Object detection and segmentation for visual surveillance

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
Prof. mr. P.F. van der Heijden
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Aula der Universiteit
op vrijdag 3 februari 2006, te 10.00 uur

door

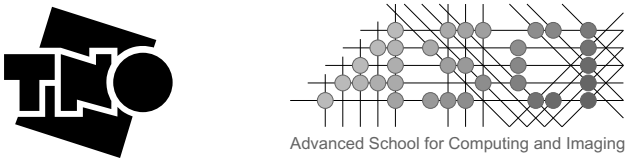
Petrus Johannes Withagen

geboren te Halsteren.

Promotiecommissie:

Promotor:	Prof. dr. ir. F.C.A. Groen
Co-promotor:	Dr. K. Schutte
Overige leden:	Prof. dr. ir. A.W.M. Smeulders
	Prof. dr. ir. R.L. Lagendijk
	Prof. dr. P.W. Adriaans
	Dr. ir. B.J.A. Kröse
	Dr. C.S. Regazzoni

Faculteit der Natuurwetenschappen, Wiskunde en Informatica



This work was carried out in the ASCI graduate school. ASCI dissertation series number 120. The research was performed at TNO Defence and Safety, The Hague, The Netherlands and at the IAS group, University of Amsterdam, The Netherlands.

ISBN-10: 90-9020282-X
ISBN-13: 978-90-9020282-2

Author email: PaulWithagen@hetnet.nl

voor Jolanda & Sanne
voor mijn ouders

Cover photo: Politie Haaglanden, the Hague, The Netherlands

Contents

Chapter 1

Introduction and tracking framework

A Journey of a Thousand Miles
begins with a single step.
(Lao Tzu)

1.1 Introduction

1.1.1 Visual surveillance

Cameras are everywhere these days: public areas are monitored by several cameras in order to increase public order and safety, private property is protected using cameras and stores use cameras to prevent shoplifting. More Closed Circuit Tele-Vision (CCTV) cameras are installed every day to combat the growing feeling of unsafety. To quote New Scientist [Hogan, 2003]:

> Some campaign groups see CCTV as an invasion of privacy but customers are likely to welcome technology that makes stations safer.

However, the presence of cameras alone will have very limited effect on crime [Gill et al., 2005], only subjective safety is increased. To make effective use of surveillance cameras, an operator must look at the images and respond to suspicious activities.

Trained and experienced human operators can do this monitoring very well, but only for a limited number of cameras simultaneously and only for a limited amount of time, after which concentration is diminished. Operators who are not concentrated will not respond promptly to important events or might even miss them altogether [Norris and Armstrong, 1999; Weitenberg et al., 2003].

For most applications, watching all camera images is not only too expensive but also practically impossible. Hogan [Hogan, 2003] writes about technology for automated visual surveillance:

> If the technology takes off it could put an end to a longstanding problem that has dogged CCTV almost from the beginning. It is simple: there are too many cameras and too few pairs of eyes to keep track of them. With more than a million CCTV cameras in the UK alone, they are becoming increasingly difficult to manage.

In common practice, only a limited number of cameras are monitored. The images from the remaining cameras will be either lost or stored for later reference. Storing images does not allow immediate action and may cause problems related to privacy [Smeets, 2004].

In order to aid operators in their task, ”intelligent” sensors can be used. These sensors can take over tedious tasks such as monitoring forbidden areas, keeping track of people and detecting suspicious motion patterns or behavior. On deviation from normal behavior, the operator is alerted. He makes the final decision and takes action upon it. Additional benefit: in principle, intelligent sensors do not invade the privacy of people with good intentions as no images will be shown to humans as long as nothing suspicious occurs.

Intelligent sensors are sensors that can make decisions based on the scene. This thesis will focus on applications using a single camera which obtains its intelligence from visual surveillance algorithms. The work in this thesis can be applied in

a multi-camera setting, for increasing positional accuracy when fields of view overlap or for increasing the area under surveillance when objects are tracked through multiple cameras [Kröse et al., 2004; Zajdel et al., 2004]. The results could also be used in combination with other sensors using sensor fusion [Arora et al., 2004], Examples are microphones [Cristani et al., 2004] to cue on specific sounds and radar [Dorp and Groen, 2003]. Radar needs virtual models for object tracking, but in return has high depth and velocity resolution, a perfect addition to the high angular resolution of a camera. In addition, a radar can be used through walls and in the dark.

Tracking of people

Visual surveillance seems trivial. Assuming a static camera and static background, moving real-world objects such as a person, a car or a group of people can be detected by comparing the current image to an image of the background. Differences between these two images result from moving objects, which are now detected. In the next frame, the objects will have moved an additional small number of pixels. Object tracking stores these movements over time and predicts the object location in the next frame. This results in an object trajectory. The objects shape and its trajectory are used for object analysis: is there something suspicious going on?

In practice however, the visual surveillance task is much more complex. Problems are mostly related to accuracy and robustness. For example, the simple detection approach given above will fail as soon as the background changes. This will occur frequently, due to changes in illumination (brightness, direction, color, shadow), background objects that change appearance (flag, waving tree, water surface) and noise. The object tracking approach given before can track a single object with constant speed and direction. For visual surveillance, many objects need to be tracked simultaneously. They can occlude each other or interact with other objects or the background scene. Often, the objects to be tracked are people. They are deformable and often change speed and direction.

Advanced background modelling techniques are being developed for updating changing scenes. Objects are more robustly tracked using a model of their appearance. The first aspect of building a robust application is to select and incorporate separate algorithms for different tasks such as detection and tracking. This requires the algorithms to communicate with each other and use results of other algorithms.

The second aspect involves selection of parameters. An operator will not be able to set tens or even hundreds of parameters such as update speeds, thresholds, tuning factors and ”magic” parameters in a sufficient way. Therefore, most parameters should be set in advance, or deducted from application dependent knowledge such as camera field of view, indoor or outdoor application, and the costs associated with missed detections and false alarms. The best parameter value for one situation will generally not be optimal in other situations. Therefore, setting parameters in advance is only effective when the performance of the algorithm is not very dependent on the optimal setting of the parameter.

These two aspects of building a robust video surveillance application, algorithms selection/integration and parameter setting, need to be re-evaluated when using the visual surveillance algorithm in a different environment or for a different task.

1.2 Application description and central question

"Visual tracking of people in complex scenes" [Haritaoglu et al., 2000; Hu et al., 2004b; Rosales and Sclaroff, 1999] is the theme of this thesis. This is an important topic where a reliable algorithm for tracking people enables a variety of applications related to security and surveillance. Examples are a car thief in a parking lot, a drug dealer in a public area and a pickpocket in a railway station.

The object segmentation and tracking algorithms proposed in this thesis are intended to generate input to trajectory analysis. Trajectory analysis can aid operators watching surveillance cameras, allowing them to operate more cameras at once for longer periods of time. Using trajectory analysis, only suspicious events need to be examined by the operator. Selecting suspicious events is performed automatically from the results of object segmentation and tracking.

The **central question** of this thesis is: Can we develop an algorithm for robustly and accurately detecting and segmenting moving objects in surveillance scenes? Robust is defined in this context as able to cope with frequent problems such as variation in global intensity, shadows, deformable objects and objects with colors similar to the background. Accurate in this context means that we intend to obtain an accurate segmentation between foreground and background.

The **scenes** in which our application is intended to be used are typical surveillance scenes, both indoor and outdoor. These scenes consist of a static background and can contain several moving objects simultaneously. A fraction of object pixels up to a quatre is possible, divided over up to ten different real-world objects. Real-world moving objects may include trucks, cars, pedestrians, cyclists, dogs, shopping carts, strollers, etcetera. The background in our application can change appearance, for instance when looking at vegetation and water surfaces. The appearance of the background is allowed to change slowly, for example due to changing time of day. Because of intended outdoor use and widespread use of affordable auto-gain cameras, we shall allow fast global changes in intensity.

We define a **tracked object**, sometimes called a foreground object, as the image pixels depicting (part of) a real-world object or a group of real-world objects that is visible separate from other real-world objects. So a human can be a tracked object, but when a group of humans walk together, the group will be regarded as a single tracked object. Real-world objects can also be split into several tracked objects, for example when the lower part of the body is tracked separately from the upper part. This has the advantage that when the person removes his jacket, and the upper-part track is therefore lost, the lower part is still tracked. At a higher level, these tracks can be combined, leading to a conclusion like "The jeans removed its jacket". This means that there generally does not exist a one-to-one mapping

between tracked objects in the image and real-world objects.

In the image, tracked objects can be as small as 50 pixels. On average, tracked objects of 200 to 500 pixels are expected. The objects are deformable and can have an arbitrary shape. Motion and shape deformation is assumed to be limited between frames, so that the center of the tracked object can be fully retrieved in the subsequent frame. The color distribution is assumed to change slowly and we expect the color distribution of the center of the tracked object to be representative for the entire object.

The scene is intended to be recorded using a static color **camera**. This camera should adhere to the simplified model of a CCD camera given in Appendix A. The proposed algorithms should be able to run in real-time at 12 frames per second in PAL resolution on a personal computer available in 2010. The normal amount of memory in such PC should be sufficient for the algorithms used.

All moving real-world objects should be **tracked**, but we allow tracked objects to split, merge and be lost, for example during occlusion. We also should keep track of real-world objects that have stopped moving. They should be distinguished from changed background, such as the street that changes appearance due to the rain, which is not considered a real-world object. Multiple trajectories of real-world objects may correspond to one calculated trajectory and vice versa. This happens for example when several persons walk in a group. The ground truth will describe separate tracks, while our algorithm might generate only one track for the entire group of people. For our application this poses no problem as we are interested in suspicious trajectories. It is then not important how often the suspected trajectory occurs simultaneously.

The **algorithms** developed should be generally applicable. They should be easy to adapt when the environment changes. We prefer to use algorithms for which the parameters are easy to set. This means that either the performance of the algorithm should not rely heavily on the value of the parameter, or we should be able to deduct the parameter value from real world variables such as camera height, expected object density, etcetera.

The **performance** of the application should be good and the application should be robust. Knowledge from higher levels of abstraction should therefore be used for updating the lower levels. Intensity information should be exploited, for example for the detection of black objects in front of a gray background. Shadows should be removed from detected tracked objects.

1.3 Probabilistic framework

An application for object detection and tracking generally consists of several algorithms, each operating at one or more of the levels of abstraction available in the application. For example, Chapters 3 to 7 of this thesis each introduce one algorithm for solving part of the object tracking problem described in the previous section.

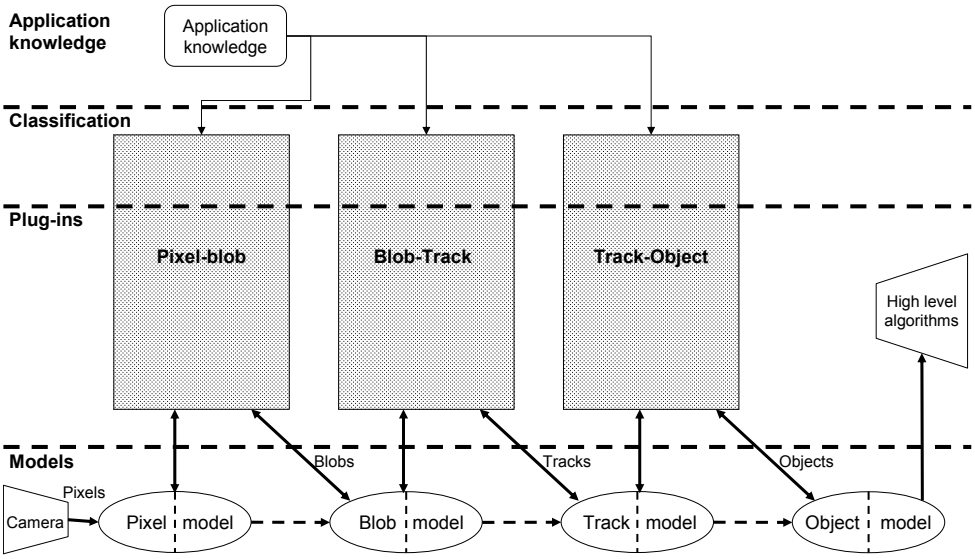


Figure 1.1: *Flowchart of the proposed probabilistic framework. Gray rectangles denote the different application-specific modules and oval-shaped boxes the models. The arrows indicate data flow. The dashed arrows indicate that model knowledge of the lower level models is also available at the higher levels. Feedback is not shown in this figure.*

Combining separate algorithms into one coherent application is an important, but generally not a simple task. For the combination of the different algorithms introduced in this thesis a probabilistic framework will be introduced here. It will be used in Chapter 7 to compose a coherent application. Strong features of this framework are minimum cost Bayes classification at each level of abstraction, a structured way of using application knowledge, a flexible layout which allows easy substitution of algorithms, and a structured communication including feedback from high level results to the lower levels of abstraction.

The separate algorithms of the application are introduced into the framework as application dependent plug-in algorithms, while the framework itself consists of application independent classification algorithms and models. Two types of plug-ins are distinguished: the correction module, working on a single level of abstraction, and the probabilities module, bridging the gap between levels of abstraction.

Figure 1.1 shows the flowchart of a possible framework configuration. From left to right are the different levels of abstraction used in the application: pixels, blobs, tracks and objects. Between levels are application specific modules that take care of the conversion between the different levels of abstraction. Figure 1.2 gives a more detailed view of such a module. The different parts of the framework are described in the next section.

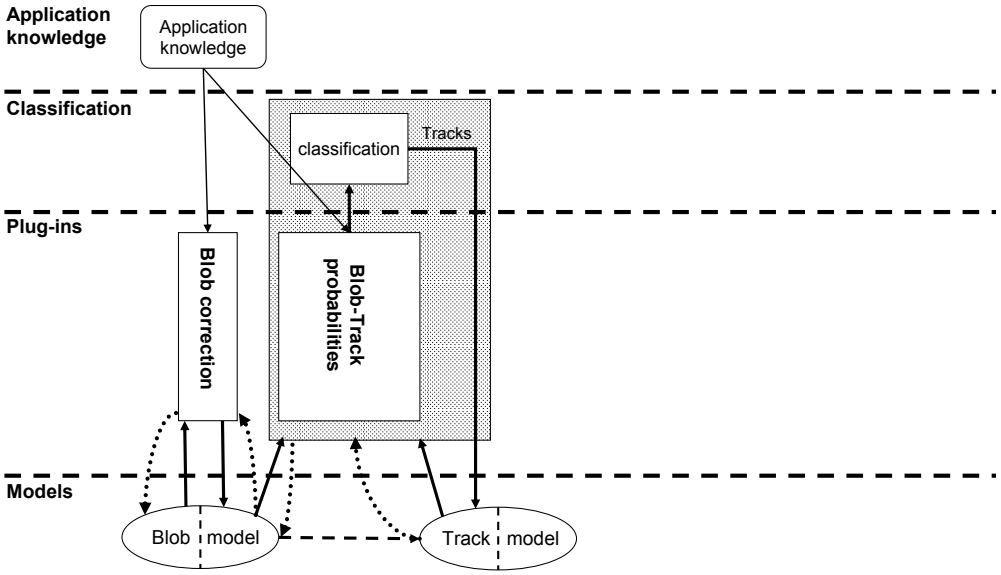


Figure 1.2: *Correction and probabilities modules. Dotted arrows indicate feedback used to update plug-in data, other arrows indicate dataflow in normal processing order.*

1.3.1 Building blocks of the framework

An application generally consists of three or more levels of abstraction. For each of the levels, algorithms should be chosen that are appropriate for the given application. Including them in a probabilistic framework allows the use of uncertainties and minimum cost classification.

This section describes the building blocks of the framework. The internal structure of the building blocks and their interface to other blocks will be discussed in Chapter 7.

Correction modules

Data at a given level of abstraction may not be usable as-is. A correction module solves this, see Figure 1.2. Correction modules are application dependent plug-in algorithms that operate within one level of abstraction. Examples discussed in this thesis are global intensity correction at the pixel level and shadow removal at the blob level.

Probabilities modules

Plug-in algorithms that operate between levels of abstraction are denoted probabilities modules, see Figure 1.2. Probabilities modules calculate features from the lower level model and compare these to output classes in the higher level model. This comparison should result in probabilities for each of the output classes. Examples discussed in this thesis are calculation of pixel probabilities using models of the background and the foreground objects, and the calculation of the probability that a blob is in fact a shadow region.

Plug-in algorithms only interact with models at their abstraction level(s). This approach makes it unnecessary that one plug-in algorithm, probabilities module or correction module, knows what the other plug-in algorithms do, as long as the data it requires is available in the models.

Classification

Between each level of abstraction minimum cost Bayes classification is performed, see Figure 1.2. Classification is based on probabilities and classification costs, independent of the source of these inputs and therefore independent of the the application.

Each of the probabilities modules calculate probabilities for the association between the input features (e.g. pixel, blob or track features) at the lower level and output classes (e.g. blob β_2, track τ_1, object ω_0) at the higher level, see Figure 1.1. The kind of mapping, e.g. many-to-one, one-to-one, etc., depends on the application and the algorithms used. The classification results are communicated to the higher level model.

Models

A model is a description of the world at a specific level of abstraction, see Figure 1.1. The model is generally independent of the application dependent plug-in algorithms, although a change in the levels of abstractions used also leads to the use of different models. The model is not the only point of data storage. Plug-in algorithms have, when necessary, data storage for algorithm-specific data.

When a correction module is used for a level of abstraction, two versions of the model at that level are available. One version of the model before and one after the correction module, see Figure 1.2. Communication between the versions occurs through the correction module.

Data from lower level models can be used in the higher levels plug-in algorithms. This data is made available though the model. For example, the blob correction module may need pixel values from the pixel model. It can obtain them through the blob model, but it cannot alter or update them.

Application knowledge

Visual surveillance algorithms require application knowledge about what is normal and what suspicious, and what is important and what not. For example: Do we want raindrops or waving leaves to be classified as moving objects or background? A man sitting motionless on a chair for one minute, should he be classified as foreground or background? These decisions depend on the application domain.

The framework itself is separated from application specific knowledge, data or algorithms, see Figures 1.1 and 1.2. Application dependent knowledge is regarded input to the framework. It is introduced using plug-in algorithms and their parameters.

Besides the choice of plug-in algorithms, prior probabilities and cost matrices introduce application knowledge in the classification process. This occurs directly from application knowledge or through the probabilities modules, as this allows learning of updating of these parameters.

1.3.2 Interactions in the framework

The proposed framework also defines the interaction between the levels of abstractions. Four main interactions can be distinguished in Figure 1.2:

- **Through plug-in algorithms** classification results from the lower levels are used as input to the higher levels (solid arrows).
- **From left to right between the models** lower level model knowledge is made available to the higher level plug-in algorithms (dashed arrows).
- **From top to bottom** application knowledge is made available at the different levels (thin arrows).
- **From right to left** final classification results from the higher levels are made available to the lower level models through feedback (dotted arrows).

Feedback is an important feature of the framework. It allows to update the lower level models, update parameters or to finalize delayed multi hypothesis decisions using the most complete information about the world. This information is available at the high levels of abstraction.

Communication between model versions at one abstraction level occurs through the correction module. Communication between models, so between abstraction levels, occurs through probabilities modules and the classification algorithm.

1.4 Thesis overview

In Chapter 2 an overview of visual surveillance literature will be given. As visual surveillance is a very broad and active research area, an overview will be given, zooming in on topics directly related to the work in this thesis.

An important aspect of many algorithms analyzing image sequences is the changing global image intensity, caused by changing illumination or camera settings. In Chapter 3 a number of algorithms will be introduced and compared for correcting global changes in image intensity. The algorithms have been derived from a camera model introduced and evaluated in Appendix A.

A model of the background will be used for object detection and tracking in this thesis. Such a model should be updated to allow for changes in appearance. How and when to update the model is the topic of Chapter 4. The updating technique introduced there enables to model only the background, providing per-pixel background probabilities that can be used in the framework. The algorithm is shown to be very robust to its parameter settings.

Objects will be tracked using an object model. In Chapter 5, a model is introduced that allows the object to deform between subsequent frames. The proposed algorithm provides foreground probabilities as well as object trajectories.

Shadows are a recurrent challenge in visual surveillance applications. In Chapter 6 a shadow detection algorithm is introduced that estimates the colors of the illumination sources. Based on this information, shadows and moving objects can be distinguished very accurately.

Based on the framework introduced in Section 1.3, in Chapter 7 an application will be described that incorporates the separate algorithms described in Chapters 1 to 6. This application will be evaluated there using real image sequences. The overall performance of the application will be compared to a reference algorithm and to the separate algorithms introduced in this thesis.

In Chapter 8, overall conclusions will be given, and the contributions of this thesis will be discussed, together with ideas for future research.

Chapter 2 Overview of visual surveillance literature

2.1 Introduction

Visual surveillance involves ”looking at people”, which has been a subject of research for many years. Research on this topic started with Johansson's psychology research on moving light displays in 1973 [Johansson, 1973, 1975]. He showed that by looking at only joint motion, humans can easily recognize a persons actions. This raised the question wether machines could do this as well. Since then, looking at people has been topic of research in several areas. Examples are vision-based human-computer interaction [Quek, 1995], biomechanics (total body movement for medical applications) [Calvert and Chapman, 1994] and choreography [Badler and Smoliar, 1979]. Presently, looking at people and specifically visual surveillance is a very active research area. This is not surprising, considering the growing demand for security and homeland safety.

From a computational point of view, the state of the art of computer vision hardware is approaching the level required by visual surveillance algorithms. Hardware with sufficient computational power for relatively simple surveillance applications is available Commercially Off The Shelf (COTS). Hardware capable of running more complex algorithms in real-time is becoming affordable. This boosts research, as ideas and algorithms can quickly be applied in real-world applications.

In this chapter a broad overview of the field will be given, together with a more focussed discussion on aspects having a direct relationship to this thesis. For more detailed surveys of literature in this area, the reader is referred to one of the following surveys, reviews or overviews: [Aggarwal and Cai, 1999; Buxton, 2003; Cédras and Shah, 1995; Gavrila, 1999; Hu et al., 2004b; Moeslund and Granum, 2001; Wang et al., 2003; Wang and Singh, 2003] or one of the special issues [Collins et al., 2000a; Gong and Buxton, 2002; Kakadiaris et al., 2003; Maybank and Tan, 2000, 2004; Namuduri and Ramaswamy, 2004; Regazzoni and Foresti, 2001; Syeda-Mahmood et al., 2004].

In Section 2.2 an overview is presented of available frameworks and algorithms for video surveillance. After this, the chapter is organized according to the chain of processing. In Section 2.3 an overview is given of image acquisition and preprocessing techniques. Detection and segmentation of moving objects is described in Section 2.4. Techniques for tracking and recognition are discussed in Section 2.5. Section 2.6 considers the higher level of abstraction: behavior and pose recognition. Finally, conclusions are given in Section 2.7.

2.2 Video surveillance frameworks and algorithms

In their general framework for visual surveillance, Hu et al [Hu et al., 2004b] divide visual surveillance applications into five different levels of abstraction:

- Environment modelling
- Motion segmentation
- Object classification
- Tracking
- Higher level tasks like:
  - Behavior understanding and description
  - Personal identification
  - Fusion of information between multiple sensors

For each level, many algorithms exist and are topic of active research.

A difficult problem is building an application by combining algorithms from different levels. This should be done in such a way that good performance of each of the different levels is combined into good overall performance. Also, it should be easy to adapt the application, when the environment changes, or when it is applied in a slightly different scene.

Algorithms of different levels of abstraction in changing applications and environments can be combined using a framework allowing easy adaptation to specific tasks. It requires a framework allowing a structured decomposition of the application into algorithms, models and prior knowledge at different levels of abstraction. In such a framework it is possible to replace components without altering the remainder of the application. Therefore, it is straightforward to re-use existing methods in the adaptation to particular needs.

At the same time, such a framework must be designed such that good performance is obtained, given the performance of the algorithms of each level of abstraction. Classification and communication between different components of the framework should be performed in a solid statistical way. The higher the level of abstraction, the more knowledge is available about the world. It is therefore important to make this high level knowledge available to the lower levels. Communication should not be limited to passing results from lower levels to higher level components. Feedback enables knowledge transfer from higher to lower levels of

abstraction. This way, all available knowledge can be used in updating lower level models and adjusting parameters in the algorithms.

Existing frameworks

Approaches to a framework, in which algorithms for components of an application can be combined in a probabilistically sound way, are rare. In [Ivanovic and Huang, 2004] three Bayesian networks are used for pixel classification, blob matching and object detection. This approach is not very flexible, as the classification algorithms are closely related to the algorithms that provide the probabilities. This makes it difficult to use another algorithm for one of the abstraction levels. In [Spengler and Schiele, 2002, 2003] the "Knowledge Hierarchy" is proposed, a Bayesian formulation for multi-object tracking. This framework is used mainly to incorporate prior knowledge in a systematic way, in order to make the huge number of hypothesis in their multiple object tracking algorithm tractable. But even with the large amount of prior knowledge introduced, many hypothesis still need to be considered, resulting in a computationally expensive algorithm. Different levels of processing are also utilized in [Park and Aggarwal, 2002], but without attempting to create a versatile framework in which components can be easily substituted by other algorithms.

None of the approaches described explicitly model feedback. This is unfortunate, because feedback is an important characteristic in such a framework. It allows the lower levels to make use of knowledge available at the higher levels. An example of the efficient use of feedback is given in [Javed et al., 2002]. Feedback from the higher levels is used to update the lower level models. However, the authors of that paper do not pursue a more generally applicable framework.

Besides general frameworks, there exist some algorithms that perform all levels of processing. These approaches will be less easy to adapt to changing environments, but as they were developed as a whole, at least basic communication between different levels will be available. For example, W^4 [Haritaoglu et al., 2000] combines shape analysis and tracking using a human appearance model. This is done using images from a static grayscale camera. Pfinder [Wren et al., 1997] uses a 3-D approach to track single persons in a room. It can handle complex scenes as long as there is no occlusion. A multiple camera approach for monitoring a large area is proposed in [Lipton et al., 1998]. It connects a number of cameras, enabling tracking of moving objects for extended periods.

Wrap-up

There is a need for a framework incorporating feedback. The framework should combine the results of algorithms on each of the levels of abstraction in a statistically correct way. Application knowledge should be incorporated in a structured way.

2.3 Image acquisition and pre-processing

Visual surveillance, as most computer vision algorithms, starts with a camera. For automatic visual surveillance, the choice between a static or a moving camera [Kruegle, 1995] is crucial as most algorithms designed for static cameras can not directly be used on video from moving cameras, and moving cameras add motion blur [Lagendijk and Biemond, 2005]. An equally important choice is that between color and black-and-white. Algorithms designed for color cameras usually cannot be used for black-and-white cameras.

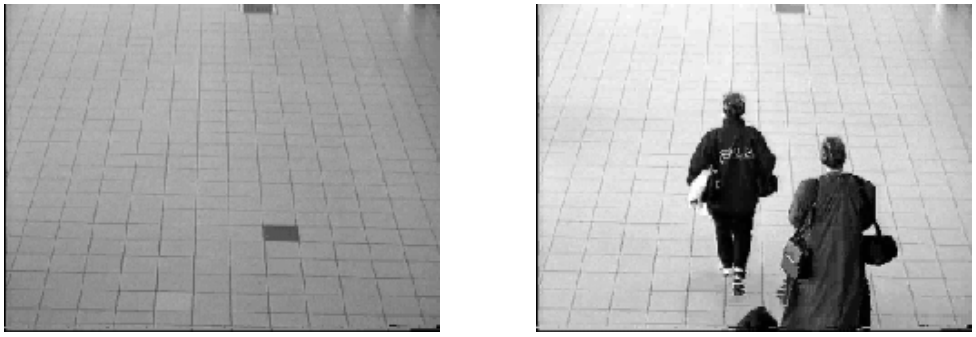
Camera models

Cameras used for video surveillance are mostly Charge Coupled Device (CCD) cameras [Castleman, 1996]. These are widely used for image processing applications, including visual surveillance. For many computer vision algorithms, including visual surveillance, it is important to have a model of the imaging process and temporal image noise. For example: a linear intensity model with gain and bias is used to derive equations for the correction of temporal changes in intensity for image compression [Kamikura et al., 1998]; a linear intensity model combined with a model for lens distortions is used for optical flow computation [Altunbasak et al., 2003]; a multiplicative model of the camera noise is used to set the threshold for moving object detection [Xie et al., 2004]; and a additive noise model together with a linear intensity model is used to derive a statistical test for object detection [Ohta, 2001].

Each author uses a model that fits his or her needs. However, little work is performed on the questions how accurate these models are for a specific camera and what the importance of the different components of the model are. A general model of the CCD sensor is introduced in [Healey and Kondepudy, 1994]. Healey models the CCD sensor, but not the post processing that is used in most modern CCD cameras. These most often use some kind of gamma adjustment to map the image in the available quantization range to obtain an image suitable for display [Castleman, 1996].

Global intensity changes

CCD cameras have a limited dynamic range. Everything outside this range will be imaged as either black or white. Therefore, constant illumination is very important using these cameras. This is generally not available in outdoor and most indoor scenes with windows. Many surveillance cameras are equipped with automatic control of their apparent gain to adapt to changing illumination. The apparent gain can be controlled by adjusting the gain, shutter time, iris or a combination of these. Using automatic processing, both changing illumination and an automatic apparent gain control often cause problems in the form of global changes in intensity. For changing intensity only this is evident, but also apparent gain control can cause



(a) Frame 220 (b) Frame 260

Figure 2.1: *Two frames recorded using a camera with automatic apparent gain correction. The scene changes when people with dark clothes enter the scene. Despite the constant illumination, the apparent gain is changed, causing the background to appear lighter.*

problems. For example, a problem occurs when a group of people with dark clothes enter a scene with a light background, see Figure 2.1. The change in overall image statistics will cause the apparent gain control to adapt, although the illumination of the static background did not change. This will lead to a changed appearance of the background, causing problems for algorithms expecting a static background.

Wrap-up

A model of the camera, if correct for the camera used, can be used for obtaining statistically optimal algorithms. It is then important to validate the model used.

Changes in global intensity prevent many algorithms to be used in outdoor applications or together with auto-gain cameras. A robust algorithm for correcting global changes in intensity would enable these algorithms to be used without decrease in performance.

Algorithms having difficulties with changing intensity include object detection algorithms using a background model [Priebe, 1994; Stauffer and Grimson, 2000], stereo matching [Scharstein and Szeliski, 2002] and optical flow computation [Bab-Hadiashar and Suter, 1998]. There is an extensive amount of literature concerning each of these algorithms. Considering that these algorithms assume constant image intensity, it is surprising how little work is done in the area of intensity correction. Some algorithms are available, but not an algorithm that is both robust and computationally efficient. A more detailed discussion of available techniques will be given in Chapter 3.

2.4 Detection and segmentation

Detection of moving objects and performing segmentation between background and moving objects is an important step in visual surveillance. Errors made at this abstraction level are very difficult to correct at higher levels. For example, when an object is not detected at the lowest level, it can not be tracked and classified at the higher levels. The more accurate the segmentation at the lowest level, the better the object shape is known and consequently, the easier the (shape based) object recognition tasks become.

An important first division between algorithms is whether they can operate on images from a moving camera or not. Only few algorithms can. Most algorithms compare images over time. In that case it is important that the camera is static, or that the images can be corrected for the motion. Correction can be performed by creating a panorama [Shum et al., 1998] or by motion compensation [Tian et al., 1996].

Motion detection and segmentation techniques can be divided into the four classes described below. All but the first class need a static camera or motion compensation.

Optical flow

Optical flow techniques can segment moving objects under camera motion, see Figure 2.2. Flow vectors are used to divide the image in segments with equal motion. Background motion will be different from that of moving objects, so moving objects will be segmented from the background. The advantage is that object motion is available as a by-product of segmentation. See [Barron et al., 1994] or [Dev, 1998] for a more in-depth discussion on optical flow.

The computational complexity of (dense) optical flow techniques is high. Real-time implementation is therefore difficult or expensive. Once moving objects are segmented, only the flow of those pixels assigned to an object is necessary. But for the detection of new objects always a global optical flow calculation is necessary, although this can be at a lower resolution or less frequent in time. Besides the computational complexity, another important disadvantage of optical flow in surveillance applications is that the flow is not always correct. It is undefined at object edges because of smoothing in the calculation of optical flow. This causes inaccurate object segmentation. The flow inside objects may also be wrong. For homogenous regions such as parts of a car for example, flow will be zero.

Space-time continuity

Space-time continuity considers moving objects as tunnels through the 3D *xyt*–space [Konrad and Ristivojevic, 2002; Niyogi and Adelson, 1994; Ricquebourg and Bouthemy, 2000], see Figure 2.3. In [Niyogi and Adelson, 1994] a smooth spatio-temporal surface is fit through the *xyt*–space. Speed is determined using the hough

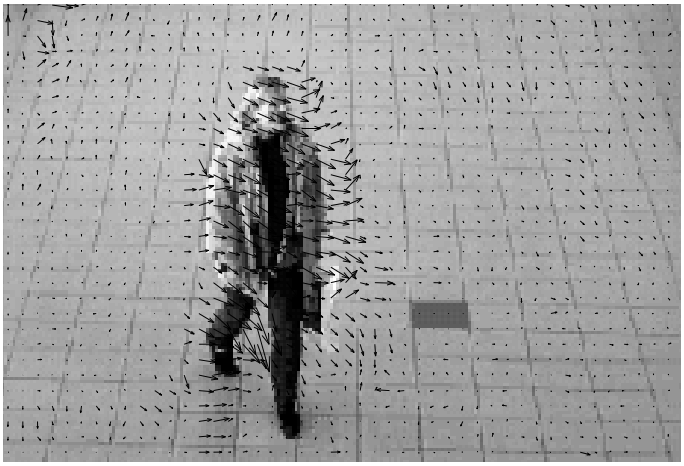


Figure 2.2: *This figure shows an image with its optical flow vectors.*

Figure 2.3: *On the left one frame of an image sequence is shown, on the right a number of xt–slices from the xyt–cube. The recurrent leg motion of a walking human can clearly be seen. From [Niyogi and Adelson, 1994].*

transform in an *xt*–slice, gait parameters are determined from the periodic motion of the legs.

An advantage is that objects are immediately tracked through the sequence. Also, recurrent motion like the motion of feet can easily be recognized. The drawbacks are that such algorithms tend to be computationally complex and require storage of many frames in memory.

Figure 2.4: *Example of the use of background modelling. On the left a frame from the input sequence is show. The center image shows the learned background. On the right the segmented foreground is shown.*

Temporal differencing

Temporal differencing is a simple, and therefore computationally attractive algorithm. A frame is pixel-wise compared to a previous frame and/or next frame. Everything exceeding a threshold is considered to be moving [Lipton et al., 1998]. Drawback of this technique is that in the case of uniformly colored objects, like cars, only the edges of the moving objects will be found. Temporal differencing also can be used as addition to other approaches such as background modelling [Collins et al., 2000a].

Background modelling

Background modelling spans a very popular class of approaches. The current image is compared to an object-free model of the background, see Figure 2.4. Usually, this model is learned over time. Together with pixel color, other parameters like the variance or extremes of each pixel can be learned.

Expectation Maximization (EM) [Dempster et al., 1977] is frequently used for learning a model of the background. The EM algorithm models the background by maintaining for each pixel a model of the probability density distribution of the color. This model consists of one or more Gaussian kernels in a mixture model. For each frame, the probability that a pixel is background is calculated by comparing its color to this mixture model. The online version of the EM algorithm [Priebe, 1994] allows a real-time implementation.

The advantage of such adaptive methods is that they learn the background from the images. The background is usually defined as the most frequent color over time. This means that objects that were in the scene when learning the background will gradually be replaced by the background. This eliminates the need of initialization with an empty background scene. It also provides a background model that automatically adapts when scene contents changes. For example when weather changes or a parked car drives off.

An important disadvantage is the trade-off between two conflicting demands [Toyama et al., 1999]: on the one hand updating should be performed fast to deal

with changes in illumination and changes in the background (Time of Day, Light Switch, Walking Person and Moved Objects problems); on the other hand, updating should be performed slow to avoid learning slowly moving objects as background (Bootstrapping and Sleeping Person problems). This makes algorithms using only one model sensitive to setting the update speed.

Wrap-up

For surveillance applications, approaches based on background modelling are most promising. Most surveillance cameras are static or have small temporal motion that is corrected for using motion compensation, so a computationally complex optical flow approach is not necessary. The use of space-time continuity is also computationally complex and requires a large amount of memory. Temporal differencing is computationally attractive, but it cannot handle homogeneously colored objects. Unfortunately, most algorithms use a shared model of foreground and background, causing false alarms and missed detections. This will be discussed in the next section.

2.4.1 Foreground and background models

Considering the models used for foreground/background segmentation, literature describes three methods:

- Using a model of the background
- Using models of the objects
- Using models of the background and objects.

The first method of detection and segmentation uses only **a model of the background** (and as implicit model of objects: anything that does not fit the background model). Only two output classes are available: background and foreground. Foreground/background segmentation introduced by Stauffer and Grimson [Stauffer and Grimson, 2000] is an example of this.

The second method uses only **a model of the moving objects**. This can be a pixel model of the color like in mean-shift tracking [Comaniciu et al., 2000; Zivkovic and Kröse, 2004]. Alternatively, it can be a model of the shape like in active contour tracking [Peterfreund, 1999]. Of course, both approaches can also be combined [Isard and Blake, 1998; Rasmussen and Hager, 2001]. Advantage of these approaches is that they can be used on imagery from a moving camera as easy as on imagery from a static camera. Disadvantage is the lack of automatic initialization of new tracks in such an approach.

The third and most advanced method uses **models of the background and objects** [Paragios and Deriche, 2000]. This makes it possible to calculate probabilities for each of the output classes. New objects can be initialized automatically and objects can be distinguished using their features.

The first method does not use a model of the objects for detection and segmentation. Association at this level is then simple, a pixel either belongs to the background or to a new blob. On the other hand, association on the blob-track level becomes more difficult. At that level the decision has to be made which blob or group of blobs associates with which track or group of tracks, so many-to-many association.

The second and third methods both use a pixel model for the moving objects. Such a pixel model is associated with the moving object, tracked over time. Therefore, the models must exist at track or object level. Association between pixels and blobs is now slightly more difficult. One pixel can be assigned to a number of available output classes. But association between blobs and tracks is given "automatically". Blob *i* is always assigned to track *i* (with the exception of the background), so one-to-one correspondence.

Besides complex blob-track classification, using only one model will also make more classification errors in difficult situations. As example, consider the popular approach for classification between foreground and background proposed by Stauffer and Grimson [Grimson et al., 1998; Stauffer and Grimson, 1999, 2000]. Their classification assumes that background colors are modelled in a limited number of kernels with highest prior over variance, see §4.2.2. A pixel is regarded to be described by one of these kernels when the pixel value lies within a threshold times the variance from the kernel average. When foreground and background colors differ only slightly, the kernels describing the actual color of the foreground and background will partly overlap. In such a case, using the approach proposed by Stauffer and Grimson will result in erroneously classifying more of the foreground pixels as background than necessary.

Algorithms using separate models for foreground and background are also more robust for setting the update speed. The models are only updated with data assigned to them. Slowly moving objects will therefore not automatically be learned into the background model, even with high update speeds.

2.4.2 Shadow

An important research topic in moving object segmentation is the handling of shadows. Often, shadows are detected as part of an object. This results in inaccurate object boundaries and may cause separate objects to appear connected by their shadows and consequently to be segmented as a combined moving object. Detection of shadows can be prevented by the use of intensity invariant color spaces [Greiffenhagen et al., 2001]. However, this causes missed objects, for example black objects in front of gray backgrounds will not be detected. Additional features or assumptions are required to detect these objects.

An overview of shadow modelling techniques is given in [Prati et al., 2003]. A simple approach uses the assumption that the intensity changes equally over small image regions [Ohta, 2001]. This allows for intensity correction for this region, eliminating shadows. More advanced is the use of additional features such as combining

color and intensity [Horprasert et al., 1999; KaewTraKulPong and Bowden, 2001] or adding gradient information [Javed et al., 2002; Javed and Shah, 2002]. In [Hsieh et al., 2003] a course-to-fine approach is used. First the rough shadow boundaries are obtained using a moment-based method. The rough approximation of the shadow region is then further refined using features such as the orientation, mean intensity, and center position of a shadow region.

Most authors make use of the knowledge that a shadow region must have lower intensity than the same region without shadow. For color images, each of the three (or more [Withagen et al., 2001]) color bands will have a lower value. Shadows are caused by the (partial) occlusion of one or more light sources. Given the color of the occluded light sources, the color of a shadow pixel can be calculated from its color without shadow [Nadimi and Bhanu, 2004]. This allows for a very robust shadow detection algorithm, causing less pixels to be erroneously classified as shadow and consequently not detected as moving object. However, not much work has been done exploiting this knowledge.

2.5 Object tracking and classification

Detected objects can be tracked and/or classified. The order of these two tasks depends on the application. If the objects are known beforehand, often object classification is performed first. This can be more efficient and robust. False detections will not be classified as one of the known objects, and consequently, they will not be tracked. Also, objects that have been split into several detections can be grouped by the recognition process. Then only one object needs to be tracked. Finally, once the object class is known, information about the object class can be used in the tracking algorithm. For example, maximal speed of cars differs significantly from that of humans. Examples of tracking known objects are tracking of hands [Isard and Blake, 1998], rigid bodies [Lipton et al., 1998; Polat et al., 2003] and people [Haritaoglu et al., 2000; Lipton et al., 1998; Polat et al., 2003; Wren et al., 1997].

Approaches that first do tracking and then classification are more flexible. All moving objects can be tracked, also those not specified in one of the object classes. This enables to initiate a track before the object is fully visible, for example when a person enters a scene behind a static car. In this approach, better classification results can be obtained because motion information can be used in the classification process. Some examples of doing tracking before classification are [Rosales and Sclaroff, 1999; Stringa and Regazzoni, 2000; Theil et al., 2000].

2.5.1 Object tracking

Four approaches for object tracking are generally distinguished [Hu et al., 2004b]: model-based, active contour-based, region-based and feature-based. Model-based techniques usually require object classification before tracking, other approaches can be used either before or after object classification, see also §2.5.2. Feature-

based techniques can also be used before object segmentation, only detection of the tracked features is needed.

Model-based

When a shape model of the tracked object is available, it can be fitted to the images in the sequence. This gives the position and motion of the tracked object, and at the same time an estimate of the object pose. This approach is very useful for tracking rigid-body objects such as robots [Yesin and Nelson, 2004] and cars [Dubuisson and Jain, 1994; Gardner and Lawton, 1996; Leuck and Nagel, 2001; Tan et al., 1998]. However, fitting a model to image data is computationally expensive.

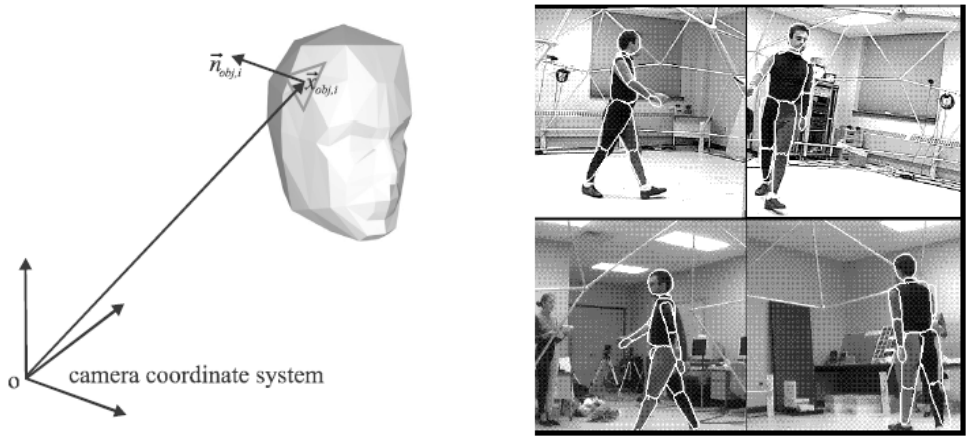
With respect to humans, body parts like the face [Decarlo and Metaxas, 2000], head [Paterson and Fitzgibbon, 2003; Zhang and Kambhamettu, 2002; Zivkovic and Heijden, 2001] and hands [Lu et al., 2003; Stenger, 2004; Stenger et al., 2001] are often tracked. See Figure 2.5(a) for an example of a head model. However, this is performed mainly in structured scenes where only one or two moving objects are in the camera view. Generally, many pixels are required on the moving objects. For full human body tracking [Leung and Yang, 1995; Wren et al., 1997] the demands on scene composition and number of object pixels are even more strict, for example only people walking parallel to the image plane are considered [Geurtz, 1993; Ju et al., 1996]. Often, several cameras are used to create a 3D scene reconstruction [Gavrila, 1996; Kakadiaris and Metaxas, 1996], see Figure 2.5(b). See also §2.5.2 for a description of the models used. As the models for humans are deformable, computational complexity rapidly increases with the amount of details in the model.

Active contour-based

After segmentation, the outline of an object is known. The outline can be tracked using active contours or snakes [Baumberg, 1995; Nguyen et al., 2002; Peterfreund, 1999], see Figure 2.6 for an example. Using energy minimization, the contour is adapted to the image data. Besides tracking of the object, an accurate object contour description is now available each frame.

At high computational cost, the active contour approach is able to describe arbitrary shapes, as long as the smoothness constraint is satisfied. However, for this approach initialization and association between frames is a problem, for example when objects form groups and split again. Another disadvantage of active contours is that tracking is based on the most deformable part of the object, its contour.

To overcome the limitation of association, the active contour algorithm can be combined with an algorithm describing the color of the object [Isard and Blake, 1998; Rasmussen and Hager, 2001].



(a) Model of a head

(b) 3D multi camera tracking

Figure 2.5: *Two model-based object tracking examples. On the left the model of the head used in [Zivkovic and Heijden, 2001] is shown. On the right an example of 3D tracking using multiple cameras is shown, from [Gavrila, 1996].*

Figure 2.6: *An example of active contour-based tracking. Three frames are shown with the active contour in white. From [Nguyen et al., 2002].*

Region-based

When a moving object is segmented, a region of pixels assigned to the object is available. This region can be tracked using approaches like cross-correlation. The location of the region in the next frame is to be determined. A moving object usually corresponds to one [Wren et al., 1997] or several [McKenna et al., 2000] tracked regions. Combination of several regions to one object is then performed at a higher level of abstraction.

Several techniques are available for modelling and tracking image regions. The regions are often modelled using a probability density distribution of their color. This distribution can be described using a color histogram [Agbinya and Rees, 1999; Capellades et al., 2003], or a mixture of Gaussian kernels [McKenna et al., 1998; Raja et al., 1998]. Instead of using one 3D probability density distribution, separate distributions for each of the colors can be used [Gasser et al., 2004; Hu et al., 2004a].

Probability density distributions of the color are relatively invariant to changes in object orientation, scale, partial occlusion, viewing position and object deformation [Swain and Ballard, 1990]. This makes them particularly interesting for tracking nonrigid objects such as humans. However, the distributions capture only the colors in an image and do not include any spatial correlation information. Therefore, they have limited discriminative power. A color correlogram, on the other hand, is a co-occurrence matrix that gives the probability that a pixel at a distance d from a given pixel of color $\vec{c}_i$ is of color $\vec{c}_j$. This way spatial information in the form of distance to pixels of a certain color is introduced [Capellades et al., 2003; Huang et al., 1998].

Other approaches taking spatial information into account are using many small regions [McKenna et al., 2000] and using the time average per-pixel color [Cucchiara et al., 2004; Senior, 2002]. Instead of choosing one color space, automatic selection of most discriminative features can be used [Chen et al., 2004; Collins and Liu, 2003]. This adapts the color space used by comparing the specific tracked region with the local background, leading to a more precise object segmentation. However, when pixels are misclassified and consequently used for updating the wrong model, this solution will become unstable.

Considering the low number of pixels in each tracked region, histograms will become quite sparse. On the other hand, it also is not easy to estimate the parameters of a Gaussian mixture model from only a few data points, specially when also the number of kernels is unknown. From the point of view of computational complexity, the use of a histogram approach is most affordable because template matching can be performed very efficiently.

Considering that a probability density function is available with these techniques, it is unfortunate that object segmentation is often based on a static threshold. Calculated probabilities could be used in a probabilistic foreground/background classification algorithm.

Moving objects can also be modelled using a fixed or parameterized shape, like in the mean-shift approach [Comaniciu et al., 2000; Zivkovic and Kröse, 2004], and

the particle filter [Nummiaro et al., 2003; Pérez et al., 2002; Spengler and Schiele, 2002]. Disadvantage of such techniques is that they are unable to describe an arbitrary shape, changing between subsequent frames.

Feature-based

Feature-based object tracking approaches are similar to region-based approaches. Instead of tracking the entire region, feature-based approaches extract features from the image and track these. Examples are tracking of line segments and corners [Coifman et al., 1998] and motion of the centroid [Polana and Nelson, 1994]. Features such as moment invariant functions and aspect ratio [Withagen et al., 1999] could also be used.

Such approaches can be implemented very efficiently and are theoretically able to handle partial occlusions. They are used frequently for traffic surveillance. Reasons they are not used often in general surveillance applications include: low recognition rate of features due to nonlinear perspective transformation and the stability of dealing with occlusions is generally poor [Hu et al., 2004b].

Wrap-up

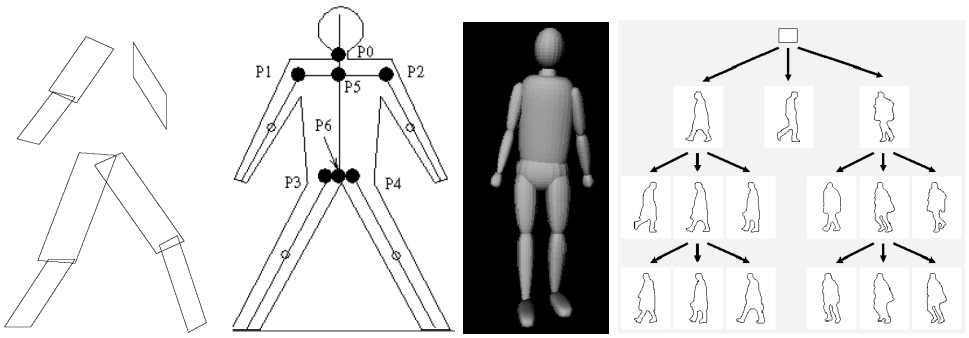
For surveillance applications, model-based approaches are computationally complex and often the amount of pixels required for an accurate fit of the model is not available. Feature-based approaches should also not be used, because of the low recognition rate of features and problems with occlusion.

Active contour-based approaches have the advantage of generating an accurate object contour. This can be a major advantage for applications requiring pose estimation for example. However, these approaches are generally more computational complex than region-based approaches, and require object detection by another algorithm. Occlusion and robustness over many frames can pose problems, as tracking is based on the most unstable part of the object. If computational complexity is not an issue, the combination of region-based tracking and active contour-based tracking is optimal, in other cases region-based tracking is a good choice.

The region-based approach is most popular for surveillance applications. It is computationally fast compared to approaches based on active contours and object models, and as region-based methods use the entire region, they are more stable and can better cope with occlusions than feature-based approaches. This approach will be used in this thesis.

2.5.2 Object classification

Two main categories of object classification approaches exist. Classification can be based on either motion or static features such as shape, color and texture. A description of both approaches will be given below.



(a) Cardboard (b) Stick figure (c) Ellipsoid (d) Silhouette

Figure 2.7: *Some models used for model-based human body tracking. The pictures are taken from [Ju et al., 1996], [Leung and Yang, 1995], [Sminchisescu, 2002] and [Gavrila and Philomin, 1999] respectively.*

Motion-based classification

Classification after tracking the object for a few to several frames allows the use of temporal characteristics of the object for classification. A number of researchers utilize the periodic motion of walking people [Cutler and Davis, 2000; Lipton, 1999; Stauffer, 1999]. For classification between humans and for example vehicles speed and motion deviations are used [Brown, 2004].

Static feature classification

Explicit shape models are available in several flavors. For people there are cardboard [Haritaoglu et al., 2000; Ju et al., 1996], stick figure [Leung and Yang, 1995], ellipsoid [Geurtz, 1993; Sminchisescu, 2002] and silhouette models [Gavrila, 2000], see Figure 2.7 for some examples.

Alternatively, blob features like aspect ratio, bounding box or blob area are used [Collins et al., 2000b; Lipton et al., 1998]. Classes that are generally used are human, group of humans, car, bicycle and other.

Combined approach

Combining motion and static features allows to use the advantages of both approaches. Examples of a combined approach are [Haritaoglu et al., 2000; Stauffer, 1999]. The latter uses a hierarchical approach to recognize both objects and behavior.

2.6 Behavior and activity recognition

After detection, segmentation and tracking of moving objects, higher level tasks such as activity classification and behavior recognition can be performed. For surveillance applications, two approaches are important. First, the behavior in a short time is important. Is someone throwing something, hitting or kicking, or falling on the ground. Detection of this kind of behavior is a result of analysis of the pose in a short time, seconds or less [Pronk, 2004; Schelven, 2002]. Second, the behavior during an extended time is important. Based on their motion pattern, people with criminal intent such as car thieves, drug dealers and pickpockets can be detected. Recognition of this kind of behavior is based on the trajectory over an extended time, minutes or more [Ockhuysen, 2006].

A representative choice of approaches will be given below. It goes beyond the scope of this thesis to give detailed discussion of available techniques.

2.6.1 Pose analysis

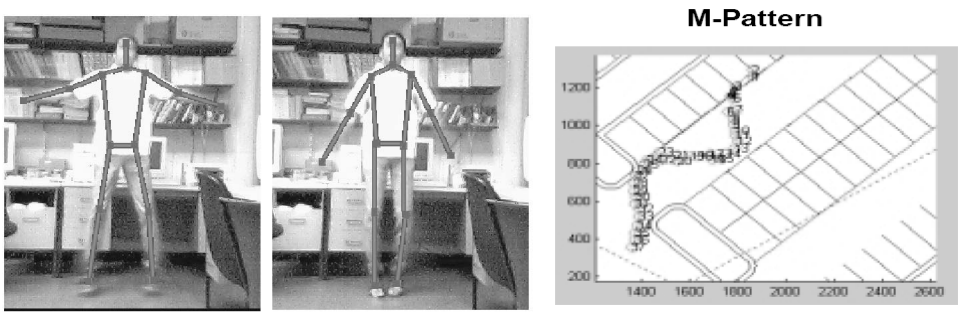
Work on activity recognition from pose can be organized into approaches either based on state-spaces or on template matching [Aggarwal and Cai, 1999]. Approaches using a state-space assume that an action can be represented as a series of states. These states correspond to static postures of the human body. Probabilities are calculated for transitions from one state to another. An advantage of these approaches is that variations in motion are part of the model. Disadvantages are that the calculations are complex, the states do not necessarily correspond to physically meaningful states and that it is difficult to add another action for recognition.

Only a few body poses are sufficient to recognize an action from the motion of the body parts [Ben-Arie et al., 2002]. These poses consist of angular and velocity vectors for the major body parts (hands, feet and torso). To combine the individual results from the body parts a voting approach is applied. Actions like walking, sitting down and jumping are recognized, see Figure 2.8(a). An armed robbery is modelled in [Dever et al., 2002] as a single pose of the arms for different people. One person has to have an arm positioned horizontally and the other person has to have his hands up. These poses are recognized from the silhouette.

Approaches using template matching extract features from image sequences and compare these to stored feature patterns. The main disadvantage of these approaches is their sensitivity to the variations in motion of different people or repeated performances.

The coordinates of the trajectory of the human body are also used for detecting interaction between people [Park and Aggarwal, 2003]. In their paper the interaction is assumed to be parallel to the camera plane, which means that there is a potential interaction when trajectories cross (passing by), follow the same path (following or travelling together) or stop close to each other (talking).

In [Guo et al., 1994] a Fourier analysis of the motion vector containing the position of the head, the direction, length of body parts and angles between joints



(a) Pose analysis (b) Trajectory analysis

Figure 2.8: *Two examples of high level processing. On the right an example of the jumping action from [Ben-Arie et al., 2002] is shown. On the left a suspicious trajectory typical of a car thieve is shown, from [Pavlidis et al., 2001].*

reveals the periodicity or frequency. This is used to determine walking, running or other periodic motion. Four coefficients of discrete Fourier transform are sufficient to use in combination with a neural network to recognize actions.

2.6.2 Trajectory analysis

Two approaches are distinguished for the analysis of trajectories. Suspicious trajectories can be explicitly defined using expert knowledge, leading to an expert systems such as hidden Markov models and decision trees. Alternatively, suspicious behavior can be learned from examples, for example using a neural network.

An expert system has been developed for automatically recognizing dealers at bus stops [Gasser et al., 2004] . However, this application is very simplified: if someone lets a certain amount of busses pass by and keeps hanging around, he is classified as a drug dealer.

DETER [Morellas et al., 2003; Pavlidis et al., 2001] is an expert system specialized in monitoring large open spaces like parking lots, plazas, crossroads, and perimeters of large industrial structures, see Figure 2.8(b). It can detect basic acts of behavior such as people running, cars speeding, and people walking in a strange trajectory, taking time zones (time within the day) and hot zones (certain designated areas within the perimeter of a building) into consideration.

Using a neural network, the system in [Sotelino et al., 1994] can find temporal relationships independent of position in time. It is based on the time trajectory generating the pattern instead of the static pattern itself. Learning is based on examples and no explicit expert knowledge is necessary.

2.7 Conclusions

In this chapter an overview of visual surveillance literature has been given. A visual surveillance application consists of different levels of abstraction. For each of the levels, optimal algorithms are application dependent. A general framework is required to integrate the separate algorithms in a proper way.

A framework should combine the good performance of algorithms at each of the levels of abstraction. This framework should preferably be based on probabilities, in order to deal with uncertainties. In the framework, it should be easy to add or replace algorithms from one of the levels, in order to adjust the algorithm for a different environment or task. Besides communication of the results from low to high levels, a framework should also allow for feedback, the use of results at higher levels to update low-level models. Such a tracking framework is not yet available, it will be presented in this thesis.

The lowest level of abstraction in a tracking application is formed by image acquisition and pre-processing. The type of camera and its placement define which algorithms can and cannot be used. A model of the imaging process of the camera is useful for determining the amount of noise and the effect of changing illumination. Often, changes in the global intensity are present and need to be corrected. Robustness of computationally efficient algorithms is still an essential issue.

Many techniques are available for the detection and segmentation of moving objects. For visual surveillance applications using a static camera the techniques based on a background model are most useful. However, the available algorithms are not very robust for parameter settings. Specifically, setting the update speed of the background can have a great impact on the performance, while different settings are optimal in different circumstances. Second problem is that most algorithms depend on only one model. Therefore, they are subject to false alarms and missed detections. Using separate models of the background and the objects improves both performance and robustness to setting the update speed.

Shadow detection is necessary to avoid false alarms. Most available shadow removal techniques do not exploit the color of the illumination. This causes unnecessary false alarms. This issue will be addressed in this thesis.

The choice of object tracking and classification algorithms depends on whether a model of the tracked objects can be used. A model-free approach is more flexible and allows tracking of partially visible objects. Region-based object tracking is then the best choice for visual surveillance applications. However, the combination of region-based object tracking and probability-based object-pixel classification would improve both accuracy and robustness to partial occlusion.

Chapter 3 Global intensity correction in dynamic scenes

3.1 Introduction

Computer vision has proved very successful in well-constrained industrial environments (for instance when illumination, objects types, and orientations are known). However, in many practical applications, including airborne or remote sensing, medical imaging, face recognition, outdoor robotics, and surveillance applications, the environment can hardly be controlled. Illumination changes by lights switching on or off, or by clouds moving in front of the sun. Automatic gain control (AGC), white balance and iris are often applied to optimally map the amount of reflected light to the digitizer dynamic range. However, when scene content changes they cause changing image intensity over time.

Problems then arise with many algorithms that assume Constant Image Brightness (CIB) or that are based on the Brightness Constancy Constraint Equation (BCCE). Applications where image intensity changes cause problems include background subtraction, object tracking, stereo matching, image retrieval, video coding and optical flow computation, see Section 2.3.

In this chapter[1] we introduce algorithms for correcting global intensity changes in image sequences. The algorithms estimate and correct for global temporal intensity variations. They can be applied as pre-processing step for other image processing algorithms as mentioned above. Based on a model of a CCD camera a number of algorithms are proposed. The algorithms are evaluated on both simulated and real images.

Evaluation of the algorithms is performed on three criteria. First, the precision of the parameter estimation is evaluated using the sum of squared differences between a reference image and the corrected image. Second, the usability of the

[1] This chapter is based on [Withagen et al., 2004a].

correction is evaluated using the performance of a representative post processing algorithm. Third, the robustness of the post processing is evaluated. For all evaluation methods we will show the impact of outlier removal.

For real-world application we will focus on the foreground-background classification problem. We use the popular online Expectation Maximization (EM) algorithm to estimate a multi-Gaussian model of the background color for each pixel, see Chapter 2.

This chapter is structured as follows: in Section 3.2 we discuss existing techniques to handle changes in intensity. We will introduce our model of changing intensity in Section 3.3. There we will consider the differences between changes in intensity caused by the combination of a changing scene and automatic gain control, and by changing illumination. In Section 3.4 the proposed methods are described. These methods are evaluated by both simulation and experiments on real images in Section 3.5. Finally, conclusions are presented in Section 3.6.

3.2 Previous Work

There exists an extensive amount of literature concerning applications like moving object detection, stereo matching or optic flow calculation. Considering that these algorithms normally expect constant image intensity, it is surprising how little work is done in the area of intensity correction. In this section we shortly introduce the different techniques that are available for intensity correction. We evaluate them on accuracy, usability and computational complexity.

The intended algorithm will be used for dynamic scenes with moving objects. These objects cause a scene change and can decrease the accuracy of the intensity correction. It is therefore important to have outlier removal, as will be shown in the experiments in Section 3.5.

The simplest way of dealing with changes in intensity is ignoring all intensity information. Intensity invariants [Siebert, 2001] or normalized colors can be used. Instead of intensity information other features can be used, for example color [Greiffenhagen et al., 2001; Horprasert et al., 1999], edges [Jabri et al., 2000; Javed and Shah, 2002], or depth [Harville et al., 2001]. However, using these features disregards the information available in the intensity.

Techniques dealing with local changes could be used to correct for global intensity changes. The drawback of local techniques is that they ignore the fact that all pixels change simultaneously. This leads to a less accurate estimate of the global effect. These techniques will be further discussed in the context of shadow removal in Chapter 6.

Literature reports complicated methods: dynamic histogram warping changes image intensities such that the histograms of the two images become equal [Cox et al., 1995], by an iteratively weighted least squares estimation, the bias, gain and gamma of an image can be estimated and corrected for [Tsin et al., 2001], and estimation of the gain and bias together with optic flow allows the use of optic flow

under global intensity changes [Altunbasak et al., 2003]. Drawback of these methods is the high computational complexity. This makes real-time implementation difficult and expensive. Also, [Cox et al., 1995] and [Altunbasak et al., 2003] do not perform outlier removal.

Considering background modelling techniques as used in this thesis, a frequently used approach for dealing with changes in image intensity is relying on the adaptation speed of the background modelling technique [McKenna et al., 2000]. However, this adaptation will only resolve the problem for relatively slow changes in intensity, and it is difficult to tune the update speed of the model. A high update speed may learn slow objects into the background model (missed objects), while a low update speed can be unable to adapt the model fast enough to cope with changes in intensity (false alarms), see [Toyama et al., 1999]. No single update speed guarantees acceptable results for all possible situations, see Section 3.5.

An approach, closely related to relying on the adaptation of the background classification algorithm, is setting a limit on the fraction of allowed foreground pixels (e.g. 70 %). When this fraction is exceeded another background model (if available) is chosen or the background model is re-initialized [Javed and Shah, 2002; Toyama et al., 1999]. The performance will decrease in applications where illumination or gain changes frequently occur, because after re-initialization a new background model must be learned. During each learning period classification results are unreliable.

The idea of using multiple instances of a scene taken under different illumination conditions is also considered in [Hager and Belhumeur, 1998; Hischof et al., 2004]. They use an eigenspace method to overcome complex changes in illumination (not only intensity but also illumination direction may change). An interesting approach in conjunction with image retrieval is given in [Jacobs et al., 1998]. Their method assumes that the pixel-wise image ratio of two images from the same object is simpler than the ratio of two different objects (they define simplicity as based on the complexity of the algebraic function needed to locally approximate the shape of the image). This assumption allows for object comparison under complex changes in illumination. Only one measure of similarity is calculated for the entire image. Therefore, usability is restricted to applications like face recognition and image retrieval, so this is less general than the scope of our research, were we are aiming at a generally applicable method.

A useful approach for real-time applications is the direct calculation of the intensity difference between two images. This is done using the average [He et al., 2003] or a least squares estimate [Kamikura et al., 1998]. However, these methods are sensible to outliers.

In this chapter we develop a method for the correction of global changes in intensity for dynamic scenes that overcomes the above limitations. The method should be generally applicable for static cameras where moving objects are part of the scene. These moving objects can be regarded as outliers in the estimation of the global intensity. The method to be developed should be robust to these outliers. Furthermore, it should have low computational cost so that it can be implemented

in real-time.

3.3 Model Description

In this section we present a correction algorithm for global changes in intensity. We then introduce a simplified model of a CCD camera. This model has been experimentally verified for a range of cameras in Appendix A. Using this model we then introduce the apparent gain factor that needs to be estimated.

3.3.1 Global intensity correction

The goal of this chapter is to correct for global differences in intensity between two images. Consider two images i_r and i_t depicting an equal scene at different time instances. There is global difference in intensity between the two images. We intend to correct for this intensity difference by

$$i_{t,\text{corrected}} = \frac{i_t}{a} , \tag{3.1}$$

with a the apparent gain factor. $i_{t,\text{corrected}}$ and i_r have equal global intensity. We will give an equation for a in this chapter.

Equation 3.1 is not valid for moving objects, causing different scene content in the two images. In estimating the apparent gain factor, pixels corresponding to changed scene content will cause outliers. Correction for such outliers will be discussed in §3.4.2.

For color cameras we will assume that the intensity of all color bands changes equally. This results in one apparent gain factor for each image.

3.3.2 Model of CCD cameras

Based on a general model of CCD cameras [Healey and Kondepudy, 1994], we report in Appendix A the experimental results of the different contributions in the CCD model. The experiments show that the general model in [Healey and Kondepudy, 1994] does not hold for a typical webcam. The other cameras evaluated do adhere to this model with added gamma correction.

For the cameras used in Appendix A that do adhere to the model, the experimental results allow for simplification of the model. For sufficiently large intensity values, both offset and additive noise can be neglected. The simplified model of the CCD camera to be used in the remainder of this chapter is now given by

$$i_t = g_t^{\gamma} \left(h_t i_0 + N_S \right)^{\gamma} , \tag{3.2}$$

with g_t the camera gain, i_0 the scene irradiance, h_t a factor related to the camera shutter time, iris size and scene illumination, N_S noise and γ the value of the gamma function. The amount of noise N_S is given by $\sigma^2_{N_S} \sim h_t i_0$. It depends on the amount of light reaching the pixel and is therefore called multiplicative.

3.3.3 The apparent gain factor

The image recorded at time t will be compared to some reference image r recorded earlier. These images are given by

$$i_t = g_t^\gamma \left(h_t i_0 + N_S\right)^\gamma \quad \text{and} \tag{3.3}$$

$$i_r = g_r^\gamma \left(h_r i_0 + N_S\right)^\gamma . \tag{3.4}$$

Changes in image intensity can be caused by either changes in the camera gain g_t or changes in the illumination intensity, iris or shutter resulting in a changed h_t. γ is considered constant in our model. Experiments in Appendix A show that this is a valid assumption for most cameras.

Considering equal scene, the relation between i_t and i_r is

$$i_t = a i_r + N_{\text{total}} , \tag{3.5}$$

with a equal to that defined in Equation 3.1. As noise N_{total} is zero-mean, this shows that Equation 3.1 can be used for global intensity correction.

The apparent gain factor a is for sufficiently large values of h_r and g_r given by:

$$a = \left(\frac{g_t h_t}{g_r h_r}\right)^\gamma . \tag{3.6}$$

h_r and g_r are sufficiently large when the signal is dominant over the noise. The apparent gain factor is undefined for pixels with small h_r or g_r. Therefore, pixels with small intensity will not be used in the estimation of a.

Considering the noise of the two images independent gives for the variance in N_{total}:

$$\sigma^2_{\text{total}} \approx \left((a^2 g_r^2 h_r + g_t^2 h_t) i_0 \sigma_t^2\right)^\gamma , \tag{3.7}$$

with σ_t^2 he variance in the noise in each of the images.

Regardless of the cause, the effect on the apparent gain factor a is the same, see Equation 3.6. Therefore, without loss of generality we can limit ourselves to the estimation of a in the remainder of this chapter. However, there is a difference in effect on the image noise. When the change in intensity is caused by camera gain the increase in noise is different than when the illumination intensity, iris or shutter time changes[2].

3.4 Proposed algorithms

To correct for changing intensity we need to estimate the apparent gain factor a between two images i_r and i_t defined in §3.3.1. To obtain constant intensity over

[2] For an equal change in intensity, the variance in the noise $\sigma^2_{\text{total},g}$ caused by changing camera gain is approximately $\sigma^2_{\text{total},g} = \frac{1+n}{2}\sigma^2_{\text{total},h}$, for $\gamma = 0.5$ and with n the amount of change $n = \frac{g_t}{g_r}$ or $n = \frac{h_t}{h_r}$ respectively.

time we compare all images to the reference image i_r. This results in the apparent gain factor between the reference image and each other image. These apparent gain factors will be used to correct the corresponding images.

Because of the noise in both images, this parameter estimation problem is not trivial. Besides a theoretically optimal estimate, several alternative algorithms will be given here. They will be compared experimentally in Section 3.5.

3.4.1 Estimation of the apparent gain factor

We intend to give an accurate non-iterative estimate of the apparent gain factor a. The less computation an algorithm requires, the easier it can be implemented in real-time. Therefore we also present simplifications to the theoretically optimal algorithm. In Section 3.5 we will experimentally compare these algorithms.

As the majority of the noise contributions are multiplicative, a Weighted Least Squares (WLS) estimate is a theoretically optimal non-iterative estimate. Minimizing the criterion

$$L_2 = \sum_{s \in S} w_s^2 (i_t - a i_r)^2 \ , \tag{3.8}$$

for all pixels s in set S in least squares sense gives

$$a_{\mathrm{WLS}} = \frac{\sum_{s \in S} w_s^2 i_{r,s} i_{t,s}}{\sum_{s \in S} w_s^2 i_{r,s}^2} \ . \tag{3.9}$$

The weights w can be calculated using the inverse of the image noise, which depends on the intensity. Using the common simplification of only multiplicative (shot) noise leads to the following weights

$$w_s^2 = \frac{i_{r,s}}{i_{t,s} i_{r,s} + i_{t,s}^2} \ , \tag{3.10}$$

using the approximation $a = \frac{i_{t,s}}{i_{r,s}}$ in the weights. This will however increase the influence of low intensity values. For these values the simplification used is not valid, as the amount of additive noise will be larger than the amount of multiplicative noise.

To reduce the influence of low intensities and at the same time reduce computational requirements, we can use equal weights. This gives us the standard Least Squares (LS) estimate

$$a_{\mathrm{LS}} = \frac{\sum_{s \in S} i_{r,s} i_{t,s}}{\sum_{s \in S} i_{r,s}^2} \ . \tag{3.11}$$

Even simpler and requiring fewer computations is using only the Quotient of the Average (QofA), related to the L_1 criterion:

$$a_{\mathrm{QofA}} = \frac{\sum_{s \in S} i_{t,s}}{\sum_{s \in S} i_{r,s}} \ . \tag{3.12}$$

All methods given above calculate the quotient of two numbers. For statistical outlier removal, see §3.4.2, it would be profitable to have an intensity ratio per pixel. This can also be useful for the extension to local intensity correction, see Chapter 6. Therefore, we also take the Average of the pixel-wise Quotient (AofQ) into account

$$a_{\text{AofQ}} = \frac{1}{|S|} \sum_{s \in S} \frac{i_{t,s}}{i_{r,s}} \ , \tag{3.13}$$

with $|S|$ the number of pixels in the set of pixels S.

The experiments will show that it is important to perform outlier removal when dealing with dynamic scenes. Also, the quotient between two images typically has a positively skewed distribution where the mean would over-estimated the apparent gain factor. For these reasons the median is also taken into account

$$a_{\text{QofM}} = \frac{\mathsf{M}_{s \in S} i_{t,s}}{\mathsf{M}_{s \in S} i_{r,s}} \quad \text{and} \tag{3.14}$$

$$a_{\text{MofQ}} = \mathsf{M}_{s \in S} \frac{i_{t,s}}{i_{r,s}} \ , \tag{3.15}$$

where M denotes the median of a set of numbers.

3.4.2 Outlier removal algorithm for dynamic scenes

Comparing the two images i_r and i_t we should take into account that not all pixels will be stationary in dynamic scenes. Pixels depicting dynamic scene violate the assumption of equal scene introduced in §3.3.1. The equations given above are thus only valid for pixels depicting stationary scene. Non-stationary pixels should be excluded from the apparent gain factor estimation.

We will investigate a outlier removal based on statistics. For applications using foreground/background classification, moving objects will cause outliers. In such case classification between foreground and background can be used for outlier removal. However, this makes the global intensity correction algorithm less general.

Statistical outlier removal is based on the observation that the majority of the pixels depict the same scene in both images. We calculate the average μ_q and standard deviation σ_q of the pixel-wise ratio q_s between the two images. Pixels for which $|q_s - \mu_q| < T_{\text{outlier}} \sigma_q$ holds are labelled as outlier. T_{outlier} should be chosen such that enough pixels remain for a statistically accurate estimate, but that only those pixels remain that are very unlikely to be outliers. We will use $T_{\text{outlier}} = 1$. This is a simple and easy to compute outlier removal algorithm that proves very effective.

Another set of pixels that should not be taken into account are pixels that are close to either the upper or lower bound of the range of pixel values. For pixels close to the lower bound the apparent gain factor is undefined and additive noise and dark current cannot be neglected, while pixels close to the upper bound may suffer from saturation problems. Therefore, we will ignore pixels within 10 percent of the upper and lower bound.

3.4.3 Creation of the Reference Image

All algorithms proposed need a reference image to compare the current image to. In our experiments, we will use the first image of the image sequences as reference image. This is the most general solution and requires the least amount of computations. For most applications however, the reference image should be updated as the scene might change. Some alternatives will be discussed below.

The reference image can be periodically renewed by selecting a new image from the input. The image can be selected at random, or based on the amount of background it contains. The latter is preferred as it will contain less moving objects.

When using background modelling, see Chapter 2 it is possible to use the background model as reference. With a Gaussian mixture model, the average of the kernel with the largest weight can be used to compare the current image to, but it is also possible to use the mean of the best fitting kernel. The latter is expected to give better results for multi-modal backgrounds. Compared to choosing an image from the sequence, we expect the amount of noise and the number of foreground pixels in the EM–model reference image to be lower. An additional advantage of using the EM–model to create the reference image is that it is always up-to-date. When the background changes, the reference image is automatically adapted. Drawbacks are a small amount of additional computation.

For a slowly moving camera, each image can be compared to the previous image. If the motion is (approximately) known, only the overlapping area between the images should be used for apparent gain factor estimation.

3.5 Experimental Evaluation

We will experimentally evaluate the proposed algorithms. We will use simulated images to evaluate the accuracy and usability of the different intensity estimation algorithms in §3.5.1. We compare the algorithms to each other and to ground truth. Also, the need for outlier removal is evaluated. The effect of intensity correction in conjunction with classification between foreground and background on real images is demonstrated in §3.5.2. We look into the runtime of the algorithms in §3.5.3. A discussion of experimental results is given in §3.5.4.

For the evaluation of the usability we demonstrate intensity correction in conjunction with foreground/background classification. For each image the apparent gain factor is calculated and the image is corrected for it. With this corrected image, a model of the background is updated using the online Expectation Maximization (EM) algorithm, see Section 4.2. Four kernels are used to model the background. They are updated with an update speed α_B. The classification algorithm proposed there is used to do classification between foreground and background, see Section 4.2.2.

3.5.1 Comparison by simulation

We will evaluate the proposed algorithms using a simulated image sequence. For each image the apparent gain factor a is estimated and the image is corrected according to Equation 3.1. The corrected image is used to measure the performance of the intensity correction algorithm and to perform foreground/background classification.

Image generation

Images are generated based on the full model of the CCD given by Equation A.3, so without simplifications. As scene we use the red channel of the first image from the Intratuin image sequence. In each frame 20 % of the pixels is selected at random and they are given a foreground value. The foreground value is drawn from a uniform distribution between 0 and 0.5, while the distribution of the background lies between 0 and 1 with peaks around 0.3, 0.6 and 0.8, see Figure 3.1(a) for a noiseless example image without foreground objects and Figure 3.1(b) for its histogram. We choose the parameter settings of the image simulation such that they approach an average camera from the cameras characterization in Appendix A. Artificial Gaussian distributed noise is added to the images according to Equation A.3.

The scene intensity is kept constant $h_t = h_r$ and the camera gain g_t is varied, see Figure 3.2. The camera gain is 1 for images 1 to 150 for training and evaluation of the algorithms on images without changes in intensity. The intensity linearly decreases from 1 at image 150 to 0.5 at image 200 for the evaluation of the algorithms on images with a changing intensity.

Besides the image sequence i_t, two additional images are created. The reference image i_r which is used for the estimation of the apparent gain factor together with the current image, and the ground truth image i_{gt}. These images have equal illumination and camera gain and do not have foreground pixels. The ground truth image is noiseless, the reference image contains noise.

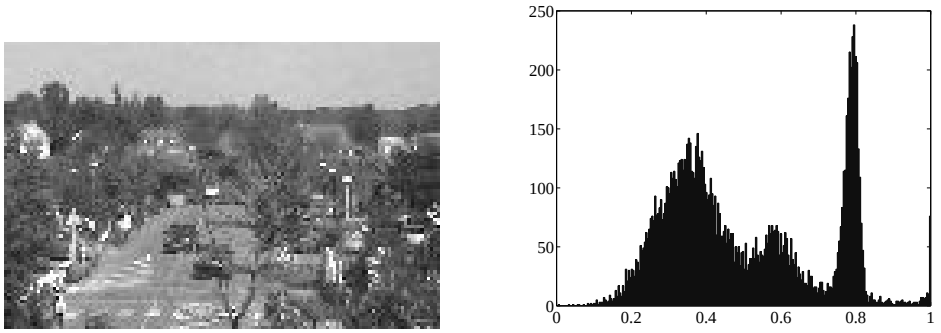
Evaluation criteria

Two criteria for evaluation are used, one for the accuracy of the apparent gain factor and one for the usability in conjunction with foreground/background classification.

For accuracy evaluation we use the average root mean squares error between the intensity corrected current image and the ground truth image for all pixels depicting background:

$$e_{\text{accuracy}} = \frac{1}{|B|} \sum_{b \in B} \left(i_{\text{gt},b} - \frac{1}{a} i_{t,b} \right)^2 , \tag{3.16}$$

with $|B|$ the number of pixels in the set of background pixels B. Note that as we use a realistic setting with noise in the current image, this criterion gives for perfect intensity correction the standard deviation of the image noise.



(a) Simulation image (b) Histogram of simulation image

Figure 3.1: *Simulation image and its histogram. The image on the left shows an example image used for the simulation experiments. On the right the histogram of this image is given. This image and its histogram does not contain noise or foreground pixels.*

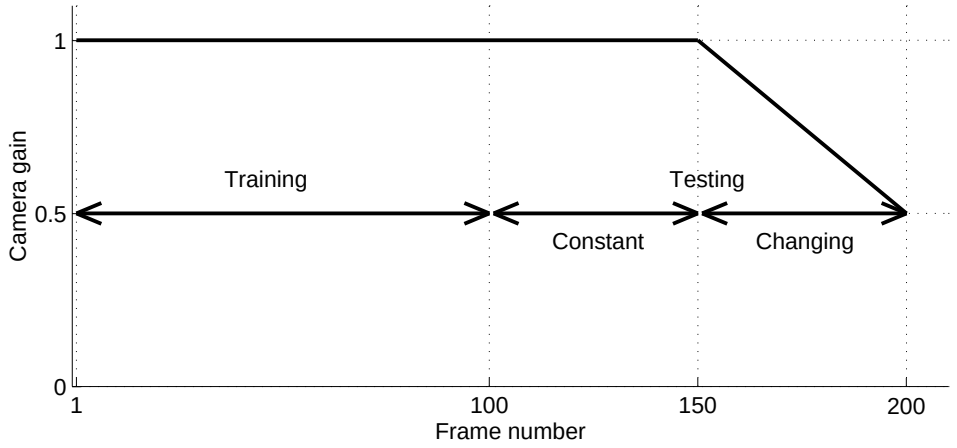


Figure 3.2: *The value of the camera gain for the simulated images.*

The usability of the intensity correction algorithm is evaluated with foreground/-background classification. A Mixture of Gaussians model is updated using online Expectation Maximization, see Section C.3, with an update speed $u_B = 0.05$. With this model, classification is performed using the algorithm of Stauffer with a threshold $T_{Stauffer} = 3$, see Section 4.2.2. These settings were found to be optimal for the reference method, without intensity correction, and the given evaluation criterion for usability defined below. Evaluation is started after 100 frames, allowing the EM background model to learn on the first 100 frames.

The impact of the intensity correction for our application is expressed in a usability measure: the classification performance. It shows wether using intensity correction gives an improvement in the results. We define the usability of the intensity correction as the fraction of erroneously classified pixels

$$e_{usability} = \frac{N_{erroneous}}{N_{total}} , \tag{3.17}$$

with $N_{erroneous}$ the number of erroneously classified pixels and N_{total} the total number of pixels. This assumes equal cost for missed foreground pixels and erroneously detected foreground pixels.

Simulation results

Besides the six proposed methods, correcting with the ground truth *a* (GT) and using the uncorrected image (No) are also presented. The different algorithms for estimating the apparent gain factor are evaluated with and without outlier removal. For all results, the mean and standard deviation over the images were calculated for the two sections: static and changing intensity. The results shown are averaged over fourteen independent noise realizations.

The results for all combinations are given in Tables 3.1 and 3.2. Most important observations are a ten times lower accuracy error and a three times lower usability error for methods using outlier correction compared to no correction. Methods without outlier correction perform significantly worse.

From the accuracy results we see that as long as there is some kind of outlier removal (statistical or methods based on the median), accuracy is for most combinations close to the image noise. Exception is the method QofM, which performs slightly worse. For changing intensity, the error with correction is ten times lower than without correction. Without outlier removal results are significantly worse, even on the section with constant intensity. Differences between methods with outlier removal are very small. Though the small standard deviation is possible given the average was calculated from over $2 \cdot 10^8$ samples, the use of different images could cause somewhat larger variation.

For usability the results are even more consistent. All combinations that do some kind of outlier removal (statistical or median) give errors close to the error of correcting with the ground truth. The error of 6.8% is caused by the fraction of the

20% foreground pixels that overlap in color with the scene. This should be compared to no correction, where the error triples in dynamic situations. Combinations without outlier removal perform again worse than those with.

As could be expected the WLS method is only optimal when its assumptions are fulfilled, in particular with respect to the probability density distributions, so without outliers. The method is extremely sensitive to outliers, shown by the results without outlier removal. Even the few outliers remaining after outlier removal are sufficient to decrease the performance of the WLS method. To a lesser extend, the same holds for LS. A solution would be to use robust iterative estimators, like M–estimators, but those are very time-consuming and difficult, if not impossible, to use in real-time. The difference in performance between LS and WLS suggests that an additive noise model (LS) gives in this case a better description of the data than the multiplicative noise model used in WLS.

3.5.2 Evaluation using real image sequences

In simulation experiments optimal parameter settings for the background classification algorithm can be used. In practical situations these optimal settings are often unknown, and the system will operate in general at sub-optimal settings of these parameters. So the sensitivity of the performance to suboptimal settings will play a major role in the evaluation on real image sequences.

Table 3.1: *Accuracy results for simulated data. Shown is the average of the error $e_{accuracy}$ in percentages (lower is better). The standard deviations over the frames are given between brackets.*

	Static intensity		Changing intensity	
Outl. rem.:	No	Yes	No	Yes
Method:				
No	1.261 (0.009)		13 (7)	
GT	1.261 (0.009)		1.266 (0.008)	
QofA	2.64 (0.09)	1.263 (0.009)	2.63 (0.08)	1.268 (0.008)
QofM	2.4 (0.2)	1.29 (0.04)	2.3 (0.2)	1.31 (0.05)
AofQ	2.36 (0.08)	1.262 (0.009)	2.36 (0.07)	1.267 (0.008)
MofQ	1.26 (0.01)	1.261 (0.009)	1.277 (0.008)	1.266 (0.008)
LS	2.9 (0.1)	1.264 (0.009)	2.87 (0.08)	1.269 (0.008)
WLS	7.5 (0.4)	1.271 (0.009)	7.2 (0.4)	1.276 (0.009)

Image sequences

Three image sequences were used for the evaluation, see Appendix B for a more detailed description of the data:

- **Intratuin**: In this sequence there is no significant variation in intensity.
- **Schiphol**: There are many global intensity variations due to automatic gain control. The sequence contains relatively large objects, some of which become stationary.
- **PETS01-3TR1**: There are large local changes in illumination intensity due to clouds. The images contain relatively few object pixels.

All sequences are RGB color video data with eight bit per color. For each sequence, five to eleven images were manually labeled. Each pixel was labeled: foreground, background or any, see Figure 3.3 and Section B.2.1 for some examples.

ROC performance

We evaluate the intensity correction algorithms based on their usability in conjunction with classification between foreground and background. The images are corrected with each of the proposed intensity correction algorithms after which the background model is updated and foreground/background classification is performed.

Table 3.2: *Usability results for simulated data. Shown is the average of the error $e_{usability}$ in percentages (lower is better). The standard deviations over the frames are given between brackets. Note that these results cannot be compared to the results on real data as only one color has been used in the simulation.*

	Static intensity		Changing intensity	
Outl. rem.:	No	Yes	No	Yes
Method:				
No	6.8 (0.7)		18 (5)	
GT	6.8 (0.7)		6.7 (0.7)	
QofA	8 (1)	6.8 (0.7)	6.9 (0.7)	6.7 (0.7)
QofM	8 (1)	6.9 (0.7)	6.9 (0.7)	6.8 (0.7)
AofQ	8 (1)	6.8 (0.7)	6.8 (0.7)	6.7 (0.7)
MofQ	6.8 (0.7)	6.8 (0.7)	6.7 (0.7)	6.7 (0.7)
LS	8 (1)	6.8 (0.7)	6.9 (0.7)	6.7 (0.7)
WLS	18 (6)	6.8 (0.7)	8 (1)	6.7 (0.7)

(a) Intratuin (b) Schiphol (c) PETS01-3TR1

Figure 3.3: *An images from the three sequence used. More images are given in Appendix B.*

The choice of the best algorithm for foreground/background classification depends on the application. In order to make a good choice, the ratio of the cost of a false alarm and the cost of a missed detection must be known. For the experiments on simulated images described in §3.5.1, unity cost was assumed.

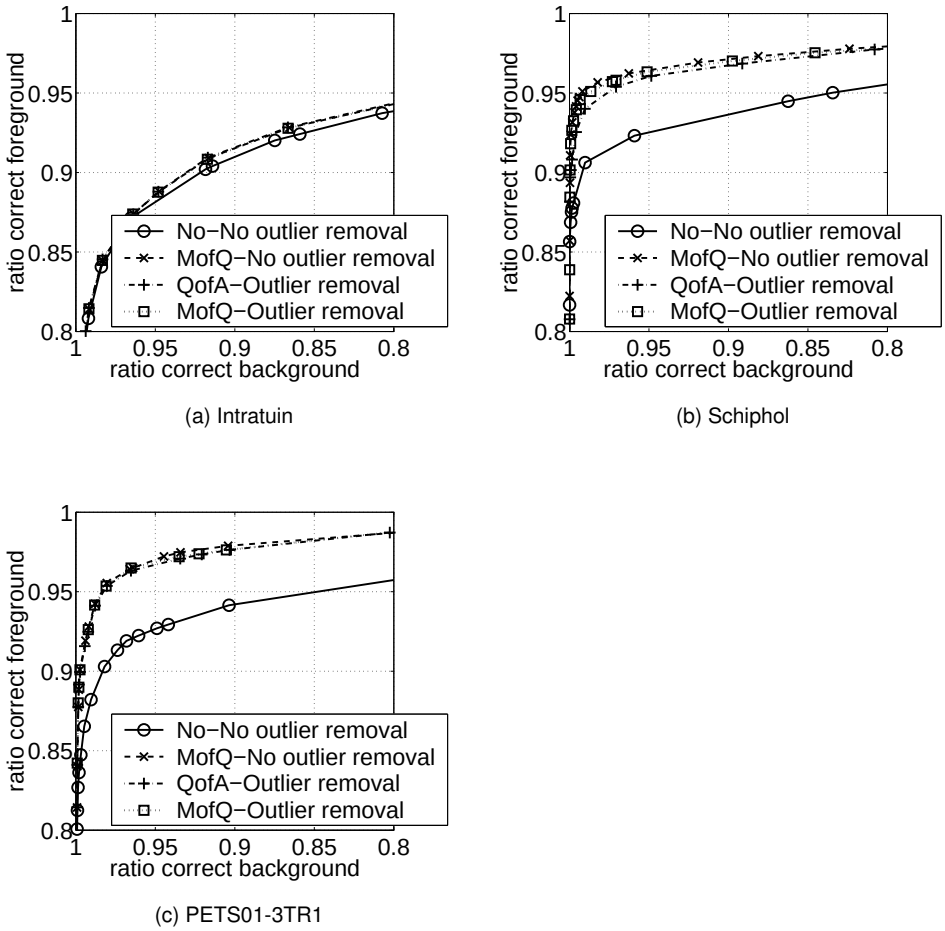
When comparing different classification algorithms, the parameter settings of all algorithms should be optimized for the chosen cost ratio. For a different application, a different ratio might be in use and therefore a different algorithm might be optimal. It would be efficient to compare for different cost rations at once.

Often, Receiver Operator Characteristics (ROCs) are used for this. We construct a ROC curve by computing the convex hull of the foreground/background classification results for a selection of parameter settings, see [Provost and Fawcett, 2001; Scott et al., 1998]. Given a certain cost fraction, the optimal method can now easily be found in the graph. We call this curve the total–ROC.

Before foreground/background classification, the EM model was initialized by updating the model for images 500, 499, and so on until image 1 using a constant update speed $u_B = 0.05$. As the first frame on which we evaluate is frame 500, this allows the algorithm to initialize during 1000 frames. This enables the evaluation of very low update speeds, for which the sequences would not be long enough to obtain a converged model of the background. The entire image sequence was processed several times. Each time using different values for the update speed u_B and threshold T_{Stauffer}.

In Figure 3.4 total–ROC curves for a number of methods are given. This are the convex hulls of the results of experiments in which both the update speed u_B and the classification threshold T_{Stauffer} of the background modelling and classification algorithm are varied. As most methods coincide, we selected only a few methods: the reference method No correction, MofQ with and without outlier removal and QofA with outlier removal. For the PETS01-3TR1 and Schiphol sequences the improvement with any of the proposed methods is significant, for the Intratuin sequence there is also a slight improvement.

Table 3.3 gives an overview of the surface above the ROC for all methods. For



(a) Intratuin

(b) Schiphol

(c) PETS01-3TR1

Figure 3.4: *Total–ROCs of the classification results on real data with different values of update speed and threshold. Only a selection of the methods is shown as most methods coincide.*

Intratuin, most methods that do some kind of outlier removal, i.e. statistical outlier removal or using the median, perform slightly better than without global intensity correction. Only exception is QofM without outlier removal. For Schiphol and PETS01-3TR1 the methods with outlier removal perform significantly better with an error reduction of a factor two and three respectively. Best performance is obtained using MofQ without statistical outlier removal.

Parameter Invariant Performance Evaluation (PIPE)

Unfortunately, the total–ROC is in this case not a sufficient criterion. The problem is that the effect of global intensity differences can be partially solved by faster updating of the background, at the cost of robustness. A method might perform well on one image sequence with a certain update speed, but this does not mean it will perform that well with equal update speed on another set of images, recorded under different conditions. This is a matter of robustness against setting the parameters, specifically the update speed, that is not taken into account by comparing total–ROCs.

We therefore propose the Parameter Invariant Performance Evaluation (PIPE) criterion. PIPE is a measure that gives in one number the average performance on an image sequence and the robustness against changing the parameters to the optimal parameter settings for other image sequences. A low error can be achieved by a method that performs well with all parameter settings, or one that has optimal performance for one setting regardless of the image sequence. Both cases are attractive to use in practice, where it is difficult and impractical to tune parameters as the circumstances change.

PIPE is based on ROCs. The parameters we intend to vary are the threshold T_{Stauffer} and the update speed u_{B}, where the update speed is the parameter we wish to be robust against. We first analyze the effect of this parameter. Therefore, we create for each image sequence I_{seq} and update speed u_{B} a u_{B}–ROC by varying threshold T_{Stauffer} only. We calculate area $A(I_{\text{seq}}, u_{\text{B}})$ under this u_{B}–ROC. This area lies between zero and one, where one corresponds to the perfect classification result and a value of 0.5 can be obtained by classifying at random.

Figure 3.5 shows the areas $A(I_{\text{seq}}, u_{\text{B}})$ under the u_{B}–ROC for different update

Table 3.3: *ROC results for real image sequences. Given is the average of the surface above the total–ROC in percentages (lower is better).*

Data:	Intratuin		Schiphol		PETS01-3TR1		Average	
Outlier removal:	No	Yes.	No	Yes	No	Yes	No	Yes
Method:								
No	3.95		2.59		2.28		2.94	
QofA	4.19	3.67	2.71	1.38	0.80	0.81	2.57	1.95
QofM	5.38	3.71	1.35	1.34	0.82	0.80	2.51	1.95
AofQ	4.21	3.68	2.03	1.35	0.82	0.79	2.35	1.94
MofQ	3.68	3.68	1.25	1.32	0.76	0.78	1.90	1.93
LS	4.44	3.69	3.01	1.42	0.81	0.80	2.75	1.97
WLS	7.67	3.68	6.08	1.60	1.16	0.80	4.97	2.03

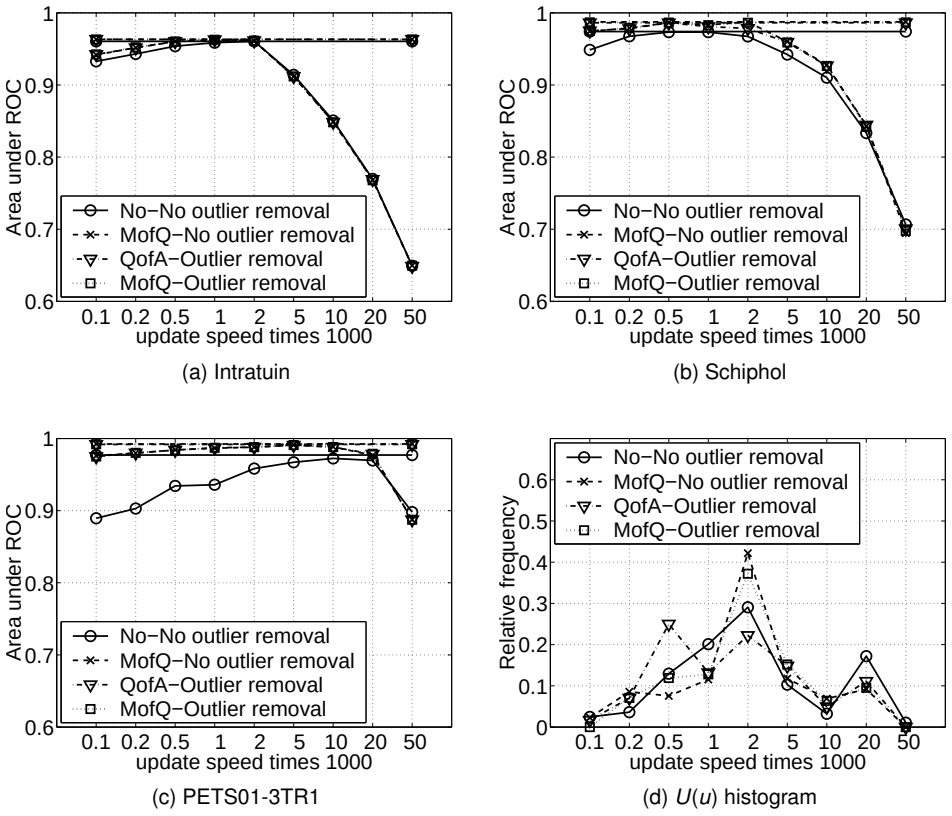


Figure 3.5: *The area under the u_B–ROC for different values of the update speed. The horizontal lines indicates the area under the total–ROC. Only a selection of the methods is shown as most methods coincide. Figure (d) shows the corresponding $U(u_B)$ histograms.*

speeds for the methods selected before. This immediately shows that different image sequences can require different parameter settings, but that intensity correction makes the algorithm more robust against setting of the update speed. As a consequence, intensity correction makes the algorithm more robust against changes of the environment. The figure also shows that intensity correction cannot be substituted by faster updating of the background model. For the PETS01-3TR1 sequence this might be a solution, but for the Intratuin and Schiphol sequences performance significantly drops for higher update speeds due to misclassification of slowly moving objects.

We wish to obtain one error measure for each image sequence. This can be achieved by averaging the results of the u_B–ROC over the different values of the

update speed. In order to introduce robustness into the evaluation criterium, we weight the different contributions. The weight is determined by the number of times this update speed was optimal for any of the image sequences. A value of the update speed is optimal when it lies on the convex hull of the total–ROC. $U(u_B)$ is the average over all image sequences of the normalized histograms of the occurrence frequency of update speeds on the convex hulls of the total–ROCs. See Figure 3.5(d) for some of these histograms.

Our Parameter Invariant Performance Evaluation error e_{PIPE} is now given by one minus this weighted average

$$e_{PIPE}(I) = 1 - \frac{1}{N_{u_B}} \sum_{u \epsilon n_{u_B}} U(u_B) A(I, u_B) \ , \tag{3.18}$$

with n_{u_B} the list of N_{u_B} different values of u_B that are used.

PIPE results

Full results according to e_{PIPE} described above are given in Table 3.4. Error reductions of almost a factor two on the Schiphol sequence and almost a factor four on the PETS01-3TR1 sequence are obtained. Below we will discuss the results for all sequences in more detail.

For the Intratuin sequence, results of combinations that either do statistical outlier removal or are based on the median are slightly better then the result without correction, except for QofM without statistical outlier removal. It is not surprising that the results are close to that without correction as this image sequence does not contain significant intensity variations. However, this data does show that the use of intensity correction does not make results worse when there is no need for correction. Methods that perform best are AofQ and QofA with statistical outlier removal and MofQ.

The Schiphol sequence contains considerable changes in intensity due to automatic gain control. The error is reduced by approximately a factor one-and-a-half for methods that do some kind of outlier removal. Without outlier removal results are significantly worse. Best performance is obtained using MofQ.

The PETS01-3TR1 sequence contains changes in intensity caused by moving clouds. Therefore, these changes are not always global. Nevertheless, results improve significantly when using any of the the proposed global intensity correction algorithms. An error reduction of almost a factor four is obtained, the methods based on the median perform best on this image sequence. Outlier removal does not seem necessary for the LS method on this image sequence. This is because compared to the other data, there are much less outliers in this image sequence as only a small fraction of the pixels depict foreground. In this case outlier removal still reduces the number of pixels and consequently the estimation accuracy of the apparent gain factor.

On average, outlier removal is essential for good results. MofQ performs best with an average error reduction of more than a factor 1.6.

3.5.3 Runtime

Our goal is to find an intensity correction algorithm to be used as pre-processing for real-time applications. Therefore, runtime is an important issue.

We ran all algorithms on an image of 100,000 pixels. The runtime of each algorithm was averaged over 100 runs. The computations were performed on a Pentium IV 2500 MHz processor under Windows 2000 Professional. The algorithms were implemented in Matlab 6.1 (www.mathworks.com). The Matlab implementations for the computation of the mean and median were used and all computations were performed in double precision.

The results are given in Table 3.5. This table shows that when no statistical outlier removal is necessary, QofA and AofQ are most affordable, followed by LS. These methods are a good choice in conjunction with other ways of outlier removal like foreground/background classification.

Otherwise statistical outlier removal should be used, except for methods based on the median as we have shown. We therefore compare the methods with a median without outlier removal to the other methods with statistical outlier removal. Then MofQ is the most affordable algorithm, followed by AofQ and QofA, all very close to each other.

It should be noted that it is not necessary to use all pixels in the image for intensity estimation. According to the amount of processing power available and the accuracy requested, a fraction of the available pixels can be used. Outlier removal also reduces the number of pixels that needs to be taken into account.

Table 3.4: *PIPE results for real image sequences. Given is the average of the error e_{PIPE} in percentages (lower is better).*

Data:	Intratuin		Schiphol		PETS01-3TR1		Average	
Outlier removal:	No	Yes.	No	Yes	No	Yes	No	Yes
Method:								
No	8.66		6.15		5.04		6.61	
QofA	7.92	7.39	5.13	3.97	1.53	1.46	4.86	4.27
QofM	8.96	7.54	3.83	3.90	1.39	1.41	4.73	4.28
AofQ	7.94	7.39	4.92	3.83	1.48	1.47	4.78	4.23
MofQ	7.14	7.23	3.47	3.63	1.41	1.35	4.01	4.07
LS	8.50	7.67	5.38	4.16	1.45	1.45	5.11	4.43
WLS	12.32	7.90	10.17	4.44	1.76	1.49	8.08	4.61

Table 3.5: *The runtime of the different methods. Times are in milliseconds, standard deviation is lower than 1.0 ms for all measurements.*

method:	WLS	LS	QofA	QofM	AofQ	MofQ
No outlier removoval	36.1	10.2	2.5	52.0	6.1	30.6
Statistical outlier removoval	73.1	46.9	39.4	89.1	38.3	62.7

3.5.4 Discussion of the experiments

According to simulation results, many algorithms seem to perform very good. Results on real images shall therefore be used to select between algorithms. We will look at the average ROC and PIPE performance over all image sequences. We then see that MofQ performs best. This is also the least computational complex algorithm. AofQ and QofA with statistical outlier removal are slightly more expensive, and their performance is also lower.

The use of WLS or LS is expensive and these methods are sensitive to remaining outliers. They are therefore not a good choice if perfect outlier removal cannot be guaranteed. Even though their average performance is good.

Both AofQ or MofQ use a per-pixel estimate of the gain factor. This allows for easy extension to local intensity correction. On the other hand, QofA and QofM have the advantage that they can also be used when the images cannot be compared pixel-wise, like in stereo vision, or when there is a small movement of the camera between the current image and the reference image.

In practice, the application requiring intensity correction should be considered before choosing a method. Considerations are the required precision, the available computing power, whether or not the camera is static and whether or not there is an outlier removal algorithm available. If images can be compared pixel-wise, MofQ is the best choice.

3.6 Conclusions

3.6.1 Comparison to previous work

Using simulation we have shown that the proposed intensity correction algorithms perform equally well as can be obtained using ground truth correction. Thus, under the assumptions of constant gamma and only global changes in intensity, there is no need to use more expensive algorithms like dynamic histogram warping [Cox et al., 1995], iterative weighted least squares estimation of the bias, gain and gamma [Tsin et al., 2001] and estimation of the gain and bias together with optic flow estimation [Altunbasak et al., 2003]. Also, [Cox et al., 1995] and [Altunbasak et al., 2003] do not perform outlier removal. We have shown in this chapter that not using outlier removal significantly reduces accuracy and usability.

Our experiments on real data clearly show that only relying on the adaptation of the background modelling technique [McKenna et al., 2000; Toyama et al., 1999] is not an option. Tweaking the update speed to reach acceptable classification performance on one sequence immediately degrades the performance on other image sequences.

[He et al., 2003] and [Kamikura et al., 1998] estimate the intensity change between two images using the average and a least squares estimate respectively. However, these methods do not consider the effect of outliers, reducing accuracy and usability.

3.6.2 Experimental evaluation

For the simulated images the RMS error between the corrected image and the ground truth image evaluates the accuracy. Results show that methods that use some kind of outlier removal (either statistical or using the median) performed very close to the noise level for image sequences without and with changes in intensity. This indicates that these methods are sufficiently accurate to be used in practical applications.

Usability of the different methods is demonstrated using background classification on the corrected image sequences. With constant image intensity, all methods with outlier removal obtained equal performance compared to using no correction. For image sequences with changing intensity these methods perform a factor ten better compared to no correction. This is similar to correcting with ground truth values.

ROC and PETS results on real images without temporal intensity changes show that most methods performing outlier removal have slightly better classification performance as without intensity correction. Two image sequences with intensity changes show that the proposed methods cause significant improvement to the classification performance. Even with local intensity changes, significant improvements are shown. Error reduction of almost a factor two on the Schiphol sequence and more than a factor three on the PETS01-3TR1 sequence are obtained.

Using the proposed Parameter Invariant Performance Evaluation PIPE we have further shown that our proposed intensity correction introduces robustness against varying the update speed of the adaptive background modelling algorithm. It allows lower update speed to be used, preventing slowly moving objects to be incorporated in the background model.

3.6.3 General conclusions

In this chapter is shown that using global intensity correction based on the ratio of pixels in conjunction with outlier removal significantly improves background classification results. For image sequences not needing intensity correction, no decrease of performance is seen.

For the evaluation on real images the Parameter Invariant Performance Evaluation (PIPE) error measure, based on the ROC, is proposed. It combines both classification error and robustness against changes in parameter settings in one number.

Chapter 4 Dynamic background estimation

4.1 Introduction

This chapter[1] introduces a novel background updating algorithm for obtaining background probabilities in the lowest level of abstraction, the pixel level. It was discussed in Chapter 2 that for surveillance applications, object detection and tracking can best be performed using an adaptive background model. The Mixture of Gaussians (MoG) model is most popular. Stauffer and Grimson [Stauffer and Grimson, 2000] were one of the first to use a MoG for background modelling. Many authors adopted their basic algorithm, although improvements were made in the update and classification process, see for example [Pavlidis et al., 2001].

The background model proposed by Stauffer updates all pixels each frame, regardless of wether the pixel depicts foreground or background. Classification is performed by labelling the kernels in the MoG as either background or foreground. This way of background modelling performs well for a standard case, but fails in cases like complex backgrounds and many or slow objects, because in that case moving objects will be learned into the background model causing classification errors, as will be discussed in §4.2.2. In this chapter EMswitch is introduced, an algorithm that updates the background model only when a pixel is classified as background. To make it robust for errors a multiple hypothesis approach is used. The EMswitch algorithm produces background probabilities which are input for the probabilistic tracking framework introduced in Section 1.3.

Our approach differs from most others using a MoG in that we will use separate models for the background and foreground. Therefore, our background model only models the background. Most other approaches update the model every frame for every pixel, resulting in a mixed model of both background and foreground. The use of such shared model has two important drawbacks. First, there is the need

[1] This chapter is based on [Withagen et al., 2003].

to classify kernels in the MoG between foreground and background. The second drawback is the way the classification result changes for changed background, for example when rain gives the road a different appearance, as well as for foreground objects that stopped moving. At pixel level this is a sudden change, one frame the pixel is classified as foreground, the next as background. At blob level however, each pixel will change from foreground to background at another instance in time.

EMswitch separates the background and foreground modelling. The model of the background is updated only for those pixels depicting background, while a different model is used for the foreground, see Chapter 5. Therefore, it circumvents the drawbacks mentioned above. All kernels in the model are used to model background only and objects that become static, or background that changes, do not automatically become part of the background. Additional advantage is that EMswitch is more robust for setting the update speed parameter.

To be able to obtain a model of the background only, two conditions need to be satisfied. First, pixels that are erroneously classified as foreground, for example due to a large noise realization in a certain frame, should still be updated. If this is not done, as is the case when only background pixels would be updated, the kernel variances will be underestimated, resulting in more and more background pixels being misclassified. Second, moving objects that become static or background that changes should at some time be updated in the background model, to prevent an unstable algorithm that will eventually classify the entire image as foreground.

The outline of this chapter is as follows: In Section 4.2 the EM algorithm and its use for background modelling is shortly discussed. In Section 4.3 the EMswitch update algorithm will be introduced. This is an affordable solution for deciding when to update the background model. It is inspired by a multiple hypothesis approach. In Section 4.4 results from the EMswitch update algorithm will be compared to the EM algorithm with classification according to [Stauffer and Grimson, 2000]. Finally, conclusions will be given in Section 4.5.

4.2 Background modelling

A MoG is used to model the the Probability Density Function (PDF) of the background. The PDF models the occurrence frequency of colors over time for each pixel using a number of Gaussian kernels. We will give the equations used to describe and update the PDF. Then the Stauffer approach to pixel classification will be given, followed by a short explanation of the amount of kernels used to describe the background. After that, the problems encountered when updating a model of only the background will be described.

4.2.1 The mixture of Gaussians model

The PDF of a MoG of data $\vec{x}$ is given by:

$$P(\vec{x}|\beta_0) = \sum_{k=1}^{N_k} \pi_k f(\vec{x}|\vec{\mu}_k, V_k) \ , \tag{4.1}$$

with β_0 the background class, N_k the number of kernels in the MoG and π_k the prior probability for the k^{th} kernel, with

$$\sum_{k=1}^{N_k} \pi_k = 1, \qquad \pi_k \geq 0 \ . \tag{4.2}$$

$f(\vec{x}|\vec{\mu}_k, V_k)$ is the value of a Gaussian distribution for the pixels feature vector $\vec{x}$ of length N_c. The kernel is described by its mean vector $\vec{\mu}_k$ and covariance matrix V_k:

$$f(\vec{x}|\vec{\mu}_k, V_k) = \frac{1}{\sqrt{(2\pi)^{N_c}|V_k|}} \exp\Big(-\frac{1}{2}(\vec{x}-\vec{\mu}_k)^{\mathsf{T}} V_k^{-1}(\vec{x}-\vec{\mu}_k)\Big) \ . \tag{4.3}$$

Color space and kernel shape

The pixel feature used in this thesis is the pixel color. The following effect cause changes in the pixel value:

- camera noise
- intensity changes
- variable scene such as water surface and tree leaves
- camera vibrations

The most robust model is that model that has the least amount of parameters while being able to describe all effects given above.

The camera noise of a CCD camera at a given intensity can be approximated using a Gaussian distribution [Healey and Kondepudy, 1994], which is best modelled in RGB color space. Intensity changes are better represented in a color space that separates intensity, such as Irg. As we use a correction algorithm for global intensities, the latter is less important. So we will use RGB color space, $N_c = 3$.

The number of parameters in the Gaussian distribution depends on the covariance matrix used. Six parameters describe a full covariance matrix, three parameters limits the orientation of the Gaussian kernels to the three main axes R, G and B. A diagonal covariance matrix with equal variances requires only one parameter, resulting in spherical Gaussian kernels.

The camera noise can be modelled using a zero or one parameter model, given that the noise characteristics of the camera are known. If no prior noise model is

available, three parameters are necessary, as the camera noise will be different in each of the color bands because of its multiplicative nature.

For constant scene, small scene variations can be modelled using three parameters are sufficient. A model with six parameters can also be used, but is not necessary.

For changing scene such as leaves or water surface the effective distribution will have multiple modes. Then the a single Gaussian distribution does not give an accurate description of the probability density distribution. For this scene, multiple kernels are available, describing the probability density distribution together. The three parameters can be estimated more efficient and more robust than the six parameter in the full covariance matrix.

At color edges, camera vibrations can results in distributions along an arbitrary axes. This is best modelled by a full covariance matrix. However, only a small fraction of the pixels will be on color edges. For these pixels, two or more kernels with a diagonal covariance matrix will also be sufficient. For all other pixels, less parameters leads to a more robust estimate. Therefore, we will use the diagonal covariance matrix given by:

$$V_k = \begin{pmatrix} \sigma_{k,R}^2 & 0 & 0 \\ 0 & \sigma_{k,G}^2 & 0 \\ 0 & 0 & \sigma_{k,B}^2 \end{pmatrix} . \tag{4.4}$$

Update equations

The PDF in Equation 4.1 is used to model the histogram in time for each separate pixel. A separate MoG has to be estimated for each (spatial) pixel. It is updated for each new frame. The online EM equations [Priebe, 1994] are used for updating the parameters for each kernel k and each color c:

$$\pi_k := \pi_k + u\big(P(k|\vec{x}) - \pi_k\big) \qquad \text{“Prior”} \tag{4.5}$$

$$\mu_{k,c} := \mu_{k,c} + \frac{u}{\pi_k} P(k|\vec{x})(x_c - \mu_{k,c}) \qquad \text{“Mean”} \tag{4.6}$$

$$\sigma_{k,c}^2 := \sigma_{k,c}^2 + \frac{u}{\pi_k} P(k|\vec{x}_i)\big((x_c - \mu_{k,c})^2 - \sigma_{k,c}^2\big) , \qquad \text{“Variance”} \tag{4.7}$$

with

$$P(k|\vec{x}_i) = \frac{\pi_k f(\vec{x}_i|\vec{\mu}_k, V_k)}{P(\vec{x}_i|\beta_0)} . \qquad \text{“Posterior”} \tag{4.8}$$

u_B represents the update speed of the algorithm. See Appendix C for a detailed explanation.

4.2.2 Stauffer classification

The pixels of each frame are classified between foreground and background by comparing the pixel color to the model. Contrary to the EMswitch approach proposed in this thesis, usually only one model is available. This model is updated each frame, also when the pixel in a frame depicts foreground. Therefore, the model will be a mixed model of foreground and background.

A frequently used classification approach is that of Stauffer [Stauffer and Grimson, 2000]. They assume that the background and foreground colors will be modelled by different kernels. Therefore, the kernels need to be classified before a pixel can be classified. They argue that the frequency of occurrence will be higher for the background, and that kernels modelling foreground color will have a higher variance. Hence, background kernels have a large prior and a small variance.

First the kernels are ordered in descending order by their value of π_k/σ_k, with π_k the prior probability and σ_k the standard deviation of kernel k. The first b kernels are then chosen as the background model, where

$$B = \arg\min_b \left(\sum_{k=1}^{b} \pi_k > T_{\text{Priorsum}} \right) , \tag{4.9}$$

with T_{Priorsum} a constant threshold. Classification is done by checking whether for color $\vec{x}$

$$| \vec{x} - \vec{\mu}_k | < T_{\text{Stauffer}} | V_k | , \tag{4.10}$$

for any of the kernels labelled as background. T_{Stauffer} is a threshold.

Disadvantages of the Stauffer classification

The classification method proposed by Stauffer and Grimson performs well for static backgrounds. Using it for non-static backgrounds causes problems deciding which kernel describes background when there are long occlusions. For example, if the background has value $\vec{b}_1$ in 60% of the time, and value $\vec{b}_2$ the remaining 40%, but is occluded for 30% of the time by one or more objects with a value $\vec{o}$, that would give the following prior probabilities: π_{b1} = 42%, π_{b2} = 28% and π_o = 30%. Using the prior probabilities only (this example is simplified by taking equal standard deviation for all kernels, T_{Priorsum} = 50% is used) will result in background when either value $\vec{b}_1$ or $\vec{o}$ is measured, and foreground when value $\vec{b}_2$ is measured. This is not the desired result of the algorithm. This problem could be solved by updating the background model slower, but that would lead to problems with illumination changes.

Also situations where the background and foreground color distributions are not distinct can cause problems. In these situations the foreground and background will not be modelled in separate kernels, so labelling of the kernels is senseless.

When foreground and background colors differ only slightly, the kernels describing the actual color of the foreground and background will partly overlap. In such

a case, using the approach proposed by Stauffer and Grimson will result in erroneously classifying more of the foreground pixels as background than necessary.

Another disadvantage is that all pixels are updated, even those depicting foreground. This is necessary for a correct estimate of the variance, see §4.2.4. As a consequence, objects that move slow compared to the update speed could be learned into the background model in such a way that their modelling kernel is assigned to the background.

4.2.3 The number of kernels in the MoG

Only one kernel is needed to model a static background, but backgrounds are not always static. Examples of non-static backgrounds are:

- Water
- Moving trees and vegetation
- Computer screens and TVs

A MoG models such backgrounds by using more kernels. Generally said: the more complex the background is, the more kernels are needed. On the other hand, the more kernels are used, the more data is necessary for the model to converge to an accurate description of the background. Using more kernels also requires more computations and memory.

An additional reason to add kernels is when, for example in the Stauffer approach, the MoG is used not only to describe the background but also the foreground. The model is updated for each frame, regardless whether a pixel depicts foreground or background. Using the assumption that the color of foreground objects and the background are distinct, they will be modelled in the PDF by different kernels, see §4.2.2. Classification is done by labelling kernels background or foreground. When there are not sufficient kernels to model all occurring foreground colors, kernels describing the background will be disturbed when a foreground object passes.

4.2.4 Updating the model

We will use separate models for the background and foreground. They need to be estimated from the data, and updated to allow for gradual changes. However, it is not trivial to decide which model to update. A naive approach is to update them according to the classification result, but this results in underestimating the standard deviation of the background kernels. Assuming to draw from a Gaussian distribution, background samples with values in the tail of the background distribution will be wrongly classified as foreground. As this does not happen frequently, it will not cause problems for the classification. However, it will cause problems when the background is not updated due to the classification error. The standard deviation will be underestimated and subsequently more pixels will be misclassified as foreground. This problem increases over time, see figure 4.1.

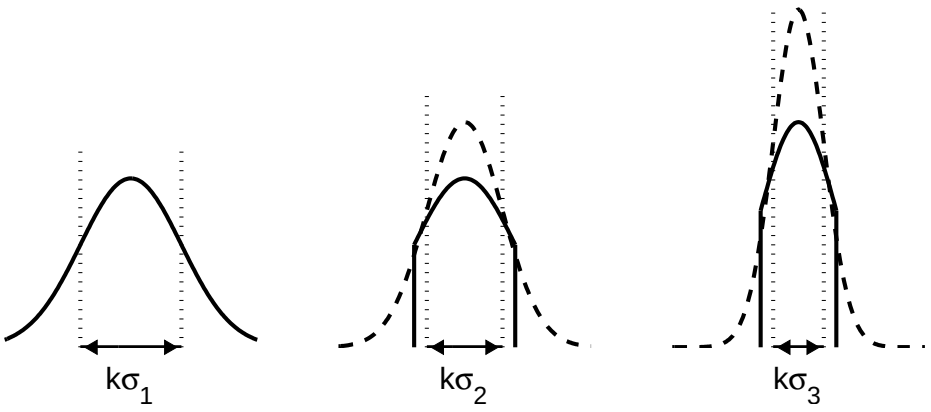


Figure 4.1: *When only updating a kernel if the new value is between an interval given by the mean and standard variation (dotted lines), the tail of the distribution is not updated, see the figure on the left. Only using this part of the distribution (drawn curve in the center figure), a new Gaussian kernel is estimated with a lower standard deviation (dashed curve). Again, only samples within the dotted lines are used to update the kernel, so in the next frame this kernel will have an even smaller standard deviation, see the figure on the right.*

A second problem is that when a pixel represents foreground, but the color is described well enough by the background model to be classified as background, the background model will be updated with its foreground colors, leading to a bad representation of the real background and subsequently bad classification results. Therefore, updates of the background model shall not be allowed when a pixel shows foreground.

Summarizing: on the one hand updates for values in the tail of the Gaussian distribution should not be missed, and on the other hand updates of the background model are not allowed when the pixel might depict foreground. To meet these contradicting demands, the update strategy for the models needs to be considered.

4.3 EMswitch, the proposed update algorithm

To decide when and when not to update a model, time, and especially the behavior of possible objects over time, will be considered. As the goal is to classify between steady background and moving foreground, the behavior of the foreground objects can help to decide when to update.

In the following section a multiple hypothesis approach is considered. It will be shown that it solves the problem, but at a huge cost. Then the EMswitch update strategy, which incorporates many of the advantages of the multiple hypothesis approach with a minimum of computations and necessary memory, is proposed.

4.3.1 Multiple hypothesis approach

The multiple hypothesis approach [Blackman and Popoli, 1999; Reid, 1979] is an approach used frequently in (radar) tracking problems. The general idea is that when there is evidence for two or more conflicting assignments, all hypothesis are propagated, expecting that future data will resolve the problem.

Applied to background modelling, this means that when we are not sure which model to assign the pixel to (background, object 1, object i), we create several versions of the models, called branches. In each of the branches we update the model according to the hypothesis. In the next frame, each of branches is evaluated, resulting in a number of different classifications for a single pixel. Most probable there will be uncertainty in the classification, leading to sub-branches. This process continues, leading to a tree of hypothesis. In Figure 4.2(a) an example of such tree is given in the case that there are only two possible classifications: background and foreground. When allowing classification between different objects, as is the case in this thesis, the tree will become more complex.

To avoid an explosion of hypothesis and the corresponding demand on memory and computational power, the multiple hypothesis tree should be pruned. This means that unlikely hypothesis are removed from the tree, see [Blackman and Popoli, 1999]. The amount of hypothesis that need to be kept depends on the problem that needs to be combatted with multiple hypothesis.

We propose to use a multiple hypothesis approach to solve the following two problems:

- When values in the tail of the distribution are not used for updates, the standard deviation will be underestimated.
- When an object stays at a certain location, it should be regarded as changed background instead of an object.

This can be obtained by inducing demands on the time that objects are visible. A foreground object should be visible in a certain pixel location during N_F frames, where

$$N_{\text{Fmin}} < N_F < N_{\text{Fmax}} , \tag{4.11}$$

with N_{Fmin} small, a few frames and N_{Fmax} large, seconds or more. These numbers depend on the expected size and speed of objects.

Objects depicted shorter than N_{Fmin} frames by a pixel will be ignored. When a pixel depicts foreground longer than N_{Fmax} frames it is assumed that it depicts changed background or a false detection, and a model of this new background should be available. Between each object (or group of objects) it is expected to see the background for at least N_{Bmin} frames, otherwise it is considered to be a single object and the background shall not be updated in between the sub-objects. N_{Bmin} can be small, a few frames. Typically, $N_{\text{Bmin}} = N_{\text{Fmin}}$. As N_{Fmin} and N_{Bmin} are small and N_{Fmax} is large, these assumptions are not too tight for surveillance applications.

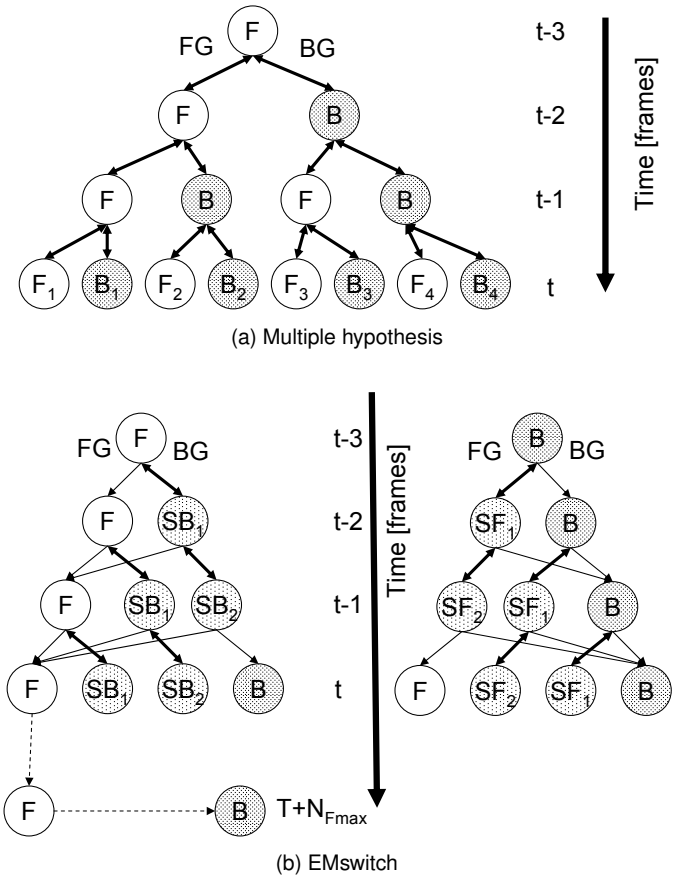


Figure 4.2: *The possible state transitions using multiple hypothesis with a memory of three frames and EMswitch for which both N_{Fmin} and N_{Bmin} are equal to two, so after three sequential classifications other than the initial state, the state is changed from* **B** *to* **F** *or vice versa. The number of frames the pixel has been in a switch state is given by the subscript, so* $\mathbf{SF}_2$ *means that the pixel has been in switch state* **SF** *for two frames. Only along bold bi-directional arrows data can be recovered. The states* $\mathbf{B}_3$, $\mathbf{F}_2$, $\mathbf{F}_3$ *and* $\mathbf{F}_4$ *from multiple hypothesis can not be obtained in EMswitch because only two background models are kept in memory simultaneously, one for the stable state* **B** *and one for the switch states* **SF** *or* **SB**.

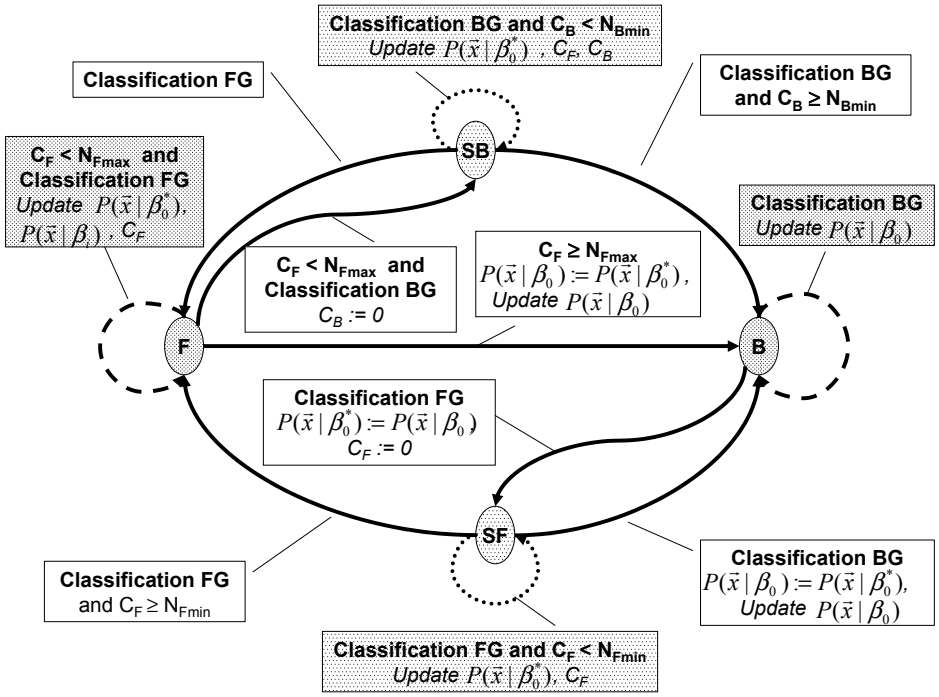


Figure 4.3: *The possible transitions in EMswitch between each state. Conditions are printed in fold face, actions in italics. β_0 depicts background, β_0^* the copy of the background and β_i depicts the i^{th} foreground blob.*

So after a number of frames N_{MH} (at least the maximum of N_{Bmin} and N_{Fmin}, but preferably as much as N_{Fmax}), the most probable branch from the multiple hypothesis tree is selected using the knowledge about the life-time of the objects given above. The other branches are removed (pruned).

This solves both problems, but at a huge cost. The complexity of this algorithm is $O(N_{clas}^{N_{MH}})$ with N_{clas} the number of possible classes. The amount of data scales accordingly.

Another problem is that the final pixel classification is only available after N_{MH} frames. This means that all higher level processing either has to wait, resulting in a delay, or has to be able to work with multiple hypothesis. For updating the models a small delay is not a big disadvantage, as slow changes in the background are expected. For object detection in a real-time application it would pose a problem.

Simplifications to multiple hypothesis

Some simplifications can be made to overcome the problems of complexity and delay.

First, multiple hypothesis will not be used for classification. This is not a big disadvantage, because misclassifications are caused by values in the tail of the Gaussian distributions, so they will not occur frequently, and if they occur it will be mostly isolated pixels, that can easily be filtered out and ignored. So the instantaneous classification result is good enough for higher level processing. This solves the delay and need to use multiple hypothesis at the higher levels.

Second, multi hypothesis is only used to update the background model. Considering the differences between the foreground and background model, it can be seen that as the foreground model is shared by all pixels in a blob, it poses less problems when it is not always updated. There are enough pixels to base the model statistics on, assuming that the spatial distribution of colors over the object is not too distinct. Therefore, this model can be updated only when it is certain that the pixel depicts foreground. When in doubt the model will not be updated.

These two simplifications reduce the complexity to $O(2^{N_{MH}})$ and does not require multiple hypothesis approaches at higher levels. However, this still demands a huge memory and computation power.

4.3.2 The EMswitch update strategy

The EMswitch update strategy is derived from the (simplified) multiple hypothesis approach described above, but uses only two simultaneous background models, $P(\vec{x}|\beta_0)$ and a copy $P(\vec{x}|\beta_0^*)$, and one foreground model $P(\vec{x}|\beta_i)$ for each foreground blob β_i with $i > 0$. For the foreground model of each blob, a single PDF of the color of all pixels belonging to the blob will be used. It will be further described in Chapter 5.

The initial reason for introducing multiple hypothesis was that the background model should always be updated to prevent underestimation of the variance. In case of doubt a copy $P(\vec{x}|\beta_0^*)$ of the current model $P(\vec{x}|\beta_0)$ will be made before updating. When at a later time it is concluded that the update was wrongly performed, the original model is restored using the copy that was made.

EMswitch uses four states: background **B**, foreground **F**, switch to background **SB** and switch to foreground **SF**. As long as the state of a pixel is **B** or **F** and the classification result, through feedback, is the same, the state remains and the corresponding model is updated. In these stable states no competing hypotheses are considered. When the classification result differs from the state of a pixel, the state of that pixel is changed to an intermediate state, from **B** to **SF** and from **F** to **SB**, indicating that there is a possible switch, see also figures 4.3 and 4.4.

When the state of a pixel was **B** and the pixel is in the current frame classified as foreground a copy of the background model is made, the foreground counter C_F is reset and the state is changed to **SF**. In this state counter C_F and the copy of

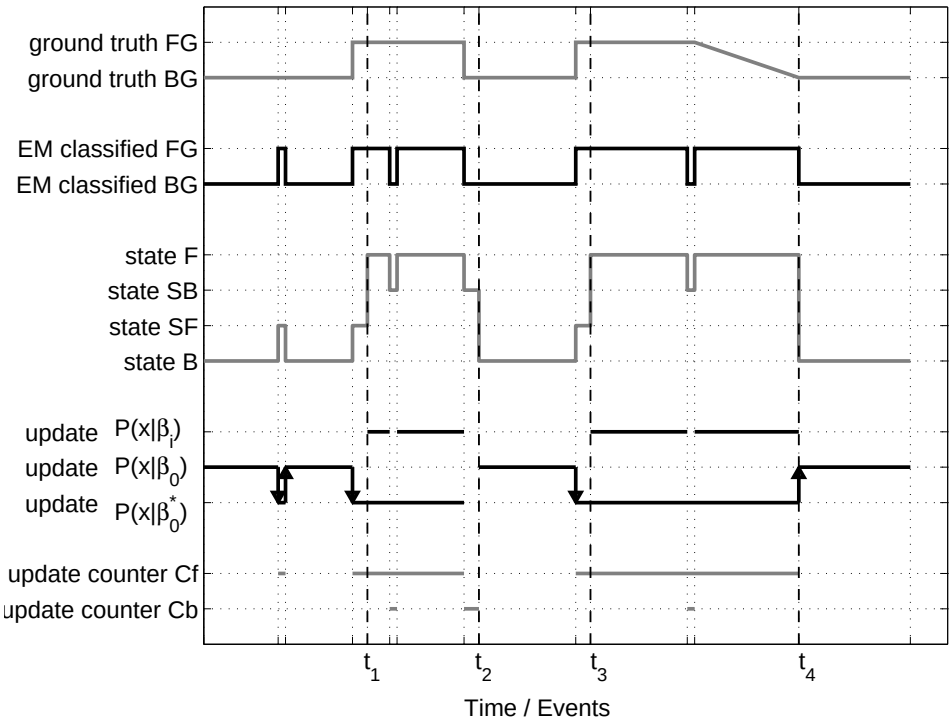


Figure 4.4: *The state transitions in EMswitch for one pixel in an example sequence. Five graphs are shown, from top to bottom: ground truth classification, instantaneous (EM) classification, current state of EMswitch, models which are updated in EMswitch, and counters which are updated in EMswitch. Arrows indicate the copy or restore of a model. The four vertical dashed lines indicate a counter reaching a threshold: for t_1 and t_3 $C_F = N_{Fmin}$, for t_2 $C_b = N_{Bmin}$ and for t_4 $C_F = N_{Fmax}$.*

the background model $P(\vec{x}|\beta_0^*)$ are updated for each frame. The state of the pixel is not changed until either the counter C_F reaches N_{Fmin}, in which case the new state becomes **F**, or when a pixel is classified as background, the state transfers back to **B** and the model of the background is replaced with the copy.

For a pixel of which the state transfers to **SB** the counter C_B is reset. In state **SB** counter C_B is updated for each frame. The state of the pixel remains **SB** until the counter C_B reaches N_{Bmin}, in which case the state transfers to **B**, or until the pixel is classified as foreground. Then, the state returns to **F**.

The rules given above solve the problem of values in the tail of a distribution. To cope with a possible change of background, counter C_F and the copy of the background model $P(\vec{x}|\beta_0^*)$ are updated in states **F** and **SB**. When the counter C_F reaches N_{Fmax} the background is considered changed, see time instance t_4 in Fig-

ure 4.4. The background model is overwritten with the updated copy and the state transfers to **B**. In this way, as soon as a change of background is assumed, the new background model is available. Transition to background can be performed per pixel, but using feedback we will perform it for the entire blob at once.

Comparison to multiple hypothesis

As can be seen in figure 4.2(b), the algorithm introduced can be seen as a partial multiple hypothesis algorithm. When the state of a pixel is **SB** for the second frame (denoted by $\mathbf{SB}_2$) the state can transfer to **F**, but it is not possible to go one step back to $\mathbf{SB}_1$. The same holds for state **SF**. In other words: each time the state changes from a switch to a stable state, all other tree branches are pruned.

A problem that remains with EMswitch occurs when a pixel depicts foreground, and while its state is still **SF**, the pixel is erroneous classified as background for one frame. The foreground color will be updated in the background model. However, this will only occur when the object closely resembles the background, in which case the effect of an erroneous update is negligible.

Using feedback from higher levels

The proposed background modelling algorithm is intended to be used in an object tracking application, together with other algorithms such as those introduced in this thesis. The framework, introduced in Section 1.3 and further described in Chapter 7 is used for the integration. A strong feature of this framework is the use of feedback from higher levels of abstraction to update lower level models. This has the advantage that higher level knowledge can be exploited for updating the model of the background.

Instead of the instantaneous classification, the pixel class from feedback will be used as input for EMswitch. So in Figure 4.3, the classification results FG and BG given by the higher levels will be used in the state diagram. Pixels assigned to the background will be treated as such, while pixels assigned to either foreground or shadow will be treated as foreground.

4.3.3 EMswitch initialization

The MoG describing only the background has to be initialized, it cannot be decided whether or not a pixel belongs to the background when there is no background model. This is solved by starting with Stauffer classification, and use this classification to initialize the background model.

A state **I** (Initialization) and a counter C_I are introduced. All pixels start with state **I**. The initialization is as follows:

- Initialize two MoG models $P(\vec{x})$ and $P(\vec{x}|\beta_0)$ by distributing the kernels evenly over the parameter space. Priors are equal, variances are equal and relatively

large.

- Use the first **frames** (until $C_l > N_{l1}$) only to update an additional model $P(\vec{x})$. This model will contain both background and foreground information.

- Until $C_l > N_{l2}$, i.e. the next number of **updates**, use $P(\vec{x})$ to classify between foreground and background using Stauffer classification. Update C_l and $P(\vec{x}|\beta_0)$ only for frames for which the classification result of the pixel is background to obtain a model of only the background and update $P(\vec{x})$ for all frames.

- To reduce the impact of foreground information that might erroneously be used to update the background model we set the highest prior probability to 0.99 and all others to $\frac{0.01}{N_k-1}$, with N_k the number of kernels in the MoG. This assumes that the background consist of only one color.

- End of initialization. Start using the combination of $P(\vec{x}|\beta_i)$ and $P(\vec{x}|\beta_0)$ for classification.

In the initialization phase the Stauffer classification is used as it can handle a mixed model of foreground and background.

4.4 Experimental evaluation

In this Section the results on some real-life image sequences will be shown. The three image sequences given in Chapter 3 will be used for evaluation, see also Appendix B. The results of two methods are compared:

- **EMStauffer + GIC**, always update the model, classify using Stauffer classification described in §4.2.2. Global Intensity Correction (GIC) is performed using MofQ with statistical outlier removal described in Chapter 3.

- **EMswitch + GIC**, update using the EMswitch algorithm, classify using probabilities described below. The same GIC algorithm is used.

Evaluation will be based on both the (total) ROC and on PIPE results, see Chapter 3. Additionally, some examples of classification results will be given.

4.4.1 Implementation details

Stauffer only updates kernels for which the new data lies within 2.5 standard deviations from the kernel mean [Stauffer and Grimson, 2000]. This leads to an underestimate of the standard deviation, see Figure 4.1 or [Pavlidis et al., 2001]. Therefore, we use the online EM update equations given in §4.2.1 for both algorithms.

The MoG models provides us with posterior probabilities $P(\vec{x}|\beta_0)$. Using Bayes, conditional probabilities $P(\beta_0|\vec{x})$ are obtained:

$$P(\beta_0|\vec{x}) = \frac{P(\beta_0)P(\vec{x}|\beta_0)}{P(\vec{x})} , \tag{4.12}$$

with $P(\beta_0)$ the prior probability for background and $P(\vec{x})$ the marginal probability of $\vec{x}$. As only a model of the background is available, $P(\vec{x})$ is generally unknown. We assume a uniform distribution, which makes $P(\vec{x})$ a constant. Choosing it $P(\vec{x}) = P(\beta_0)$ gives for Equation 4.12:

$$P(\beta_0|\vec{x}) = P(\vec{x}|\beta_0) . \tag{4.13}$$

For classification, a static threshold is used. Pixels are assigned background when

$$P(\beta_0|\vec{x}) > T_{\text{EMswitch}} . \tag{4.14}$$

In all experiments the following parameters settings were used as defaults:

- Number of kernels: $N_k = 5$.
- Threshold for EMStauffer: $T_{\text{priorsum}} = 0.5$, $T_{\text{Stauffer}} = \sqrt{5}$.
- The threshold for EMswitch classification is $T_{\text{EMswitch}} = 0.05/256^3$
- N's: $N_{l1} = 100$, $N_{l2} = 200$, $N_{\text{Fmax}} = 350$, $N_{\text{Fmin}} = 3$, $N_{\text{Bmin}} = 3$.
- Update speed of the EM algorithm $u_{\text{B}} = 0.01$ (the higher this value, the faster the model reacts on the data).

These settings were obtained by optimization on the Intratuin sequence. For the numerical results in Table 4.1 and the ROC and PIPE results in Figures 4.7 and 4.8 respectively the update speed and threshold have been varied.

4.4.2 Classification results

Results for both algorithms on the PETS01-3TR1 and Intratuin sequences are shown in figure 4.5. It can be seen that both algorithms perform comparably well on the PETS01-3TR1 sequence. On the Intratuin sequence EMswitch performs better. When using EMStauffer there are large holes in the objects caused by earlier passing cars. As the model was updated when the cars passed by, the background model was disturbed.

Figure 4.7 gives the ROC curves of the three image sequences. They show that performance is better for lower background ratios (approximately below 0.97). The performance is worse for low foreground ratios. This is assumed to be caused by shadows. As a result of the more accurate background model (lower variances), causing less (shadow) pixels to be classified as background, the EMswitch algorithm is slightly more sensitive to shadows. On average, the area above the ROC

Table 4.1: *Surface above the ROC and PIPE results for the two algorithms. All numbers are percentages where lower is better.*

Data:	Intratuin	Schiphol	PETS01-3TR1	Average
ROC EMStauffer + GIC	3.68	1.32	0.78	1.93
ROC EMswitch + GIC	2.81	0.99	0.79	1.53
PIPE EMStauffer + GIC	7.23	3.63	1.35	4.07
PIPE EMswitch + GIC	4.24	2.22	1.71	2.72

is still significantly smaller for EMswitch, see Table 4.1. Only for PETS01-3TR1 the results are practically the same. This is because this sequences has large shaded regions caused by clouds moving in front of the sun.

The EMswitch algorithm is more robust for setting the update speed than EMStauffer. This is caused by foreground objects not being updated in the background model, so their colors will not become part of this model. Only when objects are at a certain location for such a long time that it can be assumed that the background must have changed (longer than N_{Fmax} frames), the background model is renewed.

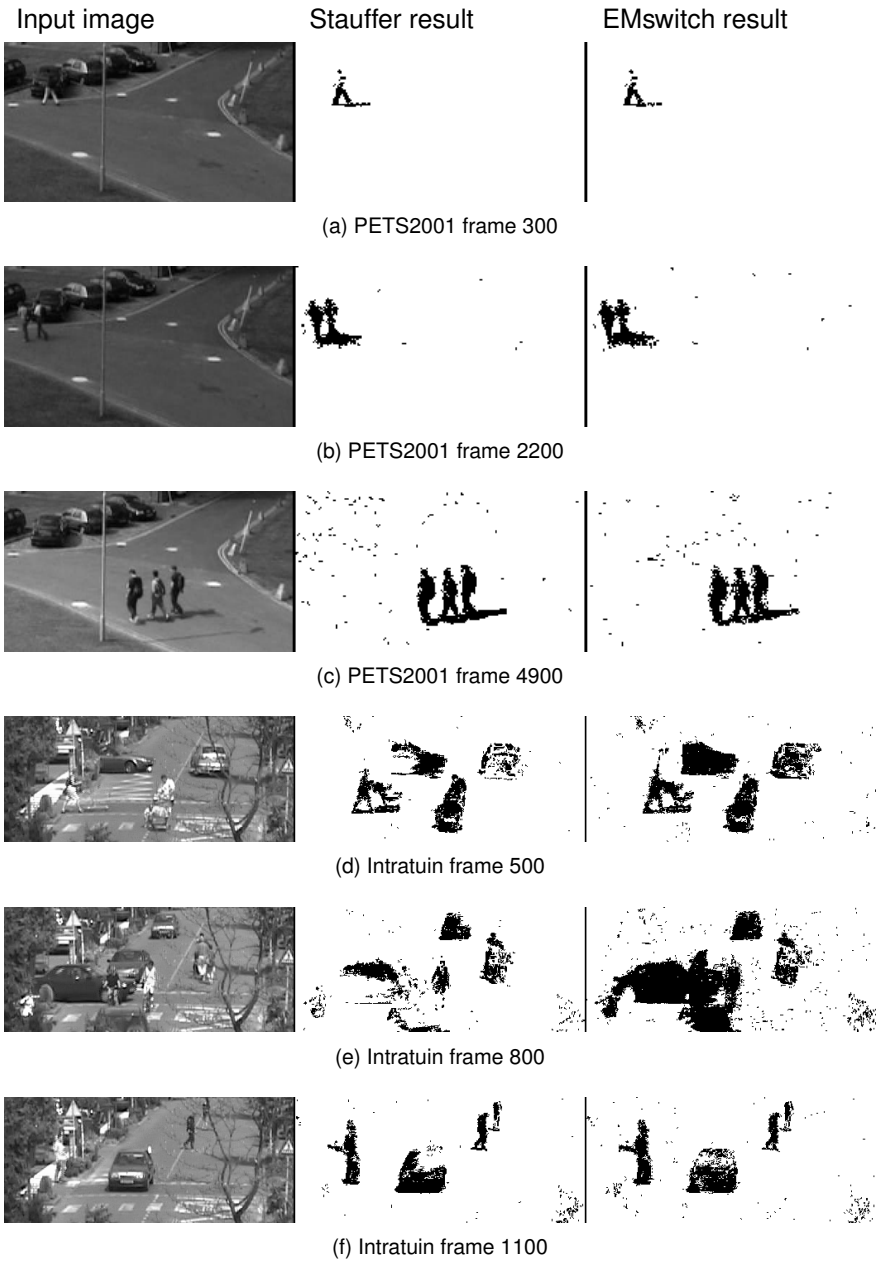
Because the background model is not updated for foreground objects, the update speed can be chosen to be relatively high, which is an advantage when there are fast illumination changes or slow objects. In figure 4.6 the results are shown of processing the Intratuin sequence with a ten times higher update speed. As can be seen, most objects are lost when using EMStauffer.

The robustness to setting the update speed is evaluated using the Parameter Independent Performance Evaluation (PIPE) described in §3.5.2. Results are given in Table 4.1 and plots of the results per update speed in Figure 4.8. Again, results improve significantly except for the PETS-1=3TR1 sequence. This is again assumed to be caused by shadows.

4.5 Conclusions

In this chapter a novel method has been introduced for updating a Mixture of Gaussians (MoG) background model using Expectation Maximization (EM). It was argued that for obtaining separate models for background and foreground, some sort of reasoning is necessary to update the background model, because the standard deviation is otherwise not estimated correctly. The proposed algorithm, EMswitch, is an update strategy based on multiple hypothesis using life-time knowledge about the expected objects.

Because of the model of only the background, foreground/background classification can be based on probabilities. This allows the use of the proposed algorithm as plug-in in the probabilistic tracking framework.



(a) PETS2001 frame 300

(b) PETS2001 frame 2200

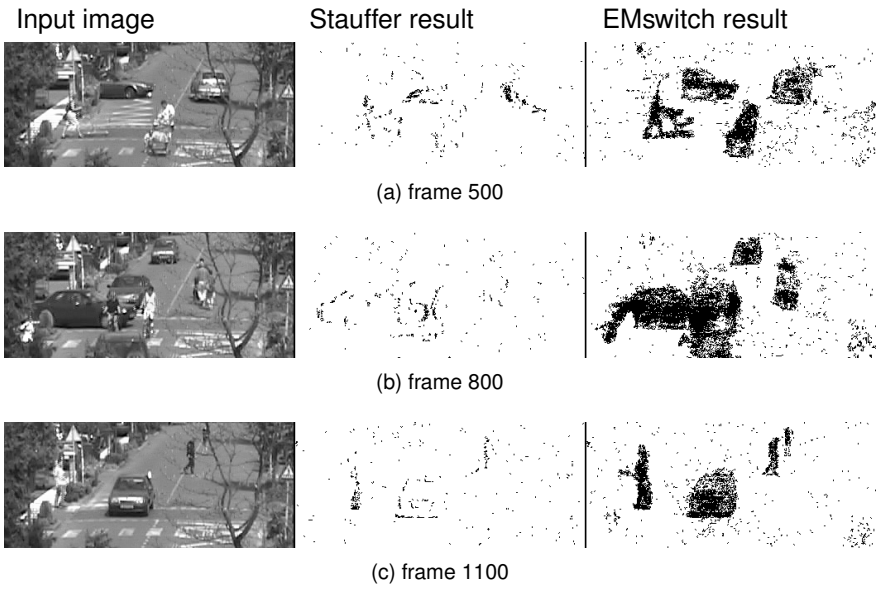
(c) PETS2001 frame 4900

(d) Intratuin frame 500

(e) Intratuin frame 800

(f) Intratuin frame 1100

Figure 4.5: *Classification results. For each frame the input frame, the results of EMStaufer and EMswitch are shown respectively. White pixels depict background, black pixels foreground.*



(a) frame 500

(b) frame 800

(c) frame 1100

Figure 4.6: *Results on the Intratuin sequence with a ten times higher update speed.*

ROC and PIPE results show that the EMswitch algorithm performs better and is less sensitive to update speed selection than the EMStauffer reference method. EMswitch does have some more problems with shadows, causing it to perform worse on one on the image sequences.

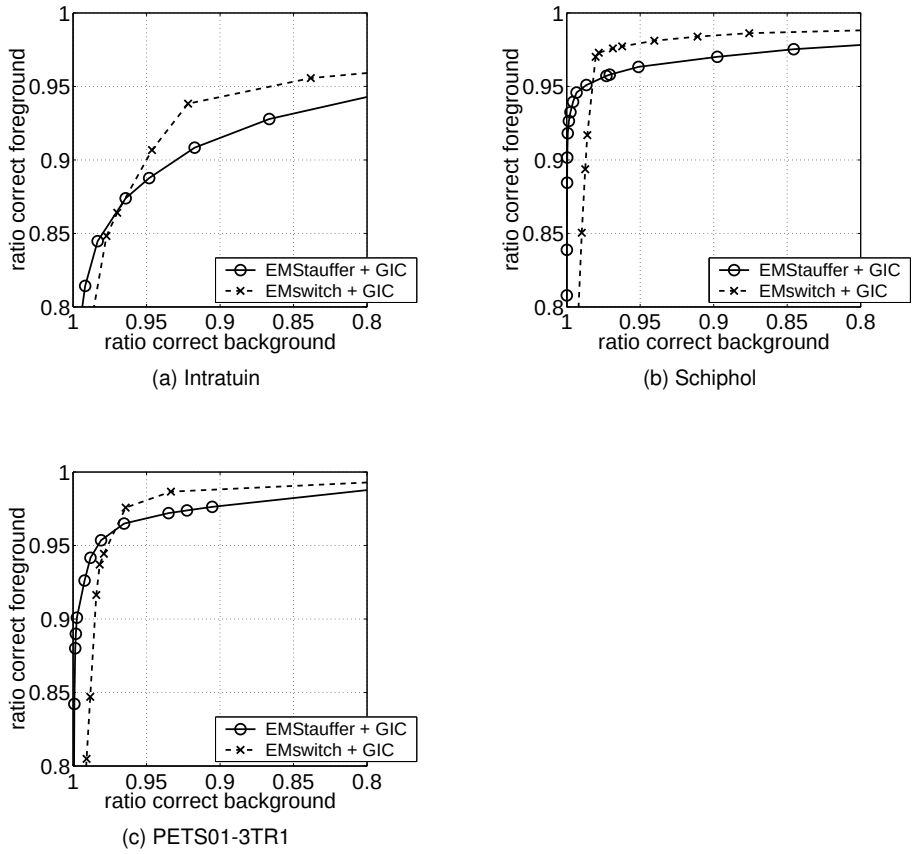


Figure 4.7: *The ROC for each of the image sequences. The algorithms EMStauffer + GIC and EMswitch + GIC are compared.*

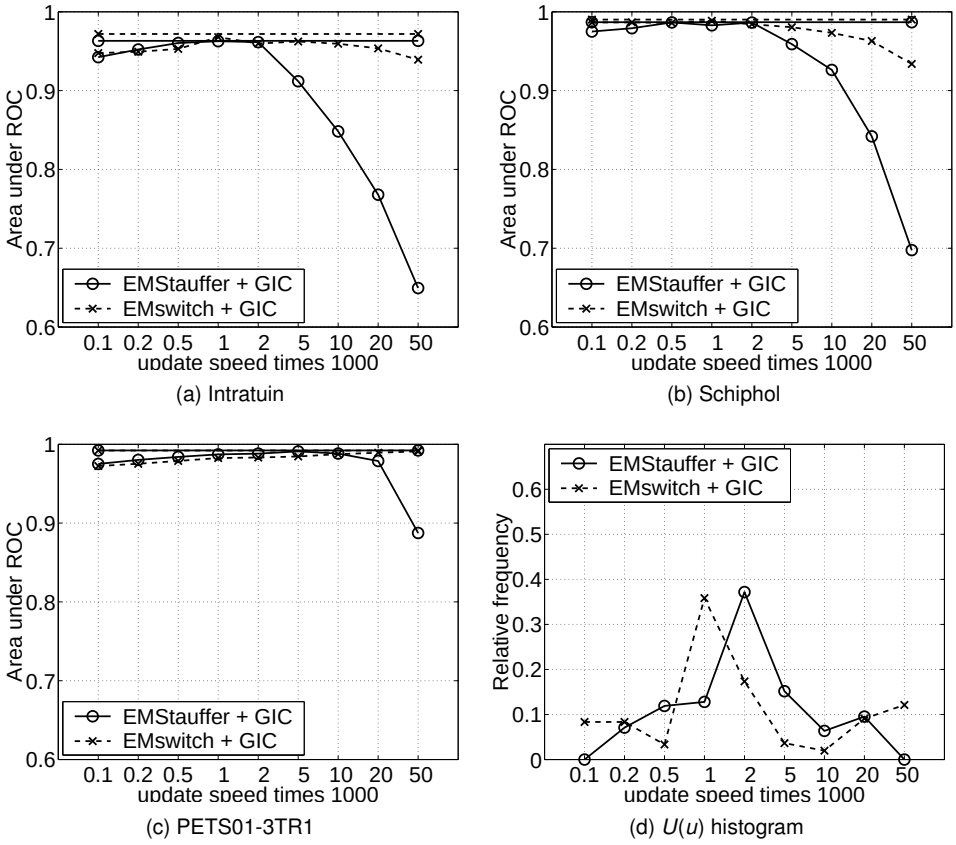


Figure 4.8: *The area under the u_B–ROC for different values of the update speed. The horizontal lines indicate the area under the total–ROC. Only a selection of the methods is shown as most methods coincide. Figure (d) shows the corresponding $U(u_B)$ histograms.*

Chapter 5 Probabilistic models of non-rigid objects

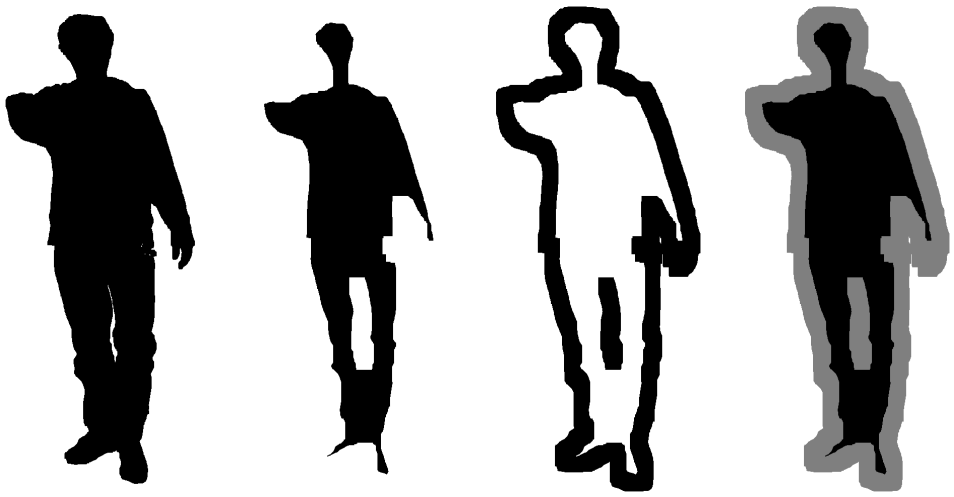
5.1 Introduction

In Chapter 2 an overview of object tracking techniques has been presented. It was concluded that an approach without explicit models of objects to be tracked is most flexible. It allows tracking of any moving object and even partial objects and groups of objects. Two approaches can be used: active contour and region-based approaches. The active contour approach can track an arbitrary shape, as long as the smoothness constraint is satisfied. However, there are a number of drawbacks, such as its computationally complexity and, more seriously, its robustness. Active contour approaches track the object boundary, which is the part of the object that changes most, specially for deformable objects as we intend to track.

Region based approaches usually have less complex algorithms and therefore lower computational complexity. They track the entire object region, which is more robust than using only its contour. For these approaches, the ability to track arbitrary shaped deformable objects is unusual. Template matching approaches track an arbitrary shape that is not assumed to change between subsequent frames [McKenna et al., 2000; Zhou and Tao, 2003]. Approaches based on the mean-shift algorithm [Comaniciu et al., 2000; Zivkovic and Kröse, 2004] or a particle filter [Nummiaro et al., 2003; Pérez et al., 2002] can allow a shape change between frames, but they need a parametric representation of the object shape. Often a rectangle or ellipse is used, both very rough simplifications of the shape of a walking human, let alone a bicyclist.

The approach presented in this chapter[1] combines advantages of both active contour-based and region-based object tracking. The main assumption is that we can split the object in a rigid part, its core, and a deformable part around the core, its shell, see Figure 5.1. This results in the Core and Shell Object Model (CaSOM).

[1] This chapter is based on [Withagen et al., 2002a,b, 2004b].



(a) Blob (b) Core (c) Shell (d) Core and Shell

Figure 5.1: *The tracked blob (a) is split in a core (b) and a shell (c). (d) shows the core in black and the shell in gray.*

The rigid object core is localized in a new frame using region-based tracking, while the deformable object shell is updated each frame in a probabilistic way.

The validity of CaSOMs main assumption is quite good if we allow holes in the object core and if we only allow small displacements of the core. It allows the object to deform between subsequent frames, but only to a limited amount. The amount of deformation is limited by the shell width.

This chapter is structured as follows. Locating the blob core will be described in Section 5.2. Once the location of the blob core is known, the probability that a pixel should be assigned to a specific blob for each of the blobs are calculated as described in Section 5.3. Initialization and updating of the models is discussed in Section 5.4 and in Section 5.5 prior probabilities for all classes are determined using an occlusion model. Finally experimental evaluation is described in Section 5.6 and conclusions are presented in Section 5.7.

5.2 Histogram based tracking of the blob core

Recall from Section 2.5 that the order between object tracking and classification can be chosen. We will use the most flexible order: first tracking and then classification. As a result, real-life moving objects can be split into a number of tracked blobs, or a tracked blob can correspond to a number of real-life objects. Without loss of generality, we will assume that a (tracked) blob corresponds to one real-life object.

CaSOM assumes a maximum speed of object motion and deformation. Therefore, the center of the blob, the core, can be fully retrieved in the subsequent frame. I.e. no part of the core will be occluded, outside the field of view or no longer visible due to shape deformation. Location of the core is obtained using region-based tracking based on the object color.

Several approaches are available for region-based tracking, see §2.5.1. The main difference is how much spatial information is used. Template matching retains all spacial information. A correlogram stores combinations of pixel colors of nearby pixels, retaining less spatial information. Even less spatial information is used when the object core is modelled using a number models without spatial information, for example using a Probability Density Functions (PDFs) of the color distribution in the core. For example, the core could be modelled using three color distributions, one for the head, one for the upper body and one for the lower body.

The advantage of not using spatial information is that it is computationally efficient and invariant to limited deformations of the blob itself. Describing different parts of the core with separate PDFs demands object knowledge and makes the algorithm less generally usable. Therefore, we will use a single PDF for the whole core.

Modelling this PDF using a normalized histogram allows for an efficient algorithm. Histogram-matching does not compare the image and a template, but only their two histograms. So when the region of interest is shifted over the image, for each new shift position, only the pixels which “shift in” and those that‘ “shift out” need to be processed to find the new match-error at that position.

A color histogram of the blob core is maintained as part of the blob model. In the new frame in a certain region around the old position of the blob, histogram matching is performed, finding the location l with minimal match error e_l given by

$$e_l = \sum_{\text{all bins}} |H_l - C| \,, \tag{5.1}$$

with C the blob-core histogram and H_l the histogram of a shifted version of the blob core at location l.

This histogram matching can be implemented efficiently by keeping the difference histogram in memory. At the first position we calculate:

$$D_1 = H_1 - C \,, \tag{5.2}$$

with D_1 the difference histogram at the first matching position, H_1 the histogram at the first position and C the histogram of the blob core.

For all subsequent positions, only those pixels that are added or removed by shifting to the next position need to be considered. The difference histogram is updated:

$$D_{n+1} = D_n - H_{\text{removed}} + H_{\text{added}} \,, \tag{5.3}$$

with H_{removed} and H_{added} the histograms of the removed and added pixels. The error

at this position is calculated by

$$e_{n+1} = e_n + \sum_{\text{added}} \text{sign}(D_n) - \sum_{\text{removed}} \text{sign}(D_n) \ , \tag{5.4}$$

with

$$\text{sign}(x) = \begin{cases} -1 & \text{for } x \leq 0 \\ 1 & \text{for } x > 1 \ . \end{cases} \tag{5.5}$$

5.3 Probability calculation

For each pixel x_i with feature vector $\vec{x}_i$, we need to calculate the probability density distribution $P(\vec{x}_i|\beta_j)$ for blob β_j. We assume that the core can be fully retrieved in the next frame. Therefore, pixels in the core are assigned unit probability. The pixels in a layer around this core, the shell, belong to the blob, to another blob, or to the background. For these pixels, foreground-blob probabilities are given by the joint probability distribution of pixel location and color:

$$P(\vec{x}_i|\beta_j) = P(\vec{c}_i, d|\beta_j) \ , \tag{5.6}$$

with d the distance to the core of the blob β_j, defined by the smallest number of pixels between the pixel and the edge of the core. $\vec{c}_i$ is the color vector of the pixel.

We assume the the distance and the color distribution independent. This simplifies Equation 5.6 to

$$P(\vec{x}_i|\beta_j) = P(d|\beta_j)P(\vec{c}_i|\beta_j) \ . \tag{5.7}$$

Assuming the color PDF independent of location assumes the color distribution over the entire object is equal. Under this assumption, the color distribution of the core must be equal to that of the entire blob. Therefore, the color probability of belonging to a certain blob is given by the color of the pixel compared to the color distribution of the core blob. $P(\vec{c}_i|\beta_j)$ is the value in the bin corresponding to $\vec{c}_i$ of the normalized color histogram of blob β_j.

We further assume small a deformation of the object shape. The probability of pixels close to the core will be high, while the probability of pixels at a distance from the core will be smaller. We assume that the blob displacement and deformation between subsequent frames is bounded by an upper value. Existing blobs therefore have a limited ”extent”, outside which the probability for these blobs is zero. The PDF of pixels lying outside the core and inside the maximal extend of the blob can be assumed or learned from the data. For example, edge information could be used, resulting in steps in the PDF at the location of edges. For simplicity, we will assume a uniformly decreasing probability.

Depending on the distance to the core of the blob, $P(d|\beta_j)$ is calculated. The uniformly decreasing probability gives:

$$P(d|\beta_j) = \begin{cases} 0 & \text{for } d > d_{\text{max}} \\ 1 - \frac{d}{d_{\text{max}}} & \text{for } 0 \leq d \leq d_{\text{max}} \ , \end{cases} \tag{5.8}$$

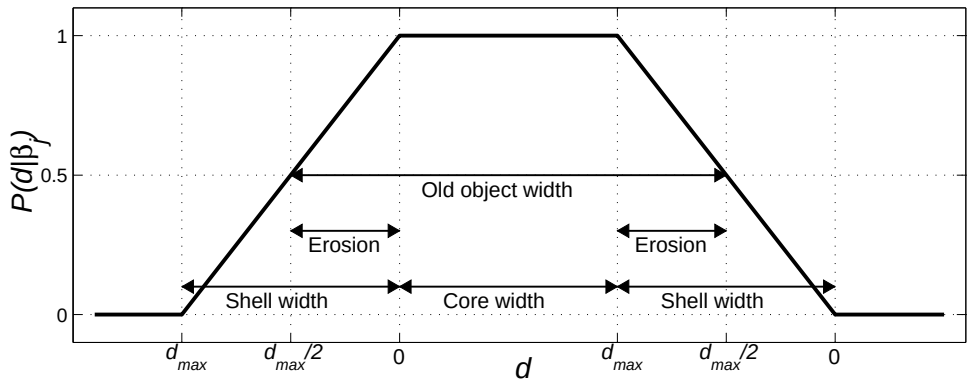


Figure 5.2: *Probability $P(d|\beta_j)$ as function of distance d to the core. The object core has unity probability, the core is constructed from the object in the previous frame by an erosion of size $d_{max}/2$. At a distance d_{max} the probability is zero.*

with $d_{\max}$ a parameter specifying the distance to the core in which the blob is relevant. This parameter depends on the maximum speed of the object and the amount it can deform between subsequent frames. Figure 5.2 shows a plot of $P(d|\beta_j)$ as a function of d.

We define "relevant" blobs as those blobs for which the pixel under classification is within its maximal extend. As such N_β, the number of relevant blobs, will depend on the location in the image. Given the location of all blob cores, we know which blobs are relevant for each pixel.

5.4 Model initialization and updating

New blobs are detected using a constant new blob probability $P(\vec{x}|\beta_{N_\beta+1})$. The expected maximal displacement and deformation between subsequent frames define the width $d_{\max}$ of the shell. In turn, this defines the minimal blob-size $N_{\min}$: blobs for which the core reduces to a few pixels cannot be tracked reliably. Blobs which are too small are re-labelled as background.

Once a blob has been tracked for a number of frames it is possible to use information of its motion in the past to reduce the width of the shell. Assumptions on limited deformation results in a thinner shell. Knowledge about the direction and speed of motion reduces the shell thickness at the "back" of the blob, because it is less likely that the object will be occluded there. Feedback enables the use of knowledge such as speed and direction. For simplicity, in this thesis a constant shell width is used.

After feedback, a new blob core is constructed by eroding the blob such that a

region with the shell width is removed. We choose this width equal to $d_{\max}/2$, see Figure 5.2. During feedback, the histogram of the blob core C_t is updated using

$$C_{t+1} = (1 - u_F)C_t + u_F C' \,, \tag{5.9}$$

with C' the histogram of the new blob core and u_F a parameter which regulates the update speed. All histograms used are normalized by letting the sum over all bins equal one.

CaSOM describes the color of a number of pixels. Together, these pixels form one, shared, model. Because several pixels contribute to the model, erroneously missing an update from a few pixels will not automatically lead to an underestimated variance. It is therefore less important to update each pixel every frame. That is why we do not use a multiple hypothesis approach as was used for updating the background in Chapter 4 for updating the blob models.

5.5 Occlusion modelling

After locating the cores of all blobs, we know which blobs are relevant to each pixel. If a pixel can be assigned to several blobs, all with a certain probability, we need to take into account the probability that the pixel belongs to the other blobs. We do this by calculating the prior probability $P(\beta_j)$ for each blob and each pixel. Prior probabilities are calculated using an occlusion modelling introduced below.

Consider a scene with multiple blobs (for simplicity we will assume each blob corresponds to one real-world object). We assume the blobs cannot be inside each other. Therefore, if multiple blobs overlap in view, one must be in front of the other. We have to deal with the occlusion problem. Assume that the ordering $\vec{\Omega}$ of blobs is known. The first element in this ordering $\Omega(0)$ denotes the relevant blob closest to the camera and the last element denotes the blob with the largest distance to the camera, typically the background. Background and new blobs are treated here as all other blobs.

The prior probability that we observe a blob in a given pixel is first of all related to its size. We define a probability based on the blobs size

$$P^*(\beta_j) = \frac{N_{\text{pixels}}(\beta_j)}{N_{\text{pixels}}} \,, \tag{5.10}$$

with $N_{\text{pixels}}(\beta_j)$ the number of pixels in blob β_j and N_{pixels} the number of pixels in the image. For the background we have $N_{\text{pixels}}(\beta_0) = N_{\text{pixels}}$ so $P^*(\beta_0) = 1$. Note that these probabilities are not independent. They assume that the current object is the only object in the scene. Therefore, their sum is always larger than one. For a new blob the size has to be defined beforehand or estimated based on the size of other blobs and the position in the image. Additionally, spatial information about the scene could be introduces here. For example, the probability of observing an object on the road is higher than on a building or in the air.

Besides its size, the probability that we observe blob $\beta_{\Omega(i)}$ in a certain pixel is given by the probability that we do not observe any blob in front of this blob i.e. blobs $\beta_{\Omega(0)}, \cdots, \beta_{\Omega(i-1)}$. These probabilities are regarded independent, which leads to the following equation for calculating the prior blob probability

$$P(\beta_{\Omega(i)}|\vec{\Omega}) = P^*(\beta_{\Omega(i)}) \prod_{n=0}^{i-1} (1 - P^*(\beta_{\Omega(n)})) \ . \tag{5.11}$$

Equation 5.11 assumed a known ordering of the blobs. For the general case we should sum over all possible orderings N_Ω:

$$P(\beta_j) = \sum_{o=1}^{N_\Omega} P(\vec{\Omega}_o) P(\beta_j|\vec{\Omega}_o) \ , \tag{5.12}$$

with $P(\vec{\Omega}_o)$ the probability that ordering $\vec{\Omega}_o$ is correct. These prior probabilities sum to one:

$$\sum_{j=0}^{N_\beta+1} P(\beta_j) = 1 \ . \tag{5.13}$$

5.5.1 Computational complexity

As more blobs become relevant, equation 5.12 becomes expensive to compute. Its complexity increases with $(N_\beta+2)!$. For real-time implementations this is prohibitive. Fortunately, there exist two situations where the complexity is lower. The first situation occurs when the ordering is known. In this case equation 5.12 has to be computed for one ordering only.

The other situation for which equation 5.12 simplifies occurs when all orderings are assumed equally probable. No information is used about the ordering of blobs. For only background, one, two, or three relevant foreground blobs, equation 5.12 simplifies to

$$P(\beta_0) = P^*(\beta_0) \ , \tag{5.14}$$

$$P(\beta_0) = P^*(\beta_0)(1 - \frac{1}{2} P^*(\beta_1)) \ , \tag{5.15}$$

$$P(\beta_0) = P^*(\beta_0)(1 - \frac{1}{2}(P^*(\beta_1) + P^*(\beta_2)) + \frac{1}{3}(P^*(\beta_1)P^*(\beta_2))) \ , \tag{5.16}$$

$$\begin{aligned} P(\beta_0) = P^*(\beta_0)(1 - \frac{1}{2}(P^*(\beta_1) + P^*(\beta_2) + P^*(\beta_3)) + \\ \frac{1}{3}(P^*(\beta_1)P^*(\beta_2) + P^*(\beta_1)P^*(\beta_3) + P^*(\beta_2)P^*(\beta_3)) - \\ \frac{1}{4} P^*(\beta_1)P^*(\beta_2)P^*(\beta_3)) \ , \end{aligned} \tag{5.17}$$

respectively. Simplified equations for more blobs also exist, but the higher order terms quickly become negligible.

5.6 Experimental evaluation

Evaluation of the CaSOM foreground model will be performed in conjunction with the EMswitch background model proposed in Chapter 4 and Global Intensity Correction (GIC) using the Median of the Quotient (MofQ) with statistical outlier removal, see Chapter 3.

The three image sequences given in Chapter 3 will be used for evaluation, see also Appendix B. The algorithm will be compared two reference algorithms: EM-Stauffer+GIC (§4.2.2) and EMswitch+GIC (Chapter 4) based on the (full) ROC and the surface above the ROC.

5.6.1 Implementation details

Images are corrected using global intensity correction. After an EMswitch initialization period of 200 frames, pixels are classified by comparing three probabilities for each pixel: background probability $P(\vec{x}|\beta_0)$ given by EMswitch, existing track probability $P(\vec{x}|\beta_j)$ for each of the tracks given by CaSOM and new track probability using a constant value $P(\vec{x}|\beta_{N_{\beta+1}}) = 0.1$.

The Bayes decision rule is used to perform minimum cost classification:

$$E(x_i) = \min_n \Big(\sum_{j=0}^{N_\beta+1} C_\beta(\beta_n, \beta_j) P(\beta_j|\vec{x}_i) \Big) \ , \tag{5.18}$$

with E the total classification cost and C_β the cost matrix for classifying object β_n as object β_j.

Conditional probabilities $P(\beta_m|\vec{x}_i)$ are calculated using Bayes' rule

$$P(\beta_j|\vec{x}_i) = \frac{P(\beta_j)P(\vec{x}_i|\beta_j)}{P(\vec{x}_i)} \ , \tag{5.19}$$

with $P(\vec{x}_i) = \sum_j P(\beta_j)P(\vec{x}_i|\beta_j)$ a normalization factor. As this factor is equal for all output classes, it is omitted.

Prior probabilities $P(\beta_j)$ are calculated using occlusion modelling without using ordering knowledge and an expected blob size of $N_{pixels} = 200$ pixels.

The following parameter settings have been used for the CaSOM algorithm in conjunction with EMswitch and GIC:

- **EMswitch** parameters have not been changed from those given in Chapter 4.
- **Cost** The cost for missing an object are chosen unity $C_\beta(\beta_j, \beta_0) = 1$. The cost for false alarm $C_\beta(\beta_0, \beta_j)$ and track confusion $C_\beta(\beta_j, \beta_n)$ are varied to create an ROC. Cost of correct classification are zero $C_\beta(\beta_j, \beta_j) = 0$.
- For **histogram matching** a search area of 11×11 is used. The histograms have 16^3 bins in RGB.

- **Sizes** used are d_{max} = 6 and N_{min} = 50 pixels.

- The background **update speed** was fixed at u_B = 0.001 and the foreground update speed at u_F = 0.1.

5.6.2 Classification results

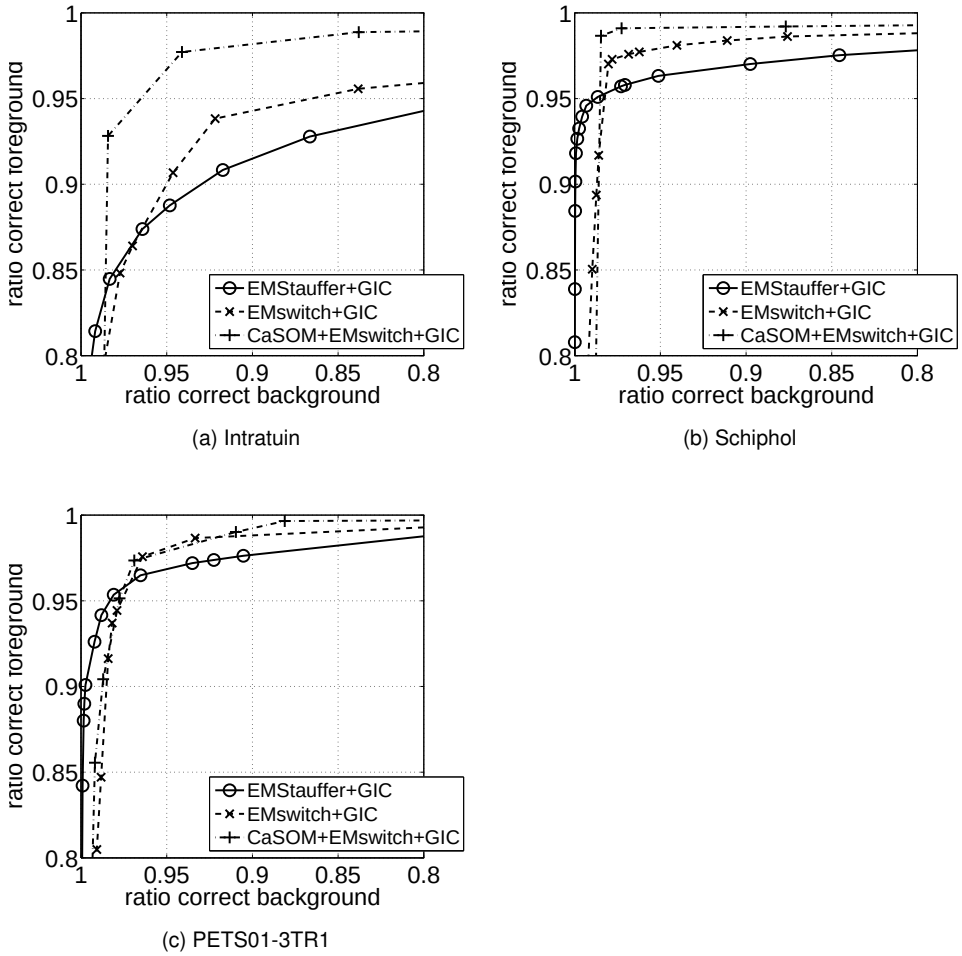
In Figure 5.3 the ROCs for each of the image sequences are given. For all image sequences the classification result is better than the reference methods for lower percentages of correct background (e.g. 90%). For higher percentages of correct background classification, using CaSOM causes the correct foreground percentage to drop more quickly than the reference methods. This is caused by shadows. Shadow pixels are classified to an object, causing the color of the (shaded) background to be learned into the object model. Correcting for shadows (Chapter 6) solves this, see also the classification results in Section 7.4.

The areas above the ROCs are given in Table 5.1. The Intratuin and Schiphol sequences show a significant improvement to both reference methods. For these sequences, the problem caused by shadow is less important than the improvement due to the model. The PETS01-TR1 sequence show an improvement compared to EMStauffer+GIC and a worsening to EMswitch+GIC. This is caused by large shadow areas caused by moving clouds. Once detected, these areas are continued to being tracked and cause a lot of false alarms.

Besides shadows, Two of our assumptions were not always valid, leading to suboptimal results. First, the CaSOM method assumes relatively large objects, and indeed, we saw that small objects cause problems. The PDF of the color, the color histogram, becomes very sparse for small objects, leading to a bad representation of the true PDF. Second, the color PDF of the core is assumed to be equal to that of the entire object. This is generally not true for humans. For example, the hair and shoes of people often caused problems. Such region is then assigned to a new, very small object, which then suffers from the first problem. These two problems are, together with shadow, the main cause of less than perfect results.

Table 5.1: *Results for CaSOM. Given is the surface above the ROC in percentages (lower is better).*

Data:	Intratuin	Schiphol	PETS01-3TR1	Average
EMStauffer+GIC	3.68	1.32	0.78	1.93
EMswitch+GIC	2.81	0.99	0.79	1.53
CaSOM+EMswitch+GIC	1.69	0.80	0.76	1.08



(a) Intratuin

(b) Schiphol

(c) PETS01-3TR1

Figure 5.3: *The ROC for each of the image sequences. The different algorithms compared are EMStauffer with global intensity correction, EMswitch with global intensity correction and CaSOM in conjunction with EMswitch and global intensity correction.*

5.7 Conclusions

In this chapter we introduces the Core and Shell Object Model (CaSOM). It is based on the assumption that the deformable objects to be tracked can be split in a rigid core and a deformable shell.

The blob core is tracked using histogram matching. This is computationally efficient and relatively invariant to object deformations.

For pixels in the shell, near the blob boundary, blob probabilities are calculated to be used in the probabilistic tracking framework. Probabilities are based on pixel color and distance to the blob core.

Prior probabilities are determined using an occlusion model. A computationally efficient implementation can be obtained when either the order of blobs is known exactly or not known at all.

Experimental evaluation shows a significant error reduction for the Intratuin and Schiphol sequences. The PETS01-TR1 sequences suffers from large shadow areas. For this image sequence the results are worse, specially for low background errors ($<$ 10%). This is caused by shaded background colors being incorporated in the foreground model. An other reason for less than perfect results are two assumption which were not always valid. The color distribution of the core is not always representative for that of the entire object and for small objects the histogram description used to model the color PDF was not a sufficient description of the true PDF.

Chapter 6 Shadow detection using a physical basis

6.1 Introduction

Shadow classification[1] is an important aspect of most object detection and tracking algorithms. Practically any scene, both indoor and outdoor, contains shadows. Object detection algorithms that do not rely solely on intensity-invariant features detect these shadows as foreground. This alone is a problem, as object segmentation will be inaccurate and objects can become connected by their shadows.

A more severe problem occurs for object tracking algorithms that use a model of the moving object, as we do with our CaSOM foreground model described in Chapter 5. Shadow is labelled part of a moving object, leading to updating the model with a shaded version of the background. The color model of the moving objects is often less sensitive than that of the background, as it should describe several object colors using less training data. Therefore, after updating the model with shaded background, the unshaded background can also give a high object probability. This may lead to situations where large parts of the background are misclassified as objects. Therefore, specifically for this kind of algorithms it is important to remove shadows from foreground objects, i.e. to reassign shadow pixels that were initially classified as foreground to the background.

Many shadow detection algorithms exist. The most simple solution is to ignore intensity information altogether by using an intensity invariant color space such as [Greiffenhagen et al., 2001]. This assumes that shadow causes the intensity to change, while leaving the color information unchanged. An important drawback of this approach is that almost black and almost white pixels are treated equal to other pixels, while for these pixels the color information is highly unreliable.

A better choice is to combine color and intensity information like [Horprasert et al., 1999; KaewTraKulPong and Bowden, 2001]. These authors allow distortions

[1] This chapter is based on [Withagen et al., 2005c].

on the pixel value: the intensity and the color. This is equivalent to assuming a cylinder in RGB–space through $(0, 0, 0)$ and the old background RGB value. A pixel located inside this cylinder is classified as (shaded or highlighted) background, while pixels outside the cylinder are classified as foreground. This approach allows to assign pixels with high or low intensity to the foreground in the case additional noise is assumed. When the image noise is multiplicative, a cone should be used instead of a cylinder.

Both the normalized intensity approach and the cylinder approach assume that under shadow the color change is small and equally probable in each direction. This is only true when all light sources have the same color. However, this is generally not the case. For example: an outdoor scene usually has two light sources, direct sunlight and blue ambient sky illumination. Shadows will cause a color shift of which the direction can be predicted.

Generally, the colors of the illumination sources will be relatively constant over time. They can therefore be learned. When the colors of the illumination sources are known, the color shift caused by occluding one or more of the light sources can be predicted. A known color shift allows a smaller color area to be classified as shadow, leading to less missed objects.

In this chapter we analyze the physical aspects of the color change caused by shadow. We propose the q–space and an algorithm to predict the color change caused by shadow. This leads to an algorithm for accurate shadow classification by taking into account the predicted color shift. The proposed algorithm is evaluated using real image sequences.

Our approach is most similar to [Nadimi and Bhanu, 2004]. However, instead of using a constant, predefined color-shift towards blue, we learn the color shift from the images. In addition, our approach is not limited to pre-defined background materials with large enough homogenous areas and we allow highlights, which may be caused by shadow of the background image.

6.2 Physics of shadow

The amount of light reaching the CCD sensor $C_c(p)$ of sensor band $c \in \{R, G, B\}$ at position p is given by [Forsyth and Ponce, 2002]

$$C_c(p) = \int_\lambda e(\lambda, p) \cdot \rho(\lambda, p) \cdot f_c(\lambda)\, \mathrm{d}\lambda \,, \tag{6.1}$$

with $e(\lambda, p)$ the illumination spectrum, $\rho(\lambda, p)$ the surface reflection (surface albedo), $f_c(\lambda)$ the sensor response function and λ the wavelength. $\rho(\lambda, p)$ will generally depend on the direction of the incoming light and the direction of the camera. We will assume only diffuse reflections.

The illumination spectrum can be modelled as a sum of all contributing, inde-

pendent, light sources $e_1 \cdots e_{N_l}$:

$$e(\lambda, p) = \sum_{l=1}^{N_l} e_l(\lambda, p) \ , \tag{6.2}$$

where $e_l(\lambda, p)$ depends on the direction of the light and the camera.

Due to changing (partial) occlusion, the illumination of each of the light sources can change over time. Under the assumption that this only affects the intensity and not the spectral composition of the light sources, this can be modelled by

$$e(\lambda, p, t) = \sum_{l=1}^{N_l} \alpha_l(p, t) \cdot e_l(\lambda, p) \ , \tag{6.3}$$

with $\alpha_l(p, t)$ a time and position dependent factor accounting for a changing illumination intensity over time.

In a local area, where the illumination spectrum of each light source can be considered spatially constant, the illumination spectrum no longer depends on the pixel location $e_l(\lambda, p) = e_l(\lambda)$. This results in a sensor illumination $C_c(p, t)$ at time t of

$$C_c(p, t) = \sum_{l=1}^{N_l} \alpha_l(p, t) \int_\lambda e_l(\lambda) \cdot \rho(\lambda, p) \cdot f_c(\lambda) \, \mathrm{d}\lambda \ , \tag{6.4}$$

under the assumption that the scene is static and the location and spectral features of the illumination and camera are constant over time.

When either the illumination spectrum or the reflectance spectrum can be considered constant over the spectral range of each of the camera bands[2] this equation can be further simplified. In this case, the term that may be considered constant can be put in front of the integral. We can then split the sensor illumination in a part that depends on time and light source $\alpha_l(p, t)$, a part that depends on the light source and camera band $L_{c,l}$ and a part that depends on the location and camera band $R_{c,p}$:

$$C_c(p, t) = R_{c,p} \cdot \sum_{l=1}^{N_l} \alpha_l(p, t) \cdot L_{c,l} \ , \tag{6.5}$$

with

$$L_{c,l} = \int_\lambda e_l(\lambda) \cdot f_c(\lambda) \, \mathrm{d}\lambda \quad \text{and} \tag{6.6}$$

$$R_{c,p} = \frac{\int_\lambda \rho(\lambda, p) \cdot f_c(\lambda) \, \mathrm{d}\lambda}{\int_\lambda f_c(\lambda) \, \mathrm{d}\lambda} \ . \tag{6.7}$$

[2] So there are no peaks in both illumination and reflection spectra This assumption holds for smooth light spectra such as that of the sun and normal light bulbs. Fluorescent light in combination with peaked reflection spectra may cause problems.

Adding the CCD model

We show in Appendix A that for sufficiently large pixel values the CCD sensor can be modelled by

$$i_c(p,t) = g(t)^\gamma \left(h(t)C_c(p,t) + N_S\right)^\gamma , \tag{6.8}$$

with $C_c(p,t)$ equal to the scene intensity i_t in Equation A.5, but now for each color channel $c \in \{R,G,B\}$. $g(t)$ denotes the camera gain, $h(t)$ is a factor related to the shutter time and iris setting. γ is the gamma-correction of the camera. The noise term N_S is described in Appendix A.

Two light sources

Let us consider one pixel at location p_0 at times t_0 and t_1. There are two light sources with colors $\vec{L}_1$ and $\vec{L}_2$. The reflectivity of the scene depicted in this pixel is given by $\vec{R}(p_0)$. This gives for the (noise free) pixel intensity in the two frames:

$$i_c(p_0,t_0) = g^\gamma(t_0)\left(R_c(p_0)h(t_0)(\alpha_1(p_0,t_0)L_{c,1} + \alpha_2(p_0,t_0)L_{c,2})\right)^\gamma \quad \text{and} \tag{6.9}$$

$$i_c(p_0,t_1) = g^\gamma(t_1)\left(R_c(p_0)h(t_1)(\alpha_1(p_0,t_1)L_{c,1} + \alpha_2(p_0,t_1)L_{c,2})\right)^\gamma , \tag{6.10}$$

With α a factor depicting the brightness of the light sources. We assume that the reflectivity is equal in both images, e.g. constant scene.

6.3 The q–space for shadow

We will assume $\gamma = 1$ and g and h constant for shadow detection. Alternatively, images can be corrected for gamma before shadow detection. The quotient of one pixel at two instances in time is given by:

$$q_c(p_0,t_{1,0}) = \frac{i_c(p_0,t_1)}{i_c(p_0,t_0)} = \frac{\alpha_1(p_0,t_1)L_{c,1} + \alpha_2(p_0,t_1)L_{c,2}}{\alpha_1(p_0,t_0)L_{c,1} + \alpha_2(p_0,t_0)L_{c,2}} . \tag{6.11}$$

This quotient is independent of the scene reflectivity. When the two images have equal illumination, so there is no shadow, $q_c = 1$ for all color bands. We now assume that non-unity values of q_c are due to shadows or highlights.

We consider a region S in the image with equal reference illumination, for which the L's and α's are assumed equal for each pixel at t_0. We define this illumination $\vec{L}_0$:

$$\vec{L}_0(p_i \in S) = \alpha_1(p_i,t_0)\vec{L}_1 + \alpha_2(p_i,t_0)\vec{L}_2 , \tag{6.12}$$

for all $p_i \in S$.

For different α values at time t_1, the $\vec{q}$–values for each pixel $p_i \in S$ lie on a flat surface in the 3D q–space. This surface is defined by the two vectors $\vec{v}_1$ and $\vec{v}_2$ with elements $v_{1,c}(p_i,t_1) = \frac{\alpha_1(p_i,t_1)L_{c,1}}{L_{c,0}(p_i)}$ and $v_{2,c}(p_i,t_1) = \frac{\alpha_2(p_i,t_1)L_{c,2}}{L_{c,0}(p_i)}$ respectively. This surface has the shape of a triangle with one corner at $(0,0,0)$ and two sides given

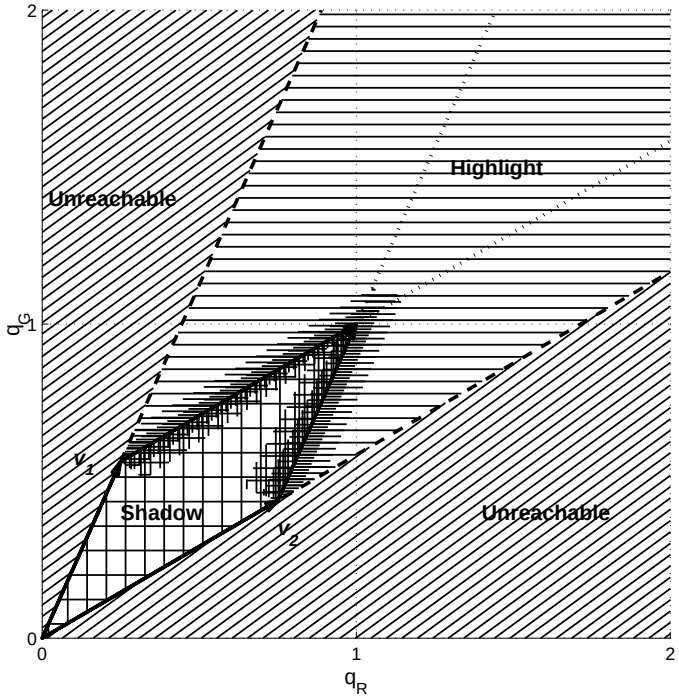


Figure 6.1: *Analysis of a 2D projection of the q–space. A surface is spawn by two vectors $\vec{v}_1$ and $\vec{v}_2$. A shaded pixel falls inside the double-hatched parallelogram, a highlighted pixel in the horizontally hatched area. When only one illumination source is (partly) occluded a pixels moves along the corresponding dotted line. The two diagonally hatched areas depict areas that cannot be reached. Some areas are hatched more dense, indicating the higher density of the q–space for a typical image.*

by the two vectors. Point $(1, 1, 1)$ always lies on this surface. See Figure 6.1 for a 2D plot of the q–space. The q–space is divided in three volumes:

- Values outside the triangle with its top at $(0, 0, 0)$ cannot occur when the assumptions of constant scene and constant illumination are valid.
- The triangle defines all possible q's. It is divided in two 2D surfaces:
  - Inside a parallelogram with corners at $(0, 0, 0)$ and $(1, 1, 1)$ and sides in the direction of $\vec{v}_1$ and $\vec{v}_2$ are all pixels that are under shadow.
  - Outside this parallelogram, but inside the triangle are all pixels that are highlighted, i.e. their illumination intensity is higher than that of the ref-

erence image for at least one of the light sources. This occurs when the reference image contains shadows.

Due to noise, the flat surface will in reality have a certain thickness.

6.3.1 Online estimation of the q–space parameters

For known colors of illumination sources, the vectors $\vec{v}_1$ and $\vec{v}_2$ can be calculated and used for shadow correction. However, in general this information is not available. The relative color of the illumination sources should then be estimated from the image sequence.

We consider the case that only one light source is (partially) occluded. The shadow area in q–space is now formed by a straight line through $(1,1,1)$. Assuming the second light source is occluded, $\vec{q}$ is given by

$$\vec{q}(p_0, t_{1,0}) = 1 + (\frac{\alpha_2(p_0, t_1)}{\alpha_2(p_0, t_0)} - 1)\vec{v}_2 \ . \tag{6.13}$$

This is a straight line through $(1,1,1)$ with the direction given by $\vec{v}_2$. In Figure 6.1 these values of $\vec{q}$ are densely hashed for both light sources. Occluding only the second light source generates the line form the end of vector $\vec{v}_1$ to $(1,1,1)$.

We project the q–space such that different points in the q–space that lie on a straight line through $(1,1,1)$ are projected on the same point. The following projection enables this:

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \frac{1}{F(\vec{q})} \begin{pmatrix} 0 & \cos(30^\circ) & -\cos(30^\circ) \\ 1 & -\sin(30^\circ) & -\sin(30^\circ) \end{pmatrix} \begin{pmatrix} q_R \\ q_G \\ q_B \end{pmatrix} \ , \tag{6.14}$$

with the normalization factor $F(\vec{q})$

$$F(\vec{q}) = \frac{q_R + q_G + q_B}{3} \ . \tag{6.15}$$

From the values b_1 and b_2 for all pixels in S (defined in Equation 6.12) we estimate the parameters $\overline{b_1}$ and $\overline{b_2}$ using the mean or median. These estimates define the color shift caused by partial occlusion of one light source for image region S.

Generally, different pixels will have a different ratio of $\alpha_2(p_0, t_1)$ and $\alpha_2(p_0, t_0)$ in Equation 6.13. For example, close to an object a larger part of the sky will be occluded than at some distance, while direct sunlight can be fully occluded for both pixels. So we need to estimate a line-segment in the b–space. This line segment will have certain thickness. It spans the V-shaped plane in the q–space, also with certain thickness. The line segment is estimated using an ellipse, where the two foci are assumed to be the endpoints the line segment, so the coordinates of the vectors $\vec{v}_1$ and $\vec{v}_2$.

The length of the line segment with respect to the width of the line is determined by:

- The amount of color saturation in the images.
- The difference in color of the two illumination sources.
- The amount of variation in the ratio of α's between pixels.
- The amount of shadow pixels.
- The amount of image noise.

6.3.2 Demonstration of the *b*–space

Figures 6.2 and 6.3 show four examples of color shifts due to shadow and object. For one pixel over a number of frames the *q*–values are calculated by dividing the value of one pixel for a number of frames by the pixel value without shadow/object. From these *q*–vales, *b*–values are calculated and using the median over all frames. The color shift is calculated. The estimated value of *b* defines the dashed line in the *q*–space.

In Figure 6.2 two examples are shown from the Intratuin sequence. Figure 6.2(a) is an example of shadow. The plot of the *b*–space shows that the *b*–values for the different frames are clustered away from the center, in the direction of red. This means that the shadow causes a color shift towards red. It gives in the *q*–space a line through $(1, 1, 1)$ away from the line through $(0, 0, 0)$ and $(1, 1, 1)$, which is the orientation of the cylinder in the reference algorithm. It is worth mentioning that most shadows in our experiments did not cause values of q_c below 0.5, lower values were caused by objects.

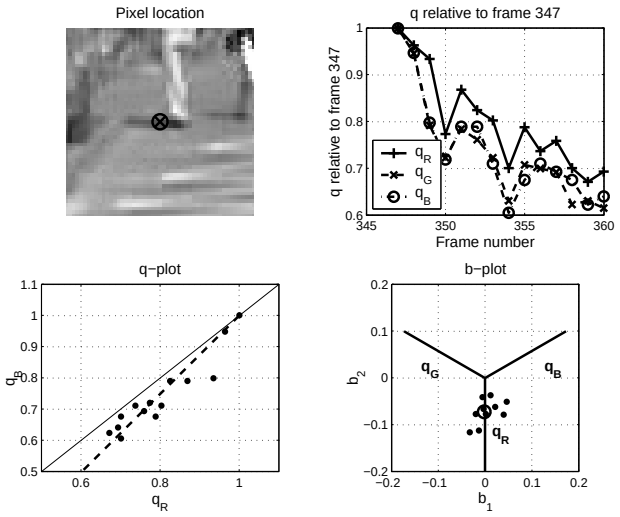
The distance to the origin in the *b*–plot gives the amount of color shift. The larger this distance, the better the proposed approach will perform compared to the approach using a cylinder or cone.

In Figure 6.2(b) an example of a blue object is given. It can be seen that the values in the *b*–plot are much less clustered. The direction of the color shift is now opposite to that in the case of shadow.

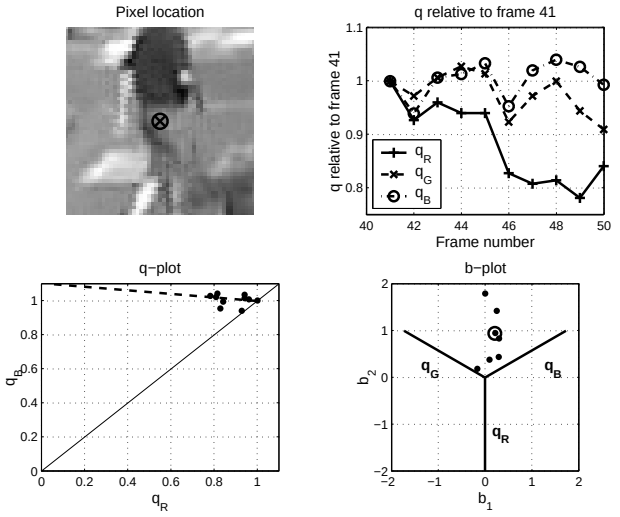
Figure 6.3 shows two examples for the PETS01-3TR1 image sequence. Again, one example of shadow and one example of an object. The color shift due to shadow is here towards blue. The example of an object in Figure 6.3(b) shows a white object passing the pixel. The *b*–values are again much less clustered.

6.4 Experimental evaluation

In order to quantitatively evaluate the proposed algorithm it will be used to remove shadow in a foreground/background classification task on real image sequences. The EMswitch classification results given in Chapter 4 will be used as initial foreground/background segmentation. Pixels classified as background are used to estimate an ellipse in the *b*–space, assuming that the background region also contains shadow pixels. This ellipse is them used to classify the foreground pixels. Evaluation is performed by comparing the classification result to that of a reference algorithm.

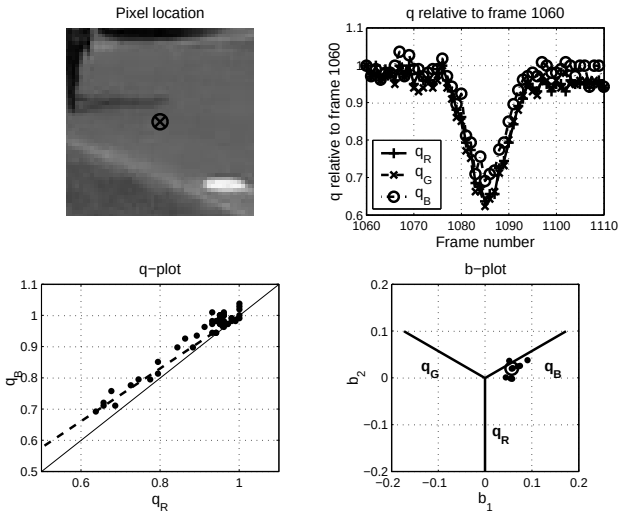


(a) Intratuin, shadow

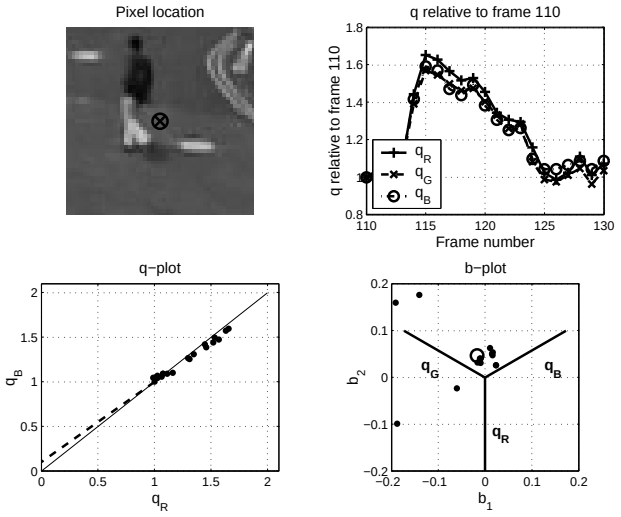


(b) Intratuin, object

Figure 6.2: *Examples of the b–space. Top left shows the location of the pixel. The q–values relative to a frame without shadow are plotted in the figure on the top-right. The bottom left figure shows a plot of q for two dimensions. The b–space is given in the bottom right figure. A circle in this plot depicts the median of all data points. This value of b defines the dashed line in the bottom-left plot.*



(a) PETS01-3TR1, shadow



(b) PETS01-3TR1, object

Figure 6.3: *Examples of the b–space. Top left shows the location of the pixel. The q–values relative to a frame without shadow are plotted in the figure on the top-right. The bottom left figure shows a plot of q for two dimensions. The b–space is given in the bottom right figure. A circle in this plot depicts the median of all data points. This value of b defines the dashed line in the bottom-left plot.*

6.4.1 Implementation details

The value of fixed parameters will be given below. The other parameters were varied for parameter optimization.

Reference algorithm

The reference method classifies pixels based on their change in color. This is equivalent to using a cylinder in the q–space with its axis through points $(0, 0, 0)$ and $(1, 1, 1)$. The cylinder has a width T_{cylinder}. Pixels with a value within the cylinder are classified as shadow, pixels outside the cylinder as foreground.

Upper and lower bounds

Pixels with a component of $\vec{q}$ below T_{qmin} or above T_{qmax} are not considered to be shadow. The same yields for pixels for which one of the components of $\vec{q}$ is closer to one than T_{qone}. These three parameters are used for both the proposed and the reference algorithm.

Threshold

The proposed algorithm iteratively estimates the smallest ellipse containing the fraction T_{ellipse} pixels on the background pixels in the b–space.

Post-processing

In order to prevent missing objects, prior knowledge is used in a post-processing stage. We know that a shadow is cast by an object. It can be either connected to that object or not. We assume that a foreground region is an unconnected shadow region when more than a fraction T_{total} of the pixels in the region are classified as shadow based on their color.

Shadows are in most cases cast on the ground. When a shadow region is connected (at least one pixel is 4–connected) to a moving object, we assume that the shadow starts near the lowest point of the object. Therefore, shadow regions that do not have shadow pixels in the lowest $T_{\text{connected}}$ fraction of the blob are relabelled object.

We used for both the reference and the proposed algorithm $T_{\text{total}} = 0.8$ and $T_{\text{connected}} = 0.2$.

6.4.2 Evaluation criterion

The shadow detection is intended to be used in conjunction with foreground/background classification. The performance of shadow detection will be evaluated with this application in mind.

As shadow pixels will be considered background, there is no sense in performing shadow classification on pixels classified as background. So only pixels classified as foreground by the foreground/background classification algorithm will be considered.

The set of pixels classified as foreground contains two sub-sets: true foreground F_T, those pixels correctly classified as foreground, and false foreground F_F, those pixels that are actually background. Pixels in set S of pixels classified as shadow belong to either of these subsets. We propose two fractions

$$r_{\text{bad}} = \frac{N(S \in F_T)}{N(F_T)} \quad \text{and} \tag{6.16}$$

$$r_{\text{good}} = \frac{N(S \in F_F)}{N(F_F)} \ , \tag{6.17}$$

with $N(S)$ the number of pixels in (sub)set S. Perfect shadow classification gives $r_{\text{good}} = 1$ and $r_{\text{bad}} = 0$. To evaluate different settings of the algorithm, a plot will be made of r_{bad} against r_{good} for different parameter settings. This plot will be called the good/bad–plot. It looks like an ROC curve. As with the ROC curve, only points on its convex hull will be drawn.

To demonstrate the contribution of shadow removal to foreground/background classification, also the ROC before and after shadow removal will be given.

6.4.3 Experimental results

In Figure 6.4 the convex hull of the good/bad–plot is given for the proposed and the reference method. Except for the PETS01-3TR1 image sequence, the proposed algorithm performs better than the reference method based on a cylinder. Numerical results in Table 6.1 show an average error reduction of a factor two. The results for the PETS01-3TR1 sequence are almost identical for both the proposed and the reference algorithm. This is probably caused by the lack of color saturation or the equal color of illumination sources in this image sequence. This causes very small color shifts, reducing the results of the proposed algorithm to be similar to that of the reference algorithm.

In order to see the effect of shadow removal, the total–ROCs (see §3.5.2) before and after shadow removal are shown in Figure 6.5. The difference between using and not using shadow removal is considerable, except for the PETS01-3TR1 image sequence. Both shadow removal techniques have little effect on the classification result for this image sequence. The shadows that occur in this sequence are large areas of shadow caused by moving clouds. The post processing used allows these regions to be classified as shadow area only when more than 80 percent of the pixels in the blob are classified as shadow. It seems that this is not obtained in most cases. This can be solved using information of all pixels in the shadow blob as will be proposed in Chapter 7.

The difference between the proposed shadow removal algorithm and the reference method seems small in the ROC's. Most of the improvement is seen for

Table 6.1: *Surface above the total–ROCs and the good/bad–plots. All numbers are percentages where lower is better.*

Data:	Intratuin	Schiphol	PETS01-3TR1	Average
Good/bad–plot Cylinder	20.9	36.6	10.63	22.7
Good/bad–plot Proposed	10.1	12.6	11.07	11.3
total–ROC none	2.81	0.99	0.79	1.53
total–ROC Cylinder	2.23	0.51	0.79	1.18
total–ROC Proposed	1.87	0.43	0.67	0.99

very high values of the correct background ratio, see the different cut-out in Figure 6.5(d). It is not surprising that there is less improvement for lower values of the correct background ratio as for these values the number of pixels classified as background is relatively small. This means that most shadow pixels will be classified as foreground and not used to estimate the direction of the color shift. The color shift is then estimated solely on the noise in the q–space of unshaded background pixels. This will in most cases lead to a shadow region which is very similar to that of the reference algorithm. The solution for this is the use of different thresholds for foreground classification and for the estimation of the color of illumination.

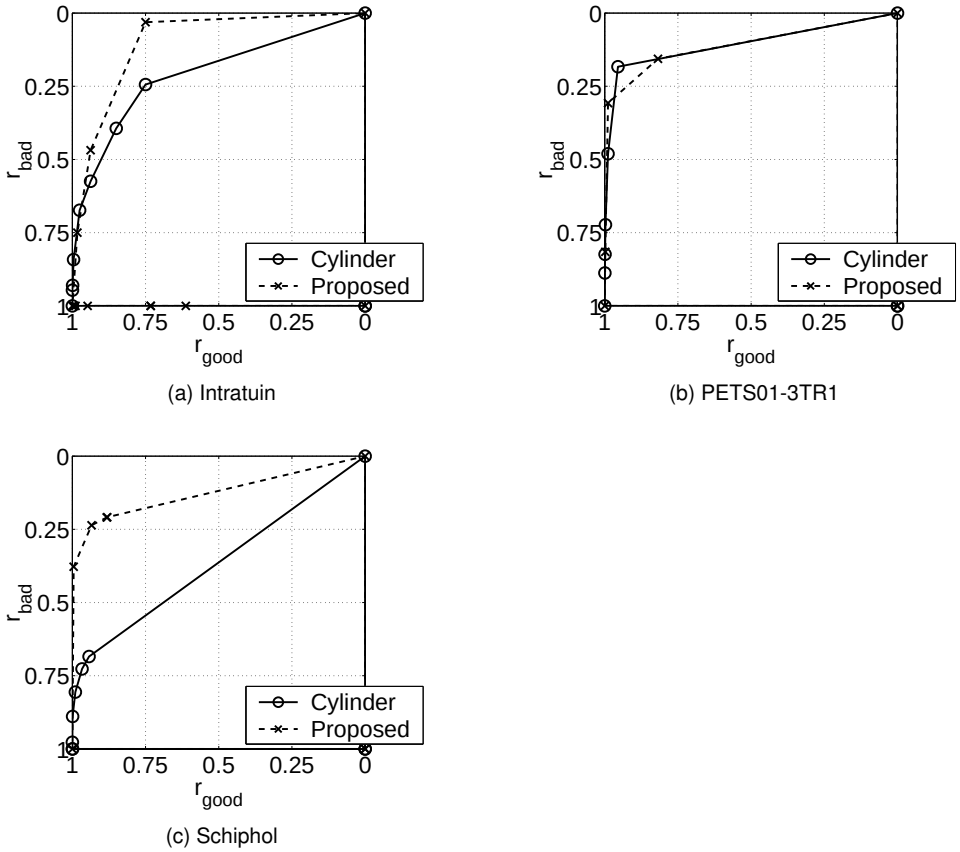
Table 6.1 gives the numerical results. On average, an error reduction of 35 percent is obtained with the proposed shadow removal algorithm. For the cylinder-based reference algorithm this is only 23 percent.

6.5 Conclusions

In this chapter a new shadow detection algorithm has been introduced. It was argued that for scenes with two or more illumination sources with different colors, shadow causes not only the intensity but also the color to change.

We have demonstrated that the color shift depends linearly on the change in intensity. The direction of the shift in color depends on the difference in color of the light sources. It has been shown that the shift direction can be estimated and used for shadow detection.

Experiments on real images with our proposed error measure show for the proposed shadow removal method an error reduction of 50 percent compared to a reference algorithm. For foreground/background classification, an error reduction of 35 percent is obtained compared to no shadow removal.



(a) Intratuin

(b) PETS01-3TR1

(c) Schiphol

Figure 6.4: *Good/bad–plots. Shown is the convex hull of results with different parameter settings for both foreground/background classification and shadow removal.*

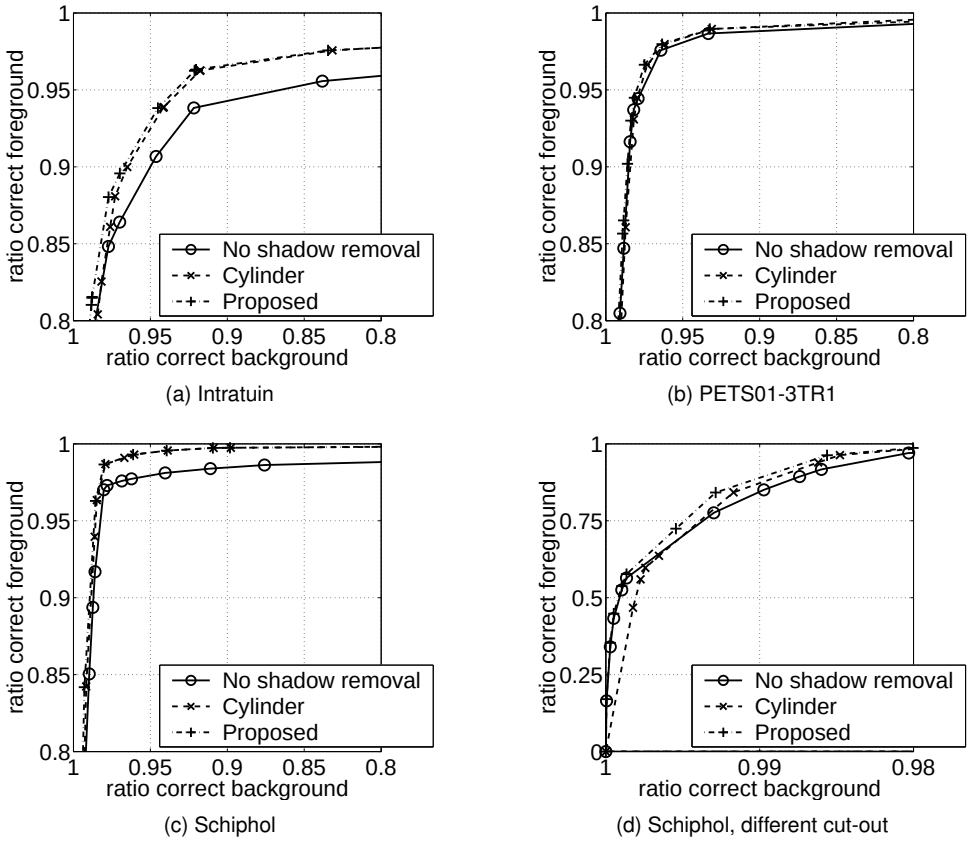


Figure 6.5: *Total–ROCs to show the effect of using shadow removal.*

Chapter 7 Object segmentation and tracking application

7.1 Introduction

Chapters 3 to 6 of this thesis introduce algorithms for parts of the object tracking application introduced in Section 1.2. In this chapter[1] these algorithms will be connected to form one coherent application. It will be experimentally evaluated using real image sequences.

For connecting the separate algorithms, the probabilistic framework introduced in Section 1.3 will be used. It integrates the separate (plug-in) algorithms for each of the levels of abstraction into a coherent visual surveillance application. Between levels of abstraction, minimum cost classification based on probabilities is used. Feedback is available to communicate higher level results to the lower levels of abstraction.

First, the framework introduced in Section 1.3 will be detailed in Section 7.2. Then in Section 7.3 our tracking application will be discussed. It will be evaluated in Section 7.4 and then conclusions will be presented in Section 7.5.

7.2 Components of the framework

The framework consists of a generic classification algorithm, application independent models, application knowledge and application dependent plug-in algorithms. First, the order of the different parts of the framework will be given. Then all components will be discussed. Their interactions will be described according to the flowchart for the blob and track levels of abstraction given in Figure 7.1.

In theory, the proposed framework is optimal, given that minimum cost classification is considered optimal, and the tracking application can be split in independent

[1] This chapter is based on [Withagen et al., 2005d].

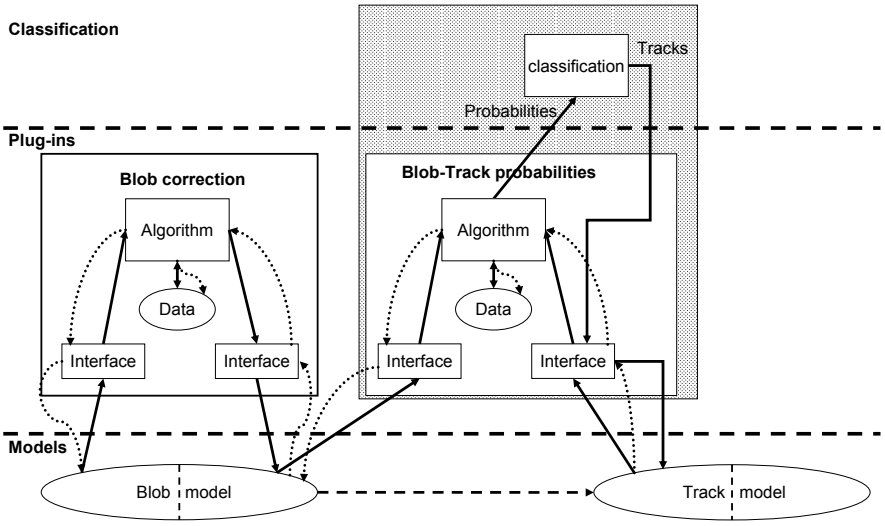


Figure 7.1: *Internal structure of modules. All communication with the models occurs through interface algorithms. Dotted arrows depict feedback, drawn arrows data.*

levels of abstraction. In practice however, classification costs are estimates, models are simplifications of the world and the division between levels of abstraction is sometimes difficult to determine.

7.2.1 Processing order

Figure 7.1 depicts one layer of abstraction of the framework. It will be used to describe the processing order of the framework. From the lower level of abstraction (or directly from the camera), data arrives in the left part of the lower level model (the blob model in the figure). The interface of the correction plug-in collects the data from the model and sends it to the correction algorithm, where the data is corrected. If necessary, data from the local data storage is used. The interface stores the data in the right side of the lower level model, see the figure.

The interface of the classification algorithm collects the corrected data from the model and sends it to the probabilities algorithm. This algorithm calculates probabilities, making use of local data if necessary. Probabilities are send to the classification algorithms, which calculates the lowest cost classification result. The classification result is used by the interface to update the higher level model.

After all levels of abstraction have processed the frame as described above, feedback is initiated. The higher level model (the track model in the figure) obtains the new description of the world from the higher level algorithms. The interface

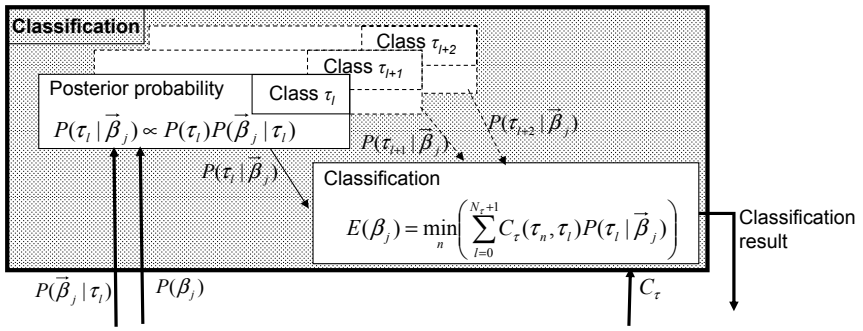


Figure 7.2: *Classification algorithm for blob-track classification. Based on the inputs: prior probabilities $P(\tau_l)$, conditional probabilities $P(\vec{\beta}_j|\tau_l)$ and costs C_τ posterior probabilities are calculated and minimum cost Bayes classification is performed.*

function of the probabilities plug-in gets this description of the world and sends it to the probabilities algorithm, which uses it to update the local data. Then the interface propagates the results to the lower level model. The interface of the lower level model gets the updated description at this level and makes it available for the correction algorithm. This algorithm updates its local data. Finally the left hand side of the model is updated.

7.2.2 Classification

As mentioned before, the framework uses minimum cost classification. This is based on three inputs. **Posterior probabilities** of the output classes calculated by the plug-in algorithm. **Prior probabilities** for the different output classes, which can be learned or given by the application. **Cost matrices** given by the application, defining the cost of a (wrong) classification to an output class. These inputs are provided by the probabilities module (for learned or calculated inputs) or by application knowledge (for user knowledge). Output of the classification algorithm is a new description of the world at the higher level. The interface to the model updates the higher level model with this knowledge. Figure 7.2 shows a flowchart of the classification algorithm.

For the blob-track example, all input blobs β_j are assigned to one of the following output track classes τ_l:

- τ_0 Background
- $\tau_1 \cdots \tau_{N_\tau}$ Any of the N_τ current tracks
- $\tau_{N_\tau+1}$ A new track

The Bayes decision rule is used to perform minimum cost **classification**:

$$E(\beta_j) = \min_n \Big(\sum_{l=0}^{N_\tau+1} C_\tau(\tau_n, \tau_l) P(\tau_l|\vec{\beta}_j) \Big) \ , \tag{7.1}$$

with E the total classification cost[2], $\vec{\beta}_j$ the feature vector of blob β_j and C_τ the cost matrix for classifying object τ_n as object τ_l.

Posterior probabilities $P(\tau_l|\vec{\beta}_j)$ are calculated using Bayes' rule

$$P(\tau_l|\vec{\beta}_j) = \frac{P(\tau_l)P(\vec{\beta}_j|\tau_l)}{P(\vec{\beta}_j)} \ , \tag{7.2}$$

with $P(\vec{\beta}_j) = \sum_l P(\tau_l)P(\vec{\beta}_j|\tau_l)$. As the denominator is equal for all output classes, it can be omitted for solving Equation 7.1.

The sum of prior probabilities $P(\tau_l)$ over all output classes must equal one

$$\sum_{l=0}^{N_\tau+1} P(\tau_l) = 1 \ . \tag{7.3}$$

C_τ and $P(\tau_l)$ depend on the application and are regarded input to the probabilistic framework. They can be given or learned during a training period.

7.2.3 Models

Models are a description of the state of the world at a given level of abstraction. The model stores data such as pixel values, blob location and shape and track history. This data is used by the plug-in algorithms.

Algorithm specific data, such as a mixture model for the background or a core histogram updated over time are stored in the plug-in.

7.2.4 Correction modules

Correction modules operate on one level of abstraction, so on one model. The use of such a plug-in requires to store two copies of the model, one before and one after correction. Interaction between these models is performed by the correction module.

The correction algorithms obtain their data from a single model. Interaction with the models occurs through an interface function. This function takes care of (simple) feature extraction, for example conversion of color space occurs here. The algorithm has its own data storage for algorithm specific data. This data is updated using feedback from the higher level models.

[2]Most pixel variables we use depend on time and image location. For clarity, we will omit these indices t, (x, y) throughout the chapter.

7.2.5 Probabilities modules

Probabilities modules operate between two models at different levels of abstraction. Input features are obtained from the lower level model and output classes from the higher level model. Output of this module are probabilities for the classification algorithm. After classification, the classification result is used to adapt the models description of the world.

The probabilities algorithms obtain their data from two models. Interaction with the models occurs through interface functions, similar to the correction modules. The algorithm also has its own data storage for algorithm specific data. This data is updated using feedback from the higher level models.

7.2.6 Application knowledge

The application defines what plug-in algorithms and models are used. It also provides values for parameters that cannot be learned and initial values for parameters that are learned. Because of the minimum cost classifier used, application knowledge can be introduced through prior probabilities and classification costs. These have a strong link to the real world, which makes them relatively easy to determine.

7.2.7 Updating and feedback

All models, whether it is a model of the background or a moving object, need to be updated to be robust to changes in the environment and the object such as weather changes or a changed view of a moving object. The classification result gives the class the data is assigned to, so also which model is updated with a given new measurement. For example, immediately after pixel-blob classification, the pixel-wise models of the background and the blobs could be updated. Alternatively, the updating can be performed after all levels of abstraction have been evaluated using the high level information available there. This latter approach is called feedback in this thesis.

The use of feedback allows for updating the models using higher order knowledge. However, it also introduces the risk of instability. Like every closed-loop system, stability requires attention. Stability is obtained using the following mechanisms

- **High level knowledge:** At a higher level more robust classifications can be obtained, because more independent measurements and features are available and it is easier to add application knowledge. Therefore, updating is postponed until a decision has been made at the higher level. This information from the higher levels is then used to update the models used in the lower level classification.
- **Multiple hypothesis:** It is important to realize that errors are inevitable. Early in the process, not all information is available to make error-free decisions.

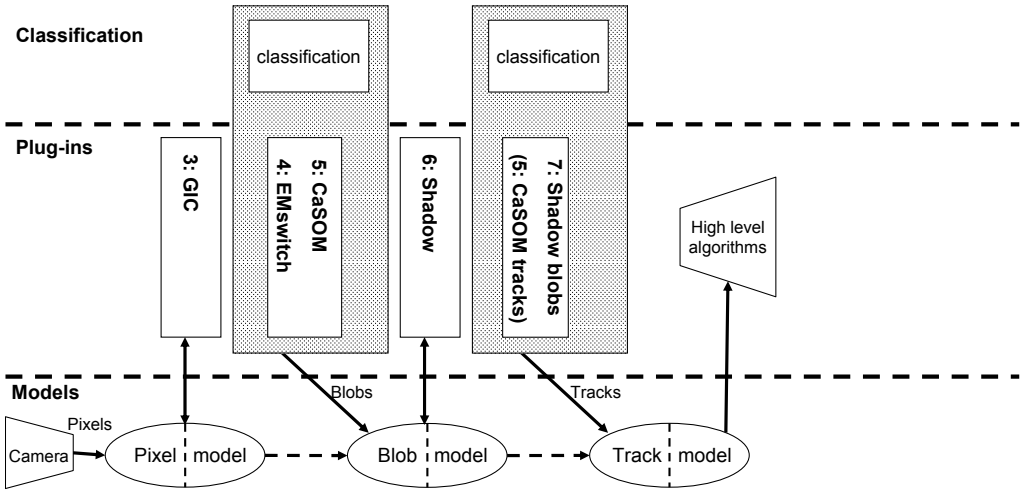


Figure 7.3: *Flowchart of our application. Numbers correspond to chapters in this thesis. As the CaSOM algorithm performs both calculation of pixel-blob probabilities and blob tracking, it is mentioned in both probabilities modules.*

Later in time, information of more frames will be available to obtain better classification. It is therefore important to be able to reconsider earlier decisions. Keeping track of multiple competing hypothesis facilitates this.

- **Slow updating:** Models are not simply replaced by new measurements. The new models are a mixture of the old model and the new measurement. An update speed defines how fast old measurements are forgotten.

7.3 Our tracking application

In this section we describe the plug-in algorithms used in our application: visual tracking of people in complex scenes, introduced in Section 1.2.

The application requires all objects to be tracked, and allows tracks to be split and merged. This is most efficiently solved using object tracking before object classification, see Chapter 2. This leads to the framework given in Figure 7.3. The different algorithms described in this thesis are used as plug-ins. The framework contains the following models:

- **Pixel model:** The initial pixel model contains for each pixel its RGB color value. The corrected pixel model contains for each pixel its RGB color value corrected for global intensity changes.

- **Blob model:** The initial blob model contains for each blob a set of pixels that belong to that blob. The corrected blob model contains for each blob a connected set of pixels that belong to that blob. An additional blob class ”shadow” is available for shadow pixels. All pixels are assigned to exactly one blob.
- **Track model:** The track model contains for each track a history of bounding box locations.

and the following modules:

- **Pixel correction module:** Global intensity correction described in Chapter 3.
- **Pixel-blob probabilities module:** EMswitch described in Chapter 4 for the background probabilities and CaSOM described in Chapter 5 for foreground probabilities.
- **Blob correction module:** Shadow correction described in Chapter 6.
- **Blob-track probabilities module:** Removal of shadow regions which will be described in Section 7.3.5. Track information is obtained from CaSOM in the Pixel-blob probabilities module.

Below the plug-in algorithms of this application will be described in detail.

Track correction and the track-object probabilities module are highly application dependent. For our application, these high levels are not important as we are only interested in suspicious trajectories. It is therefore not important that each track corresponds to a single real-world object. Our lower level plug-ins do not expect feedback from these levels. They will not be used in our application.

7.3.1 Application knowledge

The application not only defines which levels of abstraction and plug-in algorithms should be used, it also determines the classification costs. The cost matrices can become large when many output classes are possible. However, the number of unique numbers in these matrixes is often not that large. For most practical applications, the matrix given in Table 7.1 is sufficient. This matrix contains only three parameters: the cost of a missed detection a_ω, the cost of a false alarm b_ω and the cost of object confusion c_ω. That is nine parameters for a framework with three probabilities modules.

The costs between different levels of abstraction are often not independent. When these dependencies are exploited, even less parameters are needed. Classification costs depend on how easy errors from the lower level can be repaired at the higher level. We will use one-to-one correspondence between blobs and tracks, leading to equal costs for misclassifying a track τ and misclassifying a blob β. This cost is set equal to the total cost of misclassifying all pixels in the blob. We use

Table 7.1: *The cost matrix C_τ for blob-track classification. The cost matrix has vertically the ground truth classification and horizontally the used classification. a_τ denotes the cost of a missed detection, b_τ the cost of false alarm and c_τ the cost of confusing two tracks.*

$C(\tau_n, \tau_l)$	τ_0	τ_i	τ_j	$\tau_{N_\tau+1}$
τ_0	0	b_τ	b_τ	b_τ
τ_i	a_τ	0	c_τ	c_τ
τ_j	a_τ	c_τ	0	c_τ
$\tau_{N_\tau+1}$	a_τ	c_τ	c_τ	0

N_{pixels} for the number of pixels in the blob, assuming that each blob corresponds to one track and each track to one object. The relations between parameters for the different levels of abstraction then become:

$$1a_\omega = 1a_\tau = N_{\text{pixels}} a_\beta \tag{7.4}$$
$$1b_\omega = 1b_\tau = N_{\text{pixels}} b_\beta \tag{7.5}$$
$$1c_\omega = 1c_\tau = N_{\text{pixels}} c_\beta \ , \tag{7.6}$$

with a_β the cost of missed detection, b_β the cost of a false alarm, c_β the cost of a confusion between two blobs and ω an object. Actually, only the ratio between these parameters will suffice. We can therefore remove N_{pixels} and set one of the parameters equal to unity. So using this approach we only need to set two parameters. The value of these parameters should be given by the application, set by an operator or learned using training data.

Another option would be to set the costs for each blob equal, leading to lower costs for pixels that are part of large blobs. This favors classification to large blobs, resulting in less tracks. However, it is easier to combine tracks for obtaining one a real-live object than it is to split tracks to several separate real-live objects. Therefore, we prefer (more) smaller blobs, which is obtained using equal per-pixel costs.

7.3.2 Pixel correction module

Our application allows a static camera and a background that changes slowly compared to the motion of objects, similar to [Haritaoglu et al., 2000; Kim et al., 2003; Stauffer and Grimson, 2000]. Because of intended outdoor use and widespread use of affordable auto-gain cameras, we shall allow fast global changes in intensity. These are corrected in the pixel model using our global intensity correction described in Chapter 3.

7.3.3 Pixel-blob probabilities module

Pixels are the smallest building blocks of an object. Pixel x_i at position $\vec{p}_i$ has at time t a certain color $\vec{c}_i$. Additional features like depth, edge information etc. might also be available. We combine all features in the pixel feature vector $\vec{x}_i$.

Pixel-blob classification assigns all pixels to one of the possible blob classes β_j. A blob is defined as a group of connected pixels that have been classified to the same foreground class. The background is considered a special case here, because it can exist of multiple parts.

We will use separate models for the foreground and background, yielding separate probabilities for foreground and background for each pixel. These are used for minimum cost Bayes classification in the probabilistic framework.

The application allows us to use the slow change of both the background color (per pixel) and the color of moving objects (its color distribution). They both are modelled using an adaptive model.

The background is considered a special kind of blob, treated different from the other blobs. The background in our application can have several modes, for example when looking at vegetation and water surfaces. A multi-modal adaptive model of the background color sufficiently models such a background. With this model, background probabilities are calculated based on the pixel color. The multi hypothesis EMswitch approach introduced in Chapter 4 is utilized to update the model of the background. This allows the use of feedback and reconsidering decisions at later stages when more information is available.

For blob modelling our Core and Shell Object Model (CaSOM) given in Chapter 5 will be used. This object model allows arbitrary object shape as well as changes between subsequent frames. It is based on the assumption that the center of the object, the core, can be fully retrieved in the subsequent frame. This is the most stable part of the object and therefore used for tracking. A layer around this core, the shell, will be probabilistically classified based on pixel color, distance to the core and occlusion.

The CaSOM model is shared between blob level and track level. Therefore, for existing tracks, pixel-blob classification assigns pixels not only to a blob, but also to a track.

Probability calculation: Background probabilities $P(\vec{x}|\beta_0)$ are calculated from the Gaussian mixture model using Equation 4.1. Foreground probabilities $P(\vec{x}|\beta_j)$ are calculated from the CaSOM object models using Equation 5.6. Prior probabilities $P(\beta_j)$ are given by occlusion modelling, using Equation 5.12.

Updating the model: After feedback, EMswitch is used to update the background model for pixels classified as background. Pixels classified as shadow are regarded foreground in EMswitch. The object models will be updated in the blob-track probabilities module. These models are also not updated with shadow pixels.

7.3.4 Blob correction module

The blob correction module corrects for shadow pixels. When a pixel is shaded, it initially will be classified as foreground. It is important to remove it from the object **before** updating the object model. Falsely updating the object model with (shaded) background colors will eventually cause more background pixels to be assigned to this object, leading to inaccurate foreground/background segmentation. On the other hand, the background model should also not be updated, to prevent the background distribution from becoming very wide. The shadow detection algorithm described in Chapter 6 is used to remove shadow pixels.

A second task of the blob correction module is to make make sure all pixels are assigned to exactly one blob, and that all pixels assigned to one blob are connected. When a blob is split in several parts only the largest part is kept. All other parts of the blob are re-labelled background.

7.3.5 Blob-track probabilities module

Blob-track classification assigns blobs β_j to tracks τ_l, i.e. blobs tracked over time. The feature vector of the blob $\vec{\beta}_j$ contains the pixel features of all belonging pixels and additional information of the blob like shape and size.

The blob model and track model are shared, it was already discussed in the previous section. It consists of a color histogram and the CaSOM shape model. There is a one-to-one correspondence between blobs and tracks. Therefore, only two classifications are possible for each blob. It is either assigned to the background or to a track. When assigned to a track, a new blob will start a new track. An existing blob, i.e. a blob of which the pixels were assigned to an existing track, is assigned to the track the pixels were assigned to.

At this level, we reconsider pixel-blob classification of all new blobs, making use of features from all pixels assigned to the blob. We intend to remove shadow or highlight regions and other regions that depict background but are erroneously detected as foreground blob.

We base the track probability on comparing the blob histogram to that of an intensity-corrected version of the background. A more advanced method would be the approach of [Horprasert et al., 1999] based on color or to add gradient information [Javed and Shah, 2002]. However, this also would make the algorithm more computationally expensive.

Probability calculation: The probability that a blob is classified as background is based on the normalized blob histogram H_j^{β} and the histogram H_j^0 of an intensity-corrected version of the background image at the position of the blob. This background image is created from the pixel-wise EM model, using for each pixel the mean of the kernel with the highest prior probability. The background color of the pixels assigned to the blob are corrected such that the average intensity per color band is equal to the average intensity per color band of the blob. After correction, both histograms will be similar for shadow regions, resulting in a high background

probability. The prior probability that an outlier occurs is given by P_{outlier} and is assumed uniformly distributed over the histogram.

The histograms of the blob H_j^β and the intensity-corrected background H_j^0 are compared using the Resistor distance [Johnson and Sinanovic, 2001], which is a symmetrical version of the Kullback-Leibler distance [Gray, 1990; Guiasu, 1977] and is given by

$$R(H_j^0, H_j^\beta) = \frac{1}{\frac{1}{\text{KL}(H_j^0, H_j^\beta)} + \frac{1}{\text{KL}(H_j^\beta, H_j^0)}} , \tag{7.7}$$

with KL the Kullback-Leibler distance for two discrete probability density distributions:

$$\text{KL}(H_j^\beta, H_j^0) = \sum_{n=1}^{N_{\text{bin}}} h_j^\beta(n) \log_2 \frac{h_j^\beta(n)}{h_j^0(n)} , \tag{7.8}$$

with $h_j^0(n)$ and $h_j^\beta(n)$ the n^{th} bin of histograms H_j^0 and H_j^β respectively. Each bin has a value $h_j^0(n) \geq P_{\text{outlier}}$.

The Resistor distance is mapped to a probability using the half-normal distribution [Weisstein, 2005]:

$$P(\vec{\beta}_j|\tau_0) = \exp(-R(H_j^0, H_j^\beta)^2\theta^2/\pi) , \tag{7.9}$$

with θ a tuning parameter. A blob can only be assigned to a single track. Because the models for blob and track are shared, we will use for the track probabilities:

$$P(\vec{\beta}_j|\tau_l) = \begin{cases} 1 & \text{for } j = l \\ 0 & \text{otherwise} . \end{cases} \tag{7.10}$$

Updating the model: Using feedback, the track models are updated. The track model stores the position of the track, the shape of the blob and a color histogram, of which updating is described in Chapter 5.

7.4 Experimental evaluation

We will evaluate the proposed application using real image sequences. We will primarily focus on the pixel-blob classification. As it is the lowest level, it is essential to get it right. Errors at the lowest level, specifically missed detections, cannot be corrected at higher levels. Particularly, we investigate the results after feedback in the blob model using evaluation criteria given in §7.4.1. To demonstrate the other modules, results in the track model will also be given.

In §7.4.2 the image sequences used for evaluation and the used parameter settings are given. Then in §7.4.3 the experimental results are given and they are discussed in §7.4.4.

7.4.1 Evaluation criteria

Evaluation will be performed at two points in the framework. Namely, on the output of the blob correction module and on the output of blob-track classification, both after feedback has been performed. This way, the first four modules are evaluated, and at the same time the application as a whole is evaluated. The four evaluated modules are pixel correction, pixel-blob probabilities, blob correction and blob-track probabilities. These modules are generally applicable, while the track-object probabilities module is more application-specific. The evaluation criteria are defined in relation to our application but are widely applicable in this domain.

Blob evaluation

Pixel-blob classification and blob correction are evaluated using the ROC and the surface above the ROC as was described in Chapter 3.

Track evaluation

In order to demonstrate the entire application, a second evaluation criterion is used. This criterion is the number of false alarm and missed tracks.

The tracks consist of a number of trajectory points, each with the image coordinates at a certain frame number. For evaluation we use an ROC of the ratios of false and missed trajectory points.

Calculation of the ratios of false and missed trajectory points is performed in two steps. First, we calculate the overlap in time between each combination of experimental and ground truth trajectories:

$$t^{\text{begin}}_{\text{overlap}}(m,n) = \max(t_{\text{GT},m}(\text{first}), t_{\text{EXP},n}(\text{first})) \quad \text{and} \tag{7.11}$$

$$t^{\text{end}}_{\text{overlap}}(m,n) = \min(t_{\text{GT},m}(\text{last}), t_{\text{EXP},n}(\text{last})) \ , \tag{7.12}$$

where $t_{\text{GT},m}(\text{last})$ is the last frame the m^{th} ground truth track exists and $t_{\text{EXP},n}(\text{first})$ is the first frame the n^{th} experimental track exists. For this overlap period we calculate the average city-block distance D_{match} between the two tracks

$$D_{\text{match}}(m,n) = \frac{1}{t^{\text{end}}_{\text{overlap}}(m,n) + 1 - t^{\text{begin}}_{\text{overlap}}(m,n)} \sum_{t=t^{\text{begin}}_{\text{overlap}}(m,n)}^{t^{\text{end}}_{\text{overlap}}(m,n)} |x_{\text{EXP},n}(t) - x_{\text{GT},m}(t)| + |y_{\text{EXP},n}(t) - y_{\text{GT},m}(t)| \ , \tag{7.13}$$

with $y_{\text{EXP},n}(t)$ the y-location of the n^{th} experimental trajectory point at time t.

A combination of trajectories with a distance $D_{\text{match}}(m,n) < D_{\max}$ is called a match. In the remainder of the evaluation only matching combinations are considered. Note that multiple ground truth trajectories may correspond to one experimental trajectory and vice versa. This happens for example when several persons walk in a group. The ground truth will describe separate tracks, while our algorithm might generate only one track, for the entire group of people. For our application this poses no problem as we are interested in suspicious trajectories. Therefore, an experimental trajectory is allowed to match with several ground truth trajectories and vice versa.

Second step of the blob-track evaluation is the actual calculation of the ratios of missed and false trajectory points. We define the ratio $r_{\text{no false alarm}}$ as the number of trajectory points M_{EXP} of all experimental trajectories that match to one of the ground truth trajectories divided by the total number of experimental trajectory points T_{EXP}:

$$r_{\text{no false alarm}} = \frac{M_{\text{EXP}}}{T_{\text{EXP}}} \,. \tag{7.14}$$

For the fraction $r_{\text{not missed}}$ of trajectory points we calculate the number of ground truth trajectory points corresponding to any of the experimental trajectory points. We divide by the total length of all ground truth trajectories:

$$r_{\text{not missed}} = \frac{M_{\text{GT}}}{T_{\text{GT}}} \,. \tag{7.15}$$

The numbers of total trajectory points are given by

$$T_{\text{GT}} = \sum_{m=1}^{N_{\text{GT}}} (1 + t_{\text{GT},m}(\text{last}) - t_{\text{GT},m}(\text{first})) \quad \text{and} \tag{7.16}$$

$$T_{\text{EXP}} = \sum_{n=1}^{N_{\text{EXP}}} (1 + t_{\text{EXP},n}(\text{last}) - t_{\text{EXP},n}(\text{first})) \,, \tag{7.17}$$

for the ground truth and experimental trajectories respectively. N_{GT} and N_{EXP} are the number of experimental and ground truth trajectories.

The numbers of matching trajectory points are given by

$$M_{\text{GT}} = \sum_{m=1}^{N_{\text{GT}}} \max_{n \in n_{\text{match}}(m)} (t^{\text{end}}_{\text{overlap}}(m,n) + 1 - t^{\text{begin}}_{\text{overlap}}(m,n)) \quad \text{and} \tag{7.18}$$

$$M_{\text{EXP}} = \sum_{n=1}^{N_{\text{EXP}}} \max_{m \in m_{\text{match}}(n)} (t^{\text{end}}_{\text{overlap}}(m,n) + 1 - t^{\text{begin}}_{\text{overlap}}(m,n)) \,, \tag{7.19}$$

with $n_{\text{match}}(m)$ and $m_{\text{match}}(n)$ the sets of matching trajectories.

When an object is missed in one frame and re-detected in the next, the trajectory of the object is described by two separate tracks. These tracks should be

(a) PETS01-1TE1 1000 (b) PETS04 750

Figure 7.4: *A frame of each of the sequences used for track evaluation.*

connected using track-object classification. As we currently do not use track-object classification, tracks will be connected in the evaluation algorithm. Before calculation of M_{EXP} the set of experimental trajectories is extended by adding combinations of trajectories. Tracks are combined when either the first or last position lies within D_{max} pixels and within D_{max} frames from another experimental trajectory point, using the city-block distance metric as in Equation 7.13. This way, tracks that were missed for a few frames are also taken into account.

Choosing D_{max}: The value of D_{max} has a strong influence on the results. The larger this value, the lower the error ratio be, but it will also be less selective. It is therefore important to determine its value with the application in mind. For our application, the trajectory of people walking next to each other should match to the trajectory of all people together. For example, consider the PETS01-1TE1 image sequence used for track evaluation. The city-block distance between the center of two people walking next to each other (frame 825) is approximately 34 pixels. We use a four times sub-sampled version for our experiments. Therefore, a value of $D_{max} = 10$ seems a very reasonable value to be used for this image sequence.

7.4.2 Test data and parameter settings

The three image sequences given in Chapter 3 will be used for blob evaluation, see also Appendix B. For track evaluation, two image sequences for which ground truth trajectory was available were used: PETS01-TE1 and PETS04, see also Figure 7.4 and Appendix B.

Parameter settings

Several parameters can be adjusted, both in the model and in the plug-in algorithms. In this subsection, values used for the experiments will be provided. The parameters can be divided into three classes:

- Parameters that are easy to determine from physical parameters.
- Parameters for which the application is not sensitive.
- Parameters that should be tuned.

Examples from the first class of parameters are the blob size N_{pixels}, the time an object may be static N_{Fmax} and the maximum distance between corresponding tracks D_{max}. The second class includes the outlier removal threshold T_{outlier}, the background update speed u_{B} and the number of kernels N_k. Only small improvement is expected by tuning these parameters. Equal values were used for all image sequences, indicating that the algorithms are not very sensitive to these parameters. These parameters were set to a suitable value for the image sequences used and not further optimized or tuned. Below, all parameter values for parameters that are not tuned will be given.

Tuning of the remaining parameters has been performed by evaluating the total application at different combinations of parameter settings. The result of each set of parameter settings is a point in the ROC graph. Choosing the parameter values to be evaluated has been performed manually, based on the performance of previous parameter setting a new setting was selected.

In the models: Adjustable parameters are cost matrices and prior probabilities. To set the prior probabilities and cost matrices we will take the approach given in §7.3.1. We chose $a_\beta = 1$ and varied b_β and c_β. All prior probabilities were set equal, as they will be multiplied by the costs, which are varied for creating an ROC.

In the pixel model: The images are corrected for global intensity differences by the MofQ algorithm proposed in Chapter 3. The first frame is used as reference image and outliers are removed on a statistical basis.

In the pixel-blob probabilities module: We do not use order information, so we assume all orderings equally probable, see Chapter 4. For the background model we used the RGB color channels. We used an update speed of $u_{\text{B}} = 0.001$ and $N_k = 5$ kernels. For occlusion modelling, a blob size of $N_{\text{pixels}} = 200$ pixels was used. Other parameters are equal to those used in Chapter 4.

The background model uses 16^3 RGB histogram bins, a core distance $d_{\text{max}} = 6$ pixels, and a search area for the core object of 11×11 pixels. The histogram is updated with an update speed $u_{\text{F}} = 0.1$. Other parameters are equal to those used in Chapter 5

Shadow was removed using the same parameter settings as those used in Chapter 6.

In the blob-track probabilities module: Two parameters are used: $P_{\text{outlier}} = 0.001$ for each bin was fixed and θ was optimized.

Table 7.2: *Results for blob evaluation. Given is the surface above the ROC in percentages (lower is better).*

Data:	Intratuin	Schiphol	PETS01-3TR1	Average
EMStauffer	3.95	2.59	2.28	2.94
EMStauffer+GIC	3.68	1.32	0.78	1.93
EMswitch+GIC	2.81	0.99	0.79	1.53
CaSOM+EMswitch+GIC	1.69	0.80	0.76	1.08
Framework	0.19	0.40	0.20	0.26

In the experimental evaluation: For evaluation, one parameter needs to be specified. The maximal average distance between two tracks is set to $D_{\text{max}} = 10$.

7.4.3 Experimental results

Results will be given for:

- EMStauffer from §4.2.2
- EMStauffer+GIC from Chapter 3
- EMswitch+GIC from Chapter 4
- CaSOM+EMswitch+GIC from Chapter 5
- Framework: the entire application with feedback, shadow removal, CaSOM, EMswitch and GIC, proposed in this chapter

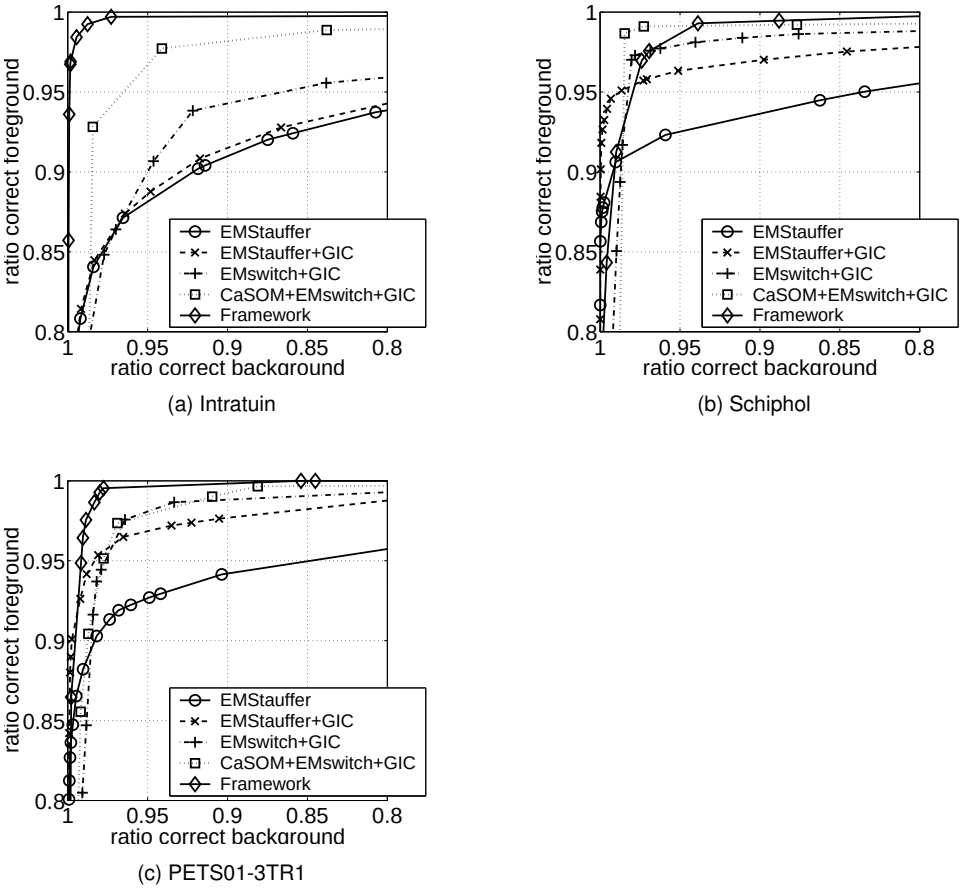
The framework adds shadow removal, blob-track classification and feedback to CaSOM+EMswitch+GIC. The feedback used contains shadow pixels and shadow regions.

Results will be compared to the EMStauffer reference algorithm. For a description of the other results see the corresponding chapters.

Blob evaluation

In Figure 7.5 the ROC curves for blob evaluation are shown. It is evident that for each of the sequences used the framework results are significantly better than those of the reference method. Even though considerable effort is used to find optimal settings of the two parameters of the reference method. The two parameters tuned for the reference method are the threshold and update speed.

Table 7.2 shows the surface above the ROC as percentage of the total area of the ROC. These results show the amount of error reduction compared to the reference algorithm. On average, a factor 11 is obtained.



(a) Intratuin (b) Schiphol

(c) PETS01-3TR1

Figure 7.5: *Framework pixel-blob results.*

Track evaluation

For the track evaluation we do not have a reference method. We do however use two publicly available image sequences that are often used for performance evaluation of tracking algorithms. Also, ground truth is publicly available for these sequences.

In Figure 7.6 the results for track evaluation are given. The results for the PETS04 data is perfect, all track-points were detected without false alarms.

The results from the PETS01-1TE1 data are not perfect. But then, this is quite a difficult image sequence for object tracking. People walk together and later split, cars stop moving, and later start moving in the opposite direction. Also, a lot of occlusions are present in the image sequence.

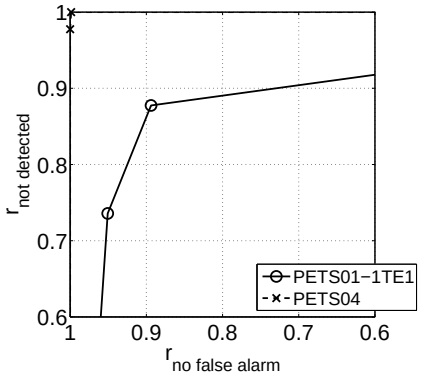


Figure 7.6: *Framework blob-track results.*

Even for this data, 90% detection with little more then 10% false alarms is obtained. We consider this not a bad result as we use a very simple plug-in algorithm for blob-track classification and no track-object classification. Results are expected to improve by feedback from a track-object classification. Note also that the PETS01-1TE1 image sequence was processed at one fourth of the original resolution. This probably has a negative impact on the results.

7.4.4 Discussion

In Figure 7.5 and Table 7.2 five algorithms for foreground/background segmentation are compared. Each consecutive algorithm increases in complexity and should increase performance.

The first improvement is obtained by adding global intensity correction to the EMStauffer reference algorithm. This significantly increases performance for the Schiphol and PETS01-3TR1 sequences. As expected, the performance increase for the Intratuin sequence is small, as no changes in intensity are present there. Besides the classification results, also the robustness for setting the update speed increases, see the PIPE results in Chapter 3.

Using the EMswitch model of the background gives the second improvement. The improvement is a shift up of the ROC. This algorithm is more sensitive to shadows, resulting in a small shift of the ROC to the right. Only for the PETS01-3TR1 sequence the overall result is not improved.

Adding the CaSOM foreground model improves results even further, although marginally for PETS01-3TR1. This algorithm again causes a shift up of the ROC.

The final improvement, that is denoted framework, is caused by three effects: the use of feedback, adding shadow removal, and adding blob-track classification, which essentially removes shadow regions not connected to moving objects.

Shadow causes the performance increase on the Intratuin and Schiphol sequences. Overall, the framework performs best for the Schiphol sequence. However, at most locations of the ROC one of our other methods locally performs better. This expected to be caused by still insufficient shadow removal for this image sequence. Main difference with the other sequences is the indoor (fluorescent) illumination. Knowing the application allows to choose the best algorithm from the ROC.

For the PETS01-3TR1 sequence blob-track classification in conjunction with feedback plays an important role. Moving clouds cause large shadow regions, which are effectively removed by this algorithm.

7.5 Conclusions

In this chapter, we composed a complete example application for visual surveillance using the separate algorithms introduced in this thesis. Integrations has been performed using our probabilistic framework. We demonstrated the application for segmenting and tracking people in a complex scene. The application has been evaluated on four individual modules inside the framework. Results therefore depend on both the plug-in algorithms and the integration of the framework.

Evaluation of the first three modules, used to perform pixel correction, pixel classification and blob correction, shows an average decrease in error of more than a factor eleven compared to the Stauffer reference algorithm. Only for the Schiphol image sequence the framework is not always the best method. This is assumed to be caused by the indoor setting with fluorescent lighting.

The fourth module demonstrated perfect results on a PETS 2004 sequence and good results on a PETS 2001 sequence.

Chapter 8 Summary, conclusions and future work

Eureka!
(Archimedes)

This thesis discusses detection and tracking algorithms for visual surveillance applications. The results obtained in the thesis are discussed in this chapter. It ends with an answer to the central questing given in Chapter 1.

Global intensity correction in dynamic scenes

Changing global intensity is a real problem in algorithms that model a static background. The proposed MofQ algorithm is both effective and computationally efficient. It effectively corrects for global intensity differences in dynamic scenes. An accuracy improvement of a factor ten is obtained on simulated images with changes in intensity. Used as pre-processing for background modelling, an error reduction of a factor two to three is obtained on real image sequences. An important result is that the use of global intensity correction makes the background modelling algorithm more robust for setting the update speed parameter. Using the introduced Parameter Invariant Performance Evaluation (PIPE) measure, an error reduction of two to four is demonstrated.

The MofQ algorithm performs significantly better than a weighted least squares algorithm. The weighted least squares algorithm is only optimal when its assumptions are fulfilled in practice. The neglection of additive noise and the presence of outliers violate this in real life. On the other hand, experiments on simulated images show a correction accuracy that is almost optimal. The limitation of the MofQ algorithm lies in the percentage of background pixels. The assumed quatre of foreground pixels lies well within its working conditions.

The approach proposed assumes that the intensity of all color bands changes equally. This is not always true, for example in the case of cameras with automatic white balance.

Instead of using a correction algorithm for differences in global intensity, the use of normalized color or an adaptive background model is often used. The first has the important disadvantage that the discrimination based on intensity is lost, leading do missed object. The use of an adaptive background model is recommended, but using it for global changes in intensity requires a high update speed. This causes slowly moving objects to be lost.

Although used here for global intensity differences, the MofQ algorithm can also be used for more local intensity effects. For example, the quotient image calculated by MofQ is also the first step towards shadow detection, discussed later in this chapter. Effects of moving clouds, resulting in a partly brighter and partly darker scene, are not addressed in this thesis. Using a grouping algorithm on this quotient image before calculating the median allows estimating different intensity changes within the image.

Many algorithms besides background modelling are based on constant image brightness. For applications that use images recorded with a camera described by the camera model used, and providing images for which the scene content is equal for a large part of the image, the algorithm is also useful. As this is beyond the scope of this thesis, it has not been evaluated, but good performance is expected.

Shadow detection using a physical basis

Shadow detection is an important challenge in object tracking applications. This thesis introduces a shadow detection algorithm based on the quotient image calculated by global intensity correction. The proposed algorithm predicts the color shift caused by unequally colored light sources.

We introduce a performance measure designed to measure shadow correction accuracy in conjunction with foreground/background classification. The proposed algorithm gives an average improvement of this measure of a factor ten compared to a cylinder approach from literature. In conjunction with foreground/background classification the improvement is smaller, a factor 1.5 compared to no shadow removal and a factor 1.2 compared to the cylinder approach.

The shadow correction algorithm has more potential than is exploited in this thesis. The proposed algorithm has been evaluated on a per-frame basis. It would be better to learn the color of the light sources over time, leading to a less computationally complex and a more stable algorithm.

Dynamic background estimation

The EMswitch background modelling algorithm proposed in this thesis shows an increase in both classification accuracy and robustness to parameter settings compared to a standard approach. Slowly moving real-world objects are no longer unintendedly updated into the background model and foreground/background classification is based on probabilities.

The background has been modelled by a mixture of Gaussian kernels with diagonal covariance matrices. This is a trade-off between model complexity and prior knowledge on the one hand and computational efficiency and training time on the other. The Gaussian kernel with diagonal covariance matrix is able to describe a limited amount of scene deviation using only three variables.

More prior knowledge or assuming a more simple model of the background would allow one or even zero parameters to describe the kernel. This is computationally efficient and the number of frames necessary to train the model is minimized, but only when the background can be described by a single kernel. In many cases the kernel will not accurately describe the background and several kernels are required to model the background. More kernels are necessary in the mixture, leading to more computations and longer training times.

The use of more complex kernels allows to describe more complex situations using a single kernel. But more computations and a longer training period are necessary. In many practical situations such a water surface, leaves moving in the wind and color edges in combination with a slightly shaking camera, multiple kernels are still necessary.

Instead of Gaussian kernels, other models such as a particle filter or a histogram could be used. However, the Gaussian mixture model is efficient in both computational complexity and memory compared to these alternatives.

EMswitch allows the update of a Gaussian mixture model of only the background using a multi-state approach. A number of consecutive frames with equal classification are necessary before a pixels state is changed from background to foreground or vice versa. This leads to a small delay in model updates. The use of feedback from the higher levels would allow the use of all pixels in a blob, resulting in faster decisions. This could be exploited by allowing higher level algorithms to adapt the internal state and the internal counters of EMswitch. On the other hand, EMswitch is used as a safety measure to prevent wrong updates to the background. As also higher level algorithms make errors, we choose for the extra robustness to classification errors, with the small disadvantage of a few frames delay in background updating.

Probabilistic model of non-rigid objects

The EMswitch background model can be fully exploited using a model of tracked objects from which foreground probabilities are obtained. The CaSOM model introduced in this thesis is such a model. The combination of EMswitch and CaSOM realises an average error reduction of almost a factor two.

An important feature of CaSOM is that once real-world objects are detected, they are not easily lost. A higher level algorithm is used in our application to prevent accumulating falsely detected objects.

There are some problems with the CaSOM model. Small tracked objects are difficult to track, as there are not enough pixels to learn the probability density distribution of the object color, and the core becomes small, making template matching inaccurate. For larger tracked objects, the assumption that the color distribution of the core is representative for the entire tracked object is not always valid. This can cause the hair and shoes of pedestrians to be undetected. On the other hand, no alternative approach is available solving both these problems. Small objects require a simple model that can be trained using a small amount of data. In order to introduce spatial information, a model such as a correlogram can be used, which is more complex. Alternative models such as a mixture model and a particle filter have approximately the same complexity. These alternatives will therefore have the same problems

Small tracked objects could be modelled by learning the color model over time, or using a different algorithm specifically designed for small tracked objects. When they grow, the object model can be converted to the CaSOM model. The use of feedback from object classification can aid in preventing small tracked objects. Tracked objects belonging to one real-world object can be combined, resulting in larger tracked objects. On the other hand, it could also be an approach to limit tracked objects to regions with equal color or color pattern. A person will then be described by a combination of separate tracked objects for his head, shirt, hands, trousers and shoes. Combination of object parts can be done at the track level or at a higher level of abstraction.

Tracking framework including feedback

In order to create a consistent, robust application from the individual algorithms proposed in this thesis, a framework is given. The application created with this framework achieved an average foreground/background classification error reduction of a factor eleven compared to a popular foreground/background classification algorithm. Partly, this reduction is obtained by MofQ, EMswitch and CaSOM. Together, they obtain an error reduction of almost a factor three. Integrating them in the framework, adding shadow removal and adding higher level removal of false detections gives errors that are again a factor four lower.

An important addition of the framework is feedback. This allows to update lower level models using the best available description of the world, which exists at the highest level of abstraction. The application we used has only two levels of abstraction, with at the highest level a simple algorithm. Limited qualitative evaluation shows the use of feedback significantly improves performance.

The framework has been designed to be flexible. The use of a structured integration of plug-in algorithms with shared models and minimum cost classification should allow to replace one plug-in algorithm without altering the remainder of the application. This reduces the effort of adapting an application to a changed environment or task by replacing one or more plug-ins with a plug-in for the same level of abstraction. Due to time limitations, this aspect of the framework has not been evaluated.

Answer to the central question

The central question of this thesis was: Can we develop an algorithm for robustly and accurately detecting and segmenting moving objects in surveillance scenes? Below, the different aspects of the question will be discussed.

First of all, we can ”develop an algorithm for detecting and segmenting moving objects”. Several algorithms existed even before our research started. It is the second aspect: ”surveillance scenes”, that makes the problem difficult. Images from typical surveillance scenes often contain slowly moving people, parking cars, changing illumination conditions and shadows. For such scenes, our research added to the state of the art, for example by introducing an algorithm for global intensity correction.

Third aspect of the question was ”robust”. This is again connected to the real-world scenes. The use of separate background and foreground models solves the problem of slowly moving people and parking cars. Feedback in the framework allows the higher level processing to be integrated efficiently, taking care of false alarms.

Fourth and last aspect of the question was ”accurate”. Our experiments show a significant accuracy improvement, specifically due to separate models in combination with minimum cost classification and accurate shadow detection. Whether this makes it accurate enough is a more difficult question. It probably is accu-

rate enough in the less difficult surveillance scenes, where not too many real-world objects appear simultaneously. Difficult scenes, with high people densities for example, need more effort on higher level algorithms such as object tracking and classification.

Future research

The work described in this thesis is a step in the direction of an application aiding operators in their visual surveillance tasks. However, the problem is certainly not solved. The work gives a solid basis of a tracking application. The next step would be the higher level algorithms for object tracking and recognition of suspicious behavior. Occlusion, object classification and re-appearance of real-world objects are important subjects, as are activity recognition and trajectory analysis.

The next major step would be the use of more and different sensors. Several cameras can be used to expand the area under surveillance, microphones can give valuable cues of suspicious events and stereo cameras or radar can be used to increase depth resolution and aid with pose estimation. Drawback is the increase in cost and complexity.

The algorithms proposed in this thesis are limited to the use in areas with low or average people densities. As people start to form a crowd, other techniques should be used. Crowds are a typical scene for criminal behavior, therefore a visual surveillance application should be equipped with algorithms that can aid operators in monitoring crowds.

Appendix A CCD characterization for a range of color cameras

A.1 Introduction

CCD cameras are widely used for computer vision and image processing applications. For many applications it is important to have a model of the imaging process and temporal image noise. For example for the correction of temporal changes in intensity [Kamikura et al., 1998], illumination-invariant optical flow computation [Altunbasak et al., 2003] or object detection [Ohta, 2001; Xie et al., 2004].

Often a model is used that fits the applications needs. However, little work is reported on the accuracy of these models for a specific camera, and the importance of the different components of the model. In this appendix[1] we evaluate a general camera model for different camera types, ranging from a high end 10–bit digital camera to a consumer webcam.

To model all possible effects of the CCD, we use the general model of a CCD camera introduced in [Healey and Kondepudy, 1994]. In addition to that model, most modern cameras use some kind of gamma adjustment to map the image in the available quantization range for obtaining a better looking image. Therefore we add gamma correction to this model. We describe experiments to evaluate this model. The experiments presented verify the model and determine the model parameters for all cameras. The experiments are repeated for different scene contents to verify the model at different settings of camera gain and shutter time.

Besides verification of the model we show the effect of the individual contributions in the model to allow for simplifications. In particular, we evaluate:

- whether the gamma function in the camera is sufficiently accurately described by our model,
- what the effect of the dark current is,

[1] This appendix is based on [Withagen et al., 2005a,b].

- what the noise distribution is.

In the remainder of this appendix, first, a (theoretical) model of a CCD camera is discussed in Section A.2. In Section A.3 we describe the measurement setup and give the results. These results lead to simplifications of the model given in Section A.4. Finally, conclusions are presented in Section A.5.

A.2 Theoretical model of a CCD camera

Healey [Healey and Kondepudy, 1994] describes the following model for a single pixel recorded at time t using a CCD camera:

$$i_t = g_t(i_0 + \mu_{DC} + N_S + N_R) + N_Q \ , \tag{A.1}$$

with g_t the camera gain, i_0 the true scene intensity and i_t the measured image intensity for a given color band. The following noise contributions are present[2]: the dark current μ_{DC} is an offset, constant over time. The shot noise N_S has a Poisson distribution with $\mu_S = 0$ and σ_S depending on i_0. The readout noise N_R has a Gaussian distribution: $\mu_R = 0$, σ_R constant. The quantization noise N_Q has a uniform distribution $U(-\frac{q}{2}, \frac{q}{2})$ with q the smallest step in pixel value.

There are three ways to control the global image intensity, we will use the term apparent gain for their joint effect. It can be controlled using the camera gain, camera shutter time or lens iris. All can be fixed (manual control) or automatically adapted to the scene (automatic gain/shutter/iris control). For the CCD, iris or shutter control causes a change of image intensity i_0 in Equation A.1. We model this by parameter h_t:

$$i_t = g_t(h_t(i_0 + \mu_{DC}) + N_S + N_R) + N_Q \ . \tag{A.2}$$

Additionally, most cameras apply a gamma adjustment to map the range of intensity values from the CCD to the available output range. Assuming it is implemented in the camera electronics just before digitization, Equation A.2 changes to

$$i_t = g_t^{\gamma}\left(h_t(i_0 + \mu_{DC}) + N_S + N_R\right)^{\gamma} + N_Q \ , \tag{A.3}$$

with γ the gamma value which is assumed to be time constant and equal for all pixels.

A.3 Measurements

For verification of the model given above, measurements were performed with a range of different camera types. The cameras used are listed in Table A.1. The following measurements will be described. For each camera the gamma will be estimated using a log-log plot of the pixel value against the true (photon-counted)

[2] We use μ_x for the mean of x and σ_x for its standard deviation.

Table A.1: *The cameras used to verify the CCD model.*

Camera	Description	Intensity control
JAI CV-M7+CL	High end 10-bit digital camera with Bayer filter	Fixed
Siemens C810	Digital computer vision color camera	Automatic
JVC GR-DVL-157	Digital consumer color camcorder	Automatic
Philips PCVC680K	Color webcam	Automatic

intensity value. We measure the dark current by plotting the gamma-corrected pixel intensity against the true intensity. This gives a straight line that intersects the pixel intensity axes at the dark current. The multiplicative noise, i.e. the amount of noise depends on the amount of light reaching the CCD, and additive noise are estimated from a plot of the per-pixel variance against its mean. This gives a straight line of which the offset and the slope give the additive and multiplicative noise respectively. Difference in the slope of this line for different illumination conditions shows the kind of apparent gain correction used by the camera.

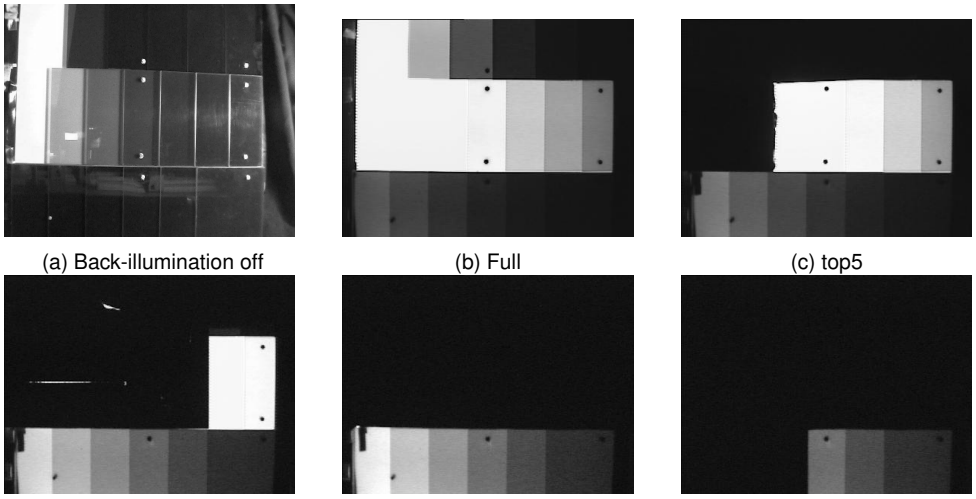
As measurement object we used a half-transparent plate which is homogeneously back-illuminated. In front of this plate are layers of gray and brown filters, see Figure A.1(a). There are eighteen different sections, zero to five layers of the gray filter on the top row and zero to thirteen layers of the brown filter on the center and bottom row. The illumination from each of the thirteen brown sections decreases according to $1/n$, with n the number of layers. This has been verified using a photon counter.

Imagery depicting this object was recorded with all cameras. To estimate the temporal mean and variance, sequences of 100 images were recorded. For investigating the automatic apparent gain control, we recorded the measurement object with different brighter parts of the object covered. This changed the covered part of the scene to black, while leaving the remainder of the scene unchanged, see Figure A.1. This will activate the automatic apparent gain control while leaving some sections to measure on.

A.3.1 Gamma estimation

For each sequence of 100 images the standard deviation and average of one (the red color) channel were calculated, both per pixel, and for the sections with equal intensity as a whole. To ignore saturation effects pixel with a value in the top 10% and bottom 10% of the intensity range are not taken into account.

The gamma is estimated from a log-log plot of the average intensity per section against the true (photon counter measured) intensity of that section. All data points should lie on a straight line (at least for points with intensity much greater than the



(a) Back-illumination off (b) Full (c) top5

(d) top2 (e) low6 (f) low3

Figure A.1: *Different sets of data are created by covering part the measurement object. Shown are the measuring object and a frame from each data set from the Siemens camera. The sets are: Full, the entire object is visible; Top5, the five darkest sections of the middle row and the entire bottom row are visible; Top2, the two darkest sections of the middle row and the entire bottom row are visible; Low6, only the six sections at the bottom are visible; Low3, only the darkest three sections are visible.*

dark current) with a slope equal to gamma. For the actual results see the figures on the top in Figures A.2 and A.3. The estimated gammas for all cameras are given in Table A.2. As expected, all cameras have a gamma smaller than one and for most cameras the individual data points are close to a straight line, and the lines for different measurements are approximately parallel, demonstrating that gamma is a constant. Therefore we conclude that **our model of the gamma is sufficient**. However, this does not hold for the webcam. The error in the fit is larger as can be explained by the measurement error. Additionally, Figure A.3(b) shows that the data is not accurately modelled by the exponential model and the gamma differs between measurements. The estimated value of gamma differs with intensity between 0.53 and 0.72.

A.3.2 Dark current estimation

Using the gamma estimated above, we correct the image sequences and recalculate the average intensity for each section. Plotting these against the true intensity of the sections now should give a straight line which intersects the section-intensity axes at a value related to the dark current, see the figures in the center of Fig-

Table A.2: *Results of the camera characterization for each of the cameras. All estimates are least squares estimates of the data available. Between brackets are the standard deviations of the estimates. The amount of multiplicative noise is given for the highest pixel value (one in our case). The additive and multiplicative noise are averaged over all sequences.*

Camera	JAI	Siemens	JVC	Philips
Gamma	0.66 (0.02)	0.53 (0.02)	0.72 (0.06)	0.70 (0.10)
Dark Current $g_t h_t \mu_{DC}$	-0.003 (0.005)	-0.0005 (0.009)	0.005 (0.016)	0.005 (0.03)
Add. noise $g_t N_R$	0.003	0.003	0.005	0.012
Multipl. noise $g_t N_S$	0.021	0.021	0.019	0.011
Quant. noise N_Q	0.0003	0.001	0.001	0.001

ures A.2 and A.3.

The dark current is independent of apparent gain. Before fitting a line, the data from all sequences is scaled such that their apparent gain is equal to the lowest of the sequences for that camera. Then one line is fitted in least squares through all data. The estimated dark current and the error of the estimate are given in Table A.2. The dark current we estimated is for all cameras smaller then the standard deviation of the estimate, so we cannot conclude that it is unequal to zero. The dark current is for all cameras lower than the additive noise, so we conclude that for pixels with a sufficiently large intensity **we can neglect the dark current**.

A.3.3 Noise estimation

The distribution of the noise in the CCD model contains both additive and multiplicative noise. We plot the standard deviation over the gamma-corrected images against its average for a number of pixels to see the effect of both contributions, see the figures on the bottom in Figures A.2 and A.3. The intersection of a straight line fitted to this data with the standard deviation axes gives the contribution of the additive noise and the slope of the line gives the amount of multiplicative noise. Estimates of the amount of additive and multiplicative noise are given in Table A.2. We conclude that **the amount of multiplicative noise exceeds the amount of additive noise at intensities above 10 to 30% of the intensity range.**

For the webcam the additive part of the noise is more important than the multiplicative part over the entire intensity range, see Figure A.3(f).

We observe that for most cameras, the fitted lines through the noise are almost parallel. This indicates that these cameras control the intensity by changing their iris of shutter time. For the Siemens camera this does not hold, see Figure A.2(f). For this camera the noise changes with a changing image intensity, but not with the same amount as the slope of the intensity in Figure A.2(d). This is caused by the combination of automatic gain and automatic shutter control.

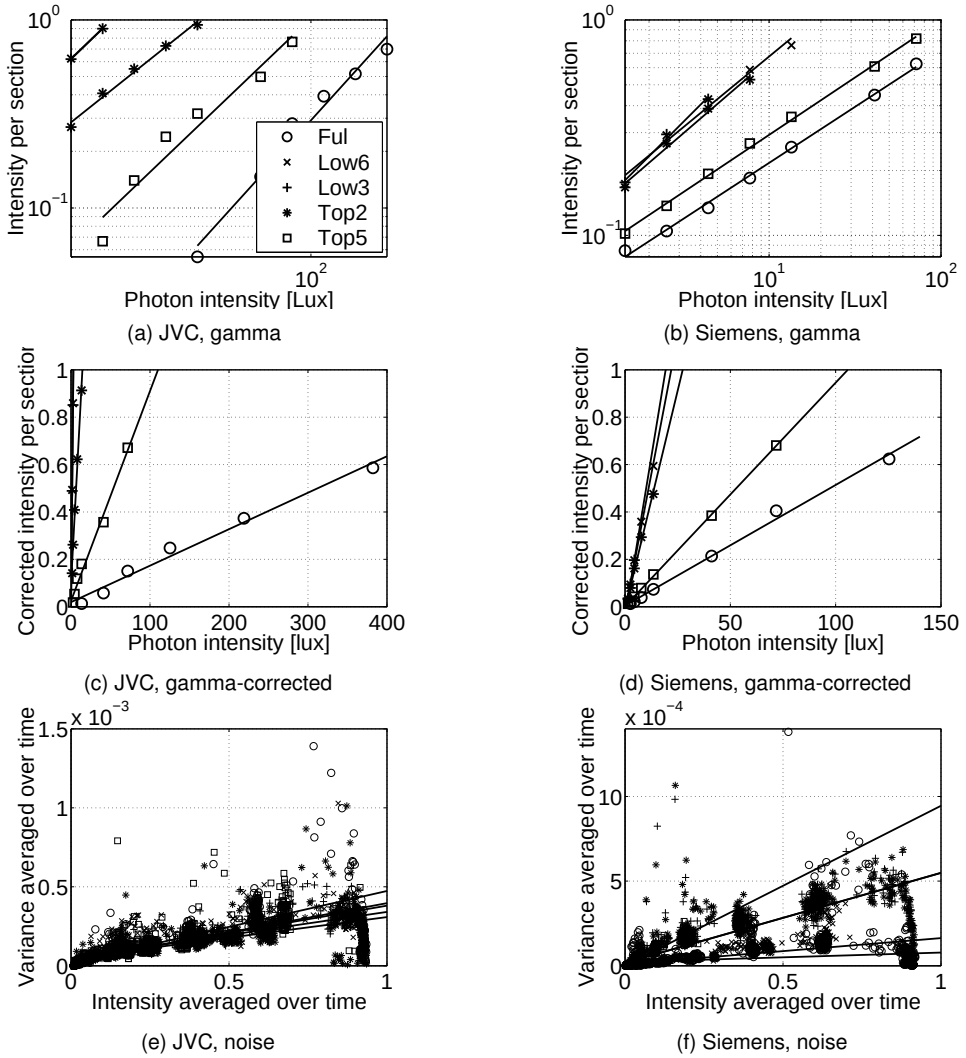


Figure A.2: *Results from the CCD verification experiments (1), continued in Figure A.3. The top shows a log-log plot of the image intensity per section against the true intensity of the section. The center graphs show a plot of gamma-corrected image intensity per section against the true section intensitiy. The lowest graphs show the standard deviation over time plot against the average over time.*

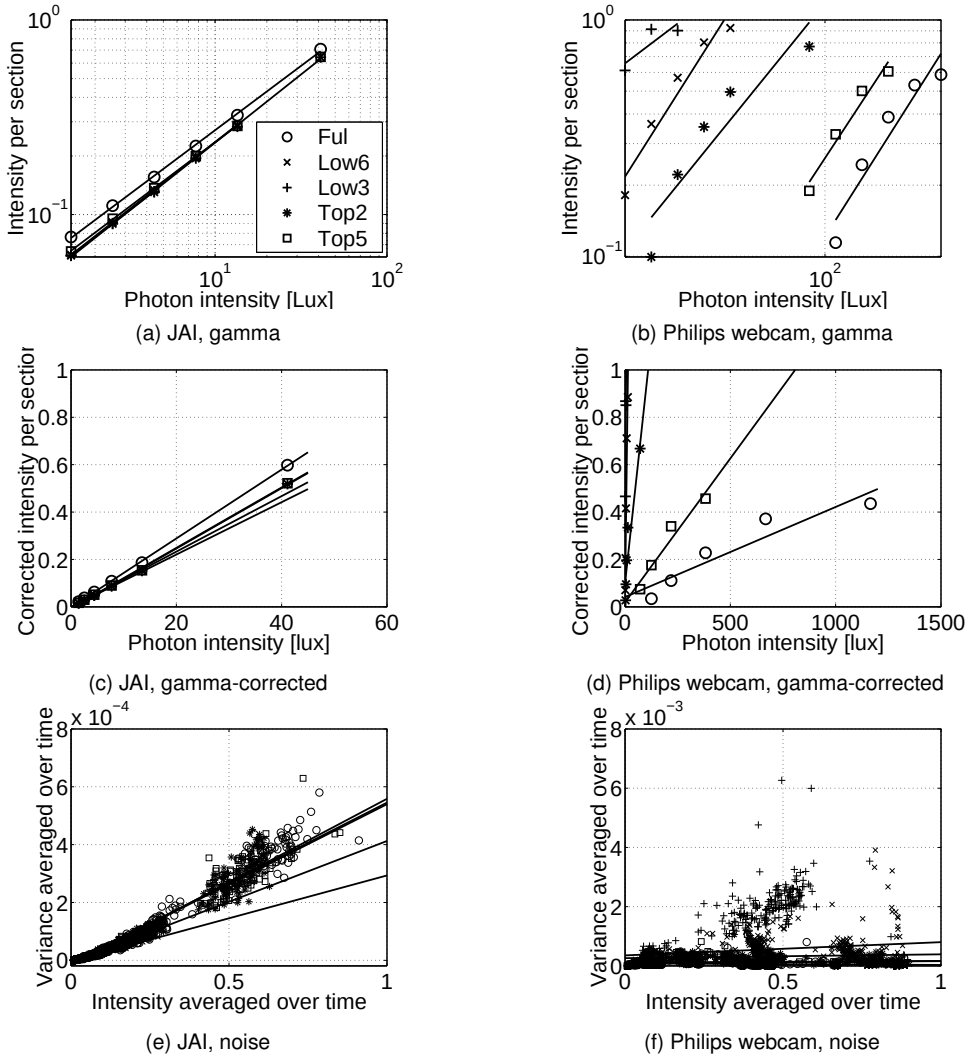


Figure A.3: *Results from the CCD verification experiments (2), continued from Figure A.2. The different sets of data are recorded with different parts of the measurement object, see the legend in subfigure (a), explained in Figure A.1.*

A.3.4 Artifacts

The Philips webcam seems to have some artifacts, see Figure A.3 on the right. Among others its gamma changes with intensity and is insufficiently modelled by the exponential. Therefore, this camera violates the general model given in Equation A.3.

A.4 Simplifications to the CCD model

Our experimental validation concludes that we can neglect the contribution of the dark current as the measured value lies within one standard deviation from zero and is smaller than the additive noise. This simplifies Equation A.3 to

$$i_t = g_t^{\gamma} \left(h_t i_0 + N_S + N_R \right)^{\gamma} + N_Q \; . \tag{A.4}$$

The remaining noise terms are all zero-mean. The shot noise N_S is multiplicative and both the readout noise N_R and the quantization noise N_Q are additive.

Our experiments also show that the additive noise contributions (N_R and N_Q) are equal to or smaller than the multiplicative noise contribution for sufficiently large intensity values (larger than 10 to 30 % of the intensity range). For sufficiently large intensity this simplifies the equation above to

$$i_t = g_t^{\gamma} \left(h_t i_0 + N_S \right)^{\gamma} \; . \tag{A.5}$$

A.5 Conclusions

A model of a CCD model was introduced and experimentally evaluated using a range of different cameras. Experiments show that the model of the CCD is sufficient for all cameras, except the low-end Philips webcam. The gamma correction in these cameras is sufficiently accurately described by an exponential gamma model.

Experiments further demonstrate that for sufficiently large pixel intensities the model can be simplified. Specifically, the dark current can be neglected and the additive noise is exceeded by the multiplicative noise at intensities over 10 to 30% of the intensity range.

Using this model, all cameras except for the webcam can be used for accurate measurements. Using the webcam for computer vision can be expected to give problems as its response cannot be predicted using a common CCD model. The gamma is not accurately modelled by the general model and depends on the image intensity.

Results on the webcam show that it is dangerous to pick a general camera model and assume its validity. It is important to validate the model for the specific camera used.

Automatic correction to intensity changes is performed differently by the different cameras. The JAI camera has no automatic intensity adjustment, the JVC camera

changes shutter time, the Siemens camera changes both shutter time and gain and the Philips camera seems to change the value of gamma. This has an effect on the image noise. For the JAI and JVC camera the amount of noise for a certain pixel intensity is independent of the apparent gain, whereas for the Philips and Siemens camera this is not true.

Appendix B Image sequences

B.1 Introduction

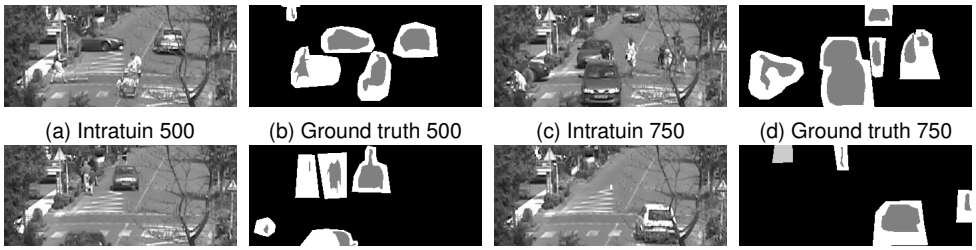
Throughout this thesis, five image sequences were used for evaluation of the proposed algorithms. In this appendix, those image sequences will be discussed. Example images will be given, together with their manually labelled ground truth.

B.2 Images for pixel classification

Three image sequences were used for the evaluating the pixel classification, see also Figures B.1,B.2 and B.3:

- **Intratuin**: Parking lot with waving tree branches. The sequence contains cars and pedestrians, moving both slowly and fast. This sequence has 150x350 pixels and 1250 frames.
- **Schiphol**: Main hall of Schiphol airport. The sequence contains relatively large objects, some of which become stationary. This sequence has 90x120 pixels and 1750 frames.
- **PETS01-3TR1**: A for computational efficiency sub-sampled cut-out of the images of dataset 3, training, camera 1 from the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2001 (PETS), [Ferryman, 2001]. The images contain relatively few object pixels. The part of the image used is that between rows 300 and 520, skipping the odd rows and between columns 350 and 750, skipping the odd columns. This sequence has 120x200 pixels and 5500 frames.

All sequences are RGB color video data with eight bit per color. The Intratuin and Schiphol sequences are recorded by us. The PETS 2001 data has been obtained from pets2001.visualsurveillance.org.



(a) Intratuin 500 (b) Ground truth 500 (c) Intratuin 750 (d) Ground truth 750

(e) Intratuin 1000 (f) Ground truth 1000 (g) Intratuin 1250 (h) Ground truth 1250

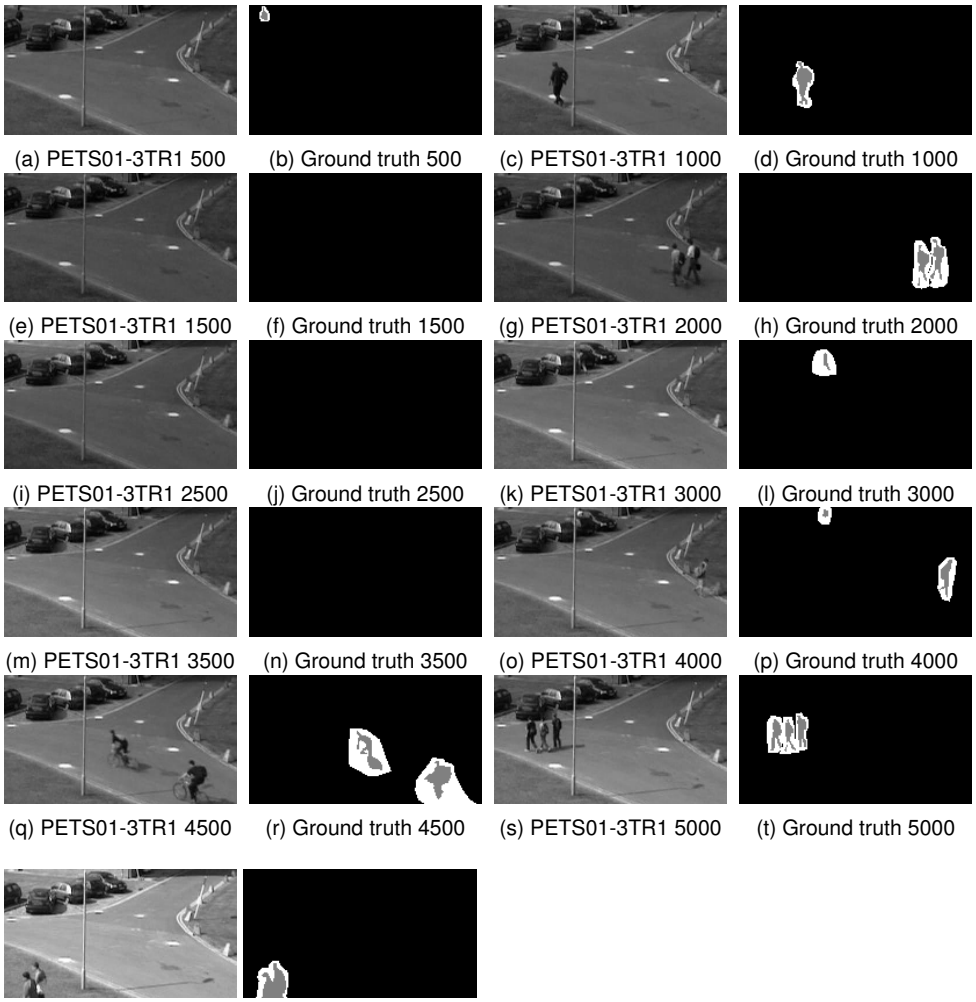
Figure B.1: *Some frames from the Intratuin sequence with associated ground truth. Ground truth labelling: black is background, gray is foreground and white is any.*

(a) Schiphol 500 (b) Ground truth 500 (c) Schiphol 750 (d) Ground truth 750

(e) Schiphol 1000 (f) Ground truth 1000 (g) Schiphol 1250 (h) Ground truth 1250

(i) Schiphol 1500 (j) Ground truth 1500 (k) Schiphol 1750 (l) Ground truth 1750

Figure B.2: *Some frames from the Schiphol sequence. Ground truth labelling: black is background, dark gray is foreground, light gray is shadow and white is any.*



(a) PETS01-3TR1 500 (b) Ground truth 500 (c) PETS01-3TR1 1000 (d) Ground truth 1000

(e) PETS01-3TR1 1500 (f) Ground truth 1500 (g) PETS01-3TR1 2000 (h) Ground truth 2000

(i) PETS01-3TR1 2500 (j) Ground truth 2500 (k) PETS01-3TR1 3000 (l) Ground truth 3000

(m) PETS01-3TR1 3500 (n) Ground truth 3500 (o) PETS01-3TR1 4000 (p) Ground truth 4000

(q) PETS01-3TR1 4500 (r) Ground truth 4500 (s) PETS01-3TR1 5000 (t) Ground truth 5000

(u) PETS01-3TR1 5500 (v) Ground truth 5500

Figure B.3: *Some frames of the PETS01-3TR1 sequence. Ground truth labelling: black is background, gray is foreground and white is any.*

B.2.1 Ground truth labelling

For each sequence, four to eleven frames were manually labelled. Each pixel was labelled as: foreground, background or any. The label any is used for the edges of objects, where it is difficult (for a human) to decide whether this pixel should be labelled as either foreground or background.

Labelling was done by defining an inner contour and an outer contour of all (groups of) moving objects. All pixels inside the inner contour are labelled ground truth foreground and all pixels outside the outer contour ground truth background. Pixels between the two contours are labelled any. These pixels are not used for evaluation.

For some frames in the Schiphol sequence, objects have become static. These objects are also labelled any. In the Schiphol sequence a third class has been labelled: ground truth shadow. Pixels inside this contour, and not inside any other contour are used only for the evaluation of the shadow detection algorithm in Chapter 6 . In all other chapters these pixels are not used.

All manually labelled images are given next to the corresponding frame in Figures B.1, B.2 and B.3.

B.3 Images for trajectory evaluation

For track evaluation, two image sequences were used, see also Figure B.4:

- **PETS01-1TE1**: We used a for computational efficiency sub-sampled version of dataset 1, testing, camera 1 from the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2001 (PETS), [Ferryman, 2001]. The images contain relatively few object pixels. We use only every fourth row and column. This sequence has 144x192 pixels and 2688 frames.
- **PETS04**: We used a for computational efficiency sub-sampled version of the Rest-FallOnFloor data from the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2004 (PETS) [Crowley et al., 2004] provided by the EC Funded CAVIAR project/IST 2001 37540. The images contain amongst others a person falling on the floor and lying there for some time. We use only the odd rows and columns. This sequence has 144x192 pixels and 1006 frames.

Both sequences are RGB color video data with eight bit per color. For each sequence, ground truth and data was available from the PETS workshops. We obtained it from pets2001.visualsurveillance.org and pets2004.visualsurveillance.org respectively. From the ground truth we used for each track and each frame the center location of the region of interest.

(a) PETS01-1TE1 1000 (b) PETS01-1TE1 1500 (c) PETS01-1TE1 2000 (d) PETS01-1TE1 2500

(e) PETS04 250 (f) PETS04 500 (g) PETS04 750 (h) PETS04 1000

Figure B.4: *Some frames of the images sequences used for blob-track evaluation.*

Appendix C Online EM and Stauffer classification

C.1 Introduction

For several applications it is necessary to distinguish between foreground and background objects, for example for detection and tracking of moving objects. This can be done by making a model of the background, and then checking for each pixel in a new frame whether or not their values can be explained by the background model. If a pixel-value can be explained by the model, it is classified background in that frame, otherwise foreground.

A very popular background model is the Mixture of Gaussians (MoG) model. The color Probability Density Function (PDF) over time is modelled using N_k gaussian kernels. Such a model can be estimated from the data using Maximum likelihood (ML) parameter estimation. Unfortunately, an analytical expression for ML estimation of a MoG model is not available. The Expectation Maximization (EM) algorithm [Dempster et al., 1977; Neal and Hinton, 1998] is a technique for ML estimation for incomplete data. With this technique simple equations can be obtained for mixture density estimation. This will be discussed in Section C.2.

In the case of image data, the online EM algorithm [Priebe, 1994] adds exponential forgetting and enables a real-time implementation. This is the algorithm used throughout this thesis for updating the model of the background. In Section C.3 its equations will be derived.

C.2 Maximum Likelihood and Expectation Maximization

We will follow the reasoning given in [Bilmes, 1997], see that paper or one of the other papers regarding Expectation Maximization (EM) mentioned below for a more in-depth discussion.

C.2.1 Maximum Likelihood estimation

Maximum Likelihood estimation [Duda and Hart, 1973] is a technique for estimation of the parameters Θ of a PDF $p(\vec{x}|\Theta)$. The parameters are estimated from a set of N_{sample} data points $X = \{\vec{x}_1, \cdots, \vec{x}_{N_{\text{sample}}}\}$ drawn independently from that distribution. The likelihood function to be maximized is given by

$$p(X|\Theta) = \prod_{i=1}^{N_{\text{sample}}} p(\vec{x}_i|\Theta) = L(\Theta|X) \ . \tag{C.1}$$

The function $L(\Theta|X)$ is called the likelihood function. Maximizing this function gives the optimal parameters Θ^*:

$$\Theta^* = \arg\max_{\Theta} \left(L(\Theta|X)\right) \ . \tag{C.2}$$

Usually the logarithm of this function, the log-likelihood, is optimized for ease of computation.

The level of difficulty of this maximization depends on the PDF $p(\vec{x}|\Theta)$. If it is a simple Gaussian distribution, the minimization has an analytical form. The derivatives can be set to zero and the standard equations for μ and σ^2 can be found easily. However, for a MoG such analytical equations are not available.

C.2.2 Expectation Maximization

Expectation Maximization (EM) enables to obtain a ML solution when an analytical equation is not available or is very complex. The EM algorithm [Bishop, 1995; Dempster et al., 1977; Ghahramani and Jordan, 1994; Jordan and Jacobs, 1994; Neal and Hinton, 1998; Redner and Walker, 1984; Verbeek, 2004; Wu, 1983] allows finding a ML estimate for incomplete data. It is used for finding the ML estimate when the data has missing values due to the observation process or when optimizing the likelihood is computationally intractable, but can be simplified by assuming the existence of additional but missing variables.

The EM algorithm solves the ML problem iteratively. Two steps can be identified. In the E–step values for the hidden variables are assumed. These values are used in the M–step to estimate the parameters that are not hidden. Each iteration the likelihood increases. It is guaranteed that the solution converges to a local maximum of the likelihood function. For prove and discussion of the rate of convergence see [Dempster et al., 1977; Jordan and Xu, 1995; Redner and Walker, 1984; Wu, 1983; Xu and Jordan, 1996].

EM for a MoG

In §4.2.1 the PDF of a MoG model is given. ML parameter estimation for a MoG model can be simplified by adding a hidden variables which specifies from which

kernel each observation was drawn. All parameters, both hidden and observed, can now be estimated using EM.

Given a sequence of N_{sample} independent samples $x_{i,c}$, the parameters π_k, $\mu_{k,c}$ and $\sigma_{k,c}$ with $c \in \{R, G, B\}$, must be estimated for each kernel k.

EM gives analytical equations for estimating the parameters iteratively. A derivation of these equations can be found, amongst others, in [Bilmes, 1997].

The E–step is given by the likelihood that value $\vec{x}$ is part of kernel k:

$$P(k|\vec{x}_i) = \frac{\pi_k f(\vec{x}_i|\vec{\mu}_k, V_k)}{P(\vec{x}_i|\beta_0)} \; . \qquad \text{“Posterior”} \tag{C.3}$$

The term $P(\vec{x}_i|\beta_0)$ in the nominator is the likelihood of the entire MoG for this measurement, so this introduces knowledge of the likelihood of the other kernels in the model.

The M–step updates the individual Gaussian kernels:

$$\pi_k = \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} P(k|\vec{x}_i) \qquad \text{”Prior”} \tag{C.4}$$

$$\mu_{k,c} = \frac{1}{N_{\text{sample}}\pi_k} \sum_{i=1}^{N_{\text{sample}}} P(k|\vec{x}_i) x_{i,c} \qquad \text{”Mean”} \tag{C.5}$$

$$\sigma^2_{k,c} = \frac{1}{N_{\text{sample}}\pi_k} \sum_{i=1}^{N_{\text{sample}}} P(k|\vec{x}_i)(x_{i,c} - \mu_{k,c})^2 \; . \qquad \text{”Variance”} \tag{C.6}$$

This solution has two drawbacks. First, the entire model has to be re-estimated each new frame. Second, the oldest frame has the same influence as the newest, so this does not model the effect of slowly varying lightning conditions very well. The online updating strategy described in the next section solves both drawbacks.

C.3 Online EM

To enable a real-time application we need to calculate the parameters recursively. In algebraic notation[1]:

$$\pi_k := \pi_k + \frac{1}{n}\big(P(k|\vec{x}) - \pi_k\big) \tag{C.7}$$

$$\mu_k := \mu_k + \frac{1}{n\pi_k} P(k|\vec{x})(x - \mu_k) \tag{C.8}$$

$$\sigma^2_k := \sigma^2_k + \frac{1}{n\pi_k} P(k|\vec{x}_i)\big((x_i - \mu_k)^2 - \sigma^2_k\big) \; . \tag{C.9}$$

[1] For clarity we will omit the index $c \in \{R, G, B\}$ in $x_{k,c}$, $\mu_{k,c}$ and $\sigma^2_{k,c}$

Derivation of these equations is given below.

To weight newer frames heavier than older frames, we can replace $\frac{1}{n}$ with a "forgetting term" u, also referred to as "update speed". This gives:

$$\pi_k := \pi_k + u\big(P(k|\vec{x}) - \pi_k\big) \tag{C.10}$$

$$\mu_k := \mu_k + \frac{u}{\pi_k} P(k|\vec{x})(x - \mu_k) \tag{C.11}$$

$$\sigma_k^2 := \sigma_k^2 + \frac{u}{\pi_k} P(k|\vec{x}_i)\big((x_i - \mu_k)^2 - \sigma_k^2\big) \ . \tag{C.12}$$

This is especially useful for slow changes in illumination.

[Neal and Hinton, 1998; Titterington, 1984] prove that the online version of EM converges to a local likelihood maximum for infinite data points. This cannot be said for the variant with exponential forgetting. However, [Neal and Hinton, 1998] shows by experiment that the use of exponential forgetting results in a log-likelihood very close to that obtained with EM and online EM. It obtains this estimate faster than the standard EM algorithm and the online EM algorithm, and has a lower computational complexity.

C.3.1 Derivation for the online EM formulas

The online EM equations can easily be derived from the standard EM equations, we use the derivation given in [Zwarthoed, 2002]. Note that the order of updating the different parameters is fixed. First the posterior probability should be calculated for the new measurement, then the prior, mean and standard deviation are updated respectively. In each parameter the updated versions of the previous parameters are used.

Prior

EM gives for the prior after update n:

$$\pi_{k,n} = \frac{1}{n} \sum_{i=1}^{n} P(k|\vec{x}_i) \ . \tag{C.4'}$$

Adding measurement $n+1$, we obtain

$$\pi_{k,n+1} = \frac{1}{n+1} \sum_{i=1}^{n+1} P(k|\vec{x}_i) \ .$$

The new measurement is split-off from the sum

$$\pi_{k,n+1} = \frac{1}{n+1} \Big(\sum_{i=1}^{n} P(k|\vec{x}_i) + P(k|\vec{x}_{n+1}) \Big) \ .$$

Assume the posterior probabilities of the old data are frozen. They do not need to be re-estimated, so the sum can replaced by the old estimate

$$\pi_{k,n+1} \simeq \frac{1}{n+1}\big(n\pi_{k,n} + P(k|\vec{x}_{n+1})\big) \ .$$

Rearranging gives

$$\pi_{k,n+1} \simeq \frac{1}{n+1}\big((n+1)\pi_{k,n} - \pi_{k,n} + P(k|\vec{x}_{n+1})\big) \ ,$$

and then

$$\boxed{\pi_{k,n+1} \simeq \pi_{k,n} + \frac{1}{n+1}\big(P(k|\vec{x}_{n+1}) - \pi_{k,n}\big) \ .} \tag{C.13}$$

Mean

EM gives for each element of the mean vector $\vec{\mu}_{k,n}$ after update n:

$$\mu_{k,n} = \frac{1}{n\pi_{k,n}} \sum_{i=1}^{n} P(k|\vec{x}_i)x_i \ . \tag{C.5′}$$

Adding measurement $n+1$, we obtain

$$\mu_{k,n+1} = \frac{1}{(n+1)\pi_{k,n+1}} \sum_{i=1}^{n+1} P(k|\vec{x}_i)x_i \ .$$

The new measurement is split-off from the sum

$$\mu_{k,n+1} = \frac{1}{(n+1)\pi_{k,n+1}}\Big(\sum_{i=1}^{n} P(k|\vec{x}_i)x_i + P(k|\vec{x}_{n+1})x_{n+1}\Big) \ .$$

As for the prior, assume the posterior probabilities of the old data are frozen. They do not need to be re-estimated, so the sum can replaced by the old estimate

$$\mu_{k,n+1} \simeq \frac{1}{(n+1)\pi_{k,n+1}}\Big(n\pi_{k,n}\mu_{k,n} + P(k|\vec{x}_{n+1})x_{n+1}\Big) \ .$$

The prior $\pi_{k,n}$ is replaced by the new prior $\pi_{k,n+1}$

$$\mu_{k,n+1} \simeq \frac{1}{(n+1)\pi_{k,n+1}}\Big(n\big(\frac{n+1}{n}\pi_{k,n+1} - \frac{1}{n}P(k|\vec{x}_{n+1})\big)\mu_{k,n} + P(k|\vec{x}_{n+1})x_{n+1}\Big) \ .$$

Rearranging gives

$$\mu_{k,n+1} \simeq \frac{1}{(n+1)\pi_{k,n+1}}\Big((n+1)\pi_{k,n+1}\mu_{k,n} - P(k|\vec{x}_{n+1})\mu_{k,n} + P(k|\vec{x}_{n+1})x_{n+1}\Big) \ ,$$

and

$$\boxed{\mu_{k,n+1} = \mu_{k,n} + \frac{1}{(n+1)\pi_{k,n+1}}P(k|\vec{x}_{n+1})(x_{n+1} - \mu_{k,n}) \ .} \tag{C.14}$$

variance

Derivation for the online update equation of the variance the reader is referred to that of the mean. The mean μ is substituted by the variance σ^2 and the measurement x is replaced by $(x-\mu)^2$. The derivation is exactly the same.

Bibliography

Johnson I. Agbinya and David Rees. Multi-object tracking in video. **Real-Time Imaging**, 5 (5):295–304, 1999.

Jake K. Aggarwal and Quin Cai. Human motion analysis: a review. **Computer Vision and Image Understanding**, 73(3):364–356, 1999.

Yucel Altunbasak, Russel M. Mersereau, and Andrew J. Patti. A fast parametric motion estimation algorithm with illumination and lens distortion correction. **IEEE Transactions on Image Processing**, 12(4):395–408, 2003.

A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: a wireless sensor network for target detection, classification, and tracking. **Computer Networks**, 46(5):605–634, 2004.

Alireza Bab-Hadiashar and David Suter. Robust optic flow computation. **International Journal on Computer Vision**, 29(1):59–77, 1998.

Norman I. Badler and Stephen W. Smoliar. Digital representations of human movement. **ACM Computer Surveys**, 11(1):19–38, 1979.

John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. **International Journal on Computer Vision**, 12(1):43–77, 1994.

Adam M. Baumberg. **Learning Deformable Models for Tracking Human Motion**. PhD thesis, University of Leeds, 1995.

Jezekiel Ben-Arie, Zhiqian Wang, Purvin Pandit, and Shyamsundar Rajaram. Human activity recognition using multidimensional indexing. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 24(8):1091–1104, 2002.

Jeff A. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley, 1997.

Christopher M. Bishop. **Neural networks for pattern recognition**. Clarendon Press, Oxford, U.K., 1995.

Samuel Blackman and Robert Popoli. **Design and analysis of modern tracking systems**. Artech House, Norwood, MA, USA, 1999.

Lisa M. Brown. View independent vehicle/person classification. In **Proceedings of the ACM International Workshop on Video Surveillance & Sensor Networks (IWVSSN)**, 2004.

Hilary Buxton. Learning and understanding dynamic scene activity: a review. **Image and Vision Computing**, 21(1):125–136, 2003.

Thomas W. Calvert and Arthur E. Chapman. **Handbook of pattern recognition and image processing (vol. 2): computer vision**, chapter Analysis and synthesis of human movement, pages 431–474. Academic Press, Orlando, FL, USA, 1994.

M.B. Capellades, David S. Doermann, Daniel DeMenthon, and Rama Chellappa. An appearance based approach for human and object tracking. In **Proceedings of the IEEE International Conference on Image Processing (ICIP)**, pages 85–88, 2003.

Kenneth R. Castleman. **Digital Image Processing**. Prentice Hall, Englewood Cliffs, NJ, USA, 1996.

C. Cédras and Mubarak Shah. Motion–based recognition: a survey. **Image and Vision Computing**, 13(2):129–155, 1995.

Hwann-Tzong Chen, Tyng-Luh Liu, and Chiou-Shann Fuh. Probabilistic tracking with adaptive feature selection. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages I:736–739, 2004.

B. Coifman, David J. Beymer, Phil McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. **Transportation Research–Part C**, 6(4):271–288, 1998.

Robert T. Collins, Alan J. Lipton, and Takeo Kanade. Introduction to the special section on video surveillance. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 22(8):745–746, 2000a.

Robert T. Collins, Alan J. Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, and Osamu Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, 2000b.

Robert T. Collins and Yanxi Liu. On-line selection of discriminative tracking features. Technical Report CMU-RI-TR-03-12, Robotics Institute, Carnegie Mellon University, 2003.

Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real–time tracking of non–rigid objects using mean shift. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 142–151, 2000.

Ingemar J. Cox, Sibastien Roy, and Sunita L. Hingorani. Dynamic histogram warping of image pairs for constant image brightness. In **Proceedings of the IEEE International Conference on Image Processing (ICIP)**, pages 2366–2369, 1995.

Marco Cristani, Manuele Bicego, and Vittorio Murino. On-line adaptive background modelling for audio surveillance. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages 399–402, 2004.

James L. Crowley, Robert B. Fisher, and José Santos Victor, editors. **Proceedings of the IEEE International Workshop on Performanse Evaluation of Tracking and Surveillance (PETS)**, in conjunction with IEEE ECCV, 2004.

Rita Cucchiara, Costantino Grana, G. Tardini, and R. Vezzani. Probabilistic people tracking for occlusion handling. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages I:132–135, 2004.

Ross Cutler and Larry S. Davis. Robust real-time periodic motion detection, analysis, and applications. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 22(8): 781–796, 2000.

Douglas Decarlo and Dimitris Metaxas. Optical flow constraints on deformable models with applications to face tracking. **International Journal on Computer Vision**, 38(2):99–127, 2000.

Andrew P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM–algorithm. **Journal of the Royal Statistical Society B**, 39(1):1–38, 1977.

Anuj Dev. **Visual navigation on optical flow**. PhD thesis, University of Amsterdam, 1998.

Jaime Dever, Niels da Vitoria Lobo, and Mubarak Shah. Automatic visual recognition of armed robbery. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages 451–455, 2002.

Philip van Dorp and Frans C.A. Groen. Human walking estimation with radar. **IEE Proceedings - Radar, Sonar and Navigation**, 150(5):356–365, 2003.

Marie-Pierre Dubuisson and Anil K. Jain. 2d matching of 3d moving objects in color outdoor scenes. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 887–891, 1994.

Richard O. Duda and Peter E. Hart. **Pattern Classification and Scene Analysis**. John Wiley & Sons, Singapore, 1973.

James Ferryman, editor. **Proceedings of the IEEE International Workshop on Performanse Evaluation of Tracking and Surveillance (PETS)**, in conjunction with IEEE CVPR, 2001.

David A. Forsyth and Jean Ponce. **Computer Vision: A Modern Approach**. Prentice Hall, Englewood Cliffs, NJ, USA, 2002.

Warren F. Gardner and Daryl T. Lawton. Interactive model-based vehicle tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 18(11):1115–1121, 1996.

Guillaume Gasser, Nathaniel Bird, Osama Masoud, and Nikolaos Papanikolopoulos. Human activities monitoring at bus stops. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages I:90–95, 2004.

Dariu M. Gavrila. **Vision–based 3–D tracking of humans in action**. PhD thesis, University of Maryland, 1996.

Dariu M. Gavrila. The visual analysis of human movement: a survey. **Computer Vision and Image Understanding**, 73(1):82–98, 1999.

Dariu M. Gavrila. Pedestrian detection from a moving vehicle. In **Proceedings of the European Conference on Computer Vision (ECCV)**, pages II:37–49, 2000.

Dariu M. Gavrila and Vasanth Philomin. Real-time object detection for "smart" vehicles. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages 87–93, 1999.

A. Geurtz. **Model-based shape estimation**. PhD thesis, Polytechnic Institute of Lausanne, 1993.

Zoubin Ghahramani and Michael I. Jordan. Learning from incomplete data. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1994.

Martin Gill, Jenna Allen, Jane Bryan, Deena Kara, Ross Little, Sam Waples, Angela Spriggs, Javier Argomaniz, Patricia Jessiman, Jonathan Kilworth, and Daniel Swain. **The impact of CCTV: fourteen case studies**. Number 15 in Home Office Online Report. Home Office, London, UK, 2005.

Shaogang Gong and Hilary Buxton. Understanding visual behaviour, special issue introduction. **Image and Vision Computing**, 20(12):825–826, 2002.

Robert M. Gray. **Entropy and information theory**. Springer–Verlag, New York, NY, USA, 1990.

Michael Greiffenhagen, Visvanathan Ramesh, and Heinrich Niemann. The systematic design and analysis cycle of a vision system: A case study in video surveillance. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages II:704–711, 2001.

W. Eric L. Grimson, Chris Stauffer, Raquel Romano, and Lily Lee. Using adaptive tracking to classify and monitor activities in a site. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 22–29, 1998.

Silviu Guiasu. **Information theory with applications**. McGraw–Hill, New York, NY, USA, 1977.

Yanlin Guo, Gang Xu, and Saburo Tsuji. Understanding human motion patterns. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages II:325–329, 1994.

Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 20(10):1125–1139, 1998.

R. Ismail Haritaoglu, David Harwood, and Larry S. Davis. W^4: Real–time surveillance of people and their activities. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 22(8):809–830, 2000.

Michael Harville, Gaile G. Gordon, and John Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In **Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video (WDREV)**, pages 3–11, 2001.

Yinghua He, Hong Wang, and Bo Zhang. Background updating in illumination–variant scenes. In **Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)**, pages I:515–519, 2003.

Glenn E. Healey and Raghava Kondepudy. Radiometric CCD camera calibration and noise estimation. **IEEE Transactions on Image Processing**, 16(3):267–276, 1994.

Horst Hischof, Horst Wildenauer, and Ales Leonardis. Illumination insensitive recognition using eigenspaces. **Computer Vision and Image Understanding**, 95(1):86–104, 2004.

Jenny Hogan. Your every move will be analysed. **New Scientist**, 2003.

Thanarat Horprasert, David Harwood, and Larry S. Davis. A statistical approach for real–time robust background subtraction and shadow detection. In **Proceedings of the IEEE FRAME-RATE Workshop**, pages II:751–767, 1999.

Jun-Wei Hsieh, Chia-Jung Chang, Wen-Fong Hu, and Yung-Sheng Chen. Shadow elimination for effective moving object detection by Gaussian shadow modeling. **Image and Vision Computing**, 21(6):505–516, 2003.

Min Hu, Weiming Hu, and Tie-Niu Tan. Tracking people through occlusions. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages II:724–727, 2004a.

Weiming Hu, Tien-Niu Tan, Liang Wang, and Steven J. Maybank. A survey on visual surveillance of object motion and behaviors. **IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews**, 34(3):334–352, 2004b.

Jing Huang, Ravi Kumar, Mandar Mitra, and Wei-Jing Zhu. Spatial color indexing and applications. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages 602–607, 1998.

Michael Isard and Andrew Blake. ICONDENSATION: Unifying low–level and high–level tracking in a stochastic framework. In **Proceedings of the European Conference on Computer Vision (ECCV)**, pages I:893–908, 1998.

Aleksandar M. Ivanovic and Thomas S. Huang. A probabilistic framework for segmentation and tracking of multiple non rigid objects for video surveillance. In **Proceedings of the IEEE International Conference on Image Processing (ICIP)**, pages 353–356, 2004.

Sumer Jabri, Zoran Duric, Harry Wechsler, and Azriel Rosenfeld. Detection and location of people in video images using adaptive fusion of color and edge information. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages IV:627–630, 2000.

David W. Jacobs, Peter N. Belhumeur, and Ronen Basri. Comparing images under variable illumination. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 610–617, 1998.

Omar Javed, Khurram Shafique, and Mubarak Shah. A hierarchical approach to robust background subtraction using color and gradient information. In **Proceedings of the IEEE Workshop on Motion and Video Computing (WMVC)**, pages 22–27, 2002.

Omar Javed and Mubarak Shah. Tracking and object classification for automated surveillance. In **Proceedings of the European Conference on Computer Vision (ECCV)**, pages IV:343–357, 2002.

G. Johansson. Visual perception of biological motion and a model for its analysis. **Perception and Psychophysics**, 14(2):201–211, 1973.

G. Johansson. Visual motion perception. **Scientific American**, 232(6):75–80, 85–88, 1975.

D. Johnson and S. Sinanovic. Symmetrizing the Kullback–Leibler distance. **IEEE Transactions on Information Theory**, 2001. Submitted, cmc.rice.edu/docs/docs/Joh2001Mar1Symmetrizi.pdf.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. **Neural Computing**, 6(2):181–214, 1994.

Michael I. Jordan and Lei Xu. Convergence results for the EM approach to mixtures of experts architectures. **Neural Networks**, 8(9):1409–1431, 1995.

Shanon X. Ju, Michael J. Black, and Yaser Yacoob. Cardboard people: A parameterized model of articulated image motion. In **Proceedings of the International Conference on Automatic Face and Gesture Recognition (FG)**, pages 38–44, 1996.

Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real–time tracking with shadow detection. In **Proceedings of the European Workshop on Advanced Video Based Surveillance Systems (AVBS)**, pages 149–158, 2001.

Ioannis A. Kakadiaris and Dimitri Metaxas. Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 81–87, 1996.

Ioannis A. Kakadiaris, Rajeev Sharma, and Mohammed Yeasin. Introduction to the special issue on human modeling, analysis, and synthesis. **Machine Vision and Applications**, 14(4):197–198, 2003.

K. Kamikura, Hiroki Watanabe, H. Jozawa, H. Kotera, and S. Ichinose. Global brightness–variation compensation for video coding. **IEEE Transactions Circuits and Systems**, 8 (8):988–1000, 1998.

Dae-Hyun Kim, Yong-In Yoon, and Jong-Soo Choi. An efficient method to build panoramic image mosaics. **Pattern Recognition Letters**, 24(14):2421–2429, 2003.

Janusz Konrad and M. Ristivojevic. Joint space-time image sequence segmentation based on volume competition and level sets. In **Proceedings of the IEEE International Conference on Image Processing (ICIP)**, pages I:573–576, 2002.

Ben J.A. Kröse, Nikos Vlassis, and Wojciech Zajdel. Bayesian methods for tracking and localization. In **Proceedings of the Philips Symposium On Intelligent Algorithms (SOIA)**, pages 27–38, 2004.

Herman Kruegle. **CCTV Surveillance, video practices and technology**. Butterworth–Heinemann, Boston, USA, 1995.

Inald R.L. Lagendijk and Jan Biemond. **Handbook of Image and Video Processing 2nd edition**, chapter Basic Methods for Image Restoration and Identification. Elsevier Academic Press, Burlington, MA, USA, 2005.

Holger Leuck and Hans-Hellmut Nagel. Model-based initialisation of vehicle tracking: Dependency on illumination. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages I:309–314, 2001.

Maylor K. Leung and Yee-Hong Yang. First sight: A human body outline labeling system. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 17(4):359–377, 1995.

Alan J. Lipton. Local application of optic flow to analyse rigid versus non-rigid motion. In **Proceedings of the IEEE FRAME-RATE Workshop**, 1999.

Alan J. Lipton, Hironobu Fujiyoshi, and Raju S. Patil. Moving target classification and tracking from real–time video. In **Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)**, pages 8–14, 1998.

Shan Lu, Dimitris N. Metaxas, Dimitris Samaras, and John Oliensis. Using multiple cues for hand tracking and model refinement. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 443–450, 2003.

Steven J. Maybank and Tie-Niu Tan. Introduction – surveillance. **International Journal on Computer Vision**, 37(2):173–173, 2000.

Steven J. Maybank and Tie-Niu Tan. Special issue on visual surveillance. **Image and Vision Computing**, 22(7):Page iii, 2004.

Stephen J. McKenna, Sumer Jabri, Zoran Duric, Azriel Rosenfeld, and Harry Wechsler. Tracking groups of people. **Computer Vision and Image Understanding**, 80(1):42–56, 2000.

Stephen J. McKenna, Yogesh Raja, and Shaogang Gong. Object tracking using adaptive color mixture models. In **Asian Conference on Computer Vision (ACCV)**, pages 615–622, 1998.

Thomas B. Moeslund and Erik Granum. A survey of computer vision–based human motion capture. **Computer Vision and Image Understanding**, 81(3):231–268, 2001.

Vassilios Morellas, Ioannis Pavlidis, and P. Tsiamyrtzis. Deter: Detection of events for threat evaluation and recognition. **Machine Vision and Applications**, 15(1):29–45, 2003.

Sohail Nadimi and Bir Bhanu. Physical models for moving shadow and object detection in video. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 26(8):1079–1087, 2004.

Kameswara R. Namuduri and Veeru N. Ramaswamy. Preface, video analysis. **Pattern Recognition Letters**, 25(7):753–754, 2004.

Radford M. Neal and Geoffrey E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In **Proceedings of the NATO Advanced Study Institute on Learning in graphical models**, pages 355–368, 1998.

Tat H. Nguyen, Marcel Worring, Rein van den Boomgaard, and Arnold W.M. Smeulders. Tracking non-parameterized object contours in video. **IEEE Transactions on Image Processing**, 11(9):1081–1091, 2002.

Sourabh A. Niyogi and Edward H. Adelson. Analyzing gait with spatiotemporal surfaces. In **Proceedings of the IEEE Workshop on Nonrigid and Articulated Motion (WNAM)**, pages 64–69, 1994.

Clive Norris and Gary Armstrong. **The Maximum Surveillance Society: The Rise of CCTV**. Berg publishers, Oxford, UK, 1999.

Katja Nummiaro, Esther Koller-Meier, and Luc J. Van Gool. Color features for tracking non-rigid objects. **Chinese Journal of Automation**, 29(3):345–355, 2003.

Florian Ockhuysen. Detection of suspicious motion patterns. Master's thesis, Utrecht University, To be published, 2006.

Naoya Ohta. A statistical approach to background subtraction for surveillance systems. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages II:751–767, 2001.

Nikos Paragios and Rachid Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 22(3):266–280, 2000.

Sangho Park and Jake K. Aggarwal. Segmentation and tracking of interacting human body parts under occlusion and shadowing. In **Proceedings of the IEEE Workshop on Motion and Video Computing (WMVC)**, pages 105–111, 2002.

Sangho Park and Jake K. Aggarwal. Recognition of two-person interactions using a hierarchical bayesian network. In **Proceedings of the IEEE International Workshop on Visual Surveillance (IWVS)**, pages 65–76, 2003.

J.A. Paterson and Andrew W. Fitzgibbon. 3d head tracking using non-linear optimization. In **Proceedings of the Brittisch Machine Vision Conference (BMVC)**, pages II:609–618, 2003.

Ioannis Pavlidis, Vassilios Morellas, P. Tsiamyrtzis, and Steven A. Harp. Urban surveillance systems: from the laboratory to the commercial world. **Proceedings of the IEEE**, 89(10): 1478–1497, 2001.

Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color–based probabilistic tracking. In **Proceedings of the European Conference on Computer Vision (ECCV)**, pages I:661–675, 2002.

Natan Peterfreund. Robust tracking of position and velocity with kalman snakes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 21(6):564–569, 1999.

Ramprasard Polana and Randal C. Nelson. Low level recognition of human motion. In **Proceedings of the IEEE Workshop on Nonrigid and Articulated Motion (WNAM)**, pages 77–82, 1994.

Ediz Polat, Mohammed Yeasin, and Rajeev Sharma. A 2d/3d model–based object tracking framework. **Pattern Recognition**, 36(9):2127–2141, 2003.

Andrea Prati, Ivana Mikic, Mohan M. Trivedi, and Rita Cucchiara. Detecting moving shadows: Formulation, algorithms and evaluation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 25(7):918–923, 2003.

Carey E. Priebe. Adaptive mixtures. **Journal of the American Statistical Association**, 89 (427):796–806, 1994.

Niko P. Pronk. Recognition of multiple actions in image sequences. Master's thesis, Delft University of Technology, 2004.

Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. **Machine Learning**, 42(3):203–231, 2001.

Francis K.H. Quek. Eyes in the interface. **Image and Vision Computing**, 13(6):511–525, 1995.

Yogesh Raja, Stephen J. McKenna, and Shaogang Gong. Segmentation and tracking using color mixture models. In **Asian Conference on Computer Vision (ACCV)**, pages 607–614, 1998.

Christopher Rasmussen and Gregory D. Hager. Probabilistic data association methods for tracking complex visual objects. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 23(6):560–576, 2001.

Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood and the EM algorithm. **SIAM Review**, 26(2):195–239, 1984.

Carlo S. Regazzoni and Gianluca L. Foresti. Guest editorial: Video processing and communications in real–time surveillance systems. **Real-Time Imaging**, 7(5):381–388, 2001.

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

Yann Ricquebourg and Patrick Bouthemy. Real-time tracking of moving persons by exploiting spatio-temporal image slices. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 22(8):797–808, 2000.

Rómer Rosales and Stan Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages II:117–123, 1999.

Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. **International Journal on Computer Vision**, 47(1-3): 7–42, 2002.

Gijs O. van Schelven. Reconstructie van menselijke houding uit camerabeelden (reconstruction of human body pose from camera images, in Dutch). Bachelor's thesis, TH Rijswijk, 2002.

M.J.J. Scott, M. Niranjan, and Richard W. Prager. Realisable classifiers: Improving operating performance on variable cost problems. In **Proceedings of the Brittisch Machine Vision Conference (BMVC)**, pages 306–315, 1998.

Andrew Senior. Tracking people with probabilistic appearance models. In **Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)**, pages 48–55, 2002.

Heung-Yeung Shum, Mei Han, and Richard Szeliski. Interactive construction of 3d models from panoramic mosaics. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 427–433, 1998.

Andreas Siebert. Retrieval of gamma corrected images. **Pattern Recognition Letters**, 22 (2):249–256, 2001.

A.H.C.M. Smeets. **Camera's in het publieke domein (Cameras in public, in Dutch)**. College Bescherming Persoonsgegevens(CBP), The Hague, The Netherlands, 2004.

Cristian Sminchisescu. **Estimation algorithms for ambiguous visual models - Three Dimensional Human Modeling and Motion Reconstruction in Monocular Video Sequences**. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

Luis Gonzalez Sotelino, Marco Saerens, and Hugues Bersini. Classification of temporal trajectories by continuous-time recurrent nets. **Neural Networks**, 7(5):767–776, 1994.

Martin Spengler and Bernt Schiele. Multi-object tracking: Explicit knowledge representation and implementation for complexity reduction. In **Proceedings of the Cognitive Vision Workshop (CVW)**, 2002.

Martin Spengler and Bernt Schiele. Multi–object tracking based on a modular knowledge hierarchy. In **Proceedings of the International Conference on Vision Systems (ICVS)**, pages 376–385, 2003.

Chris Stauffer. Automatic hierarchical classification using time-based co-occurrences. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 2333–2339, 1999.

Chris Stauffer and W. Eric L. Grimson. Adaptive background mixture models for real–time tracking. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages II:246–252, 1999.

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

Bjoern Stenger. **Model-Based Hand Tracking Using A Hierarchical Bayesian Filter**. PhD thesis, University of Cambridge, 2004.

Bjoern Stenger, Paulo R.S. Mendonça, and Roberto Cipolla. Model based 3D tracking of an articulated hand. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages II:310–315, 2001.

Elena Stringa and Carlo Regazzoni. Real–time videoshot detection for scene surveillance applications. **IEEE Transactions on Image Processing**, 9(1):69–80, 2000.

Michael J. Swain and Dana H. Ballard. Indexing via color histograms. In **Defense Advanced Research Projects Agency (DARPA)**, pages 623–630, 1990.

Tanveer Syeda-Mahmood, R. Ismail Haritaoglu, and Thomas S. Huang. CVIU special issue on event detection in video. **Computer Vision and Image Understanding**, 96(2):97–99, 2004.

Tie-Niu Tan, G.D. Sullivan, and Keith D. Baker. Model-based localisation and recognition of road vehicles. **International Journal on Computer Vision**, 27(1):5–25, 1998.

Arne Theil, Rob A.W. Kemp, Katherin Romeo, Leon J.H.M. Kester, and Eloi Bosse. Classification of moving objects in surveillance algorithms. In **Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)**, 2000.

Tina Yu Tian, Carlo Tomasi, and David J. Heege. Comparison of approaches to egomotion computation. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 315–320, 1996.

D.M. Titterington. Recursive parameter estimation using incomplete data. **Journal of the Royal Statistical Society B**, 46(2):257–267, 1984.

Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages 255–261, 1999.

Yanghai Tsin, Visvanathan Ramesh, and Takeo Kanade. Statistical calibration of CCD imaging process. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages I:480–487, 2001.

Jakob J. Verbeek. **Mixture Models for Clustering and Dimension Reduction**. PhD thesis, University of Amsterdam, 2004.

Jessica J. Wang, Wei-Ming Hu, and Tie-Niu Tan. Recent developments in human motion analysis. **Pattern Recognition**, 36(3):585–601, 2003.

Jessica J. Wang and Sameer Singh. Video analysis of human dynamics-a survey. **Real-Time Imaging**, 9(5):321–346, 2003.

Eric W. Weisstein. Half–normal distribution. From MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/Half-NormalDistribution.html, 2005.

Natasha A.I.M. Weitenberg, E.J.M. Jansen, I. van Leiden, José H. Kerstholt, and H.B. Ferwerda. **Cameratoezicht: De menselijke factor Video surveillance, the human part, in Dutch**. Kerckebosch, Zeist, The Netherlands, 2003.

Paul J. Withagen, Eric den Breejen, Erik M. Franken, Arie N. Jong, and Hans Winkel. Band selection from a hyperspectral data-cube for a real-time multispectral 3CCD camera. In **Proceedings of the SPIE International Symposium on Aerospace/Defense Sensing, Simulation, and Controls AeroSense**, pages 84–93, 2001.

Paul J. Withagen, Frans C.A. Groen, and Klamer Schutte. EMswitch: a multi–hypothesis approach to EM background modelling. In **Proceedings of the Advanced Concepts for Intelligent Vision Systems Conference (ACIVS)**, pages 199–206, 2003.

Paul J. Withagen, Frans C.A. Groen, and Klamer Schutte. Global intensity correction in dynamic scenes. **International Journal on Computer Vision**, Under review, 2004a.

Paul J. Withagen, Frans C.A. Groen, and Klamer Schutte. CCD characterization for a range of color cameras. In **Proceedings of the IEEE Instrumentation and Measurement Technical Conference (IMTC)**, pages III:2232–2235, 2005a.

Paul J. Withagen, Frans C.A. Groen, and Klamer Schutte. Ccd color camera characterization for image measurements. **IEEE Transactions on Instrumentation and Measurement**, Under review, 2005b.

Paul J. Withagen, Frans C.A. Groen, and Klamer Schutte. Shadow detection using a physical basis. **International Journal on Computer Vision**, Under review, 2005c.

Paul J. Withagen, Klamer Schutte, and Frans C.A. Groen. Likelihood–based object tracking using color histograms and EM. In **Proceedings of the IEEE International Conference on Image Processing (ICIP)**, pages 589–592, 2002a.

Paul J. Withagen, Klamer Schutte, and Frans C.A. Groen. Object detection and tracking using a likelihood based approach. In **Proceedings of the annual conference of the Advanced School for Computing and Imaging (ASCI)**, pages 248–253, 2002b.

Paul J. Withagen, Klamer Schutte, and Frans C.A. Groen. Probabilistic classification between foreground objects and background. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages 31–34, 2004b.

Paul J. Withagen, Klamer Schutte, and Frans C.A. Groen. A probabilistic framework for object detection and tracking. **International Journal on Computer Vision**, Under review, 2005d.

Paul J. Withagen, Klamer Schutte, Albert M. Vossepoel, and Marcel G.J. Breuers. Automatic classification of ships from infrared (flir) images. In **Proceedings of the SPIE International Symposium on Aerospace/Defense Sensing, Simulation, and Controls AeroSense**, volume 3720, pages 180–187, 1999.

Christopher R. Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real–time tracking of the human body. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 19(7):780–785, 1997.

C.F. Jeff Wu. On the convergence properties of the EM algorithm. **Annals of Statistics**, 11 (1):95–103, 1983.

Binglong Xie, Visvanathan Ramesh, and Terrance E. Boult. Sudden illumination change detection using order consistency. **Image and Vision Computing**, 22(2):117–125, 2004.

Lei Xu and Michael I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. **Neural Computing**, 8(1):129–151, 1996.

Kemal B. Yesin and Bradley J. Nelson. Robust cad model based visual tracking for 3d microassembly using image space potentials. In **Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)**, pages II:1868–1873, 2004.

Wojciech Zajdel, Ali T. Cemgil, and Ben J.A. Kröse. Online multicamera tracking with a switching state–space model. In **Proceedings of the IEEE International Conference on Pattern recognition (ICPR)**, pages IV:339–343, 2004.

Ye Zhang and Chandra Kambhamettu. 3d head tracking under partial occlusion. **Pattern Recognition**, 35(7):1545–1557, 2002.

Yue Zhou and Hai Tao. A background layer model for object tracking through occlusion. In **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, pages 1079–1085, 2003.

Zoran Zivkovic and Ferdinand van der Heijden. A stabilized adaptive appearance changes model for 3d head tracking. In **Proceedings of the IEEE Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS)**, pages 175–181, 2001.

Zoran Zivkovic and Ben J.A. Kröse. An EM–like algorithm for color–histogram–based object tracking. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, pages 798–803, 2004.

Frank Zwarthoed. Real–time object detectie in video (real–time object detection in video, in Dutch). Master's thesis, University of Amsterdam, 2002.

Samenvatting*

Voor het bewaken van de veiligheid op straat en in publieke ruimten wordt de laatste jaren steeds vaker gebruik gemaakt van beveiligingscamera's. Het aantal camera's dat een beveiligingsmedewerker tegelijkertijd effectief kan bekijken is echter zeer beperkt. Het is onbetaalbaar om alle camera beelden te laten bekijken. De meeste beelden worden daarom opgeslagen en eventueel achteraf geraadpleegd indien zich een incident heeft voorgedaan. Hierdoor is directe actie niet mogelijk, wat de effectiviteit van de camera's sterk vermindert.

Met behulp van beeldverwerking algoritmes kunnen wel alle beelden geanalyseerd worden. Een computer algoritme maakt dan een selectie van mogelijk verdachte situaties, waarna de beveiligingsmedewerker alleen deze beelden hoeft te bekijken. Deze combinatie van camera en automatische verwerking van de beelden wordt ook wel een intelligente camera genoemd.

Dit proefschrift behandelt het automatisch vinden en volgen van mensen in camera beelden. Dit zijn de basis stappen voor een intelligente camera. De drie belangrijkste wetenschappelijke bijdragen van dit proefschrift betreffen een algoritme voor de correctie van verschillen in gemiddelde (belichtings-) intensiteit tussen beelden, het herkennen van schaduw op basis van de kleurverandering veroorzaakt door de schaduw en het op statistische wijze onderscheid maken tussen voorgrond en achtergrond.

Programmatuur voor intelligente camera's, maar ook veel andere programma's binnen en buiten de beeldverwerking, bestaan uit deelalgoritmen op verschillende abstractie niveaus. De verschillende delen zijn veelal beschikbaar in de literatuur, probleem is de onderlinge koppeling. In dit proefschrift wordt een flexibel raamwerk geïntroduceerd voor deze koppeling. Het raamwerk is gebaseerd op waarschijnlijkheden en zorgt voor een optimale classificatie op elk abstractie niveau. Een belangrijke toevoeging van het raamwerk is terugkoppeling, wat er voor zorgt dat

*Summary in Dutch

het meest complete wereldbeeld, beschikbaar op het hoogste abstractie niveau, wordt gebruikt voor het bijwerken van de modellen op de lagere niveaus.

Bij veel toepassingen van beeldverwerking, waaronder intelligente camera toepassingen, kan men weinig invloed uitoefenen op de belichting van de scène. Het gebeurd dan regelmatig dat de intensiteit tussen verschillende beelden niet hetzelfde is. Bijvoorbeeld door een wolk die voor de zon schuift of een camera die automatisch zijn versterking regelt als er mensen met lichte of donkere kleding in beeld komen. Dit geeft problemen met het vergelijken van beelden, onder anderen voor de detectie van bewegende objecten voor een achtergrond. In dit proefschrift wordt een algoritme beschreven waarmee voor zulke verschillen in intensiteit gecorrigeerd kan worden. Het betreft een dermate simpel algoritme dat het direct toepasbaar is met de huidige snelheid van computer hardware. Experimenten op echte beeldsequenties tonen aan dat dit algoritme een enorme verbetering geeft van de resultaten van object detectie en segmentatie.

Ook schaduwen vormen een uitdaging, en zijn niet te verwaarlozen in situaties waar geen invloed uitgeoefend kan worden op de verlichting. Schaduwen zorgen voor een lagere intensiteit, dit kan gebruikt worden voor het herkennen ervan. In veel gevallen echter, zorgt schaduw ook voor een kleurverschuiving. Dit wordt veroorzaakt doordat lichtbronnen met een verschillende kleur niet in de zelfde mate tegengehouden worden door een object. In dit proefschrift wordt een methode beschreven die uit de beelden het kleurverschil van de verlichting kan bepalen, en kan voorspellen wat de kleurverschuiving van schaduw zal zijn. Hierdoor kan een beter onderscheid worden gemaakt tussen voorgrond, achtergrond en schaduw.

Het modelleren van de achtergrond is een belangrijke eerste stap in het vinden van objecten. Bestaande aanpakken gebruiken daarbij een model dat gedeeld wordt door voor- en achtergrond. Dit leidt tot imperfecte classificatie resultaten. Bovendien zijn de parameters van het algoritme erg moeilijk instelbaar. In dit proefschrift wordt een aanpak gepresenteerd waarbij gescheiden modellen voor voor- en achtergrond gebruikt worden. Hierdoor is dit algoritme veel eenvoudiger in te stellen. Doordat de keuze tussen voor- en achtergrond is gebaseerd op waarschijnlijkheden zijn de classificatie resultaten beter, vooral in complexe situaties.

Een applicatie voor het vinden en volgen van objecten is ontwikkeld, op basis van het voorgestelde raamwerk. Naast de correctie voor verschillen in intensiteit en schaduwen wordt gebruik gemaakt van het verbeterde model van de achtergrond en een nieuw model van de voorgrond. De prestaties van dit algoritme zijn voor een aantal beeldreeksen vergeleken met de prestaties van een veelgebruikt referentie algoritme. Gemiddeld wordt de fout hierbij gereduceerd tot minder dan een tiende van de originele fout. Er wordt dus veel nauwkeuriger onderscheid gemaakt tussen bewegende objecten en achtergrond. Hierdoor wordt het mogelijk om ook in complexere situaties met meer bewegende objecten gebruik te maken van intelligente beveiligingscameras

De in dit proefschrift beschreven technieken vormen samen de basis voor een intelligente camera. Ze zijn echter ook op zichzelf bruikbaar in een breder toepassingsgebied.

Dankwoord*

Veel mensen hebben, direct of indirect, een bijdrage geleverd aan dit proefschrift of het achterliggende onderzoek. Voor allen die ik niet bij naam noem: Bedankt!

Grote dank ben ik verschuldigd aan mijn promotor Frans en co-promotor Klamer. Zonder hun inzet en begeleiding was dit proefschrift er nooit gekomen. De positieve houding van Frans en zijn heldere commentaar op zowel de grote lijn als de details zijn zeer belangrijk voor me geweest. Ook was hij altijd tot alle kwaad bereid bij het uitproberen van nieuwe dingen. De regelmatige discussies met Klamer leidden meestal tot meer vragen dan antwoorden. Dit heeft een enorme bijdrage geleverd aan zowel de kwaliteit van het werk als aan mijn persoonlijke ontwikkeling.

Ik wil mijn directe collega's bedanken voor hun interesse, tips en inhoudelijke discussies. John en Peter, als kamergenoten heb ik jullie het meest lastig gevallen met van alles en nog wat. Bas, Ronald, en bij mooi weer veel meer collega's, de dagelijkse lunchwandeling in de duinen zal ik erg missen. Sjaak, Rien, Ben en Leo, discussies met jullie maakten mijn bezoekjes aan de UvA altijd nuttig. Wannes, als "lotgenoot" hebben we heel wat tips uitgewisseld.

Een promotie onderzoek houdt niet op wanneer je naar huis gaat. Het is dan onmisbaar dat er vrienden om je heen zijn. Mensen die interesse tonen en met je mee denken, maar ook zorgen voor de nodige ontspanning. Brian en Victoire, op jullie kon ik altijd rekenen. Of het nu was voor een luisterend oor, een attent telefoontje, of lekker een weekendje weg.

Yvonne, Rob, Annemiek, mijn ouders en schoonouders wil ik bedanken voor de voortdurende steun. Zowel emotioneel als praktisch stonden ze altijd voor me klaar. En een weekendje Brabant voelt altijd als een beetje vakantie.

Jolanda, wat zal ik zeggen... Je was er gewoon altijd! Jij en Sanne wisten me in no time over dipjes te helpen, hoe kan je ook met je hoofd bij je werk blijven met zo'n ontvangst comité elke dag.

*Acknowledgements in Dutch

About the author

Petrus Johannes Withagen (Paul) was born on 19 January 1975 in Halsteren, the Netherlands. He received the MSc degree in Applied Physics from the Pattern Recognition group of the Delft University of Technology, the Netherlands in 1998. The subject of his MSc thesis was: ”Automatic classification of ships using forward looking infrared (FLIR) images. He then started working as a researcher in the Electro-Optics group of the TNO Physics and Electronics Laboratory in The Hague, the Netherlands. He worked on a number of image processing and pattern recognition projects, two examples of projects being scene-based non-uniformity correction and band selection for hyper-spectral images.

From 2001 until 2005 Paul worked on his PhD research: a collaboration between the Intelligent Autonomous Systems group of the University of Amsterdam and the Electro-Optics group of TNO Defense Security and Safety. The subject of the research was: ”Object detection and segmentation for visual surveillance”.

Pauls scientific interest include image processing and pattern recognition in general, and ”looking at people”, image-sequence improvement, automatic object classification and tracking in particular.