The goal of this chapter is to outline how these components interact to achieve the complete system. This chapter also addresses aspects of multiple target tracking such as the handling of merging and splitting tracks.

Object detection is accomplished using background subtraction, in particular mixture of Gaussian distributions [36], [79]. The OpenCV mixture of Gaussian distributions implementation [16] is used in the thesis. This component provides unlabelled measurements and given that multiple objects are being tracked, a method is required to solve the measurement-track assignment problem. This is the data association problem and is solved using multiple hypothesis tracking (MHT) [68], [77].

Filtering iteratively predicts the state of objects and then uses measurements to update those estimates. In our case the linear Kalman filters are used mainly because they simplify the MHT equations. Filtering assumes a one-to-one relationship between the tracks and measurements, a task solved using MHT. In fact, MHT sits between the prediction and update stages of the Kalman filter. The Kalman filter used in the thesis is an extension of the one implemented in OpenCV [16]. The last component of the system is online learning of object appearances which is accomplished using one-class support vector machines (OC-SVM) as implemented in OpenCV [16]. This component is used to re-identify objects after tracking failure or when they reappear.

In integrating these components we consider the software development approach. Note that these components are to a large extend independent, therefore each one is represented with a class. A class is a data type that groups the functionalities of a category of objects and their data. Consider the Kalman

filter class as an example. The data includes the state of the object, the transition matrix and the prior error covariance matrix. The functionalities include prediction and updating of the state and the prior error covariance matrix. A variable of a class is referred to as an object.

This approach requires that additional classes be defined. The first one is the Pedestrian class which defines what is being detected and tracked. Each pedestrian is tracked using a Kalman filter and its global appearance is learnt concurrently. As a result the Kalman filter and online learning classes are data members of the Pedestrian class. The second class is the integer programming problem solver (IPPSolver) class which poses the MHT problem as an integer programming problem and then solves it. This class is implemented using the Symphony mixed integer programming problem library [66]. The last class is the single camera system (SCS) class which coordinates the interaction between the aforementioned classes.

7.1 System classes: global view

This section gives an overview of the various classes and their interactions. We start with the class diagrams which list the name of each class, its primary function as well as the major modules and variables it requires to achieve this function. These are shown in Figures 7.1, 7.2 and 7.3. A negative sign before the modules and variables indicates that these are private to each variable of the class and may not be accessed by other variables directly. A positive sign before a module indicates that the module can be accessed by other variables directly. In each class diagram the first row gives the name of the class, and the second row summarizes the main function of the class. The third and fourth rows contain the methods and data members, respectively, of the class.

Figure 7.4 provides the global view of the class interactions. The diagram indicates that each SCS object must have one and only one of each of the Data Association and Background Subtraction objects as data members. The same holds for the relationship between the Pedestrian object, and the Kalman Filter and Online Learning objects. This makes sense because each Pedestrian represents what is being tracked as modelled by the Online Learning object, as well as the track which is obtained using the Kalman Filter object. Therefore, if the Pedestrian object exists then the corresponding Kalman Filter and Online Learning objects must also exist. The Data Association object contains a list of Pedestrian objects. As a result the Data Association object may exist even when there are no Pedestrian objects. This type of relationship is referred to as composition and is represented with a hollow diamond near the Data Association class and a star near the Pedestrian class.

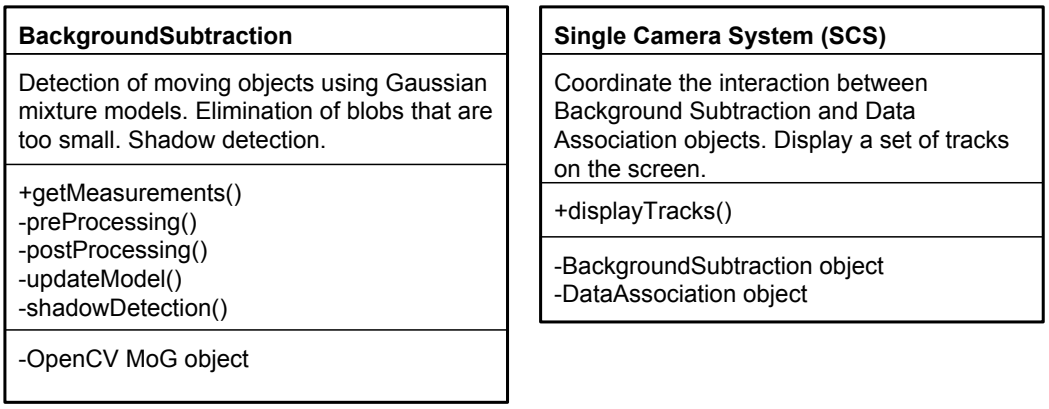


Figure 7.1: Background Subtraction and Single Camera System class diagrams.

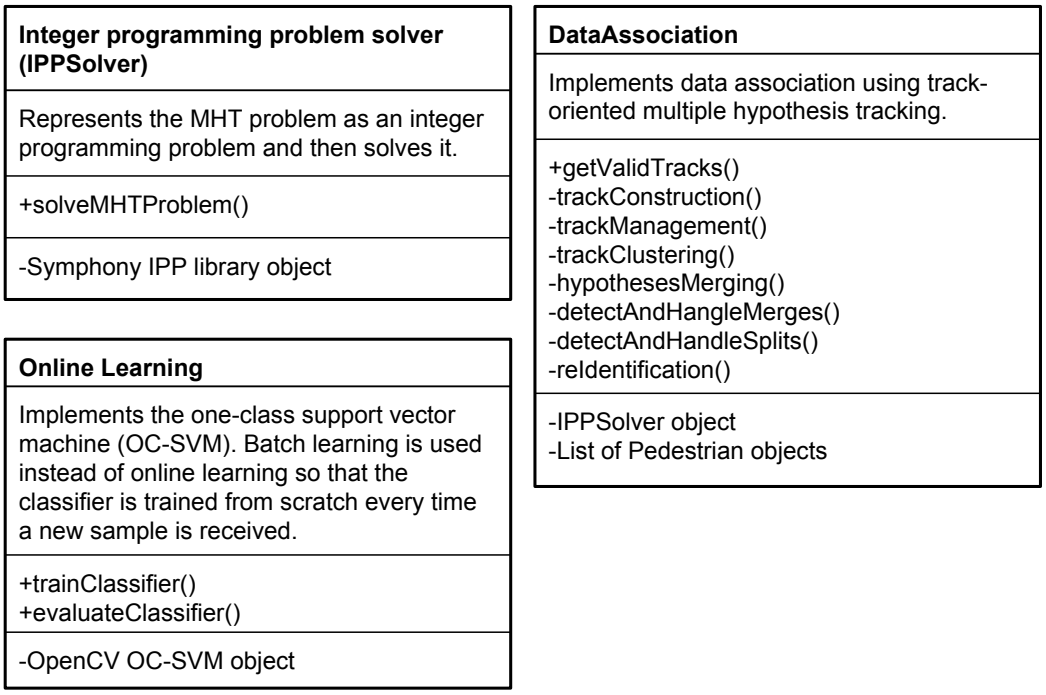


Figure 7.2: IPPSolver, Online Learning and Data Association class diagrams.

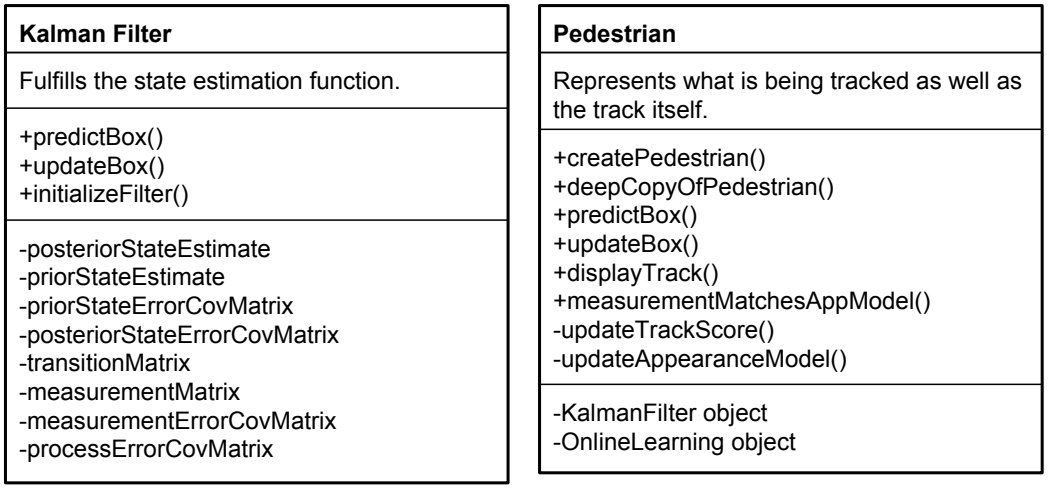


Figure 7.3: Kalman filter and Pedestrian class diagrams.

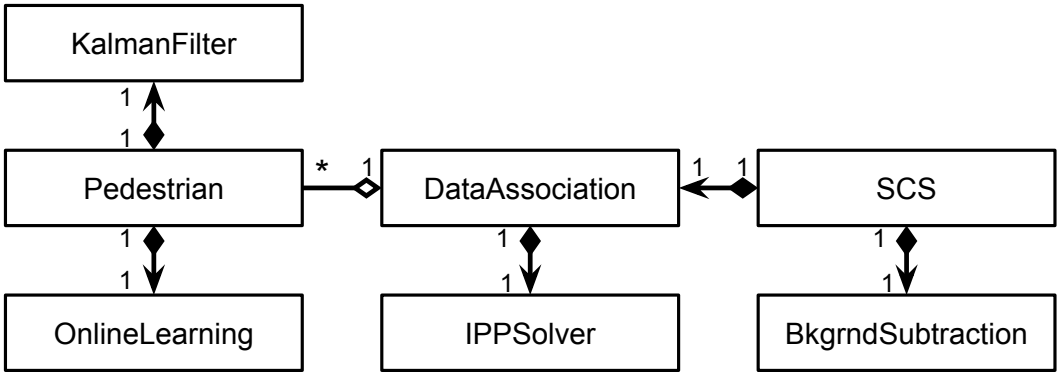


Figure 7.4: The interactions between the system classes.

7.2 Flow diagrams: some details

The interactions between the Single Camera System (SCS), Background Subtraction (BS) and Data Association (DA) classes are simple. The SCS receives a frame and then passes it to the BS object. This object returns a list of measurement bounding boxes which, along with the current frame, are then passed to the DA object. This object returns a list of valid tracks to the SCS object which are then visualized. Figure 7.5 provides the graphical representation of this process. Note that the class that is responsible for a particular function is highlighted in bold letters in the corresponding flow diagram block.

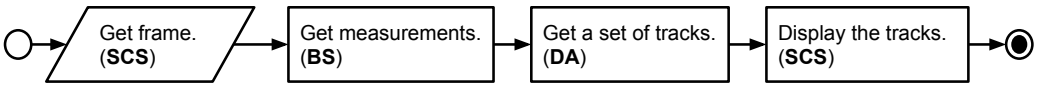


Figure 7.5: The high level functions of the detection and tracking system.

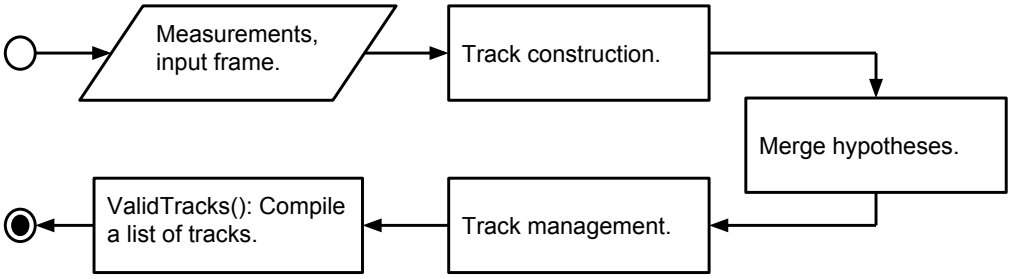


Figure 7.6: Design of the main data association function: getValidTracks().

Most of the processing is done through the DA class and we provide additional details on its implementation. The DA class diagram in Figure 7.2 indicates that only the function **getValidTrack()**, preceded by a positive sign, may be used by clients to interact with a variable of this class. It is the main function and Figure 7.6 shows its flow diagram.

The track construction module evokes functions that detect merges and splits. Most importantly, it implements the multiple hypothesis tracking algorithm. Also, it uses every measurement to try to re-detect tracks. See Figure 7.7 for the graphical representation and the order in which these functions are performed. Note the use of the notation **Pedestrian:KalmanFilter** in the block responsible for the construction of the track-measurement association matrix. In this case it indicates that some aspect of the function is performed using a Kalman filter object which is tied to a Pedestrian object. This also emphasizes that the DA object which performs track construction does not have a direct relationship with the Kalman filter as shown in Figure 7.4.

The class that has not featured in our diagrams is the IPPSolver class. It is involved in compiling a list of tracks that must be returned to the SCS object which then displays them on the screen. Figure 7.8 shows where the IPPSolver class fits in this process. Note that it is involved only when all the frames in an MHT window have been processed. Otherwise validated tracks with the highest scores, and therefore most likely tracks, are chosen and returned to the SCS.

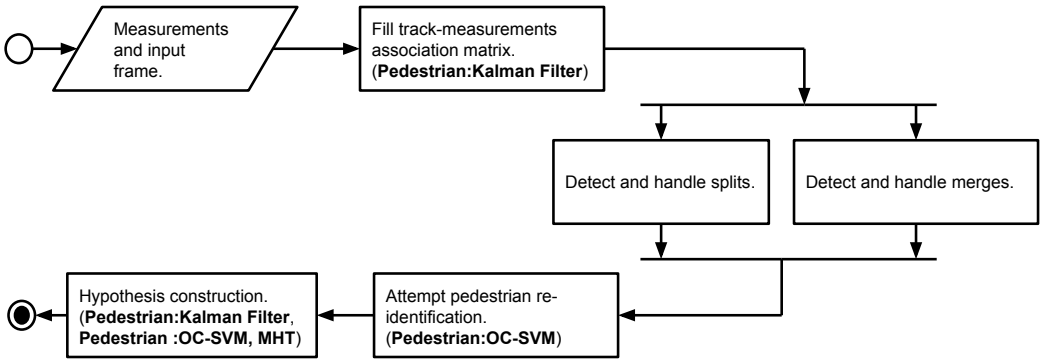


Figure 7.7: Some details into the track construction process.

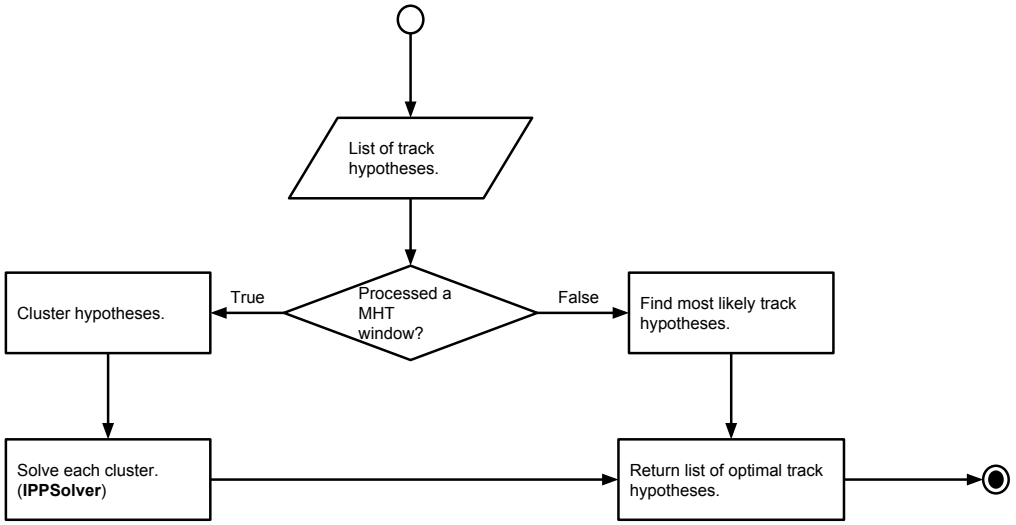


Figure 7.8: The processes involved in compiling a list of valid tracks.

7.3 Implementation details

This section considers topics that do not fit into any of the preceding chapters but form part of the system. An example of this is detecting and handling merging and splitting events which are achieved using measurements from background subtraction, predictions from the Kalman filters, and the learnt appearance models. The section also considers topics that are directly related to a previous chapter but their exact implementation details were deemed to be irrelevant at that point. The case in point is the choice of parameters, in particular ϵ in evaluating the quality of tracks in Chapter 4.

7.3.1 Merging and splitting tracks

Multiple targets in the environment are likely to interact. Given the use of background subtraction to provide measurements, these interaction will result in merging and splitting of blobs. The multiple hypothesis tracking (MHT) algorithm can handle short-lived interactions mainly because of the use of the validation region which is used to eliminate unlikely track-measurement associations. When two objects merge, the bounding box of the resultant measurement will be too large to associate with either one of the merging tracks.

In the case of splitting tracks, it is clear that the measurement bounding boxes of splitting objects will eventually become much smaller than the predicted bounding box of the merged track as the targets walk away from each other. Short-lived in this case means that objects should merge and then split before their tracks are deleted because they have not been detected in a given number of consecutive frames. This will not always be the case. As a result we explicitly detect and handle the merges and splits in this thesis. Once splits are detected, the splitting objects may be re-identified.

At this point we assume that we can detect both merges and splits. Once merging conditions are satisfied all the objects that are involved in the merger are marked to indicate that they may not be used for re-identification. This is required to deal with the weakness in the colour histograms that is used to represent the appearance of objects. Recall that the colour histogram loses shape information and the appearance of the merged object may be similar to either one of the merging objects. The merged object is automatically used to initialize a new track as per the MHT algorithm. When objects split, all tracks of objects that merged are marked to indicate that they may be used for re-identification.

Our system handles a single merger and split at a time and so it is enough to maintain a boolean variable for each track that indicates whether that track may be used for re-identification or not. In the case where multiple objects may merge and multiple mergers may take place, a more complex scheme is required such that each merger can be uniquely identified. A track involved in a merger must record the identity number of that merger. When an object splits, only tracks that carry the identity number of the splitting object may be cleared to be used for re-identification purposes. The rest of this section outlines the conditions that must be satisfied in order to conclude that either merging or splitting is taking place.

7.3.1.1 Detecting merges

This section outlines the conditions that indicate that a merger between two or more targets has taken place. A set of N tracks with predicted bounding boxes B_1, B_2, ..., B_N are involved in a merger if:

- for all $n = 1, 2, \ldots, N$, there exists a $k \in \{1, 2, \ldots, N\}$, $n \neq k$, such that $B_n \cap B_k \neq \emptyset$,
- they do not share a measurement in the given MHT window,
- There exists a measurement M such that $B_i \cap M \neq \emptyset$ for all $i = 1, 2, \ldots, N$.

The first condition merely states that if two objects merge, then their predicted bounding boxes must intersect. The second condition is used to ignore tracks from the same MHT tree because they represent the same object. It also takes into account the fact that each measurement may be used more than once. Recall that every measurement is used to initialize a new track, hence to track a new object. The last one simply states that we should be able to find a merged blob.

7.3.1.2 Detecting splits

Conditions that indicate that a tracked object O has split into two or more objects are that:

- the track of object O is not associated with a measurement,
- the predicted bounding box of object O intersects two or more measurements,
- the ratio of the total area of the measurements to the area of the predicted bounding box of the object O exceeds a preset threshold which is 0.7 in the experiments.

7.3.2 Gaussian mixture models

We made the decision to use Gaussian mixture models to detect moving objects. This, however, is only one component of the larger object detection stage. In this subsection we outline the preprocessing and post-processing functions that we use to improve the results obtained using Gaussian mixture models. We also consider the actual implementation of the condition that a pixel value matches one of the densities in the mixture.

7.3.2.1 Preprocessing

The input frame is converted from the RGB colour space to the hue-saturation-value (HSV) colour space. The global histogram equalization is performed on the value channel of the HSV space. Finally the image is smoothed using a Gaussian filter.

7.3.2.2 Post-processing

The first step of the post-processing stage is the detection and removal of shadows using the algorithm developed by Cucchiara et al. [23]. Given the input frame **I** and the background image **B** both in the HSV colour space, a pixel **F**(x, y) in the foreground mask **F** is marked as a shadow if the following conditions are satisfied:

$$\alpha \leq \frac{\mathbf{I}^V(x,y)}{\mathbf{B}^V(x,y)} \leq \beta, \tag{7.3.1}$$

$$|\mathbf{I}^H(x,y) - \mathbf{B}^H(x,y)| \leq \tau_H, \tag{7.3.2}$$

$$\mathbf{I}^S(x,y) - \mathbf{B}^S(x,y) \leq \tau_S, \tag{7.3.3}$$

where $0 < \alpha < \beta < 1$ and the superscripts indicate the channel of the HSV colour space used. The darker the shadow the smaller the value of α. The value of β is used to increase robustness to noise because the lightness of the frame cannot be similar to that of the background image.

Shadow detection and removal is followed by median filtering which fills in some of the missing foreground regions. Thereafter erosion and dilation are applied in that order with erosion removing some of the noise and dilation growing the foreground region. Median filtering and dilation can join two regions of the same object. Finally, connected components analysis is performed. Connected components that comprise at least a certain number of pixels are retained as candidate pedestrians. The smallest rectangle that encloses a candidate pedestrian is returned as a measurement.

7.3.2.3 Defining a match

Given a Gaussian density function $\mathcal{N}(\mu, \sigma^2)$, a greyscale pixel value X belongs to this density function if

$$|X - \mu| \leq T_s\sigma, \tag{7.3.4}$$

where $T_s = 2.5$ is used in Grimson and Stauffer [36]. This choice of T_s ensures that the area under the density function within the given interval is just over ninety-nine percent of the total area. In our case we are using colour images and one approach to implementing this matching condition is to assert that it

is satisfied by each channel. The approach used in the OpenCV library, and in this thesis, is to re-formulate the distance as a χ^2 variable:

$$(X-\mu)^T\Sigma^{-1}(X-\mu) \quad \leq \quad T_c, \tag{7.3.5}$$

where Σ is the covariance matrix. Here $T_c = 12.8$ is read off the χ^2 table such that the degrees of freedom, which correspond to the number of channels, are three and the area under the χ^2 density function is 0.995.

7.3.3 Multiple hypothesis tracking

The multiple hypothesis tracking problem is solved using a moving window. That is, the problem is formulated and solved using $N+1$ frames $t, t+1, \ldots, t+N$, and then for frames $t+D, t+D+1, \ldots, t+D+N$, and so on. Here $D < N$ is the number of frames the window is moved.

We now consider the task of quantifying the quality of a track. Only one term of the track quality score can contribute positively to the score given a measurement. We rewrite the formula here for convenience:

$$\begin{aligned} L_i = &\ln \frac{\lambda_0\lambda_s}{\lambda_N(\lambda_s+\frac{1}{\tau_0})} + \sum_{k=2}^{m_i} \ln \frac{\lambda_s e^{-(\lambda_s+1/\tau_0)\,t_k}}{\lambda_N|2\pi(\epsilon_k + t_k\,\mathbf{S}_{ik})|^{\frac{1}{2}}} \\ &- \frac{1}{2}\sum_{k=2}^{m_i}(\mathbf{z}_{ik}-\mathbf{H}_k\hat{\mathbf{x}}_{ik})'(\epsilon_k+t_k\mathbf{S}_{ik})^{-1}(\mathbf{z}_{ik}-\mathbf{H}_k\hat{\mathbf{x}}_{ik}) \\ &+ \ln\left[\frac{\lambda_s e^{-(\lambda_s+1/\tau_0)(T-v_i)}+\frac{1}{\tau_0}}{\lambda_s+\frac{1}{\tau_0}}\right]. \end{aligned} \tag{7.3.6}$$

The only term that can contribute positively to the score is the second term of the formula. To ensure that this happens, the parameter ϵ_k must be chosen such that:

$$f(\epsilon_k) = \left[\ln \frac{\lambda_s e^{-(\lambda_s+1/\tau_0)t_k}}{\lambda_N|2\pi(\epsilon_k+t_k\mathbf{S}_{ik})|^{\frac{1}{2}}}\right]_{t_k=0} \quad > \quad 0. \tag{7.3.7}$$

A simplifying assumption is made that $\epsilon_k = \epsilon_0\mathbf{I}$ where $\mathbf{I}$ is an identity matrix of the same dimension as $\mathbf{S}_k$ and $\epsilon_0 \in \mathbb{R}^+$. Moreover, $\varphi \in \mathbb{R}^+$ is chosen such that $f(\epsilon_k) = \varphi > 0$. The choice of φ is not very important because its impact is offset by the third term in the quality score formula. In the implementation $\varphi = 1$ is used. At this point we can solve for ϵ_0:

$$\epsilon_0 = \frac{1}{2\pi}\sqrt{\frac{\lambda_s}{\lambda_N}e^{-\varphi}}. \tag{7.3.8}$$

7.4 Conclusion

This chapter integrates the background subtraction, Kalman filter, the multiple hypothesis tracking, the integer programming problem solver (IPPSolver) and the one-class support vector machines (OC-SVM) introduced in previous chapters to yield a complete system. These methods are represented as classes which facilitate modular design. The Pedestrian and the Single Camera System classes were introduced to complete the system. The Pedestrian class represents what is being tracked as well as the the track itself. The Single Camera System coordinates the interactions between the Background Subtraction and Data Association classes.

Class and flow diagrams were used to demonstrate the interactions between the various classes and functions. The class diagrams were used to summarize the services the classes can provide (functions) as well as the variables required to provide those services. Moreover, we provided the high level interactions of the classes. Flow diagrams were used to provide more details and to highlight where the IPPSolver and OC-SVM classes fit in the system.

This chapter also addressed parts of the system that do not fit into any one method or were deemed to be peripheral to a method section. Detection and handling of merging or splitting tracks do not belong to any one method section as their implementations use the Kalman filter and the OC-SVM methods. We also outlined the pre-processing and post-processing stages of the background subtraction method.

Chapter 8

Experiments

In this chapter we discuss the experiments conducted to test the ability of our system to handle various tracking scenarios, and the results. These include the re-identification of people, and tracking of two people walking side-by-side and crossing paths. We also provide examples of tracking failures in the system. One such case is the failure of the re-identification function which is attributed to the use of colour features. The datasets collected by Baltieri et al. [5], [4] and Rasid and Suandi [67] are used. These videos were chosen because they met the assumptions of the system which are that each pedestrian occupies a small fraction of the frame, the camera looks down on the pedestrians and the only moving objects are pedestrians. We tried to make optimal decisions based on sound theoretical arguments when choosing the components of the system and the results in this chapter should be viewed as more of a proof of concept.

The intersection between the estimated and the ground truth bounding boxes is used as the basis for performance measurement. In particular, we use the Jaccard similarity coefficient:

$$J(S, G) = \frac{|S \cap G|}{|S \cup G|} \tag{8.0.1}$$

where S and G are the system and ground truth bounding boxes, respectively. $|S|$ is the cardinality of the set S which is interpreted as the area of the bounding box. A person is said to be correctly tracked in a given frame if the Jaccard similarity coefficient (JSC) exceeds 0.65.

In order to quantify the performance of the object detection component of the system we use the number of true and false positives and negatives. The number of true positives N_{TP} is the number of instances that the system and ground truth bounding boxes for which the JSC exceeds the threshold. The number of false positives N_{FP} is the number of instances that the system indicates that an object is present when the ground truth indicates otherwise. The

number of true negatives N_{TN} is the number of frames where both the system and the ground truth indicate that there is no object. The number of false negatives N_{FN} is the number of instances where the ground truth bounding box has no matching system bounding box. These quantities are not measured at a pixel-level because we only care about how much of the object is within the bounding box that is returned by the system.

The ground truth used in this thesis was generated using the MATLAB toolbox developed by Dollàr [27]. We manually drew the bounding boxes of pedestrians in a few key frames. The toolbox then interpolated the bounding boxes in the other frames. We could then adjust individual bounding boxes that we were not satisfied with. The track length and the mean square error normalized using the track length are also used to measure the performance of the system. The latter is the normalized mean square error (NMSE).

A set of experiments is performed for the three scenarios. The first one involves finding an optimal parameter T for the background subtraction method. For a given pixel, N of the M sorted densities in the mixture with weights $w_i, i = 1, 2, \ldots, M$ represent the background if

$$\sum_{j=1}^{N} w_i \leq T$$

where T is the minimum fraction of the data that should account for the background.

In quantifying the performance of the background method we can use the receiver operating characteristic (ROC) curve or the precision-recall curve. The ROC curve plots the true positive rate (TPR) versus the false positive rate (FPR). These quantities are defined below:

$$\text{Recall} = TPR = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{8.0.2}$$

$$FPR = \frac{N_{FP}}{N_{FP} + N_{TN}} \tag{8.0.3}$$

$$\text{Precision} = \text{Pre} = \frac{N_{TP}}{N_{TP} + N_{FP}}. \tag{8.0.4}$$

The ideal point within the ROC space is $TPR = 1.0$ and $FPR = 0.0$. In the precision-recall space, the ideal point is $TPR = 1.0$ and $\text{Pre} = 1.0$. We note that false detections will result in false tracks which are handled automatically by the multiple hypothesis tracking algorithm. So, we could choose the point (TPR_1, FPR_1) over the point (TPR_2, FPR_2) as long as $TPR_1 > TPR_2$. This is the case even when $FPR_1 > FPR_2$. As a result we use the precision-recall

curve to select the optimal parameter. Finally, we perform experiments that test the ability of the system to handle various scenarios.

The following parameters are used unchanged throughout the experiments. These correspond to background subtraction, the Kalman filter, learning the appearance of object and the multiple hypothesis tracking algorithms:

1. Background subtraction:
   a) threshold on the definition of a match: $T = 12.8$ which ensures that the area under the χ^2 distribution of three degrees is 0.995.
2. Kalman filter:
   a) the small dynamic process matrix multiplier: $q = e^{-0.25}$,
   b) the measurement covariance matrix: $Q = 50 * \mathbf{I}$, $\mathbf{I} \in \mathbb{R}^{4\times4}$,
   c) the initial prior state error covariance matrix:

$$\mathbf{P}_0 = \begin{pmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 \end{pmatrix}.$$

3. Learning object appearances:
   a) Gaussian kernel parameter: $\gamma = 0.125$,
   b) adaptive upper bound on the ratio of misclassification: $\nu = \frac{1}{N}$, where N is the number of training examples.
4. χ^2 threshold:
   a) $\lambda = 1.650$ which implies a cutoff probability of 0.80. A χ^2 value less than the threshold implies that the deviation of the measurement and the predicted state can be explained by chance alone as opposed to incorrectly modelled dynamics, for example.
5. MHT parameters:
   a) $\lambda_0 = 25 \times 10^{-9}$, $\lambda_N = 35 \times 10^{-9}$,
   b) $\lambda_s = 0.8$, $\tau_0 = 50$.

Any changes to the parameters when performing an experiment will be highlighted before the results of that experiment are given. The values of the parameters λ_N and λ_0 might seem to be too small but consider λ_N which is the number of false detections per unit time per unit area, that is:

$$\begin{aligned} \lambda_N &= \frac{N}{(704-30)\times(564-70)\times NF} \\ &= \frac{1}{(704-30)\times(564-80)\times NF} \qquad (8.0.5) \\ \Rightarrow \quad NF &= 85.4654 \simeq 85 \qquad (8.0.6) \end{aligned}$$

where 30 and 70 are the minimum width and height of a pedestrian, respectively. The frame is of size 564×704. Here we fix the number of false detections (N) and vary the unit time (NF) which is the number of frames. In this case we expect one false detection per unit area every 85 frames.

8.1 Scenario 1: two people walking together

The first scenario tests the ability of the system to track two people walking side-by-side. The first experiment that is performed is to determine the optimal parameter T for the background subtraction method identified in the introduction. Figure 8.1 shows the changes in the TPR, precision and distance to the ideal precision-recall point as the parameter T changes. The value $T = 0.65$ was used in the experiments because it resulted in the highest true positive rate. In the second experiment, the system is used to track the two people as they walk across the scene. In this case re-identification is not used and the tracks are not fragmented as shown in Figure 8.2.

Finally, Figure 8.3 shows the tracks generated by the system in black as well as the respective ground truth tracks in red. Pedestrian 1 was correctly tracked for 78 of the 79 frames he was in the scene with an NMSE of 3.79 pixels. Pedestrian 9 was correctly tracked for 75 of the 77 frames he was in the scene with NMSE of 4.55 pixels.

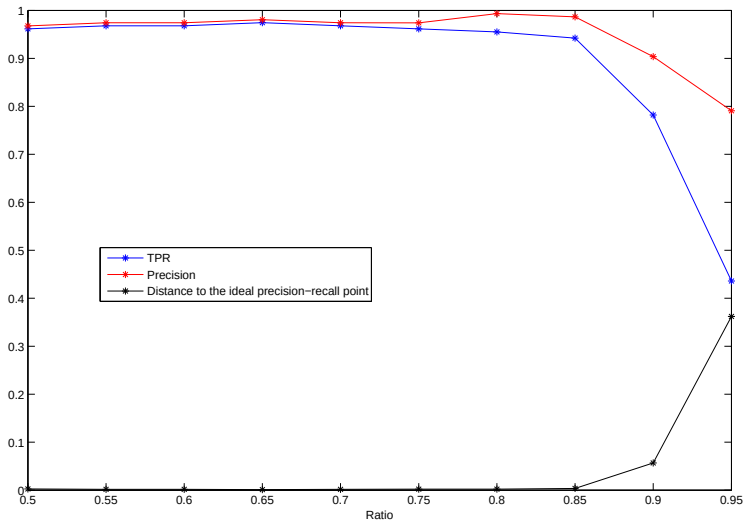


Figure 8.1: Changes in TPR, precision and distance to the ideal point as the parameter T changes.

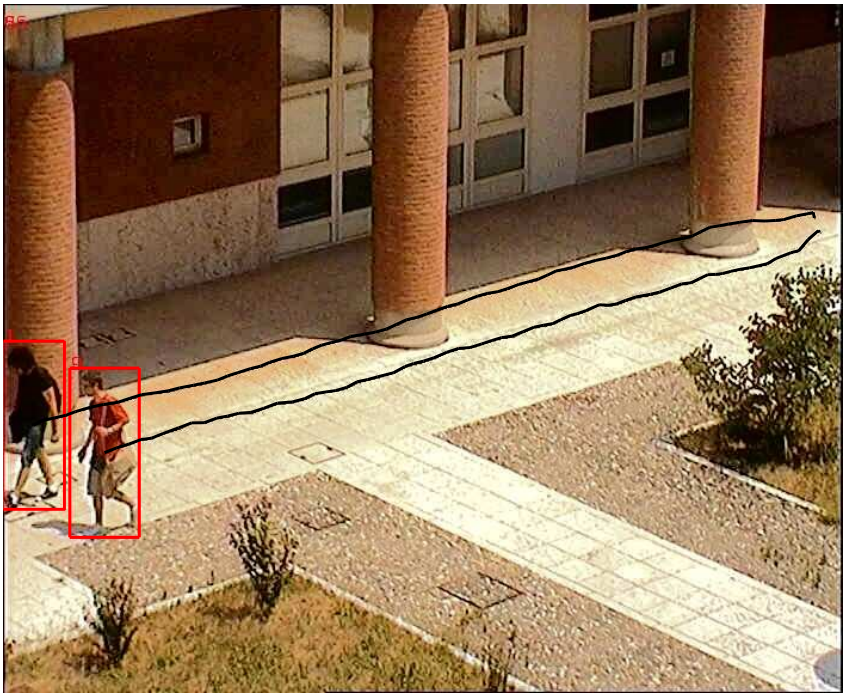


Figure 8.2: Tracks of two people walking side-by-side generated by the system.

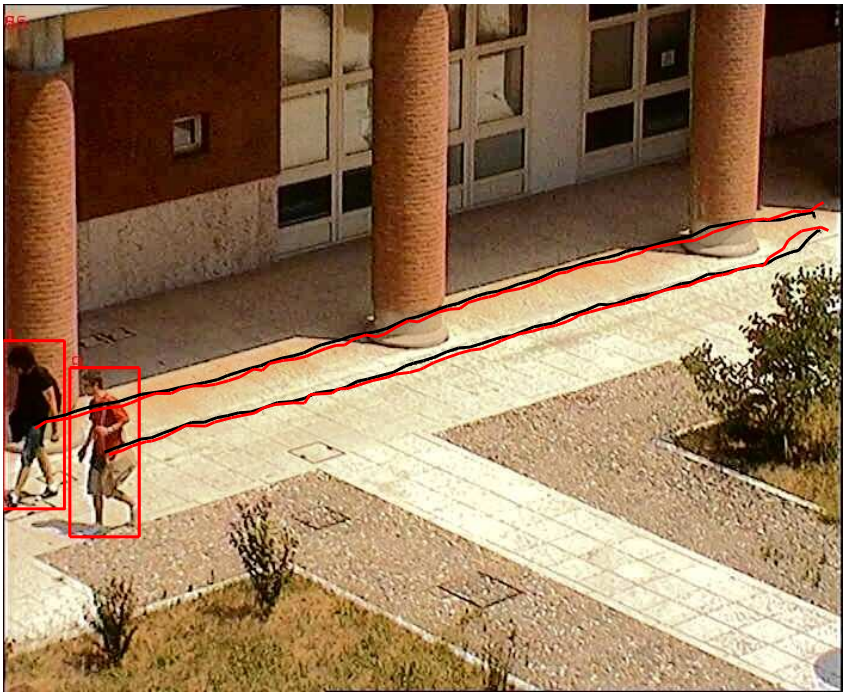


Figure 8.3: System generated tracks in black along with the ground truth tracks in red.

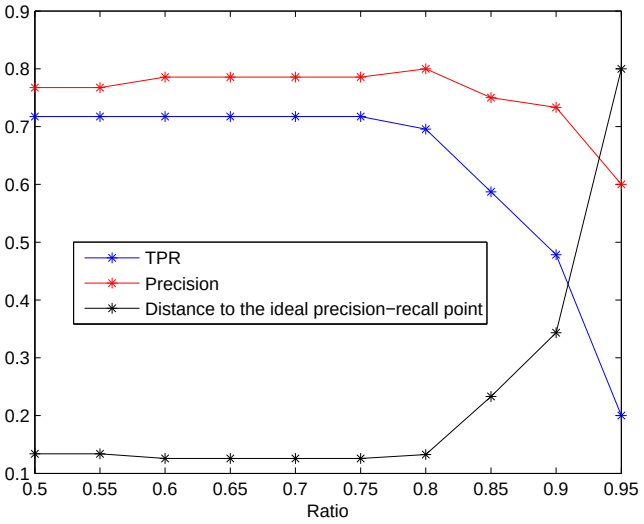


Figure 8.4: Changes in TPR, precision and distance to the ideal point as the parameter T changes.

Figure 8.5: Testing the re-identification abilities of the system.

8.2 Scenario 2: re-identification capability

In this section we test the re-identification ability of the system. The first experiment that we perform is to determine the optimal value of the background subtraction parameter T as explained in the introduction. Figure 8.4 shows the changes in the TPR, precision and distance to the ideal precision-recall point as the parameter T changes. The values of T between 0.5 and 0.75 yield the same value for the true positive rate. However, the values between 0.6 and 0.75 yield better precision rates. As a result, $T = 0.65$ was used in the experiments.

In the second experiment, a tracked pedestrian is completely occluded by a pillar and the system manages to re-identify him when he reappears and extend his tracks. The results are shown in Figure 8.5. The pedestrian then disappears behind the third pillar and is not re-identified when he reappears. This is due to the dark shadow in the region which changes the appearance of the pedestrian. This highlights the reality of re-identification which is that an object can only be re-identified if its current appearance is similar to one of the previously seen appearances. Finally, Figure 8.6 shows the track generated by the system in green as well as the ground truth track in red. The system successfully tracked the pedestrian in 38 of the 48 frames with an NMSE of 9.72 pixels which is slightly higher than the NMSEs in the first scenario. This is due to the jump in the track to which we referred above.

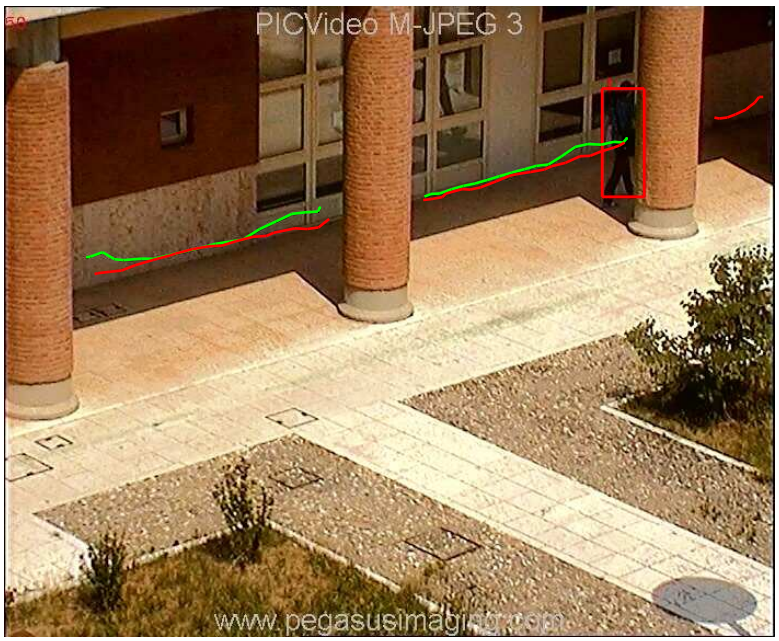


Figure 8.6: The track obtained using the system in green along with the ground truth in red.

8.3 Scenario 3: two people crossing paths

In this section we test the ability of the system to track two pedestrians whose paths cross. The first experiment is to determine the optimal value for the background subtraction parameter T. Figure 8.7 shows the changes in the TPR, precision and the distance to the ideal precision-recall point as the parameter T changes. The graph indicates that values of T between 0.55 and 0.70 yield the same results for all measures. A value of $T = 0.65$ is used in the experiments to match those used in previous experiments.

The second experiment demonstrates how the system handles merging and splitting tracks. Figure 8.8 shows the tracks before merging takes place. The pedestrians are tracked individually for three more frames once merging has taken place as shown in Figures 8.9, 8.10 and 8.11. This is the case even though the pedestrians are not detected as separated entities in those frames. Recall that a track that is not associated with a measurement in a given number of consecutive frames, in this case three, is assumed to have left the scene and is deleted. This is used to manage the number of hypotheses in the MHT tracking algorithm.

Figure 8.12 shows that the group is being tracked as a single object. This track was created when the two pedestrians first merged but had to be supported by additional measurements to ensure that it is not a false track. Figure 8.13

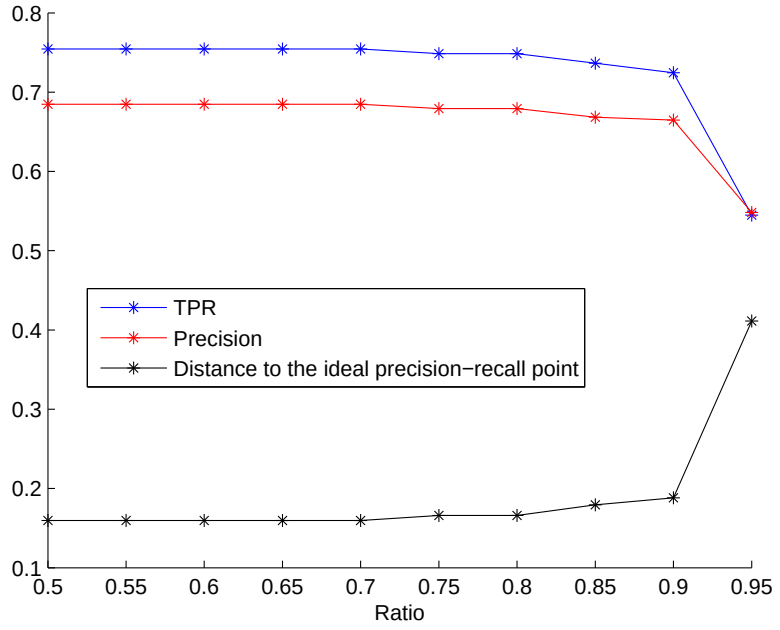


Figure 8.7: Changes in TPR, precision and distance to the ideal point as the parameter T changes.

shows that the group is still tracked as a single object even though the pedestrians have finally separated. It will only be deleted after three consecutive missed detections. At this point the splitting event has been detected and the measurements are used for re-identification. The pedestrians are successfully re-identified as shown in Figure 8.14.

Finally, Figure 8.15 shows the system generated tracks along with the associated ground truth tracks in red and magenta. Pedestrian 4, facing to the left, was correctly tracked for 51 of the 53 frames he was in the scene with an NMSE of 8.89 pixels. Pedestrian 27, facing to the right, was correctly tracked for 30 of the 31 frames he was in the scene with an NMSE of 6.91 pixels. Note that the system tracks overshoot the ground truth tracks substantially when going into the merger. This is because tracks are retained for at most three consecutive frames even though the tracked objects are not detected in those frames as shown in Figures 8.10 and 8.11. This can be avoided by not recording the bounding box of objects if it is determined that they are involved in a merger. This should not be difficult because tracks that have merged cannot be associated with a measurement.

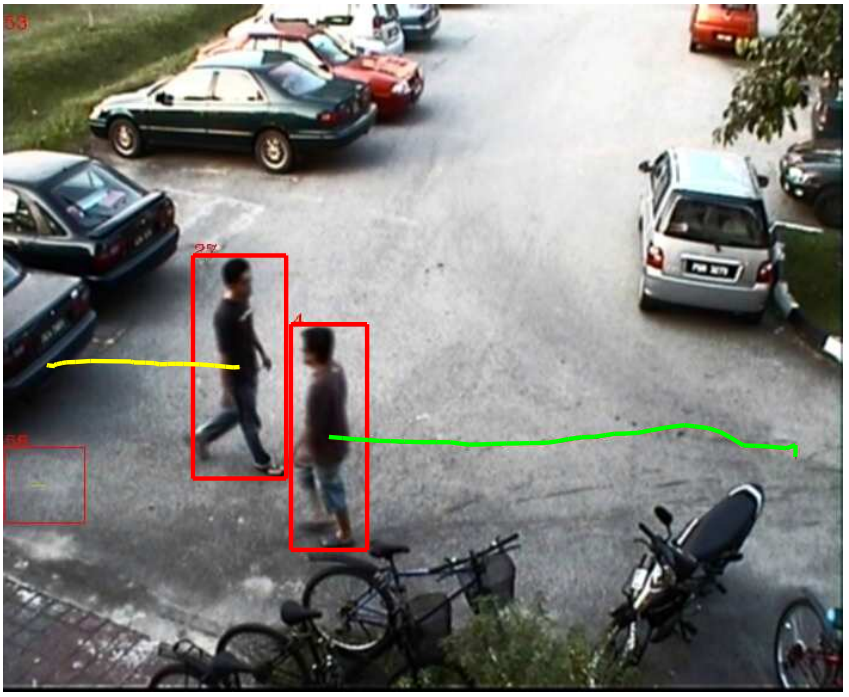


Figure 8.8: System generated tracks in green and yellow for pedestrians 4 and 27, respectively, before merging takes place.

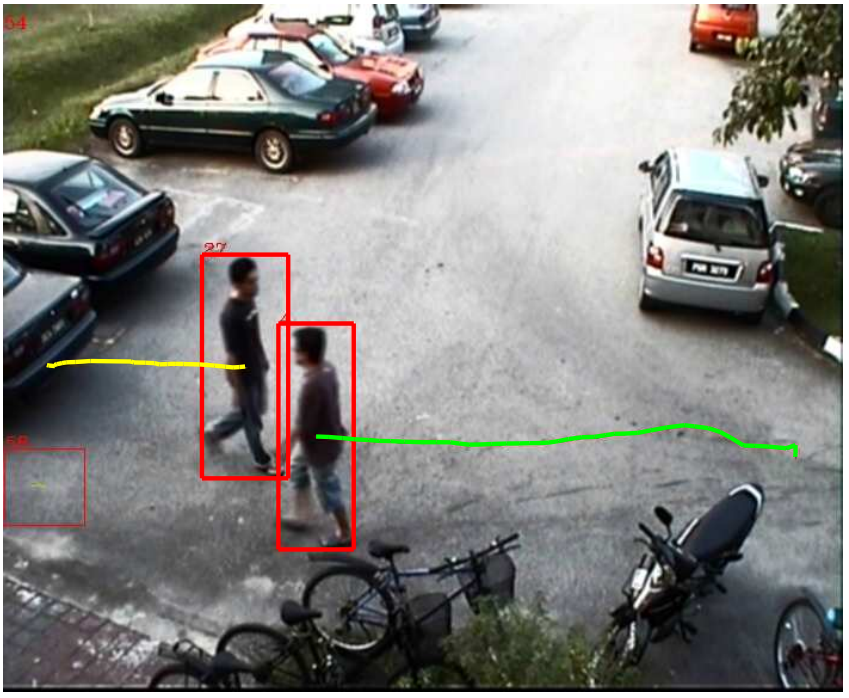


Figure 8.9: System generated tracks in green and yellow for pedestrians 4 and 27, respectively, in the first frame of the merging event.

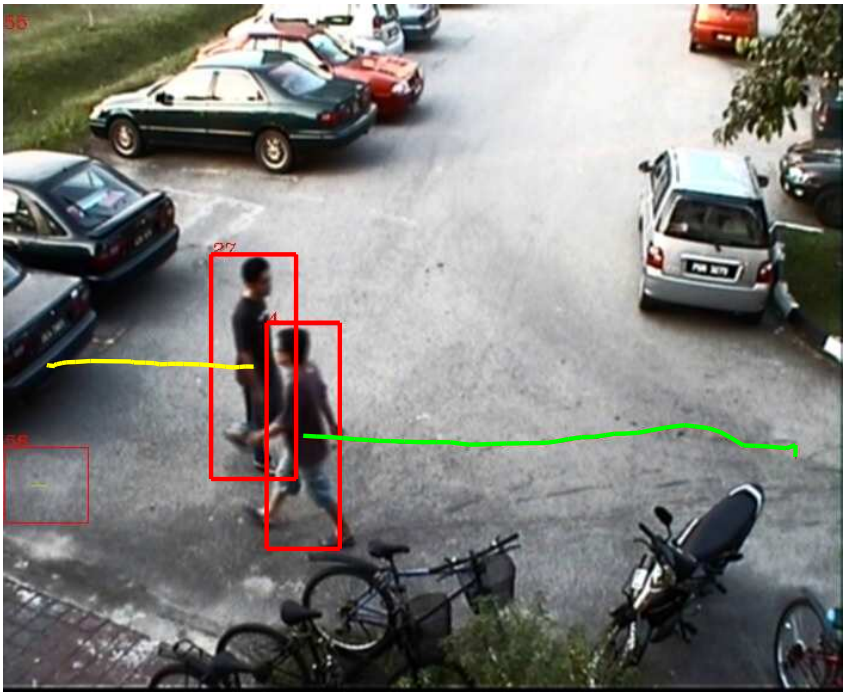


Figure 8.10: System generated tracks in green and yellow for pedestrians 4 and 27, respectively, in the second frame of the merging event.

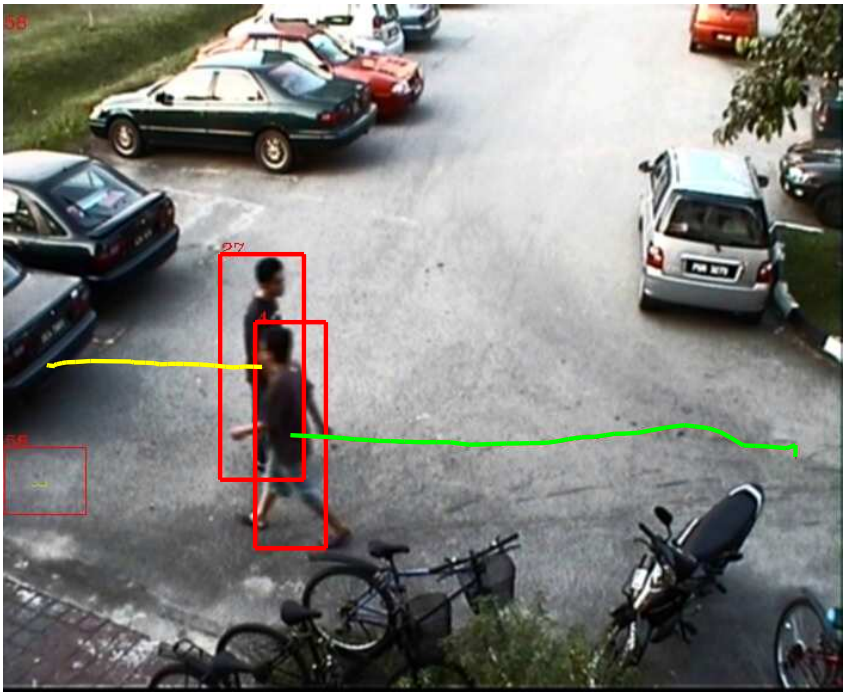


Figure 8.11: System generated tracks in green and yellow for pedestrians 4 and 27, respectively, in the third frame of the merging event.

Figure 8.12: The track of the merged object in green is finally displayed four frames after the merging event because it has received enough supporting measurements.

Figure 8.13: The pedestrians have split but are still tracked as single object.

Figure 8.14: Both tracks are successfully re-identified after the splitting event.

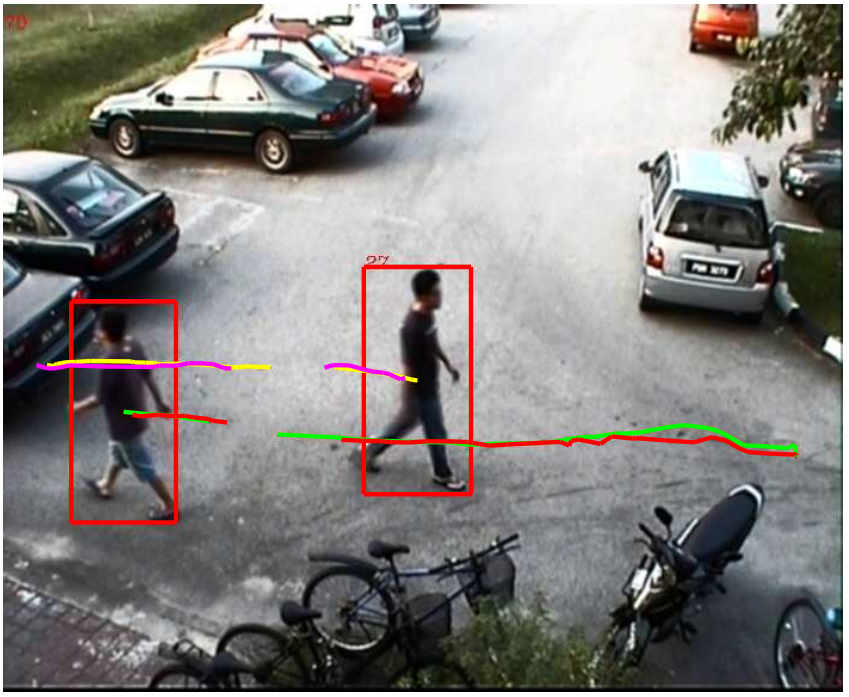


Figure 8.15: System generated tracks and the corresponding ground truth tracks in red and magenta.

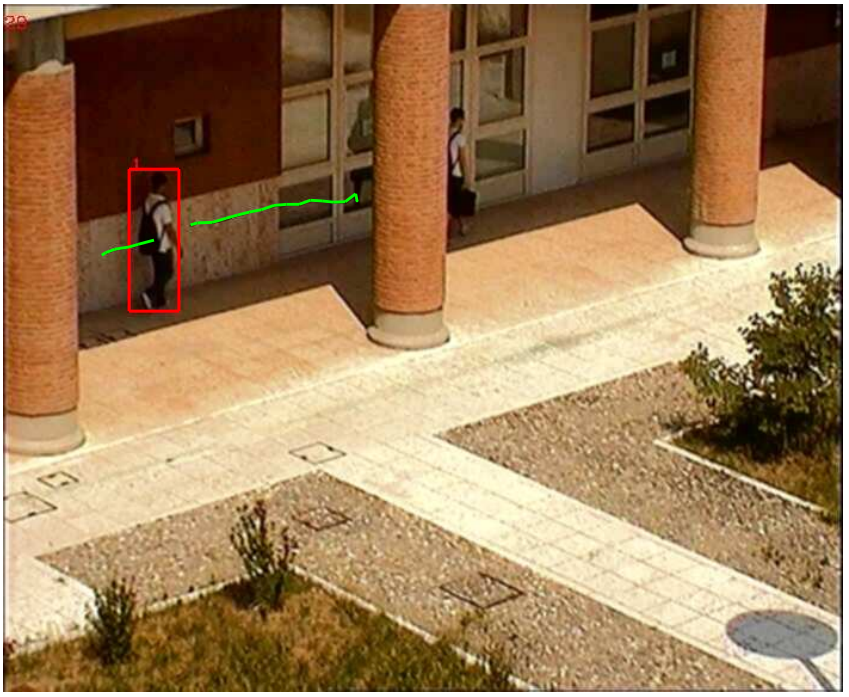


Figure 8.16: A case where the system fails because the two pedestrian have similar appearances.

8.4 System failure

This section demonstrates a scenario where the system fails because of the similar appearance of pedestrians. In Figure 8.16 the pedestrian on the right, the one leading, was occluded by the middle pillar when the pedestrian on the left, the one lagging, entered the scene. This meant that the track of the leading pedestrian could be used for re-identification. As a result, the lagging pedestrian was mistaken for the leading pedestrian because their appearances are similar. This can be resolved by using a different or additional feature that preserves the shape information of the pedestrian such as the histogram of orientation gradients. In this case, re-identification can also be improved by using a rule that humans can only move with finite speed and thus cannot jump in position from one frame to the next.

8.5 Sensitivity to MHT parameters

In this section we investigate the sensitivity of the results to the MHT parameters which are λ_s, λ_0 and λ_N. For convenience, we rewrite the track score

formula:

$$\begin{aligned} L_i = & \ln \frac{\lambda_0 \lambda_s}{\lambda_N(\lambda_s + \frac{1}{\tau_0})} + \sum_{k=2}^{m_i} \ln \frac{\lambda_s e^{-(\lambda_s + 1/\tau_0)\, t_k}}{\lambda_N |2\pi(\epsilon_k + t_k\, \mathbf{S}_{ik})|^{\frac{1}{2}}} \\ & - \frac{1}{2} \sum_{k=2}^{m_i} (\mathbf{z}_{ik} - \mathbf{H}_k \hat{\mathbf{x}}_{ik})' (\epsilon_k + t_k \mathbf{S}_{ik})^{-1} (\mathbf{z}_{ik} - \mathbf{H}_k \hat{\mathbf{x}}_{ik}) \\ & + \ln \left[\frac{\lambda_s e^{-(\lambda_s + 1/\tau_0)(T - v_i)} + \frac{1}{\tau_0}}{\lambda_s + \frac{1}{\tau_0}} \right]. \end{aligned} \tag{8.5.1}$$

The re-identification capability of the system is turned off. A video with a single person following a curved path is used in this set of experiments.

8.5.1 Parameter: λ_s

This parameter is involved in three of the four terms of the track score formula (8.5.1). In the first term, which is the entrance score, its impact should be minimal as long as that term evaluates to a negative value. Its impact on the other two terms is not clear as a large value for the parameter incurs a large discount due to the negative exponential. This parameter is also involved in the cost of terminating a track:

$$B = -\ln(1 + \lambda_s \tau_0). \tag{8.5.2}$$

A larger value of λ_s should mean a higher cost of terminating a track. However, this term kicks in when there are missed detections which are the unclear cases stated above.

The parameter is varied from 0.05 to 0.95 in steps of 0.05 while the other parameters are fixed ($\lambda_0 = 25 \times 10^{-9}$, $\lambda_N = 35 \times 10^{-9}$, $\tau_0 = 100$). The results indicate that the track either exists or it does not exist. Values of the parameter between 0.45 and 0.95 yield exactly the same results when we consider the track length (47) and the mean square error (4.8154) which is normalized using the length of the track. From 0.15 to 0.4 the track length remains the same but the mean square increases slightly to 4.9678. The system is not able to pick up the track for $\lambda_s = 0.05$ and $\lambda_s = 0.10$.

8.5.2 Parameter: λ_0

This parameter is involved in determining the entrance score of a track. The larger it is, the smaller the cost of initializing a track and the easier it is to initialize new tracks. There is also the constraint

$$\lambda_0 < \lambda_N (1 + \frac{1}{\lambda_s \tau_0}) \tag{8.5.3}$$

Constraint	Track ID	Track length	MSE	Time(sec)
Satisfied	2	47	4.7954	75
Violated 1	5	44	4.8975	371
Violated 2	7	44	4.8803	6326

Table 8.1: Results when constraint (8.5.3) is satisfied vs when it is violated.

which enforces the assumption that a single measurement is not enough to conclude that a track exists. A new track is given a negative entrance score and subsequent measurements associated with the track should drive the score to exceed zero, hence confirming that the track is valid. We perform an experiment to determine what happens if this constraint is violated.

The first experiment we perform is to determine what happens when the constraint mentioned above is violated. In the first experiment the parameter values $\lambda_0 = 25 \times 10^{-8}, \lambda_N = 35 \times 10^{-9}, \lambda_s = 0.65$ and $\tau_0 = 100$ are used. In the second experiment $\lambda_0 = 95 \times 10^{-8}$ is used. The only notable difference is that it takes longer to process a video sequence. This does indicate how easy it is for tracks to be confirmed. A large number of confirmed tracks implies a high number of variables when the data association problem is converted to an integer programming problem. This implies a higher computational cost. Table 8.1 summarizes the results.

Making λ_0 very small exposed a weakness in our system. A very small value of this parameter means a very high cost must be paid to initialize a track. Hence, it is difficult for tracks to be confirmed. Consider the example in Figure 8.17 with two tracks that have not been confirmed. Assume that three frames later all the track hypotheses are still not confirmed. The expanded T2 tree resembles the left branch of the expanded T1 tree shown in Figure 8.18. This leads to an explosion of track hypotheses. It is alleviated when most of them are confirmed and the data association problem is solved but then

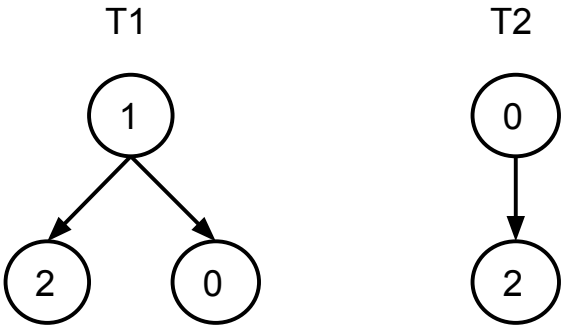


Figure 8.17: An example where one tree can replicate a subset of another tree.

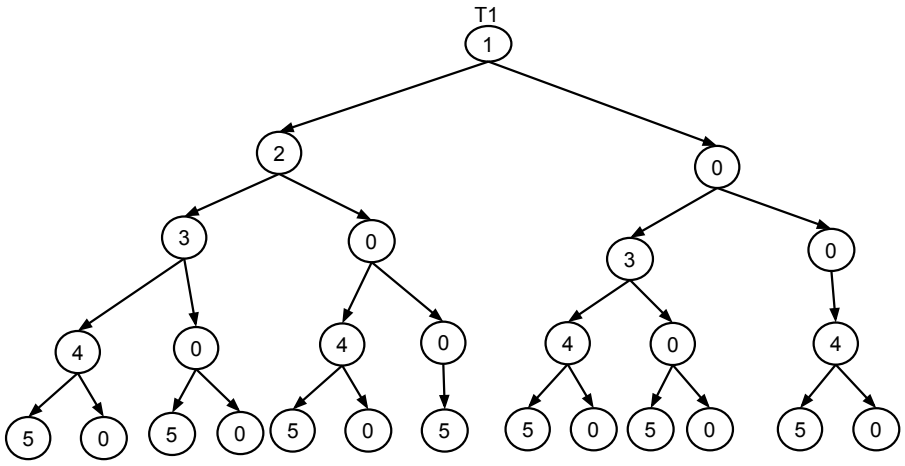


Figure 8.18: An example where one tree can replicate a subset of another tree.

it starts all over again. Thus, the computational and memory costs increase but the final track is of the same quality when we consider the track length and the normalized mean square error. This problem may be resolved by using measurements that have not been associated with tracks to initialize new tracks.

8.5.3 Parameter: λ_N

This parameter is involved in two terms in the track score formula and the larger its value is, the higher the expected number of false targets. The first term is the track entrance score where a larger value means a higher cost of initializing tracks. The second term is the only term that can contribute positively to the track score. In this case the impact of the parameter should be greater when there are missed detections.

Table 8.2 shows the results obtained when λ_N is varied. NF is the number of frames, NMSE is the normalized mean square error and TTCT is the time to confirm a track. In the TTCT column, the number in brackets is the frame number in which the track was first detected. As the expected number of false detections increases the quality of the track remains fairly stable as measured using the NMSE and the track length. Also, note that it takes longer for tracks to be confirmed. This is to be expected as increasing λ_N should increase the cost of initializing a track. The track cannot be confirmed when the unit time drops below 14 frames.

$\lambda_N(10^{-9})$	nF	NMSE	Track Length	TTCT	ID
35	85	4.8463	47	14(10)	2
75	40	4.9128	47	16(10)	2
195	15	4.8429	47	19(10)	2
213	14	4.6171	44	23(13)	5

Table 8.2: The impact of the λ_N on the quality of the track. NF is number of frames, NMSE is the normalized mean square error and TTCT is the time to confirm a track.

8.6 Analysis and conclusion

In this chapter a number of experiments were performed and analyzed to test the different aspects of the system. Four data sets were used. The first data set tested the ability of the system to track two people walking side by side. The second and third sets tested the re-identification capabilities of the system. In particular, the first of these two tested the ability of the system to recover from occlusions. The second one tested the ability of the system to handle merge and split events. The last data set was used to analyze the sensitivity of the system to the MHT parameters.

The experiments also demonstrated two cases where the system fails which are directly related to the colour features used. In the first case, the system failed to re-identify a pedestrian when they re-appeared in the scene. This was attributed to the darker shadow in the region the pedestrian walked into when he re-appeared. This highlights the fact that objects may only be re-identified if their current appearance is similar to one of their historical appearances. In the second case a pedestrian was mistaken for another one because the two of them had similar colour appearances. This highlights the weakness of colour histogram features which is that they lose the shape information. This can be overcome by using a combination of features.

Apart from when determining the sensitivity of the system to the MHT parameters, the same set of parameters was used for all the experiments which suggests that the system is robust. The impact of the MHT parameters on the track score formula is supported by the experiments. The system does not seem to be sensitive to λ_s because results of the same quality are received for a large range of values. However, λ_s can be so small that the cost of terminating a track is so small that tracks cannot be confirmed.

Increasing the value of λ_0 so that the constraint $\lambda_0 < \lambda_N(\lambda_s + \frac{1}{\tau_0})$ is violated still yields good results when we consider the track length and the normalized mean square error. However, the time required to process a video sequence increases the more the constraint is violated. This makes sense because the larger the value of the parameter, the smaller the entrance score and the easier

it is for a track to be confirmed. This means that there is a large number of confirmed tracks which translates to a high number of variables when the problem is converted to an integer programming problem. Making λ_0 very small also increases the computational cost of the algorithm. However, it does not affect the quality of the final track when the track length and the normalized mean square error are used as quantitative measures.

Increasing the value of λ_N increases the cost of initializing a track. However, it does not have the same impact on the system (increased computational cost) as decreasing the value of λ_0, which also increases the cost of initializing tracks. This is because λ_N is also involved when there are missed detections and serves to accelerate the decay in the track score. Increasing the value of this parameter beyond a certain point results in tracks not being confirmed. This is understandable because the parameter has a negative impact on the only term that can contribute positively to the score.

Chapter 9

Conclusion

The goal of this thesis was to design and implement a system that can detect and track multiple interacting pedestrians over a prolonged period of time. Such a system is necessitated by the challenges due to the growing number of closed circuit cameras in our cities, malls and airports. These networks of cameras, though cheaper to install, are expensive to monitor. They also raise issues of targeted surveillance. Automation of surveillance through the application of computer vision algorithms can solve some of these challenges. This is automated surveillance which involves detecting, classifying and tracking objects, and understanding and describing their behaviours.

We focused on pedestrian detection and tracking. The components of the system that were identified are object detection, motion estimation using filters, data association, and learning of appearances of pedestrians. Object detection is an important first stage of many computer vision applications because it focuses the attention of subsequent stages on dynamic regions of the scene. A number of object detection algorithms which include background subtraction, optical flow and learning methods were identified and reviewed. The decision was taken to use background subtraction which relies on the motion of objects to detect them. It is comparatively computationally inexpensive. In particular, Gaussian mixture models are used to detect moving objects.

Long-term tracking is achieved using data association, filtering and online learning of appearances of objects of interest. Data association is required because measurements from the object detection stage are not labelled and could be from false targets. In our system data association is realized using multiple hypothesis tracking (MHT) which uses information from a number of frames to solve the measurement origination problem. Note that incorrect data association decisions can lead to fragmented or lost tracks. MHT is used because it implicitly provides track initiation and management facilities. Moreover, data association enables the use of filtering algorithms which essentially assume a one-to-one relationship between tracks and measurements. Our system uses

the Kalman filter for this purpose mainly because it makes the MHT equations easier to evaluate.

Standard tracking algorithms like the Kalman filter can fail. Also, tracked objects may be temporarily occluded or leave the area under surveillance and then return. One approach is to initiate new tracks when any of these issues occur. In our case, however, the one-class support vector machine (OC-SVM) is used to learn the appearance of tracked objects and to re-identify those objects. Thus, tracks may be picked up and extended.

Last is object representation which comprises object shape and appearance. Object shape is relevant to the Kalman filter and is modelled using a rectangle. Object appearance modelling is required for online learning of appearances of objects. The hue saturation value (HSV) colour histograms were used to model the appearance. Colour features can be discriminative. However, they lose the shape information and this emphasizes that a combination of appearance models is required to handle real-world conditions.

These components are represented as classes in the software development sense. Moreover, they must interact in order to produce a complete system. This required the introduction of the Pedestrian, Integer Program Problem Solver (IPPSolver) and Single Camera System classes. The Pedestrian class represents what is being tracked as well as the track itself. The IPPSolver converts the MHT problem into an integer programming problem and then solves it. The Single Camera System coordinates the interactions between the various classes.

Three tracking scenarios were used to test the ability of the system. The first one demonstrated the ability of the system to track two people walking side by side which was achieved. In this case it is possible that one track steals the measurement of another track when the tracks are close to each other. This is possible when the nearest neighbour tracker is used for data association. It is also possible that the tracks drift towards each other which can happen when the joint probabilistic data association method is used measurement to track association.

The second scenario demonstrated the ability of the system to recover from occlusions. The system was able to recover from the first occlusion event but failed to recover from the second one. The failure was attributed to the darker shadow of the region the pedestrian walked into when they reappeared. This highlighted the weakness of re-identification. It shows that objects can only be re-identified if their current appearance is similar to one of the historic appearances.

The third scenario demonstrated the ability of the system to handle merging and splitting events. Merging and splitting can be handled by the MHT tracker if their duration is less than the MHT window (which is the number of frames that must be processed before the data association problem can be solved). However, we could not rely on this to happen every time and therefore these are handled explicitly. The system handles merges and splits by detecting split and then evoking re-identification. It was able track two people whose paths crossed.

In addition to the re-identification failure in the second scenario, we demonstrated another failure which is attributed to the histogram colour feature. Colour features lose the shape information of objects. While a tracked pedestrian (A) was temporarily occluded, another pedestrian (B) with a similar appearance entered the scene. Pedestrian B was mistaken for pedestrian A and assigned the track history and ID of A. This highlights that a combination of features may be required for robust re-identification.

The qualitative performance of the system is good as can be seen from the images in the experiments chapter. The performance was measured against ground truth using the Jaccard index, track length and normalized mean square error (NMSE). The Jaccard index quantifies performance at a frame level by comparing the ground truth and system bounding boxes of a given object. If it exceeds a preset threshold then the object is correctly tracked. The system track length flags fragmentation issues when compared against the ground truth track. The NMSE quantifies performance when the object is correctly tracked. All these measures indicate that the system performs well for all but the re-identification scenario. In this scenario the system failed to re-identify a pedestrian thus negatively affecting the track length.

Finally, a set of experiments were performed to determine the sensitivity of the system to the parameters of the MHT tracker. The experiments indicated that the system returned similar results for a wide range of parameter values, hence it is robust. The track length and normalized mean square error were used to measure performance. The noticeable effect of different parameter values was the computational time required to process the given video sequence and the time required to confirm tracks. Increasing λ_0 and decreasing λ_N tend to decrease the time required to confirm tracks. Increasing λ_N and and decreasing λ_0 tend to increase the computational cost. All these support the analysis of the track score function. The fact that the same set of parameters were used in all experiments but the sensitivity analysis experiments indicate that the system is robust.

There are improvements that can be made to the system. The first improvement would be to use a number of features to represent objects. Colour fea-

tures are discriminative but lose the shape information. This was highlighted in one of the experiments when a pedestrian was mistaken for another pedestrian with a similar colour histogram. Complementing the colour histogram with a histogram of oriented gradients can solve the mistaken identity problem.

Re-identification can also be improved by using rules. One such is that humans can only move with finite speed and therefore cannot jump in position between frames. This could have solved the mistaken identity problem as well.

Another improvement would be to extend the system to handle multiple merges and splits. Currently, the system can handle a single merge and split event at a time. This is because once a merge event is detected all the track hypotheses are locked so that they may not be used for re-identification. All the track hypotheses are unlocked and may be used for re-identification once a split event is detected. All the components required to handle multiple merge and split events are implemented in the current system. All that is required is the data structure to maintain the multiple merges and manage the locking and unlocking of the corresponding tracks.

A set of track hypotheses in the same tree was maintained separately when transforming the track-oriented MHT problem into an integer programming problem. This increased the memory requirements of the system but simplified the implementation. We could optimize the implementation by using a tree structure and using the current implementation for validation purposes.

List of References

[1] A. Amditis, G. Thomaidis, P. Maroudis, P. Lytrivis, and G. Karaseitanidis. *Laser Scanner Technology*, chapter : Multiple hypothesis tracking implementation, pages 199–220. InTechOpen, 2012.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Transactions on Signal Processing*, 50(2):174–188, 2002.

[3] D. Avitzour. A maximum likelihood approach to data association. *Transactions on Aerospace and Electronic Systems*, 28(2):560–566, 1992.

[4] D. Baltieri, R. Vezzani, and R. Cucchiara. 3DPes: 3D people dataset for surveillance and forensics. In *Joint ACM Workshop on Human Gesture and Behavior Understanding*, pages 59–64, 2011.

[5] D. Baltieri, R. Vezzani, and R. Cucchiara. SARC3D: A new 3D body model for people tracking and re-identification. In *International Conference on Image Analysis and Processing*, pages 197–206, 2011.

[6] Y. Bar-Shalom, S. S. Blackman, and R. J. Fitzgerald. The dimensionless score function for multiple hypothesis tracking. *Transactions on Aerospace and Electronic Systems*, 43(1):392–400, 2007.

[7] Y. Bar-Shalom and X. Li. *Multitarget-multisensor Tracking: Principles and Techniques*. YBS Publishing, 1995.

[8] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[9] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *Computer Vision and Pattern Recognition*, pages 3457–3464, 2011.

[10] D. Beymer and K. Konolige. Real-time tracking of multiple people using continuous detection. In *International Conference on Computer Vision Frame Rate Workshop*, 1999.

[11] J. Bilmes. A gentle introduction of the EM algorithm and its application to parameter estimation for GMM and HMM. Technical Report TR-97-021, University of Carlifornia, Berkeley, 1998.

[12] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.

[13] L. Bottou. Large-scale machine learning with stochastic gradient descent. Technical Report, NEC Labs America, 2009.

[14] F. Bourgeois and J. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804, 1971.

[15] T. Bouwmans, F. El Baf, and B. Vachon. Background modelling using mixture of Gaussians for foreground detections: a survey. *Recent Patents on Computer Science*, 1(3):219–237, 2008.

[16] G. Bradski. Dr. Dobb's journal of software tools. Online: `http://opencv.org/documentation.html`, 2000.

[17] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems*, 13:409–415, 2001.

[18] S. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic videos. In *Electronic Imaging*, pages 881–892, 2004.

[19] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Carnegie Mellon University, 2000.

[20] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[21] E. Corvee, S. Bak, and F. Bremond. People detection and re-identification for multi-surveillance cameras. In *International Conference on Computer Vision Theory and Applications*, 2012.

[22] I. J. Cox and S. L. Hingorani. An efficient implementation of Reid's MHT algorithm and its evaluation for the purpose of visual tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[23] R. Cucchiara, C. Grana, , M. Piccardi, and A. Prati. Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In *International Conference on Image Analysis and Processing*, pages 360–365, 2001.

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[25] M. Davy, F. Desobry, A. Gretton, and C. Doncarli. An online support vector machine for abnormal event detection. *Signal Processing*, 86(8):2009–2025, 2006.

[26] H. M. Dee and S. A. Velastin. How close are we to solving the problem of automated visual surveilance? *Machine Vision and Applications*, 19(5-6):329–343, 2008.

[27] P. Dollár. Piotr's Image and Video Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`, Sept 2013.

[28] A. Doucet, S. Godsill, and C. Andrieu. On the sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[29] A. Elgammal, A. Duraiswami, R. Harwood, and L. Davis. Background and foreground modelling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, 2002.

[30] A. Elgammal, R. Harwood, and L. Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision*, pages 751–767, 2000.

[31] S. Y. Elhabian, K. M. El-Sayed, and S. H. Ahmed. Moving object detection in spatial domain using background removal technique: state-of-the-art. *Recent Patents on Computer Science*, 1(1):32–54, 2008.

[32] T. Fletcher. Support vector machines explained. Tutorial, University College London, 2009.

[33] A. Gilbert and R. Bowden. Multi person tracking within crowded scenes. In *Human motion- understanding, modelling, capture and animation*, pages 166–179. Springer, 2007.

[34] H. Grabner and H. Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition*, pages 260–267, 2006.

[35] H. Grabner, C. Leister, and P. Fua. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, volume 5302, pages 234–247. Springer, 2008.

[36] W. E. L. Grimson, C. Staufer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Computer Vision and Pattern Recognition*, pages 22–29, 1998.

[37] J. Heikkilä and O. Silvén. A real-time system for monitoring cyclists and pedestrians. In *Workshop on Visual Surveillance*, pages 74–81, 1999.

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

[39] A. D. Jepson, D. G. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *Transactions on Pattern Recognition and Machine Intelligence*, 25(10):1296–1311, 2003.

[40] S. J. Julier and J. K. Uhlmann. Technical report.

[41] S. J. Julier, J. K. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *American Control Conference*, pages 1628–1632, 1995.

[42] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-Based Surveillance Systems*, pages 135–144. 2002.

[43] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *International Conference on Computer Vision Workshops*, pages 1417–1424, 2009.

[44] J. Kato, T. Watanabe, S. Joga, J. Rittscher, and A. Blake. An HMM-based segmentation method for traffic monitoring movies. *Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1291–1296, 2002.

[45] S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, 2003.

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

[47] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure. Robust silhouette extraction technique using background subtraction. In *Meeting on Image Recognition and Understanding*, pages 552–557, 2007.

[48] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[49] P. Konstantinova, A. Udvarev, and T. Semerdjiev. A study of a tracking algorithm using global nearest neighbor approach. In *Computer Systems and Technologies*, pages 290–295, 2003.

[50] O. Lanz. Approximate Bayesian multibody tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1436–1449, 2006.

[51] K-C. Lee and D. Kriegman. Online learning of probabilistic appearance manifolds for video-based recognition and tracking. In *Computer Vision and Pattern Recognition*, pages 852–859, 2005.

[52] II Liggins, M. E., C-Y. Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85:95–107, 1997.

[53] M. Linderoth, K. Soltesz, A. Robertsson, and R. Johansson. Initialization of the Kalman filter without assumptions on the initial state. In *International Conference on Robotics and Automation*, pages 4992–4997, 2011.

[54] B. D. Lucas and T. Kanade. An iterative technique of image registration and its application to stereo. In *Internationl Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[55] K. Macnish. Unblinking eyes: the ethics of automated surveillance. *Ethics and Information Technology*, 14(2):151–167, 2012.

[56] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *Transactions on Image Processing*, 17(7):1168–1177, 2008.

[57] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Computer Vision and Pattern Recognition*, pages 302–309, 2004.

[58] C. L. Morefield. Application of 0-1 integer programming to multitarget tracking problems. *Transactions on Automatic Control*, 22(3):302–311, 1977.

[59] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou. Discriminative local binary patterns for human detection in personal album. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[60] K. G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16(3):682–687, 1968.

[61] J. C. Nascimento and J. Marques. Performance evaluation of object detection algorithms for video surveillance. *Transactions on Multimedia*, 8(4):761–774, 2006.

[62] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[63] A. B. Poore, N. Rijavec, T. N. Barker, and M. Munger. Data association problems posed as multidimensional assignment problems: problem formulation. In *Optical Engineering and Photonics in Aerospace Sensing*, pages 552–563, 1993.

[64] P. W. Power and J. A. Schoonees. Understanding background mixture models for foreground segmentation. In *Image and Vision Computing*, pages 267–271, 2002.

[65] G. W. Pulford and B. F. la Scala. Multihypothesis Viterbi data association: algorithm development and assessment. *Transactions on Aerospace and Electronic Systems*, 46(2):583–609, 2010.

[66] T. K. Ralphs and M. Güzelsoy. SYMPHONY 5.2.3 user manual. Online: `www.coin-or.org/download/binary/SYMPHONY/`, 2010.

[67] L. N. Rasid and S. A Suandi. Versatile object tracking standard database for security surveillance. In *International Conference on Information Science, Signal Processing and Their Applications*, pages 782–785, 2010.

[68] D. B. Reid. An algorithm for tracking multiple targets. *Transactions on Automatic Control*, 24(6):843–854, 1979.

[69] I. Reid. Estimation II. Tutorial, University of Oxford, 2001.

[70] B. Ristic, S. Arulampalam, and N. J. Gordon. *Beyond the Kalman filter: particle filters for tracking applications*. Artech House, 2004.

[71] D. A. Ross, J. Lim, R-S. Lin, and M-H. Yang. Incremantal learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[72] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. Online random forests. In *International Conference on Computer Vision*, pages 1393–1400, 2009.

[73] B. Saha. Introduction to particle filters. Technical Report, NASA, 2008.

[74] B. Schökopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research, 2000.

[75] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *International Conference on Robotics and Automation*, pages 1665–1670, 2001.

[76] M. Shah, O. Javed, and K. Shafique. Automated visual surveillance in realistic scenarios. *Transactions on Multimedia*, 14(1):30–39, 2007.

[77] R. W. Sittler. An optimal data association problem in surveillance theory. *Transactions on Military Electronics*, 8(2):125–139, 1964.

[78] A. Statnikov, L. Wang, and C. F. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9(1):319–329, 2008.

[79] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[80] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. M. Buhmann. Topology free hidden Markov models: application to background modeling. In *International Conference on Computer Vision*, pages 294–301, 2001.

[81] R. L. Streit and T. E. Luginbuhl. A maximum likelihood method for probabilistic multihypothesis tracking. In *International Symposium on Optical Engineering and Photonics in Aerospace Sensing*, pages 394–405, 1994.

[82] A. Tavakkoli, M. Nicolescu, G. Bebis, and M. Nicolescu. Non-parametric statistical background modelling for efficient foreground region detection. *Machine Vision and Applications*, 20(6):395–409, 2009.

[83] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *International Conference on Computer Vision*, pages 255–261, 1999.

[84] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. A comparison of color features for visual concept classification. In *International Conference on Content-based Image and Video Retrieval*, pages 141–150, 2008.

[85] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[86] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[87] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report TR-95-041, University of North Carolina, 2006.

[88] C. R. Wren, A. Azarbayejani, T. Darrel, and A. Pentland. Pfinder: real-time tracking of the human body. *Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[89] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song. Recent advances and trends in visual tracking: a survey. *Neurocomputing*, 74(18):3823–3831, 2011.

[90] T. Yang, Q. Pan, J. Li, and S. Z. Li. Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *Computer Vision and Pattern Recognition*, pages 970–975, 2005.

[91] A. Yilmaz, O. Javed, and M. Shah. Object tracking: a survey. *ACM Computing Surveys*, 38(4):13, 2006.

[92] Q. Yu, T. B. Dinh, and G. Medioni. Online tracking and reaquisition using co-trained generative and discriminative trackers. In *European Conference on Computer Vision*, pages 678–691. 2008.

[93] J. Zhou and J. Hoang. Real time robust human detection and tracking system. In *Computer Vision and Pattern Recognition Workshop*, pages 149–164, 2005.

[94] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition*, pages 28–31, 2004.