Robust wide-area multi-camera tracking of people and vehicles to improve CCTV usage

By

Fei Yin

This thesis is submitted to the Faculty of Computing, Information Systems and Mathematics, Kingston University, London, for the degree of

Doctor of Philosophy

February, 2011

Kingston University London
Faculty of Computing, Information Systems and Mathematics

Acknowledgements

It would not have been possible to complete the work in this thesis without the help and support of my friends and colleagues at Kingston University, London.

First of all, I would like to thank my supervisors: Dr Dimitrios Makris, Dr James Orwell and Prof. Sergio A.Velastin, for being my supervisors during my time at Kingston University. They have given me direction, motivation, guidance and ensured that I kept a clear set of research goals and objectives, while at the same time encouraging me to work independently. The experience I have gained from my study in Kingston will be of great benefit for both my life and work in the future.

I am grateful to BARCO for sponsorship of my PhD project.

I would also like to thank my colleagues and friends that supported me and accompanied me all these years: Norbert Buch, Alberto Colombo, Damien Simonnet, Zhen Cai, Beibei Zhan, Paul Kuo and so many others.

Last but not least, I would like to express my love and gratitude to my wife JunLin Zhang and my parents, Jun Yin and XiaoLin Tian, for their continuous encouragement and support during my PhD period. These thanks extend to all my family members in China.

Table of contents

Declarations

The author confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to others' work.

Some parts of the work presented in this thesis have been published or submitted for publication in the following articles:

Journals:

F. Yin, D. Makris, S.A. Velastin, J. Orwell, "Quantitative Evaluation of Different Aspects of Motion Trackers under Various Challenges", in The Annals of the BMVA, (5) British Machine Vision Association, pp. 1-11, 2010.

F. Yin, D. Makris, S.A. Velastin, "Time efficient ghost removal for motion detection in visual surveillance systems" in 'IET Electronics Letters', 44(23) Institution of Engineering and Technology, pp. 1351-1353. ISSN 0013-5194, Nov, 2008.

In preparation for journal: F. Yin, D. Makris, S.A. Velastin, "Homography-based multiple camera object tracking ", IEEE Transactions on Systems, Man & Cybernetics, Part C: Applications & Reviews

Conferences:

F. Yin, D. Makris, J. Orwell, S.A. Velastin, "Learning non-coplanar scene models by exploring the height variation of tracked objects", in The Tenth Asian Conference on Computer Vision (ACCV2010), Queenstown, New Zealand, November, pp. 1564-1577, 2010.

N. Buch, F. Yin, J. Orwell, D. Makris and S. A. Velastin, "Urban Vehicle Tracking using a Combined 3D Model Detector and Classifier", In 13th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES 2009, Part I, LNCS 5711, pp. 169-176, Santiago, Chile, September 2009.

F. Yin, D. Makris, S.A. Velastin, "Real-time ghost removal for foreground segmentation methods", IEEE International Workshop on Visual Surveillance (VS2008), October 17, Marseille, France, 2008.

F. Yin, D. Makris, S.A. Velastin, "Performance Evaluation of Object Tracking Algorithms", 10th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2007), October, Rio de Janeiro, Brazil, 2007.

Presentations:

F. Yin, D. Makris, S.A. Velastin, J. Orwell, "Quantitative evaluation of different aspects of motion trackers under various challenges", One Day BMVA symposium on Security and surveillance: performance evaluation, 12th December, British Computer Society, London, 2007.

List of Abbreviations

Blob	Binary large object
CCTV	Closed Circus Television
CDT	Correct Detected Track
CT	Closeness of Track
EM	Expectation Maximization algorithm
FAT	False Alarm Track
FN	False Negative
FOV	Field of view
FP	False Positive
GMM	Gaussian Mixture Model
GP	Ground Plane
GT	Ground Truth
HSV	Hue Saturation Value colour space
IDC	ID change
i-LIDS	Imagery library for intelligent detection systems
KF	Kalman Filter
LT	Latency of the system Track
OpenCV	Open Computer Vision library
PAL	Phase Alternating Line (TV encoding)
PCA	Principal Component Analysis
PETS	Performance Evaluation of Tracking and Surveillance
RGB	Red Green Blue colour space
SIFT	Scale Invariant Feature Transform
TC	Track Completeness
TDE	Track Distance Error
TDF	Track Detection Failure
TF	Track fragmentation

TFL	Transport for London
TN	True Negative
TP	True Positive
VIPER	Video Performance Evaluation Resource
Viper -GT	Video Performance Evaluation Resource - Ground Truth

List of figures

List of Tables

Abstract

This thesis describes work towards a more advanced multiple camera tracking system. This work was sponsored by BARCO who had developed a motion tracker (referred to as the BARCO tracker) and wanted to assess its performance, improve the tracker and explore applications especially for multi-camera systems. The overall requirement then gave rise to specific work in this project: Two trackers (the BARCO tracker and OpenCV 1.0 blobtracker) are tested using a set of datasets with a range of challenges, and their performances are quantitatively evaluated and compared. Then, the BARCO tracker has been further improved by adding three new modules: ghost elimination, shadow removal and improved Kalman filter. Afterwards, the improved tracker is used as part of a multi-camera tracking system. Also, automatic camera calibration methods are proposed to effectively calibrate a network of cameras with minimum manual support (draw lines features in the scene image) and a novel scene modelling method is proposed to overcome the limitations of previous methods. The main contributions of this work to knowledge are listed as follows:

A rich set of track based metrics is proposed which allows the user to quantitatively identify specific strengths and weaknesses of an object tracking system, such as the performance of specific modules of the system or failures under specific conditions. Those metrics also allow the user to measure the improvements that have been applied to a tracking system and to compare performance of different tracking methods.

For single camera tracking, new modules have been added to the BARCO tracker to improve the tracking performance and prevent specific tracking failures. A novel method is proposed by the author to identify and remove ghost objects. Another two methods are adopted from others to reduce the effect of shadow and improve the accuracy of tracking.

For multiple camera tracking, a quick and efficient method is proposed for automatically calibrating multiple cameras into a single view map based on homography mapping. Then, vertical axis based approach is used to fuse detections

from single camera views and Kalman filter is employed to track objects on the ground plane.

Last but not least, a novel method is proposed to automatically learn a 3D non-coplanar scene model (e.g. multiple levels, stairs, and overpass) by exploiting the variation of pedestrian heights within the scene. Such method will extend the applicability of the existing multi-camera tracking algorithm to a larger variety of environments: both indoors and outdoors where objects (pedestrians and/or vehicles) are not constrained to move on a single flat ground plane.

1 Introduction

The main application area of this thesis is automatic visual surveillance using single or multiple cameras. Nowadays, the demand of security leads to the growing need of visual surveillance in many environments and hence tens of thousands of CCTV cameras have been installed for monitoring public areas in the UK, especially for public areas such as train and tube stations, airports, motorways, main streets, car parks, banks and shopping centres.

Generally speaking, video surveillance systems often have a set of cameras which send their video signals to display monitors and often at the same time to either digital or analogue recording devices. Video surveillance systems can provide more centralized, cost effective and efficient monitoring of traffic and public areas to ensure security and to prevent crime actions. Figure 1-1 is an illustration of camera installation in London and the resulting views.

Figure 1-1 Illustration of installation of CCTV cameras (top left), the control room (top right), and CCTV camera views from the i-LIDS dataset (bottom)

Nowadays, with the increase of processor speeds and reduced hardware costs it has become applicable to install large networks of CCTV cameras in order to have a larger visual accessibility of the monitored area. However, this raises the problem of how to continuously (24 hours), reliably (no misses or false alarms) and effectively watch over and obtain information from those CCTV video sequences since there are some limitations for human resources to do so:

- On-line: The monitors mainly display trivial and boring events for the majority of time, it is very likely that a significant percentage of interesting events are missed by human operators. (Wallace and Diffley, 1998)
- Off-line: Sometimes, it is required to recall an event that occurred during a specific date and time which is laborious task to look through a huge amount (several days or months) of video data for human operators.

To overcome the limitations mentioned above and to assist human operators, a significant amount of research has been done during the last 20 years to automatically analyze and extract information from digitised video data using digital image processing and computer vision techniques. Many algorithms have been developed for automatic object detection, object tracking, system/camera calibration, event detection, activity/behaviour analysis, face detection/recognition and object recognition (Hu *et al*., 2004).

Although some algorithms have been put into real time surveillance systems for practical use, current surveillance systems are limited at object tracking and event detections. For instance, Transport for London (TFL) launched a project called Image Recognition and Incident Detection (IRID) (Cracknell, 2007, Cracknell, 2008) in order to test the performance of existing surveillance systems on the following criteria: congestion, stopped vehicles, banned turns, vehicle counting, subway monitoring etc, The outcome of the project shows poor performance in tracking based detection (~20% tracking completeness rate), clearly showing limitations in capability.

Poor performance is often caused by failure of a specific part of a surveillance system (e.g. object detection, data association, tracking etc.) and may be due to different reasons: fast illumination changes, non-interesting apparent motion, weather conditions, intersection between objects, occlusions etc.

For multiple camera systems, significant amount of manual work is required for the calibration of each camera view which is not effective at all if the camera is moved often or a many new cameras are installed. Furthermore, existing camera calibration methods have an important constraint that all objects must move on a single coplanar ground plane so that scenes with stairs, overpasses, etc. will present problems for such methods.

Regarding to the issues mentioned above, this thesis first investigates existing methods for object detection and tracking using either single or multiple cameras, and then proposes new methods for more robust tracking, more effective camera calibration and quantitative performance evaluation for both single and multiple camera surveillance systems.

The work presented in this thesis can provide multiple benefits to current surveillance systems: firstly, a tracking based evaluation framework is proposed for comprehensive evaluation of different aspects of a tracking system (e.g. object detection, data association, tracking etc) and it can be further used to identify specific failures of tracking systems and quantitatively measure improvement that has been done for a tracking system. Secondly, object detection and tracking will benefit from the addition of new proposed modules to detect non-interesting apparent motions (referred to as 'ghosts'). The installation and maintenance of a surveillance system will be much faster and effective because of the proposed semi-automatic camera calibration method. Finally, a new algorithm for automatic learning of a 3D non-coplanar scene model is proposed which can overcome the constraint of previous tracking methods which assume a single flat ground plane model.

The work in this thesis (e.g. automatic motion detection, target tracking, camera calibration and scene learning) can mainly be used to support smart CCTV surveillance to improve public security by detecting crime activities (e.g. illegal entering of a building, illegal parking etc). In addition to security applications, the methods proposed in this thesis can also be used in other applications such as traffic management (e.g. traffic flow measurement, traffic accident detection, abandoned items detection), medical imaging (e.g. blood flow detection), military

tasks (e.g. patrolling national borders, measuring the flow of refugees), and entertainments (e.g. 3D games, Virtual 3D space modelling).

1.1 Research aims and objectives

The principal aim of this thesis is to evaluate existing object tracking algorithms and propose methods for more advanced and effective multi-camera tracking. The objectives of this thesis are listed as follows:

- To propose a track based evaluation framework to quantitatively measure the performance of different aspects of motion[1] (object) tracking systems. In addition, it will help to identify possible failures of specific modules of the tracking system and measure the improvements that applied to it.
- To provide a motion detection and tracking system which is more accurate and robust against challenges such as illumination changes, shadows and apparent motions.
- To provide a framework for multiple camera calibration and object (both pedestrians and vehicles) tracking across cameras. The method needs to be robust against segmentation noises, occlusions etc.
- To provide a new scene environment modelling method that is able to learn a 3D non-coplanar ground model (e.g. scenes where multiple levels exists such as stairs, overpasses and so on).

1.2 Organisation

Chapter 2 presents background information for the context of this thesis. Firstly, popular computer vision techniques for motion detection and motion tracking are investigated. Then, previous works related to multi-camera calibration and tracking are discussed. Finally, methods for performance evaluation of object detection and tracking algorithms are discussed.

[1] In this work we use the terms “motion detection” and “motion tracking” as they are usually used in the visual surveillance literature, generally meaning object/foreground detection and object tracking. This therefore does not preclude detection and tracking of stationary objects.

In chapter 3, a rich set of tracking based metrics is proposed to assess different modules of tracking systems (e.g. motion segmentation, motion tracking and data association) and identify specific failures of motion tracking (e.g. incomplete tracking of objects, ID confusing between objects, false alarms etc). The practical value of the proposed evaluation metrics is that if any improvements have been made to a given tracker, it is crucial to develop a way in which changes, however small, could be objectively assessed on large amounts of data, so as to measure improvements and, as importantly, failures. Therefore a great deal of effort was put to this aspect of the project. In fact, the proposed evaluation framework has not only been used by this work but also by many other projects within the research group. In particular, it is demonstrated that the BARCO tracker over performs the OpenCV tracker which is considered by many as a reference tracking system.

In chapter 4, additional modules such as ghost elimination based on edge similarity, shadow detection based on HSV colour space and improved Kalman filter have been added to the BARCO tracker to improve its performance. The track based evaluation frame work proposed in chapter3 is used to measure the improvements quantitatively.

In chapter 5, an object correspondence method based on the vertical axes of objects is proposed which can be applied to both pedestrians and vehicles and it is shown to be robust against segmentation noise and occlusion. The spatial relationships between multiple cameras are determined by an semi-automatic homography based calibration method. The proposed method is quick and effective to produce a single view map for a camera network and does not require any site map.

In chapter 6, a novel method is proposed to learn a non-coplanar ground model by exploring the pedestrian height variation within the scene. The proposed method extends the applicability of previous multi-camera calibration and tracking algorithms to a larger range of environments where objects that are not constrained to move on a single flat ground plane.

Finally, conclusions and suggestions for further work are presented in chapter 7.

2 Literature Review

The purpose of this chapter is to provide a survey of the research that has already been published in relation to performance evaluation, camera calibration, single and multiple camera tracking.

An overview of the architecture of visual surveillance systems is given in section 2.1. The basic tasks and commonly used computer vision techniques for visual surveillance (e.g. foreground segmentation and object tracking) are discussed in section 2.2. A survey of existing multiple camera calibration and tracking methods is given in section 2.3. Then, basic concepts and methods for performance evaluation are discussed in section 2.4. Please note that some of the methods discussed in this chapter are outside the scope of this thesis but are included for completeness.

2.1 Visual surveillance

The traditional visual surveillance system, firstly introduced in the early 60s, consists of a set of CCTV cameras, connected to display monitors and possibly to some recording devices. Later, the ability to record on magnetic tape (typically by time lapsing) was added. Afterwards, in the late 90s, rapid advances in digital technologies allowed networking and digital video recording. This then made it possible to conceive of computer-based image analysis methods that could enhance the productivity and effectiveness of human operators. This is what became known as Visual Surveillance. Since then, visual surveillance has become one of the most active research topics in computer vision and numerous techniques have been developed to detect, recognize and track objects of interest (e.g. people, vehicles, etc.) from image sequences. The current trend is to develop intelligent visual surveillance to replace the traditional passive video surveillance which has been proved ineffective (Wallace and Diffley, 1998).

Generally speaking, visual surveillance in dynamic scenes includes the following steps although different systems may have slightly different steps: modelling of environments (background modelling, camera calibration for single/multi-cameras etc.), foreground detection, data association between frames, object tracking, behaviour analysis and data fusion of multiple cameras. Figure 2-1 shows a general framework of visual surveillance.

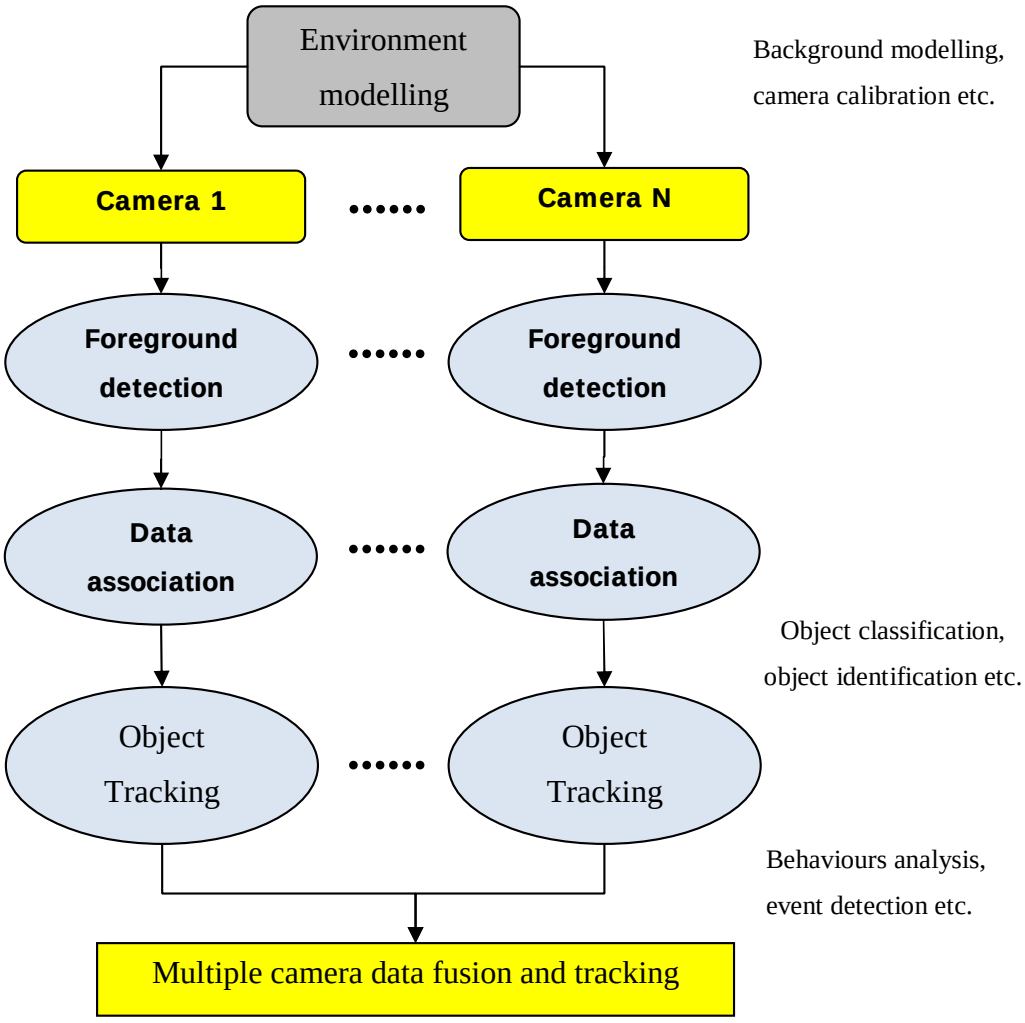


Figure 2-1 Framework of a typical visual surveillance system

2.2 Single camera tracking

Once the video data has been captured by a CCTV camera, the next step would be to identify any object activity in the scene or camera FOV (field of view). In order

to do that, most existing visual surveillance systems start with background modelling and motion detection which aims to detect moving objects of interest. A statistical background model is usually used to estimate foreground pixels, which are then grouped with a basic model to form objects (e.g. connected component analysis) and then propagated through the system until the tracking stage. In Figure 2-2, the basic structure of motion detection and tracking is illustrated.

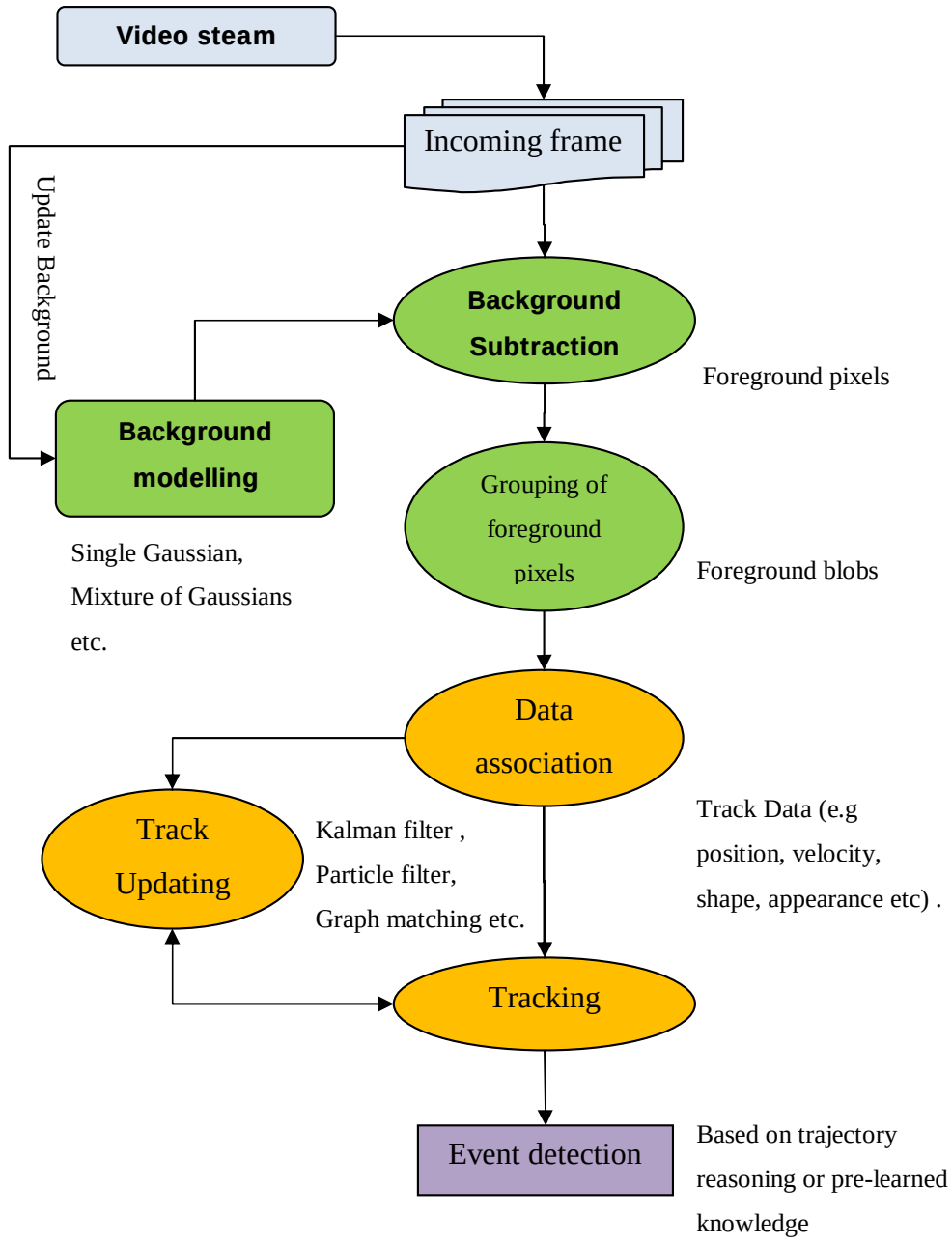


Figure 2-2 Basic motion detection and tracking framework

In this section, the basic elements of single camera tracking will be discussed and many of the popular techniques for foreground segmentation and object tracking will be reviewed.

2.2.1 Foreground segmentation

Foreground segmentation (or motion detection) is the first step of many surveillance systems and applications. Foreground is defined as every object (vehicles and pedestrians), which is not fixed furniture (buildings, plants etc.) of a scene where fixed could normally mean months or years. Although this definition is clear and easy to human understanding, it is not easy to implement an automatic solution. The most common and widely used approach to estimate the foreground is that the current frame is compared against a background model to identify the difference (or 'motion'), provided that the camera is stationary. Methods for using motion as the main cue for foreground segmentation will be discussed in the next few sections.

The main challenges for motion based detection come from illumination changes, irrelevant motion and occlusion of objects. Illumination changes often occur in outdoor scenes. For instance, fast moving clouds may cause sudden change of lighting conditions and shadows. Irrelevant motion can be caused by various reasons: vibrating of vegetation, moving cast shadows, reflection from windows or puddles on the ground, the appearance of a non-existing object (referred to as "ghost") caused by the moving off of a previously background object etc. Any of these conditions can cause an error or failure in the motion detection process. A few examples of those failures are shown in Figure 2-3.

Figure 2-3 Examples of detection errors caused by puddles on the ground (top left) and quick illumination change (top right) ghost (bottom left) and vibrating of vegetation (bottom right)

2.2.1.1 Optical flow and frame differencing

Many different approaches have been proposed to estimate the foreground in the last few decades. Optical flow (Horn and Schunk, 1981) was one of the earliest motion detection methods but later considered as too computationally expensive and too sensitive to noise thus not suitable in real time systems.

At the same period, a quicker and simplified version of optical flow: frame differencing (Jain, 1981) was proposed. Motion is detected by computing the pixel by pixel difference map between two consecutive frames. If the difference is larger than a predefined threshold, the pixel will be classified as foreground pixel. In (Park *et al.*, 2007), frame differencing is used to detect street parking vehicles. However, the frame differencing method is very sensitive to the threshold and cannot cope with challenges like rapid illumination changes or vibrating of vegetation etc.

2.2.1.2 Background subtraction

Later, an evolution of the frame differencing method: background subtraction is proposed, which uses an image as reference (often called background image) to identify foreground moving objects in the video sequence. A threshold is applied to compute the difference between the current frame and the background image. The threshold can either be constant (Rosin, 1997) or dynamic as used in (Gupte *et al.*, 2002).

One of the most commonly used methods to estimate a background image was the background averaging: (Gupte *et al.*, 2002), (Huang and Liao, 2004) and (Chen and Zhang, 2007). This algorithm has little computational cost, however, it is likely to produce tails behind moving objects due to contamination of the background with the appearance of the moving objects.

In order to improve robustness of foreground detection, a single Gaussian model is used for modelling the background. Instead of using only the average, the mean and standard deviation of each pixel are computed. A new pixel is classified depending on its position in the Gaussian distribution, which is the statistical equivalent to a dynamic threshold. (Kumar *et al.*, 2003, Morris and Trivedi, 2006, Su *et al.*, 2007) used the single Gaussian background model. It is proved to be more robust than previous background subtraction methods (frame differencing and optical flow). However, a single Gaussian model cannot cope with multimodal backgrounds.

2.2.1.3 Mixture of Gaussians

An alternative to the background subtraction methods mentioned in the previous section is a more sophisticated and promising algorithm proposed by (Stauffer and Grimson, 1998) and (Stauffer and Grimson, 2000). They presented the idea of modelling each pixel by a mixture of Gaussians and updating each pixel with new Gaussians during run-time. Their method is robust enough for outdoor environments and slow scene changes but cannot handle rapid illumination changes very well (e.g. fast moving clouds). Also the computational load is significantly higher than previous background subtraction methods. Later, a modified version of

their method proposed by (Xu and Ellis, 2001), allows the background model to adapt to illumination changes very fast.

An improved version of Gaussian Mixture Model proposed by (KadewTraKuPong and Bowden, 2001) is available in the OpenCV library (OpenCV, nd) and is commonly used in research. The proposed method employed online EM algorithm for the initial learning stage to improve convergence and switching to recursive filter learning after sufficient samples were observed. However, the limitation of the approach remains its computational complexity and higher time requirement.

2.2.1.4 Shadow detection

During the process of background subtraction, the issue of moving shadows incorrectly being detected as foreground needs to be considered carefully. Shadows can cause distortion of object shape and incorrect merging of objects. Therefore, significant efforts have being put in this area.

A comprehensive evaluation of moving shadow detection is presented in (Prati *et al*., 2003). The authors grouped shadow detection methods into four different categories (statistical class: parameter based or non parameter based and deterministic class: model based or non-model based). Evaluation results show that no single approach performs best, furthermore, the type of applications determines the best suited algorithm. For a general-purpose shadow detection system, with minimal assumptions, a pixel based deterministic non-model based (also called pixel based) approach assures best result. An example of this is (Cucchiara *et al*., 2001) which uses HSV colour space for shadow suppression and it is also used in later systems (Cucchiara *et al*., 2003), (Johansson *et al*., 2009), (Huang and Wu, 2010). For shadow detection in specific environment or specific tasks (penumbra, objects with certain texture etc), the model based deterministic approach which is generally more complex and with heavier computational load is more reliable. For indoor environment where scene illumination is more constant, the statistical approaches are the more effective (Trivedi *et al*., 2000), (Hu and Su, 2007).

2.2.2 Object tracking

After moving objects have been detected, it is also very important for the system to be able to keep a constant id and record the path for each detected object when it moves through the camera FOV. Tracking is performed in two steps: Firstly, features for the object or foreground regions are generated in every video frame (e.g such as position, size, velocity, colour etc). Secondly, a data association step has to provide correspondences between the regions of consecutive frames based on object features and a dynamic model. Temporal and spatial consistency constraints are required to avoid confusion of tracks and to smooth noisy position measurements from detection. Tracking data is generally expressed using the 2D coordinates (the bounding box) of the image plane. However, it can be converted to the 3D scene coordinates using a ground plane model and/or multiple views of the scene. In this section, motion models and algorithms for object tracking are discussed.

2.2.2.1 Kalman filter

One of the most famous mathematical tools for object tracking is the Kalman filter which was originally introduced by (Kalman, 1960). A dynamic system (an object in our case) is modelled by a linear state transition equation (also called the process equation) and a measurement equation. Both are assumed to be corrupted by independent Gaussian noise. Most people in visual surveillance use a constant velocity process model. The task of the filter is then to dynamically calculate the values of the state vector. The optimal state of a linear motion model with constant velocity is estimated. There are two main steps of Kalman filter: the prediction stage is used to extrapolate the new position of an object in the next time step based on the state transition model. The prediction can be associated with new measurements or can be used to trigger detectors. An updating step uses the detection as measurement and updates the filter state. Most people assume a constant velocity model and the measurement noise and processing noise are both Gaussian and independent to each other. The Kalman filter has been used successfully in many works such as (Xu and Ellis, 2002), (Black *et al.*, 2004), (Messelodi *et al.*, 2005b), (Song and Nevatia, 2007) etc. The extended Kalman

filter (EKF) can facilitate non-linear model of the system. However, the Kalman filter only propagates a single object state between frames compare to the multiple hypotheses for particle filters in the next section.

2.2.2.2 Particle filter

Besides the Kalman filter, the particle filter is a more generalized and advanced tool for object tracking and it was originally introduced by (Gordon *et al.*, 1993) and first used by (Isard and Blake, 1998) for a computer vision application. Nowadays, particle filter is used more and more for object tracking: (Czyz *et al.*, 2007), (Mauthner *et al.*, 2008), (Wang *et al.*, 2009) etc. Even for real-time applications: (Nummiaro *et al.*, 2003), (Zhou *et al.*, 2004), (Kwok *et al.*, 2004) etc. The particle filter approximates any probability distribution with a large set of particles. These particles are propagated through time using importance sampling, allowing any arbitrary process model to be used, thus offering flexibility. This overcomes the constraint of a single Gaussian distribution of Kalman filters and getting better results. However, the disadvantages of particle filter are its difficulty to determine optimal value for crucial parameters (e.g. importance density, number of particles) and the potential problems of degeneracy and loss of diversity.

2.2.2.3 Mean shift tracking

There are some other methods proposed for object tracking. mean shift algorithm (Carnegie, 2003) is one of the most popular appearance based object tracking method. A colour histogram is used to describe the target region. The similarity between the template region and the current target region is measured; finally, tracking is accomplished by interactively finding the local minima of the similarity functions. However, the difficulty of choosing a proper kernel size scale makes the method good at tracking object with relative constant size on the image plane. Also, the similarity measures require a calculation that is quadratic in the number of samples which makes it hard for real-time applications. Later, (Yang *et al.*, 2005) proposed a new simple symmetric measure to make it effective for real-time tracking. (Yilmaz, 2007) proposed a new mean shift algorithm with automatic scale and orientation selection to overcome the problem of object sizes.

2.3 Multiple camera tracking

In the previous section, the basic structure of single camera tracking has been outlined and a review of the most commonly used methods for object detection and tracking for single camera view has been presented. In real world applications, a visual surveillance system normally comprises more than one cameras. Using multiple cameras for object tracking brings some advantages over single view tracking.

- Visual surveillance using multiple cameras can provide an increased field of coverage.
- Surveillance using multiple cameras may be able to solve the problem of static or dynamic occlusions that cannot be solved with single camera. For example, when a pedestrian is walking behind a tree, he/she cannot be seen in one camera view but may still be visible in another camera view.

For the reasons listed above, visual surveillance using multiple cameras has attracted much attention in the past decade. Usually, an architecture is required to support communication between cameras (e.g. passing of track and identification data). One of the most commonly used camera network architecture is the distributed architecture (see Figure 2-4) where some low level processing (object detection, classification etc) is performed at each camera view before communication with the central processing unit (multi-view correspondence and tracking).

There are different cases for multi-camera tracking: one is matching objects between overlapping camera views. Usually, location and appearance of objects are used as important cues for matching. The other case is tracking objects between non-overleaping cameras. In such case, the transition times between cameras are specified or learned to handover objects between adjacent cameras with temporal and spatial gaps in addition to the appearance cues. This significantly restricts the search space when trying to pick up an object on a different camera. In this section, some of the state of the art methods used for camera calibration and object tracking across multiple cameras will be reviewed later in this section.

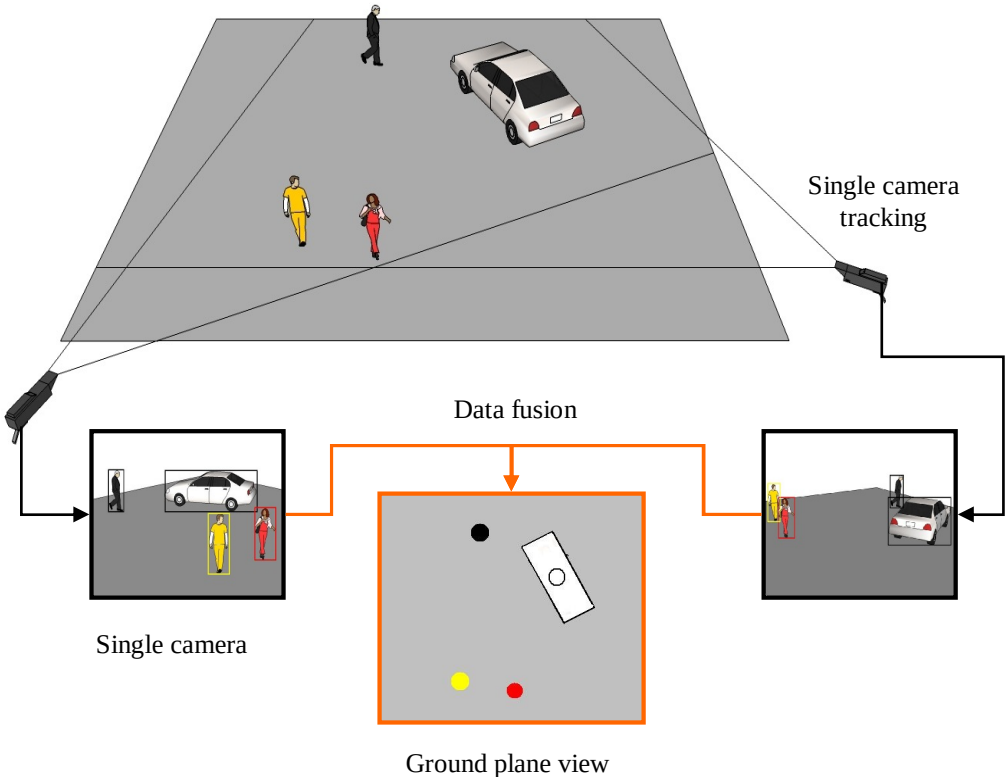


Figure 2-4 illustration of distributed multi-camera tracking architecture

2.3.1 Object correspondence

A basic requirement of multi-camera tracking system is that a unique id should be assigned for an object moving across a network of cameras. Therefore, object correspondence which involves finding the match for the same object from different camera views is an essential task for multi-camera tracking (see for example Figure 2-4). There are mainly two types of matching: one is the geometric based that uses geometric features transformed to the same space (e.g. a common ground plane) to establish object correspondence between camera views. The other one is an appearance based method which uses appearance cues (e.g. colour, size, texture etc) to establish correspondence. More recently, some statistical learning methods have been proposed to learn the spatial relationship or transition time between adjacent cameras in order to do handover objects between adjacent cameras with temporal and spatial gaps (Makris *et al.*, 2004).

2.3.1.1 Geometric based methods

Geometric location of an object is the most commonly used feature for object correspondence between cameras. Object correspondence based on location is simplified if the scene conforms to the ground plane constraint which assumes that there is a single flat ground plane and moving objects are constrained to move on it. This assumption allows the matching of moving objects between overlapping camera views to be simplified to a planar transformation. Usually, this type of matching requires that some form of camera calibration is available.

(Khan and Shah, 2003) used the points located on a pedestrian's feet to do object correspondence, based on the homography mapping from one camera view to another. However, in many cases, people's feet are likely to be occluded which can significantly affects the reliability of their method. (Chilgunde *et al*., 2004) used a ground plane Kalman filter to track vehicles across blind regions of multiple cameras. However, the method only works well under the conditions that the direction and speed of the vehicles is stable (e.g. as on a highway). (Thirde *et al*., 2005) employed the Kanade Lucas Tomasi (KLT) feature tracking algorithm to track independent features from frame to frame, and they associated 3D ground plane tracks with measurements from multiple cameras based on a nearest neighbour constraint. However, predefined threshold for object correspondence is sensitive to noise and can be significantly affected by occlusion. (Hu *et al*., 2006) proposed a simple and robust method, based on the principal axes of people, to match people across cameras. Their method is less sensitive to segmentation noise and able to locate people's foot points accurately even under occlusion. However, their method of calculating the principal axis only applies to pedestrians and is not appropriate for vehicles.

All the geometric based methods mentioned above are based on the assumption of a flat ground plane and camera calibration is required in order to obtain an object's location on the ground plane.

2.3.1.2 Appearance based methods

Besides geometric location, appearance of objects is also used as an important cue for object correspondence. One of the most popular appearance cues used in object correspondence between cameras is colour. (Javed *et al.*, 2005) used a Gaussian distribution to model the change of appearance of the same person from one camera to another in order to establish correspondence. (Cheng *et al.*, 2006) proposed the idea of Major Colour Representation (MCR) which clusters the colour histogram of an object into major colours: the colours of upper clothing, lower clothing and global appearance. Later, (Madden *et al.*, 2007) proposed to add two extra colour features relating to the upper and lower clothing colours of an individual to the global colours to allow a more sensitive analysis of the spatial positioning of a person's colours.

However, issues still remain for object correspondence using colour no matter which colour space (RGB, HSV, YCbCr etc.) or what kind of colour descriptor is used. For instance, it is difficult to do colour matching when pedestrians wear clothes with similar colours (e.g. a black or gray suit). In addition, colour appearance matching is also affected by different camera types and illumination conditions. For instance, colour calibration (Javed *et al.*, 2005) is required when colour responses are different from different cameras. Also, colour features become unreliable under poor lighting conditions where everything turns into gray (e.g. dark cloud or night).

Besides colour, other appearance cues are also used or mixture with colour for object matching, such as height and gait of pedestrian used in (Madden and Piccardi, 2007) and (Madden *et al.*, 2007).

2.3.1.3 Transition model based methods

In order to handle temporal and spatial gaps between cameras, (Makris *et al.*, 2004) proposed a statistic method for leaning multi-camera topology using temporal correlation of objects that transiting between adjacent camera views. Furthermore, the entry and exit zones of a network of non-calibrated cameras were identified and the links between those entry and exit zones were established. (Stauffer, 2005) proposed an unsupervised hypothesis testing method for estimating transition

probabilities between non-overlapped views. The method seemed to provide reasonable results, although it was tested only with synthetic data. Similar methods based on transition model are proposed by (Tieu *et al.*, 2005), (Stauffer, 2005), (Farrell *et al.*, 2007) etc.

The transition model methods can deal with object correspondence and tracking between cameras with non-overlapping views and can be further improved by fusion of other types of information (e.g. appearance cues). However, when the network dynamics are complex or the traffic distribution has big variation, the technique will have substantial difficulty. For instance, problems may arise if an object stops in the blind region between camera views, and after a while, moves into the next camera view or even moves to the opposite direction. It is unlikely that under these circumstances the system will be able to identify the target correctly.

Although there are drawbacks for the transition time based methods mentioned above, it can be beneficial to use both space-time and appearance features to complement each other to achieve successful object tracking between cameras. Quite a few works, for example (Javed *et al.*, 2008), (Chen *et al.*, 2008) used both space-time and appearance models for tracking objects across multiple non-overlapping cameras. Time interval and colour transfer functions are learned between cameras during a training phase. However, none of those methods have evaluated the effect of using different weights between each cue which may lead to better or worse results. In addition, because of the diversity of applications and lack of public available datasets, all those works are done on the author's proprietary data which makes it very difficult to compare the performance of their methods.

2.3.2 Camera calibration

As mentioned in the previous section, camera calibration is required for object tracking across cameras. In this section, some of the most popular camera calibration methods are reviewed.

2.3.2.1 Geometric calibration

Geometric camera calibration means to find the parameters of a projection that map a point from one camera view to another camera view or to the real world coordinates.

The most fundamental and well established geometric calibration method was originally proposed by (Tsai, 1986). Although this type of calibration is accurate for object tracking as it recovers the full camera parameters (5 intrinsic and 6 extrinsic parameters), it requires a significant amount of manual work and cannot adapt to changes of environment easily (e.g. adding or removing of cameras).

Based on Tsai's camera calibration method, some automatic geometric camera calibration methods are proposed for quick and effective camera registration by exploit the observed activities of the scene (e.g. tracked objects). (Renno *et al.*, 2002) developed an auto-calibration procedure to recover the image to ground plane homography by accumulating tracks. However, their method still needs some site parameters such as the height of each camera and based on the assumption of an average pedestrian height which may not always be true. (Krahnstoever and Mendonca, 2006) proposed an automatic calibration method that uses foot-to-head plane homology of tracked pedestrians. However, there are many sources of noise (e.g. pedestrian heights variation, segmentation noise etc) that a careful Bayesian formulation of the problem is required.

Besides Tsai's method, homography is also frequently used for calibration purposes. The homography transform between camera views can be recovered either manually (Kayumbi and Cavallaro, 2008) or automatically (Stauffer and Tieu, 2003). The selection of correspondence points (automatic methods) can be obtained through trajectory points correspondence (Black and Ellis, 2005) (Khan and Shah, 2009), feature points or feature lines correspondence (Zhang and Scanlon, 2008). However, trajectory points based methods heavily depend on trajectories which are prone to have errors and feature-point/line based methods assume that the ground plane in each view is sufficiently textured in order to facilitate a reliable point correspondence.

All the methods mentioned above are limited by a single flat ground plane constraint which means that they are not able to deal with scenes where multiple non-coplanar planes present (e.g. stairs, overpass, multiple levels).

2.3.2.2 Topographic Calibration

Besides geometric relationship, cameras can also be linked together by learning the activities between each other thus a topology of the camera network is built. Much work has been done to learn the camera network topology: (Ellis *et al*., 2003), (Makris *et al*., 2004), (Tieu *et al*., 2005), (Stauffer, 2005), (Farrell *et al*., 2007) etc.

Generally speaking, topographic calibration methods collect lists of entry and exit events of objects in different camera views. Then, this list is used to find correlations between the exit time of an object in one camera view and the entry time of another object in another camera view. If a correlation is found, a topological link is assumed between the two camera views. The link is usually represented by the transition time between entry and exit zones and the likelihood of the transition. The advantage that topographic calibration has over geometric calibration is that it is capable of automatically creating and updating the relationships of a camera network, especially for cameras with non-overlapping views. In (Black *et al*., 2005), camera topography based method is proved to be much more reliable than a 3D Kalman filter on object handover through blind regions. However, topographic calibration often requires significant training data and the performance can be further improved if appearance information is used.

Another issue is that topographic based methods often deal with multiple cameras with non-overlapping views, however, there is rarely this type of dataset available for public use. Because of the generally lack of benchmark dataset, most authors use their proprietary data and synthetic data as well which makes it too difficult to objectively evaluate and compare algorithms.

2.3.2.3 Colour calibration

Many multi-camera tracking systems assume a common colour response for all cameras. However, different cameras may have radically different colour responses and the difference can cause errors in object correspondence using colour. Hence,

colour calibration is needed to compensate for the change of colour across different cameras.

(Madden *et al.*, 2007) proposed a form of controlled equalization to the object colour histogram to compensate for illumination changes. (Prosser *et al.*, 2008) proposed a method that uses inter-camera brightness mapping function to adapt to new illumination conditions. However, their methods are only able to deal with illumination changes between cameras, not for changes of colour response. (Javed *et al*., 2005) proposed a method to use a Gaussian distribution to model the appearance change from one camera to another. They learned the usual change in colour histogram of a person that moved across camera views and used the leaned model for colour correspondence. However, a set of training data need to be chosen carefully and the assumption that all colours will change in a similar way is not always true. (Illie and Welch, 2005) suggest several methods (Linear Least Squares, RGB to RGB transform, General Polynomial Transform) to match the different colour responses of the adjusted cameras, providing a colour transformation from any camera colour space to any other. However, their method is not robust against lighting changes thus the system needs re-calibration when there are dramatic lighting changes.

2.4 Performance evaluation

In recent years, there has been an increased interest in performance evaluation of surveillance systems. The motivation for such work varies depending on the requirements. Public bodies, such as the UK Home Office (i-LIDS, nd), use high level metrics (e.g. F measure for event detection) to evaluate the performance of a surveillance system for commercial validation. Research communities, such as (Pets Metrics, nd), (ETISEO,nd), (CAVIAR, nd), etc, use one or more metrics to compare systems or algorithms. Individual researchers/research groups, evaluate their systems to identify deficits so they can further improve the performance of their algorithms.

A common approach of performance evaluation is summarized as follows: First of all, ground truth is generated from pre-recorded video. The ground truth

normally takes the form of the tracked object id and the object's bounding box. Then, the video tracking algorithm is applied to the video sequence. Finally, the ground truth and tracking results can be compared in order to get an indication of how well and bad the tracking system performances. Figure 2-5 shows a general framework of performance evaluation.

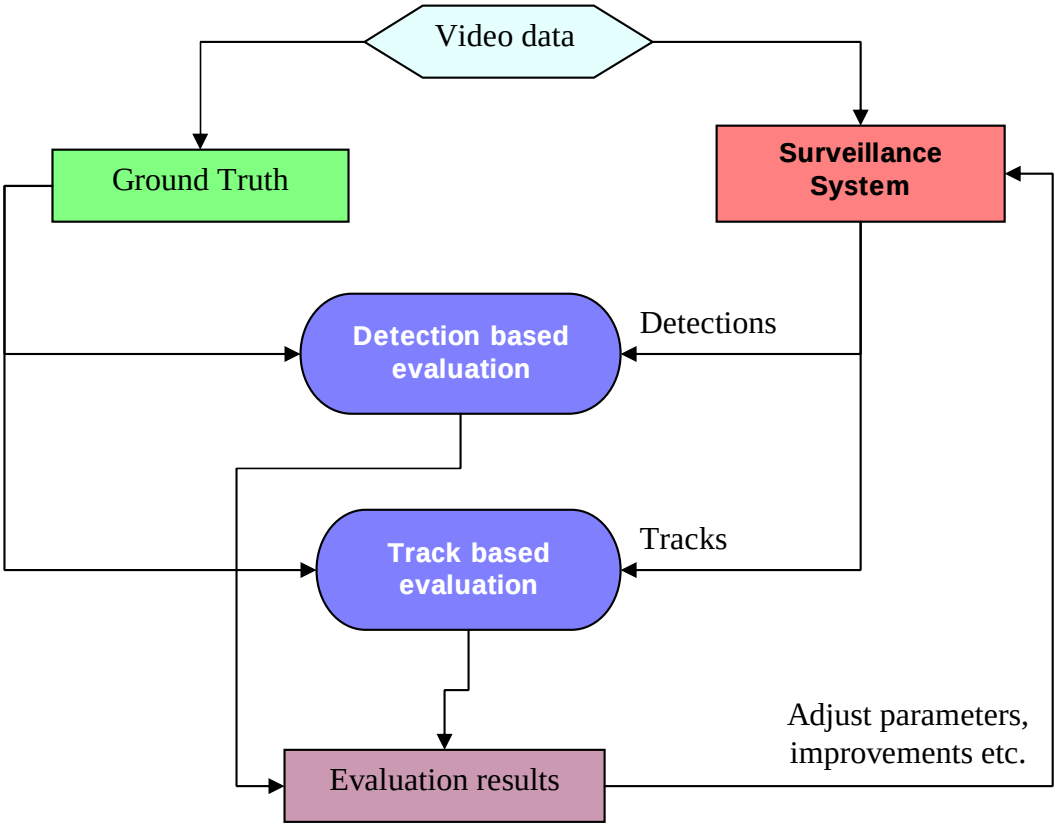


Figure 2-5 General framework of performance evaluation

Generating ground truth for video can be a time-consuming task, particularly for video sequences that contain a large number of objects. Therefore, a number of semiautomatic tools (e.g. VIPER-GT, ODViS) have been developed to speed up the process of ground truth generation. One of the most popular tools for ground truth generating is provided by (VIPER-GT, nd) and is also used in this work.

Numerous research works have been done to propose metrics for performance evaluation, both detection based and trajectory based: (Ellis, 2002), (Nascimento and Marques, 2005), (Lazarevic-McManus *et al.*, 2007), (Bashir and

Porikli, 2006), (Brown *et al.*, 2005). However, none of the previous evaluation framework is sufficient enough to give a comprehensive and quantitative measurement of the performance of tracking algorithms. Therefore, a more complete evaluation framework which can measure the performance of specific module (e.g. motion segmentation, tracking, data association etc.) of motion tracking algorithms is proposed in this thesis. More detailed review of literature on performance evaluation of tracking algorithms will be given in section 3.1, the chapter on performance evaluation.

2.5 Discussion

In this chapter, an overall introduction of framework for visual surveillance systems was presented. Computer vision techniques related to different aspects of visual surveillance which includes object detection, object tracking, multiple camera calibration and object correspondence between cameras have been discussed. Although classical approaches for background modelling and tracking have been successfully applied for many applications, problems still remain. Because of the large diversity of camera views and complexity of scenes and activities, there is a great potential to improve the existing works and introduce new applications.

In the next chapters, a few remaining problems of single camera tracking will be handled and new methods for multi-camera tracking and scene modelling will be proposed. Chapter 3 presents a track based performance evaluation framework for quantitatively evaluating and comparing object tracking algorithms. It is an essential step for validating a tracking system before it can be used for real world applications. Chapter 4 presents three new modules (ghost elimination, shadow removal, improved Kalman filter) that have been done to improve the performance of a single camera tracker. The track based evaluation framework is used to measure those improvements. Chapter 5 proposes an effective camera calibration method based on homography and a multi-camera object correspondence method based on vertical axis. Then, using the proposed methods, a whole tracking system is built to achieve more robust object tracking across

multiple cameras. Chapter 6 proposes a novel method to learn a non-coplanar scene model to extend the capability of current surveillance systems to a wider range of environments where multiple levels presents in the scene (e.g. stairs, overpasses).

3 Performance Evaluation

Effectively evaluating the performance of algorithms for detection and tracking of moving objects is an important step to achieve and demonstrate robust digital video surveillance systems with sufficient accuracy for practical applications.

A rich set of track based metrics is proposed in this work in order to evaluate the performance of object tracking systems.

The results of performance evaluation can provide us with quantitative measurements of motion detection and tracking performance of the trackers. Then, it will be easier for us to identify the limitations of the system, and furthermore, to make improvements to the tracker and measure the subsequent improvements.

3.1 Background

In the last two decades, researchers and industry have shown a growing interest for object tracking systems. Performance evaluation has played an important role on developing, assessing and comparing motion tracking algorithms. However, performance evaluation has different meanings and usages to different categories of people. End-users and public bodies are interested in assessing systems for validation and standardisation and therefore are interested in measuring high-level metrics of performance, as specified by end-users. For example, the (i-LIDS, nd) evaluation programme, developed by the UK Home Office focuses on measuring the accuracy of detection of high-level events such as “vehicle illegal parking”, “sterile zone intrusion”, “abandoned bag” and “door entering and exiting”. On the other hand the research community, as expressed by workshops (e.g. PETS), projects (e.g. ETISEO, CAVIAR) and the peer-reviewed publication process, aims to compare algorithms and systems in order to identify state-of-the-art techniques. Individual researchers and practitioners, when they work to develop and improve their systems, are very interested in identifying which modules of the tracking system fail. This work aims to address this issue and proposes a framework that estimates the potential reasons of failure.

(Ellis, 2002) investigated the main requirements for effective performance analysis for surveillance systems and proposed some methods for characterising video datasets. (Nascimento and Marques, 2005) (CAVIAR) proposed a framework which compares the output of different motion detection algorithms against given ground truth and estimates objective metrics such as false alarms, detection failures, merges and splits. (Lazarevic-McManus *et al*., 2007) evaluated performance of motion detection based on ROC-like curves and the F-measure. The latter allows straight-forward comparison using a single value that takes into account the application domain.

While the above work mainly deals with evaluation of motion detection, other researchers attempt to deal with the evaluation of both motion detection and object tracking. (Needham and Boyle, 2003) proposed a set of metrics and statistics for comparing trajectories to account for detection lag, or constant spatial shift. However, they take only trajectories (sequences of points over time) as the input of evaluation and therefore their approach may not give sufficient information on the precision of the object size estimation and spatial extent over time. (Bashir and Porikli, 2006) gave definitions of evaluation metrics based on the spatial overlap of ground truth and system bounding boxes that are not biased towards large objects. However they are counted in terms of frame samples. Such an approach is justified when the objective of performance evaluation is object detection. In object tracking, counting TP, FP and FN tracks is a more natural choice that is consistent to the expectations of the end-users. (Brown *et al*., 2005) suggests a framework for matching ground truth tracks and system tracks and computing performance metrics. However their definition, based on the comparison of the system track centroid and an enlarged ground truth bounding box which favours tracks of large objects. Although such an approach is useful for evaluating the system's performance, it does not provide a clue about the source of potential system failures. (Nghiem *et al*., 2007) (ETISEO) proposed a large set of metrics that can address each object detection and tracking problem separately, and could be used for comparison between algorithms. However, they do not provide a rigorous mathematical definition for each of those metrics.

Therefore, in this work, a rich set of metrics is proposed to reflect aspects of different modules of tracking systems (such as motion segmentation, motion tracking and data association) and identify specific failures of motion tracking. The approach is illustrated through a number of sequences which represent a wide variety of challenges for tracking systems.

3.2 Track based evaluation Metrics

3.2.1 Introduction

Before the track based metrics are introduced, it is important to discuss the concept of motion detection and tracking, define the output of a tracker and the concept of track overlap.

In this work, tracking is defined as the problem of estimating the spatial extent of the foreground objects for each frame of a video sequence. Given an image sequence from a static camera, the first step is to separate moving foreground objects from the background, often called foreground segmentation. Foreground segmentation is performed by comparing the pixel values of the current frame against a background model (e.g. GMM): pixels with values sufficiently different from the background distribution are labelled as foreground. Then, the detected foreground pixels are grouped together to form foreground blobs or objects using connected component analysis.

The second task is to link a sequence of the same foreground object across image frames in order to determine the identity and location of the object at given time. Normally, a foreground object is described by a set of attributes (e.g. its id, position and velocity). Kalman filter is used to perform tracking and keep temporal and spatial consistency for each object. An illustration of a tracking system is shown in Figure 3-1.

The result of tracking is a set of tracks for all foreground objects T_j. A track is defined as $T_j=\{ [x^{min}_{j,k}, x^{max}_{j,k}, y^{min}_{j,k}, y^{max}_{j,k}], V_{j,k} \}, k=1..N$, where $[x^{min}_{j,k}, x^{max}_{j,k}, y^{min}_{j,k}, y^{max}_{j,k}]$ defines the image spatial extent (usually called the bounding box) and $V_{j,k}$ is the velocity of track j at frame k.

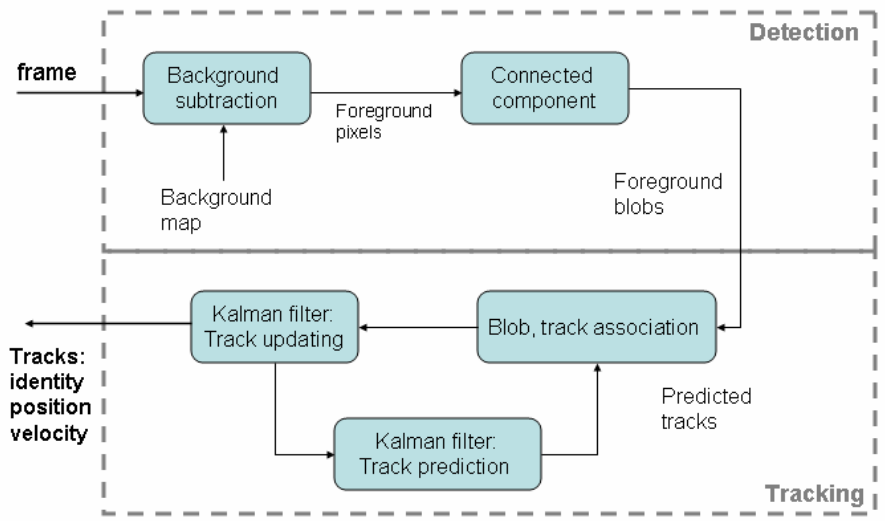


Figure 3-1 Block diagram of motion tracking system

Before the evaluation metrics are introduced, the concepts of spatial and temporal overlap between tracks are defined. The spatial and temporal overlap are required to quantify the level of matching between ground truth (G_i) tracks and system (S_j) tracks, both in space and time. The idea of spatial overlap proposed by (Nascimento and Marques, 2005) is adopted which is the bounding box overlapping $A(G_{ik}, S_{jk})$ between G_{ik} and S_{jk} tracks in a given frame k.

$$A(G_{ik}, S_{jk}) = \frac{Area(G_{ik} \mathbf{I} S_{jk})}{Area(G_{ik} \mathbf{U} S_{jk})} \qquad 3.1$$

A binary variable $O(G_{ik}, S_{jk})$ is also defined to make decision of weather the two bounding boxes are overlapped or not based on a threshold T_{ov}

$$O(G_{ik}, S_{jk}) = \begin{cases} 1 & if \ A(G_{ik}, S_{jk}) > T_{ov} \\ 0 & f \ A(G_{ik}, S_{jk}) < T_{ov} \end{cases} \qquad 3.2$$

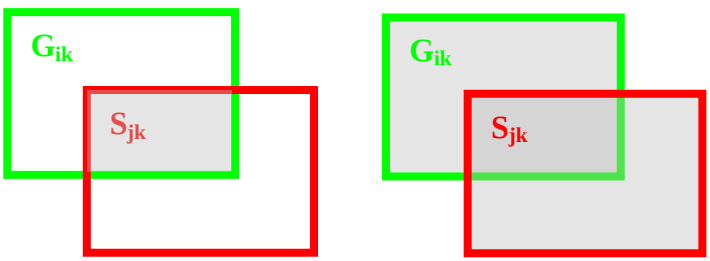


Figure 3-2 $G_{ik} \cup S_{jk}$(left) and $G_{ik} \cap S_{jk}$(right)

Frame based evaluation is used to measure how the overlapping threshold T_{ov} between detections and ground truth affects the values of correct detection, false alarm and missing rate. As we can see in Figure 3-3, for different values of T_{ov} over the range zero to one (tested on PETS2001 dataset, camera1), the correct detection rate and false alarm rate do not change significantly when the threshold is set between 10% and 40%, therefore, the threshold is set to 20% which is reasonable. Note that the rates shown in Figure 3-3 are computed according to the frame based metrics (correct detection rate and false alarm rate) in (Lazarevic-McManus *et al.*, 2007).

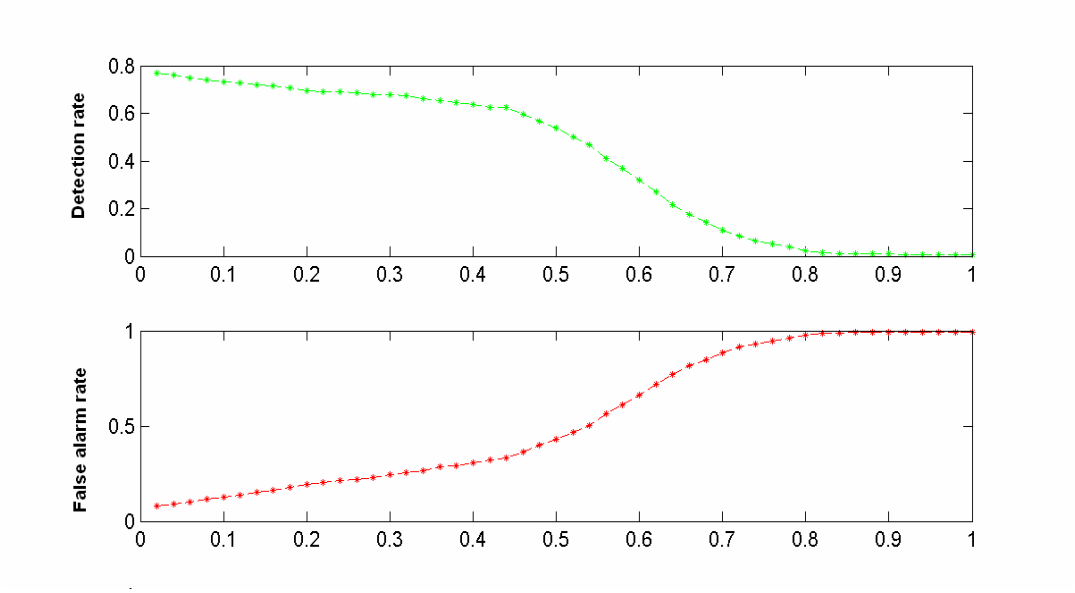


Figure 3-3 Detection rate and false alarm rate for different values of T_{ov}

In this work, the temporal overlap $TO(G_i, S_j)$ is defined as a number that indicates the overlap of frame span between system track j and GT track i :

$$TO(G_i, S_j) = \begin{cases} TO_E - TO_S & \text{if } TO_E > TO_S \\ 0 & \text{if } TO_E \le TO_S \end{cases} \qquad 3.3$$

where TO_S is the beginning frame of temporal overlap between system track j and GT track i , while TO_E is the ending frame of temporal overlap between the two tracks (see figure 3-4, two black points along the x-axis indicates the starting and ending frames of temporal overlapping between two tracks).

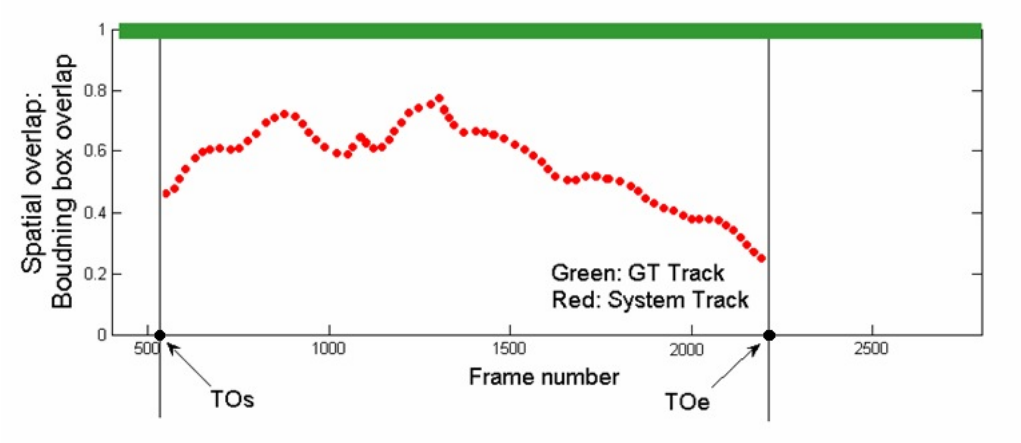


Figure 3-4 Example of temporal and spatial overlap between a GT track and a system track

A temporal-overlap criterion is used to associate system tracks to GT tracks according to the following condition in order to find candidates for GT and system track associations:

$$\frac{L\left(G_i \mathbf{I} S_j\right)}{L\left(G_i\right)} \geq TR_{ov} \qquad 3.4$$

where L(.) is the number of frames and TR_{ov} is an appropriate threshold (a range of thresholds for TR_{ov} is tested, and it is found out that, the choice of different threshold does not significantly affect the evaluation results. Hence, the value 15% is chosen in our work which is low and will not miss possible associations, for both trackers). If Eq.3.4 is true, then, all the track based metrics are computed for that pair of tracks to evaluate the performance of that system track. If more than one system track satisfies the condition of Eq.3.4, the GT track is still considered as one correct detected track, and the multiple associations will be reflected in the track fragmentation metric (Sec.3.2.5). On the other hand, if there are multiple GT tracks associated with one system track, the multiple associations will be reflected in the ID change metric (Sec.3.2.5). Therefore, multiple track associations are addressed by the track fragmentation and ID change metrics and they do not affect the correct detection, detection failure and false alarm metrics. Note that the same values of T_{ov} and TR_{ov} are used for the whole evaluation procedure to ensure a fair comparison between trackers.

3.2.2 Performance overview

In this section, high level metrics such as Correct Detected Track (CDT), Track Detection Failure (TDF) and False Alarm Track (FAT) are introduced to obtain an overall view of performance of the tracking system. (Brown *et al.*, 2005) also proposed similar metrics to count the number of track true positives, track false negatives and track false positives. But their definition of track spatial overlap was whether the centroid of a system track is inside an enlarged bounding box of a GT track which favours bigger bounding boxes.

Correct detected track (CDT) or True Positive (TP):

A GT track will be considered as been detected correctly, if it satisfies both of the following conditions:

Condition 1: The temporal overlap between GT track *i* and system track *j* is larger than a predefined track overlap threshold TR_{ov} (Eq. 3.4)

Condition 2: The system track *j* has sufficient spatial overlap with GT track *i*. (Eq.3.5)

$$\frac{\sum_{k=1}^{N} A\left(G_{ik}, S_{jk}\right)}{N} \geq T_{ov} \qquad 3.5$$

where N is the number of temporal overlapping frames between G_i and S_j. Each GT track is compared to all system tracks according to the conditions above. Even if there is more than one system track meets the conditions for one GT track (which is probably due to fragmentation), the GT track is still considered as one correct detection.

False alarm track (FAT) or False Positive (FP):

Although it is easy for human operators to realise what is a false alarm track (event) even in complex situation, it is hard for an automated system to do so. Here, a practical definition of false alarm track is given.

A system track will be considered as false alarm, if the system track meets any of the following conditions:

Condition 1: A system track *j* has temporal overlap smaller than TR_{ov} with any GT track *i*. (Eq.3.6)

$$\frac{L\left(G_i \mathbf{I} S_j\right)}{L\left(S_j\right)} < TR_{ov} \qquad 3.6$$

Condition 2: A system track *j* does not have sufficient spatial overlap with any GT track *i* although it has enough temporal overlap with GT track. (Eq.3.7)

$$\frac{\sum_{k=1}^{N} A\left(G_{ik}, S_{jk}\right)}{N} < T_{ov} \qquad 3.7$$

FAT is an important metric for end users, because they usually require it as low as possible to ensure that operators will not be overwhelmed by false alarms and therefore will not tend to ignore the system.

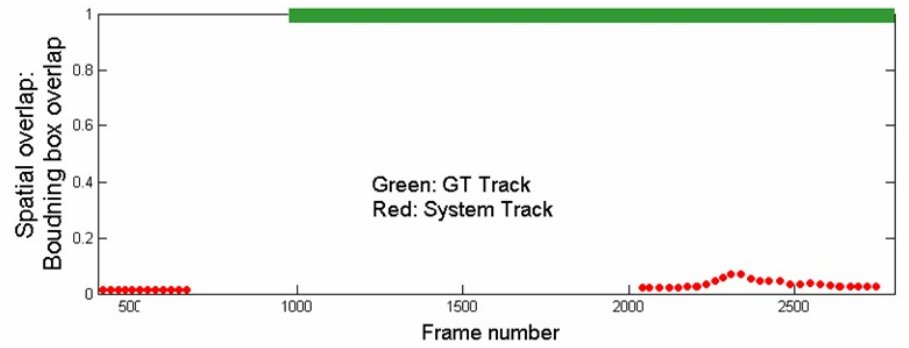


Figure 3-5 Example of false alarm tracks (red)

Track detection failure (TDF) or False Negative (FN):

A GT track will be considered as not detected (i.e. as a track detection failure), if it satisfies any of the following conditions.

Condition 1: A GT track *i* has temporal overlap smaller than TR_{ov} with any system track *j*. (Eq. 3.4 is false)

Condition 2: Although a GT track *i* has enough temporal overlap with system track *j*, it has insufficient spatial overlap with any system track according to (Eq.3.7)

3.2.3 Motion segmentation evaluation

The following metric, Closeness of Track is introduced to evaluate the performance of the motion segmentation module. If a system track is overlapped perfectly with a GT track through the temporal overlapping period, the Closeness of Track will be

100%. On the other hand, if a system track is not overlapped with a GT track at all, the Closeness of Track will be 0%. The proposed metric measures how accurate the localization of foreground objects is, in other words, it measures the spatial accuracy of the motion segmentation module.

Closeness of Track (CT)

The idea of frame based bounding box area matching proposed by (Nascimento and Marques, 2005) is adopted and extended to a track based metric in this work. For a pair of associated GT track and system track, the closeness of track is defined as a sequence of spatial overlaps for the period of temporal overlap:

$$\mathrm{a}_t = \{A(G_{i1}, S_{j1}), A(G_{i2}, S_{j2}).....A(G_{iN}, S_{jN})\} \quad 3.8$$

From Eq.3.8, the average closeness $\bar{\mathrm{a}}_t$ for that pair of GT and system tracks can be estimated. Suppose there are *M* pairs of tracks in total. The closeness for this video sequence is defined as a weighted average of the closeness of all *M* pairs of tracks:

$$\bar{A} = \frac{\sum_{t=1}^{M} L(\mathrm{a}_t) \cdot \bar{\mathrm{a}}_t}{\sum_{t=1}^{M} L(\mathrm{a}_t)} \quad 3.9$$

Where $\bar{\mathrm{a}}_t$ is the average closeness for the t^{th} pair of tracks. The weighted standard deviation of track closeness for the whole video sequence is defined as:

$$s_A = \frac{\sum_{t=1}^{M} L(\mathrm{a}_t) \cdot s_{a_t}}{\sum_{t=1}^{M} L(\mathrm{a}_t)} \quad 3.10$$

Where s_{a_t} is the standard deviation of CT for the t^{th} pair of tracks.

$$s_{a_t} = \sqrt{\frac{\left(A(G_{i1}, S_{j1}) - \bar{\mathrm{a}}_t\right)^2 + \left(A(G_{i2}, S_{j2}) - \bar{\mathrm{a}}_t\right)^2 + ...\left(A(G_{iN}, S_{jN}) - \bar{\mathrm{a}}_t\right)^2}{(N-1)}} \quad 3.11$$

3.2.4 Motion tracking evaluation

In this section, two metrics (Latency of the system track and Track Distance Error) are introduced to evaluate the performance of the motion tracking module. Latency

of the system track measures the delay of registering a track. Track distance error measures the error of points (centriod) tracking. For a perfect tracking system, the value of these two metrics should be zero.

Latency of the system track (LT):

Latency (time delay) of the system track is defined as the time gap between the time that an object starts to be tracked by the system and the first appearance of the object in the camera FOV. The optimal latency should be zero. A very large latency means the system may not be sensitive enough to trigger the tracking in time or indicates that the detection is not good enough to trigger the tracking.

Latency is measured by the difference in frames between the first frame of system track, $F(S_j)$ and the first frame of GT track, $F(G_i)$

$$l_i = F(S_j) - F(G_i) \tag{3.12}$$

If there are more than one system track associated with GT track i, the shortest latency will be chosen. Suppose the total number of GT tracks is I, the average latency for all the GT tracks in one video sequence is calculated as:

$$\overline{l} = \frac{\sum_{i=1}^{I} l_i}{I} \tag{3.13}$$

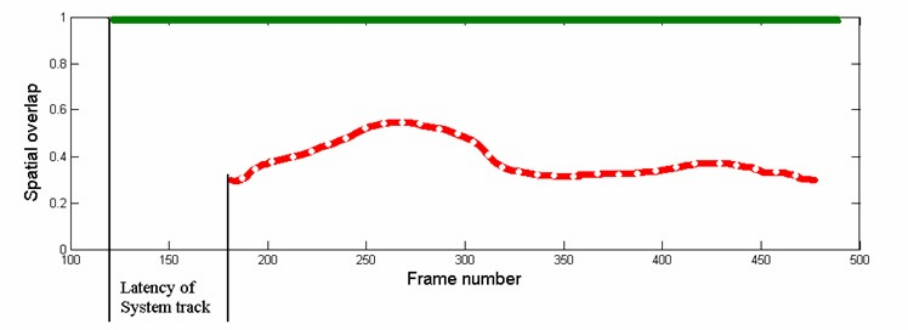


Figure 3-6 Example of latency

Track Distance Error (TDE):

This metric measures the positional error of system tracks and is adopted from the work by (Needham and Boyle, 2003). In Figure 3-7, *(x,y)* and *(p,q)* are the trajectory points (centroid of bounding boxes) for a ground truth track and a system track respectively.

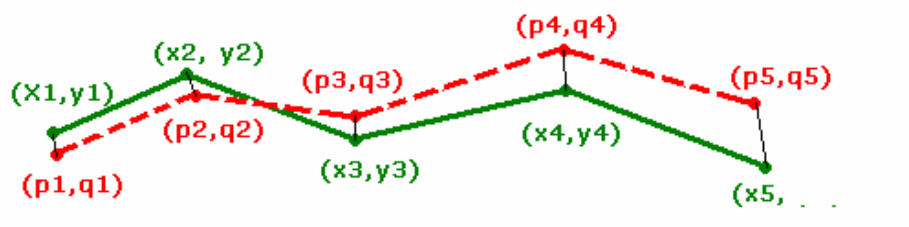


Figure 3-7 Example of a pair of trajectories

Track distance error $\overline{D}$ for the whole video sequence is defined as the weighted average with the duration of overlapping of each pair of tracks as the weight coefficient:

$$\overline{D} = \frac{\sum_{t=1}^{M} L(d_t) \cdot \overline{d_t}}{\sum_{t=1}^{M} L(d_t)} \qquad 3.14$$

Where $\overline{d}_t$ is the average distance error for the t^{th} pair of tracks. The standard deviation of track matching errors for the whole sequence is defined as:

$$s_D = \frac{\sum_{t=1}^{M} L(d_t) \cdot s_{dt}}{\sum_{t=1}^{M} L(d_t)} \qquad 3.15$$

Where s_{dt} is standard deviation of distance errors for the t^{th} pair of tracks.

3.2.5 Data association evaluation

Finally, three metrics (track fragmentation, ID change and track completeness) are used to quantify data association errors. These three metrics measure the continuity of single tracks (track fragmentation), whether the tracker is able to maintain the same ID (ID change) for an object and how complete the track is (track completeness). The value of these metrics indicates how good or bad the tracker at dealing with the inter frame matching between tracked objects (referred to as data association). (Nghiem *et al.*, 2007) also proposed similar metrics (ID persistence and ID confusion). However, in this work, more strict constraints are used for the metric ID change to deal with object intersection or occlusion and hence obtain more reliable evaluation result.

Track Fragmentation (TF):

Fragmentation indicates the lack of continuity of system track for a single GT track. In an optimal condition, track fragmentation error should be zero which means the tracking system is able to produce continuous and stable tracking for the ground truth objects. As mentioned in section 3.2.1, if there are multiple associations between GT track and system track, fragmentation will be measured from the track correspondence results.

$$TF = \sum_{i=1}^{I} TF_i \qquad 3.16$$

where TF_i is the number of system tracks that associated with GT track i and I is the total number of GT tracks (the condition for association is mentioned in Sec. 3.2.1).

Figure 3-8 Example of track fragmentation

ID Change (IDC):

The metric IDC_j is used to count the number of ID changes for system track j. Note that such a metric provides more elementary information than an ID swap metric.

For each frame k, the bounding box $D_{j,k}$ of the system track S_j may be overlapped with $N_{Dj,k}$ GT areas out of total I_k, where $N_{Dj,k}$ is given by:

$$N_{Dj,k} = \sum_{i=1}^{I_k} O\left(G_{ik}, D_{jk}\right) \qquad 3.17$$

where $O\left(G_{ik}, D_{jk}\right)$ is a binary value that indicates whether G_{ik} and D_{jk} are overlapped or not, same as Eq.3.2. Only the frames that $N_{Dj,k}$ =1 (which means that the track S_j is associated (spatially overlapped) with only one GT Track for each of these frames) are taken into account. Those frames are used to estimate the ID changes of S_j as the number of changes of associated GT tracks. The total number of ID changes in a video sequences is estimated as:

$$IDC = \sum_{j=1}^{J} IDC_j \qquad 3.18$$

Where J is the total number of system tracks.

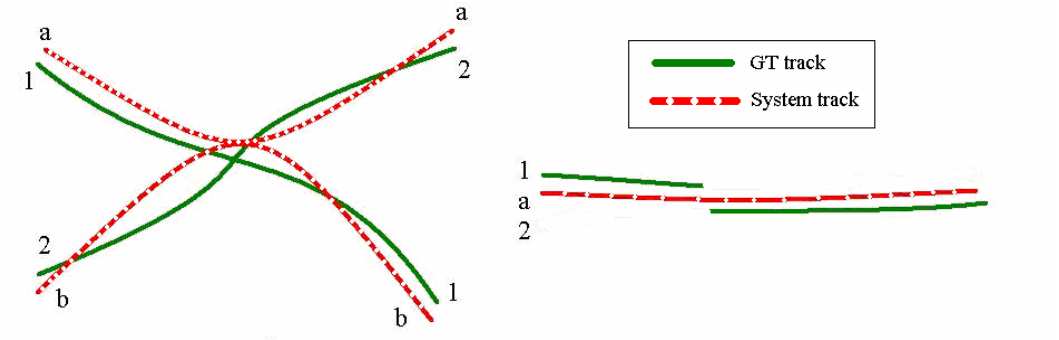


Figure 3-9 Example of ID changes (left: two IDC right: one IDC)

Track Completeness (TC):

The time span that the system track overlapped with GT track divided by the total time span of GT track. A fully complete track is where this value is 100%.

$$c_{ij} = \frac{\sum_{k=1}^{N} O\left(G_{ik}, S_{jk}\right)}{L\left(G_i\right)} \qquad 3.19$$

If there is more than one system track associated with the GT track, then the maximum completeness for each GT track will be chosen. Also, the average track completeness of a whole sequence is defined as:

$$\overline{C} = \frac{\sum_{i=1}^{I} \max(c_i)}{I} \qquad 3.20$$

Where I is the total number of GT tracks and $\max(c_i)$ is the maximum completeness for GT track i and the standard deviation of track completeness for the whole sequence is defined as:

$$s_c = \sqrt{\frac{\sum_{i=1}^{I}\left(c_i - \overline{C}\right)}{I-1}} \qquad 3.21$$

3.3 Evaluation Results

After all the evaluation metrics are introduced, the practical value of the proposed metrics is demonstrated by evaluating two motion tracking systems: an

experimental industrial tracker (the BARCO tracker) and the OpenCV 1.0 blobtracker (oepnCV, nd). The openCV tracker mainly includes: the adaptive mixture of Gaussian models for background estimation, connected component analysis for data association and then, a Kalman filter for tracking foreground blobs (see parameters in Appendix A). The trackers are tested on six video sequences (shown from Figure 3-10 to Figure 3-15) that represent a variety of challenges, such as illumination changes, shadows, snow storm, quick moving objects, blurring of FOV, slow moving objects, mirror image of objects and multiple object intersections. The parameters of each tracker are kept constant (parameters are not tuned specifically for each video sequence) throughout the whole test which reflects their performance in real world conditions. The ground truth for all video sequences was manually generated by the author using (Viper GT, nd).

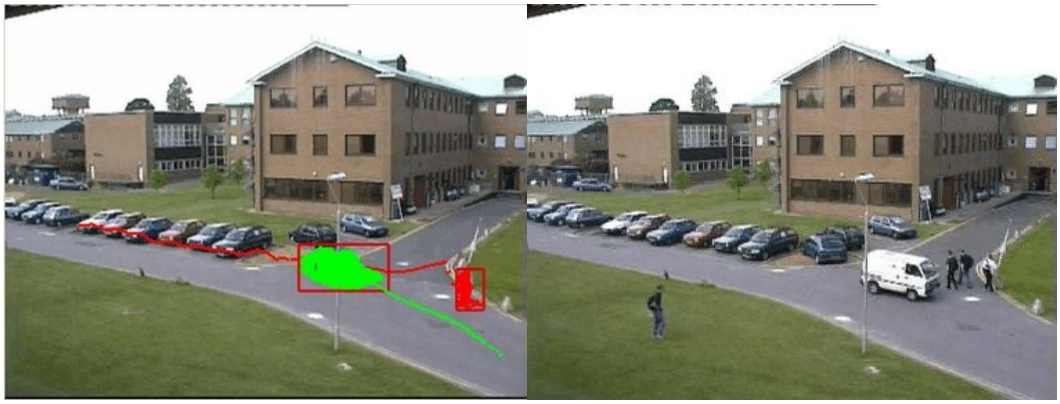


Figure 3-10 PETS2001 PetsD1TeC1.avi sequence is 2686 frames (00:01:29) long and depicts 4 persons, 2 groups of persons and 3 vehicles: Its main challenge is the multiple object intersections. Tracking example (left) and frame example (right)

Figure 3-11 i-LIDS SZTRA103b15.mov sequence is 5821 frames (00:03:52) long and depicts 1 person. Its main challenges are the illuminations changes and a quick moving object

Figure 3-12 i-LIDS SZTRA104a02.mov sequence is 4299 frames (00:02:52) long and depicts one person

Figure 3-13 i-LIDS PVTRA301b04.mov sequence is 7309 frames (00:04:52) long and depicts 12 persons and 90 vehicles. Its main challenges are shadows, moving object in the beginning of sequence and multiple object intersections

Figure 3-14 BARCO 060306_04_Parkingstab.avi is 7001 frames long and depicts 3 pedestrians and 1 vehicle. Its main challenge is the quick illumination changes

Figure 3-15 BARCO 060306_02_Snowdivx.avi is 8001 frames long and depicts 3 pedestrians. Its main challenges are snow storm, blurring of FOV, slow moving objects and mirror image of objects

The results of performance evaluation for both trackers are presented from Tables 3-1 to 3-6.

PetsD1TeC1.avi	BARCO tracker	OpenCV tracker
Number of GT Tracks	9	9
Number of System Tracks	12	17
Correct Detected Track	9	9
False Alarm Track	3	6
Track Detection Failure	0	0
Track Fragmentation	3	3
ID Change	5	7
Latency of Track	46	66
Average Track Closeness	0.47	0.44
Deviation of Track Closeness	0.24	0.14
Average Distance Error	15.75	5.79
Deviation of Distance Error	23.64	5.27
Average Track Completeness	0.67	0.58
Deviation of Track Completeness	0.24	0.89

Table 3-1 Evaluation results for PETS2001 Sequence

SZTRA103b15.mov	BARCO tracker	OpenCV tracker
Number of GT Tracks	1	1
Number of System Tracks	8	15
Correct Detected Track	1	1
False Alarm Track	3	12
Track Detection Failure	0	1
Track Fragmentation	2	0
ID Change	0	0
Latency of Track	50	9
Average Track Closeness	0.65	0.23
Deviation of Track Closeness	0.21	0.10
Average Distance Error	9.10	15.05
Deviation of Distance Error	12.48	3.04
Average Track Completeness	0.68	0.42
Deviation of Track Completeness	0.00	0.00

Table 3-2 Evaluation results for i-LIDS SZTRA103b15

SZTRA104a02.mov	BARCO tracker	OpenCV tracker
Number of GT Tracks	1	1
Number of System Tracks	4	9
Correct Detected Track	1	1
False Alarm Track	0	5
Track Detection Failure	0	0
Track Fragmentation	2	2
ID Change	0	0
Latency of Track	74	32
Average Track Closeness	0.79	0.34
Deviation of Track Closeness	0.21	0.17
Average Distance Error	7.02	16.69
Deviation of Distance Error	15.67	7.55
Average Track Completeness	0.73	0.44
Deviation of Track Completeness	0.00	0.00

Table 3-3 Evaluation results for i-LIDS SZTRA104a02

PVTRA301b04 .mov	BARCO tracker	OpenCV tracker
Number of GT Tracks	102	102
Number of System Tracks	225	362
Correct Detected Track	90	95
False Alarm Track	67	112
Track Detection Failure	12	7
Track Fragmentation	62	98
ID Change	95	101
Latency of Track	57	78
Average Track Closeness	0.30	0.22
Deviation of Track Closeness	0.21	0.16
Average Distance Error	49.70	24.65
Deviation of Distance Error	60.31	22.85
Average Track Completeness	0.34	0.26
Deviation of Track Completeness	0.57	0.65

Table 3-4 Evaluation results for i-LIDS PVTRA301b04

Parkingstab.avi	**BARCO tracker**	**OpenCV tracker**
Number of GT Tracks	4	4
Number of System Tracks	9	17
Correct Detected Track	4	4
False Alarm Track	1	11
Track Detection Failure	0	0
Track Fragmentation	0	0
ID Change	0	0
Latency of Track	72	35
Average Track Closeness	0.50	0.39
Deviation of Track Closeness	0.20	0.14
Average Distance Error	13.32	11.82
Deviation of Distance Error	11.55	8.16
Average Track Completeness	0.82	0.77
Deviation of Track Completeness	0.11	0.96

Table 3-5 Evaluation results for Parkingstab

Snowdivx.avi	**BARCO tracker**	**OpenCV tracker**
Number of GT Tracks	3	3
Number of System Tracks	28	29
Correct Detected Track	3	3
False Alarm Track	19	20
Track Detection Failure	0	0
Track Fragmentation	2	5
ID Change	0	0
Latency of Track	590	222
Average Track Closeness	0.14	0.42
Deviation of Track Closeness	0.23	0.12
Average Distance Error	28.50	16.69
Deviation of Distance Error	35.44	11.62
Average Track Completeness	0.33	0.35
Deviation of Track Completeness	0.47	0.71

Table 3-6 Evaluation results for Snowdivx

From the results provided by the tables above, one can note that the overall performance of the BARCO tracker is better than the OpenCV tracker, because the BARCO tracker has higher number of correct detected tracks, and lower number of

false alarm tracks and track detection failures. One can also figure out the weakness of trackers against specific challenges. For instance, in video sequences SZTRA104a02 (Table 3-3) and Parkingstab (Table 3-5) both with significant illumination variations, very few false alarms were generated by the BARCO tracker, but quite a lot by the OpenCV tracker. Therefore, the BARCO tracker seems to be more robust against illumination changes than the OpenCV tracker.

Regarding the motion segmentation module, the BARCO tracker performs better than the OpenCV one, as reflected by the track closeness metric in most of the sequences. However, in snowy conditions (e.g. the Snowdivx sequence, see Table 3-6), the OpenCV motion segmentation module performs better.

The motion tracking module of the OpenCV tracker seems to overcome the above disadvantage and performs better, as it produces lower Average Distance Error for similar or slightly worse track completeness. Also, it responds quicker than the BARCO tracker as it can be seen from its smaller track latency.

The data association module of the BARCO tracker performs slightly better than the OpenCV tracker, since it has lower number of track fragmentations and ID changes and higher track completeness. For example, in the PETS2001 sequence (Table 3-1), the intersections of multiple objects, cause a few ID changes for both trackers which indicates that their data association modules need to be improved.

Generally, if any changes have been made to a specific module of a tracker (e.g. motion segmentation, tracking and data association), the proposed evaluation framework should be able to reflect these changes of performance in the evaluation metrics related to this module. To demonstrate that, for each module of the OpenCV 1.0 blobtracker (OpenCV, nd), different parameters (see parameters in Appendix A) are tested using the Pets2001 sequence. Some quantitative evaluation results are shown in Table 3-7.

First of all, the OpenCV blobtracker is tested by a set of parameters that can achieve its best performance (referred to as Baseline in the second column). Then, for each module, parameters are changed away from its best value. For instance, for motion tracking module, no post-processing filter for tracking is chosen instead of Kalman filter (in the third column). One should expect some performance drawbacks on track closeness and distance error as the Kalman filter has the

function of smoothing trajectories and filtering out noises. Without Kalman filter, it is also possible to lose a track and regain it later which can lead to track temporal fragmentation. In the forth column, single Gaussian is used rather than GMM, one should expect more segmentation noises which could lead to more false alarms, track spacial fragmentations, larger track distance error etc. In the last column, simple connected components is used rather than CCMSPF for inter-frame data association, one should expect performance drawbacks on the data association module (e.g. more track fragmentations, ID changes).

Although, changes in one module may cause problems in other modules (e.g. segmentation errors can lead to failure of tracking), the overall evaluation results are within the expectation and agree with the discussions above. Thus, a conclusion can be made that the proposed evaluation metrics are able to reflect the performance change of each module of the tracker which will allow us to reason about failures of particular modules and furthermore, to measure the improvements made to those modules.

PetsD1TeC1.avi/OpenCV tracker	Baseline	noKalman	SingleG	simpleCC
Number of GT Tracks	9	9	9	9
Number of System Tracks	17	22	37	23
Correct Detected Track	9	9	9	8
False Alarm Track	0	0	19	0
Track Detection Failure	0	0	0	1
Track Fragmentation	3	6	5	8
ID Change	1	1	1	3
Latency of Track	72	72	69	73
Average Track Closeness	0.44	0.40	0.32	0.37
Deviation of Track Closeness	0.13	0.17	0.18	0.17
Average Distance Error	6.11	15.08	12.58	17.56
Deviation of Distance Error	5.42	33.92	28.36	38.64
Average Track Completeness	0.63	0.59	0.57	0.49

Table 3-7 Evaluation results for different parameters of OpenCV blobtracker

3.4 Summary

In this chapter, a rich set of track based metrics has been presented to measure the performance of specific module of motion tracking algorithms. Metrics, such as Correct Detected Track (CDT), False Alarm Track (FAT) and Track Detection Failure (TDF) provide a general overview of the algorithm performance. Closeness of Track (CT) metric indicates the spatial extent of the objects and it is closely related to the motion segmentation module of the tracker. Metrics, such as Track distance Error (TDE) and Latency of Track (LT) indicate the accuracy of estimating the position and how quick the tracker responses respectively and they are related to the motion tracking module of the tracker. Track Fragmentation (TF) show whether the temporal or spatial coherence of tracks is established. ID Change (IDC) and Track Completeness (TC) are useful to test the data association module of multi-target trackers.

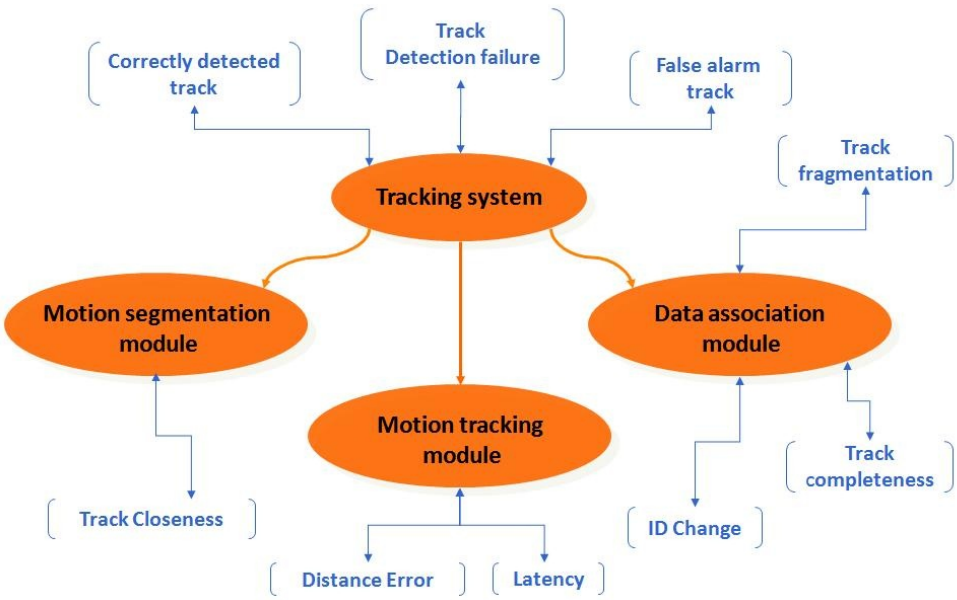


Figure 3-16 Performance evaluation framework

Six video sequences have been used to test two state of the art motion trackers. Those test sequences contain more than 30,000 frames which provide a variety of challenges, such as illumination changes, shadows, snow storm, quick moving objects, blurring of FOV, slow moving objects, mirror image of objects and multiple object intersections. The variety of metrics and datasets allows us to reason about the weaknesses of particular modules of the trackers against specific

challenges, assuming orthogonality of modules and challenges. Performances of different trackers are compared against each other as well. This approach is a realistic way to understand the drawbacks of motion trackers, which is important for improving them.

In the next chapter, a few new modules will be added to improve the performance of the BARCO tracker and change of performance on different aspects of the BARCO tracker will be observed using the proposed evaluation framework.

4 Single Camera Tracking

In the previous chapter, a track based evaluation frame work has been proposed. Using the proposed evaluation framework, strengths and weaknesses of object tracking systems are identified. Furthermore, it allows the developer to reason about the weaknesses of particular modules of the trackers against specific challenges and to improve the tracker.

In this chapter, three extensions or modules (Ghost elimination, shadow removal and improved Kalman filter) are added to the BARCO tracker mentioned in chapter 3. To demonstrate the improvements that have been done to the tracker, the tracking evaluation framework proposed in the previous chapter is used to (re)evaluate and compare the performance of BARCO tracker with and without those new modules. This is important as sometimes improvements may be little more than anecdotal and also improvements in one area may result in deterioration in other areas. Such effects might not be picked up by simple manual inspection without the proposed evaluation framework.

4.1 Introduction

For motion segmentation based tracking algorithms, detection of moving objects is the key stage that ultimately limits successful tracking in whatever application such as traffic monitoring and analysis, access control in special areas, human and vehicle identification, and detection of anomalous behaviours.

One of the most common and well established approaches for detecting moving objects is background subtraction, in which each frame is compared against a background model. The pixels in the current frame that have significant difference from the background are considered as moving pixels or foreground pixels. These foreground pixels will be grouped to form an object and then be tracked.

A large number of background subtraction algorithms have been proposed so far, but problems still remain for identifying moving objects identification under certain conditions. For example, one tough problem is that background subtraction causes the spurious moving object (often referred to as “ghost”) when an object that belongs to the background starts to move away, It is important to address the problem because ghost objects will adversely affect many tasks such as object classification, tracking and event analysis (e.g. abandoned item detection). Therefore, a ghost object needs to be separated from real objects and eliminated.

Another issue is caused by moving cast shadows. For most background subtraction methods, all the moving points of both objects and shadows are detected at the same time because shadow points and object points share two important features: motion model and detectability. In addition, shadow points are usually adjacent to object points so they are merged into a single blob. This will significantly affect the accuracy of object geometrical properties (e.g. shape, centroid, area) and may cause a false adjacency of object (wrong merge of objects) that will affect higher level tasks such object counting and classification. Therefore, shadow points needs to be separated from object points to avoid the errors caused by shadows.

This chapter deals with the problems of ghosts and moving cast shadows. New modules will be added to the BARCO tracker to identify and remove ghosts and shadow points, also, another new module will be added to improve the accuracy of object tracking (improved Kalman filter). Finally, the BARCO tracker with and without those three modules will be systematically evaluated and compared.

4.2 Background

Generally, a robust background model should be able to automatically recover and update itself from a dynamic sequence and be insensitive to illumination changes, shadows, weather conditions such as rain or snow. In chapter 2, a variety of background modelling and motion detection methods has been presented (work

related to shadow detection and Kalman filter is discussed in chapter 2 in section 2.2.1.4 and 2.2.2.1).

However, there are still some drawbacks of using a background model. One problem is that, to account for changing illumination conditions, objects that become stationary are relatively quickly incorporated into the background and when they move again they leave behind an area of foreground (or "ghost") that can be mistakenly taken as a new object. So the ghost problem is a direct result of background modelling methods. An approach that would feedback high level object detection/tracking information to the background model might be able to reduce this effects, for example by enforcing constraints such as that slow or stopped objects cannot simply disappear (merged into background).

(Durucan *et al.*, 1999) proposed a method which combines an illumination change detector with an optical flow method to deal with background effects such as randomly moving objects (plants, curtains), reflections and moving shadows. However, as a basic detection method, it is not designed to detect apparent moving objects (e.g. ghosts) and a contour detection method with a filling algorithm is still needed to be able to form complete moving blobs. (Cucchiara *et al.*, 2003) proposed a general frame work of background subtraction called Sakbot which combines statistical assumptions to detect moving objects, apparent objects (ghosts), and shadows with the object level knowledge of those from previous frames. For Ghost detection, they calculate the average optical flow, over all the detected moving pixels. They assume that moving objects should have significant motion, while ghosts should have a near-to-zero average optical flow since their motion is only apparent. However, optical flow is computational expensive for real time processing and also there is a danger of inaccurate classification of stationary objects as ghosts. (Cheung and Kamath, 2005), proposed a different method to detect a ghost in traffic surveillance. They used a frame difference mask which was computed using the current frame and its previous frame. Objects with no corresponding blob in the frame difference mask were identified as ghosts. However, the method cannot distinguish between ghosts and abandoned objects. In (Guler *et al.*, 2007)'s method, a specific processing layer is used for detection of objects that become stationary. The detected stationary foreground objects are

maintained in the specific layer, and the motion history of the immobile objects are recorded. The main drawback of this method is that the method needs motion history of the object which is not available if the object belonged to background from the start of the video sequence. (Lu *et al*., 2007) proposed to recover the "real" background by using a revised image inpainting method. Thus they could differentiate between ghosts and abandoned objects: The object will be declared as ghost if there are no difference between current frame and the "real" background image, otherwise it is an abandoned item. However, image inpainting is computational expensive for real time processing, and their method has not been evaluated quantitatively.

In this chapter, a time efficient ghost removal method is proposed to identify ghosts on object level. The proposed method is able to classify foreground objects as real objects (whether moving or stationary) or ghosts by comparing edges between extracted track regions from multiple images. The method is discussed and evaluated in the sections that follow.

4.3 Methodology

The BARCO tracker uses background estimation to extract motion blobs from a video frame. Then those blobs are used to form objects or tracks by connected components analysis. Afterwards, those formed objects are tracked using Kalman filter.

For the background estimation part, a shadow detection module is added to remove shadow points. For the object tracking part, a ghost detection module and improved Kalman filter are added to eliminate ghosts and improve the accuracy of object's localization during tracking. Each module is described in detail later in this section.

4.3.1 Ghost elimination

Ghosts mainly appear in two cases: In the first case, when a moving object becomes stationary, it will be adapted (merged) into the background, and then, when it starts to move again sometime later, there will be a ghost left behind. In the

second case, an existing object that belongs to the background starts to move (e.g a parked vehicle) and will also cause a ghost problem. In the former case, the history of object trajectory is available, but in the latter case, there will be no previous trajectory information about the object. So it is preferable to make the tracker able to recognise ghosts in time without additional history information rather than analyse the motion history of the object. In this work, a motion tracker is available with the following assumptions:

- Tracking is performed on pixel level and no calibration information is needed.
- The motion tracker has a background subtraction module that can provide a difference map which is the difference of the values of each pixel between the background image and the current (incoming) frame.
- The motion tracker is able to estimate the bounding box and the velocity of moving objects at each frame.

The proposed framework of ghost identification and elimination is summarized as follows (Fig 4-1):

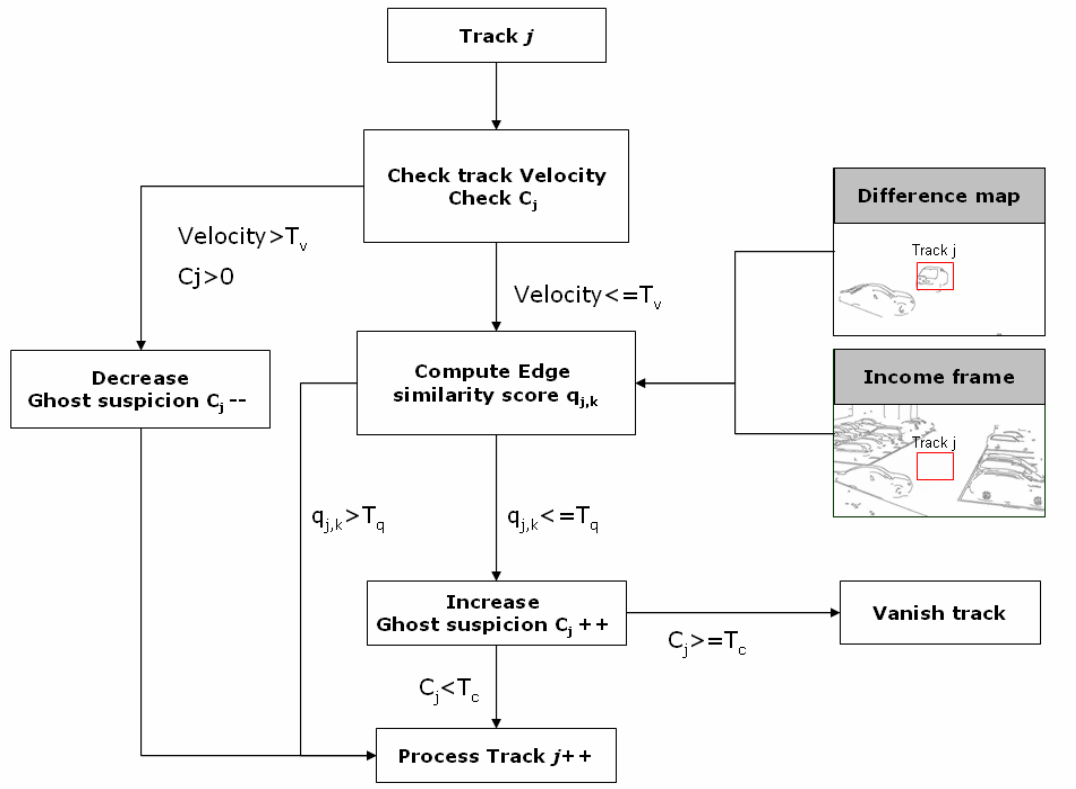


Figure 4-1 Ghost detection framework

- For every k frames (k =12 in this implementation), the speed (pixel based) of each track is checked. If the track's speed is below a threshold T_v (e.g. T_v= 0.1 pixel/frame), it will be proceed further to the ghost detection. This implements the observation that ghosting shows itself as a stationary or slow-moving object.
- Canny edge detection are applied within regions of tracks (bounding boxes) on both the difference map (difference between the background and the current frame) and the current frame. Hence, two edge maps are obtained for every track for the same frame (see Figure 4-2).

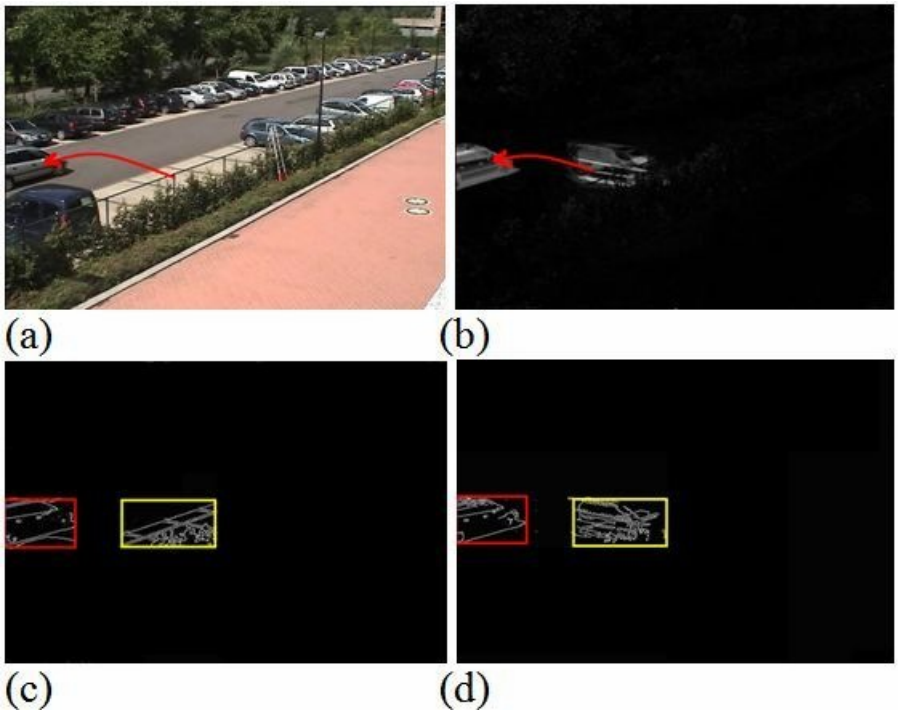


Figure 4-2 KND2-JULY-EAST-S2.avi sequence frame 5880 a) current frame: a car starts to move out of its parked area along the red arrow b) difference map: the detected moving area by background subtraction c) canny edge detection of "(a)" in track regions d) canny edge detection of "(b)" in track regions Note that red and yellow boxes indicate the regions of two different tracks

- Then, the differences between the edges from the two images (current frame and the frame difference map) within the area of the bounding box for each track are compared using a edge similarity score respectively (e.g. in Figure 4-2, the edges in the yellow bounding box in (c) and the yellow bounding box in (d) are compared). The edge similarity which is defined in

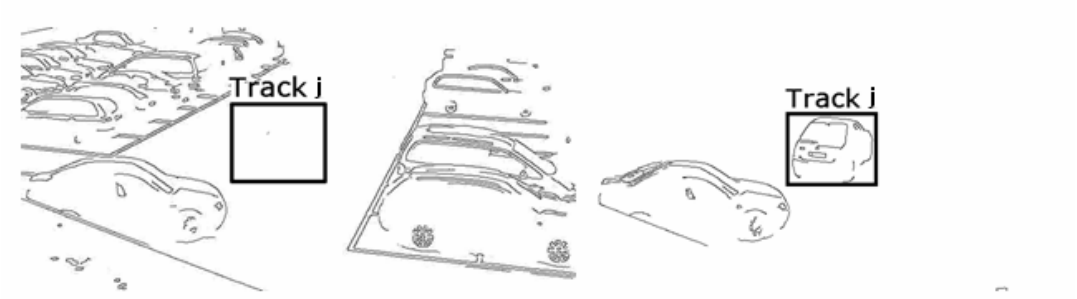
Eq.4.3 is calculated, and then a threshold is set up for making the decision of whether the edges have enough similarity or not (usually, score is from 0% (do not match at all) to 100% (fully matched).

Suppose for system track *j*, frame *k*, a set of edge points from the difference map is obtained:

$$S_{j,k}^{D} = \left\{ (x_1^D, y_1^D), (x_2^D, y_2^D) ... (x_n^D, y_n^D) \right\} \quad 4.1$$

and another set of edge points from the current frame for track *j* is obtained as well (see Figure 4-3):

$$S_{j,k}^{C} = \left\{ (x_1^C, y_1^C), (x_2^C, y_2^C) ... (x_m^C, y_m^C) \right\} \quad 4.2$$



(a) (b)

Figure 4-3 Edges of current image (a) and Edges of difference map (b)

The similarity score between the two edge sets is calculated as follows:

$$q_{j,k} = \frac{S_{j,k}^{D} \mathbf{I} S_{j,k}^{C}}{S_{j,k}^{D} \mathbf{U} S_{j,k}^{C}} \quad q_{j,k} \in [0 \sim 1] \quad 4.3$$

- Normally, similarity scores are very high when the tracks are from real existing objects (typically close to 1). If $q_{j,k}$ is smaller than a threshold T_q (set to 30% according to experiments), a ghost condition has been detected for track *j* for the current frame.
- Evidence is accumulated over time using a counter C_j which indicates how many times the track *j* has been detected as ghost. The initial value for C_j is zero.

If $q_{j,k} > T_q,\ C_j \leftarrow C_j + 1$ 4.4

As long as $C_j > 0$, we label the track as "ghost", the track is prevented from being "valid" or "real moving object" and does not trigger a false alarm. After several times of ghost suspicion (using the criterion of Eq.4.4), if C_j becomes larger than a predefined number T_c (set to 3 in this work), then the track will be confirmed as "ghost" and vanished (e.g. feedback the confirmation to the motion detector so that the motion detector enforces the ghost blobs to become part of the background).

- In order to make sure that a "real" moving object is not classified as ghost and vanished by mistake, the motion of the track is also taken into account. Let (x_j^O, y_j^O) be the coordinates of the centroid of track *j* where it appears firstly. $(x_{j,k}, y_{j,k})$ be the coordinates of the centroid of track *j* in current frame *k*. (W_j, H_j) be the average width and height of track *j* during its life time. Mv_j is the normalized distance that the object has moved from where it originally appeared and is defined as:

$$mv_j = \frac{\left(x_{j,k} - x_j^O\right)^2 + \left(y_{j,k} - y_j^O\right)^2}{W_j^2 + H_j^2} \quad 4.5$$

$$if \begin{matrix} mv_j >= 1 \\ C_j > 0 \end{matrix} \Bigg\} \ C_j \leftarrow C_j - 1 \quad 4.6$$

which means if the track has moved a long enough distance from where it originally appeared, then its ghost suspicion will decrease as the counter C_j will be decremented to zero gradually. As a result, it will not be identified as "ghost".

4.3.2 Shadow removal

In motion detection algorithms, shadows can cause serious problems (object shape distortion, incorrect merging of objects, even object missing due to the shadow cast over another object) while segmenting and extracting moving objects due to the misclassification of shadow points as foreground. Therefore, the framework by

(Cucchiara *et al*., 2001), a general purpose deterministic approach based on HSV colour space is adopted to detect moving cast shadows. An important reason to choose HSV colour space is that HSV colour space corresponds closely to the human perception of colour and it has revealed accuracy in distinguishing visible moving shadows (Shan *et al*., 2007).

The steps to detect and remove shadow points are summarised as follows:

- In this work, background image is modeled as Gaussian. So mean and standard deviation of each pixel are calculated for each colour channel respectively (RGB colour image is used).
- In the current frame, apply motion detection for each pixel. If the pixel value is within a predefined range comparing to the mean of the background pixel, then declare it to be a background pixel. Otherwise, it is classified as a foreground pixel.
- The pixels which are classified as foreground will be used again for the shadow suspicion step. In order to do this, the foreground pixel values are converted from RGB colour space to HSV colour space. Then, shadow points are detected by the following equation:

$$SP_K(x,y) = \begin{cases} 1 & if\ a \le I_k^V(x,y)/B_k^V(x,y) \le b \\ & \Lambda \mid I_k^S(x,y) - B_k^S(x,y) \mid \le t_S \\ & \Lambda \mid I_k^H(x,y) - B_k^H(x,y) \mid \le t_H \\ 0 & otherwise \end{cases} \quad 4.7$$

  where, $I^V_K(x,y)/\ B^V_K(x,y)$ shows the difference of V(luminance) component. And $|\ I^S_K(x,y) - B^S_K(x,y)\ |$ indicates the difference of S(Saturation) component. Then, $|\ I^H_K(x,y) - B^H_K(x,y)|$ indicates the difference of H(Hue) component (Cucchiara *et al*., 2001). Where β is typically close to 1 (0.97 or 0.98), α ranges from 0.75 to 0.85. Typical values of r_S and r_H are both 0.15. In this work, α, r_S and r_H are set to 0.4, 0.3 and 0.4 respectively. Higher values of r_S and r_H can remove more shadow points from foreground pixels, but on the other hand, it causes losing some part of foreground objects which may cause wrong splitting of object.
- Shadow pixels will not be grouped into moving objects to form a track, and at the same time, shadow pixels will not trigger background updating.

- Finally, all foreground pixels which are not classified as shadow will be used to form foreground blobs (moving objects).

The framework of shadow detection is shown in the following figure:

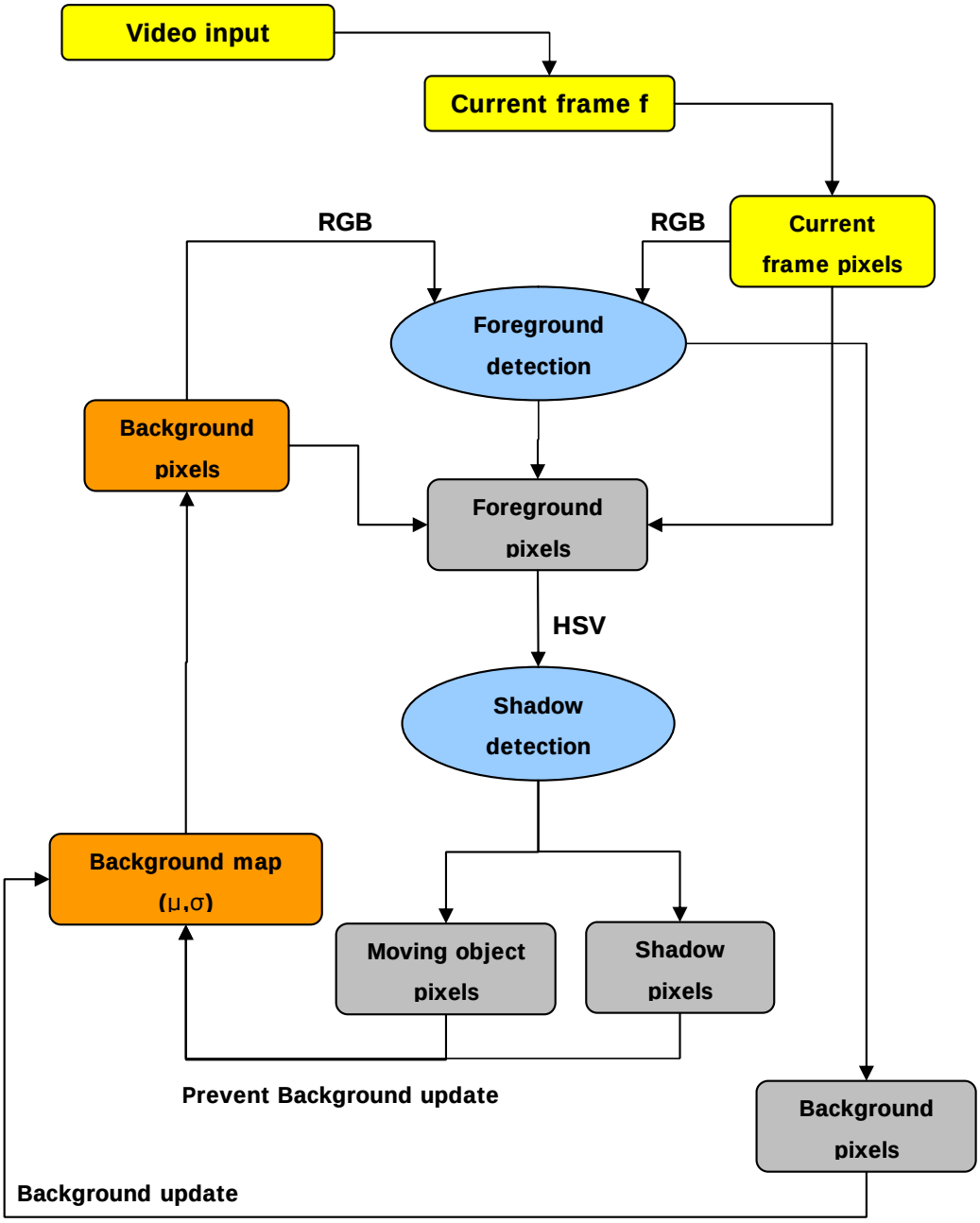


Figure 4-4 Shadow detection framework

4.3.3 Improved Kalman filter

In the BARCO tracker, the Kalman filter is employed to provide temporal consistency for each tracked object and only the centroid of each tracked object is used. Hence, the state vector at frame *k* for track *j* is

$$\mathrm{x}_k = \left(x, v_x, y, v_y\right)^T \tag{4.8}$$

However, this implies that determination of bounding box borders fully depends on the measurement. As a result, the size of an object can become very unstable and further affect higher level tasks which are based on object sizes (depth estimation, camera calibration etc).

Inspired by the work of (Xu and Ellis, 2006), the author improved the BARCO tracker by taking into account the borders of tracks in the Kalman filter. Then, the state vector becomes:

$$\mathrm{x}_k = \left(x, v_x, y, v_y, x^{\max}, x^{\min}, y^{\max}, y^{\min}\right)^T \tag{4.9}$$

where (x,y) and (v_x,x_y) are the centroid and velocity of the object and [x^{min}, x^{max}, y^{min}, y^{max}] is the bounding box. This form incorporate height and width information in the Kalman filter to provide temporal consistency of object's size.

The state and measurement equations are:

$$\mathrm{x}_k = \mathrm{A}\mathrm{x}_{k-1} + \mathrm{w}_{k-1} \tag{4.10}$$

$$\mathrm{z}_k = \mathrm{H}\mathrm{x}_k + \mathrm{v}_k \tag{4.11}$$

where w_{k-1} is the constant process noise and v_k is the measurement noise. w_{k-1} and v_k are assumed to be Gaussian and have to be specified. **A** is the state transition matrix which propagates the old state x_{k-1} to the current state x_k and **H** is the measurement matrix:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

The prediction and final estimation of each are given by:

$$x_k^- = Ax_{k-1}^+ \quad 4.13$$

$$x^+_k = x^-_k + K(z_k - Hx^-_k) \quad 4.14$$

where subscript denotes the time interval, *k*-1 is the previous frame. Superscript - means prediction and superscript + means estimation of the state. And **K** is the Kalman gain matrix which is sought to minimize a priori estimate error covariance P^-_k.

$$K_k = P_k^- H^T \left(HP_k^- H^T + R \right)^{-1} \quad 4.15$$

During each Kalman circle, the current state of track is predicted ahead (Eq.4.13) and then updated and adjusted by the actual measurement at that time (Eq.4.14). When the measurement agrees more with the predication, the estimate error covariance P_k will become smaller. As a result, the Kalman gain will become smaller which means it will give more weight on prediction. On the other hand, determination of bounding box borders will depend more on measurement. During occlusion, if no measurement is available, no update is performed and the determination of bounding box borders will be fully depend on the prediction.

The parameters to operation the Kalman filter are as follows: The process noise w is set to 0.5 pixel/frame for velocity and 0.1 pixel for position. Those values can be derived from the expected acceleration of vehicles in the centre area of the image plane. The measurement noise v is set to 3 pixels. The initial position state corresponds to the detection position with zero velocity. The velocity is updated during the second detection using the first motion vector.

4.4 Results and evaluation

In this section, three new modules are added to the BARCO tracker to deal with shadows and ghosts and to achieve more accurate tracking. The improvement is demonstrated by using the track based evaluation metrics proposed in chapter 3. All different versions of BARCO tracker (with or without the new modules) are tested on the same PC with Pentium IV 3.4 GHz processor and 2GB RAM. Performance as well as the computational load of different versions of the BARCO

tracker with the proposed new modules is compared against the original version using the same parameters through out the whole tests. Part of the datasets mentioned in chapter 3 (e.g. PetsD1TeC1.avi, SZTRA103b15.avi and PVTRA301b04.mov from the i-LIDS dataset) is used here again and also several new sequences specifically suitable for ghost detection are used.

4.4.1 Evaluation of Ghost elimination

The tracking and ghost detection were performed on four video sequences with six ghost events and two abandoned items. The proposed ghost elimination method managed to remove all the six ghost objects, hence avoided all false alarms that caused by ghosts.

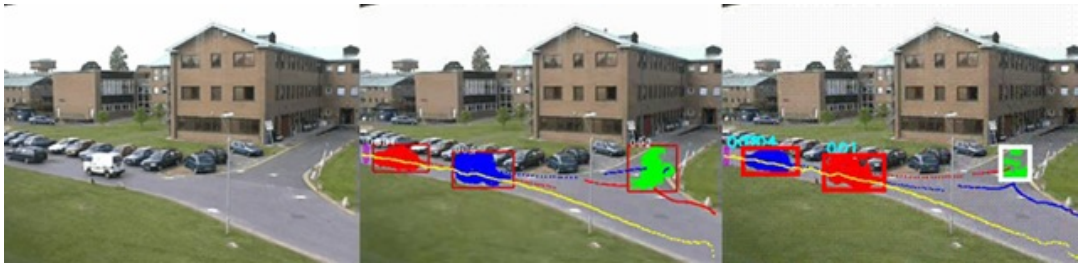
Figure 4-5 to 4-7 present some visual results from the original BARCO tracker as well as from the tracker with ghost detection module. Bounding boxes with red borders represent the valid moving tracks generated by the tracker, bounding boxes with black borders represent the valid but static tracks, while bounding boxes with white borders represent detected ghosts. The patches with colours inside each bounding box present the foreground pixels which are detected as moving pixels.

In figure 4-5f and 4-6f, ghost objects are successfully identified using the proposed ghost detection method. In figure 4-7f, a box was dropped from the car, and the proposed method can still successfully detect the abandoned item (static obejects) without misclassifying it as a ghost.

From the figures shown below, it is demonstrated that the proposed method is effective to detect ghost objects, to prevent the tracks from becoming “valid”, and further more to avoid false alarms. At the same time, the algorithm can differentiate between ghost object and abandoned or static items.

Different values of the frequency (every k frames) of ghost detection were tested (e.g. k = 1, 2, 5, 10, 12, 15, 20…). There is no effect on the performance as long as k is under 20. The decision to set k to 12 ensures reliable as well as fast ghost detection.

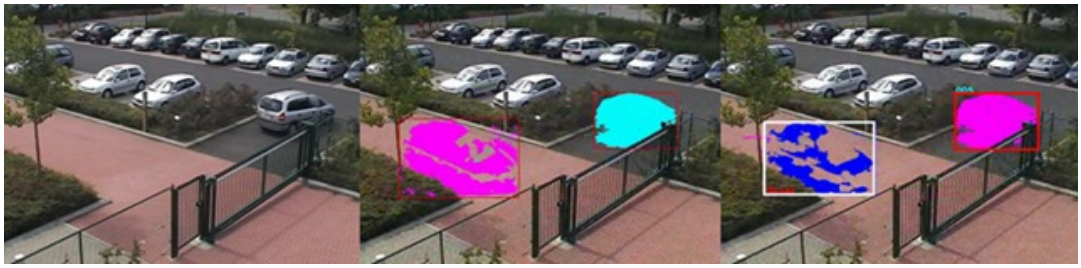
(a) Frame: 2350 (b) BARCO tracker (c) with Ghost detection



(d) Frame: 2630 (e) BARCO tracker (f) with Ghost detection

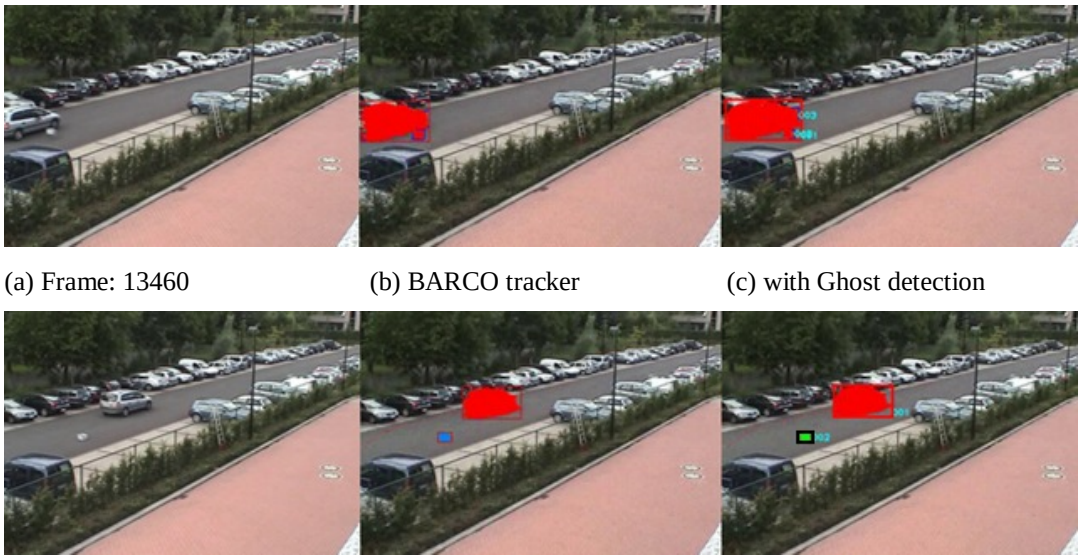
Figure 4-5 PETS2001 PetsD1TeC1.avi sequence is 2686 frames (00:01:29) long and depicts 4 persons, 2 groups of persons and 3 vehicles. Its main challenge is the multiple object intersections and ghosts.

(a) Frame: 9500 (b) BARCO tracker (c) with Ghost detection



(d) Frame: 9550 (e) BARCO tracker (f) with Ghost detection

Figure 4-6 KND2-JULY-GATE1-S2.avi sequence is 18180 frames (00:12:32) long and depicts 29 objects. The main challenges are quick illumination changes and ghosts.



(a) Frame: 13460 (b) BARCO tracker (c) with Ghost detection

(d) Frame: 13530 (e) BARCO tracker (f) with Ghost detection

Figure 4-7 KND2-JULY-EAST-S2.avi sequence is 18102 frames (00:10:04) long and depicts 29 objects. The main challenges are quick illumination changes, abandoned bags.

Quantitative evaluation was carried out using the track based evaluation framework described in chapter 3. The improvement is mainly reflected by the metric: False alarm tracks (FAT). The metrics that mainly reflect the changes of performance are highlighted in the table of results (the same with shadow removal and improved Kalman filter). In PETS2001 sequence (table 4-1), there are two false alarms caused by ghost objects among the three false alarms in the evaluation results for the original BARCO tracker. On the other hand, ghosts have been successfully detected and eliminated by the tracker with the ghost detection module; therefore, the number of false alarms is reduced from 3 to 1 in the evaluation results for the ghost detection tracker. Similarly, the number of FAT is reduced from 5 to 3 in table 4-2 and from 14 to 11 in table 4-3.

The speeds of the original BARCO tracker and ghost detection tracker are both fast enough to be performed in real-time, although there is about 15% to 20% decrease in speed by adding the ghost detection part into the original tracker. In addition, except false alarm track, other metrics such as correct detected track, track detection failure, track fragmentation, average latency, distance error remain more or less the same for both trackers, which means that the ghost detection module does not cause any undesirable side-effects on other modules of the tracker.

According to the overall evaluation results, the ghost detection algorithm can effectively eliminate ghost objects and brings no negative effects to the whole tracking system.

PetsD1TeC1.avi	BARCO tracker	Ghost detection
Total num of frames	2686	2686
Time to process the sequence	33seconds	40
Speed	**81.4 f/s**	**67 f/s**
Number of GT Tracks	9	9
Number of System Tracks	12	11
Correct Detected Track	9	9
False Alarm Track	**3**	**1**
Track Detection Failure	0	0
Track Fragmentation	3	4
ID Change	5	6
Latency of Track	46	49
Average Track Closeness	0.47	0.55
Deviation of Track Closeness	0.24	0.29
Average Distance Error	15.75	16.84
Deviation of Distance Error	23.64	27.23
Average Track Completeness	0.67	0.72

Table 4-1 Evaluation of ghost detection for Pets2001

KND2-JULY-EAST-S2.avi	BARCO tracker	Ghost detection
Total num of frames	21794	21794
Time to process the sequence	753 seconds	824
Speed	**28.9 f/s**	**26.4 f/s**
Number of GT Tracks	23	23
Number of System Tracks	30	28
Correct Detected Track	19	20
False Alarm Track	**5**	**3**
Track Detection Failure	4	3
Track Fragmentation	1	2
ID Change	1	0
Latency of Track	25	29
Average Track Closeness	0.69	0.70
Deviation of Track Closeness	0.29	0.29
Average Distance Error	10.65	9.92
Deviation of Distance Error	19.97	20.66
Average Track Completeness	0.52	0.51

Table 4-2 Evaluation of ghost detection for KND2-JULY-EAST-S2.avi

KND2-JULY-GATE1-S2.avi	BARCO tracker	Ghost detection
Total num of frames	18180	18180
Time to process the sequence	721	810
Speed	**25.2 f/s**	**22.5 f/s**
Number of GT Tracks	29	29
Number of System Tracks	58	57
Correct Detected Track	26	26
False Alarm Track	**14**	**11**
Track Detection Failure	3	4
Track Fragmentation	5	6
ID Change	0	0
Latency of Track	37	38
Average Track Closeness	0.58	0.57
Deviation of Track Closeness	0.25	0.24
Average Distance Error	27.92	27.21
Deviation of Distance Error	55.64	54.08
Average Track Completeness	0.75	0.75

Table 4-3 Evaluation of ghost detection for KND2-JULY-GATE1-S2.avi

4.4.2 Evaluation of shadow detection

The BARCO Tracker with shadow detection module is tested on two i-LIDS video sequences with significant illumination changes and moving cast shadows.

Figure 4-8 and Figure 4-9 present some visual results of shadow detection. Blue points present detected shadow points while red points present real moving objects. Generally speaking, the shadow detection method is able to detect most shadow points (shadows caused by illumination change or moving cast shadows by objects), however, mistakenly classify some part of an object as shadow at the same time.

The track based evaluation framework is used to quantitatively measure the change of performance of the BARCO tracker caused by adding the shadow detection module.

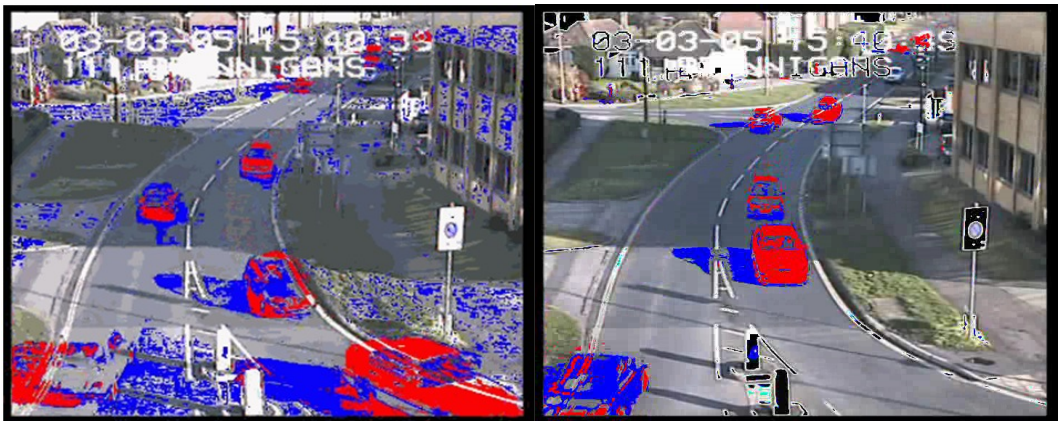


Figure 4-8 Shadow detection of i-LIDS PVTRA301b04.avi sequence

Figure 4-9 Shadow detection of i-LIDS SZTRA103b15.avi sequence

Refer to evaluation results in Table 4-4, better performance is observed for metrics such as track closeness, distance error, ID change and track completeness, which means that the removal of shadow points from real object indeed can improve the accuracy of localization of objects, prevent object from false contact thus preventing ID change, and as a result tracking completeness is better. However, metrics such as track fragmentation is worse than before. This is due to part of an object being classified as shadow thus causing wrong splitting of the object. Another drawback is that using HSV colour space significantly increases the computational load which slows the tracker by 40%. From Table 4-5, except that track closeness and distance error are better, no significant change has been observed for other metrics.

According to the evaluation results, the shadow detection algorithm can successfully detect most shadow points which make the segmentation and tracking

better. However, it brings more wrong splits of objects and much more computational load which is not effective for real-time tracking.

PVTRA301b04.avi	BARCO tracker	Shadow detection
Total num of frames	7309	7307
Time to process the sequence	271	470
Speed	**26.9 f/s**	**15.5 f/s**
Number of GT Tracks	102	102
Number of System Tracks	225	230
Correct Detected Track	90	91
False Alarm Track	67	69
Track Detection Failure	12	11
Track Fragmentation	**62**	**67**
ID Change	**95**	**75**
Latency of Track	57	66
Average Track Closeness	**0.30**	**0.43**
Deviation of Track Closeness	0.21	0.26
Average Distance Error	**49.70**	**29.95**
Deviation of Distance Error	60.31	47.63
Average Track Completeness	0.34	0.45

Table 4-4 Evaluation of Shadow detection for i-LIDS PVTRA301b04.avi

SZTRA103b15.avi	BARCO tracker	Shadow detection
Total num of frames	5821	5821
Time to process the sequence	215	359
Speed	**27.1f/s**	**16.2f/s**
Number of GT Tracks	1	1
Number of System Tracks	8	9
Correct Detected Track	1	1
False Alarm Track	3	4
Track Detection Failure	0	0
Track Fragmentation	2	1
ID Change	0	0
Latency of Track	50	51
Average Track Closeness	**0.65**	**0.67**
Deviation of Track Closeness	0.21	0.25
Average Distance Error	**9.10**	**7.09**
Deviation of Distance Error	12.48	9.26
Average Track Completeness	0.68	0.71

Table 4-5 Evaluation of Shadow detection for i-LIDS SZTRA103b15.avi

4.4.3 Evaluation of improved Kalman filter

In this section, evaluation results of different versions of BARCO tracker (the original tracker and tracker with improved Kalman filter) are presented.

Refer to the evaluation results in Table 4-6 and Table 4-7, minor changes in performance have been observed, although not significant improvements. Metrics such track closeness and distance error are getting better, as well as their deviations. This is mainly due to the predication and updating of each border of tracks using Kalman filter which makes the object size more constant and smooth during tracking. The speed of the algorithm remains more or less the same, as well as other metrics which means that the modification of Kalman filter does not bring any side effects to the tracker.

KND2-JULY-GATE1-S2.avi	BARCO tracker	Improved Kalman filter
Total num of frames	18180	18180
Time to process the sequence	721	723
Speed	**25.2 f/s**	**25.1f/s**
Number of GT Tracks	29	29
Number of System Tracks	58	57
Correct Detected Track	26	26
False Alarm Track	14	13
Track Detection Failure	3	3
Track Fragmentation	5	5
ID Change	0	0
Latency of Track	37	45
Average Track Closeness	**0.58**	**0.62**
Deviation of Track Closeness	**0.25**	**0.24**
Average Distance Error	**27.92**	**25.55**
Deviation of Distance Error	**55.64**	**52.64**
Average Track Completeness	0.75	0.73

Table 4-6 Evaluation of improved KF for i-LIDS KND2-JULY-GATE1-S2.avi

KND2-JULY-EAST-S2.avi	BARCO tracker	Improved Kalman filter
Total num of frames	21794	21794
Time to process the sequence	753 seconds	768 seconds
Speed	28.9 f/s	28.3 f/s
Number of GT Tracks	23	23
Number of System Tracks	30	29
Correct Detected Track	19	20
False Alarm Track	5	4
Track Detection Failure	4	3
Track Fragmentation	1	1
ID Change	1	0
Latency of Track	25	25
Average Track Closeness	**0.69**	**0.70**
Deviation of Track Closeness	**0.29**	**0.26**
Average Distance Error	**10.65**	**9.37**
Deviation of Distance Error	**19.97**	**15.52**
Average Track Completeness	0.52	0.50

Table 4-7 Evaluation of improved KF for KND2-JULY-EAST-S2.avi

Besides evaluation on different modules separately, a combined version of the BARCO tracker with both ghost detection and improved Kalman filter is tested and evaluated. According to the evaluation results shown in Table 4-8 and Table 4-9, the performance of BARCO tracker is under expectation with two new modules adding to it. With the effort of ghost elimination module, the number of false alarms is reduced. Track closeness and Distance error are improved thanks to the improved Kalman filter. At the same time, because of the computational load that brought by the two modules, the speed of BARCO tracker slows down by about 15% which is acceptable. Other metrics remains the same or without significant changes which means putting the new modules together does not bring any negative effects to the tracker. In table 4-10, the performance of BARCO tracker with all three modules is evaluated. The speed of BARCO tracker is significantly reduced because of the extra computational load caused by shadow detection. More track fragmentations are caused by the shadow detection module, but at the same time, fewer ID changes and smaller distance errors. Also, there are fewer false alarm tracks and better track closeness thanks to the ghost detection

module and improved Kalman filter module. Generally speaking, the overall evaluation results are within expectation. Thus, a conclusion can be drawn that putting all the new modules together improves the tracking performance, at the same time, does not bring any other negative side effects to the tracker except the decreasing of processing speed.

KND2-JULY-GATE1-S2.avi	BARCO tracker	Ghost detection & Improved Kalman filter
Total num of frames	18180	18180
Time to process the sequence	721	858
Speed	**25.2 f/s**	**21.2 f/s**
Number of GT Tracks	29	29
Number of System Tracks	58	55
Correct Detected Track	26	26
False Alarm Track	**14**	**9**
Track Detection Failure	3	3
Track Fragmentation	5	5
ID Change	0	0
Latency of Track	37	39
Average Track Closeness	**0.58**	**0.62**
Deviation of Track Closeness	**0.25**	**0.24**
Average Distance Error	**27.92**	**25.61**
Deviation of Distance Error	**55.64**	**53.08**
Average Track Completeness	0.75	0.75

Table 4-8 Evaluation of ghost detection&improved KF for i-LIDS KND2-JULY-GATE1-S2.avi

KND2-JULY-EAST-S2.avi	BARCO tracker	Ghost detection & Improved Kalman filter
Total num of frames	21794	21794
Time to process the sequence	753 seconds	864 seconds
Speed	28.9 f/s	25.2 f/s
Number of GT Tracks	23	23
Number of System Tracks	30	27
Correct Detected Track	19	20
False Alarm Track	**5**	**2**
Track Detection Failure	4	3
Track Fragmentation	1	2
ID Change	1	0
Latency of Track	25	27
Average Track Closeness	**0.69**	**0.70**
Deviation of Track Closeness	**0.29**	**0.25**
Average Distance Error	**10.65**	**9.92**
Deviation of Distance Error	**19.97**	**12.62**
Average Track Completeness	0.52	0.51

Table 4-9 Evaluation of ghost detection&improved KF for KND2-JULY-EAST-S2.avi

PVTRA301b04.avi	BARCO tracker	Shadow detection, Ghost detection, Improved Kalman filter
Total num of frames	7309	7307
Time to process the sequence	271	512
Speed	**26.9 f/s**	**14.3 f/s**
Number of GT Tracks	102	102
Number of System Tracks	225	230
Correct Detected Track	90	92
False Alarm Track	**67**	**63**
Track Detection Failure	12	10
Track Fragmentation	**62**	**73**
ID Change	**95**	**75**
Latency of Track	57	65
Average Track Closeness	**0.30**	**0.42**
Deviation of Track Closeness	**0.21**	**0.26**
Average Distance Error	**49.70**	**29.95**
Deviation of Distance Error	**60.31**	**47.63**
Average Track Completeness	0.34	0.45

Table 4-10 Evaluation of all three modules together for PVTRA301b04.avi

4.5 Discussion

In this chapter, three new modules (ghost elimination, shadow removal and improved Kalman filter) have been added to the BARCO tracker to make it more robust for object tracking and their performance is compared against the original tracker quantitatively. While the shadow removal and improved Kalman filter modules have been based on existing methods, the ghost elimination module is a novel method proposed by the author.

According to the visual results and performance evaluation results, the new modules indeed improve the performance of the BARCO tracker in terms of eliminating ghosts, removing shadows and keeping object size more precise and constant. The three new modules bring no negative side-effects to the original tracker except that the shadow removal makes the tracker much slower than before and brings more track fragmentation.

With the new ghost elimination module, one can effectively avoid false alarms caused by ghost objects while still differentiating between ghosts and abandoned objects. The method works well in all the test scenarios: PETS2001 and another two car park video sequences. The Ghost removal version of the tracker runs almost as fast as the original version and does not have any negative impact on detection and tracking. It can be applied to any other motion detection and tracking systems that uses background subtraction and provide blobs (bounding boxes) for foreground objects.

However, the proposed method is limited by the performance of the tracker. For example, if the tracker cannot provide accurate bounding boxes or cannot detect individual objects in crowd scenes or under occlusions, the method will not be able to detect ghosts either. Also, for videos with highly textured background, the proposed method may confuse between the edges of the background pattern and the real object.

In the next chapter, the improved single camera tracker will be used as part of a multiple camera tracking system. New methods for camera calibration and object correspondence across cameras will be proposed.

5 Multi-camera Tracking

The previous chapter discussed a few new modules that have been added to improve the performance of a basic motion tracker. Also the performance of each new module has been systematically evaluated and compared against the original tracker using the track based evaluation framework proposed in chapter 3.

In this chapter, a method for semi-automatic calibration of multiple camera views to a single view map (called the "ground plane" throughout this chapter) will be proposed. The proposed method requires minimum manual work and does not require a site map. Then, a new method for object correspondence across cameras based on vertical axes is proposed. Finally, Kalman filter is employed for tracking objects on the ground plane and the tracking results are quantitatively evaluated.

5.1 Introduction

Visual surveillance using multiple cameras has attracted the interest of many researchers and practitioners in recent years. However, multiple camera surveillance requires solutions to problems such as calibration of multiple cameras, object correspondence between multiple cameras, object tracking across cameras, automated camera switching, etc. among which, camera calibration and object correspondence between camera views are the fundamental and most important issues.

First of all, camera calibration is required in order to fuse information from different camera views into a single coordinate system. However, full calibration for cameras (recovery of all the intrinsic/extrinsic parameters) monitoring a wide outdoor area requires significant manual work and sometimes it is not possible at all as there might not be physical access to a site. Therefore, an easier and quicker method for camera calibration is required. Then, an accurate and effective data fusion method should be applied for reliable tracking of objects that move across cameras.

To address the issues above, a semi-automatic camera calibration method is proposed to obtain a ground plane map for the whole camera network (with overlapping between camera views) using only camera view images and trajectories from a single camera tracker. Also, a framework is proposed for tracking pedestrians and vehicles based on their vertical axes on the ground plane.

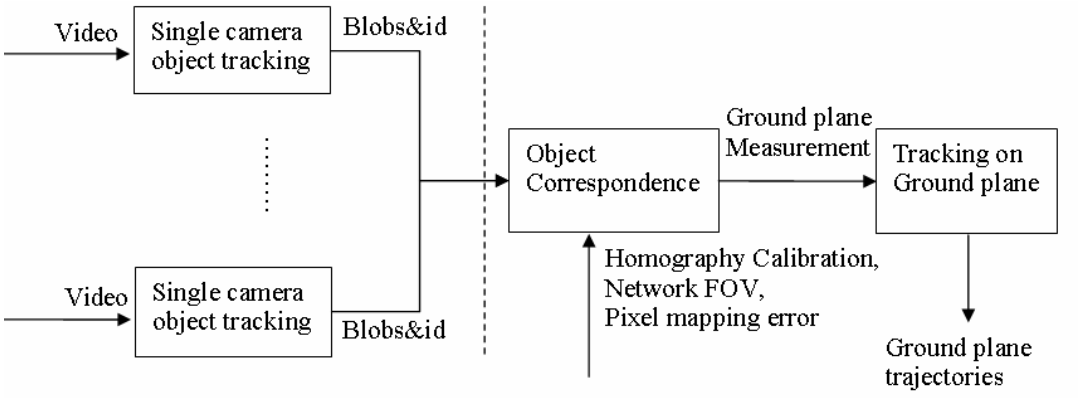


Figure 5-1 Proposed multiple camera tracking framework

5.2 Background

Much work has been done on camera calibration, object correspondence and tracking between multiple cameras in recent years.

Method for camera view relations is presented in (Black *et al.*, 2004) and (Black *et al.*, 2005). Moving objects are detected by background subtraction, and viewpoint correspondence between the detected objects is established by a Least Quantile of Squares approach, then those correspondence points are used to recover the homography mapping between the two camera views. However, their method is only applied for a pair of cameras with significant overlap and sensitive to segmentation noise and occlusion. (Zhang and Scanlon, 2008) proposed a method for calibrating multiple cameras (from 2 to 50) to a single site map (a plane view of a site). They used intersection points of line features which are defined by the user to recover the homography mapping between a camera view and the site map. However, a plan view of the complete site is not always available and no quantitative result or evaluation has been provided. (Yue *et al.*, 2004) introduced a

method which used homography relation between the two views to make correspondence across cameras and handle occlusions. However, their method only works on two camera views and when it comes to more than two cameras, they need to fuse every pair of camera views' tracking results which imposes a heavy computational load.

For object correspondence between camera views, (Hu *et al.*, 2006) proposed a simple and robust method, based on the principal axes of people, to match people across cameras. Their method is less sensitive to segmentation noise because principal axes can still be well determined even when people are in a group or under occlusion. However, their method cannot be directly applicable to vehicle tracking. (Borg *et al.*, 2005) used the Kanade Lucas Tomasi (KLT) feature tracking algorithm to track independent features from frame to frame, and they associated 3D ground plane tracks with measurements from multiple cameras based on a nearest neighbour constraint. However, their object association results can be significantly affected by occlusion of objects. (Khan and Shah, 2003) used the points located on feet to match people in multiple views, based on the homography mapping from one camera view to another. However, in many situations, people's feet are likely to be occluded which will significantly affects the reliability of their method.

In the work described in this chapter, an object correspondence method based on vertical axes of objects is proposed which can be applied to both pedestrians and vehicles and be robust against segmentation noise and occlusion of objects. The spatial relationships between cameras are determined using the proposed homography-based automatic calibration framework which is able to produce a single view map for the whole camera network.

5.3 Homography calibration

In this section, a method is proposed to automatically recover the homography relationship between each pair of camera views, and then project all camera views to a single view map thus building up a common reference plane for the camera network.

5.3.1 Concept of Homography

Before the calibration method is introduced, it is important to first describe the concept of homography mapping. A homography defines a planar mapping between two images that have some level of overlap. It maps points (x,y) of one image to points (x',y') on the other image.

$$x' = \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + h_9} \quad 5.1$$

$$y' = \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + h_9} \quad 5.2$$

Equation 5.1 and 5.2 can be transformed to the following equations:

$$[x, y, 1, 0, 0, 0, -xx', -yx', -x'] H = 0 \quad 5.3$$

$$[0, 0, 0, x, y, 1, -xy', -yy', -y'] H = 0 \quad 5.4$$

Where H is the vector form of homography matrix:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^T \quad 5.5$$

Each pair of correspondence points provides two equations (equation 5.3 and 5.4). Given n (n$\geqslant$4) correspondence points, a (2n by 9) matrix A can be constructed and then the coefficients of the homography can be computed using Singular Value Decomposition (SVD). Once the homography mapping is done, it allows us to establish correspondence between points from two images.

$$A \times H = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 x_1' & -y_1 x_1' & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 y_1' & -y_1 y_1' & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 x_2' & -y_2 x_2' & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 y_2' & -y_2 y_2' & -y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n x_n' & -y_n x_n' & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_n y_n' & -y_n y_n' & -y_n' \end{bmatrix} \times \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = 0 \quad 5.6$$

Generally speaking, the result of homography depends on two conditions (Black and Ellis, 2005): accurate registration of correspondence points and good coverage of the image plane by the correspondence points. Correspondence points

between two camera views can be obtained directly from detection or tracking, however, they are likely to have noises. Therefore, manually selected line features are used to achieve more accurate homography mapping in the next section.

5.3.2 Semi-automatic Homography

In this section, a framework is proposed which can automatically recover the homography matrix H using trajectories points generated by a motion tracker (BARCO tracker) and manually defined lines features in the camera view. For each camera view, a number of trajectory points are obtained: $\{(x^P_{j,k}, y^P_{j,k})\}$ is the centre bottom point of the bounding box of track j at frame k , p (p = 1,2…N) indicates the camera view. The steps to estimate the homography for a pair of camera views are summarized as follows:

1. For each pair of synchronized image frames, all the possible combinations of pairs of bounding boxes are formed and the image coordinates of their centre bottom points are used as potential correspondence points.
2. After all M pairs of potential correspondence points are selected (could be thousands of pairs), four pairs of points are randomly taken to estimate a homography transform using Eq.5.6. The points that have been chosen need to satisfy the following condition: each point has to be further than a certain distance (e.g. a quarter of the image diagonal) from any other points from the same camera view. This ensures a good coverage of points across the image plane mentioned in section 5.3.1.
3. Then, lines features which were manually selected are used to check whether the estimated homography is accurate or not (two line segments are selected which represent geometric features in both views, see Figure 5-3. whether the lines are parallel to each other or have the same length does not matter). Then the equations of the lines in the slope-intercept form are recovered: $y = cx + b$.

Figure 5-2 Correspondence Lines features in different camera views

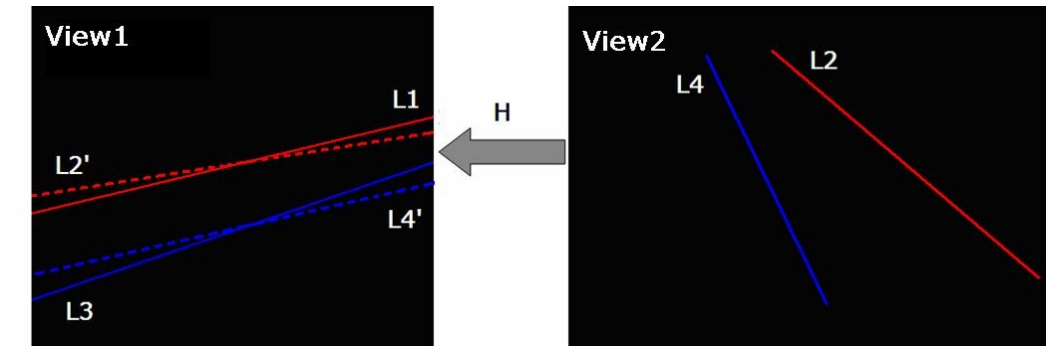


Figure 5-3 Mapping of line features (the dashed lines) from a second camera view

In Figure 5-3, L1 and L2 are the projections of the same line in real world on camera views 1 and 2 respectively (the same applies for L3 and L4). Then, line L2 is projected from view 2 to view 1 (L2') using the estimated homography. Let $y = c_{l1}x + b_{l1}$ and $y = c_{l2'}x + b_{l2'}$ be the line equations of L1 and L2' respectively. L1 and L2' have to satisfy the following conditions for the homography mapping to be considered as a successful one (the conditions for L3 and L4' are also checked):

- The slope of two lines should be similar (e.g. $T_c = 0.1$)

$$\frac{|c_{l1} - c_{l2'}|}{|c_{l2'}|} < T_c \qquad 5.7$$

- Both the y-intercept and x-intercept of two lines have differences smaller than T_b pixels (e.g. $T_b = 5$ pixels):

$$|b_{l1} - b_{l2'}| < T_b, \qquad |b_{l1}/c_{l1} - b_{l2'}/c_{l2'}| < T_b \qquad 5.8$$

4. If the lines do not satisfy the above conditions, the homography needs to be re-estimated by using a different set of random selected points (back to step 3).

5. Refining step: Once a set of four points can produce a homography mapping with sufficient accuracy, small random Gaussian noise with zero mean and small standard deviation will be added to the four points. One point is selected randomly from each Gaussian distribution around the four points respectively and the homography is estimated again. The aim of this step is that the initial estimation of homography parameters from points that directly from object detection and tracking is prone to have errors. Their neighbourhood pixels are searched and used to estimate the homography parameters again in order to obtain a more accurate estimation. Then, the procedure is terminated when line features fits each other with error that are presented in equations 5.7 and 5.8 but with stricter thresholds (e.g. T_c = 0.01 and T_b =1). Finally, the homography mapping between two views is established.

5.3.3 Single view map

After all the homographies between pairwise camera views are calculated, a main view or single view map which can represent the total coverage of the network still needs to be built.

In this work, a ground plane is created by manually defined feature points as seen in Figure 5-4 (based on the assumption that there is no precise site map available) and then all the N camera views are mapped to the ground plane. However, the ground plane needs to be flat without changes of altitude. In what follows an example of four camera views will be used, however, the procedure is applicable to any number of cameras as long as they have overlapping views.

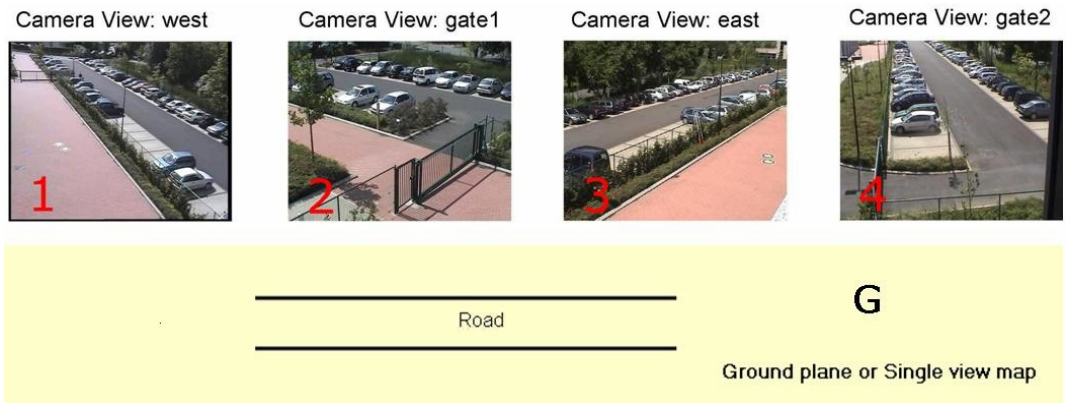


Figure 5-4 Different camera views and the single view map

Let H_{pq} represent the homography mapping from view p to q ($p,q = 1,2…N,G$). The numbers 1 to N represent all the camera views and G represent the ground plane map (see Figure 5-5, for example, H_{31} means homography from view 3 to view 1, H_{1G} means homography from view 1 to the ground plane). The steps to build up a single view map are summarised as follows:

- Homographies between pairs of camera views are estimated as explained in section 5.3.2. However, since trajectory points are only available for single camera view, several (e.g. four) feature points need to be manually drawn on the ground plane to compute homography from any of the N camera views to the ground plane (see Figure 5-5, red points).

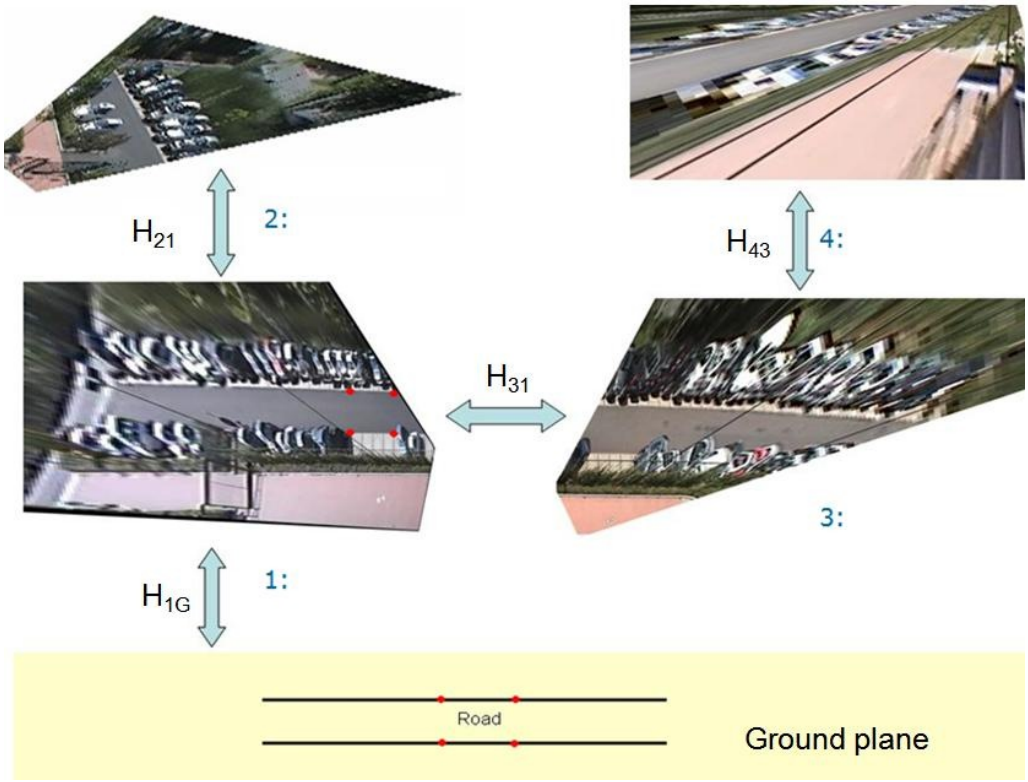


Figure 5-5 Mapping different camera views to the ground plane

- Then, the homography relationships between the views (H_{21}, H_{31}, H_{43} etc.) are used to calculate the homography relationship between the camera views and the ground plane. For instance, points from view 3 can be firstly mapped to view 1 using H_{31}, then to the ground plane using H_{1G}.
- After all these homographies have been calculated, all N camera views can be projected onto the ground plane (see Figure 5-6), thus a single

view map which represents the FOV for the whole camera network is built.

- Finally, based on the assumption that the average width of a car is about 1.8 metres (about 90 pixels in width when projected from single camera view onto the ground plane map), measurements on the ground plane map can then be converted into metres.

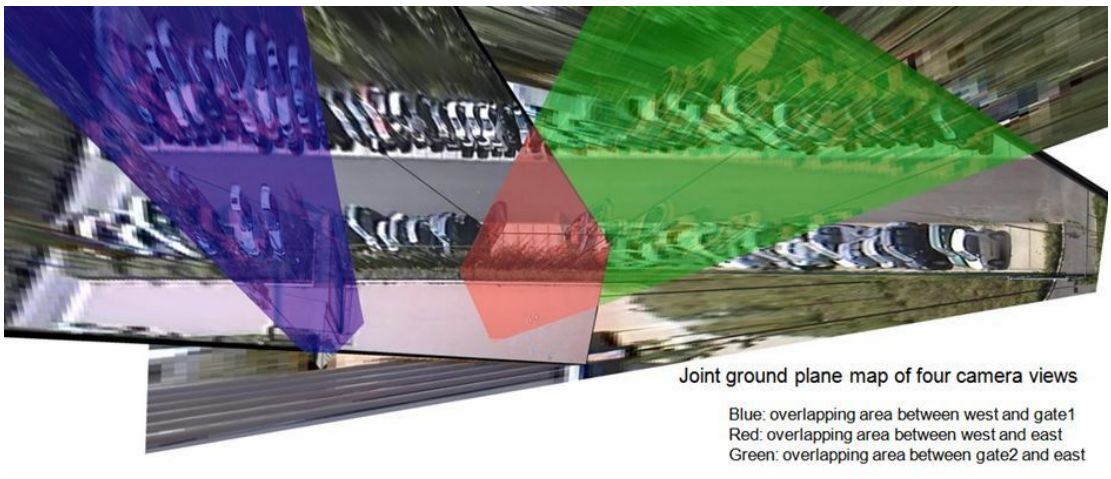


Figure 5-6 Ground plane map for all the camera views

5.4 Object correspondence and tracking

In the previous section, a homography-based camera calibration method has been proposed which establishes a ground plane map for the whole camera network. In this section, a framework is proposed for object correspondence across cameras and object tracking on the ground plane.

5.4.1 FOV, vertical axis and pixel mapping error

Before the multi-camera object tracking framework is discussed, it is important to introduce the concepts of camera network FOV, vertical axis and pixel mapping error.

5.4.1.1 Camera/network FOV

In order to integrate tracking data from multiple cameras, it is beneficial to consider the visibility of targets within the entire environment, and not just each camera view separately. Therefore, during the procedure of camera calibration, the

ground plane is automatically divided into the following regions according to semantic relevance in the scene: camera FOV, network FOV, overlapping and non-overlapping areas and regions out of coverage (Ellis *et al.*, 2003), as illustrated in Figure 5-7.

In later sections, how to use these predefined categories of regions to assist object correspondence and tracking will be described. For instance, if an object is located in an overlapping area of two cameras, detections from both cameras will be expected. If there is detection from only one camera, the detection will be labelled as "weak" or non-interesting (see Sec.5.4.3).

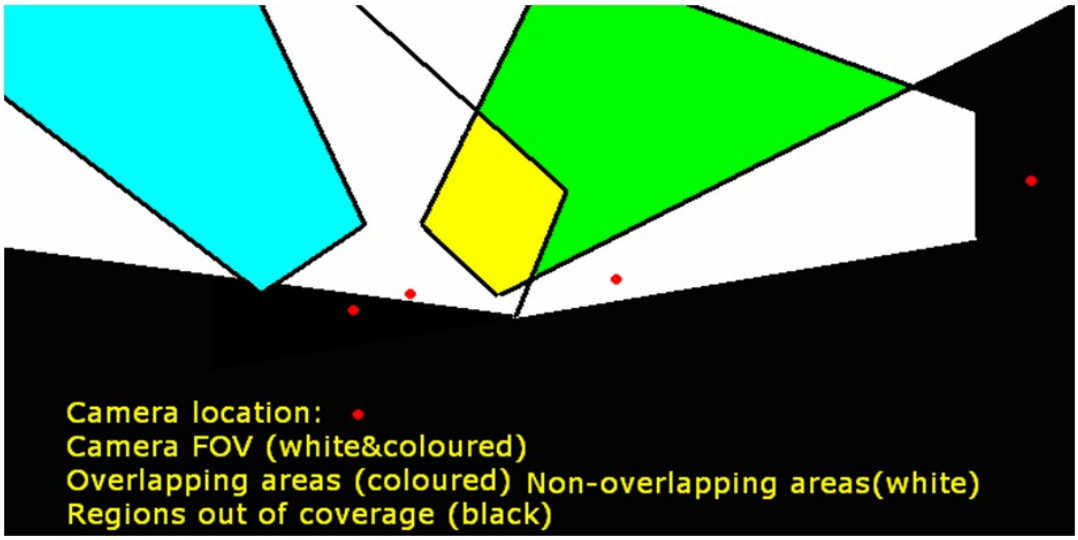


Figure 5-7 Camera network FOV

5.4.1.2 Vertical axis

For object correspondence between camera views, different features have been used as discussed in section 5.2 (e.g. bottom or head points, SIFT features, principal axis). In this work, vertical axis is used to match objects from different camera views due to its robustness against segmentation noises and occlusions compare to point based methods as demonstrated in Sec.5.5.

Principal Component Analysis (PCA) is applied to the foreground pixels located in each blob to determine the two predominant axes of each blob for each frame. The axis line with the smaller angle to the y-axis is deemed to be the vertical axis of the object. The vertical axis is different from the principal axis defined in

(Hu *et al.*, 2006); while the principal axis is normally suitable for pedestrians, it cannot apply generally for vehicles, because it may be the axis parallel to the ground. Figure 5-8 shows the difference between the principal axis and the vertical axis for a vehicle.

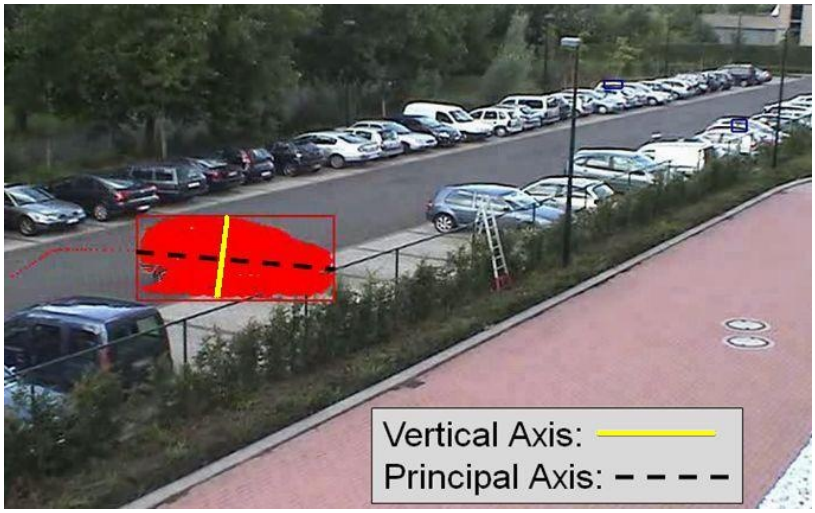


Figure 5-8 Difference between Principle axis and Vertical axis

The vertical axis works for most CCTV camera installations, where there is no significant roll angle for the camera. When it is not the case, an image transformation can be applied to satisfy this condition if the roll angle is known. In those cases, a vertical line on the ground plane projects to a line which is approximately parallel or have small angle (less than 45°) to the y-axis in the image plane. Most of the current surveillance camera setup can satisfy the above conditions (see Figure 5-9, yellow line segments represent the detected vertical axes). However, there are cases in which the vertical axis does not work. For example, in Figure 5-10, when the camera view is almost top down, or a fish eye view, the vertical axis of objects can face any direction on the image plane, therefore, the algorithm sometimes fail to detect the right vertical axis. Instead, the wrongly detected "vertical axes" represent lines that are parallel to the ground plane in 3D real world coordinates and cannot be used to estimate the ground point of an object.

Figure 5-9 Examples of vertical axes for PETS multi-camera datasets (PETS01 and PETS06)

Figure 5-10 Situations where the vertical axis does not work. left: CAVIAR dataset (CAVIAR, nd), right: PETS2001 dataset (PETS, nd) The vertical axes(yellow lines) estimated by the proposed method are not always estimated properly (labelled as "wrong")

5.4.1.3 Pixel mapping error

Pixel mapping error is defined as the distance on the ground plane that corresponds to a difference of one pixel on the image plane. The main reason is that when data from different cameras are fused later, a weighting function will be used and that will favour cameras that have smaller pixel mapping errors (i.e. which camera is more accurate at the given ground plane position). Moreover, more tolerance needs

to be set when tracking in an area that has larger pixel mapping error. The pixel mapping error for each image pixel is calculated by:

$$E_{X_p} = \frac{\left| X_p^G - \left(X_p - 1 \right)^G \right| + \left| X_p^G - \left(X_p + 1 \right)^G \right|}{2}; \qquad E_{Y_p} = \frac{\left| Y_p^G - \left(Y_p - 1 \right)^G \right| + \left| Y_p^G - \left(Y_p + 1 \right)^G \right|}{2} \tag{5.9}$$

Where X_p is the x-coordinate of a pixel on the image plane for camera view p, and X^G_p is the ground plane coordinate for X_p using the homography mapping mentioned in Sec.3. Similarly, $(X_p\text{-}1)^G$ is the ground plane coordinate for X_p-1.

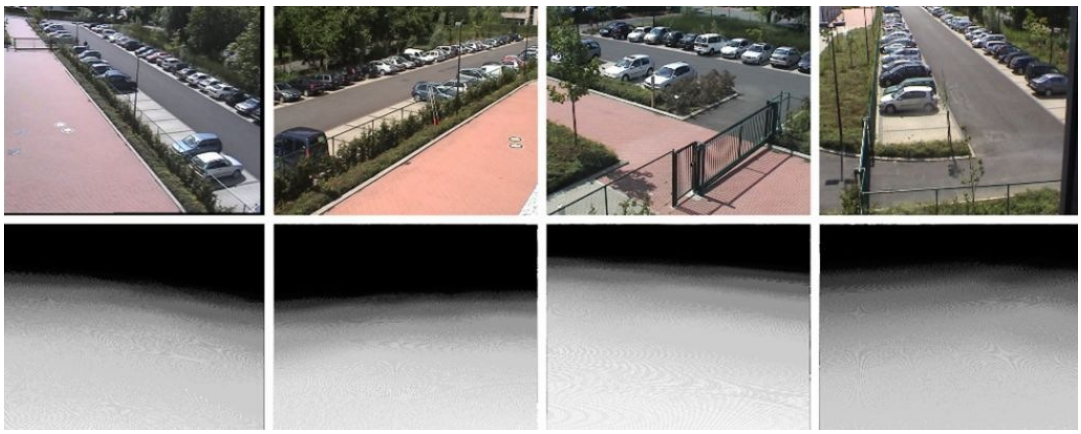


Figure 5-11 Pixel mapping error field for each camera view represented by brightness: the darker the pixel, the higher the mapping error for the specific pixel

Generally speaking, areas that have higher pixel mapping error are areas that are far away from the cameras (darker areas shown in Figure 5-11). The areas which are closer to the cameras have smaller pixel mapping error which means measurement will have more certainty. The pixel mapping error will be used at the data fusion step for ground truthing in Sec.5.5.

5.4.2 Obtaining ground plane measurements

Once the camera calibration and camera network FOV are ready, the homography relationships, derived as explained in Sec.5.3, between each camera view and the ground plane are used to make correspondences between detected objects from different camera views.

For a given frame, the following steps are used to match and fuse observations from different camera views:

1. Compute the vertical axes of all detected objects from all camera views and project them onto the ground plane: creating a list of all candidate matches for each object from each camera view (objects from the same camera view cannot be a match).
2. Calculate the matching distance error D for each pair of objects, as follows: Let g_m^p and g_n^q be the ground plane projections of the vertical axes of the m^{th} object from camera p and the n^{th} object from camera q respectively (see Figure 5-12). Let b_m^p and b_n^q be the ground plane projections of the middle bottom points (or foot points) of the objects' bounding box in each camera view respectively. Let $X_{m,n}^{p,q}$ be the ground plane intersection of g_m^p and g_n^q (see Figure 5-12). The matching distance errors can be calculated as:

$$D_{m,n}^{p,q} = dist(b_m^p, X_{m,n}^{p,q}) + dist(b_n^q, X_{m,n}^{p,q}) \qquad 5.10$$

where $dist(b_m^p, X_{m,n}^{p,q})$ is the Euclidean distance between points $X_{m,n}^{p,q}$ and b_m^p

3. Objects with the smallest matching distance error and at the same time, smaller than a threshold D_T will be considered as a match. For instance, if $D_{m,n}^{p,q}$ is on top of the matching distance list for the m^{th} object from camera p and the n^{th} object from camera q, then the match between them is established. D_T is defined as the height dimension of the ground plane projection of the object's bounding box.
4. Once a match has been established, that pair of objects is removed from the candidate list.
5. There are two cases for objects that do not have a match. If the bottom point b is located in a non-overlapping area (the white regions in Figure 5-7), then the point will be used as a ground plane measurement. If b is located in an overlapping area, it will still be used as a ground plane measurement but labelled as "weak". The information will be passed to the ground plane track that it is associated with.

6. If an object is partly occluded (it is assumed that this condition can be detected by the single camera tracker), the threshold D_T is increased to improve correspondence.

$$D_T^e = \frac{S}{S_{occ}} D_T \qquad 5.11$$

where D_T^e is the larger threshold, S is the object's dimension before occlusion, and S_{occ} is the object's dimension under occlusion.

7. If an object is totally occluded or touches the borderline of its camera view, it will not be used for correspondence purposes because the shape and position of the object are not reliable any more for ground plane tracking.

After object matching is finished, the correspondence information is used to calculate ground plane measurements. The intersection point $X_{m,n}^{p,q}$ will be taken as the approximate measurement of "ground point" for an object. As shown in Figure 5-12, even if the lower part of an object is occluded in one camera view, the intersection point $X_{m,n}^{p,q}$ can still be calculated, which means the ground plane location of the object can still be estimated.

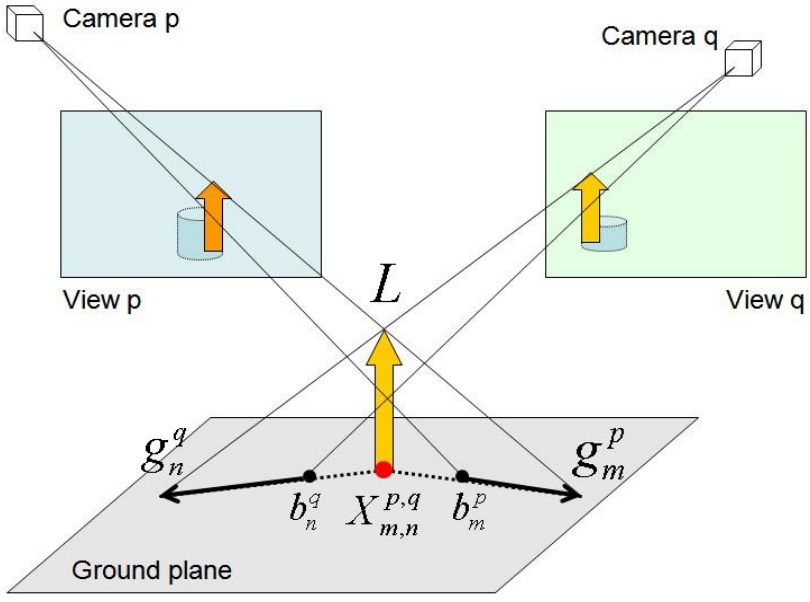


Figure 5-12 Example of object's (the yellow arrow) location (red dot) on the ground plane under occlusion

5.4.3 Tracking on the ground plane

Recapping from what was described in the previous section, at any given frame, the combination of cameras identifies *N* tracked objects. The vertical axis based approach merged detected objects from different camera views so that at the end of each frame there are *M* (*M*<=*N*) ground plane measurements (ground points). Then, Kalman filter is used to perform tracking on the ground plane. The standard formulation of the Kalman filter for a constant velocity model is used and define by Eq. 4.10 and 4.11. The state and measurement vector are defined as:

$$\mathrm{x}_k = \left(x, v_x, y, v_y\right)^T \tag{5.12}$$

$$\mathrm{Z}_k = \left(x, y\right)^T \tag{5.13}$$

[*x*,*y*] is the location on the single view map coordinates, [v_x, v_y] is the velocity. The state transition matrix and measurement matrix are defined as:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.14}$$

For a giving frame, the steps to update each tracked ground plane track is summarised as follows:

- Generate predictions for each ground plane track using the state transition matrix from the set of ground plane tracks at time t-1.
- Each time tracks are associated with ground plane measurements in the following order: strong tracks that have longer existing time go first, then strong tracks that have shorter existing time, afterwards weak tracks with longer existing time, weak tracks with shorter existing time go last.
- To solve the data association problem, for each track, a Mahalanobis distance table is created and the distances are sorted in descending order, according to the following equation.

$$d = \sqrt{\left(Hx_{k-1} - Z_k\right)^T R_t^{-1} \left(Hx_{k-1} - Z_k\right)} \tag{5.15}$$

Where Hx_{k-1} is the predicted ground plane position for the nth track, Z_k is the m^{th} ground plane measurement. R_t^{-1} is the covariance matrix of the measurement noise.

- Select the ground plane measurement which has the smallest distance to the track and also smaller than a threshold T_{assoc}. Update the ground plane track using the associated ground plane measurement.
- For each ground plane measurement that does not have a match, create a new track for it.
- For a ground plane track that does not match any ground plane measurement, the track will be updated using Kalman filter for a few seconds. Then, it will be vanished.
- Each ground plane track has three possible statuses: weak, strong, and vanishing (see Figure 5-13). The idea of introducing strong and weak tracks is to make sure enough evidence has been accumulated before a track turns to be a "real" track that the end user are interested and moreover, to avoid possible false alarms. Only strong tracks are taken as output of the system and used for performance evaluation.

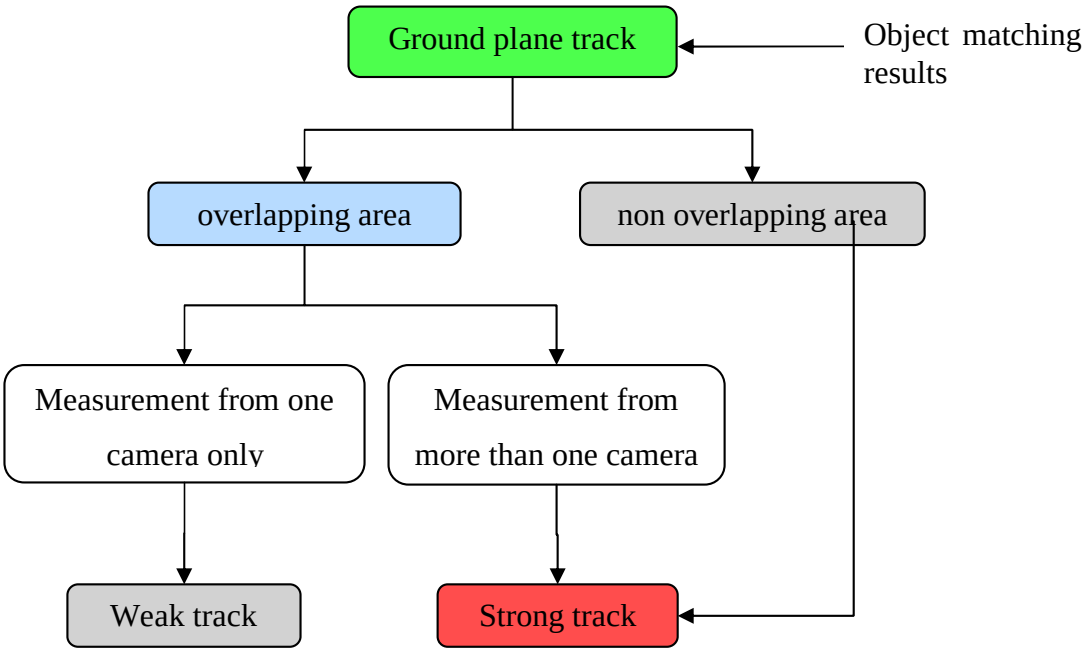


Figure 5-13 Conditions to determine track status

To configure the ground plane Kalman filter, the covariance of the ground plane position measurement noise R and the system process noise covariance Q are assumed Gaussian and need to be estimated. The processing noise is set to 1.0 m/s for velocity and 0.6 m for position (estimated by measuring the departure of the real data based on the assumption of constant velocity). The ground plane measurement noise is set to 1.5 m for position.

5.5 Results and evaluation

The multiple camera calibration and tracking framework described in previous sections has been tested using the SERKET multiple camera datasets, which has four cameras and three video sequences (about 15 minutes each) for each camera view.

Figure 5-14 Examples of the SERKET dataset

The performance of the system is evaluated quantitatively using the performance evaluation framework proposed in Chapter 3. The ground truth for the ground plane is obtained using the following steps:

- Manually obtain ground truth bounding boxes for each object in each camera view using viper-GT.
- Maintain unique ID for each object for all camera views.
- Project the centre bottom point of each ground truth bounding box onto the ground plane as the ground truth point.

For fusing ground truth points from different cameras, a weighted average is used by taking into account the pixel mapping error:

$$L_{fused} = \left(w_1 L_1 + \mathbf{K}\, w_N L_N \right) \qquad 5.16$$

Where L_{fused} is the fused ground truth point for a ground truth object, L_i is the ground plane point for the i^{th} correspondence point and w_i is the weight which is calculated as:

$$w_i = \frac{w_i^{'}}{\sum_{i=1}^{N} w_i^{'}} \quad and \quad w_i^{'} = 1 \Big/ \mathrm{cov}\left(E_{x_i}, E_{y_i} \right) \qquad 5.17.$$

Table 5-1 shows the evaluation results for the ground plane tracker using different features for object correspondence: ground plane tracker0 is based on principal axis proposed by (Hu *et al.*, 2006), tracker1 is based on ground plane projection of the "foot" points, similar to (Khan and Shah, 2003) and tracker2 is based on vertical axis proposed in this work. Table 5-2 shows the evaluation results from the single camera trackers only. One can notice that the ground plane tracker (both 1 and 2) successfully avoids false alarms which is better than single camera trackers (15 false alarms in total), and at the same time maintains very high detection rate (91% successful detection). This is mainly because ground plane tracker 1&2 collects detection information from multiple cameras. If one camera fails to detect the object, there are still other cameras to rely on. Even when it is classified as weak, the system still keeps tracking the object and it can become a strong track later on. On the other hand, because of the distinction between weak and strong tracks, false detection from single cameras (which usually do not have enough movement as well) will be classified as weak and will not trigger a false

alarm. Also, the foot points of single camera trackers are projected onto the ground plane in order to compare with ground plane trackers (see Table 5-2). As we can see that by introducing the vertical axis, the ground plane tracker2 can achieve more accurate localization of objects on the ground plane (0.7m error against single cameras' 1.1m on average). On the other hand, ground plane tracker1 only uses foot points for object correspondence and tracking, therefore, the distance error is similar in comparison to single camera trackers (1.15m against 1.1m).

By introducing the rules for object correspondence and tracking in section 5.4.2 (removing non-reliable detection of objects such as totally occluded or reaches the borderline of a camera FOV), ground plane tracker2 performs well in terms of continuously tracking ground plane objects when it moves across multiple cameras. The reason for two detection failures is that tracks which move a very short distance (shorter than T_D) are classified as weak and missed. Track fragmentation is caused by mistakes such as large temporal gaps between detections from different cameras for the same ground plane object or correspondence failure of detections from different cameras.

As we can see from Table 5-1, ground plane tracker2 provides better results than tracker1 and tracker0 in terms of track completeness, track fragmentation, and obtains better accuracy of localization of objects on the ground plane (0.7m against 1.15m). The main reason is that vertical axis is more robust against object occlusions and segmentation noises by using the intersection of vertical axes as an approximation of object's ground location, while using foot points may cause loss of tracking when an object is partially occluded or inaccurate localization of the object due to segmentation noise. Tracker0's large distance errors on estimating ground plane tracks is explained by the fact that principal axis cannot deal with vehicles as shown in Figure 5-8 (some visual results are shown in Figure 5-15). Therefore, vertical axis can be considered as a more reliable feature for object matching across cameras.

EAST(WEST,GATE1,GATE2)-S2	Ground plane tracker0: based on principal axis	Ground plane tracker1: based on bottom points	Ground plane tracker2: based on vertical axis
Number of GT Tracks:	22	22	22
Number of System Tracks:	31	29	26
Correct Detected Tracks:	19	20	20
False Alarm Tracks:	0	0	0
Track Detection Failure:	3	2	2
Latency of Track (frames) :	93	72	72
Track Fragmentation:	8	6	3
ID Change:	0	0	0
Average Distance Error (metre):	2.06 m	1.15 m	0.71 m
Deviation of Distance Error:	2.78 m	0.77 m	0.59 m
Average Track Completeness :	0.41	0.67	0.70

Table 5-1 Evaluation results for ground plane tracking

Single camera tracking	EAST	WEST	GATE1	GATE2
Number of GT Tracks:	11	21	19	6
Number of System Tracks:	16	24	49	7
Correct Detected Tracks:	9	21	19	6
False Alarm Tracks:	2	1	11	1
Track Detection Failure:	2	0	0	0
Latency of Track (frames) :	26	71	66	126
Track Fragmentation:	0	1	7	0
ID Change:	0	0	0	0
Ground plane Distance Error	1.53 m	0.88 m	1.36 m	0.73 m
Average Distance Error (pixels):	6.62	8.00	28.51	11.58
Deviation of Distance Error:	9.56	7.56	56.87	5.93
Average Track Completeness	0.70	0.71	0.78	0.51

Table 5-2 Evaluation results for each single camera tracking

The following figures show some visual results of ground plane object tracking. On the ground plane, colour bounding boxes represent projections of detections from single camera views. Coloured dots are trajectories of strong

(interesting) ground plane tracks (a random colour is assigned to each ground plane track), while white dots are tracks that are classified as weak (non-interesting).

Figure 5-16 and Figure 5-17 show tracking results by ground plane tracker 1 and 2. Ground plane tracker 1 uses foot (or bottom) point for object correspondence and obtaining ground plane measurement. However, the position of bottom point for cars can be very different from different view point and very sensitive to segmentation noise and occlusions which causes the discontinuity of trajectory in Figure 5-16(middle) and Figure 5-17(bottom). However, using the vertical axis, a more accurate approximation of ground point can be estimated thus a more accurate trajectory can be obtained by tracker 2: Figure 5-16(top) and Figure 5-17(top).

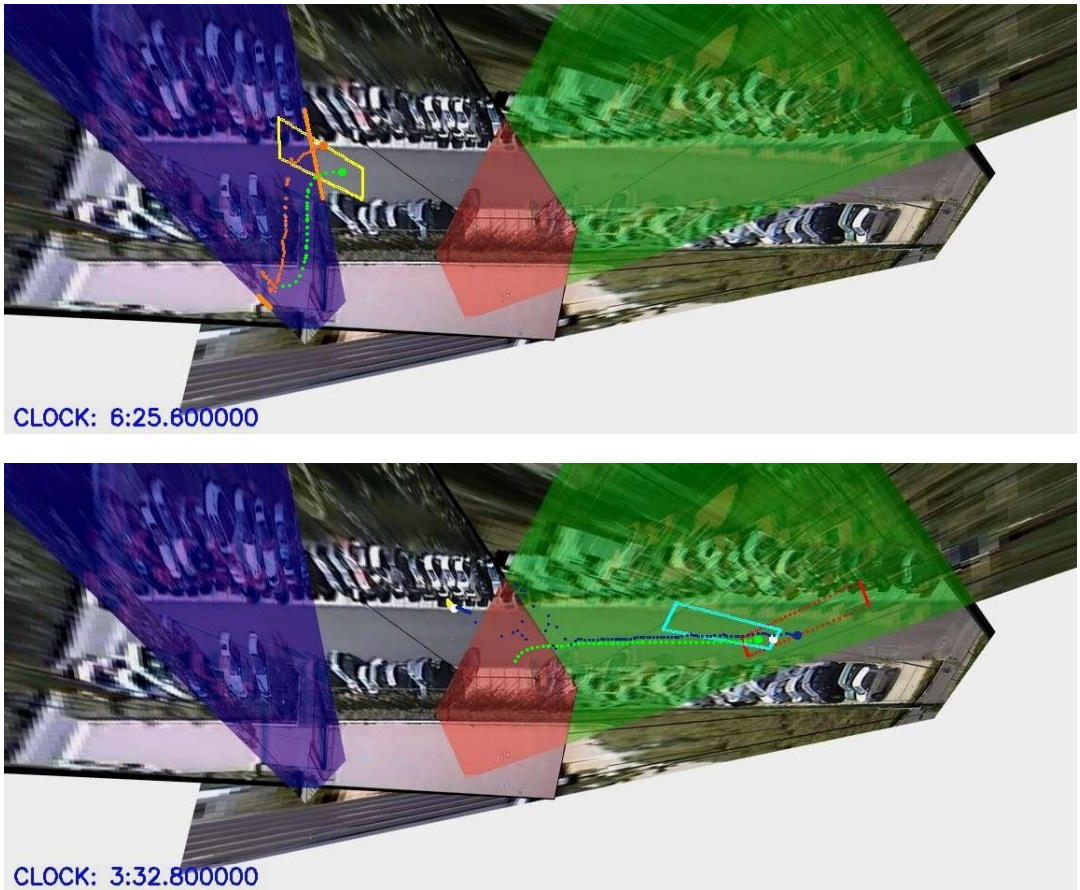


Figure 5-15 Examples of distance errors of ground plane tracker0 based on principal axis (green points are the ground truth points, coloured points are the estimated trajectories by tracker0 based on principal axis)

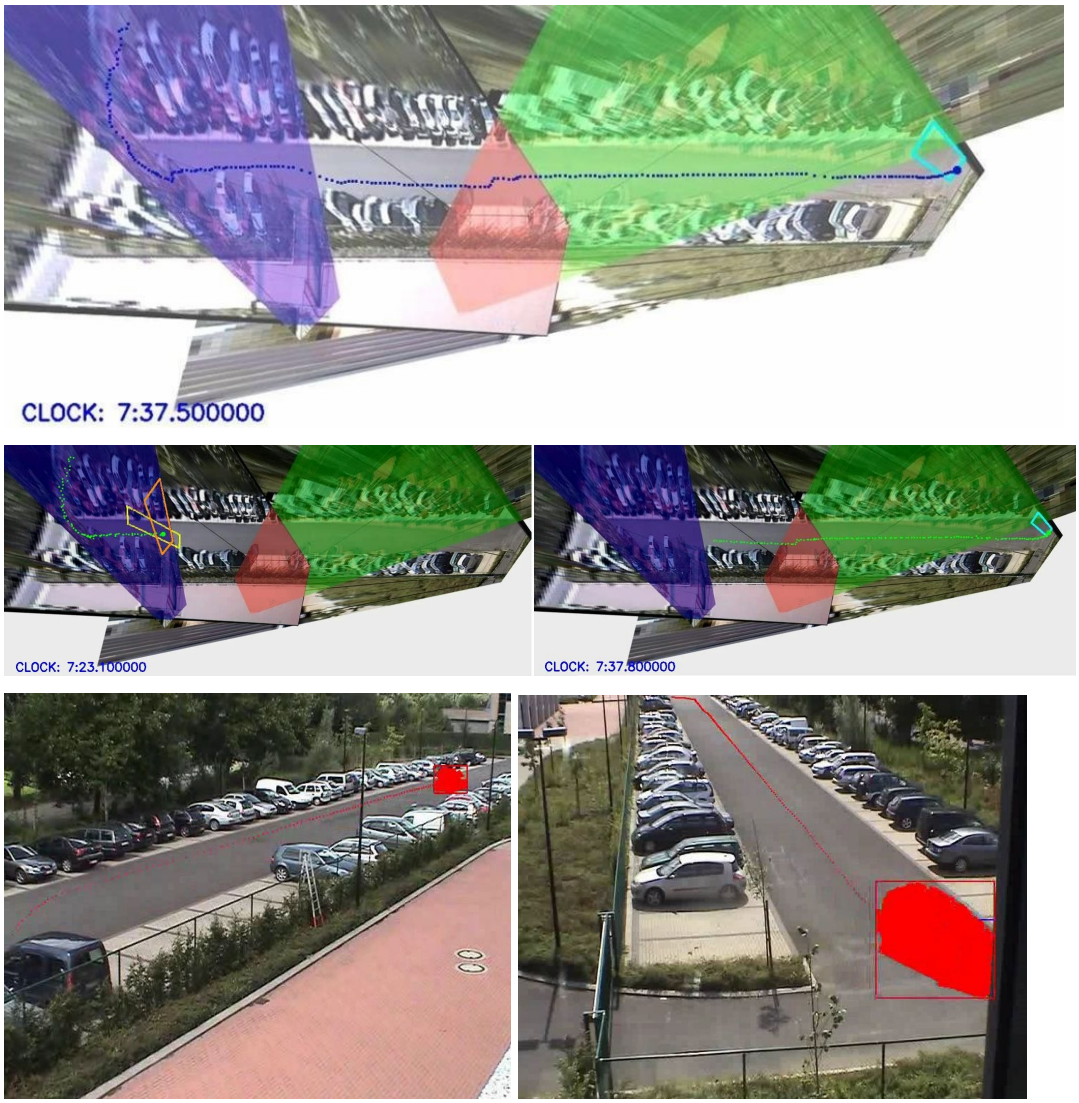


Figure 5-16 Example of successful tracking by tracker2 (top) and track fragmentation by tracker1(middle): a car moved across the whole camera network

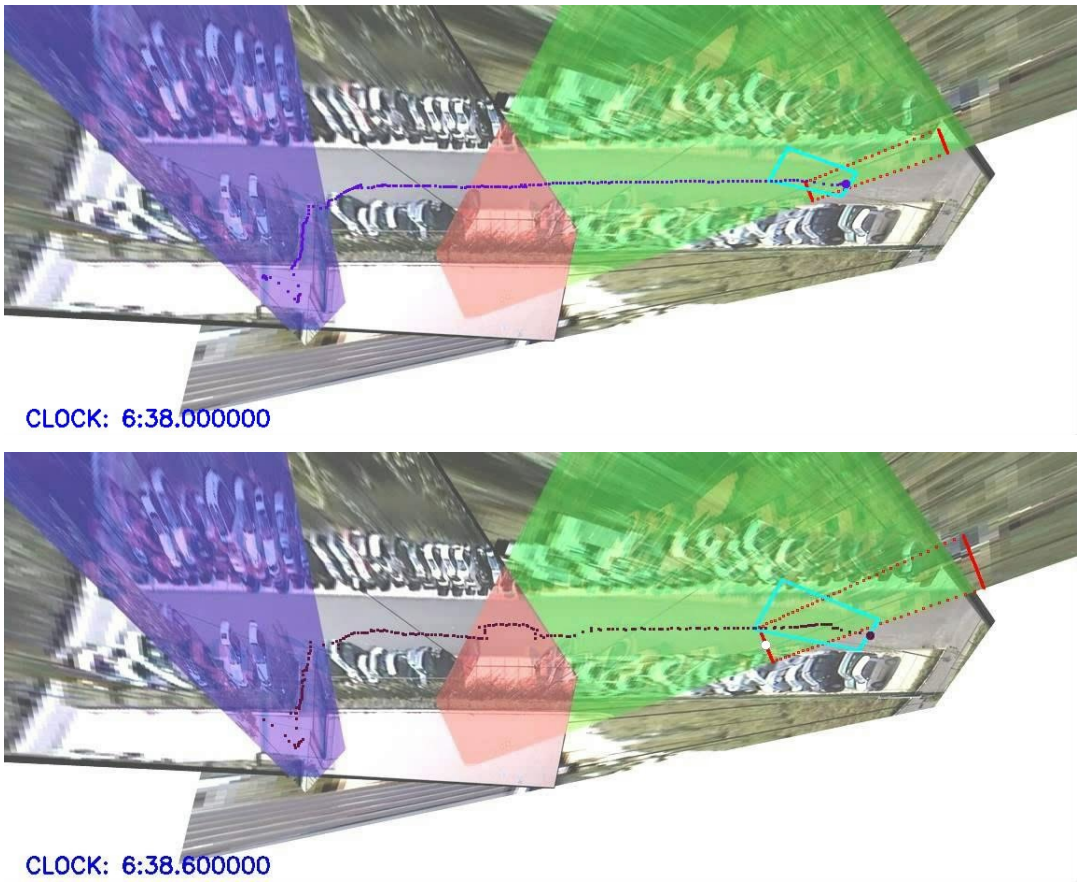


Figure 5-17 Example of trajectory accuracy by tracker2(top) and tracker1(bottom): a car (blue bounding box) moves across camera views

5.6 Discussion

In this chapter, a semi-automatic camera calibration method based on homography is proposed to calibrate multiple camera views into a single coordinate system. Compared with previous homography calibration methods, the idea of using a refining step and manually selected line features overcomes the problem of using points that directly from object detection which is prone to have errors. In addition, compared to traditional calibration method which uses manually selected feature points in the scene to calibrate cameras, line features are more visible and easy to pickup for human operators.

Objects' vertical axis is introduced for object correspondence between multiple overlapped camera views. Vertical axis is more robust against segmentation noise and occlusions compared to point based methods. Also, it can

be applied to scenes with mixture of pedestrians and vehicles compared with principle axis which can only be applied to pedestrians. Our method to estimate the vertical axis gives more accurate results when the camera parameters satisfy the condition that a vertical line on the ground plane projects to a line that approximately parallel (within 45°) to the image y-axis of the camera view which is true for most CCTV camera installations.

The proposed method has been tested using the SERKET multiple camera dataset and compared against a previous method which uses "foot" points for object matching between cameras. Some qualitative and quantitative evaluation results for both trackers are shown in section 5.5. It is demonstrated that the proposed method is capable of robust tracking on the ground plane in terms of avoiding false alarms and at the same time, maintaining high correct track detection rate. Also, vertical axes based object matching over performs foot point based method in aspects of more accurate object localization and tracking under occlusions.

For further work, there are a few possible directions for improving the tracking: appearance models (colour feature, or shape of object) can be used to provide more confident object matching across cameras, also a more complex model (e.g. rectangles, 3D models) for objects will help to achieve a more accurate localization of vehicles on the ground plane.

6 Non-coplanar Ground Model

In the previous chapter, a semi-automatic camera calibration method for a network of cameras based on homography mapping was proposed. In addition a method for object correspondence across cameras and object tracking on the common ground plane was suggested. However, in common with most of the literature, both camera calibration and object tracking assumed that all the activity observed by the network of cameras occurred on a single flat ground plane.

In this chapter, a novel method is proposed to extend the applicability of previous multi-camera tracking algorithms to a larger range of environments where objects are not constrained to move on a single coplanar ground plane (e.g. scenes where multiple levels exists such as stairs, ramps, overpasses and so on).

6.1 Introduction

In recent years, a significant amount of research effort has been put on 3D pedestrian tracking from single or multiple surveillance cameras. Most of the existing methods that perform 3D object tracking assume that object motion is coplanar and therefore pedestrian motion is constrained to a flat ground plane that is defined either manually or automatically from tracking observations. However, such a simple model is not able to handle scenes that contain multiple non-coplanar structures such as ramps, stairs and overpasses.

Researchers have tried to exploit the variation of sizes of tracked objects in surveillance. For instance, (Hoiem *at al*., 2008) proposed a probabilistic modelling of the scale and location variance of objects in the scene, thus they can build up a relationship between the size of objects and their positions so as to filter out false detections. (Saxena *et al*., 2009) assume that the world consists of vertical structures and a single flat ground surface. Then, a classifier is trained to model the relation between local material properties (colour and texture), 3D orientation, and image location. Other automatic ways for calibrating a ground plane from observed tracks of walking people are proposed by (Renno *et al*., 2002), (Krahnstoever and

Mendonca, 2006) and (Lv *et al*., 2006), which assume accurate measurements of head and foot positions for single pedestrians. (Rother *et al*., 2007) improved the previous methods by leaning a shadow model and as a result, obtaining more precise head and foot points of pedestrians which are then used to recover camera parameters. However, all the above methods can only deal with situations where all the objects move on a single coplanar ground plane.

(Breitenstein, *et al*., 2008) proposed an online learning approach for estimating a rough 3D scene structure from the outputs of a pedestrian detector. They divide the image into small cells and compute the relative depth for each image cell. However, their scene model is a depth map that does not explicitly represent the real 3D spatial dimensions of scene features.

Different from others, the method proposed in this work can estimate a scene model with non-coplanar planes by exploring the variation of pedestrian heights across the camera FOV in a statistical manner. The method can automatically segment the scene image into plane regions, estimate a relative depth and estimate the relative height (referred to as "altitude" throughout this chapter) for each image pixel, thus building up a 3D structure where multiple non-coplanar planes exist.

In this chapter, it is also demonstrated that scene structures can be estimated with sufficient accuracy. By being able to estimate the non-coplanar planes, the method can extend the applicability of multi-camera tracking algorithms to a range of environments where objects (pedestrians and/or vehicles) can move on multiple non-coplanar planes. In addition, the visualization of the multiple views on a common combined view will be more realistic, when the non-coplanar geometry is known. The main novelty of the proposed approach is the accumulation of evidence for the presence of different planar regions in the scene through pedestrian tracking, once enough tracks are available, a form of clustering is applied and each image pixel is associated with a cluster which defines a separate planar surface in the scene.

The framework for estimating a non-coplanar ground model (NCGM) is summarized in the following diagram.

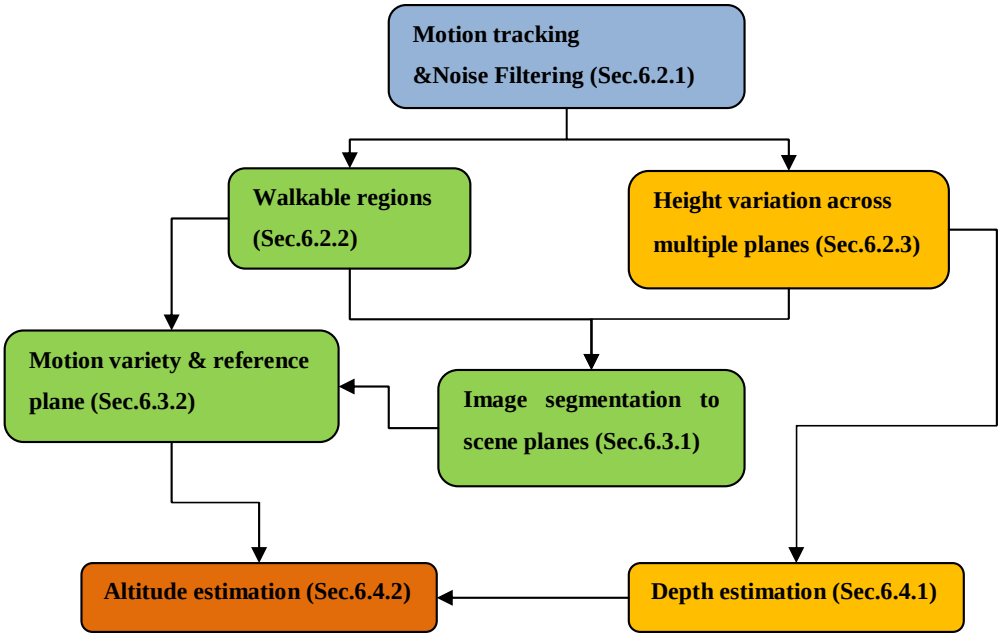


Figure 6-1 NCGM Framework overview

6.1.1 Camera projection model

In this work, a linear approximation model which assumes a linear relationship between the 2D image height of an object and its image vertical position is adopted, similar to (Greenhill *et al*, 2008). This object height model is derived from the typical CCTV view geometry illustrated in Figure 6-2.

$$h = R(y_B - H_L) \tag{6.1}$$

where h is the object 2D image height, y_B is the vertical image position of the detected object (foot position), H_L is the image y-coordinate of the horizon line and R is the object height expansion rate, a ratio that defines how object height h and its foot position y_B are related to each other. The object pixel height h is zero at the horizon H_L and is maximum at the bottom row of the image. This height projection model can be parameterised and updated by collecting observations of pedestrians walking through the scene. Note that this model can only be applied for objects moving on a single flat plane.

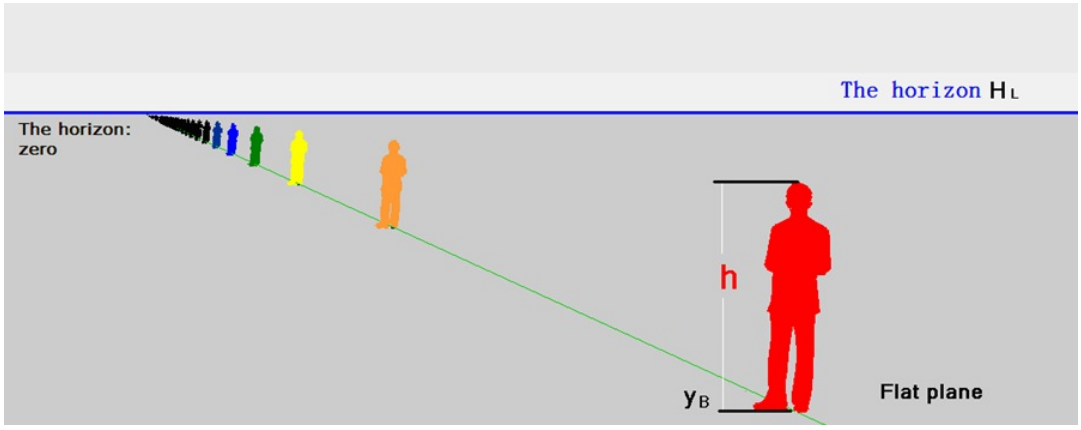


Figure 6-2 Camera projection model

This camera projection model assumes that the camera roll angle is zero so the horizon is parallel to the x-axis. When this is not the case, an image transformation can be applied to satisfy this condition. In addition, the rest of camera parameters (e.g. tilt angle, height, focal length) have appropriate values that allow the variation of objects sizes with respect to their y coordinates in a linear way. The above assumptions are typical for the majority of CCTV cameras.

6.1.2 Image patch model

The image plane is divided uniformly into patches $P_{m,n}$ to save computational load, where m and n is the row and column index of each patch:

$$P_{m,n} = \{ W_{m,n}, M_{m,n}, A_{m,n}, (c_{m,n}, d_{m,n}) \} \quad 6.2$$

where $W_{m,n}$ is a binary variable that indicates whether this image patch is walkable or not (Sec.6.2.2), $M_{m,n}$ is the average pedestrian height located in this patch (Sec.6.4.1), $A_{m,n}$ is the estimated altitude (Sec.6.4.2), and $(c_{m,n}, d_{m,n})$ are line parameters that indicate the relationship between pedestrian height and image vertical positions (Sec. 6.3.1).

6.2 Processing of track observations

6.2.1 Motion tracking

For each pedestrian in the scene, a track (or an observation) is derived by a blob tracking algorithm, (the BARCO tracker). For a pedestrian $j = [1..M]$, an observation (track) O_j is defined as:

$$O_j = \{ [x^{min}_{j,k}, x^{max}_{j,k}, y^{min}_{j,k}, y^{max}_{j,k}] \} \quad 6.3$$

where k is the frame number. The bounding box $[x^{min}_{j,k}, x^{max}_{j,k}, y^{min}_{j,k}, y^{max}_{j,k}]$ defines the object width $(W_{j,k} = x^{max}_{j,k} - x^{min}_{j,k})$ and height $(H_{j,k} = y^{max}_{j,k} - y^{min}_{j,k})$ and its centre bottom point $(B_{j,k}, C_{j,k})$. where $B_{j,k}$ is the lower y-coordinate of the bounding box $(B_{j,k} = y^{max}_{j,k})$ and $C_{j,k}$ is the middle x-coordinate $C_{j,k} = (x^{min}_{j,k} + x^{max}_{j,k})/2$.

In practice, before any further processing, it pays to filter tracks to remove unreliable measurements. For each track, the following steps are used to filter out unreliable bounding boxes:

- The image height of an object will be unreliable if it does not fully enter the camera FOV, therefore, bounding boxes that touch the borders of the image are discarded.
- Since the proposed method is currently based on average height of pedestrians, vehicles need to be filtered out since the height of vehicles can vary significantly (from 1.4 to a few meters). Based on the assumption that the shape of vehicles is generally more horizontal, bounding boxes whose width is larger than the height ($W_{i,j} / H_{i,j} > T_{HW}$) will be discarded. T_{HW} may change by estimating the average ratio from a large number of observations. Of course, a bounding box which contains a group of people may be discarded at the same time.
- Bounding boxes which are partly or totally occluded are discarded since their sizes are not reliable any more (occlusion status is derived by the motion tracker).
- The LOWESS (Locally Weighted Scatterplot Smoothing) method proposed by (Cleveland and Devlin, 1988) is used to smooth the sequence of bounding boxes' heights for each track. The smoothness of the curve depends on the size of the regression window. Here, 20% of the length of each track is chosen as the window size. Therefore if there are a few frames with segmentation errors, the system will still be able to correct and make them closer to the true value. Examples of smoothed tracks are shown in Figure 6-3 and Figure 6-4.

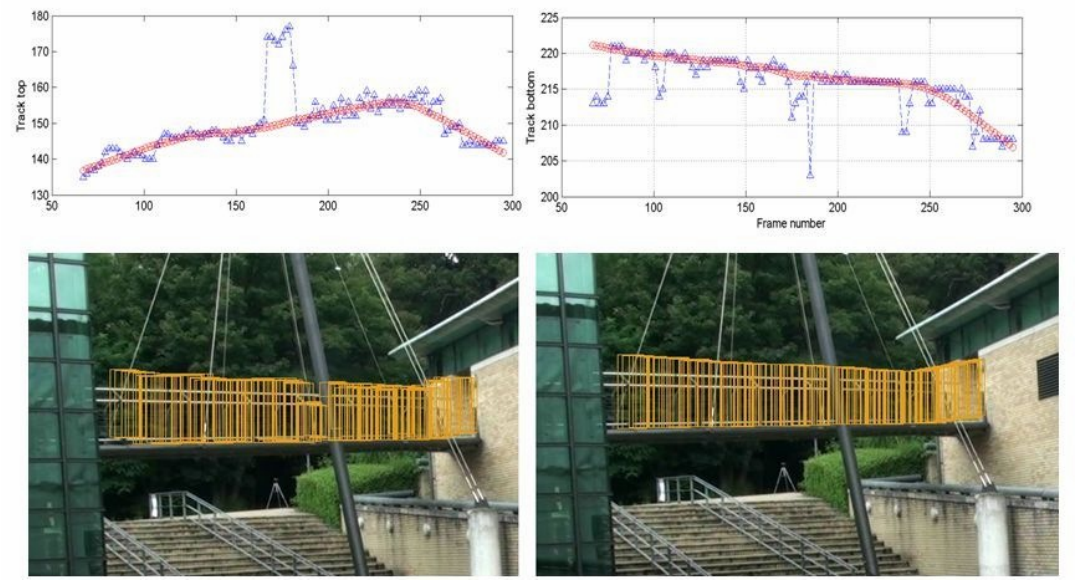


Figure 6-3 Top: y^{min} (left) and y^{max} (right) of track O_{14} before(blue) and after(red) smoothing Bottom: bounding boxes before (left) and after (right) smoothing

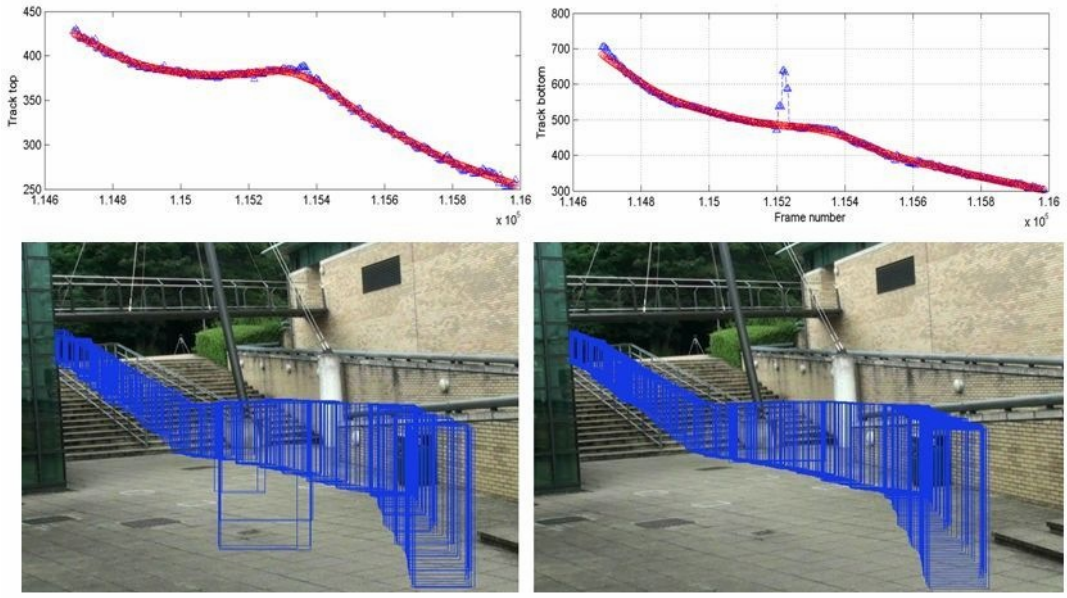


Figure 6-4 Top: y^{min} (left) and y^{max} (right) of track O_{162} before(blue) and after(red) smoothing Bottom: bounding boxes before (left) and after (right) smoothing

6.2.2 Walkable regions

For a given scene, people normally appear on regions which can be called "walkable" (e.g. not on walls or buildings). Detecting where people appear can help us to identify walkable regions in the camera FOV.

A patch is walkable if the number of observations ($B_{j,k}$,$C_{j,k}$) located inside a patch is above a threshold T_w. Then, by a connected component analysis, image patches are grouped and labeled as walkable regions (see Figure 6-12). Walkable regions will be further segmented in Sec.6.3.1.

6.2.3 Height variation across multiple planes

The linear camera projection model Eq.6.1 is valid if objects move on a single flat plane. However, this is not true for scenes that contain ramps or stairs. Figure 6-5 shows that when a pedestrian moves across different planes (at the boundary between the flat area and the stairs at around y=480), there is a noticeable change in object height expansion rate (slope of the object height/image y-axis plot) which means that there are different height expansion rate for different planes.

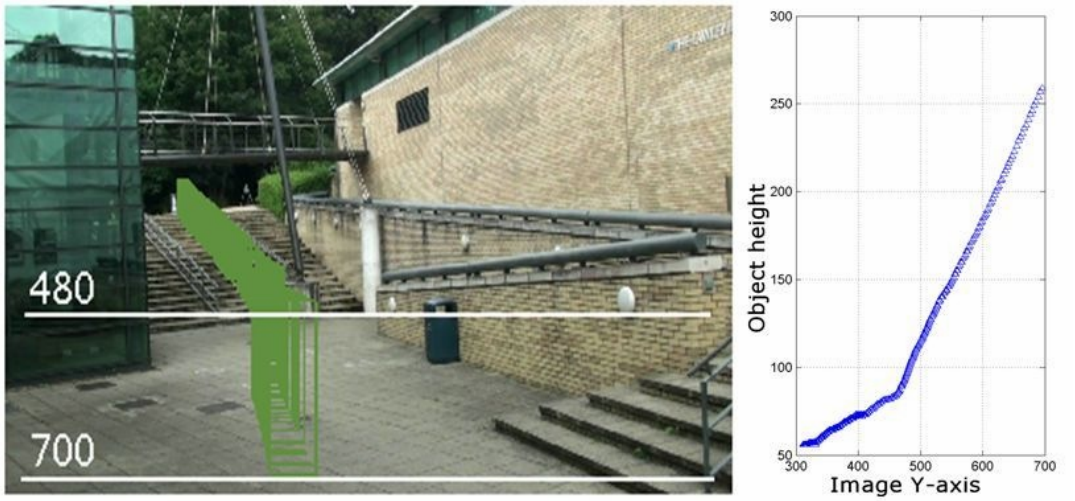


Figure 6-5 Bounding boxes of a tracked pedestrian j (left) and the relationship between object heights $H_{j,k}$ and vertical position on the image of $B_{j,k}$ (right)

An approach inspired by the Hough transform is applied to detect the slope change and consequently determine the number of planes for each walkable region. Let's assume that the frame span of a track O_j is from $K_{j,o}$ to $K_{j,p}$. Firstly, the track is divided uniformly in time into N parts. Each track segment i (i = *[1…N]*), consists of a set of points $Q_i = \{B_{j,k}, H_{j,k}\}$,

where $k = \left[K_{j,o} + \frac{(i-1)(K_{j,p} - K_{j,o})}{N}, \ldots \; K_{j,o} + \frac{i(K_{j,p} - K_{j,o})}{N} \right]$ is the frame index of Q_i, $\left(K_{j,p} - K_{j,o}\right)/N$ is the length of each track segment. Each point ($B_{j,k}$, $H_{j,k}$) reflects

the relationship between pedestrian heights and the vertical position on the image plane. Then, least square line fitting is performed for all points between $K_{j,o}$ and $K_{j,p}$. The line parameters $(c_{j,i},d_{j,i})$ are obtained in slope-intercept form, which minimize the average square distance from points to the line segment.

The i^{th} fitted line function for track j is:

$$H_{j,k} = c_{j,i}B_{j,k} + d_{j,i} \tag{6.4}$$

and the average square distance error is:

$$E = \sum_{k} \frac{\left(H_{j,k} - c_{j,i}B_{j,k} - d_{j,i}\right)}{\left(K_{j,p} - K_{j,o}\right)/N} \tag{6.5}$$

Therefore, for each track O_j, a set of line parameters $\{c_{j,i},d_{j,i}\}$ or equivalently $\{\theta_{j,i}\ S_{j,i}\}$ are obtained, where $\theta_{j,i}=arctanc_{j,i}$ is the angle between each line and the x-axis and $S_{j,i}= -d_{j,i}/c_{j,i}$.is the intercept. Each fitted line represents a linear relationship between the pedestrian height and the image vertical position or equivalently a plane that the pedestrian moves on.

For further analysis, a histogram of angles $\{\theta_{j,i}\}$ is obtained. Figure 6-7 shows that for pedestrians moving across planes, their height curves (object height vs image y-axis) will change in slope and more than one peak will occur in the histogram (left side of Figure 6-7). For pedestrians moving only on one of the planes (right side of Figure 6-7), their height curves will be a single line ideally, the variation of angles of fitted lines will be small and one peak will occur in the histogram.

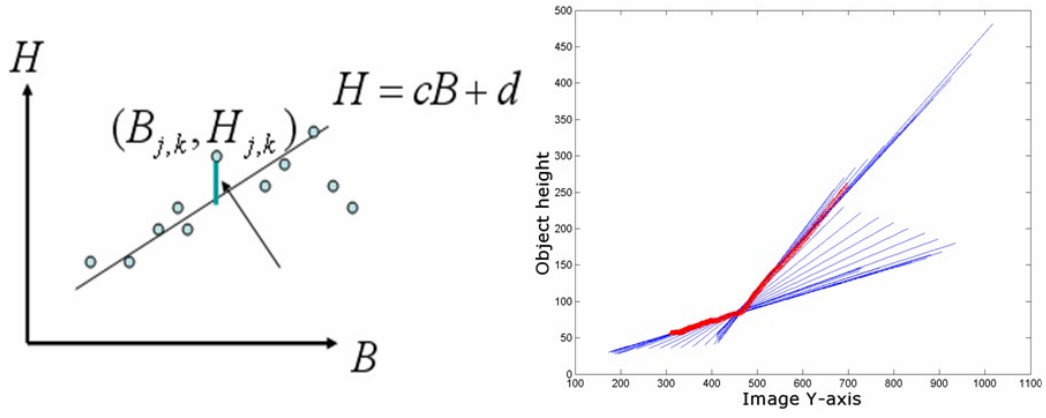


Figure 6-6 Least square line fitting (red: tracking data points, blue: fitted lines)

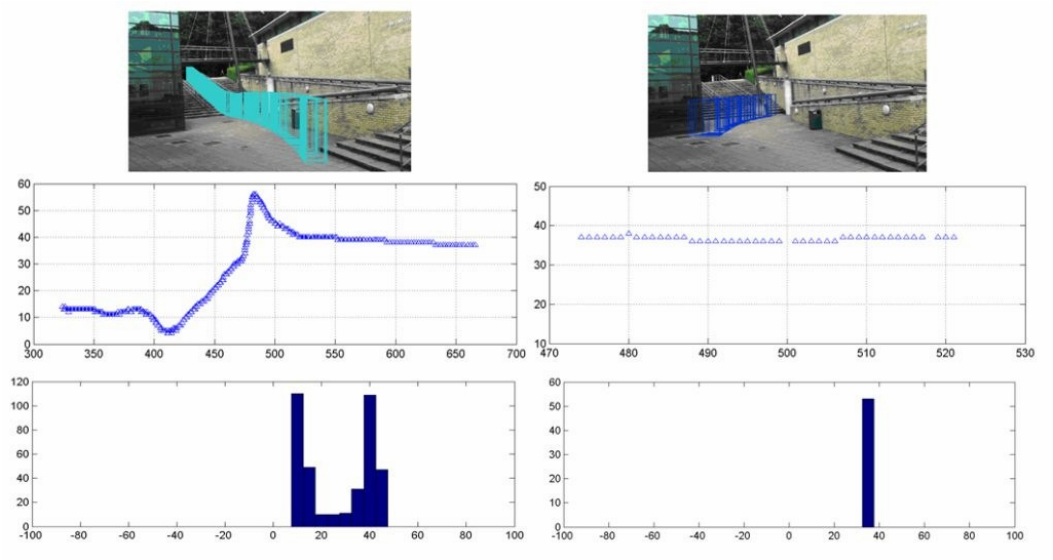


Figure 6-7 Example of line angles and their histograms (Top: bounding boxes of pedestrians, middle: angles, bottom: histogram of angles)

Finally, the histogram of angles from all tracks for a specific walkable region is obtained. After applying a moving average to smooth the histogram, all the peaks (local maxima) are found. Each peak corresponds to a plane in the scene and is described as a single Gaussian:

$$(\mu^{\theta}_i, \sigma^{\theta}_i, \mu^{S}_i, \sigma^{S}_i) \quad i = 1 \ldots N_{class} \tag{6.6}$$

where μ^{θ}_i, σ^{θ}_i, are the mean and standard deviation of angle $\{\theta_{j,i}\}$, and μ^{S}_i, σ^{S}_i is the mean and standard deviation of intercepts $\{S_{j,i}\}$ for each class i. N_{class} is the total number of classes for the given walkable region.

6.3 Image Segmentation to scene planes

6.3.1 Segmentation of walkable region

After the number of planes (classes) for a given walkable region have been estimated as described in the previous section, all image patches are classified into different planes. The steps to segment a walkable region into planes are summarized as follows:

1. For each image patch $P_{m,n}$ of the walkable region, all the tracked pedestrians $(B_{j,k}, H_{j,k})$ whose centre bottom points are located inside this patch are obtained (see Figure 6-8).

2. A least square line fitting algorithm is applied to obtain the line parameters $(c_{m,n}, d_{m,n})$ for this image patch. The angle between the line and the x-axis, $\theta_{m,n}$= *arctangent*$(c_{m,n})$ and the intercept $S_{m,n}=-d_{m,n}/c_{m,n}$, will then be used as a feature of this image patch in order to classify it into different planes.
3. A segmentation method similar to the one described in (Lin *et al.*, 2008) is applied. The image patch $P_{m,n}$ is labeled by the class (plane) i (Eq.6) that minimizes the difference:

$$\text{Arg} \min_{i\in[1,\,N_{class}]} \left[a \frac{(q_{m,n}-m_i^q)^2}{\left(s_i^q\right)^2} + (1-a)\frac{\left(S_{m,n}-m_i^S\right)^2}{\left(s_i^S\right)^2}, i \right] \qquad 6.7$$

   where $\theta_{m,n}$ is the angle feature for the image patch, and $S_{m,n}$ is the intercept for the image patch, and σ controls the combination weights between the two parts.
4. Due to noise, a few image patches get an incorrect label during step 3. To address this issue, the label of an image patch may change by minimizing the following cost function:

$$\text{Arg} \min_{i\in[1,\,N_{class}]} \left[\frac{(q_{m,n}-m_i^q)^2}{\left(s_i^q\right)^2} + b \sum_{o=m-c,k=n-c}^{m+c,n+c} \frac{h_{o,k}}{|q_{m,n}-q_{o,k}|} \right] \qquad 6.8$$

   where $\sum_{o=m-c,k=n-c}^{m+c,n+c} \frac{h_{o,k}}{|q_{m,n}-q_{o,k}|}$ takes the difference between this patch and its neighbour patches into consideration (assuming eight neighbours here), $h_{o,k}=0$, when $P_{m,n}$ and $P_{o,k}$ have the same label, $h_{o,k}=1$, when $P_{m,n}$ and $P_{o,k}$ have different label. The parameter β is set experimentally.
5. Step 4 is repeated until no change of class label is observed.

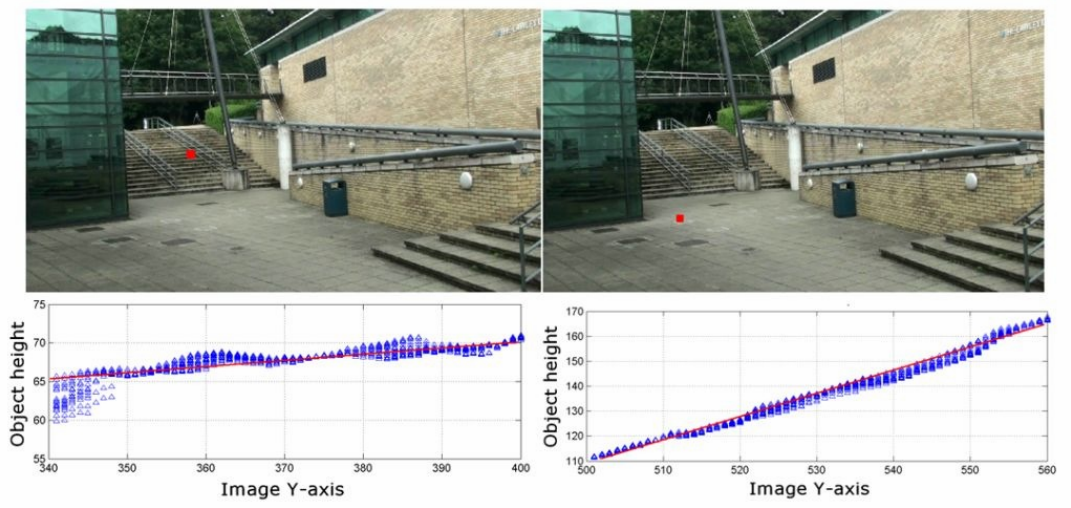


Figure 6-8 Examples of line features for image patches (the red rectangles)

6.3.2 Global motion variety

People are likely to move towards certain directions when they move on certain geometric structures. For example, people often follow the path on a bridge, also, go straight up or down on stairs statistically. Since their motion patterns can differ on different planes (e.g. the overpass, stairs, the ground), such difference are detected to distinguish between planes and define a reference plane. The statistics are computed to identify which direction each pedestrian takes and how many times: each time a pedestrian's centre bottom point $B_{j,k}$ is located within $P_{m,n}$, a motion vector is computed for the next few frames which indicates the direction of the pedestrian's motion., Then, all motion directions are accumulated to a histogram of motion directions, consisting of four predominant directions, as shown in Figure 6-9.

$$\{ V_i \}, \ i = [1,..N_V] \qquad 6.9$$

where i indicates the direction of motion ($N_V = 4$ in this work, see Figure 6-9 Four direction motion mode), and V_i is the count of the number of times pedestrians have taken that direction.

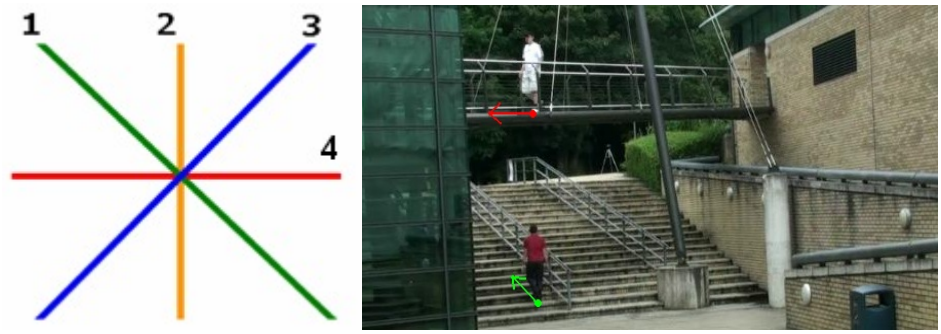


Figure 6-9 Four direction motion mode

The "motion variety" for each image patch $P_{m,n}$ is calculated as follows:

$$V_{m,n} = \left\{ V_1 \Big/ \sum_{i=1}^{N_V} V_i,\ V_2 \Big/ \sum_{i=1}^{N_V} V_i \ \ .....V_{Nv} \Big/ \sum_{i=1}^{N_V} V_i \right\} \qquad 6.10$$

Then, a reference plane needs to be chosen arbitrarily and the rest of the planes are defined relative to this reference plane. The region with largest motion variety is chosen as the reference plane. Although this reference plane is not necessarily the flat ground plane, it is more likely to be a plane parallel to the ground plane, as stairs and slopes tend to have smaller motion variety (see Figure 6-15, calculated motion varieties on different planar planes).

6.4 3D scene model estimation

6.4.1 Estimating average heights

A relative depth map is established by accumulating height observations of tracked objects for each image patch. The object heights of each patch are modeled with a single Gaussian to address the issue of noisy measurements. Specifically, for each patch $P_{m,n}$, the pedestrian height $H_{j,k}$ information is obtained, whose $(B_{j,k}, C_{j,k})$ is located inside this patch. Then, the heights are modelled by a mean $M_{m,n}$ and standard deviation $D_{m,n}$. Note that some image patches will have very few or no observation at all which implies that those areas are not walkable by pedestrians, or just not sufficiently observed.

6.4.2 Altitude estimation

As the reference plane has been chosen in section 6.3.2 and the pedestrian height information for each image patch has also been obtained in section 6.4.1, the next step is to estimate the relative altitude for each image patch in the scene with regard to the reference plane.

As illustrated in Figure 6-10, for each image patch (red rectangles), a mean pedestrian height $M_{m,n}$ is obtained as mentioned in section 6.4.1. One can always find a position with the same pedestrian height somewhere on the reference plane using Eq.6.11. The $y^r_{m,n}$ can be called the reference vertical position (green rectangles).

$$y^r_{m,n} = M_{m,n} / R_r + y_h \qquad 6.11$$

The expansion rate R_r and the horizon y_h (where the pedestrian height is zero) for the reference plane is estimated using the line fitting method mentioned in section 6.2.3.

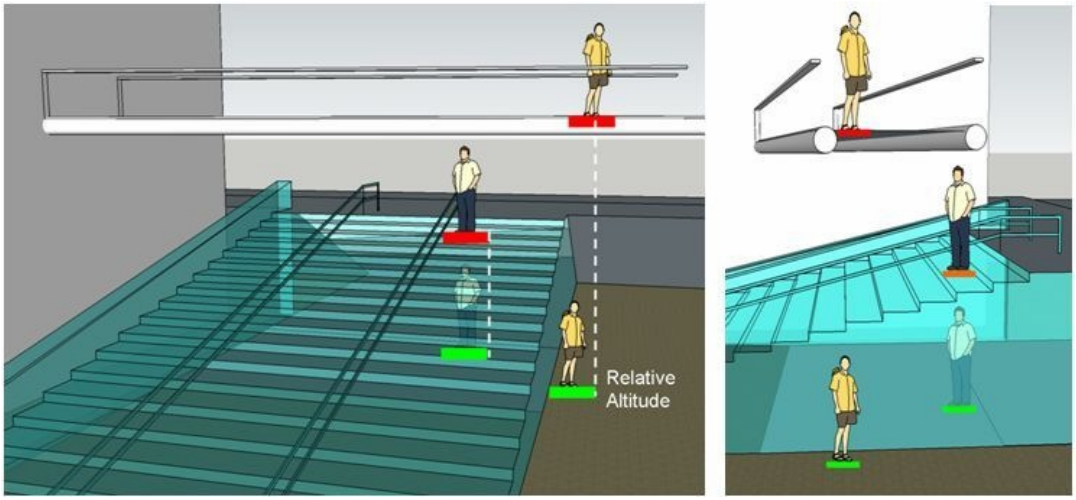


Figure 6-10 Illustration of how altitude has been estimated

If there is a difference between the vertical position of the image patch and the reference vertical position $y^r_{m,n}$, this indicates that the image patch may not be located on the reference plane but on other planes that are higher or lower than the reference plane. The relative altitude $Ar_{m,n}$ is estimated for the image patch $P_{m,n}$ by taking the difference of vertical positions and normalizing it by the average pedestrian's height $M_{m,n}$:

$$Ar_{m,n} = (y_{m,n} - y^r_{m,n}) / M_{m,n} \qquad 6.12$$

Finally, based on the assumption of average pedestrians heights of H_{av} (e.g. 1.70 meters), the altitude of each image patch $P_{m,n}$ can be converted into metres.

$$A_{m,n} = Ar_{m,n} \cdot H_{av} \qquad 6.13$$

6.5 Dataset and results

6.5.1 Kingston Hill dataset

The dataset used in this work is called Kingston Hill dataset which was captured in the Kingston Hill campus of Kingston University, London and can be found online at: http://dipersec.kingston.ac.uk/NCGMdata. It is a multiple camera dataset with two cameras monitoring roughly the same area and time synchronized. These videos were recorded by HD cameras. The image resolution is 1280×720. The dataset contains several hours of videos with pedestrians moving around frequently (with low object density in the scene). There are non-coplanar structures in the scene such as stairs and overpass.

To the best of our knowledge, there is no existing public surveillance dataset which deals specifically with scenes of multiple non-coplanar planes. Therefore, the dataset will be published to allow researchers to work on tracking in multi-planar environments and compare results.

Figure 6-11 Camera one and camera two of Kingston Hill dataset

6.5.2 Results and evaluation

In order to verify our method, the proposed algorithm is tested on the Kingston Hill dataset described above and also on the benchmark PETS2001 dataset. The frames from HD videos are divided into regular 10pix×10pix patches. A motion tracker is

used to obtain the position and size of each pedestrian when they walk through the scene and more than 400 tracks are obtained.

The results of grouping the camera FOV into walkable regions are shown (Sec.6.2.2) in Figure 6-12.

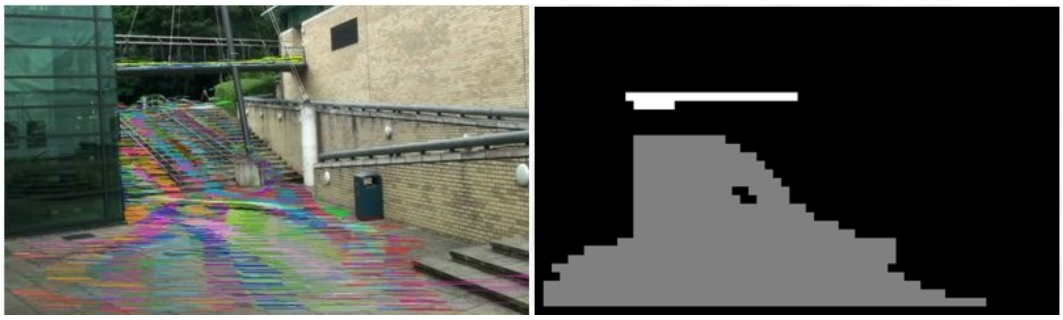


Figure 6-12 Walkable regions, different colours represent different tracks (left) and grouped walkable regions for camera 1(right)

Figure 6-13 shows the result of the histogram of angles of all the tracks for each walkable region (Sec.6.2.3). Figure 6-14 shows the intermediate and final scene segmentation results respectively (Sec.6.3.1). At this stage, the walkable region (1) was split to a flat area (red) and the stairs (green).

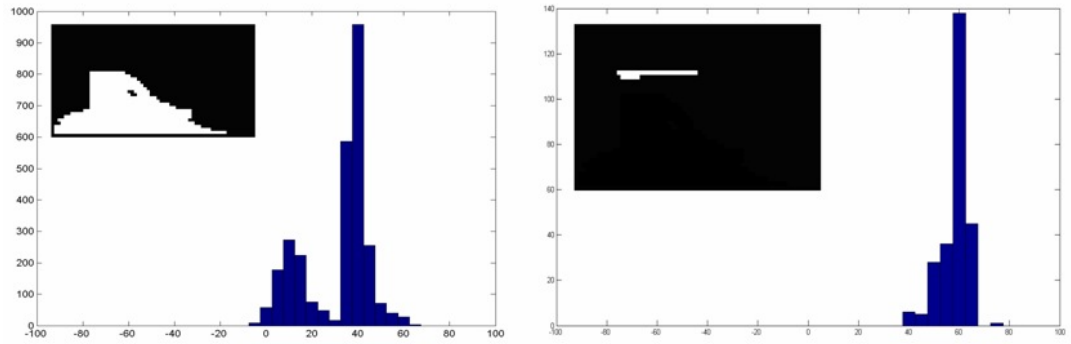


Figure 6-13 Histogram of angles for each walkable region

Figure 6-14 Intermediate (left) and final (right) segmentation result for camera 1

Figure 6-15 shows the global motion variety for different planar regions for camera view one (Sec.6.3.2). One can see that the motion vectors on the overpass are very clear and uniform (mainly on direction 0). Motion on the stairs is fairly uniform (mainly on direction 1). However, on the flat area, the motion vectors are less uniform, therefore, motion variety will be larger.

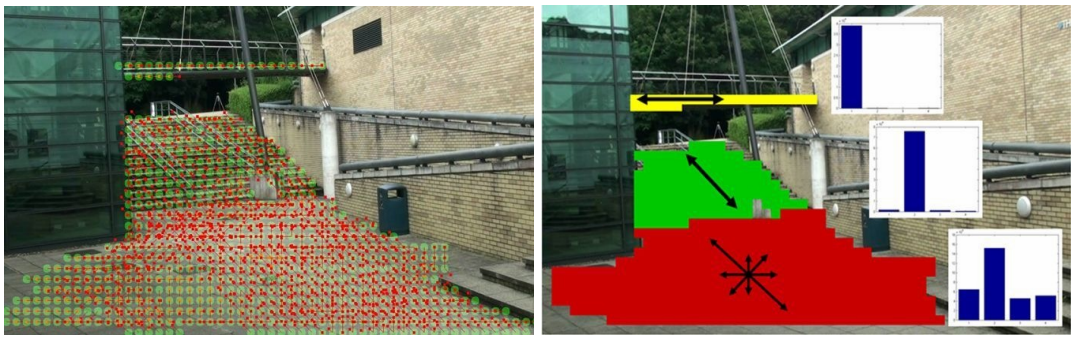


Figure 6-15 Local (left) and global (right) motion variety for camera 1

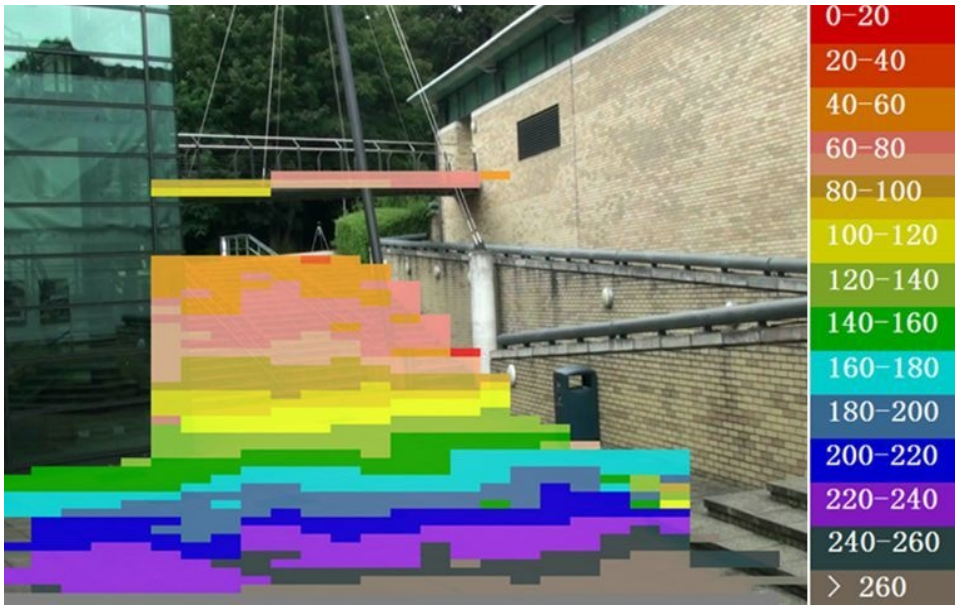


Figure 6-16 Average pedestrian height for each image patch for camera 1

Figure 6-16 shows a relative depth map based on average pixel-wise pedestrian height for each image patch where different colours represent different pedestrian heights in pixels (Sec.6.4.1).

Finally, Figure 6-17, Figure 6-18 and Figure 6-19 show the results of estimated altitude for each image patch of both camera views. The x, y axes are the image coordinates and the z axis is the estimated altitude. A rough 3D structure of

the scene can be seen: the flat area, the stairs and the overpass which is higher than the stairs.

In order to evaluate the accuracy of the altitude estimation, the real sizes of the stairs and overpass have been measured. There are 19 steps, the first 18 of them are 18cm in height, and the last step is 16cm in height. Hence the height of the stairway is 3.4 meters in total. The height of the overpass is 5 meters. In table 1, the estimations of the heights of the stairs and the overpass for the first camera view are 3.3 and 5.1 meters respectively (3.5 and 5.0 for the second camera view). Therefore, the proposed method estimated accurately (overall error: less than 0.1 metre, see Table 6-1) the real altitude for 3D scene structures.

	Ground truth	Camera 1	Camera 2
Overpass	5.0	5.1	5.0
Stairs	3.4	3.3	3.5
Flat area	0	0	0.1

Table 6-1 Evaluation of altitude estimation in meters

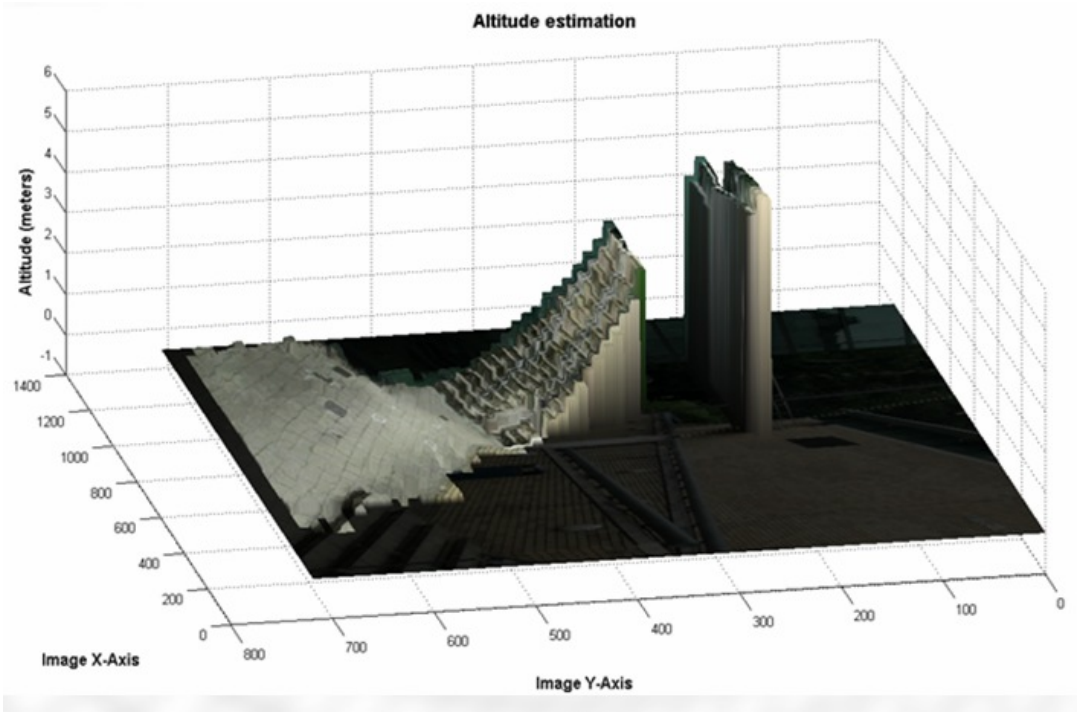


Figure 6-17 Estimated attitude for each image patch for camera 1

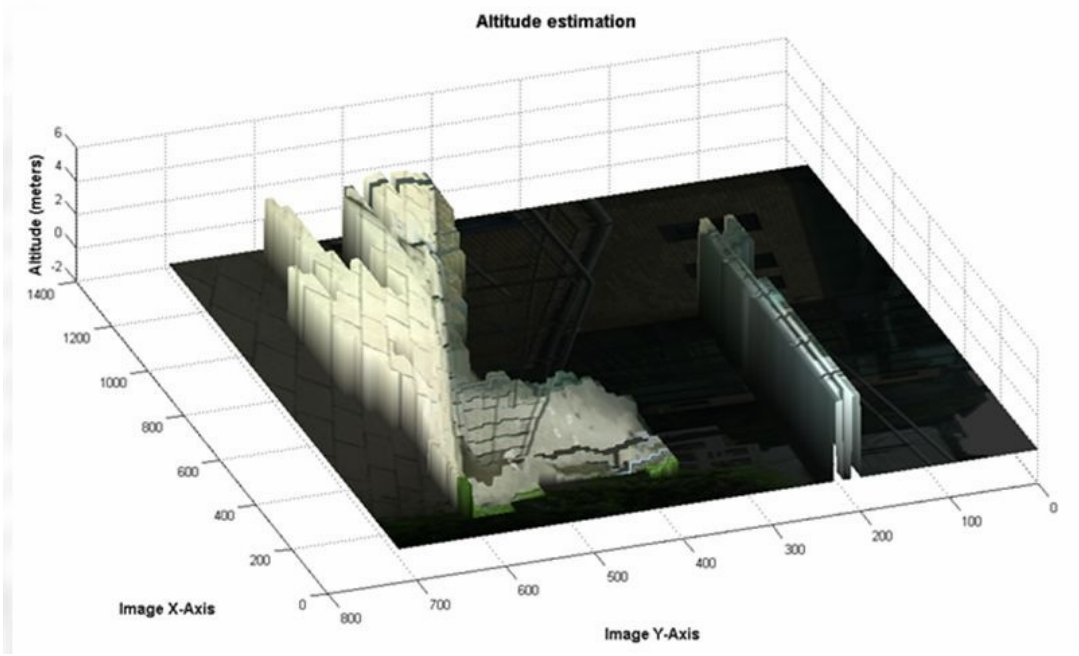


Figure 6-18 Estimated attitude for each image patch for camera 2

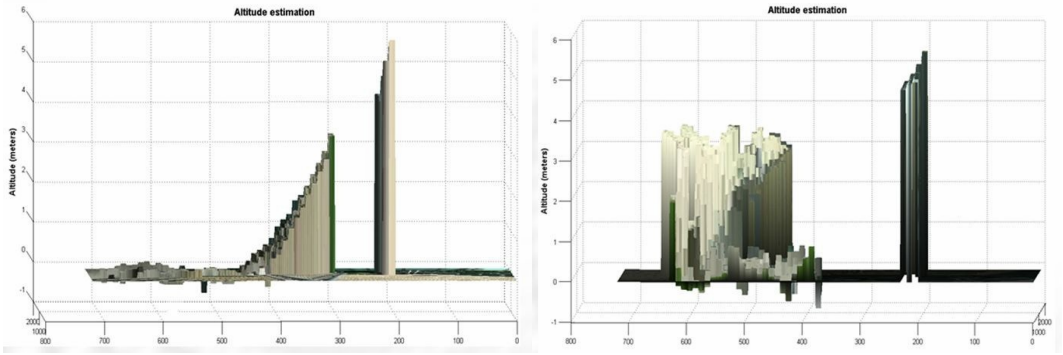


Figure 6-19 Side view for camera 1(left) and camera 2 (right)

Besides the author's own dataset, the proposed method is also tested on the well known PEST2001 dataset which depicts a flat ground area with people moving around frequently. From figure 6-20, we can see that the overall results of estimating the altitude for each pixel (within the walkable area in the scene image) is sufficiently accurate (0.05m on average). However, because of segmentation errors (especially for the far away part of the image where pedestrians are less than 30 pixels high) and occlusions in the car park area, there are a few pixels whose depth and altitude are not correctly estimated. As long as the tracker can provide

accurate bounding boxes, the proposed method will be able to estimate the altitude for 3D scene model more accurately.

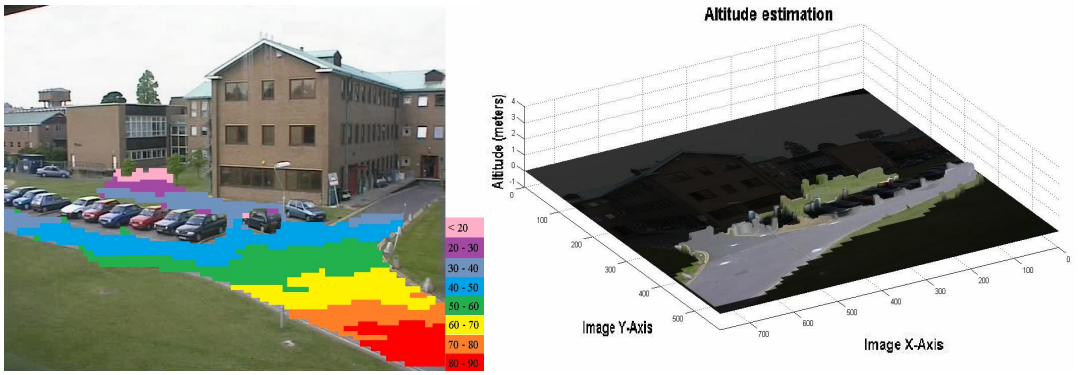


Figure 6-20 Average pedestrian height and Estimated attitude for each image patch for PETS2001 dataset

6.6 Discussion

With typical CCTV cameras monitoring scenes which contain non-coplanar structures such as overpass and stairs, a novel method is proposed to automatically estimate a non-coplanar scene model by statistically exploring the variation of pedestrian heights across the camera FOV. The proposed method is able to find out the relative depth, segment the image plane into regions which belong to the same geometric coplanar plane, identify a reference plane and estimate the altitude for each image pixel, thus building up a 3D scene model which contains multiple non-coplanar planes. The proposed method works on both the author's K.hill dataset (scene with multiple levels) and the well known PETS2001 dataset (scene with a flat ground plane). It is also demonstrated that the estimation of altitude is sufficiently accurate.

For future work, a more complete model can be built up which reflects the real world scale of the scene structures by taking the real scale of pedestrians into consideration. Also, more work can be done to not only produce more accurate 3D representations but also map each of the 2D image views into a common 3D scene model which will allow multiple-camera tracking in wide area non-coplanar environments.

7 Conclusion

7.1 Summary

The work presented in this thesis aimed to achieve a more robust, flexible and effective multi-camera tracking by solving problems that previous systems had and systematically evaluating the performance. Generally speaking, multi-camera tracking comprises three main aspects: camera calibration/scene modelling, detection and tracking within the camera FOV, correspondence/tracking between camera views. Algorithms that have been presented in this thesis can bring multiple benefits to these three fundamental aspects of multi-camera tracking systems.

For ease of reading, the research is summarised in the order in which it was presented in the thesis. The next section will briefly list the contributions of the work in this thesis and an outlook for further work will be provided in section 7.3.

7.1.1 Performance evaluation

In chapter 3, a comprehensive track based evaluation framework has been proposed. The framework allows the developers to identify specific weaknesses of trackers, such as the performance of specific modules or failures under specific conditions. Furthermore, it allows the developer to quantitatively measure the improvement that has been made to the tracker. This is an essential step to determine the reliability of the system before it can be developed for real world applications.

In the quantitative experiments presented in chapter 3, two trackers (OpenCV blob tracker and BARCO tracker) have been evaluated and compared using over 30,000 frames of videos (part of i-LIDS dataset and a few industrial datasets which were ground truthed by the author). For performance evaluation, longer and more comprehensive video data for testing is desirable, but comes with the high cost of ground truth generation. Overall, a good attempt has been made on evaluation, especially in terms of measuring the improvement that has been done for specific modules of the BARCO tracker which was presented in chapter 4.

In addition, the proposed track evaluation metrics have not only been used in this work but also in many other projects within the research group.

7.1.2 Single camera tracking

In chapter 4, three new modules (ghost elimination, shadow removal and improved Kalman filter) have been introduced to improve the performance of BARCO tracker. The tracker with and without those new modules has been tested on a set of urban video sequences and the performance has been accessed through the track based metrics to quantitatively measure the improvements that have been made by adding these new modules.

In that context, the main contribution is a novel method to deal with the problem of "ghosts" that appear when using foreground/background separation methods. The proposed ghost detection algorithm is fast and effective in detecting "ghosts" and can be easily integrated to any motion detection system that estimates a background model and a difference map between the current frame. The ghost removal algorithm has been tested using a set of urban traffic video sequences that would normally cause ghost problems. Results showed that the ghost removal algorithm is able to eliminate all ghosts, at the same time, differentiate them from abandoned immobile items, and does not bring any negative side-effects or much more computational cost to the tracker.

7.1.3 Multi-camera tracking

In chapter 5, a method has been proposed to automatically calibrate a network of cameras to a single view map based on homography with limited human intervention. Vertical axis has been introduced for object correspondence between camera views.

The proposed method was tested using the SERKET multiple camera dataset (urban traffic dataset from the project sponsor), and both qualitative and quantitative results have been shown in chapter 5. It has been demonstrated that the proposed vertical axis based method is more against segmentation noises and occlusions compare to point based methods and at the same time, maintaining a high correct tracking rate.

7.1.4 Non-coplanar ground model

In chapter 6, a novel statistical approach has been proposed to learn a non-coplanar ground model. The proposed method is able to automatically estimate the depth and altitude for each image pixel by learning the variation of pedestrian heights within the scene, thus building up a 3D structure with multiple non-coplanar planes. This overcomes the constraint of a single coplanar ground of all the previous camera calibration/scene modelling methods, and could lead to a whole new set of applications where the single coplanar ground plane assumption does not hold (e.g. when tracking between floors of a building, stairs, overpasses etc).

Due to the lack of a public available dataset, the proposed method was tested on the author's own dataset (referred to as "Kingston Hill dataset") which was captured by the research group in Kingston University. The results shown in chapter 6 demonstrated that the proposed algorithm is sufficiently accurate on estimation of altitude for the scene structures.

7.2 Contributions

A brief of the main contributions of the thesis is presented here.

- Developing a track based evaluation framework to assess different aspects of tracking systems (such as motion segmentation, motion tracking and data association) and to identify failures or improvements of specific modules of tracking systems.
- Developing a quick and effective method for ghost identification and elimination based on edge comparison.
- Developing a quick and effective homography based method of calibrating a set of cameras (with overlapping view) to a single view map with the absence of plane map for the whole site.
- Developing a multi-camera tracking method which uses vertical axes for object correspondence between cameras. The proposed vertical axes based method is more robust against segmentation noise and occlusions compared to point based methods.

- Developing a novel method that can automatically learn a non-coplanar ground model by exploring the height variation of tracked objects. The proposed method overcomes the single flat ground plane constraint of previous methods.

7.3 Future research

The problem of multiple camera surveillance is complex and highly application dependant which the human visual system can solve in many cases with relative ease. Clearly, computer vision based algorithms are far from perfect compared to human perception for object recognition, identification and scene understanding. Nevertheless, there is great potential for improvements.

The work that has been done in this thesis touches several aspects of visual surveillance: evaluation of object tracking algorithms, single view tracking and scene modelling, multiple camera calibration and object tracking. In this section, some directions to extend and improve the work in this thesis are proposed.

7.3.1 Multi-camera tracking

In future work, it should be possible to develop a fully automatic method which does not require the user to manually define line features of the scene. A fully automatic method will make the calibration more effective and faster for the operators.

Spatial temporal cues are used to coordinate object tracking between multiple camera views. However, the proposed vertical axis is still not sufficient to describe large size vehicles (e.g. bus, lorry). Therefore, a more complex geometry model (rectangles, polygons, or 3D models) can be learned and used for vehicles. In addition, when it comes to non-overlapping cameras, the vertical axis which based on geometry location cannot be used for object matching at all. Therefore, alternative methods can be added to the system to match objects between camera views. Assuming that colour calibration between cameras is available, then it is possible to use appearance cues as an additional method to improve object

matching. Alternatively, a time transition model can be used to hand over objects across cameras, or both can be used to enhance the system.

7.3.2 Non-coplanar ground model

The proposed method which estimates a non-coplanar ground model in terms of estimating the altitude for each pixel in meters did not measure the real depth of scene structures in meters. Therefore, a more complete 3D model can be built up which reflects the real geometry depth of the scene structures by taking the scale (e.g. average width) of pedestrians into consideration. Also, more work can be done to map each of the camera views (overlapping or non-overlapping, also, more dataset is desirable) into a common 3D scene model which will allow multiple-camera tracking in wide area non-coplanar environments. Thus, the view of 3D scene will be more realistic and additional geometry information (e.g. which level or what altitude the object is moving at) can be used to help tracking in a 3D non-coplanar environment.

7.4 Epilogue

Visual surveillance and monitoring is an active topic that has been investigated by the computer vision research community for decades. This thesis has presented new methods and work which can bring multiple benefits to the research field.

A track based performance evaluation framework was proposed to access different aspects of visual surveillance systems and quantitatively measure the tracking performance. This will allow practitioners and researchers to reason about failures of a specific module of their tracking systems and measure the improvements they have made to their systems.

Three new modules (Ghost elimination, shadow removal and improved Kalman filter) have been added to an industrial tracker (the BARCO tracker) and the improvements or any possible deterioration was measured by the track based evaluation framework.

A semi-automatic homograpy based camera calibration method was proposed which can calibrate multiple camera views with overlapping onto a single view map with limited manual intervention. In addition, the proposed vertical axes based object correspondence method can help to achieve more accurate and robust tracking.

A novel method was proposed to automatically learn a non-coplanar ground model where scene activities occur on multiple levels (e.g. stairs, overpass). The proposed method can greatly extend the applicability of previous object tracking systems to a larger range of environments where objects are not constrained to move on a single flat ground plane.

The presented work was motivated not only by the numerous potential applications in visual surveillance, but also by the ambition to build an automatic computer vision system that senses, learns and understands its environment and scene activities with a minimal support from human operators.

A. Tracker Parameters

A.1 parameters of OpenCV blobtracker 1.0

Parameters:
blobtrack [fg=<fg_name>] [bd=<bd_name>]
[bt=<bt_name>] [btpp=<btpp_name>]
[bta=<bta_name>
[bta_data=<bta_data_name>
[bt_corr=<bt_corr_way>]
[btgen=<btgen_name>]
[track=<track_file_name>]
[scale=<scale val>]
[noise=<noise_name>]
[IVar=<IVar_name>]
[FGTrainFrames=<FGTrainFrames>]
[btavi=<avi output>]
[fgavi=<avi output on FG>]
<avi_file>
<bt_corr_way> is way of blob position corrrection for "Blob Tracking" module
<bt_corr_way>=none,PostProcRes
<FGTrainFrames> is number of frames for FG training
<track_file_name> is file name for save tracked trajectories
<bta_data> is file name for data base of trajectory analysis module
<avi_file> is file name of avi to process by BlobTrackerAuto

Modules:
<fg_name> is "FG/BG Detection" module name and can be:
1. FG_0 - Foreground Object Detection from Videos Containing Complex Background. ACM MM2003.
2. FG_0S - Simplyfied version of FG_0

3. FG_1 - Adaptive background mixture models for real-time tracking. (Stauffer and Grimson, 1998)

<bd_name> is "Blob Entrance Detection" module name and can be:

1. BD_CC - Detect new blob by tracking CC of FG mask
2. BD_Simple - Detect new blob by uniform moving of connected components of FG mask

<bt_name> is "Blob Tracking" module name and can be:

1. CCMSPF - connected component tracking and MSPF resolver for collision
2. CC - Simple connected component tracking
3. MS - Mean shift algorithm
4. MSFG - Mean shift algorithm with FG mask using
5. MSPF - Particle filtering based on MS weight

<btpp_name> is "Blob Trajectory Post Processing" module name and can be:

1. Kalman - Kalman filtering of blob position and size
2. None - No post processing filter

<btgen_name> is "Blob Trajectory Generation" module name and can be:

1. YML - Generate track record in YML format as synthetic video data
2. RawTracks - Generate raw track record (x,y,sx,sy),()... in each line

<bta_name> is "Blob Trajectory Analysis" module name and can be:

1. HistPVS - Histogramm of 5D feture vector analysis (x,y,vx,vy,state)
2. None - No trajectory analiser
3. HistP - Histogramm of 2D feture vector analysis (x,y)
4. HistPV - Histogramm of 4D feture vector analysis (x,y,vx,vy)
5. HistSS - Histogramm of 4D feture vector analysis (startpos,endpos)
6. TrackDist - Compare tracks directly
7. IOR - Integrator (by OR operation) of several analysers

The values of parameters of OpenCV blobtracker that have been used for performance evaluation are:
blobtrack.exe fg=FG_1 bd=BD_CC bt=CCMSPF btpp=Kalman btgen=RawTracks bta=HistPVS (Table 3-1, Table 3-2, Table 3-3, Table 3-4, Table 3-5, Table 3-6, Table 3-7 column 1)

blobtrack.exe fg=FG_1 bd=BD_CC bt=CCMSPF btpp=Kalman btgen=RawTracks bta=HistPVS (Table 3-7 column 2)

blobtrack.exe fg=FG_1 bd=BD_CC bt=CCMSPF btpp=None btgen=RawTracks bta=HistPVS (Table 3-7 column 3)

blobtrack.exe fg=FG_0S bd=BD_CC bt=CCMSPF btpp=Kalman btgen=RawTracks bta=HistPVS (Table 3-7 column 4)

blobtrack.exe fg=FG_1 bd=BD_Simple bt=CC btpp=Kalman btgen=RawTracks bta=HistPVS (Table 3-7 column 5)

A.2 Parameters of the BARCO tracker

Descriptions and values of parameters of the BARCO tracker used for performance evaluation are shown below:

Parameter=value	Type	Description
perform_AGC = true	Bool	Automatic Gain Control for fast illumination changes
time_constant = 50	Integer	Frequency of background learning, 50 means updating the background every 50^{th} frame
std_threshold_new = 2.6	Double	Threshold for foreground detection, which means the value of current pixel differs how many times from the

		background pixel
bord = 4	Integer	Defines the width (pixels) of the image border
dilation_length = 2	Integer	Number of pixels for morphological dilation on detected blobs
min_lifetime = 6	Integer	The frame number that a track exists before it becomes a real track
min_pixel_area = 15	Integer	The minimum number of pixels for a blob to be tracked
min_track_distance = 30	Integer	The distance (pixels) that a track has moved before it becomes a real track
max_ntrackpoints = 1000	Integer	Prevent the full scene detected as foreground, reset the system when detected pixels is larger than max_ntrackpoints
max_time_since_match_mov =30	Integer	Time (number of frames) that a track need to be deleted since fully occlusion
time_to_immobile = 150	Integer	Time (number of frames) that a track becomes stationary
time_to_vanish = time_to_immobile * 1	Integer	Time (number of frames) that a track can be deleted after it becomes stationary
max_occluded_distance = 100	Integer	The maximum distance (pixels) to track an object when it is occluded
merging_distance_norm = 20	Integer	Merge if the closest distance (pixels) between two tracks is smaller than a threshold
merging_distance_o	Integer	Merge distance (pixels) during occlusion

ccl = 25		
splitting_distance = 25	Integer	Split if the closest distance (pixels) between two blobs is larger than a threshold
var_pos = 0.01	Double	Kalman filter: Process noise for location (pixels²)
var_v = 0.25	Double	Kalman filter: Process noise for velocity (pixels/frame)²
var_z = 9.0	Double	Kalman filter: Measurement noise for location (pixels²)
v2_expected = 3.0	Double	Kalman filter: Initial error covariance (pixels/frame)²

B. Additional evaluation results

The track based evaluation framework proposed in chapter3 is further used to compare the performance of a 3D single camera vehicle classifier&tracker proposed by (Buch *et al.*, 2009) and the OpenCV blob tracker on more i-LIDS datasets.

The performance evaluation results are shown in the tables below. The 3D tracker generally outperforms the OpenCV tracker on high level metrics such as track detection failure, false alarm tracks and track fragmentation. Although the number of correct detections between the two trackers is similar, but the 3D tracker has much smaller number of false alarm tracks. This is mainly because the 3D tracker uses additional prior information (calibrated cameras, 3D vehicle models) to classify the content of the input video and reject many ambiguous detections.

For metrics that evaluate the segmentation, the 3D tracker also performs better than OpenCV tracker due to 3D models which are more accurate and more robust against shadows. Because of the complexity of the 3D models, it takes longer for the 3D tracker to detect and classify vehicles which lead to bigger latency of track. The 3D tracker is designed to be more persistent, occasionally wrongly continuing a track which lead to more ID change but therefore generating much less track fragmentations.

Figure B-1 Detection results (vehicles) in the regions of interest Left: the 3D tracker Right: OpenCV blobtracker

PVTRA101a03	**3Dtracker**	**OpenCV tracker**
Number of GT Tracks	12	12
Number of System Tracks	13	39
Correct Detected Track	12	10
False Alarm Track	0	18
Track Detection Failure	0	2
Track Fragmentation	1	12
ID Change	2	1
Latency of Track	20	11
Average Track Closeness	0.64	0.27
Deviation of Track Closeness	0.16	0.12
Average Distance Error	19.77	32.49
Deviation of Distance Error	17.11	18.88
Average Track Completeness	0.83	0.93

Table B-1 Evaluation results for i-LIDS PVTRA101a03

PVTRA101a07	**3Dtracker**	**OpenCV tracker**
Number of GT Tracks	8	8
Number of System Tracks	10	23
Correct Detected Track	8	8
False Alarm Track	2	16
Track Detection Failure	0	0
Track Fragmentation	0	0
ID Change	0	0
Latency of Track	16	6
Average Track Closeness	0.56	0.33
Deviation of Track Closeness	0.20	0.12
Average Distance Error	25.56	20.48
Deviation of Distance Error	49.96	6.88
Average Track Completeness	0.53	0.39

Table B-2 Evaluation results for i-LIDS PVTRA101a07

PVTRA101a13	3Dtracker	OpenCV tracker
Number of GT Tracks	10	10
Number of System Tracks	18	17
Correct Detected Track	7	8
False Alarm Track	4	9
Track Detection Failure	3	2
Track Fragmentation	1	0
ID Change	2	0
Latency of Track	14	13
Average Track Closeness	0.65	0.33
Deviation of Track Closeness	0.15	0.10
Average Distance Error	16.28	30.86
Deviation of Distance Error	12.78	27.19
Average Track Completeness	0.22	0.15

Table B-3 Evaluation results for i-LIDS PVTRA101a13

PVTRA101a19	3Dtracker	OpenCV tracker
Number of GT Tracks	9	9
Number of System Tracks	13	30
Correct Detected Track	9	9
False Alarm Track	0	21
Track Detection Failure	0	0
Track Fragmentation	1	0
ID Change	1	0
Latency of Track	10	6
Average Track Closeness	0.58	0.34
Deviation of Track Closeness	0.16	0.09
Average Distance Error	18.80	22.85
Deviation of Distance Error	6.73	16.15
Average Track Completeness	0.90	0.77

Table B-4 Evaluation results for i-LIDS PVTRA101a19

PVTRA101a20	3Dtracker	OpenCV tracker
Number of GT Tracks	11	11
Number of System Tracks	14	23
Correct Detected Track	11	11
False Alarm Track	2	11
Track Detection Failure	0	0
Track Fragmentation	1	1
ID Change	0	1
Latency of Track	13	4
Average Track Closeness	0.58	0.34
Deviation of Track Closeness	0.14	0.12
Average Distance Error	16.87	20.08
Deviation of Distance Error	7.75	12.68
Average Track Completeness	0.92	0.90

Table B-5 Evaluation results for i-LIDS PVTRA101a20

PVTRA102a05	3Dtracker	OpenCV tracker
Number of GT Tracks	20	20
Number of System Tracks	41	27
Correct Detected Track	17	17
False Alarm Track	16	8
Track Detection Failure	3	3
Track Fragmentation	3	3
ID Change	3	1
Latency of Track	18	7
Average Track Closeness	0.49	0.35
Deviation of Track Closeness	0.19	0.12
Average Distance Error	22.92	21.05
Deviation of Distance Error	11.02	15.49
Average Track Completeness	0.47	0.19

Table B-6 Evaluation results for i-LIDS PVTRA102a05

PVTRA102a10	**3Dtracker**	**OpenCV tracker**
Number of GT Tracks	20	20
Number of System Tracks	23	27
Correct Detected Track	20	17
False Alarm Track	2	7
Track Detection Failure	0	3
Track Fragmentation	1	2
ID Change	2	0
Latency of Track	19	7
Average Track Closeness	0.50	0.33
Deviation of Track Closeness	0.19	0.15
Average Distance Error	24.10	25.15
Deviation of Distance Error	10.03	18.04
Average Track Completeness	0.45	0.31

Table B-7 Evaluation results for i-LIDS PVTRA102a10

PVTRA102a15	**3Dtracker**	**OpenCV tracker**
Number of GT Tracks	10	10
Number of System Tracks	12	17
Correct Detected Track	10	8
False Alarm Track	1	9
Track Detection Failure	0	2
Track Fragmentation	1	0
ID Change	0	0
Latency of Track	16	9
Average Track Closeness	0.53	0.31
Deviation of Track Closeness	0.25	0.13
Average Distance Error	15.34	13.61
Deviation of Distance Error	7.95	8.35
Average Track Completeness	0.90	0.71

Table B-8 Evaluation results for i-LIDS PVTRA102a15

References:

(Bashir and Porikli, 2006) Bashir, F., and Porikli, F. (2006) Performance evaluation of object detection and tracking systems, IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, (PETS2006).

(Brown *et al.*, 2005) Brown, L. M., Senior, A. W., Tian, Ying-li., Jonathan Connell, Arun Hampapur, Chiao-Fe Shu, Hans Merkl, Max Lu, (2005) Performance Evaluation of Surveillance Systems Under Varying Conditions, IEEE Int'l Workshop on Performance Evaluation of Tracking and Surveillance, Colorado.

(Black and Ellis, 2005) Black J., Ellis T., (2005). Multi camera image tracking, in Image and Vision Computing, Vol 24, No.11, pp.1256-1267.

(Black *et al.*, 2004) Black, J., Ellis, T., Makris, D., (2004). Wide Area Surveillance With a Multi Camera Network, IEE Intelligent Distributed Surveillance Systems, London, pp. 21-25.

(Black *et al.*, 2005) Black, J., Makris, D., Ellis, T. (2005). Validation of Blind Region Learning and Tracking, Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation, pp. 9-16.

(Black and Ellis, 2002) Black J., Ellis T., (2002). Multi Camera Image Measurement and Correspondence, The Journal of the International Measurement Confederation (IMEKO), Vol. 32, No. 1, pp. 61-71.

(Borg *et al.*, 2005) Borg, M., Thirde, D. J., Ferryman, J. M., Fusier, F., Valentin, V., Brémond, F., Thonnat, M., Aguilera, J., Kampel, M., (2005). Automated Scene Understanding for Airport Aprons, The 18th Australian Joint

Conference on Artificial Intelligence (AI05) in Sydney, Australia, December, Vol. 3809, pp. 593-603.

(Boninsegna and Bozzoli, 2000) Boninsegna, M. and Bozzoli, A. (2000). A tunable algorithm to update a reference image. Signal Processing: Image Communication, 16(4): 353 – 365.

(Brown *et al.*, 2005) Brown, L. M. A., Senior, W., Tian, Y. l., Connell, J., Hampapur, A., Shu, C. F., Merkl, H., Lu, M. (2005). Performance Evaluation of Surveillance Systems Under Varying Conditions, IEEE Int'l Workshop on Performance Evaluation of Tracking and Surveillance, pp.1-8, Colorado.

(Bevilacqua, 2003) Bevilacqua, A. (2003). Effective shadow detection in traffic monitoring applications, Journal of WSCG, Vol.11, No.1., ISSN 1213-6972.

(Buch *et al.*, 2009) Buch, N., Yin, Fei., Orwell, James., Makris, D. and Velastin, S. A. (2009). Urban Vehicle Tracking using a Combined 3D Model Detector and Classifier. In 13th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES 2009, Part I, LNCS 5711, pages 169–176, Santiago, Chile.

(Breitenstein *et al.*, 2008) Breitenstein, M.D., Sommerlade, E., Leibe, B., Gool, L. V. and Reid, I. (2008). Probabilistic Parameter Selection for Learning Scene Structure from Video, in British Machine Vision Conference (BMVC'08).

(Cai and Aggarwal, 1999) Cai, Q. and J. K. Aggarwal, Tracking human motion in structured environments using a distributed-camera system, IEEE Trans. Pattern Anal. Machine Intell., vol. 21, no. 11, pp. 1241-1247.

(Comaniciu *et al*., 2003) Comaniciu D., Ramesh V. and Meer P., (2003). Kernel-Based Object Tracking, IEEE Trans. Pattern Anal. Machine Intell., pp. 564-575.

(Corvee *et al*., 2003) E. Corvee, S.A. Velastin, G.A. Jones, (2003). Occlusion Tolerent Tracking using Hybrid Prediction Schemes. In Acta Automatica Sinica, Special Issue on Visual Surveillance of Dynamic Sc 23(3) pp. 356-369.

(Cucchiara *et al*., 2003) Cucchiara, R., Piccardi, M., and Prati, A. (2003). Detecting moving objects, ghosts, and shadows in video streams, IEEE Transactions on Pattern Analysis and Machine Intelligence 25, pp. 1337-1342.

(Cucchiara *et al*., 2001) Cucchiara, R., Grana, C., Piccardi, M., Prati, A., and Sirotti, S. (2001). Improving shadow suppression in moving object detection with hsv colour information, in Proceedings of IEEE Int'l Conference on Intelligent Transportation Systems, pp. 334-339.

(Cheung and Kamath, 2005) Cheung, S-C. S., Kamath, C., (2005). Robust background subtraction with foreground validation for urban traffic video. Eurasip Journal on Applied Signal Processing 14, pp. 2330–2340.

(Chen *et al.*, 2007) Chen, Y.-T., Chen, C.-S., Huang, C.-R., and Hung, Y.-P. (2007). Efficient hierarchical method for background subtraction. Pattern Recognition, 40(10): 2706 – 2715.

(Chen *et al.*, 2007) Chen, K. W., Lai, C. C., Hung, Y. P. and Chen C. S. (2008). An adaptive learning method for target tracking across multiple cameras, IEEE Conference on Computer Vision and Pattern Recognition (CVPR08), pp.1-8.

(Cheng *et al.*, 2006) Cheng, E., Madden, C., Piccardi, M. (2006). Mitigating the Effects of Variable Illumination for Tracking across Disjoint Camera Views, International Conference on Advanced Video and Signal Based Surveillance, pp. 32-38.

(Chilgunde *et al.*, 2004) Chilgunde, A., Kumar, P., Ranganath, S., WeiMin, H. (2004). Multi-Camera Target Tracking in Blind Regions of Cameras with Non-Overlapping Fields of View, British Machine Vision Conference (BMVC 2004), London, pp 397-406.

(Cleveland and Devlin, 1988) Cleveland, W.S., Devlin, S.J. (1988). Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting. Journal of the American Statistical Association 83 (403): 596-610.

(CLEAR, 2007) CLEAR (2007). Clear classification of events, activities and relationships evaluation and workshop. http://isl.ira.uka.de/clear07/ [Last accessed: February 2009].

(Collins *et al.*, 2000) Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., and Wixson, L. (2000). A system for video surveillance and monitoring, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., CMU-RI-TR-00-12.

(Carnegie, 2003) Carnegie., R. C., Mellon University, Mean-shift Blob Tracking through Scale Space, IEEE Conference on Computer Vision and Pattern Recognition.

(Czyz *et al.*, 2007) Czyz, J., Ristic, B. and Macq, B. (2007). A particle filter for joint detection and tracking of color objects, Image and Vision Computing, vol. 25, no. 8, pp. 1271–1281

(CAVIAR,nd) CAVIAR, (no date) Available at: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/ [Last accessed: July 2010]

(Cracknell, 2007) Cracknell, M. (2007). Image detection in the real world – interactive session. In Intelligent Transportation Systems ITS ’07 Aalborg.

(Cracknell, 2008) Cracknell, M. (2008). Image detection in the real world – a progress update. In Intelligent Transportation Systems World Congress ITS WC 2008 – New York.

(Davies *et al*., 1998) Davies, D., Palmer, P., Mirmehdi, M., (1998) Detection and Tracking of Very Small Low Contrast Objects, in Proceedings of the 9th British Machine Vision Conference.

(Durucan *et al.*, 1999) Durucan, E., Snoeckx, J., Weilenmann, Y., (1999). Illumination Invariant Background Extraction, iciap, pp.1136, 10th International Conference on Image Analysis and Processing (ICIAP'99), 1999

(Ellis, 2002) Ellis, T. (2002). Performance Metrics and Methods for Tracking in Surveillance, Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, June, Copenhagen, Denmark.

(Ellis *et al*., 2003) Ellis, T., Makris, D., Black, J. (2003). Learning a Multi-camera Topology, Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Nice, France, pp. 165-171.

(ETISEO, nd) ETISEO, (no date) Available at: http://www-sop.inria.fr/orion/ETISEO/ [Last accessed: July 2010]

(François and Medioni, 1999) François A.R.J. and Medioni G.G. (1999). Adaptive Color Background Modelling for Real-Time Segmentation of Video Streams. In Proc. Int. Conf. on Imaging Science, Systems, and Technology, pp. 227-232.

(Farrell *et al.*, 2007) Farrell, R., Doermann, D., Davis, L. S. (2007). Learning Higher-order Transition Models in Medium-scale Camera Networks, pp.1-8, ICCV.

(Guler *et al.*, 2007) Guler, S., Silverstein, Pushee, J. A., Ian, H. (2007). Stationary Objects in Multiple Object Tracking, IEEE International Conference on AVSS.

(Greenhill *et al.*, 2008) Greenhill D., Renno J.R., Orwell J., Jones G.A. (2008). Occlusion Analysis: Learning and Utilising Depth Maps in Object Tracking, in Image and Vision Computing, Special Issue on the 15th Annual British Machine Vision Conference Elsevier Publishing, 26(3), pp. 430-444.

(Gao *et al.*, 2009) Gao, T., Liu, Z., Gao, W., and Zhang, J. (2009b). A robust technique for background subtraction in traffic video. In Advances in Neuro-Information Processing, volume 5507 of Lecture Notes in Computer Science, pages 736–744. Springer.

(Gupte *et al.*, 2002) Gupte, S., Masoud, O., Martin, R., and Papanikolopoulos, N. (2002). Detection and classification of vehicles. Intelligent Transportation Systems, IEEE Transactions on, 3(1):37–47.

(Gordon *et al.*, 1993) Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. Radar and Signal Processing, IEE Proceedings F, 140(2):107–113.

(Haritaoglu *et al*., 2000) Haritaoglu, I., Harwood, D. and Davis, L. S. (2000). W4: Real-time surveillance of people and their activities, In IEEE Transactions on Pattern Analysis & Machine Intelligence, pp. 809- 830.

(i-LIDS, nd) Home Office Scientific Development Branch: Imagery library for intelligent detection systems (i-LIDS), (no date). Available at: http://scienceandresearch.homeoffice.gov.uk/hosdb/cctv-imaging-technology/video-based-detection-systems/ i-lids/, [Last accessed: August 2008]

(Hu *et al*., 2004) Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors, IEEE Trans. on Systems, Man, and Cybernetics - Part C, 34(3): 334-352.

(Huang *et al*., 2008) Huang, C., Wu, Bo., and Nevatia, R. (2008). Robust Object Tracking by Hierarchical Association of Detection Responses, ECCV, Marseille, France.

(Huang and Wu, 2010) Huang, C.H. and Wu, R. C. (2010). A Multi-layer Scene Model for Video Surveillance Applications, in Lecture Notes in Computer Science (LNCS), Vol. 6297/2010, pp. 68-79

(Huang and Liao, 2004) Huang, C. L. and Liao, W. C. (2004). A vision-based vehicle identification system. In Proceedings of the 17th International Conference On Pattern Recognition, Vol 4, pp. 364–367.

(Hoiem *at al*., 2008) Hoiem D., Efros A. A., and Hebert M. (2008). Putting objects in perspective, International Journal of Computer Vision, pp. 0920-5691.

(Hu *et al*., 2006) Hu, M., Zhou, X., Tan, T., Lou J., Maybank, S., (2006). Principal Axis-Based Correspondence between Multiple Cameras for People

Tracking, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE Intelligence, Vol.28, No. 4, pp 663-671.

(Hu and Su, 2007) Hu, J.S., Su, T.M., (2007). Robust background subtraction with shadow and highlight removal for indoor surveillance. EURASIP Journal on Applied Signal Processing. Vol. 2007, Issue 1

(Hsieh *et al.*, 2006) Hsieh, J. W., Yu, S. H., Chen, Y. S., and Hu, W. F. (2006). Automatic traffic surveillance system for vehicle tracking and classification. IEEE Transactions On Intelligent Transportation Systems, 7(2):175–187.

(Horn and Schunk, 1981) Horn, B.K.P., Schunk, B.G. (1981). Determining optical flow, Artificial Intelligence, Vol. 17, pp.185-203.

(Isard and Blake, 1996) Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density, In Proc. European Conf. Computer Vision, pp. 343 - 356.

(Isard and Blake, 1998) Isard, M. and Blake, A. (1998). Condensation-conditional density propagation for visual tracking. International Journal of Computer Vision, 29(1):5-28.

(Ipsotek, nd) Ipsotek (n.d.). http://www.ipsotek.com/ [Last accessed: December 2009].

(Jang and Choi, 2000) Jang, D. S. and Choi, H. I. (2000). Active models for tracking moving objects, In Pattern Recognition., vol. 33, no. 7, pp. 1135-1146.

(Javed *et al.*, 2003) Javed, O., Rasheed, Z., Alatas, O. and Shah, M. (2003), KNIGHTTM: A real time surveillance system for multiple overlapping and

non-overlapping cameras. In International Conference on Multimedia and Expo.

(Javed *et al.*, 2005) Javed, O., Shafique, K. and Shah, M. (2005). Appearance Modeling for Tracking in Multiple Non-overlapping Cameras, IEEE CVPR, pp.26-33.

(Jain, 1981) Jain, R. (1981). Dynamic Scene Analysis Using Pixel-Based Processes, in IEEE Computer, vol. 14, no. 8, pp. 12-18.

(Jaynes *et al.*, 2002) Jaynes C., Webb S., Matt Steele R., Xiong, Q. An Open Development Environment for Evaluation of Video Surveillance Systems. The Third International Workshop on Performance Evaluation of Tracking and Surveillance (PETS'2002), Copenhagen, pp 32-39.

(Johansson *et al.*, 2009) Johansson, B., Wiklund, J., Forssén, P.-E., and Granlund, G. (2009). Combining shadow detection and simulation for estimation of vehicle size and position. Pattern Recognition Letters, 30(8):751 – 759.

(Krahnstoever and Mendonca, 2006) Krahnstoever, N. and Mendonca P. (2006). Autocalibration from Tracks of Walking People, British Machine Vision Conference, Edinburgh, UK.

(Kayumbi and Cavallaro, 2008) Kayumbi, G., Cavallaro, A. (2008). Multiview trajectory mapping using homography with lens distortion correction. EURASIP Journal on Image and Video Processing.

(Khan and Shah, 2003) Khan, S. and Shah, M., (2003). Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 25, No. 10, pp.1355-1360.

(Khan and Shah, 2009) Khan, S. and Shah, M. (2009). Tracking multiple occluding people by localizing on multiple scene planes. Trans. on Pattern Analysis and Machine Intelligence 31, 505–519.

(Kalman, 1960) Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems, Transaction of the ASME—Journal of Basic Engineering, pp. 35-45.

(Kumar *et al*., 2002) Kumar, P., Sengupta, K. and Lee, A. (2002), A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system. In The IEEE 5th International Conference on Intelligent Transportation Systems, pp. 100-105.

(KadewTraKuPong and Bowden, 2001) KadewTraKuPong, P. and Bowden, R. (2001). An improved adaptive background mixture model for real-time tracking with shadow detection. In Proceedings of 2nd European Workshop on Advanced Video-Based Surveillance Systems.

(Kanhere, 2008) Kanhere, N. (2008). Vision-based Detection, Tracking and Classification of Vehicles using Stable Features with Automatic Camera Calibration. PhD thesis, Clemson University, USA.

(Kettnaker and Zabih, 1999) Kettnaker, V., Zabih, R. (1999). Bayesian Multi-Camera Surveillance. IEEE Conference on Computer Vision and Pattern Recognition, CVPR99, Fort Collins, Colorado, pp 2253-2561.

(Kim and Davis, 2006) Kim, K. and Davis, L. S. (2006) Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering, In ECCV06, pp.98-109.

(Kwok *et al*., 2004) Kwok, C., Fox, D., Meila, M., (2004). Real-time particle filters, in Proceedings of the IEEE, Vol.92. Issue no.3, pp.469-484

(Lazarevic-McManus *et al*., 2007) Lazarevic-McManus, N., Renno, J.R., Makris, D. and Jones, G.A. (2007). An Object-based Comparative Methodology for Motion Detection based on the F-Measure, in Computer Vision and Image Understanding, Special Issue on Intelligent Visual Surveillance, pp. 74-85.

(Lee, 2005) Lee, D.S., (2005). Effective Gaussian mixture learning for video background substraction, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (5), pp.827–832.

(Lu *et al*., 2007) Lu, S.J., Zhang, J., Feng, D. (2007). An efficient method for detecting ghost and left objects in surveillance video, IEEE Conference on Advanced Video and Signal Based Surveillance, pp.540-545, London.

(Lv *et al*., 2006) Lv F., Zhao T., and Nevatia R. (2006). Camera calibration from video of a walking human, IEEE Trans on Pattern Analysis and Machine Intelligence, pp.1513-1518.

(Lin *et al*., 2008) Lin L., Zhu L., Yang F., Jiang T. (2008). A novel pixon-representation for image segmentation based on Markov random field , Image and Vision Computing, Vol. 26, No. 11. pp. 1507-1514.

(Loy, 2009) Loy C. C., Xiang T., Gong S.G. (2009). Modelling Multi-object Activity by Gaussian Processes, British Machine Vision Conference (BMVC’09).

(Michael *et al*., 2008) Michael D. Breitenstein, Eric Sommerlade, Bastian Leibe, Luc van Gool and Ian Reid, (2008). Probabilistic Parameter Selection for Learning Scene Structure from Video, British Machine Vision Conference (BMVC'08).

(Makris *et al*., 2004) Makris, D., Ellis, T., Black, J. (2004). Bridging the Gaps between Cameras, IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR Washington DC, USA, pp. 205-210.

(Madden *et al*., 2007) Madden C., Cheng E. D., Piccard M. (2007). Tracking people across disjoint camera views by an illumination-tolerant appearance representation, Machine Vision Applications, pp. 18(3-4): 233-247.

(Madden and Piccardi, 2007) Madden, C., Piccardi, M. (2007). A Framework for Track Matching Across Disjoint Cameras using Robust Shape and Appearance Features ,AVSS London, pp.188-193.

(McKenna *et al*., 2000) McKenna, S.J., Jabri, S., Duric, Z., Rosenfeld, A., and Wechsler, H. (2000). Tracking groups of people, Computer Vision and Image Understanding, vol. 80, no. 1, pp. 42–56.

(Messelodi *et al*., 2005a) Messelodi, S., Modena, C. M., Segata, N., and Zanin, M. (2005a). A kalman filter based background updating algorithm robust to sharp illumination changes. In Roli, F. and Vitulano, S., editors, Lecture Notes in Computer Science, volume 3617, pp. 163–170. Springer.

(Messelodi *et al*., 2005b) Messelodi, S., Modena, C. M., and Zanin, M. (2005b). A computer vision system for the detection and classification of vehicles at urban road intersections. Pattern Analysis & Applications, 8(1-2):17–31.

(Martel-Brisson and Zaccarin, 2007) Martel-Brisson, N. and Zaccarin, A. (2007). Learning and removing cast shadows through a multidistribution approach. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 29(7):1133–1146.

(Mauthner *et al*., 2008) Mauthner, T., Donoser, M., and Bischof, H. (2008). Robust tracking of spatial related components. In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, pages 1–4.

(Nascimento and Marques, 2005) Nascimento, J. and Marques, J. (2005). Performance evaluation of object detection algorithms for video surveillance, In IEEE Transactions on Multimedia, pp. 761-774.

(Needham and Boyle, 2003) Needham, C.J. and Boyle, R. D. (2003). Performance Evaluation Metrics and Statistics for Positional Tracker Evaluation, In International Conference on Computer Vision Systems (ICVS'03), Graz, Austria, pp. 278 – 289.

(Nghiem *et al*., 2007) Nghiem, A. T., Bremond, F., Thonnat, M. and Valentinn, V. (2007). Etiseo, performance evaluation for video surveillance systems, Proceedings of AVSS.

(Nummiaro *et al*., 2002) Nummiaro, K., Koller-Meier, E. and Gool, L.Van. (2002). Object Tracking with an Adaptive Color-Based Particle Filter, Symposium for Pattern Recognition of the DAGM, pp.353-360.

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

(OpenCV, nd) OpenCV Computer Vision Library (no date) Available at: http://www.intel.com/ technology/ computing/opencv/index.htm, [Last accessed: August 2009].

(Oh *et al*., 2004) Oh, S., Russell, S. and Sastry, S. (2004). Markov chain monte carlo data Association for general multiple-target tracking problems, 43rd

IEEE Conference on Decision and Control, pp735-742, Paradise Island, Bahamas.

(Pets Metrics, nd) Pets Metrics, (no date) Available at: http://www.petsmetrics.net, [Last accessed: December 2009].

(Peterfreund, 2000) Peterfreund, N. (2000). Robust tracking of position and velocity with Kalman snakes, IEEE Trans. Pattern Anal. Machine Intell., vol. 22, pp. 564–569.

(Project PASCAL, nd) project PASCAL (n.d.). The pascal visual object classes homepage. http://pascallin.ecs.soton.ac.uk/challenges/VOC/ (Last accessed: November 2009).

(PASCAL,nd) PASCAL, (no date) Available at: http://pascallin.ecs.soton.ac.uk/challenges/VOC/[Last accessed: July 2010]

(Pasula *et al*., 1999) Pasula, H., Russell, S. J., Ostland, M. and Ritov, Y. (1999) Tracking many objects with many sensors. In IJCAI, pp. 1160–1171.

(Rosin, 1997) Rosin, P. L. (1997). Thresholding for Change Detection, British Machine Vision Conference, BMVC97, pp. 212-221, Colchester, UK.

(Park *et al*., 2007) Park, K., Lee, D., and Park, Y. (2007). Video-based detection of street-parking violation. In Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition CVPR 2007.

(Porikli and Divakaran, 2003) Porikli, F., Divakaran, A. (2003). Multi-Camera Calibration, Object Tracking and Query Generation, International Conference on Multimedia and Expo, ICME2003, pp 653-656, Baltimore.

(Prati *et al.*, 2003) Prati, A., Miki, I., Cucchiara, R., and Trivedi, M. M. (2003). Comparative evaluation of moving shadow detection algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25:918–923.

(Prosser *et al.*, 2008) Prosser, B., Gong, S.G. and Xiang, T. (2008). Multi-camera matching under illumination change over time, In Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2).

(Renno *et al.*, 2002) Renno, J. R., Orwell J., Jones G.A. (2002). Learning Surveillance Tracking Models for the Self-Calibrated Ground Plane, In British Machine Vision Conference, September, Cardiff, pp. 607-616.

(Remagnino *et al.*, 1998) Remagnino, P., Maybank, S., Fraile, R., Baker, K., and Morris, R. (1998). Advanced Video-based Surveillance Systems, chapter Automatic Visual Surveillance of Vehicles and People, pp. 97–107. Hingham, MA., USA.

(Rahimi *et al.*, 2004) Rahimi, A., Dunagan, B., Darrell, T. (2004). Simultaneous Calibration and Tracking with a Network of Non-Overlapping Sensors, Proc. of Computer Vision and Pattern Recognition Converence (CVPR), pp.187-194, Washington DC.

(Rother *et al.*, 2007) Rother,D., Patwardhan,K.A., Sapiro, G., (2007). What Can Casual Walkers Tell Us About A 3D Scene?, iccv, pp.1-8, IEEE 11th International Conference on Computer Vision, 2007

(Saxena *et al.*,2009) Saxena A., Sun M., and Ng A. (2009). Make3D: Learning 3D Scene Structure from a Single Still Image, IEEE Trans on Pattern Analysis and Machine Intelligence, pp. 824-840.

Shan, Y., Yang, F. and Wang, R.S. (2007). Colour Space Selection for Moving Shadow Elimination, In Fourth International Conference on Image and Graphics, pp.496 – 501.

(Smith, nd) Smith, L. I., (no date). A Tutorial on Principal Component Analysis, http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, [last accessed: April 2009]

(Stauffer and Grimson, 1998) Stauffer, C., Grimson, W.E.L. (1998). Adaptive background mixture models for real-time tracking, In Computer Vision and Pattern Recognition, Santa Barbara, CA.

(Stauffer and Grimson, 2000) Stauffer, C. and Grimson, W. (2000). Learning patterns of activity using real-time tracking. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(8):747–757.

(Stauffer and Tieu, 2003) Stauffer, C and Tieu, K. (2003). Automated multi-camera planar tracking correspondence modeling. In proceedings of CVPR, pp.259.

(Stauffer, 2005) Stauffer, C. (2005). Learning to Track Objects Through Unobserved Regions, IEEE Workshop on Motion and Video Computing (WACV/MOTION'05), pp.96-102, Breckenridge, Colorado.

(Senior *et al.*, 2001) Senior, A., Hampapur, A., Tian, Y. L., Brown, L., Pankanti, S. and Bolle, R. (2001). Appearance models for occlusion handling, In Second International workshop on Performance Evaluation of Tracking and Surveillance systems.

(Senior *et al.*, 2006) Senior, A., Hampapur, A., Tian, Y. L., Brown, L., Pankanti, S. and Bolle, R. (2006). Appearance models for occlusion handling, in Image and Vision Computing, Volume 24, Issue 11, pp. 1233-1243.

(Sketchup, nd) Sketchup, (no date) Available at:

http://sketchup.google.com/intl/en_uk/ [Last accessed: April 2011]

(Song and Nevatia, 2007) Song, X. and Nevatia, R. (2007). Detection and tracking of moving vehicles in crowded scenes. In Motion and Video Computing. WMVC '07. IEEE W. on, page 4.

(Stein, 1998) Stein, G.P. (1998). Tracking from MultipleView Points: Self-calibration of Space and Time, Proc. DARPA IU Workshop, pp. 1037–1042.

(Thirde *et al*., 2005) Thirde, D., Borg, M., Valentin, V., Fusier, F., Aguilera, J., Ferryman, J., Bremond, F., Thonnat, M., Kampel, M., (2005). Visual Surveillance for Aircraft Activity Monitoring, 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp 255-262 , Beijing.

(Tan *et al*., 1994) Tan, T. N., Sullivan, G. D., Baker, K.D. (1994). Recognising Objects on the Ground-plane, Image and Vision Computing, vol.12, pp.164-172.

(Taj *et al*., 2008) Taj, M., Maggio, E., and Cavallaro, A. (2008). Objective evaluation of pedestrian and vehicle tracking on the clear surveillance dataset. Lecture Notes In Computer Science, 4625:160-173.

(Tsai, 1986) Tsai, R. Y. (1986). An efficient and accurate camera calibration technique for 3d machine vision. In Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR), (1986), pages 364–374.

(Tieu *et al*., 2005) Tieu, K., Dalley, G. and Grimson, W. (2005). Inference of nonoverlapping camera network topology by measuring statistical Dependence. In Proc. IEEE International Conference on Computer Vision, pages II: 1842–1849.

(Trivedi *et al*., 2000) Trivedi, M.M., Mikic,I. and Kogut, G. (2000). Distributed video networks for incident detection and management, in Proceedings of IEEE Int'l Conference on Intelligent Transportation Systems, pp. 155–160.

(Utsumi and Yachida, 1998) Utsumi, A., Mori, H., Ohya, J., and Yachida, M. (1998). Multiple-view-based tracking of multiple humans, in Proc. Int. Conf. Pattern Recognition, pp. 197-601.

(Valera and Velastin, 2005) Valera, M. and Velastin, S. (2005). Intelligent distributed surveillance systems: a review. Vision, Image and Signal Processing, IEE Proceedings , 152(2):192-204.

(Velastin *et al*., 2005) Velastin, S. A., Boghossian, B. A., Lo, B. P. L., Sun, J. and Vicencio-Silva, M. A.(2005). PRISMATICA: Toward Ambient Intelligence in Public Transport Environments. SMC, pp.164-182.

(VISOR, nd) VISOR (no date). Video surveillance online repository. http://www.openvisor.org/ (Last accessed: November 2009).

(ViPER-GT, nd) Authoring Video Metadata with ViPER-GT (no date). Available at: http://viper-toolkit.sourceforge.net/docs/gt/, (Last accessed: February 2010).

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

(Virage, nd) Virage (n.d.). http://www.virage.com/ (Last accessed: December 2009).

(Welch and Bishop, 2000) Welch, G., Bishop, G. (2000). An introduction to the Kalman filter, from http://www.cs.unc.edu, UNC-ChapelHill, TR95-041 (Last accessed: October 2009).

(Wang *et al.*, 2009) Wang, J., Ma, Y., Li, C., Wang, H., and Liu, J. (2009b). An efficient multi-object tracking method using multiple particle filters. In Computer Science and Information Engineering, 2009 WRI World Congress on, volume 6, pages 568–572.

(Wallace and Diffley, 1998) Wallace, E., Diffley, C. (1998). CCTV control room ergonomic: Making it Work, Police Scientific Development Branch of the Home Office (PSDB) publication No 14/98.

(Xu and Ellis, 2002) Xu, M. and Ellis, T.J. (2002). Partial observation vs. blind tracking through occlusion, British Machine Vision Conference, BMVA, September, Cardiff, pp. 777-786.

(Xu and Ellis, 2001) Xu, M. and Ellis, T.J. (2001). Colour-Invariant motion detection under fast illumination changes, 2nd European Workshop on Advanced Videobased Surveillance, AVBS2001, pp. 335-345.

(Xu and Ellis, 2006) Xu, M. and Ellis, T.J. (2006). Augmented tracking with incomplete observation and probabilistic reasoning, Image and Vision Computing 24(11) pp. 1202-1217.

(Yue *et al.*, 2004) Yue, Z., Zhou, S.K., and Chellappa, R., (2004). Robust two-camera tracking using homography. In Proc. of IEEE Intl Conf. on Acoustics, Speech, and Signal Processing, volume 3, pp.1-4.

(Yang *et al.*, 2005) Yang, C. j., Duraiswami, R., Davis, L. (2005). Efficient Mean-Shift Tracking via a New Similarity Measure, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, pp.176-183.

(Yilmaz *et al*., 2006) Yilmaz, A., Javed, O. and Shah, M. (2006). Object Tracking: A Survey, ACM Journal of Computing Surveys, Vol. 38, No. 4.

(Yilmaz, 2007) Yilmaz, A. (2007). Object Tracking by Asymmetric Kernel Mean Shift with Automatic Scale and Orientation Selection, Conference on Computer Vision and Pattern Recognition, (CVPR 2007), pp.1-6.

(Zhang and Scanlon, 2008) Zhang, Z., Scanlon, A., (2008). Video Surveillance using a Multi-Camera Tracking and Fusion System, IEEE International Workshop on Multi-camera and Multi-modal Sensor Fusion: Algorithms and Applications, Marseille, France.

(Zheng *et al*., 2005) Zheng, J., Wang, Y., Nihan, N. L., and Hallenbeck, M. E. (2005). Extracting roadway background image: Mode-based approach. Transportation Research Record, (1944):82–88.

(Zhou *et al*., 2004) Zhou, S.K., Chellappa, R., Moghaddam, B., (2004). Visual tracking and recognition using appearance-adaptive models in particle filters, IEEE Transactions on Image Processing, Vol.13, pp.1491-1506