RuG

Object Tracking in Presence of Occlusion Using a Layered Image Representation

Rogier Falkena

August 18, 2006

Abstract - Daily life scenes are often recorded on video, for future analysis or to be watched at real-time. What makes those scenes of interest to the people who record them, are the objects moving around in them. Information gained from tracks travelled by the objects can be used in numerous application fields, amongst which are video compression, analysis of animal behavior, and, in the real-time case, video surveillance and traffic monitoring.

Tracking an object in a scene is an easy task for a human being. With the least of effort we follow the object around, even when its appearance or shape changes, or when it is temporarily out of sight. Amongst other problems, object occlusion is one of the hardest problems faced by an automated object tracking system. A lot of work is done on methods able to track objects through occlusions at real-time.

This Master's Thesis concentrates on object tracking in presence of occlusion. A promising method from the class of trackers using a layered image representation was selected and implemented, to be compared to a fast multi-resolution graph-based method developed at the University of Salerno. Comparison is done using a specially developed database of artificial test sets, which tests both trackers on specific basic and advanced events possibly occurring in daily life scenes. A well-known real-world test set is also used. Based on the outcome of the tests, suggestions are done for further research in the field of real-time trackers with occlusion handling.

Keywords - *object tracking, layered image representation, real-time, occlusion, image segmentation, image analysis, video analysis, motion estimation, expectation maximization, appearance model, shape estimation, Bayesian framework, multi-resolution graph, PETS 2001 test set.*

Author - Rogier Falkena (`rogierfalkena@gmail.com`) is a Computing Science graduate student with the group of Intelligent Systems of the Department of Computing Science at the University of Groningen, the Netherlands. The research for this Master's Thesis was conducted at the MIVIA department of the University of Salerno, Italy, under supervision of Professor Mario Vento.

Contents

1 Introduction

When a human being looks at a scene, the human visual system does not present a flat image but an arranged collection of objects. Our visual system provides us with such a segmentation in a very efficient way, that we are not even aware of it happening. We can easily assign meaning to the segmented objects we see and thus reason about one object being in front of another and the direction in which an object is moving. When an object disappears for a while and shows up again, we know that we are dealing with the same object without hesitation, and if the appearance of an object is different due to changing lighting conditions we still recognize it as the same object we have just seen in another light. The same holds for objects changing shape due to movement in the three dimensional world we see.

Although easy for human beings, presenting a computer with the task of object tracking in a scene over time, is all but trivial. Amongst other problems, *occlusion* is one of the hardest to cope with. When object A moves (partly) in front of object B, object B gets occluded. Human beings can easily conclude that the now partly visible B, is the same B that was completely visible before the occlusion. Computers have considerable more problems with making this association.

This Thesis will focus on object tracking in case of occlusion, using a layered image representation. A state-of-the-art tracking algorithm was selected and implemented, to be compared to a tracking method developed at the University of Salerno, Italy.

Object tracking will be introduced in section 1.1, section 1.2 will give a brief overview of the application fields in which object tracking and image segmentation play an important role, section 1.3 is about research previously done on the subject, in section 1.4 two promising tracking methods are compared and in section 1.5 the research goals of this Master Thesis are elaborated.

1.1 Image Analysis

Object tracking is part of the *image analysis* field. An object tracking system is generally made up of three independent layers, illustrated in figure 1.1. The first layer detects objects of interest present in a single frame and segments them from the background. The second layer performs the tracking of the objects, by establishing associations between them over a number of frames. The third layer uses the tracking results in a way useful to the application that the tracking scheme is part of. This last layer is highly dependent on the application at hand, and will be left out of consideration. The focus in this Thesis will be on the second layer, the first one will be regarded to a lesser extent. Techniques tend to have vague borders between the layers, especially the detection and tracking steps are often intertwined. Each layer will be briefly introduced below.

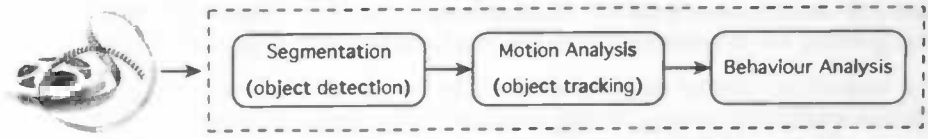


Figure 1.1: An object tracking system generally contains three independent layers.

1.1.1 Segmentation

Image segmentation is the partitioning of an image into a set of regions covering it. A goal of image segmentation is to identify meaningful parts of a scene and group their pixels. For example the segmentation of a soccer match image in which players are segmented from the grass on which they play. Once the parts are identified they can be processed in different ways, depending on the application at hand. In the soccer match example the segmented players could be input to object recognition software, that uses either their faces, shirt numbers or other features to identify them. Another goal of image segmentation can be to change the representation of an image. By organizing pixels into higher-level units a more efficient or meaningful representation can be acquired. Saving an image in a more efficient representation leads to a reduction of the image file size, while the more meaningful representation is needed to make further analysis possible. An example of a segmented image can be found in figure 1.2.

(a) *original* (b) *segmenatation*

Figure 1.2: Example of an image segmentation. (b) is a possible segmentation of (a) in which the background, the table, Garfield, and John are each seen as a separate region. *Image (a) and characters are ©Jim Davis.*

Image segmentation is an ill-posed problem: there is not always a solution and if there is one, the solution does not necessarily need to be unique. What is a correct segmentation depends on the application at hand. Given an image of an outdoor scene, a group of cars parked next to each other may be regarded as one object, but each car could also be seen as a separate object. Or, in even more detail, the license plate and windshield of a car maybe the targets to segment. This way of increasing the amount of detail to be segmented is applicable on many images and

illustrates the ill-posedness of image segmentation.

Segmentation of an image can be done in various ways. *Thresholding* and *edge finding* are among the most popular. The meaningful parts that need to be segmented are called *foreground* objects, the rest of the image is referred to as *background*. Thresholding uses either a fixed threshold or one based on the image histogram. All pixels with a brightness or color higher than the threshold are marked as foreground, the rest is background. Groups of foreground pixels can be regarded as separate objects. Techniques based on edge finding try to identify the contours of foreground objects using filters. A survey of a large number of techniques can be found in [HS85, SK94].

In case of a video sequence, multiple images are available in which a (stationary) background is present on which objects take up different positions from frame to frame. This presents the possibility to segment foreground objects by using their motion. To detect motion, that is, find groups of pixels moving together over multiple frames, three approaches exist [CFG+04]. In *temporal differencing* techniques a pixels belongs to the foreground if its difference in color or intensity between two consecutive frames is greater than some threshold. An obvious problem with this technique is that an object that stops moving becomes invisible. Slow moving objects with a uniform appearance also form a problem. Because of the overlap the object has with itself in the previous frame, this approach might not be able to detect all pixels belonging to the object. An advantage of temporal differencing is that it is insensitive for gradual changes in lighting conditions since those are negligible in consecutive frames.

Background subtraction techniques also use differences between pixels to detect objects. In this case it is not the previous frame that is used as a reference, but a frame in which no object are present. Such a frame is called a background frame and is often the first frame of an image sequence. Because objects are not present in the background frame, they will even be detected if they are moving at low speeds or if they are not moving at all. Gradual changes in lighting conditions can form a problem because of the substantial difference in time between the moment the background was captured and the current frame. Adaptive background update methods are developed to overcome this problem.

Previous techniques only establish that pixels are moving, what the actual motion is remains unknown. The *optical flow* group contains techniques that do estimate the velocity or optical flow of an object. To this end, a 2D motion field of the intensity pattern of an object is computed. An advantage of the optical flow approach is that foreground objects can also be detected in case of a moving camera. A performance comparison of different optical flow methods can be found in [BFBB94].

Over the years many segmentation techniques have been developed. Most of them apply to one specific application domain, a general method that works for all problems is yet to be discovered. When implementing an application which makes use of image segmentation, domain knowledge has to be used to choose the most optimal methods available.

1.1.2 Motion Analysis

After the segmentation is done, the motion of the objects found is analyzed. The motion analysis step is responsible for the actual *tracking* of objects throughout the video sequence.

Object tracking uses motion or *temporal* information to keep track of objects. As will become clear in section 1.3, using only temporal information is often not enough for reliable tracking.

Information found in the *spatial* domain is used to make tracking more robust. For example shape and/or appearance information of an object can be incorporated in the tracking scheme.

Difficulties arising when analyzing motions of objects in outdoor scenes, are formed by local lighting conditions like shadows and reflections, and global lighting conditions dictated by clouds covering and uncovering the sun. Those sudden changes in lighting conditions result in an incorrect detection of moving objects (e.g. its shadow is seen as part of the object), thereby influencing the tracking of those object over time. Another class of problems is formed by parts of the background that cause a false foreground object detection, like opening doors and waving trees. Solutions exist, but fall outside the scope of this Thesis.

The goal of the tracking step is to establishing correspondences between (parts of) objects over multiple consecutive frames. This can be done in different ways which will be briefly explored in section 1.3. The biggest problem in tracking is formed by occlusions among objects, either partly or complete. In case of an occlusion two (or more) objects known to be separate by the tracker are suddenly merged into one, while possibly having two (or more) different motions. The tracker has to establish a connection between the separate objects from the previous frame and the new object formed by the occlusion in the current frame.

1.1.3 Behavior Analysis

Once the tracks of objects are determined by the tracker, they can be analyzed. The behavior analysis step can be used to perform many different tasks, varying from very simple data processing like counting the number of unique objects present over time and determine the distances they travelled, to complex learning or classification jobs.

1.2 Application Fields

Information gained from the tracking of objects and the image segmentation this is build upon, is useful in many different fields. A brief explanation of some of the possibilities is given. This overview is by no means extensive.

For a good user experience, video broadcasting over the internet demands high quality video at a low bit rate. Motion segmentation provides *compression* techniques with the possibility to apply different coding strategies to different elements in a scene. For example, relevant (foreground) objects can be coded at a higher bit rate than a non-changing background. An other possibility is to describe a sequence by only using the segmentation of the first frame and describe the rest of the sequence frames with motion vectors applied on those segments.

Fast access to video databases is of growing importance for professional and personal tasks. Video *indexing* or *annotation* is the process of attaching content based labels to video sequences, making it possible to retrieve video based on its content. Video indexing is performed in three steps [GB98]. The first step segments the sequence into shots, the second step uses motion-based segmentation to identify foreground objects, and the third step involves the tracking of those objects. The second steps makes it possible to analyze and classify background and foreground objects separately.

Various applications of *video surveillance* are made possible by a real-time object tracking system. Behavior of animals could be monitored without collars or tags, traffic observation could be

done without magnetic loops embedded in the highway and burglars could be caught and followed on camera while wandering around the premises. This field is relatively new, since it is only recent that inexpensive hardware has the minimal computing power needed for real-time tracking.

1.3 Previous Research

Though independent, the image segmentation and object tracking layers of a tracking scheme build very strong upon each other. The border between where segmentation stops and tracking begins, and if tracking is done based on segmentation or the other way around, differs from method to method. In the upcoming overview on previous research in the object tracking field, segmentation refers to both the segmentation and tracking layers.

The first segmentation methods where based on dense optical flow. Optical flow considers motion in a visual representation by letting a vector originate from, or point to a pixel in a digital image sequence. Dense optical flow means that each pixel has a vector describing its motion. The idea of segmenting an image into multiple overlapping layers was introduced by Wang and Adelson [WA93]. Their method represents each object in a layer that describes the object's motion, texture pattern, shape and opacity. The parameters of the motion model are fit to optical flow estimates over small initial regions of an image, which are subsequently merged using k-means clustering.

To be able to segment images based on motion, optical flow algorithms assume that the motion is modeled by a low dimensional parametrization. The most used models are the six parameter affine model and the eight parameter projective model, both corresponding to rigid motion in the plane. The affine model handles motion in an orthographic projection, a parallel projection of a 3D scene onto the perpendicular plane, and the projective model handles motion in perspective projections. The problem with using a parametrization is that if the dimension is low, the model will be too restrictive to handle more complex motions, and if the dimension is high, the model estimation is unstable. Weiss [Wei97] developed further upon [WA93] and presented a nonparametric model which uses a motion smoothness favoring prior for each layer.

The known difficulty with optical flow based methods is their limited ability to handle a large motion between frames and objects with overlapping motion fields [WAB06]. Coarse-to-fine methods are developed to overcome the large motion problem and they manage to do so to some extent. The largest manageable movement between two frames is about 15% of the frame dimensions [IA99].

The techniques mentioned above try to segment an image by exclusively using motion information and belong to the group of *motion-based* methods. However, there is more information to be found in an image. For example in the intensity or color of a pixel and the here from originating image regions. The group of *spatio-temporal* techniques makes use of those features. Those techniques use the same motion estimation schemes as the motion based ones, but use spatial information to guide this estimation.

Classification of image segmentation techniques is inconsistent and varies from author to author. Zhang and Lu [ZL01] suggest the two previously mentioned groups of techniques: motion-based and spatio-temporal. Motion-based techniques are split into two subgroups: *2D models* and *3D models*, based on the motion model used. It is also possible to classify motion based

techniques on their actual segmentation criteria. However, motion models play a more important role in this class and are generally what the design of an algorithm is built upon. Clustering criteria are used to distinguish techniques in the 2D and 3D subgroups.

The group of spatio-temporal segmentation techniques is relatively new compared to the motion-based group. They are grouped according to the temporal and spatial methods used. The temporal subgroup is similar to the classification of the motion-based techniques. The spatial subgroup can be divided into *region-based* and *contour-based* methods. A brief explanation of different methods along with a comparison is given in [ZL01]. An introduction on image segmentation can be found in [SS01].

1.4 Two Promising Methods Compared

The first part of the research for this Thesis consisted of a suitability comparison between the *Bayesian estimation* method of Tao *et al.* [TSK02] and the *edge tracking* method of Smith *et al.* [SDC04]. Both methods use a layered image representation and were suggested by Donatello Conte, currently working on his Ph.D. Thesis at the MIVIA department (section 1.5). The goal was to select the method best capable of tracking objects in real-time in low-resolution video sequences.

The Bayesian method continuously estimates a dynamic representation of the layers, based on the representation of the previous time instant and the current sequence frame. The representation consists of three parameters; a motion model, a segmentation prior, and the appearance of each layer. The motion model assumes objects to have a simple 2D rigid transformation and uses a constant velocity model. The uncertainty in motion is modeled by a Gaussian distribution. The segmentation prior represents shapes of foreground layers with a Gaussian distribution. The background layer has a constant prior. The segmentation prior distribution is normalized over all layers. The appearance model of a layer holds color information of the object it represents. The appearance model is dynamically updated over time.

Expectation Maximization [DLR77] is used to improve the representation estimates. Each turn one of them is improved with the other two fixed. Change blobs (color differences between consecutive frames) are used to detect moving objects. Those change blobs together with the layered image representation are input to a state machine that determines the state of each object.

The positive aspects of the Bayesian method are, according to Tao, its speed and confident tracking of multiple objects in complex scenes. The method is able of handling two objects at a ten frames per second rate, and four objects at five frames per second. Negative aspects are the inability of handling camera zoom, its optimization for bird-view use, the simple representation of objects, the absence of depth ordering, and the fact that the author used dedicated hardware for his tests.

The edge tracking method estimates motion of objects based on the movement of their edges. Smith explains his method based on a two frames, two motions case. Next this basic case is extended to multiple frames and multiple motions. The method first finds the edges in a frame by using Canny edge detection [Can86] and subsequently groups them into chains. Next, the motion of each edge is determined at sample points in the direction of the edge normal. The motion is modeled by a 2D affine transformation and uses constant velocity model.

Prior to the Expectation Maximization, edges are randomly divided into two groups. The motion of each group is determined based on the motions of their respective edges. The expectation step relabels edges according to how well they fit each of the two group motions. The maximization step determines the new group motions, based on the new edge labeling. After determining the edge labeling, the frame is divided in regions of similar color, using the edges as hard barriers. A motion label is determined for each regions based on the edges belonging to it. The region labeling is optimized using simulated annealing [KGV83].

Using multiple frames can resolve ambiguities that maybe present in between two frames and result in a more robust labeling, since the initial motion and edge labeling can be based on the previous frame. The extension of the edge labeling method to three or more motions is nontrivial. The more motions edges can be assigned to, the less information is available with which to estimate each motion and the less certain the assignment of an edge to a motion. The expectation maximization has many local maxima when using multiple motions, making an accurate initialization is necessary. The solution Smith suggest requires multiple EM runs, e.g. seven runs for fitting three motions.

Positive aspects of the method are that motions of regions can solely be determined from edge motions and its capability of tracking complex shapes. The downside is that the algorithm requires eight seconds per frame in case of two motions and three minutes in case of three motions. Furthermore textures and reflections form a problem and the provided layer ordering is unreliable.

For the task of real-time object tracking in low resolution video the Bayesian method seems the best choice, because it appears to be suitable for real-time tracking and it appears to be capable of confidently tracking multiple objects. The coarse object representation used by the method is of no problem, because the resolution of the video is very low and thus object are not displayed in great detail. The Edge method is unsuitable for the task at hand because is appears to be to slow for real-time application and cannot confidently track more than two object. Its capability of determining the exact shape of an object makes this method most useful in for example video compression.

1.5 Research and Trajectory

The method with the highest real-time tracking potential, the Bayesian method of Toa *et al.*, was to be implemented for comparison with the *graph-based* method of Conte [CFJV05]. The former was developed for the Sarnoff Corporation, for one of its commercial aerial tracking products. The latter is developed within the *Gruppo di Ricerca su Macchine Intelligenti per il riconoscimento di Video, Immagini e Audio*, the research group of Intelligent Machine recognition of Video, Image and Audio (MIVIA), at the University of Salerno, Italy, where the research for this Master's Thesis was also conducted. The main comparison focus was on the occlusion handling capabilities of both methods.

During the course of implementation of the Bayesian method significant details seemed to be omitted from the paper and the promised real-time tracking was not met by far. Several speed optimizations were incorporated along the way. Due to the large amount of time spent on the implementation part, unfortunately no effort could be directed to occlusion handling improvement.

This Thesis describes the theory of the implemented method in chapter 2, theory needed for the implementation but not discussed by Tao is to be found in chapter 3, chapter 4 will go into

implementation details, test sets used for the comparison and the result of the conducted tests are elaborated in chapter 5, chapter 6 presents a short discussion and chapter 7 will conclude and suggest improvements on the implemented method.

2 Bayesian Method

The main idea of layer-based motion analysis is to estimate both the motion and segmentation of independent moving objects simultaneously, based on motion coherency across images. Each layer possesses a coherent two-dimensional motion that can be modeled in different ways. Starting from an initial solution, the motion and the segmentation are iteratively estimated. From the estimated segmentation the motion is refined and from the estimated motion a better segmentation is computed.

Object Tracking with Bayesian Estimation of Dynamic Layer Representations of Tao *et al.* [TSK02] introduces a complete dynamic layer representation in which spatial and temporal constraints on shape, motion, and layer appearance are modeled and estimated in a maximum a posteriori framework using the generalized Expectation Maximization algorithm. This representation is continuously estimated and updated over time.

The combination of temporal coherency of motion layers and the domain constraints on shapes have not been exploited before. The main new ideas presented by Tao *et al.* are:

- **Global shape constraint** - A new global shape constraint is used that incorporates a priori knowledge about the shapes of objects in the estimation process. The constraint consists of a two parameter shape prior which main purpose is to prevent shapes from transforming into arbitrary shapes. Due to the use of only two parameters to represent shape, computational complexity is limited.
- **Tracking with complete representation** - Tracking is done with a complete layer representation, taking appearance, motion, shape and segmentation into account .
- **Expectation Maximization** - The use of a generalized Expectation Maximization algorithm to estimate and update the dynamic layer representation over time.

The methods and models used for the Bayesian method will be explained in section 2.1, section 2.2 will deal with initialization questions and the determination of object states, and section 2.3 will briefly describe the tests conducted by the author.

2.1 Methods and Models

In this section the methods and models used by Tao will be elaborated. Methods necessary for the implementation of the algorithm but not explained by Tao, will be dealt with in chapter 3.

2.1.1 Dynamic Layer Representation

A dynamic layer representation at any time instant t is proposed as $\Lambda_t = (\Phi_t, \Theta_t, A_t)$, where Φ_t is the shape prior, Θ_t is the motion model, and A_t is the layer appearance. This representation

is continuously estimated based on its value Λ_{t-1} at the previous time instant and the current image observation I_t. The dynamic layer estimation problem is defined as finding the *maximum posterior probability* (MAP)

$$\max_{\Lambda_t} \arg P(\Lambda_t | I_t, \ldots, I_0, \Lambda_{t-1}, \ldots, \Lambda_0). \tag{2.1}$$

By using the Markovian assumption and Bayes' rule, this can be simplified to

$$\begin{aligned} & \max_{\Lambda_t} \arg P(\Lambda_t | I_t, \ldots, I_0, \Lambda_{t-1}, \ldots, \Lambda_0) \\ = & \max_{\Lambda_t} \arg P(\Lambda_t | I_t, I_{t-1}, \Lambda_{t-1}) \\ = & \max_{\Lambda_t} \arg P(I_t | \Lambda_t, I_{t-1}, \Lambda_{t-1}) P(\Lambda_t | I_{t-1}, \Lambda_{t-1}), \end{aligned} \tag{2.2}$$

where $P(I_t|\Lambda_t, I_{t-1}, \Lambda_{t-1})$ is the likelihood function and $P(\Lambda_t|I_{t-1}, \Lambda_{t-1})$ is the dynamic model of the state Λ_t. The Markovian assumption states that state Λ_t depends on all of the previous states, but because of how the probabilities tend to zero, it relies most heavily on the most recent state Λ_{t-1}. Hence probability P can be found leaving $I_{t-2}, \ldots, I_0$ and $\Lambda_{t-2}, \ldots, \Lambda_0$ out.

$$P(A|BC) = \frac{P(A|C)P(B|AC)}{P(B|C)} \tag{2.3}$$

Bayes' rule (equation 2.3) says that if A is one of several explanations for the new observation B and C summarizes all prior assumptions and experience, the probability of A in case of C should be adapted to A in case of B and C. Explanation A needs to be in such a way that together with the prior assumptions and experience C it fixes $P(B|AC)$. In the light of dynamic layer representation prior probability $P(A|C)$ is $P(\Lambda_t|I_{t-1}, \Lambda_{t-1})$ and posterior probability $P(B|AC)$ is $P(I_t|\Lambda_t, I_{t-1}, \Lambda_{t-1})$. $P(B|C)$ serves solely as a normalization factor and is therefore not mentioned in equation 2.2.

Thus the maximum is sought of the posterior probability of the current image observation I_t given the current layered image representation Λ_t and previous knowledge, multiplied by the posterior probability of current representation Λ_t given the previous knowledge. A solution to equation 2.2 can be found using Expectation Maximization, as explained in section 2.1.5.

With the proposed dynamic layer representation scenes as found in motion video can completely be described. The complete description of a scene can be analytically formulated and dynamically estimated. Details will be discussed in sections 2.1.2 to 2.1.9.

2.1.2 Motion Model

The motion model of a layer describes its coherent motion. Several motion models exist, each with a specific set of parameters to deal with different kinds of situations.

- **Translation** - A two-parameter model capable of handling x and y translation in the 2D plane.
- **Rigid** - A three-parameter model which uses the translation along the x and y axes and the rotation angle.

- **Rigid and Scale** - A four-parameter model which makes use of a 2D translation vector, rotation angle, and a scaling factor.
- **Affine** - A six-parameter model describing rotation about the optical axis, zoom, translation and shear.
- **Projective** - An eight-parameter model which extends the affine motion model with pan and tilt angles.

Object tracking in aerial videos involves two kinds of motions. Tao models the ground plane motion with a projective motion. The motion of foreground layer j on time instant t is modeled by a *2D rigid motion*, using displacement vector $\vec{\mu}_{t,j} = [x, y]^T$ and rotation $\omega_{t,j}$. The rigid motion model is derived from the more complex affine model and is with its three parameters conveniently compact.

The motion parameters for layer j are denoted by $\Theta_{t,j} = [\vec{\mu}_{t,j}^T, \omega_{t,j}]^T$. To model the dynamic behavior of a layer over time, a *2D constant velocity model* is used. Usage of such a model is possible in the case of traffic monitoring, because vehicles tend to move at constant speeds. Given the motion at the previous time instant $\Theta_{t-1,j}$, the current motion of layer j is described by a Gaussian distribution

$$P(\Theta_{t,j}|\Theta_{t-1,j}) = \mathrm{N}(\Theta_{t,j} : \Theta_{t-1,j}, diag[\sigma_{\vec{\mu}}^2, \sigma_{\vec{\mu}}^2, \sigma_{\omega}^2]), \tag{2.4}$$

where $\sigma_{\vec{\mu}}^2$ and σ_{ω}^2 in covariance matrix $diag$ represent the model uncertainty in translation and rotation. The variances in $diag$ are on the diagonal of the matrix, with the other cells being 0:

$$diag[\sigma_{\vec{\mu}}^2, \sigma_{\vec{\mu}}^2, \sigma_{\omega}^2] = \begin{bmatrix} \vec{\mu}^2 & 0 & 0 \\ 0 & \vec{\mu}^2 & 0 \\ 0 & 0 & \sigma_{\omega}^2 \end{bmatrix}$$

$N(x : m, s^2)$ denotes a normal distribution for a random variable x with mean m and variance s^2 (s being the standard deviation):

$$f(x) = \frac{1}{s\sqrt{2\pi}} \exp\left[-\frac{(x-m)^2}{2s^2}\right]$$

2.1.3 Dynamic Segmentation Prior

A *dynamic Gaussian segmentation prior* is proposed which encodes the domain knowledge that foreground objects have compact shapes. The dynamic prior is modeled such that gradual changes over time are allowed. Tao motivates the use of such a global shape assumption in two ways. In the first place, because the prior prefers Gaussian like shapes, it prevents foreground objects from evolving into arbitrary shapes in the course of tracking and thus it prevents the tracker for working with ambiguous or cluttered measurements. Secondly, because only the compact parametric form of the shape prior needs to be estimated, efficient computation is possible.

The parametric representation of the segmentation is only used as a compact way to represent shapes in motion. The actual segmentation at each time instant is provided by the layer ownership which combines the segmentation prior with an observation model (section 2.1.4). Layer

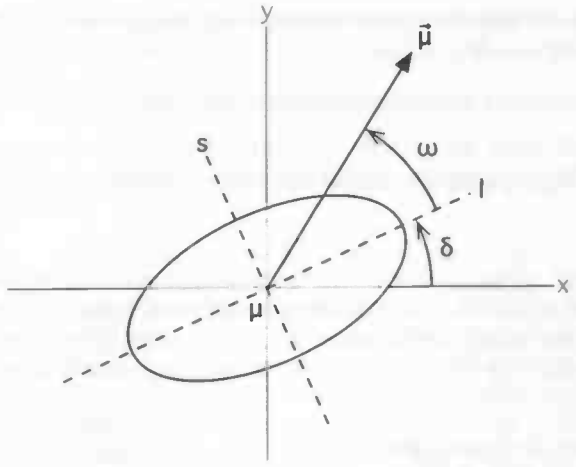


Figure 2.1: Motion model with translation $\vec{\mu}$ and rotation ω. μ is the center of the distribution, δ the angle the distribution makes with the image coordinate system, and l, s are parameters of the shape prior as explainded in section 2.1.3. Note that distribution center μ is placed in the center of the image coordinate system for explaining purposes only. μ is the center of the distribution as it appears in the image.

ownership will be discussed in section 2.1.6. As a result only the shape prior parameters have to be carried over time for each foreground layer to represent its shape.

When dealing with vehicle tracking from airborne platforms, the dominant region in the scene is the ground. Its motion can either be modeled with a projective motion, or, in case of a stationary camera, motion of the background is 0. The segmentation prior function for each pixel belonging to the ground layer is a constant value β. Moving objects are foreground layers and their segmentation prior function is modeled as a Gaussian distribution.

Suppose current image I_t has g motion layers with layer 0 being the background, then the segmentation prior function for pixel p_i belonging to layer j is defined as

$$L_{t,j}(x_i) = \begin{cases} \gamma + \exp\left[-(p_i - \mu_{t,j})^T \Sigma_{t,j}^{-1}(p_i - \mu_{t,j})/2\right] & \text{j = 1,...,g-1} \\ \beta & \text{j = 0} \end{cases} \tag{2.5}$$

where γ is the uncertainty of the layer shape, $\mu_{t,j}$ is the center of the segmentation prior distribution (as it appears in the coordinate system of I_t) and $\Sigma_{t,j}$ is the covariance matrix defining the span of the distribution. $\Sigma_{t,j}$ is defined as

$$\Sigma_{t,j} = R^T(-\delta_{t,j}) diag[l_{t,j}^2, s_{t,j}^2] R(-\delta_{t,j}), \tag{2.6}$$

where $l_{t,j}$ and $s_{t,j}$ are proportional to the lengths of the major and minor axis of the distribution's iso-probability contour and thus describe the shape of each foreground layer. $\delta_{t,j}$ is the angle of the distribution's major axis with respect to the coordinate system of I_t. Figure 2.1 shows these parameters. Σ^{-1} denotes the inverse of Σ and is calculated as

$$\Sigma^{-1} = \frac{1}{|\Sigma|} \begin{bmatrix} \Sigma_{2,2} & -\Sigma_{1,2} \\ -\Sigma_{2,1} & \Sigma_{1,1} \end{bmatrix} \tag{2.7}$$

Since a matrix only has an inverse if its determinant is not zero, l and s must both be greater than zero. R is the rotation matrix which is defined for vector-rotation as

$$R(\delta) = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \tag{2.8}$$

with $R^T(\delta) = R(-\delta)$. The *shape prior* parameter for layer j at time instant t is denoted as $\Phi_{t,j} = [l_{t,j}, s_{t,j}]$.

Figure 2.2 shows a cross-section of function 2.6 applied on the background layer and a single foreground layer. A consequence of the simple Gaussian shape model is that pixels with larger distances to any foreground layer center will have a higher prior of belonging to the background. This is compensated for with constant value γ, which allows pixels to belong to a foreground layer even though they are far away from an foreground layer center. Uncertainty of shape γ is important because the shape of an object is seldom perfectly elliptic and may also change over time.

The *normalized prior distribution* is calculated as

$$S_{t,j}(p_i) = \frac{L_{t,j}(p_i)}{\sum_{j=0}^{g-1} L_{t,j}(p_i)} \tag{2.9}$$

To describe the dynamic behaviour of the shape prior *constancy of shape* is used. Shapes of objects stay fairly constant over time, because the airborne platform changes its altitude slowly and only a small amount of camera zoom is used. The constancy of shape over time is modeled as a Gaussian distribution

$$P(\Phi_{t,j}|\Phi_{t-1,j}) = \mathrm{N}(\Phi_{t,j}, \Phi_{t-1,j}, diag[\sigma_{ls}^2, \sigma_{ls}^2]), \tag{2.10}$$

where σ_{ls} is the uncertainty of the model.

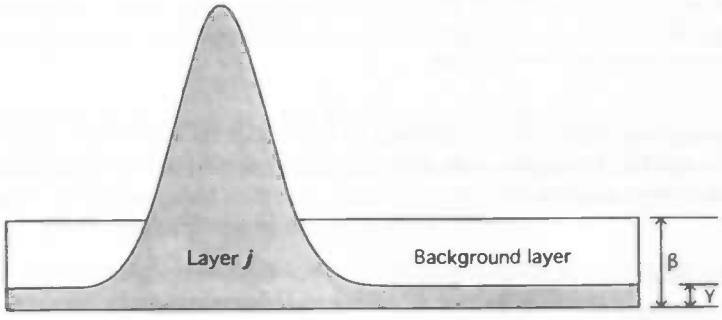


Figure 2.2: $L_{t,j}$ (equation 2.6) for a background layer and one single foreground layer. β is the constant prior for the ground layer, γ represents the uncertainty of the layer shape.

2.1.4 Image Observation Model and Dynamic Layer Appearance Model

The *appearance* of layer j at time instant t is denoted as $A_{t,j}$. The appearance image of a layer is defined in its own local coordinate system by the center and axis of the segmentation prior

distribution. The relation between pixel p_i in layer j of original image I_t and its equivalent q_i in appearance model $A_{t,j}$ is defined as

$$q_i = R(-\delta_j)(p_i - \mu_j), \tag{2.11}$$

with μ_j being the center of the segmentation prior and δ_j the angle the segmentation prior makes with the coordinate system of I_t (figure 2.3). For any pixel p_i in the original image, the *observation model* for layer j is

$$P(I_t(p_i)|A_{t,j}(q_i)) = \mathrm{N}(I_t(p_i) : A_{t,j}(q_i), \sigma_I^2), \tag{2.12}$$

where variance σ_I^2 accounts for the noise in image intensity. Basically, the observation model says how well the appearance model of layer j fits on current image I_t, based on the intensities of p_i and q_i. To this end, the appearance model is *warped* from its local coordinate system to that of the current image, using distribution center μ and distribution angle δ. The observation model gives the probability that p_i given q_i for each pixel p_i in layer j. The lower σ_I^2, the better the appearance model needs to fit on the current frame to give high output.

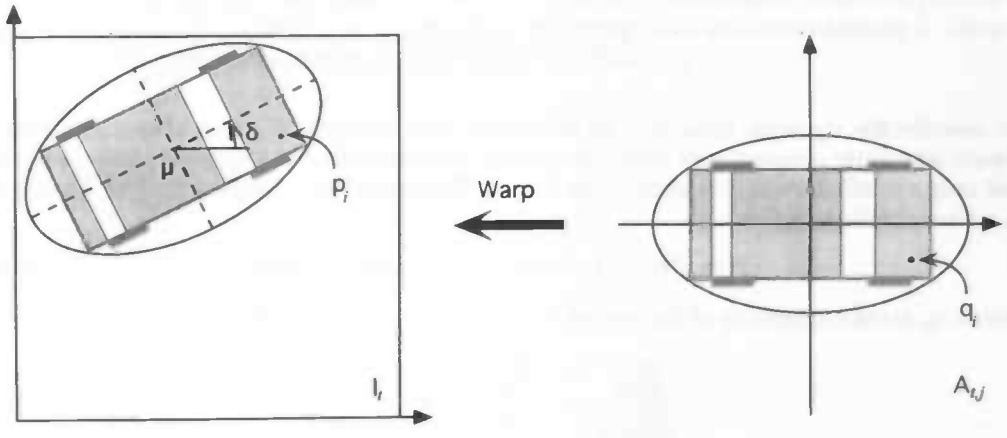


Figure 2.3: Appearance model $A_{t,j}$ is defined in its own local coordinate system. The appearance model is warped onto the original image I_t using distribution center μ and distribution angle δ.

The appearance model is a representation of a layer in its own coordinate system. Since moving objects do not take up the whole image, a layer is likely to consist of a subset of pixels from the original image. An observation model of a layer is constructed using all pixel coordinates of the original image, even tough there are pixels which do not belong to the layer. Which values can best be used in this situation will be discussed in section 3.2.

Appearance of the background layer and foreground layers can change over time. The *dynamic layer appearance model* copes with this a priori knowledge. By letting the intensity of a pixel belonging to layer j be a Gaussian distribution, the model is defined as

$$P(A_{t,j}(q_i)|A_{t-1,j}(q_i)) = \mathrm{N}(A_{t,j}(q_i) : A_{t-1,j}(q_i), \sigma_A^2), \tag{2.13}$$

where σ_A^2 is the appearance model uncertainty variance that accounts for layer appearance changes over time.

2.1.5 Expectation Maximization

The goal of the algorithm is to find the dynamic layer representation Λ_t for each time instant t, thereby fulfilling the dynamic layer estimation problem that was defined in equation 2.2. At each time instant t a new segmentation has to be estimated and layer parameters have to be updated. So the algorithm has to establish the correspondence between pixels and layers (segmentation) and compute the optimal parameters for each layer. *Expectation Maximization* (EM) [DLR77] can be used to achieve both goals.

EM provides an iterative scheme for obtaining maximum likelihood estimates by replacing a hard to solve problem by a series of smaller, simpler problems. The algorithm is useful in cases where missing or hidden data is involved. Each iteration of the EM algorithm consists of two steps. In the E-step (expectation step) hidden data are estimated based on the observed data and the current estimate of the model parameters. This is achieved using *conditional expectation*: estimate one parameter, with the other parameters fixed. In the M-step (maximization step) the likelihood function is maximized under the assumption that the hidden data are known. The estimate of the hidden data from the E-step are used in stead of the actual missing data.

At every iteration of the EM process, the estimated model parameters provide an increase in the likelihood function. The process will continue until a local maximum is reached, at which the likelihood function cannot increase further, but will not decrease either. There is no guarantee that the found maximum is also the global maximum. For likelihood functions with multiple maxima, EM will converge to a local maximum depending on its starting point. Since EM is guaranteed to increase the likelihood estimates at each iteration, convergence is assured. For more details on convergence of the EM algorithm see [MK96].

Derived from the EM algorithm is the *generalized EM algorithm* (GEM). The main difference is that the goal of generalized EM is to simply increase and not necessarily maximize the likelihood estimates. GEM is useful in situations where maximization is difficult.

In Tao's algorithm, the actual layer segmentation is the hidden data EM makes use of. Using GEM, a local maximum likelihood estimate can be achieved by iteratively optimizing with respect to Λ_t

$$Q = E\big[\log P(I_t, z_t|\Lambda_t, \Lambda_{t-1}, I_{t-1})|I_t, \Lambda'_t, \Lambda_{t-1}, I_{t-1}\big] + \log P(\Lambda_t|\Lambda_{t-1}, I_{t-1}), \tag{2.14}$$

where hidden variable z_t is the layer segmentation that associates each pixel to one of the layers and Λ'_t is the result of the previous iteration. Prove that the dynamic layer estimation problem (equation 2.2) can be written as Q can be found in [TSK02].

Let n be the number of pixels and g the number of layers (with $g = 0$ being the background layer), then

$$\begin{aligned}
&\sum_{i=0}^{n-1}\sum_{j=0}^{g-1} h_{i,j}\{ \quad \log S_{t,j}(p_i) + \log P\big(I_t(p_i)|A_{t,j}(q_i)\big) \quad \}+ && \text{(equation 2.9, equation 2.12)}\\
&\quad \sum_{j=1}^{g-1}\{ \quad \log \mathrm{N}(\Phi_{t,j}, \Phi_{t-1,j}, diag[\sigma_{ls}^2, \sigma_{ls}^2])+ && \text{(shape: equation 2.10)}\\
&\qquad \log \mathrm{N}(\Theta_{t,j} : \Theta_{t-1,j}, diag[\sigma_{\vec{\mu}}^2, \sigma_{\vec{\mu}}^2, \sigma_{\omega}^2])+ && \text{(motion: equation 2.4)}\\
&\qquad \sum_{i=0}^{n-1} \log \mathrm{N}(A_{t,j}(q_i) : A_{t-1,j}(q_i), \sigma_A^2) \quad \} && \text{(appearance: equation 2.13)}
\end{aligned} \tag{2.15}$$

where $h_{i,j}$ is the *layer ownership*, the posterior probability of pixel p_i belonging to layer j given Λ'_t. Note that shape, motion and appearance are only estimates for foreground ($j > 0$) layers. Prove that Q is equivalent to equation 2.15 can be found in [TSK02]. The actual segmentation of image I at time instant t can be derived from $h_{i,j}$ by assigning each pixel to the layer for which its ownership value is maximal. During computation, this actual segmentation is not used.

It is difficult to optimize shape Φ_t, motion Θ_t and appearance A_t from equation 2.15 simultaneously. Therefore each of them is improved in turn with the other two fixed, as the general approach of Expectation Maximization suggests. A graphical representation of the EM process can be found in figure 2.4. Motion parameters are estimated first, followed by the shape prior and appearance model. After re-estimation of each of the parameters of Λ, ownership $h_{i,j}$ is updated. The EM process can consist of multiple iterations. Initial input to the EM process is the layered image representation at the previous time instant Λ_{t-1} and current image I_t. Output is the layered image representation at the current time instant Λ_t.

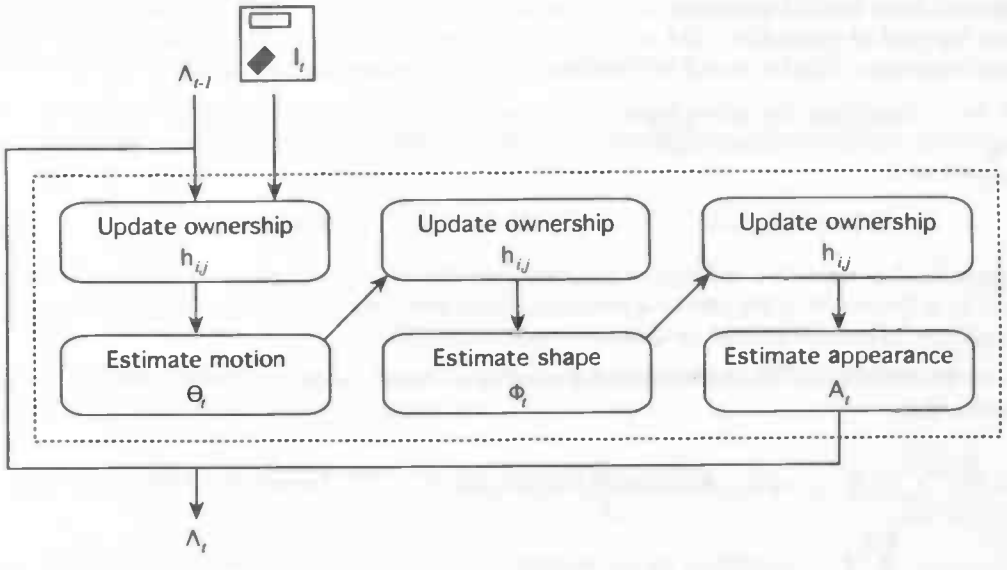


Figure 2.4: The Expectation Maximization process to estimate layered image representation Λ_t. Input for EM on time instant t are the estimates of the previous time instant Λ_{t-1} and current image I_t.

2.1.6 Layer Ownership

Layer ownership $h_{i,j}$ is the posterior probability of pixel p_i belonging to layer j conditioned on Λ'_t and can be derived using Bayes' rule in a similar manner as in section 2.1.1:

$$\begin{aligned} h_{i,j} &= P\big(z_t(p_i) = j | I_t, \Lambda'_t, \Lambda_{t-1}, I_{t-1}\big) \\ &= \frac{P\big(I_t | z_t(p_i) = j, \Lambda'_t, \Lambda_{t-1}, I_{t-1}\big) P\big(z_t(p_i) = j | \Lambda'_t, \Lambda_{t-1}, I_{t-1}\big)}{P\big(I_t | \Lambda'_t, \Lambda_{t-1}, I_{t-1}\big)} \\ &= \frac{P\big(I_t(p_i) | A'_{t,j}(q_i)\big) S_{t,j}(p_i)}{Z}. \end{aligned} \tag{2.16}$$

The normalization Bayes' rule introduces is carried out by Z, so that $\sum_{j=0}^{g-1} h_{i,j} = 1$. The first term is the observation model (equation 2.12) that measures how well current image I_t fits the appearance of layer j. The second term is the segmentation prior (equation 2.9) that describes the prior probability of pixel p_i belonging to layer j. Ownership $h_{i,j}$ is thus influenced by both appearance and shape of layer j.

2.1.7 Motion Estimation

If shape prior Φ_t and appearance A_t are known, motion parameter Θ_t can be estimated. Given ownership $h_{i,j}$, current image I_t with g layers, n pixels and for each foreground layer j an appearance model A_t and a shape prior Φ_t, motion estimation finds the Θ_t that improves

$$\begin{aligned} &\sum_{j=1}^{g-1} \log \mathrm{N}\big(\Theta_{t,j} : \Theta_{t-1,j}, diag[\sigma_{\vec{\mu}}^2, \sigma_{\vec{\mu}}^2, \sigma_{\omega}^2]\big) + \\ &\sum_{i=0}^{n-1} \sum_{j=1}^{g-1} h_{i,j} \big\{ \log S_{t,j}(p_i) + \log P\big(I_t(p_i) | A_{t,j}(q_i)\big) \big\}. \end{aligned} \tag{2.17}$$

Note that this function leaves out the background layer, its motion is handled outside the EM process. The motion estimation for each individual foreground layer is derived from equation 2.17 as

$$\begin{aligned} \min_{\Theta_{t,j}} \arg \; &\left(\frac{|\vec{\mu}_{t,j} - \vec{\mu}_{t-1,j}|}{\sigma_{\vec{\mu}}^2} + \frac{|\omega_{t,j} - \omega_{t-1,j}|}{\sigma_{\omega}^2} \right) - \\ &\sum_{i=0}^{n-1} 2 h_{i,j} \big\{ \log S_{t,j}(p_i) \big\} + \sum_{i=0}^{n-1} h_{i,j} \big(I_t(p_i) - A_{t,j}(q_i) \big)^2 / \sigma_I^2. \end{aligned} \tag{2.18}$$

The first term is the logarithm of the motion prior. The second term is the correlation between the layer ownership and the segmentation prior. The third term is the weighted sum of the differences between the current image and the appearance model of layer j under motion $\Theta_{t,j}$. Variances $\sigma_{\vec{\mu}}^2$ and σ_{ω}^2 represent the uncertainty in translation and rotation, variance σ_I^2 represents the uncertainty in intensity between appearance of layer j and current image I_t.

With respect to the previous time instant $t-1$, the first term favours no change in motion. Penalties given by the other two terms are lowest when the correct motion from $t-1$ to t is estimated, even though this means that rotation and/or translation have changed. If a change in translation and/or rotation with respect to the previous time instant $t-1$ is necessary, the

penalty given by the first term must be compensated for by the other two. Since the logarithm is taken of values < 1, the second turn produces a sum < 0. Hence the subtraction of the second term.

To estimate the translation and rotation for time instant t, values from the space of translation and rotation are used. Which values can best be used depends on the application at hand and the frame rate at which the tracking camera produces images. A higher frame rate means a lower per frame movement / rotation.

2.1.8 Shape Estimation

The fourth step in the EM process is to re-estimate the shape prior for each foreground layer. The background layer does not have a shape prior. The prior function (equation 2.5) has constant value β for each pixel belonging to the background. Shape prior Φ_t for all foreground layers j is estimated as

$$\max_{\Phi_t} \arg f = \sum_{j=1}^{g-1} \log \mathrm{N}\left(\Phi_{t,j} | \Phi_{t-1,j}, diag[\sigma_{ls}^2, \sigma_{ls}^2]\right) + \sum_{i=0}^{n-1} \sum_{j=0}^{g-1} h_{i,j} \log S_{t,j}(p_i), \tag{2.19}$$

where the first term is the logarithm of the constancy of shape (equation 2.10) and the second term the correlation between layer ownership and the logarithm of the segmentation prior. The ownership $h_{t,j}$ is calculated in the third EM step, $S_{t,j}(p_i)$ is recalculated with the new shape prior estimates.

The constancy of shape bivariate normal distribution evaluates to

$$\begin{aligned} f(x) &= \frac{1}{(2\pi)^{2/2}\sqrt{|diag|}} \exp\left[-\frac{(\Phi_{t,j} - \Phi_{t-1,j})^T diag^{-T} (\Phi_{t,j} - \Phi_{t-1,j})}{2}\right] \\ &= \frac{1}{2\pi\sqrt{\sigma_{ls}^4}} \exp\left[-\frac{(l_{t,j} - l_{t-1,j})^2/\sigma_{ls}^2 + (s_{t,j} - s_{t-1,j})^2/\sigma_{ls}^2}{2}\right] \\ &= \frac{1}{2\pi\sigma_{ls}^2} \exp\left[-\frac{(l_{t,j} - l_{t-1,j})^2 + (s_{t,j} - s_{t-1,j})^2}{2\sigma_{ls}^2}\right]. \end{aligned} \tag{2.20}$$

Conjugate gradient descent (section 3.4) is used to optimize equation 2.19. The derivatives of equation 2.19 needed for gradient descent are

$$\frac{\partial f}{\partial l_{t,j}} = \sum_{i=0}^{n-1} \frac{h_{i,j}\left(D(p_i) - L_{t,j}(p_i)\right)}{L_{t,j}(p_i) D(p_i)} \frac{\left(L_{t,j}(p_i) - \gamma\right) y_{i,j,x}^2}{l_{i,j}^3} - \frac{l_{t,j} - l_{t-1,j}}{\sigma_{ls}^2} \tag{2.21}$$

and

$$\frac{\partial f}{\partial s_{t,j}} = \sum_{i=0}^{n-1} \frac{h_{i,j}\left(D(p_i) - L_{t,j}(p_i)\right)}{L_{t,j}(p_i) D(p_i)} \frac{\left(L_{t,j}(p_i) - \gamma\right) y_{i,j,y}^2}{s_{i,j}^3} - \frac{s_{t,j} - s_{t-1,j}}{\sigma_{ls}^2} \tag{2.22}$$

where $D(p_i) = \sum_{j=0}^{g-1} L_{t,j}(p_i)$, the sum of all layer priors for pixel p_i, and $[y_{i,j,x}, y_{i,j,y}]^T = R(-\delta)(p_i - \mu_j)$. The lower uncertainty of shape variance σ_{ls}^2, the more the shape is preserved.

2.1.9 Appearance Estimation

The last step is to update the appearance model of each layer with motion Θ_t and shape prior Φ_t fixed. The appearance model of each layer is updated according to

$$\max_{A_{t,j}} \arg \sum_{i=0}^{n-1} \{ \begin{array}{l} \log\left(\mathrm{N}(A_{t,j}(q_i) : A_{t-1,j}(q_i), \sigma_A^2)\right) + \\ h_{i,j} \log P\left(I_t(p_i) | A_{t,j}(q_i)\right) \end{array} \}, \tag{2.23}$$

where the first term is the logarithm of the dynamic layer appearance model (equation 2.23) and the second term is the correlation between the layer ownership and the observation model (equation 2.13). By taking the derivative of equation 2.24 with respect to the appearance model pixel intensity and setting the gradient to 0, $A_{t,j}(q_i)$ can be computed directly as

$$A_{t,j}(q_i) = \frac{A_{t-1,j}(q_i)/\sigma_A^2 + h_{i,j} I_t(p_i)/\sigma_I^2}{(1/\sigma_A^2 + h_{i,j}/\sigma_I^2)}. \tag{2.24}$$

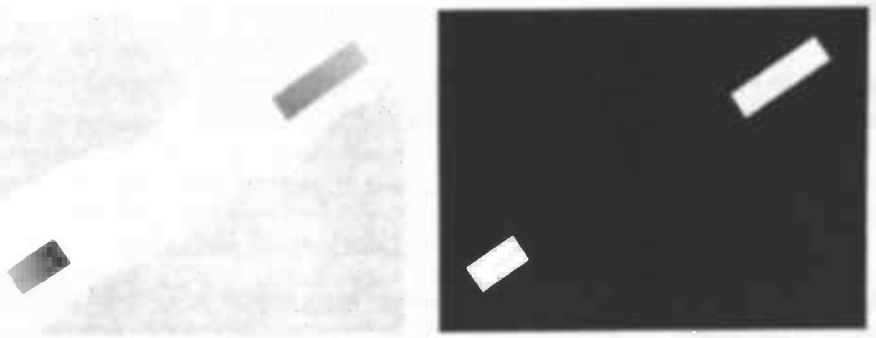
The appearance model for layer j at time instant t is the weighted average between the appearance model for this layer at $t-1$ and current image I_t. The weight is controlled by the ownership, uncertainty in appearance variance σ_A^2 and uncertainty in image intensity variance σ_I^2. The larger the ownership value for pixel p_i in combination with layer j, the more certain it is that pixel p_i really belongs to this layer j. Therefore this p_i contributes more to the update of the appearance model for layer j. The lower σ_A^2, the more the appearance model of the previous time instant is preserved. In case of a high σ_A^2, more weight is carried by the state of p_i in current image I_t. The denominator normalizes the update for each pixel to make sure the new intensity values do not rise out of the intensity range.

2.2 Initialization and Status Determination

The core component of the layered image representation tracking system is called the *layer tracker*. This component consists of the EM algorithm as explained in section 2.1.1. Initialization, addition and removal of foreground layers and object status determination are handled in a separate module. This module is driven by a *state machine*, that handles the tasks mentioned based on change blobs and the current image representation Λ_{t-1}. A *change blob* is a group of connected pixels which indicate an intensity difference between consecutive frames. An example of change blobs found for a certain video frame can be seen in figure 2.5. Details about change blob detection are elaborated in section 3.1.

The state machine knows of five different states in which an object can be: a new object appears, an object disappears, an object moves, an object is stationary or an object is occluded. The states are linked by directed edges which represent the state transitions. The schematics of the state machine can be found in figure 2.6, the states will be elaborated below.

- **New object** - A new object has entered the image frame when a change blob is detected far away from any existing objects. The new object (the new layer) is initialized with a zero velocity, its shape priors and shape angle are estimated using *principle component analysis* (section 3.3) and it's appearance model is build from the current image using the pixel coordinates that belong to the change blob.



(a) Original frame (b) Change blob image

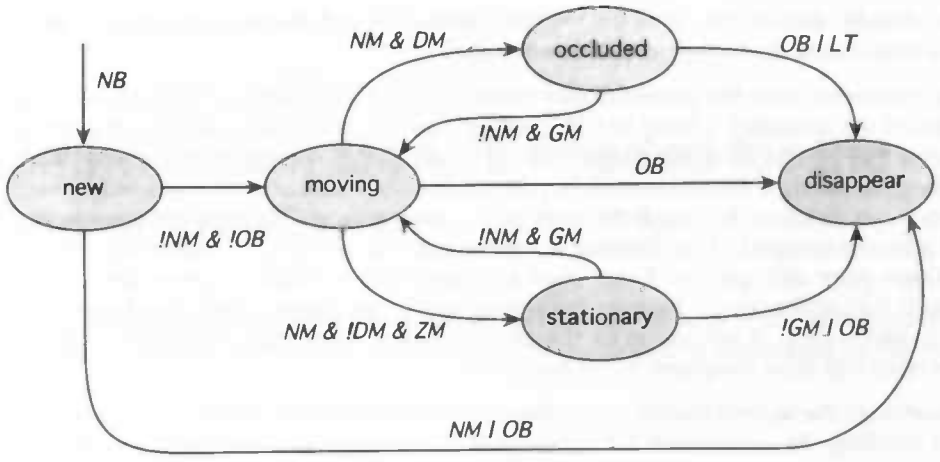
Figure 2.5: (b) is the change blob image of frame (a). Frame (a) is taken from an artificial test set.

- **Moving objects** - During their life span in the layer tracker, objects are in the moving state most of the time. The state of an object is transferred to moving if it is within the image boundaries and has an associated change blob. If an object was stationary or occluded, its state will become moving again if an associated change blob reappears and its appearance model can be matched with the current image.
- **Object disappearance** - An object is deleted from the layered image representation if it moves outside of the image boundaries. If an object is stationary and it's appearance models does not fit on the current image or if an object is occluded and no change blob is detected around it for a long period of time, the object is also deleted.
- **Stationary object** - A moving object becomes stationary if it has no associated change blob, its estimated velocity is zero and its appearance model fits good on the current image.
- **Occluded object** - A moving object becomes occluded if it has no associated motion blob and it does not fit on the current image.

2.3 Tests Conducted by the Author

Although initially developed for a real-time airborne tracking platform, the dynamic layered image representation algorithm is also capable of tracking objects in ground based surveillance systems. The difference between tracking vehicles in a top-down view and people and vehicles in a pan-tilt view could be overcome with the fine-tuning of the variance parameters.

Tao used a system in which the video stream from an airborne platform was sent into a dedicated hardware ground station through a wireless connection. The ground station consisted of a Sarnoff Video Front End processor (VFE, a real-time system for video processing) and a Silicon Graphics Octane workstation on which the layer tracker resided. The VFE handled the ground plane motion estimation. Those estimation parameters along with the original video stream were fed into the workstation, that besides object tracking, also calculated a per frame



& - and
/ - or
! - negation

GM = good appearance match
OB = out of scope
LT = NM for a long time
ZM = zero motion estimation

NB = new blob, no object covering blob
NM = no motion blob covering object
DM = degraded appearance match

Figure 2.6: State machine which handles state transitions of the objects for the dynamic layer tracker. Conditions for the transitions are marked along the edges and explained below the diagram.

low resolution change blob image. The size of the objects in the 320 × 240 video varied from 10 × 10 to 40 × 40 pixels.

The main bottleneck in the computation process is the motion estimation. This estimation accounts for about 95% of the time needed to process of a frame. Although multiple iteration could be used for the EM process, a single iteration proved to be sufficient in practice. The system Tao used was able of handling two moving objects at a speed of ten frames per second and four moving objects at five frames per second.

The layer tracker was compared to a correlation-based tracker and a change-based tracker, both trackers developed by the author. A correlation-based tracker computes the motion of an object by correlating its appearance model with the current images. Once a motion is computed, the appearance model is modified by linearly combining the old model with the information from the current image. An important difference between the layer tracker and the correlation tracker is that the former takes the ownership of individual pixel into account in the correlation and update stages, while the later handles all pixels equally. As a result, the correlation tracker is easily confused by background clutter or other objects that are nearby.

The change-base based tracker completely relies on information gathered from change blobs. When a change blob disappears, this tracker is unable to determine if the associated object has become stationary or occluded. When moving objects pass each other at close range, the possibility that their change blobs merge exists. When the merged blob splits after the passing, the change-based tracker is only able to assign the right change blob to the right object based on their motions. When a merge lasts for a longer period of time, this measure alone can be unre-

liable. Besides motion, the layer tracker has appearance and shape information at its disposal and is thus able to do a better job at this situation.

The experiments with the ground-based video include a non-moving background. Objects to be tracked are between 5×5 and 40×40 pixels in size. The main problem that arises when using the layer tracker in a 3D environment (pan-til camera), is that the objects undergo 3D motions violating the 2D rigid motion model. In practice the layer tracker works reasonably well when objects are at distance. Although the body of a (walking) human being with its arms, legs and a head is hardly compact, if the distance to the camera is large enough it can be regarded as such. The shape prior still applies. Even when a human body is visible in more detail, the tracker is able to follow the person because the largest part of the human body (head and torso) does have a rigid motion. A last reason for the tracker to work reasonably well is that walking people move relatively slow compared to the frame rate.

Parameters of the layered motion representation have to be altered in order to cope with ground-based tracking. To compensate for appearance changes due to 3D rotation, the uncertainty in constant appearance variance σ_A^2 should be increased. This means that the current image will carry a larger weight and thus has a larger influence on the appearance model in the appearance estimation stage. Also due to the 3D nature of the scene, changes in size of the foreground objects can be of a greater magnitude than encountered in areal tracking. The uncertainty of shape variance $\sigma_{l_s}^2$ should be raised accordingly. Since the layer tracker was developed originally for areal tracking, Tao does not go into much detail about the result gained with ground-based tracking.

3 Methods and Models Needed for the Implementation

This chapter will elaborate the methods needed for the implementation of the Bayesian method as presented by Tao *et al.* in [TSK02], but which are not explained or mentioned by the author in his paper.

3.1 Change Blobs

If an image I_b is subtracted from an image I_c, the resulting image I_r will contain pixels with an intensity greater than zero only if the absolute intensity difference between those pixels in I_b and I_c is greater than zero. This is defined as

$$I_r(x,y) = |I_c(x,y) - I_b(x,y)|. \tag{3.1}$$

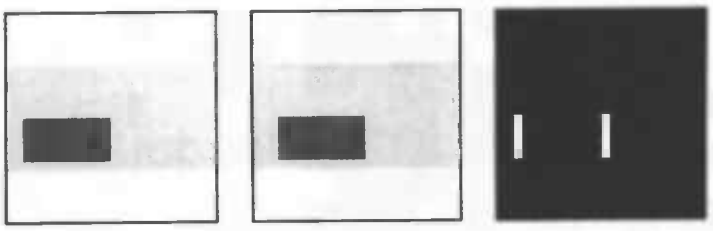
Let I_c be the current image in a video sequence and I_b the background image, the image of the scene with no foreground objects present. Than the subtraction of I_b from I_c will result in an I_r showing where the foreground objects in the current frame are situated. To be able to control the sensitivity with which pixels are selected to be foreground object or not, a minimal absolute intensity difference is used as threshold.

A group of *connected* pixels in I_r is called a *change blob*, indicating the group consists of pixels that changed with respect to I_b. The actual value of the pixels in I_r is not of interest. Because the techniques used to post process the change blob image require a binary image as input, I_r is thresholded so that pixels belonging to a change blob will be white (or 1) and black (or 0) otherwise.

Tao uses consecutive frames to calculate the change blobs. The advantage of doing so is that changes in lighting conditions between consecutive frames are minimal, whereas the difference in lighting over time can be a problem when using a background image to detect change blobs. A disadvantage is that if an object has a simple texture (like most cars do), the change blob will only cover the parts which actually differ. This can result in two separate blobs for one object. This is illustrated in figure 3.1. Although this situation is simplified to a great extent, a similar result is gained with real-world frames.

To overcome the problem of changing lighting conditions, the background image must be updated according to the current frame. If the background image is regarded as appearance model of the background layer, it is possible to update the background in a similar way as the foreground layers by using equation 2.24. Because the ownership value of pixels belonging to foreground objects is low for the background layer, the background will mainly be updated with intensity values of pixels actually belonging to the background.

When working with real-world video, a change blob image is likely to contain noise and background artifacts. Background artifacts can for example be a door or window that is opened or a



(a) t (b) $t+1$ (c) Change blobs

Figure 3.1: Change blob detection using two consecutive frames.

waving tree in the wind. Changes caused by such events are of a small magnitude, delivering small blobs. An example of a change blob image from a real-world video sequence can be found in figure 3.3. Noise removal has to be done in order to only use blobs which actually cover a moving foreground object. In order to be able to delete noise blobs easily, a *connected components labeling* is done first. Both connected components labeling and noise removal will be explained next.

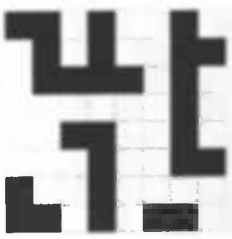
3.1.1 Connected Components Labeling

To be able to perform operations, for example deletion, on change blobs found, it is necessary to know which pixels belong to which blobs. If each blob has an unique identifier, its pixels can be given this identifier accordingly thus making it easy to tell if a pixel in a frame belongs to the blob sought after. Connected components labeling can be used to label each pixel with the identifier of the blob it is part of.

Let B be a binary image and $B[x,y] = B[x',y']$ have a value v where $v = 0$ (white) or $v = 1$ (black). Pixel $[x,y]$ is said te be connected to $[x',y']$ with respect to v if there exists a sequence of pixels $[x,y] = [x_0,y_0],\ldots,[x_n,y_n] = [x',y']$ in which $B[x_i,y_i] = v$ with $i = 0,\ldots,n$ and $[x_i,y_i]$ and $[x_{i-1},y_{i-1}]$ are neighbors for each $i = 0,\ldots,n$. Sequence $[x_0,y_0],\ldots,[x_n,y_n]$ forms a connected path from $[x,y]$ to $[x',y']$. In [SS01] a *connected component* of value v is defined as a set of pixels C, each having v, and each pair of pixels in C are connected with respect to v. Next a connected components labeling of B can be defined as a labeled image LB in which the value of each pixel is the label of its connected component. Each connected component has an unique label. An example of connected components labeling can be found in figure 3.2.

A number of different algorithms for performing a connected components labeling are developed over the years. Among those are recursive algorithms that label one component at a time and iterative methods working on two image lines at a time. The former is useful in situations where the image fits entirely in memory, the latter when this is not the case. The iterative method available from the framework (section 4.1 used, is based on the *row-by-row* algorithm described by Rosenfeld and Pfalz [RP66].

The algorithm makes two passes over the image. The first pass is used to record equivalences and assign temporary labels. The second pass is used to replace each temporary label by the label of its equivalence class. In between the two passes the recorded equivalences, which are binary relations, are processed two determine the equivalence classes. During the first pass, for



1	1	0	1	0	0	2	0
0	1	0	1	0	0	2	2
0	1	1	1	1	0	2	0
0	0	0	0	0	0	2	0
0	0	3	3	0	0	2	0
0	0	0	3	0	0	2	2
4	0	0	3	0	0	0	0
4	4	0	3	0	5	5	0

(a) B (b) LB

Figure 3.2: Connected components labeling LB of binary image B. Five connected components are present.

every pixel p its left and top neighbors are regarded. If both left and top neighbors have the same label, assign this label to p, if only one of them has a label assign this label to p, and if both neighbors have different labels assign one of the labels to p and record the equivalence between the two.

3.1.2 Noise Removal by Opening, Gap Filling by Closing

Morphological operators are used to understand the structure of an image. Morphological operators are usually applied on binary images, but can be extended to grey-scale. Since the connected components image is binary of nature, operations on grey-scale image will not be elaborated.

Binary operators have a binary image B as input, as well as a *structuring element* S. This structuring element is a binary image itself, representing a shape much smaller than B. S serves as a kernel that moves over B, with one pixel designated as the origin. While the kernel moves over the image, it is checked if the shape of S fits in the region under inspection of B. Accordingly an action can be triggered, for example the region can be enlarged with the shape.

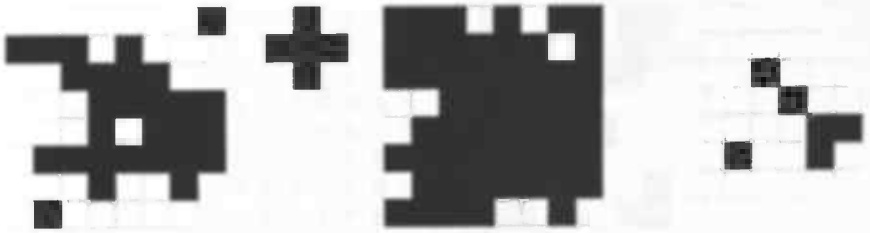
Two important primary morphological operators are *dilation* and *erosion*. The former enlarges a region, the latter makes it smaller. The definitions for both operators are found in [SS01]. The dilation of binary image B by structuring element S is denoted as $B \oplus S$ and defined by

$$B \oplus S = \bigcup_{b \in B} S_b. \tag{3.2}$$

Structuring element S is swept over B and every time the origin of S encounters a binary 1-pixel (or black pixel) the translated shape of S is OR'ed to the output image. The output image is initialized with 0-pixels (or white pixels). While sweeping, it can happen that pixels of S fall outside of image B. Those are ignored. The erosion of image B by structuring element S is denoted as $B \ominus S$ and defined as

$$B \ominus S = \{b | b + s \in B \, \forall s \in S\}. \tag{3.3}$$

S is swept over B as seen with the dilation, but this time at each position where every 1-pixel of S covers a 1-pixel in B the origin pixel of S is OR'ed to the output image. See figure 3.3 for an illustration of both the dilation and erosion operations.



(a) B (b) S (c) Dilation $B \oplus S$ (d) Erosion $B \ominus S$

Figure 3.3: Dilation (c) and erosion (d) of binary image B (a) with structuring element S (b).

The application of an erosion directly followed by a dilation with the same structuring element is called an *opening*. The opening of a binary image B with structuring element S is denoted by $B \circ S$ and defined by

$$B \circ S = (B \ominus S) \oplus S \tag{3.4}$$

The operation thanks its name to the fact that it tends to open small gaps or spaces between objects that are connected trough a thin line. After an opening objects in an image are better isolated, making it easier to detect them. Besides the isolating property, an opening also is able to remove noise from an image. The erosion of an image in which clusters of pixel are present along with isolated pictures in random locations, will remove the isolated pixels and the boundaries of the clusters. The dilation of the eroded image will restore (most of the) boundaries of the clusters.

Closing means that an image is dilated, immediately followed by a erosion with the same structuring element. The closing of binary image B with structuring element S is denoted as $B \bullet S$ and defined as

$$B \bullet S = (B \oplus S) \ominus S \tag{3.5}$$

The closing of an image will do the opposite of an opening: small gaps are filled. This is useful in case of change blobs detection because holes in blobs can be closed this way.

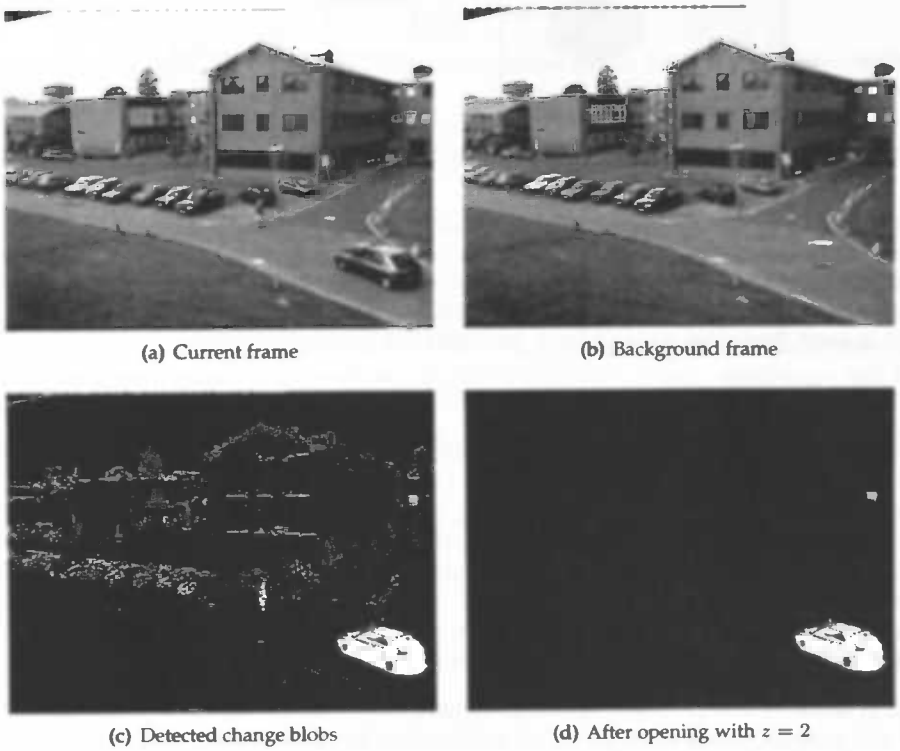
It is possible to do z erosions, followed by z dilations with the same structuring element. This will create an opening with a *depth* of z. An opening with a 4-connected structuring element and a depth of $z = 2$ will remove all objects smaller than or equal to $4 \times y$ or $x \times 4$ pixels in size.

A side effect of the opening operator is that it is not guaranteed that objects dilate back to their original shape after the erosion steps. To overcome this problem the *conditional dilation* is defined. The conditional dilation operator makes sure that a cluster of pixels cannot grow back beyond its original set of pixels. Given an original binary image B, a processed image C and a structuring element S, let $C_0 = C$ and $C_n = (C_{n-1} \oplus S) \cap B$, the conditional dilation of C by S with respect to B is defined as

$$C \oplus |_B S = C_m, \tag{3.6}$$

where in index m is the smallest index that satisfies $C_m = C_{m-1}$. Eq. 3.6 says that set C_0 is repeatedly dilated by S, while after each step being reduced to the subset of pixels that where present in B. The noise and background clutter filtering used in the Layered Image Representation algorithm makes use of opening with the conditional dilation instead of the dilation defined in equation 3.2.

Using an opening to remove noise and background clutter from an image can be tricky in situations where the objects to be detected are very small. For example, a person can be represented by 2×10 pixels in real-world video frames, being just as big as artifacts due to changes in lighting conditions. In those cases, noise and artifacts removal with opening is not possible. An example of a such a difficult situation can be found in figure 3.4. Here the person is removed, while a window in the building is preserved. Applying the opening operator with $z = 3$ would remove the window and only keep the car.



(a) Current frame

(b) Background frame

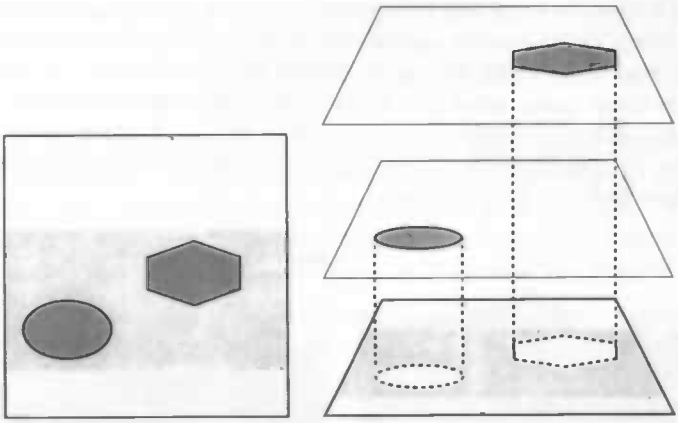
(c) Detected change blobs

(d) After opening with $z = 2$

Figure 3.4: (b) shows the change blobs detected by subtracting the background frame (b) from current frame (a). (d) is the result of noise removal with the opening operator.

3.2 Appearance Model

A Layered Image Representation of frame I_t from a video sequence, segments the objects present in the image from the background. An object is an entity which is part of a scene for a certain amount of time, thus not being part of the background. An object moves around in the scene most of the time while present. Background and objects are each put in their own layer, with the background being layer 0. A segmentation example is given in figure 3.5. A layer has the same dimensions as I_t.



(a) Frame (b) Segmentation

Figure 3.5: Frame (a) is part of a video in which an oval and a polygon move in opposite directions over a road. (b) is the segmentation in different layers of this frame. In this case, the white inside a layer indicates that those pixels are not part of the layer.

Every layer j in the Layered Image Representation of a frame at time instant t has an appearance model $A_{t,j}$; a model which reflects the current visual state of its layer. The appearance model is defined in its own local coordinate system and used during the construction of the observation model (equation 2.12), motion estimation (equation 2.18), and its own updating process (equation 2.24) . To be able to use the appearance model, it has to be *warped* from its own coordinate system to that of current frame I_t (figure 2.3). This is accomplished by translating the model according to estimate μ, the center of the object, and rotating it according to estimate δ, the angle the object makes with the coordinate system of I_t.

Since it is unlikely that an object takes up all pixels in a frame, the layer which represents the object will contain pixels that cannot be assigned an intensity value. Those pixels can be regarded as transparent and they belong either to the background or to another object. To be able to recognize them as such, pixels in a layer that do not belonging to it are set to -1.

When a new object enters the scene a new layer is initialized, as well as its appearance model. Pixels under the change blob that triggered the initialization (section 2.2) are taken from the current frame and put in the center of the new appearance model coordinate system. The appearance model center coincides with the center of the frame.

Rotation Tao defined the appearance model in such a way that the major axis of the segmentation prior distribution is situated on its x-axis. This means the appearance model has to be rotated with δ, the angle the major axis makes with the coordinate system of I_t, to be able to map it on the current image. In case of an appearance model update, the model is first warped onto I, updated and than warped back, involving two rotations. Rotating with angles other than multiples of 90° results in a loss of quality and should therefore be minimized. Since the appearance model is needed in its rotated form in all cases during Expectation Maximization,

it is better to keep it that way instead of aligning its major and minor axis with the coordinate system of I_t.

Rotation is not completely unavoidable of course. Objects can turn and their appearance models have to turn with them accordingly. During motion estimation the appearance model is rotated with various angles ω to see if there was a rotation between time instant $t-1$ and t. To rotate the appearance model a basic point rotation is used, defined by

$$R(\omega) = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix}. \tag{3.7}$$

Due to rounding of coordinates, multiple pixels are sometimes rotated to the same destination pixel, resulting in holes in the rotated image. Those holes are eliminated by sweeping a 4-connected kernel over the rotated image. In case all four neighbors of a pixels are filled (do not have value -1) and the center pixel of the kernel is not, the average intensity of the neighbors is written to the kernel center pixel.

A rotation scheme which makes use of anti-aliasing was implemented first, unfortunately not giving the desired results. Anti-aliasing smooths the rotated result image by finding the four source pixels a destination pixel partially covers, assigning the area-weighted average intensity of those four source pixels to the destination pixel (figure 3.6). Although this scheme produced a smoothed rotated appearance model without holes, the intensities of the rotated pixels differed too much from their originals. While estimating the motion parameters, matching a non rotated appearance model on I gave a lower error rate than matching the correctly rotated one, due to this intensity difference. The basic rotation scheme of equation 3.7 in combination with the hole filling, yielded better results.

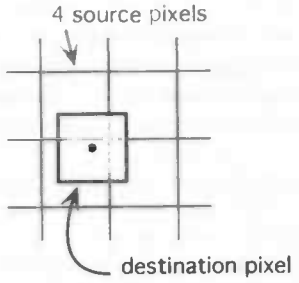


Figure 3.6: A pixel is rotated to its exact destination coordinates. The intensity of the destination pixel is based on the overlap area it has with the the four source pixels it falls on.

Transparency To construct the observation model, any pixel p_i from I_t is compared to its equivalent q_i in $A_{t,j}$. Every pixel in I_t has an intensity, but that does not hold for $A_{t,j}$. To overcome this problem, the background intensity value is used for a pixel when it is transparent in layer j. This results in an observation model for layer j with high probability values for pixels belonging either to the background or the layer itself. Pixels belonging to other layers have a low probability. The fact that also background pixels have a good score in the observation model is not a problem when calculating the layer ownership (equation 2.16). The normalized segmentation prior for a layer is low for pixels outside the distribution forming the shape of an

object and those are exactly the pixels that belong to the background in the observation model. High scores for background values are thus neglected when calculating the ownership for a layer. An example of a normalized prior distribution, observation mask and ownership mask for a certain video frame can be found in figure 3.7.

In the motion estimation step a same sort of situation is encountered, since the third term of equation 2.18 requires that $A_{t,j}$ is subtracted from I_t. By taking the sum of the intensity difference times the ownership value for each pixel, a matching score is obtained. The lower the score, the better the match. By involving the ownership in the matching score intensity differences for pixels belonging to the object or very close to the object carry a larger weight.

During motion estimation a range of possible translations and rotations is tried to find the best matching motion parameters from $t-1$ to t. The appearance model is warped onto I_t according to the parameters that are tested each step. To be able to do the subtraction, the background intensity value is used in case a $A_{t,j}$ pixel has value -1. Note that if the background would be used in the appearance model instead of transparent pixels, the background would also be warped each motion estimation step. Although intensity differences for pixels belonging to the object are most important, the matching score would be negatively influenced by the warped background.

Over time an object can shrink or grow, can have a change in shape or can change of appearance. During the appearance estimation step the appearance model of each layer is updated with new information. Eq. 2.24 makes use of all pixels in frame I_t to accomplish the update. In this case, transparent pixels cannot be substituted by their background equivalents since the background does not contain information on the object to be updated. Current frame I_t contains the object in its current shape and appearance and is therefor the source which has to be used in case a -1 pixel is encountered in $A_{t,j}$. It is desired to only update pixels that belong to the object and leave or make the others transparent. Because equation 2.24 works on all pixels it results in an appearance model for which all pixel contain an intensity value, although faint due to the use of the ownership mask. It is this ownership mask that can be used to indicate which pixels belong to the layer that is updated, by deriving the actual segmentation from it.

The actual layer segmentation can be derived from the ownership mask by assigning the ownership value to a layer pixel in case this pixel has the highest ownership value and zero otherwise. A layer can be updated with this mask by applying equation 2.24 to all pixels for which the ownership is greater than zero. If the ownership is greater than zero but the pixel in $A_{t,j}$ is transparent, than it is substituted with the intensity of the equivalent pixel from I_t. Pixels with an ownership of zero are set to transparent. This way an appearance model grows and slinks according to the shape prior that works on the ownership mask.

3.3 Initialization of Shape Priors and Distribution Angle

When at time instant t a new foreground object j is detected by the layer tracker, a shape prior $\Phi_{t,j} = [l_{t,j}, s_{t,j}]$ has to be initialized, as well as the angle δ the segmentation prior distribution makes with the axis of the coordinate system of current image I_t (figure 2.1). To this end, *principal component analysis* (PCA) is used. PCA provides a covariance matrix which contains the information needed to initialize all three parameters. This information is obtained by calculating the eigenvalues and eigenvectors with the *QL algorithm*.

Principal Component Analysis Principal component analysis is used to simplify a data set: it is a linear transformation that chooses a new coordinate system in such a way that the greatest variance of the data set comes to lie on the first axis, the second greatest variance on the second axis and so on. The first axis is called the first *principal component*, the second axis the second principal component, etc.

In the case of initializing the shape prior, the data set is a two dimensional image I_{am} and hence two principal components are used. The analysis is performed on the appearance model after its initialization as described in section 3.2, taking into account the pixel for which $I_{am}[x, y] \neq -1$. The result of PCA in the two dimensional case is a covariance matrix

$$C = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) \\ \text{cov}(y,x) & \text{cov}(y,y) \end{bmatrix} \tag{3.8}$$

where the covariance $\text{cov}(a, b)$ indicates how much the dimensions a and b vary from the mean with respect to each other. The covariance for a dimension against itself is the variance (spread of the distribution) for that dimension. PCA needs a zero empirical mean, thus the center of the distribution needs to be subtracted from its pixel coordinates. Covariance $\text{cov}(a, b)$ is defined as

$$\text{cov}(a,b) = \frac{\sum_{i=0}^{n-1}(a_i - \bar{a})(b_i - \bar{b})}{n-1} \tag{3.9}$$

where n is the number of sample points in the data set. In this case, the mean needs not to be calculated, because it is known the appearance model is situated at the frame center coordinates $[x_{amc}, y_{amc}]$.

In case a data set consists of pure data points, the above will deliver the desired covariance matrix. In case the data points are pixels, the nature of those little squares has to be taken into account. A pixel is not just a point, it is a tiny area. Consider for example an object (data set) consisting of two horizontally adjacent pixels, as displayed in figure 3.8. When subtracting the x-coordinate of object center c from the x-coordinate of pixel p, the resulting distance will be 0.5, although the actual x-distance from c to the farthest border of p is 1. Since c and the centers of pixels p and q are laying on one line, the y-distance is zero. This also is incorrect, since the height of the object is 1. A distance correction is thus necessary to get the correct covariance matrix.

Since the distances in the x-direction and y-direction are handled separately (it is not the euclidean distance between an object center and a pixel that is calculated!), a rather simple correction can be used. When the difference $a_i - \bar{a}$ in any of the directions is positive, a correction of $+0.5$ is added. If the difference is negative, the added correction is -0.5. In case the difference is zero, meaning the center of the object and the pixel lay on a line parallel to one of the axis (depending if it is the x or y distance that is calculated), a correctional value of $+1$ is added. An example of the the distance correction can be seen in figure 3.9.

The covariance matrix holds important information about the object. From the eigenvalues of this matrix the l and s can be derived. The eigenvalues are the variances of both principal components. Since l and s are used as standard deviations, the square root of the eigenvalues has to be taken. The square root of the largest eigenvalue gives l whereas the square root of the smallest gives s. The two perpendicular eigenvectors give the angles of l and s with respect to the normal coordinate system.

Due to the matrix inverse (equation 2.7) that is used for calculation of the segmentation prior, it is of importance that both l and s ar greater than zero. Since objects in a video frame always

have a width and height, it is highly unlikely l or s will be zero. If the width:height ratio of an object approaches zero problems due to rounding can occur. Luckily, most objects in nature have a ratio of 1:Phi.

Both eigenvalues and eigenvectors are calculated with the QL algorithm, as explained in section 11.3 of [PTVF02].

3.4 Conjugate Gradient Descent

Gradient descent is used to optimize the shape estimation function (equation 2.19), by searching for maximum argument Θ_t that satisfies it. Gradient descent is often explained under the assumption that it is a minimum that is sought after, that manner will be followed here as well.

Let $f(x)$ be the function for which the minimum is sought and x the unknown variables. The optimization procedure starts by choosing a starting point x_0; the initial guess for the unknown parameters of $f(x)$. Once x_0 is chosen, two decisions need to be made before the next point can be evaluated. First, the direction along which the next point is to be chosen has to be picked. Second, the step size in the chosen direction has te be set. That gives the following iteration

$$x_{k+1} = x_k + \lambda_k \vec{d}_k, \tag{3.10}$$

where $k = 0, 1, 2, \ldots$, $\vec{d}_i$ is the direction, and $|\lambda_i \vec{d}_i|$ is the step size. Different classes of techniques exists, presenting different choices for λ_i and $\vec{d}_i$.

The method of *steepest descent* (also called *gradient descent* method) finds the nearest minimum of a function $f(x)$, by using the gradient of that function. It starts at x_0 and moves as many times as needed from x_k to x_{k+1} by minimizing along the line extending from x_k in the direction of the local downhill gradient: $-\nabla f(x_k)$. This is the direction in which f decreases most quickly. Taking the next step in the direction of the negative gradient at each point, results in a zig-zag pattern search, as illustrated in figure 3.10.

The method of steepest descent has a fast iteration cycle and is guaranteed to find the minimum if there exists one, but generally has slow convergence (see section 10.6 of [PTVF02] for details). The *conjugate gradient* method often is a better choice. The method of steepest descent converges slowly because it has to take a right angle turn after every search step, continuing its search in the same direction as previous steps. The conjugate gradient method uses conjugate directions for going downhill, letting each search direction $\vec{d}_i$ be dependant on all other directions already searched to locate the minimum. If f is n-dimensional, the conjugate gradient methods needs n line searches (n^2 in worst-case) to reach the minimum.

The conjugate gradient descent method as explained in section 10.6 of [PTVF02] is used to optimize the shape estimation function.

3.5 Image Scaling with a Bartlett Filter

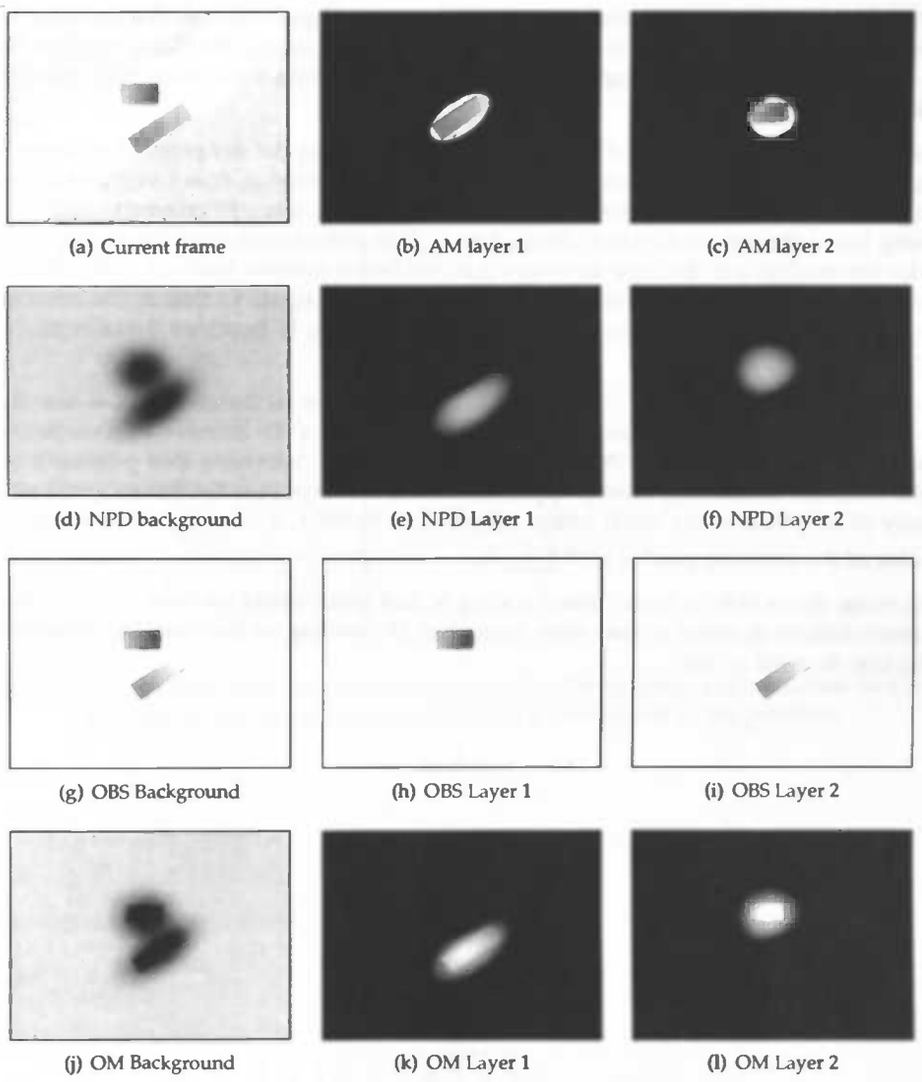
The speed of the Bayesian method is related to the dimensions of a video frame (section 4.2.3). Scaling down the frames before they are used for change blobs detection and object tracking, results in a speed up.

The down scaling of an image causes groups of pixels to collapse into one. A smoothing filter is necessary to minimize aliasing problems in the down scaled image. For this purpose a Bartlett filter can be used. After the image is smoothed, a convolution mask is used to select which pixels are kept.

Given a scale factor f, a Bartlett filter is constructed by taking the dot product of a row vector and a column vector. Both contain $2 \times f - 1$ elements, counting up from 1 to f and then down to 1 again. The sum of the elements in the Bartlett filter is always f^2 greater than the desired intensity, hence the dot product is multiplied by $1/f^2$ in order to maintain the average intensity level for the result pixels. Because the image is scaled down, another factor of $1/f^2$ is introduced in order to make the average intensity of the result image equal to that of the original. An example Bartlett filter used to smooth an image that is going to be scaled down with $f = 2$ is illustrated in figure 3.11.

What the Bartlett filter actually does is weighing pixels closer to the center more heavily than those further away. It can be seen as placing a cone on top of the kernel that sweeps over the image, and using the height of the cone at each location to determine that position's relative weight. After the smoothing filter is applied, a kernel is swept over the image to calculate the intensity of the pixels in the result image. Given scale factor f, a $f \times f$ kernel k is used, where the value of the resulting pixel is $1/f^2 \sum_{i=0}^{f^2} k_i$.

The obvious down side to frame down scaling is that small object become even smaller (and thus more difficult to track) or may even disappear. Depending on the situation at hand, image scaling may be used or not.



(a) Current frame (b) AM layer 1 (c) AM layer 2

(d) NPD background (e) NPD Layer 1 (f) NPD Layer 2

(g) OBS Background (h) OBS Layer 1 (i) OBS Layer 2

(j) OM Background (k) OM Layer 1 (l) OM Layer 2

Figure 3.7: The layered image representation of frame (a) has appearance models (b) and (c). The appearance model of the background (not displayed) consists of the green slabs with the sandy road. The normalized prior distribution for the three layers is shown in (d), (e) and (f). Priors are mapped to grey levels, from black being the lowest prior to white being the highest. The observation model for the layers is seen in (g), (h) and (i). Values are mapped to grey levels, likewise as the normalized prior distribution. The observation model values for a layer are high for the object the layer represents and the background. The ownership mask can be found in (j), (k) and (l), with the same grey level mapping as the normalized prior distribution. Since background pixels for a foreground layer have a low prior, their high observation model value is of little influence to the ownership mask.

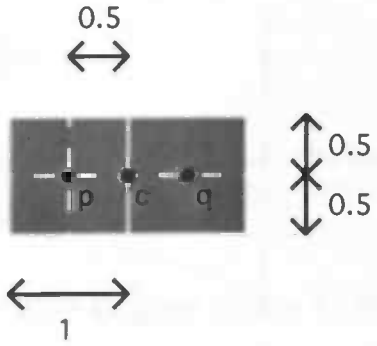


Figure 3.8: A two pixel object with object center c. The distance from c to p in the horizontal direction is 1 in stead of 0.5.

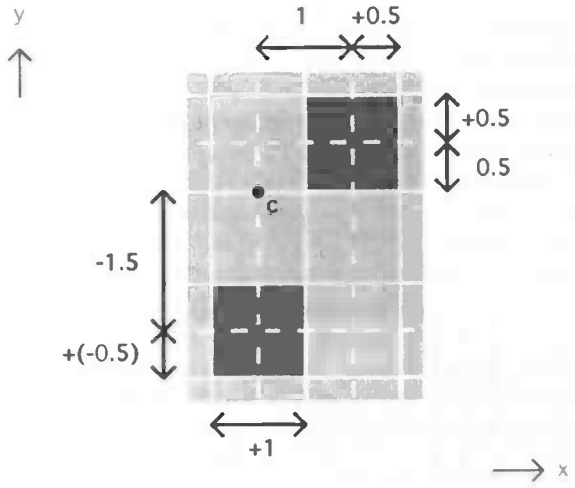


Figure 3.9: When measuring the x or y-distance from object center c to a pixel, a correction is in order. A correctional value in the different cases is indicated by a + sign.

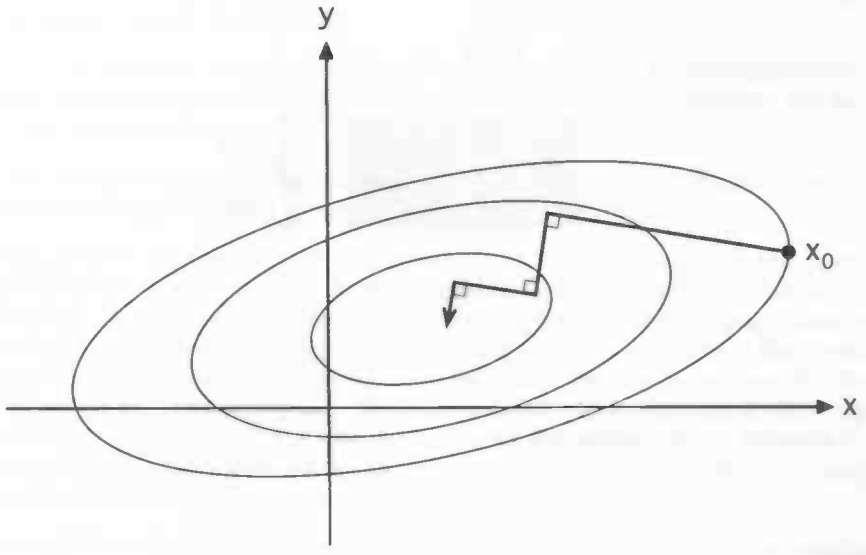


Figure 3.10: The method of steepest descent approaches the minimum of a function in a zig-zag manner, where the new search direction is orthogonal to the previous.

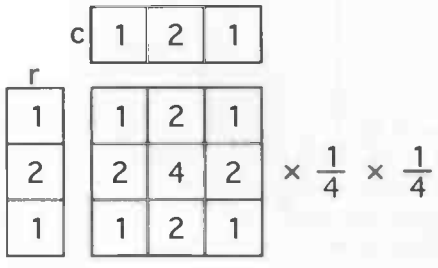


Figure 3.11: A Bartlett filter used to smooth an image that is going to be scaled down with a factor of 2.

4 Implementation

This chapter presents issues regarding the implementation of the Bayesian object tracking method developed by Tao *et al.* [TSK02]. The algorithm is implemented in a framework developed at the MIVIA research group at the University of Salerno.

4.1 MIVIA Framework and Global Code Outline

At MIVIA a framework is developed for implementing and testing various object tracking methods. The framework is built using a combination of Java and C++, communicating via the Java Native Interface. The Java part is responsible for the *reading* of JPEG frames or MPEG video sequences, while the C++ part handles the *processing* of the frames and thus the actual tracking. The framework was initially developed in cooperation with a company that already had an image processing library available in Java, explaining the reason to involve the somewhat slow Java programming language.

A frame of a video sequence is represented in a `Frame` structure, featuring information like frame width and height, and parameters regarding the organization of the pixel data. The pixel data is kept in an array, per pixel a red, green and blue band value are present.

The implementation of the Bayesian method was build from ground up, using only the video sequence input reading, frame representation, ground truth system, and connected components detection provided by the framework. Its main data structure is the vector containing the background and foreground objects, defined in the `Object` class. This class represents a layer in the layered image representation Λ (section 2.1.1), with its motion, shape and appearance parameters. Also present is information about layer with and height, state, and the bounding box around the object represented. The background layer forms a special part of the vector. This layer is always present at vector position 0, has no motion parameters, has a constant shape prior β, and its appearance model is made up from the scene without any object present.

Files part of the implementation are briefly explained below.

- `shape_blob` Contains the `Blob` class which represent a change blob. Information about the blob center, its size, and its state is available. A blob can have two states; it either is located on or within the frame border. Each blob has an unique id, assigned to it during the determination of all change blobs.
- `shape_conjugateGradient` Contains the conjugate gradient method found and explained in Numerical Recipes [PTVF02] section 10.6. The conjugate gradient method is used within the Expectation Maximization step to estimate the shape prior.
- `shape_display` Contains functions to display on screen the current frame, appearance models, background frame, normalized prior distributions, observation models, and ownership masks. This file makes use of the open source CImg Library (http://cimg.sourceforge.net/) developed by David Tschumperle.

- `shape_eigen` Contains the QL algorithm found and explained in Numerical Recipes [PTVF02], section 11.3. This algorithm is used to compute the eigenvalues and eigenvectors from the covariance matrix derived from the appearance model during principal components analysis (section 3.3).
- `shape_em` Contains functions needed for the Expectation Maximization step, among which functions to estimate motion, shape, and appearance parameters. EM parameters are also set in this file.
- `shape_morph` Contains functions to perform an opening or closing (section 3.1.2) on a frame containing connected components labeled change blobs.
- `shape_object` Contains the `Object` class which represents a layer in the layered image representation. The class holds information about the estimated object center in the current frame, the state of the object, its estimated motion and shape, its appearance model, the angle δ the prior distribution makes with the coordinate system of the current frame (figure 2.1), the bounding box that encloses the object, the appearance match of the object, and the change in appearance match with respect to the previous frame. The appearance match indicates how well the appearance model fits on the current frame and is determined during motion estimation. The appearance match and its delta are used by the state machine to determine if an object is occluded.
- `shape_print` Contains functions to print frames, normalized prior distributions, observation masks, and ownership masks to standard output.
- `shape_statemachine` Contains the state machine (figure 2.6) that determines the state of each object. Here new objects are added to or delete from the objects vector.
- `track_shape` Contains the layer tracker, which is invoked by the Java part and handles the tracking of the objects. The objects vector is a global parameter that exists over time, during the processing of a video sequence. Opening, closing, and frame scale factors are set in this file, as well as the change blobs threshold.
- `tracking_MotionTracking_shape` Contains the interface between JAVA and C++. The video sequence that is used as input is set in this file.

4.2 Layer Tracker

The layer tracker is the central part of the implementation. Each time the Java frame reader invokes the tracker, change blobs are determined, the objects vector is updated, and Expectation Maximization is performed to estimate the motion, shape and appearance parameters for the current frame. The schematics of the layer tracker can be found in figure 4.1.

The layer tracker is also used for the initialization of the objects vector with the background layer. The first frame that is passed on to the tracker is assumed to be the background frame and is used to initialize the background layer's appearance model. Other parameters a layer normally has, are not of any interest regarding the background layer and are not used in the tracking process.

A ground truth system (chapter 5) is used to quantitatively measure the performance of the method. The ground truth is a per frame list of bounding boxes, one for each object present.

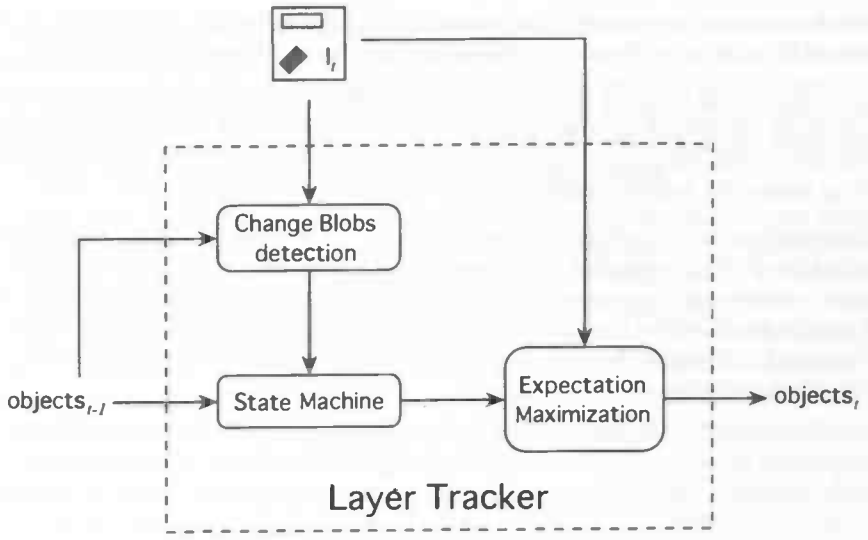


Figure 4.1: Layer tracker schematics.

This list is defined by a human being and assumed to be error free. The layer tracker outputs the bounding boxes of each object it currently tracks. This output list is compared to the ground truth list.

The layer tracker provides the option to scale frames (section 3.5) before processing them, resulting in a speed up. When scaling is used, the option to use ground truth initialized change blobs disappears. Scaling ground truth boxes was not implemented because performance testing was to be done on full scale frames. The obvious problem with using scaled frames is that small objects get as small as noise or disappear completely, and that an amount of detail is lost. Both make object tracking more difficult. Motion estimation translation parameters should be adjusted according to the amount of scaling used, since scaling influences the speed in pixels per frame of foreground objects.

Each step of the layer tracker will be briefly elaborated below.

4.2.1 Change Blobs Detection

Given the current frame I_t and the background frame, present in the objects vector as the appearance model of the background layer, the change blobs (section 3.1) for time instant t can be derived. If the difference in color between I_t and the background frame is larger than a certain threshold, a white pixel is written to the change blobs frame, otherwise a black one. The use of a background frame for change blobs detection differs with the approach taken by Tao, who uses consecutive frames.

Change blobs can be detected with or without using ground truth. When ground truth is used, changes between I_t and the background frame are only sought after within the boundaries of each ground truth bounding box, that is, objects are sought in the areas of the current frame

where objects are actually present. This option is provided to be able to test only the tracking capabilities of the method without the problems arising in the object detection phase.

Connected components labeling is applied to the detected change blobs to be able to identify every blob easily. To get rid of noise and small image artifacts, an opening filter is applied to the labeled blob frame. Next, to fill up gaps in the change blobs, a closing filter is applied. With the resulting frame, the actual change blobs for time instant t are initialized.

During change blobs initialization, the size and center of the blobs is calculated and the label given to the blob during connected components labeling is used as id. Depending on the blob being on the frame border or completely in the frame, the state of the blob is set. With this state, an object can be re-initialized by the state machine (see next section for details). Change blobs are saved into a `Blob` vector. Besides this vector, the filtered connected components are passed along to the state machine.

The appearance model of the background layer is updated along with the appearance models of the foreground layers, during Expectation Maximization. If the ownership mask indicates that a pixels belongs to the background, the background appearance model is updated at this location with information from I_t. This way the background frame is dynamically updated to cope with for example gradual changes in lighting conditions.

4.2.2 State Machine

The state machine updates the objects vector with information gained from the change blobs. To this end, two types of associations are established. The first type is $blob \rightarrow object$, the association of a blob to the object closest to it, thereby regarding a minimal distance threshold. This thresholds assures that a blob is not associated if it is far away from all currently existing objects. A not associated blob indicates a new object. An object can be associated to multiple blobs.

The second type is $object \rightarrow blob$, the association of an object to the blob closest to it, with the same distance threshold used for the other type. A blob can be associated to multiple objects. It is not always true that $blob \leftrightarrow object$, the object closest to a blob does not necessarily have to have this blob as closest blob in return. In case of an occlusion two blobs that were previously separate, are merged to one. The two objects under this blob will have it associated, while the blob is only associated to one of the objects.

After establishing $blob \rightarrow object$ for all blobs present and $object \rightarrow blob$ for all objects present, the associations are processed. The $blob \rightarrow object$ associations are only used to initialize new objects, that is when an unassociated blob is encountered. The appearance model of a new object is initialized with pixel values from I_t found under the blob, its state is new, its center is equal to that of the blob, motion is zero, and the initial shape prior and angle δ are derived using principal component analysis (section 3.3).

The processing of the $object \rightarrow blob$ associations is done using the state machine as seen in figure 2.6. Here object states are updated and objects are removed if they are not present anymore (out of scope). The fact that a background is used to determine change blobs means that stationary objects also produce a blob. This requires changes in state transition rules since the NB (No motion Blob covering the object) condition cannot be used to determine if an object has become stationary. Instead, an object is marked as stationary if it has a zero motion and no degraded appearance match.

Tao also uses the absence of a change blob, in combination with a degraded appearance match, to determine if an object is occluded. When objects A and B are involved in an occlusion, the motion blob covering both objects is assigned to both A and B, because the blob center is closest to both object centres. Even if the blob would be assigned to only one of the objects, it could not be concluded that the object without blob is occluded by the one with blob. Thus marking an object as occluded is based solely on its degraded appearance match.

When the associated blob of an object touches the frame border, the appearance model of the object is reinitialized and principal components analysis is run again. Translation parameters are derived from the current and previous object centres. Reinitialization is done to be able to cope with objects entering or leaving the scene. While not completely in the frame, those objects have a radically changing shape and appearance, making correct estimation difficult. An incorrect shape prior and appearance model result in faulty motion estimation, causing faulty tracking. If an object A is occluded, it should not be reinitialized because the object in front of A will the be wrongly incorporated in A's appearance model. It is therefor assumed to occlusions do not take place on frame borders.

Changes and additions with respect to Tao's state machine are summarized below.

- **New** - Object reinitialization when associated blob touches frame border.
- **Moving** - Object reinitialization when associated blob touches frame border. The transition to stationary is based on an appearance match that is not degraded and a zero motion. The transition to occluded is solely based on a degraded appearance match.
- **Stationary** - Object reinitialization when associated blob touches frame border.

The state machine outputs the updated objects vector to the expectation maximization step.

4.2.3 Expectation Maximization

Expectation maximization (EM) is used to estimate the model parameters for all layers, based on the previous parameters found in the objects vector, and current frame I_t. Besides implementation remarks, a time complexity analysis will be done for the distinctive EM steps (figure 2.4). D stands for the $x \times y$ frame dimension, L for the number of layers and B for the number of bands used.

Tao uses one parameter σ_I, that is used during the construction of the observation model, motion estimation, and appearance estimation. In practice it seemed necessary to able to set the value of σ_I separately for all three parts, and therefore it is split into a $\sigma_{I,obs}$ for the observation model, a $\sigma_{I,mot}$ for the motion estimation, and a $\sigma_{I,app}$ for the appearance model.

Ownership The ownership (OWN) is calculated from the normalized prior distribution (NPD) and the observation model (OBS), where

$$NPD = D \times L, \tag{4.1}$$

$$OBS = L \times 2 \times D \times B, \tag{4.2}$$

$$\begin{aligned} OWN &= NPD + OBS + D \times L \\ &= 2 \times D \times L \times (B+1) \\ &= 8 \times D \times L \\ &= D \times L. \end{aligned} \tag{4.3}$$

OBS involves an initialization step and warp/value calculation step for all layers L. In most cases $B = 3$, making the OWN time complexity dependent on frame dimension and number of layers.

As discussed in section 3.2, the calculation of the observation model requires an appearance model with values for all pixels, while the appearance model due to its nature contains transparent pixels. This problem is solved by using the corresponding background value whenever a transparent pixels is encountered during calculation.

Motion Estimation To estimate the motion of a foreground object from time instant $t-1$ to t, the appearance model of the object is warped ($WARP$) several times, using different translation $\vec{\mu}$ and rotation ω combinations. Each time, the warped model is compared to I_t and an appearance match value is calculated. The warp yielding the best match value is chosen as estimate for the object. Warping consists of initialization, applying the translation and rotation to the appearance model, and the filling of holes. This result in a time complexity of

$$\begin{aligned} WARP &= 3 \times D \times B \\ &= D. \end{aligned} \tag{4.4}$$

Let MUX be the possible translation in the x-direction, MUY the possible translation in the y-direction, and $OMEGA$ the possible rotations, then the time complexity of the motion estimation (MOT) step is

$$\begin{aligned} MOT &= OMEGA \times MUY \times MUX \times (L + NPD + L \times (WARP + D \times B)) \\ &= OMEGA \times MUY \times MUX \times (L + D \times L + L \times (4 \times D \times B)) \\ &= OMEGA \times MUY \times MUX \times L \times (1 + D + 4 \times D \times B) \\ &= OMEGA \times MUY \times MUX \times L \times (1 + 13 \times D) \\ &= OMEGA \times MUY \times MUX \times L \times D. \end{aligned} \tag{4.5}$$

An optimization is gained by using the bounding box of an object as region in which the translations and rotations are tried. To make sure the appearance model fits in this region after translation and rotation are applied, the bounding box is enlarged by a number of pixels in every direction. Using a region instead of the whole frame hardly influences the match score, since the ownership value of pixels further away from the object is very low.

The tested translation and rotation parameters are fixed. A typical set consists of five x, y and ω values (two positive, two negative and the neutral value), resulting in 125 motion estimation steps for each layer. Each of the three parts of equation 2.18 is normalized. The logarithm of the motion prior is situated in the range of $[0, 2]$ (both translation and rotation are situated in the $[0, 1]$ range), the correlation between the layer ownership and the segmentation prior is $[0, 1]$, as is the weighted sum of the squared differences between the image and the appearance of layer j under motion Theta. On top of these normalized values, variances $\sigma_{\vec{\mu}}^2$, σ_{ω}^2 and σ_I^2 are applied.

For each foreground layer, the match score accompanying the chosen motion parameters is saved along with the difference with respect to the match score of the previous time instant. This information is used by the state machine to establish if the appearance match for the object is good or degraded.

After motion estimation, the estimated translations and rotations are applied to the respective foreground layers. This way the ownership mask, shape estimation, and appearance estimation directly benefit from an appearance model and prior distribution that better resemble the

actual situation in the current frame. That is, compared to the situation in which the motion parameters would be applied by the state machine.

Shape Estimation The conjugate gradient method (section 3.4) is used to optimize equation 2.19 ($FUNC$) by using its gradient as calculated by equations 2.21 and 2.22 (both $DFUNC$). The time complexity of both functions is

$$\begin{aligned} FUNC &= L + D \times 2 \times L, && (4.6) \\ DFUNC &= 2 \times L + D \times 2 \times L. && (4.7) \end{aligned}$$

Conjugate gradient uses L line minimization's [PTVF02] (worst case L^2). Every line minimization requires one $DFUNC$ evaluations and *a few* $FUNC$ evaluations. Most $FUNC$ evaluations are made during the bracketing of the function minimum. In practice around 35 $FUNC$ evaluations are made for every line minimization, giving an estimated time complexity for shape estimation:

$$\begin{aligned} SHAPE &\approx L \times (DFUNC + 35 \times FUNC) \\ &= 37 \times L^2 + 72 \times D \times L^2. && (4.8) \end{aligned}$$

Appearance Estimation The appearance model of an object is defined in its own coordinate system, as shown in figure 2.3. As explained in section 3.2, in practice the appearance model is not rotated according to angle δ.

The appearance models of the foreground layers are updated using the actual image segmentation s derived from the ownership mask. The appearance model of layer j is updated in four steps:

1. Initialize new appearance model.
2. Translate old appearance model onto I_t.
3. Loop over all pixels. If a pixel belongs to j according to s, update it according to equation 2.24 if the pixel already belonged to the appearance model. Otherwise use its value in I_t. If a pixel does not belong to j, give it a -1 value to indicate its transparent. During this process, the new object bounding box is defined.
4. Use the center of the new bounding box to translate the updated appearance model onto the center of the new appearance model.

The background appearance model is updated in a similar fashion, but without the two translation steps. The time complexity of the appearance estimation (APP) of the foreground layers is:

$$\begin{aligned} APP &= L \times 4 \times D \times B \\ &= 12 \times L \times D \\ &= L \times D. && (4.9) \end{aligned}$$

Overall Complexity As shown in the previous paragraphs, the motion estimation step and the shape estimation step require most computation. During Expectation Maximization, the Bayesian method loops a rough 200 times the number of layers over the dimensions of the image, for every frame. A great deal of speed can be gained from using an other ways to estimation both motion and shape. Suggestions will be proposed in chapter 6.

5 Experiments and Results

This chapter describes the tests performed to compare the Bayesian method of Tao *et al.* [TSK02] with the graph-based method of Conte [CFJV05]. The method of comparison will be elaborated in section 5.1, the basic and advanced test sets will be described in sections 5.2 and 5.3, a short description of the graph-based method is presented in section 5.4, test results are to be found in sections 5.5 and 5.6, and the comparison between both methods is done in section 5.7.

5.1 Method of Comparison

The goal of both algorithms is to detect and track objects in motion video. Events occurring over time can be be divided into two classes. Events in which objects have motion, pass one another at close range, rotate, have slight changes in appearance / shape, become stationary and accelerate are regarded as *basic*. Occlusion is regarded as *advanced*, whether this is due to objects moving in front of one another or to objects moving behind parts of the background. Splitting objects, aggressive changes in appearance, fast changes of shape and objects entering or leaving the scene also belong in the advanced category. Descriptions of all events can be found in sections 5.2 (basic) and 5.3 (advanced).

The algorithms will be assessed on several features, which are briefly introduced below. The quality of detection is left out, since the same method for object detection can be used in both algorithms. The main focus lies on the tracking qualities and speed of the methods.

- **Quality of Tracking** - Indicates how close the tracker can stay with the actual objects, how well the estimated object track fits the real one. Quality of tracking can also be expressed in terms of how well the internal representation of an object matches the actual situation.
- **Computational Effectiveness** - Indicates how fast a tracker can process frames.

Ground Truth To be able to quantitatively measure the performance of both algorithms, *ground truth* is used [SHT+01]. The algorithms output a per frame list of bounding boxes, one for each object currently being tracked. The *bounding box* of an object is the smallest rectangle around the object, completely enclosing its pixels (figure 5.1). The output or *result* list is compared to the user generated ground truth list, also containing a bounding box for each object being present each frame. Human beings make use of a superior tracking mechanism, being able to segment scenes and keeping track of multiple objects in complex occlusion cases, with the least of effort. Although there is a reasonable chance human beings make mistakes during the ground truth generation process (one gets tired of determining bounding box coordinates for thousands of frames), ground truth is assumed to be error free. To be noted is the subjective aspect of ground truth generation. Human judgement may differ when determining which pixels belong to an object, especially when dealing with objects consisting of only a few.

The *track* of an object is the path it travels while present in the scene. Given a ground truth track and a result track, the quantitative error measures (performance metrics) mentioned below are defined in the same manner as in [SHT+01].

- **Object Centroid Position Error** - Objects in the ground truth are represented by bounding boxes. The object centroid position error E_{ocp} is approximated by the distances between the centers of the bounding boxes in ground truth and the centers of the associated objects in the tracking results. This error measure is useful to determine how close the tracker is to the actual position of the object. For a video sequence, E_{ocp} is the average over all frames over all tracks.
- **Object Area Error** - The object area is approximated by the area of the bounding box. Although an object seldom fits perfectly in a bounding box, using the bounding box to determine the size of an object is considered a good approximation. Labeling the exact boundary of an object is a tedious and time consuming piece of work for thousands of frames and therefore not considered an option. The object area error E_{oa} measures how well the object bounding box found by the tracker fits the ground truth bounding box for this object (assumed that is exists in the ground truth). For a video sequence, E_{oa} is the average over all frames over all tracks. A negative outcome of E_{oa} indicates that the output bounding boxes are generally smaller than the ground truth ones. The E_{oa} is illustrated in figure 5.2.
- **False Positive** - A false positive A_{fp} is a tracker result not present in the ground truth.
- **False Negative** - A false negative A_{fn} is a result not found by the tracker that is present in the ground truth.
- **Track Incompleteness Factor** - The track incompleteness factor F_{ti} measures how well the track of an object as determined by the tracker fits the ground truth track of this object:

$$F_{ti} = \frac{\#A_{fp} + \#A_{fn}}{T_i} \tag{5.1}$$

  where $\#A_{fp}$ is the false positive frame count, $\#A_{fn}$ is the false negative frame count, and T_i is the number of frames present in both result and ground truth. The smaller the factor, the better the result.

- **False Positive Rate** - The false positive rate R_{fp} tells something about the amount of non-existing object tracks found by the tracker, normalized with the total number of truly existing tracks in the ground truth. It is defined as

$$R_{fp} = \frac{\text{output tracks which are not present in the ground truth}}{\text{total number of ground truth tracks}}. \tag{5.2}$$

- **False Negative Rate** - The false negative rate R_{fn} tells something about the amount of ground truth tracks not found by the tracker, normalized with total number of truly existing tracks in the ground truth. It is defined as

$$R_{fn} = \frac{\text{ground truth without corresponding output track}}{\text{total number of ground truth tracks}}. \tag{5.3}$$

The computational effectiveness will be expressed in C_{mspf}, the number of milliseconds needed to process one frame, and C_{fps}, the equivalent number of frames per second a method is able to handle.

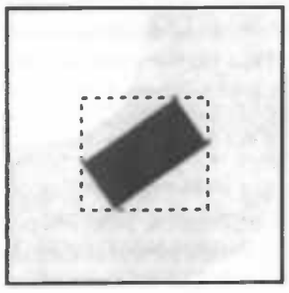


Figure 5.1: A bounding box (dotted line) of an object is the smallest rectangle enclosing all pixels belonging to that object.

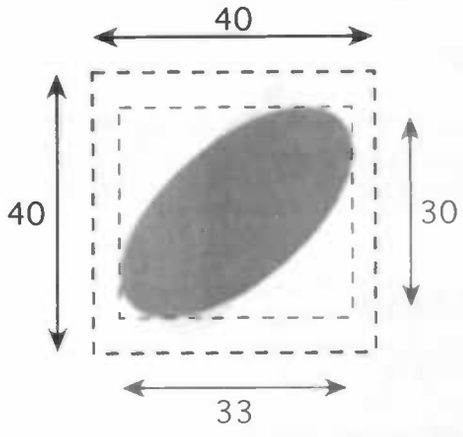


Figure 5.2: The 40×40 dotted rectangle is the ground truth bounding box, the 30×33 dotted rectangle is the bounding box encapsulating the object as tracked by the object tracker. The object area error for this situation is $E_{oa} = -610$.

Test Sets Two types of test sets are used. The hand made, *artificial* sets are created to test the algorithms on specific events. The level of detail is very low, they contain either uniformly faced backgrounds or backgrounds with simple scenery, and the foreground objects have basic shapes. The scenes are viewed top-down, and motion and rotation are limited to the 2D plane. The reason to incorporate artificial sets in both basic and advanced events testing, is that it is important to know how the algorithms perform on abstract representations of the real-world. If the performance on artificial basic and advanced events is satisfactory, those same events can be tested with all problems and difficulties that come with real-world test sets.

The *real-world* test sets are taken from the PETS '01 database [PET01]. The 2001 *Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* used this set to test the participating tracking algorithms. Since the workshop was held, the test set is widely used by authors to compare performance of their new tracking techniques. The PETS '01 set contains a scene with depth, with motion and rotation occurring in 3D. To be able to test the methods on the different events present in the PETS '01 set separately, the set is divided into four subsets. The frame numbering for each set is altered such that it starts at frame 1.

The dimensions of the artificial test sets are 380×285 pixels, the PETS '01 set is 384×288 pixels. Those are typical dimensions of frames produced by inexpensive surveillance hardware. A brief description of the test sets used can be found in table 5.1. A more detailed explanation of the various events in the basic and advanced test sets can be found in sections 5.2 and 5.3.

Test Environment The Bayesian method and the graph-based method are tested on a 2.6 GHz Intel Pentium IV system with 512 Mb RAM. The operating system was version 3 of the Fedora Core Linux distribution.

5.2 Basic Test Sets

Various basic events are tested with the basic test sets. Each set contains a subset of events possible to occur, to be able to test how well the tracker handles each event separately. Each test set is briefly described in words and on a per frame basis. Various sample frames from each test set are also present.

5.2.1 Description of Events

Basic events are described below.

- **Motion** - An object can have a motion $\vec{\mu} = [x, y]$, with at least $x \neq 0$ or $y \neq 0$. Motion in the test sets used is either in the x-direction, y-direction or a combination of both, between -3 and 3 pixels per frame.
- **Rotation / Turning** - An object can rotate with an angle ω. In the test sets used, the rotation in either direction is never greater than 5° per frame. A rotating object forms a challenge because its appearance model is based on the object before rotation and thus its observation model can not fit as good on the current frame situation as without rotation. The same holds for the shape priors: the normalized prior distribution is based on the angle δ of the object before rotation.

- **Passing** - If objects pass one another at close range, a tracker might not be able to correctly associate the objects to their corresponding change blobs.
- **Stationary** - A stationary object has a motion $\vec{\mu} = [0, 0]$. If two consecutive frames are used for change blob determination, stationary objects do not have a change blob and are thus not trivial to be handled.
- **Accelerate** - Stationary objects can start moving again after a certain period of time. Trackers have to associate an already existing object with a change (new) blob.
- **Gradual Change of Shape** - The shape of an object can change over time. This is likely to go very gradually and can for example occur due to rotation in a 3D scene. Nevertheless must the tracker be able to update internal representations of the objects and the frame. The more frames per second a camera produces, the more gradual changes will take place.
- **Gradual Change of Appearance** - Due to lighting conditions that differ over time in outdoor scenes the appearance of an object can change. This is likely to go gradually. Again must the tracker be able to handle this situation and again a higher frame rate is preferable.

5.2.2 Description of Test Sets

Cars Artificial test set *Cars* contains two rectangular objects with a slight gradient from back to front. Both the *blue car* and the *red car* start completely within the frame. The blue car has a dimension of 51 × 26 pixels, the red one takes up 90 × 26 pixels. The events in this test set include motion by 1, 2 or 3 pixels per frame in either the (positive or negative) x-direction, (positive or negative) y-direction or a combination of those two, rotation with an angle of 3° per frame, becoming stationary, accelerating and objects passing each other at close range. The background resembles a sandy road with slaps of green on both sides. Additive Gaussian noise with a σ of 3 is applied to the background, to mimic distortion normal seen in frames produced by cheap surveillance cameras. A per frame description can be found in table 5.2, example frames from the test set can be seen in figure 5.3. The test set is a simplification of a real-world traffic monitoring scene which tests the tracker on handling the basic events mentioned.

Splash Artificial test set *Splash* contains one object with a complex shape and a slight gradient. The splash like, or oil stain like, object starts completely within the frame and moves around with a speed of 1, 2 or 3 pixels per frame, in the same manner as the objects in the Cars test set. The splash also rotates with a maximum of 3° per frame. Like the Cars set, additive Gaussian noise with a σ of 3 is applied to the background. The events occurring in the test set can be found in table 5.3, an example frame can be found in figure 5.4. The main purpose of this set is to see how the tracker handles a complex shape, that is, a shape that does not fit nicely in the shape representation proposed by Tao (section 2.1.3).

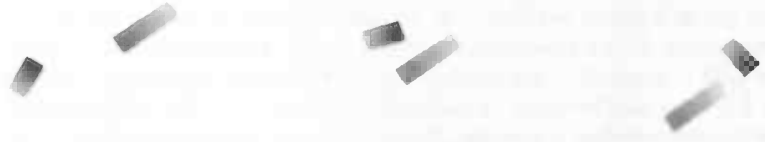
Trailer Artificial test set *Trailer* contains one object, consisting of two independently moving but connected parts. The object resembles a trailer (61 × 21 pixels) with truck (20 × 22 pixels), both having a white to red gradient. The truck moves with a speed of 1, 2 or 3 pixels in the same manner as the objects in the Cars test set. When making a turn, the 3° rotation initiated by the truck is followed by the trailer in a fairly natural way. The combination becomes stationary for

Id	Name	Kind	Frames	Characteristics
1.	Cars	Artificial, basic	107	Two objects, rotation, motion, stationary, acceleration.
2.	Splash	Artificial, basic	98	One object, oil stain shaped, rotation, motion.
3.	Trailer	Artificial, basic	60	One object resembling a truck, rotation, motion, shape change.
4.	Morph	Artificial, basic	120	One object, both shape and appearance change.
5.	Dawn	Artificial, advanced	104	One object, motion, rotation, dramatic scene intensity change.
6.	Set 1	PETS '01, basic	358 (90 - 448)	One object.
7.	Set 2	PETS '01, basic	106 (445 - 512)	Two objects.
8.	Set 3	PETS '01, advanced	224 (538 - 762)	Simple occlusion.
9.	Set 4	PETS '01, advanced	430 (670 - 1100)	Complex occlusion.

Table 5.1: Short description of the test sets used to compare the Bayesian method and the graph-based method. The numbers between brackets indicate the original PETS '01 frame numbers.

Frames	Events
1 - 3	Only the background is visible.
3 - 21	Cars emerge and drive towards each other, with a slight diagonal motion.
22 - 47	Red car is stationary, blue car takes evasive action by turning left, followed by a right turn.
48 - 110	Red car accelerates and continues its way in a straight diagonal line, blue car continues its right turn and passes the rear of the red car. The cars are close to each other during those events.

Table 5.2: Events occurring in the Cars test set.



(a) Frame 35 (b) Frame 70 (c) Frame 102

Figure 5.3: Three sample frames from the Cars test set. (a) Blue car is turning, red car is stationary. In (b) and (c) the blue car moves around the red one, which moves a little to the left (seen from the driving direction).

a little while halfway the sequence. Like the Cars set, additive Gaussian noise with a σ of 3 is applied to the background. For a detailed sequence description see table 5.4, sample frames can be seen in figure 5.5. The intention of this artificial set is to test the trackers on their handling of objects of which the shape of a moving object changes gradually over time.

Morph Artificial test set *Morph* contains one object. While stationary, its shape first grows into the x-direction, followed by growth in the y-direction. This resembles a car turning in a 3D volume from front (or back) view to side view. Next, the appearance of the object changes. The objects fades to grey, resembling the occurrence of shade in a scene. Like the Cars set, additive Gaussian noise with a σ of 3 is applied to the background. A detailed video frame analysis can be found in table 5.5, sample frames are displayed in figure 5.6. The intention of this test set is to test the tracker for gradual changes in both appearance and shape of a stationary object.

Set 1 Real-world PETS '01 test set contains a scene with a t-junction, a parking lot, a range of parked cars, slabs of grass, several two or three story buildings, and a lamppost. *Set 1* is a subset of the PETS '01 sequence. In Set 1 a person enters the scene and moves in a rather straight line towards the bottom-right corner. The person takes up approximately 6×16 pixels on entry and evolves slowly up to 11×24. A sample frame from Set 1 can be found in figure 5.7. Set 1 is used to test the methods on their capability of tracking one object.

Set 2 Real-world *Set 2* test set continues where Set 1 stopped. A blue car enters the scene from the left, and moves towards the person. The car has dimensions of approximately 70×40 pixels. While moving towards the right screen border, the person gets occluded by the lamppost. Although occlusion by a background object belongs in the advanced category, it was more convenient to make it part of this test set than of Set 3. Sample frames from Set 2 can be found in figure 5.8. This set intends to test the trackers on handling two objects and occlusion by a background object.

5.3 Advanced Test Sets

Advanced test sets are used to test the methods on their capability of handling advanced events. The events classified as advanced are described, as well as the test sets containing those events.

5.3.1 Description of Events

Advanced events are described below.

- **Enter / Disappear** - The shape and appearance of an object change dramatically when it enters a scene or disappears from it. The tracker must be able to recognize an object over multiple frames while entering or leaving the frame.
- **Aggressive Change of Shape** - When an object has a fast rotation in a three dimensional scene, its shape can change rapidly. This can also happen when an object moves at high speed, especially in any direction other than perpendicular to the camera viewpoint. When using low frame rate camera equipment, changes over time in shape are also more likely to be aggressive.

Frames	Events
1 - 3	Only the background is visible.
3 - 64	Splash moves from bottom right to the center of the frame by making a wide turn.
65 - 70	Splash becomes stationary.
71 - 81	Splash moves straight down under a slight rotation.
82 - 99	Splash moves straight right, comes to a stop and moves back in the same line.

Table 5.3: Events occurring in the Splash test set.

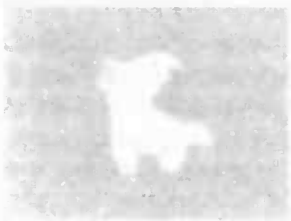
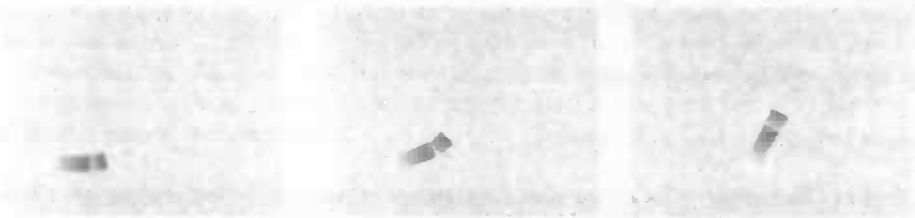


Figure 5.4: One sample frame from the Splash test set.

Frames	Events
1 - 3	Only the background is visible.
3 - 6	Truck and trailer move in a straight line to the right.
6 - 40	Truck initiates a turn, followed by the trailer.
41 - 47	Truck and trailer are stationary.
48 - 60	Track and trailer move in a straight diagonal line in the direction of the top-right frame corner.

Table 5.4: Events occurring in the Trailer test set.

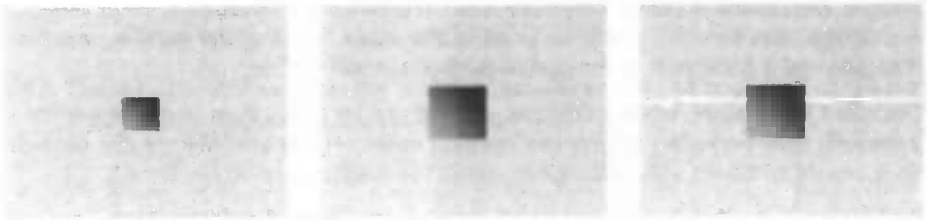


(a) Frame 11 (b) Frame 25 (c) Frame 40

Figure 5.5: Three sample frames from the Trailer test set. In (a) the truck just started its turn, the trailer is not yet turning. (b) shows both truck and trailer in the middle of a turn. Figure (c) shows the combination with a straight line motion.

Frames	Events
1 - 3	Only the background is visible.
3 - 25	49 × 44 pixels object is stationary in center of frame.
26 - 40	Object grows mainly in x-direction to 77 × 46 pixels.
41 - 48	Object is stationary.
49 - 63	Object grows in the y-direction to 77 × 72 pixels.
64 - 120	Appearance of object flows from blue to grey.

Table 5.5: Events occurring in the Morph test set.



(a) Frame 4 (b) Frame 63 (c) Frame 120

Figure 5.6: Three sample frames from the Morph test set. In (a) the initial object is shown. (b) shows the object after growth in both x and y-direction. Figure (c) shows the object after its appearance changed.

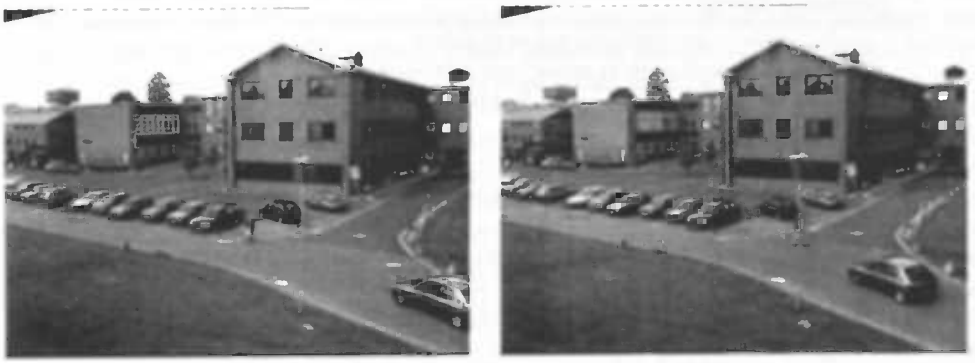
Figure 5.7: Frame $t = 50$ from the Set 1 test set. The person is hardly visible, but can be seen on the road a little to the right of the left image border.

- **Aggressive Change of Appearance** - When an object moves into a shaded area of an outdoor scene, its appearance will change dramatically. The full object appearance is likely to change in several steps depending on speed and frame rate. The possibility exists that the front of the object leaves the shaded area before its rear enters. In all cases, the tracker must be able to cope with fast, temporal changes in appearance.
- **Occlusion with Background** - A foreground object is wholly or partly occluded behind an object that is part of the background. Because this object belongs to the background, it cannot be segmented and is therefore not seen as an object by the tracker. The occluded foreground object will be split in to two parts, become partly visible or completely disappear for a certain period of time. Since the occluding part of the background is not an object, the tracker cannot classify the state of the occluded foreground object as occluded. This type of occlusion will most likely appear in scenes with depth.
- **Simple Occlusion Among Objects** - Two foreground objects pass each other, during which one object moves partly in front of the other. During the passing, motion is not changed and a good part of the occluded object is still visible.
- **Complex Occlusion Among Objects** - There can be more than two foreground objects involved, thus occluded objects can occlude other objects and objects can occlude more than one other object. Also change in motion or rotation can be involved.
- **Splitting of an Object** - If a group of people is walking closely together, they are most likely to be recognized as one object. Not only the group formation can change, but also people can walk away from the group. A similar sort of event occurs when a car is parked and its driver steps out and walks away. In both situations new objects appear. In case of the walking group both shape and appearance of the initial object change drastically.

5.3.2 Description of Test Sets

Dawn Artificial test set *Dawn* contains one rectangular (86 × 49 pixels) object with a blue gradient from back to front. The background resembles a crossing with houses along a horizontal road. Of interest in this sequence is the intensity change of the whole scene. An early sunrise is simulated, where the scene turns from fairly dark to bright white. The object moves like the blue car in the Cars test set and rotates with steps of 5°. A full sequence analysis can be found in table 5.6, samples from the sequence are on display in figure 5.9. This test set is qualified as advanced because of the high speed with which the intensity rises, not only transforming the appearance of the objects but of the background as well. If the tracker lacks a good way of coping with intensity changes it will not only be unable to track the object over time, but also fail in doing useful change blob detection. That can lead to the situation in which the whole background is seen as a change blob.

Set 3 Real-world test set *Set 3* picks up where Set 2 left off. The person continues his (or her) way and passes behind the blue car, which itself continues its movement towards a parking spot in the center of the frame. After passing behind the lamppost, the blue car comes to a full stop on the parking spot. After the person has left the scene on the right, a white minivan (34 × 24 pixels) enters from the left and moves towards the parked car. Sample frames are shown in figure 5.10. A per frame sequence analysis can be found in table 5.7. Set 3 is used to test the methods on tracking three objects in a real-world scene, of which two of them form an occlusion.

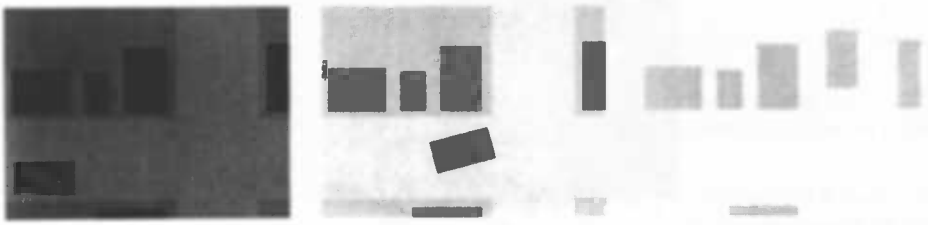


(a) Frame 38 (b) Frame 79

Figure 5.8: Two sample frames from the Set 2 test set. (b) shows the occlusion of the person by a part (the lamppost) of the background.

Frames	Events
1 - 3	Only the background is visible.
3 - 104	Scene intensity (background and object) goes from dark to light in equal steps.
3 - 32	Object moves straight right.
33 - 35	Object makes a small turn.
36 - 63	Object moves in a diagonal line towards the top-right corner.
64 - 79	Object turns further, till it faces straight up.
80 - 104	Object moves straight up.

Table 5.6: Events occurring in the Dawn test set.



(a) Frame 4 (b) Frame 50 (c) Frame 100

Figure 5.9: Three sample frames from the Dawn test set. (a) shows the darkest frame of the sequence, (b) is the neutral situation and (c) shows the brightest frame.

Frames	Events
1 - 3	Only the background is visible.
4 - 20	Car and person move towards each other.
21 - 48	Person is partly occluded by car, both continue along their previous paths.
49 - 114	Occlusion of person by car is over. Car is partly occluded by part of the background (lamppost). Car turns slowly in 3D towards parking spot.
115 - 135	Person leaves the scene on the right. Car slows down.
136 - 224	Car becomes stationary. White mini-van appears from the left and drives towards t-junction.

Table 5.7: Events occurring in the Set 3 test set.

(a) Frame 37

(b) Frame 69

(c) Frame 200

Figure 5.10: Three sample frames from the Set 3 test set. (a) shows the partly occluded person by the blue car, (b) the partly occluded car by a part of the background, and (c) the white minivan.

Set 4 Real-world test set Set 4 contains various complex events and starts at the point where the white mini-van enters the scene (and thus has some overlap with Set 3). The white mini-van continues its way towards the parked car and passes in front of it. Meanwhile a group of three persons enters the scene on the right, followed by a single person entering the scene in the lower left corner. While the group walks in the direction of the three story building, the mini-van drives to the right scene side to become stationary while only partly in the scene. The single person moves over the slab of grass in a perpendicular line, to eventually pass in front of the car. A fifth person appears, while stepping out of the parked car. An analysis of all events occurring can be found in table 5.8. Sample frames are presented in figure 5.11. This set tests the methods on various basic and advanced events, among which a simple occlusion (parked blue car by the white mini-van, and white mini-van by the grass person), the tracking of three objects moving close together (group of three persons), a complex occlusion (group of three persons by the white mini-van), objects entering and leaving the scene, object becoming stationary and accelerating, splitting of objects (person that appears from the parked car), occlusion with a part of the background (mini-van by the lamppost), and gradual changes in shape and appearance due to movement in a 3D scene.

5.4 Graph-Based Multi-Resolution Method

Most object tracking approaches work on a pixel-by-pixel basis, making them virtually unsuitable for real-time application. Conte [CFJV05] presents a tracking method that exploits the wealth of information gained from spatial coherence between pixels, using a graph-based multi-resolution representation of moving regions in a video sequence.

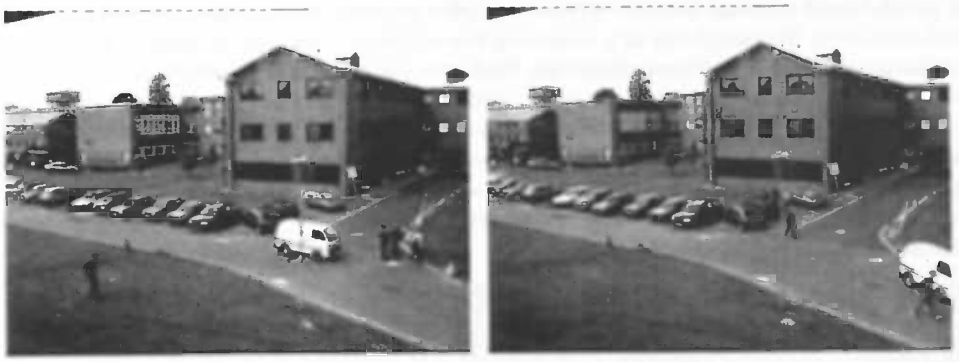
A bounding box provides the simplest representation of a moving region (change blob). It contains enough information to track objects over time in case there are no occlusions present. The graph-based method features a more complex representation based on a graph pyramid, which maintains the simplicity of a bounding box in case occlusions are absent, but offers more accurate matching possibilities otherwise. Each moving region is represented at different resolution levels, using a graph for each level. At the top level of the pyramid, the graph consists of one node and contains information about the bounding box and average color of the represented region, at the lowest level the graph nodes present single pixels of the region and the edges encode the connectivity relation. Intermediate levels are obtained using bottom up classical decimation-grouping, in which node merging is guided by color similarity. An example pyramid is illustrated in figure 5.12.

During the tracking process each pyramid node receives a label associating the (sub)region to the object it belongs to (an object can consist of multiple regions and a region can contain multiple objects). When a (sub)region contains multiple objects, the corresponding node receives a *multilabel* label indicating that the separate objects can be reconstructed using lower pyramid levels.

Given the labeled representation of frames I_{t-1} and current frame I_t, the graph-based method should find a representation for I_t featuring a labeling consistent with the identities of the objects present. Therefore it compares the topmost levels of the pyramids (every moving region has its own graph pyramid) of I_{t-1} with the topmost levels of the pyramids in I_t. The pyramids in I_t contain only a top-level, that is, only the bounding boxes of the moving regions are known. If the outcome of the comparison is sufficient (above threshold T_1) to assign a label to each node, the lower level of the pyramids of I_{t-1} are propagated to their corresponding pyramids in I_t. If the comparison outcome between pyramid m in I_{t-1} and pyramid n in I_t is not

Frames	Events
1 - 3	Only background is visible.
4 - 83	Blue car becomes stationary (parks). White mini-van enters scene on the right, moves along the road to the center of the scene.
84 - 150	White mini-van continues along the road. Group of three people enters from the right.
151 - 219	White mini-van and group of people continue there ways. Person enters the scene from bottom right and walk right on the grass. Mini-van is occluded by the lamppost for a period of time.
220 - 243	White mini-van occludes group of people, grass person continues.
244 - 350	White mini-van occludes group of people, which is underway to the buildings. White mini-van becomes stationary with its nose outside of the frame. Person steps out of the previously parked blue car. Grass person walks towards stationary white mini-van, being occluded by the lamppost for a short period of time.
351 - 430	Person who stepped out of the car begins to move towards mini-van (lamppost occlusion). Group of three people is occluded by lamppost for several frames, while continuing towards the building. Grass person partly occludes white mini-van and leaves the screen.

Table 5.8: Events occurring in the Set 4 test set.



(a) Frame 202 (b) Frame 415

Figure 5.11: Two sample frames from the Set 4 test set. (a) shows a white mini-van, a group of people closely walking together and a person walking on the grass. (b) show the occlusion of the white mini-van by the person previously walking on the grass. The group is now situated between the cars on the parking lot. The driver of the stationary blue car just got out. The difference in size of the people in the scene illustrates the 3D character of the set.

sufficient (below T_1) but still promising (above T_2), the complete graph pyramid for n is constructed and comparison at lower levels of m and n is performed. If the comparison outcome is less than promising, the pyramid is regarded as a new object and initialized as such. In case of an occlusion, the comparison outcome falls between T_1 and T_2.

5.5 Bayesian Method Results

Results from the Bayesian method on the basic and advanced test sets are described below. The artificial sets are processed without using the ground truth bounding boxes to initialize objects, while the PETS '01 sets are processed with the ground truth boxes. This is because the PETS '01 sets contain small objects which were eliminated during noise removal. Although the focus of comparison is not on the detection step, using the detection mechanism of the Bayesian method is preferred over using the ground truth bounding boxes. This is to give a complete view on the abilities of the method. Furthermore, the computational time needed by the detection step is negligible with respect to the time needed by the tracking step.

Parameters What values should be used for the segmentation prior parameters explained in section 2.1.3? The lower constant background prior β, the higher the normalized priors in the prior distributions of the foreground layers will be (figure 2.2). Uncertainty of layer shape γ is a percentage of β, allowing pixels with a high observation model value (image likelihood) to belong to a foreground layer even though they are far away from its center. A high γ (in the range $[0, \beta]$) results in large appearance models, since pixels may be very far away from their respective layer centers. A β of 0.5 yields good results, with a γ of 20%. Lower β's convert the appearance models to much into ovals, as is the result of using higher γ's.

Standard deviation $\sigma_{I,obs}$ (used to construct the observation model) accounts for image noise. The lower $\sigma_{I,obs}$, the harder a pixel mismatch will be punished. A $\sigma_{I,obs}$ that can accommodate for a per band color difference of 10 is used.

Motion parameters tried during motion estimation for the artificial sets are a μ_x and μ_y from the set $\{-2, -1, 0, 1, 2\}$ pixels, and an ω from the set $\{-2°, 0, 2°\}$. Standard deviation $\sigma_{I,mot}$ is chosen in such a way that the third term of equation 2.18 carries most weight, but can still be influenced by the normalized first two terms. The enlargement of the bounding box (section 4.2.3) is set to 10 pixels. For the PETS '01 test sets the same translation set is used. The use of a rotation set proved not useful. Chances are small that objects in a 3D scene rotate in such a way that it can be caught by the simple 2D rotation used during motion estimation. Letting a faster update of the appearance model handle 3D rotations seemed more useful.

Standard deviation σ_{ls} is chosen very low, to prevent the shape prior from growing to enthusiastic. Giving the shape estimation too much space results in faulty motion estimation due to appearance models in which too much of the background is present. This problem is also related to the choices of β and γ.

The appearance estimation of the background layer is done separately from the foreground layers, allowing different standard deviation parameters $\sigma_{I,app}$ and σ_A for each type. In both cases $\sigma_{I,app}$ is set to 1, regulating the update behaviour with σ_A. In case of the artificial test sets, a low σ_A suffices since the appearance of the objects hardly changes. That is not the case for the PETS '01 sets, so there a higher parameter value is used. A σ_A of 0.5 gave good results. For both test set types the background σ_A is 0.1.

Slightly different parameter settings where used for each test test. If radical changes were made to the settings described above, this is noted. Parameter settings for the opening depth, closing depth, change blob threshold, and minimal distance threshold (used by the state machine to determine if an objects is new) are of less importance. Typically, a filter depth of 2, and a threshold of 20 (compared to the sum of the color differences for all bands) are used. A good rule of thumb for the distance threshold is the longest side of the largest object present in the video sequence.

5.5.1 Basic Test Sets

Cars Test results are summarized in table 5.9. All object tracks present in the ground truth are recognized by the method ($R_{fn} = 0/2$) and no false positives are generated, resulting in a flawless track completeness. The E_{ocp} is 4.8 pixels on average, indicating the tracker estimated the motions, and thus locations of the objects, rather precisely over time. The tracker produced a per frame E_{oa} of about -418, indicating that the objects as represented in Λ in general were slightly smaller than they actually are. With an average processing speed of 85 seconds per frame, the method is not able to track the two object at real-time. Figure 5.13 shows the tracking situation on $t = 73$.

Splash Test results are summarized in table 5.10. Tracking results are similar to those of the Cars set. Since there is one object present in Splash, the average per frame processing time is cut in half with respect to the Cars C_{mspf} results. The object centroid position error is slightly bigger, meaning the tracking path deviated more from the ground truth path. Figure 5.14 shows the tracking situation on $t = 80$. Clearly visible is the effect of the simple shape prior on the complex shape, regarding the appearance model.

Trailer Test results are summarized in table 5.11. The trailer results follow the line of the previous two result, only the E_{oa} is larger negative. This is caused by the fact that truck and trailer are one object, but move independently. When the truck deviates from its original line with the trailer, the ground truth bounding box gets larger. It takes a few frames for the appearance model, and thus the size of the output bounding box, to catch up with the actual situation. Figure 5.15 shows the tracking situation on $t = 30$.

Morph Test results are summarized in table 5.12. While stationary and growing, the motion estimation step occasionally estimates a motion, moving the object to the right. The appearance model is smaller than the object it represents because the appearance estimation needs a couple of frames to adapt to the new size, after the shape estimation step has let the appearance model grow. After the object has stopped growing, the shape estimation step continues to estimate a larger l and s until the whole object fits in the appearance model. Due to the motion estimation, the estimated object center does not coincide with the actual center making a slap of the background part of the appearance model. Figure 5.16 shows the tracking situations on $t = 48$ (after growing in the x-direction and $t = 120$ (after appearance change).

A problem discovered while performing tests on sets in which objects shrink, is that the shape estimation step does not adapt to such an event. If an object starts shrinking, its l and s do not change, while its appearance model does update itself. After shrinking, the object is correctly

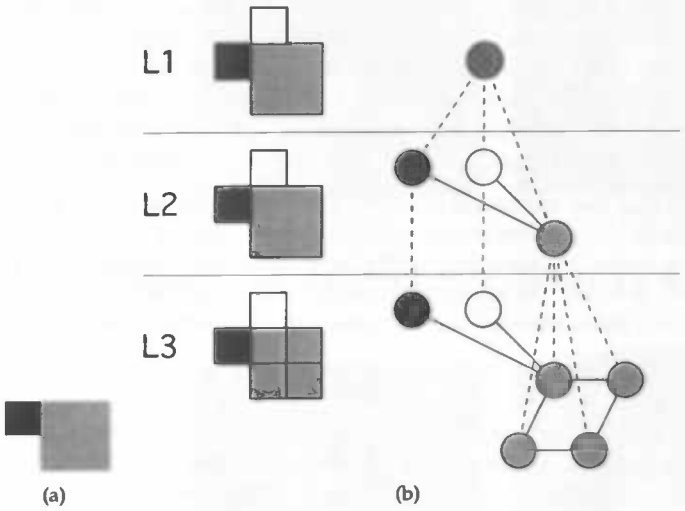


Figure 5.12: Graph pyramid hierarchical structure (b) of a moving region (a).

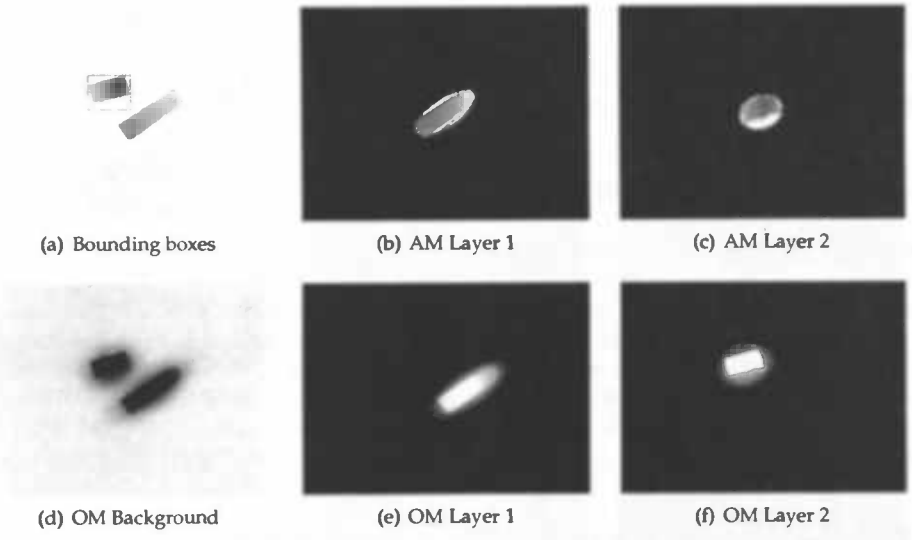


Figure 5.13: Bayesian method test result on Cars set at $t = 71$. Bounding boxes of the objects that are being tracked are projected on the current frame (a). They are based on the appearance models (AM) of object 1 (b) and object 2 (c), and their respective motion estimates from Λ_{71}. (d)-(f) show the tree layers of the ownership mask (OM).

visible in an appearance model that is too large, e.g. it contains too much background pixels. An appearance model in which a part of the background is present negatively influences motion estimation, since this background is also warped on to the current frame while it is supposed to be static. The reason that the shape estimation does not react to a shrinking object represented by layer j, is that prior distribution of j still has a good correlation with the ownership h_j. The latter has not only high values for pixels actually belonging to the object, but also for the background pixels that now belong to j, causing the good correlation.

One might expect that this shrinking problem is solved when an object moves over a textured background, since its observation model will have low values for the background pixels present in the appearance model because they differ from the actual background. Though in practice this does not seem to hold. The appearance model, either updated fast or slow, in combination with the normalized prior distribution still gives an ownership mask that correlates too good during shape estimation.

Set 1 Test results are summarized in table 5.13. During the change blob detection phase, one of the windows of the building showed up as blob (due to reflection it differs much from the background frame). This window being even bigger than the person around $t = 4$, was impossible to filter without also eliminating the person. The use of the ground truth bounding boxes to initialize objects was necessary. Other than that, this real world set was processed without complications. Figure 5.17 shows the tracking situation on $t = 343$.

Set 2 Test results are summarized in table 5.14. The two objects are tracked with a very low E_{ocp}. Both objects are represented slightly bigger than they actually are, as can be seen in figure 5.18 (tracking situation on $t = 76$) and derived from the E_{oa}. The partly occlusion of the person with the lamppost does not trouble the Bayesian method. With respect to the tracking of one object in Set 1, the frame processing time for Set 2 almost doubled.

5.5.2 Advanced Test Sets

Dawn Test results are summarized in table 5.15. To accommodate for the fast intensity build up, a high σ_A for the background appearance model is needed. Due to this high value, parts of the object falling outside of the area the appearance model occupies, are also incorporated in the update process. As can be seen in figure 5.19 the top left and bottom left corners of the blue car are not part of the appearance model, causing the blurry stripe on the background appearance model. Of course this stripe disappears gradually when the object is not located above it anymore. Because of the 5° turns the object makes, the rotation set used is $\{-5°, 0, 5°\}$. Tracking is done as expected after previous tests. The only remark can be made on the high E_{oa}.

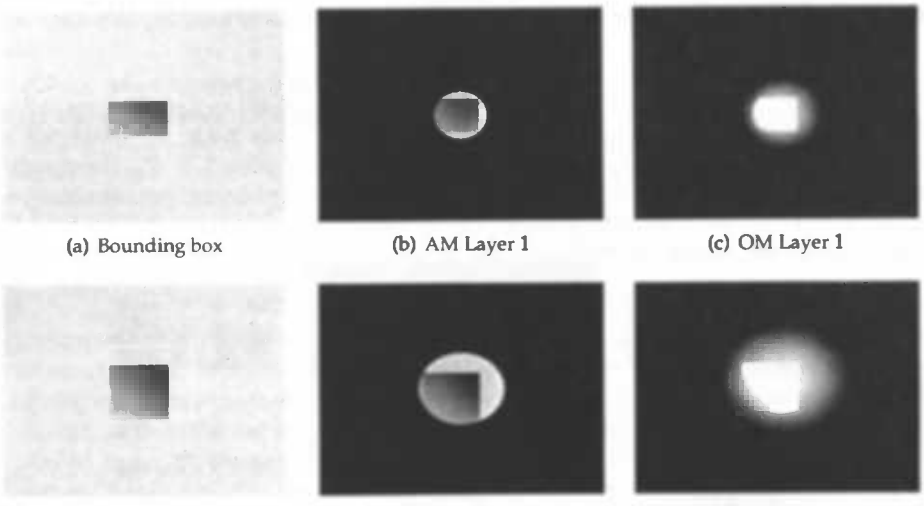
Set 3 Test results are summarized in table 5.16. The Bayesian method handles the occlusion well; the person is tracked correctly while passing partly behind the car. As can be seen in figure 5.20 (tracking situation on $t = 40$), the bottom of the appearance model of the person (layer 1) adopts a few blue car pixels. After the car parked and the person left the scene, a white minivan enters the scene, accounting for the third track found. The positive track incompleteness score is caused by the fact that after the person leaves the scene, it is not deleted for several frames. While leaving, the appearance model of the person incorporates the grass and thus does not move out of bounds.

(a) Bounding box (b) AM Layer 1 (c) OM Layer 1

Figure 5.14: Bayesian method test result on Splash set at $t = 80$.

(a) Bounding box (b) AM Layer 1 (c) OM Layer 1

Figure 5.15: Bayesian method test result on Trailer set at $t = 30$.



(a) Bounding box (b) AM Layer 1 (c) OM Layer 1

(d) Bounding box (e) AM Layer 1 (f) OM Layer 1

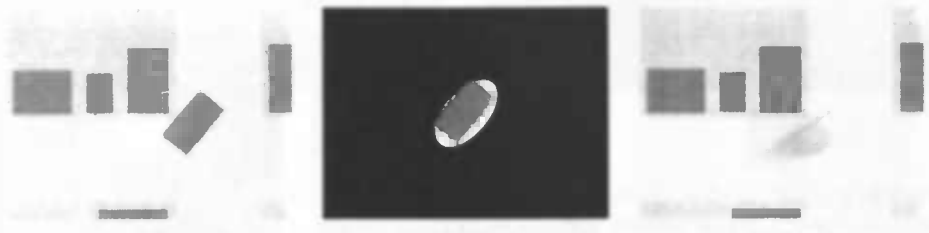
Figure 5.16: Bayesian method test result on Morph set. (a), (b) and (c) show the bounding box imposed on the current frame, the appearance model and ownership mask at $t = 30$. (c), (d) and (e) show those at $t = 120$.

(a) Bounding box (b) AM Layer 1 (c) OM Layer 1

Figure 5.17: Bayesian method test result on Set 1 set. (a) shows the bounding box imposed on the current frame ($t = 343$), (b) is the layer 1 appearance model (zoomed in), and (c) the ownership for layer 1.

(a) Bounding box (b) AM Layer 1 (c) AM Layer 2

Figure 5.18: Bayesian method test result on Set 2 set. (a) shows the bounding boxes imposed on the current frame ($t = 76$), (b) is the layer 1 appearance model (zoomed in), and (c) the appearance model of layer 2 (zoomed in).



(a) Bounding box (b) AM Layer 1 (c) AM Layer 0

Figure 5.19: Bayesian method test result on Dawn set. (a) shows the bounding box imposed on the current frame ($t = 70$), (b) is the layer 1 appearance model, and (c) the background appearance model.

Set 4 Test results are summarized in table 5.17. Not all objects in this set are successfully tracked. The tracking of the white mini-van is successful up to the point it occludes the parked blue car (figure 5.21(a)). The mini-van is recognized as new object just before it gets stationary on the frame border (figure 5.21(b)). This does not happen earlier due to the fact that its change blob is too close to existing objects. As explained in [SHT+01], each ground truth track can be assigned to one or more result tracks, to accommodate for fragmented tracks. Thus both tracks produced by the objects initialized due to the movement of the mini-van, correspond to the mini-van ground truth track.

When the group of three people enters the scene, the first two people overlap and are thus seen as one change blob, initializing one object. Subsequently, the third person who enters right after the other two is too close to the existing objects to be seen as new object and is thus ignored (figure 5.21(a)). This failure causes a $R_{fn} = 2/7$.

The person walking on the grass is not tracked correctly right from the moment he enters the scene (figure 5.21(a)). His contrast with the grass slab is too low for the motion estimation step, causing a better appearance match (the lower the match value the better) for the incorrect motion. The third term of equation 2.18 for the correct motion is actually lower than that of the estimated motion, but not low enough to compensate for the first two terms. After a period of time, the person is far away from existing objects (including its own) and thus initialized again. Again tracking fails right from the start, the new object moves down and out of the scene. The two objects initialized for the grass person do not cause higher false positive or false negative rates, because they both are situated close to the ground truth track of this person.

When the white mini-van occludes the group of people, the motion estimation temporarily fails, but restores itself after the occlusion is over. This is possible because the appearance model still resembles the two persons in such a way that a good appearance match can be made after the occlusion.

The person stepping out of the car (figure 5.21(b), visible within the yellow bounding box of the parked car object) is not initialized as an object because it is too close to already existing objects. This causes an $R_{fn} = 1/7$, setting the total to $R_{fn} = 3/7$. The rather high F_{ti} of 2.40 is caused by the false positives generated by objects drifting off of their ground truth tracks (mini-van, grass person), and the false negatives generated by the absence of objects on their expected ground truth locations (mini-van, grass person, two of the people in the group, the person that steps out of the parked car).

The ground truth does not contain bounding boxes for objects that are stationary for a period longer than n frames. However, the Bayesian method keeps regarding a stationary object as object as long it is present in the scene, hereby causing two occlusion events more than intended by the ground truth. If the method would have eliminated both stationary objects, the occlusion of the parked car with the mini-van, and the occlusion of the mini-van by the grass person would not have taken place. Because the background is used to fill in transparent pixels in the appearance model of an object in several of the estimation steps, it would in this case be important that the background updated itself rapidly with the stationary objects. The absence of ground truth bounding boxes for stationary objects also causes false positives weighing in the F_{ti}.

The E_{ocp} is higher than in previous tests, because of the faulty tracking of the mini-van and the grass person. The E_{oa} is in line with previous results.

(a) Bounding box (b) AM Layer 1 (c) AM Layer 0

Figure 5.20: Bayesian method test result on Set 3 set. (a) shows the bounding boxes imposed on the current frame ($t = 40$), (b) is the layer 1 appearance model (zoomed in), and (c) the appearance model of layer 2 (zoomed in).

(a) $t = 195$ (b) $t = 330$

Figure 5.21: Bayesian method test result on Set 4 set. (a) shows the failing tracking of the mini-van and the grass person. Two of the persons belong to the group are regarded as one object, while the third is ignored. (b) shows that the mini-van is again initialized, while the grass person is not. The appearance model of the group has incorporated the third person.

5.6 Graph-Based Method Results

Results from the graph-based method on the basic and advanced test sets are summarized below. For all tests, ground truth bounding boxes are used to detect objects present in a frame. The graph-based method currently not includes a detection mechanism of its own. Parameter settings for the thresholds are taken from the author; $T_1 = 0.95$ and $T_2 = 0.65$. Pictures clarifying the way the method handles the tracking internally, are not available.

5.6.1 Basic Test Sets

Cars Test results are summarized in table 5.9. This set does not form any problem for the graph-based method. The two objects are processed (bounding boxes on t and $t-2$ are associated) at a speed of 4 frames per second, with high tracking precision.

Splash Test results are summarized in table 5.10. With respect to the Cars set the frames per second rate more than doubles to almost 11, since the Splash only contains one object. The oil stain-like shape of the object causes a slight E_{oa}.

Trailer Test results are summarized in table 5.11. Tracking of the Trailer set is done in the line of the previous two sets. A slightly negative E_{oa} indicates objects were represented smaller then they actually are.

Morph Test results are summarized in table 5.11. The frame rate at which the Morph set is processed is a surprising 2 frames per seconds. Comparing bounding boxes of different sizes or comparing bounding boxes in which the object changed of appearance, apparently consumes more processing time.

Set 1 Test results are summarized in table 5.13. The frame rate at which the Set 1 set is processed is again surprising: 52 frames per second. This is caused by the fact that the object tracked consists of a merely 10×20 pixels on average. The very low E_{ocp} and E_{oa} indicate tracking is done precisely.

Set 2 Test results are summarized in table 5.14. Tracking results on the Set 2 set are in the line of previous tests, except for the high A_{ae}. Visual tracking output (figure 5.22) shows a bounding box problem, including the person in the bounding box of the car several frames. Those errors boost the A_{ae}. The occlusion of the person with the background lamppost does not trouble the tracker.

5.6.2 Advanced Test Sets

Dawn Test results are summarized in table 5.15. The graph-based method handles this set perfectly, with only a small E_{ocp}. The change in appearance of the whole scene results in a performance rate of 1.7 frames per second. This is in line with the frame rate at which the

morph set was processed. Rapid changes in appearance do not seem to form a problem for the graph-based method, though its frame per second rate is lower than in cases the appearance stays the same.

Set 3 Test results are summarized in table 5.16. The occlusions present in Set 3 trouble the graph-method. As shown in figure 5.23, during the occlusion the tracking of the object representing the person (white bounding box) fails. A new object (red bounding box) is initialized for this person, while the other object previously representing him (or her) floats with the car. The problems with the person and car occlusion result in the R_{fp} of $1/3$.

The car passing behind the lamppost is troublesome when the occlusion just started; the car is split in two parts, where the front part is regarded as new object (green bounding box). The association of the back part (blue bounding box) with the previous car objects is made. When the car is halfway behind the lammpost, it is recognized as one object again. The tracking of the white minivan is without problems.

Set 4 Test results are summarized in table 5.17. With a R_{fn} and R_{fn} of both zero, and a low F_{ti}, the results look good. There is a catch here though, illustrated in figure 5.24. Every person in the group entering from the right is tracked as single object, recognizable by a red, green, and yellow bounding box (figure 5.24(a)). Next, the group is occluded by the mini-van. At this point the tracker loses track of all group members (figure 5.24(b)). After the occlusion each group member is seen as new object, this time recognizable by a light blue, dark blue, and white bounding box (figure 5.24(c)). Since multiple result tracks can be assigned to one ground truth track, the two object tracks belonging to one person in the group are taken as one track.

The F_{ti} of 0.60 is a result of the false negatives caused by the absence of result bounding boxes for the group members during their occlusion with the van. The tracking of all objects present is done fast (14 frames per second on average) en precise (R_{ae} and R_{ocp} are both low) by the graph-based method. Even the person stepping out of the car is tracked correctly (figure 5.24(d), red bounding box). The remark here is that the objects are detected based on their ground truth bounding boxes, but still the tracker associates them correctly over time. The big down side here is that the graph-based method does not handle the occlusion at all. The tracker is not able to tell that the group is occluded, and does not associate the individual group members before the occlusion to the same members after the occlusion.

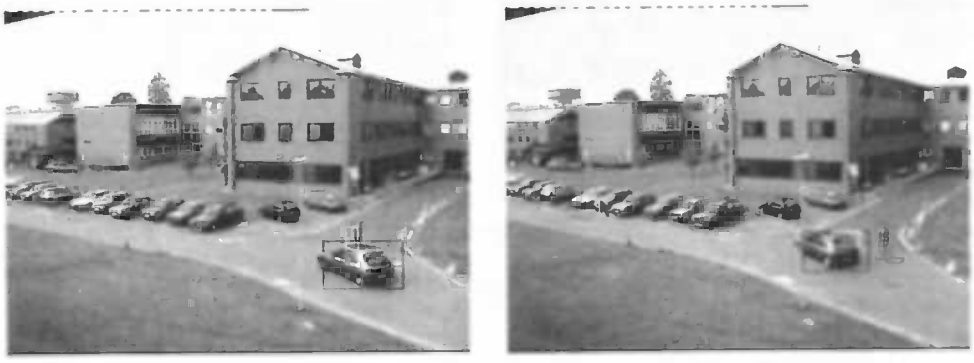
Since the object detection is based on ground truth and ground truth does not contain bounding boxes for objects stationary for a longer period of time, the tracker does not attempt to track stationary objects as long as they are present in the frame. This result in the absence of two occlusions that would be present otherwise. The encounter between the mini-van and the parked car (figure 5.24(a)) is not an occlusion, as holds for the encounter between the grass person and the mini-van (figure 5.24(d)).

5.7 Comparison

The conclusion that can be drawn from the test results is rather straight forward. Quality of tracking is similar for both methods on artificial sets, while the computational effectiveness of

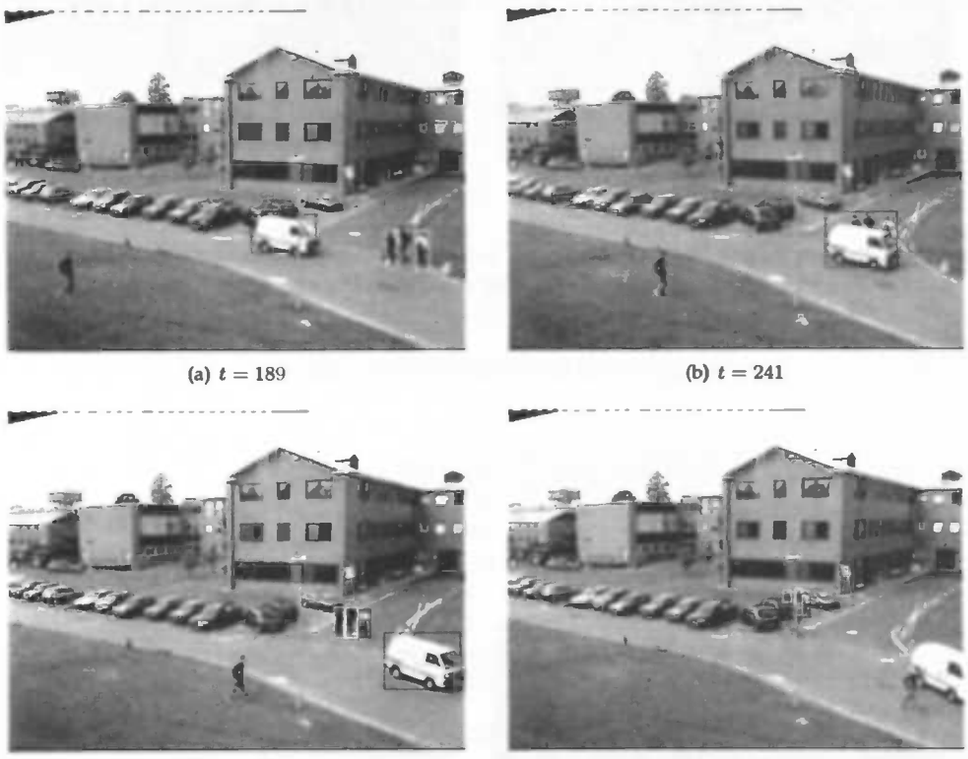
(a) Bounding boxes (b) Blob detection

Figure 5.22: Graph-based method test result on Set 2 set. (a) shows the bounding boxes imposed on the current frame ($t = 50$), (b) shows the detected change blobs within the ground truth boxes.



(a) $t = 31$ (b) $t = 56$

Figure 5.23: Graph-based method test result on Set 3 set. Bounding boxes as determined by the tracker are imposed on the scene at (a) $t = 31$, and (b) $t = 56$.



(a) $t = 189$ (b) $t = 241$

(c) $t = 285$ (d) $t = 402$

Figure 5.24: Graph-based method test result on Set 4 set.

the graph-based method is much better than that of the Bayesian method for this class. Real-time tracking is impossible with the Bayesian method, while the graph-based method can handle multiple frames per second.

Real-world test sets form a challenge for the Bayesian method when contrast between objects and background is low. When able to track the objects, the Bayesian method performs good on simple occlusions. The graph-based method has no problems with real-world tracking, except when occlusions are present. In the latter case this method gets confused and loses track of the objects involved in the occlusion.

Chapter 6 will present a discussion, and improvements on the Bayesian method will be suggested.

5.7.1 Basic Test Sets

Both the Bayesian method and the graph-based method show a good quality of tracking. The E_{ocp} is low and almost equal for both methods. The (absolute) E_{oa} is generally larger for the Bayesian method, but still within acceptable limits (see figure 5.2 to get an idea of how the E_{oa} builds up). No false positive or false negative tracks are detected. The major advantage of the graph-based method is its computational effectiveness. All basic sets are processed with more than one frame per second, making it suitable for real-time tracking. With a rough 50 seconds per object per frame, the Bayesian method is far from being able to track in real-time.

5.7.2 Advanced Test Sets

While not perfect, the Bayesian method does a better job at handling occlusions than the graph-based method. The latter has not enough information to establish the correct correspondences between the objects just before, during and after the occlusion. It did only handle the occlusion with a part of the background correctly, while failing on the other cases.

In the real-world sets, the graph-based method beats the Bayesian method on the quality of tracking when no occlusions are present. A lack of enough contrast between the objects and the background, as well as blurry appearance models, do not help the Bayesian method's motion estimation to determine the correct movement of an object. Once the motion of an object is estimated incorrectly, its appearance model is updated with background pixels making subsequent correct tracking more unlikely.

In principle, the Bayesian method has enough information to handle partly occlusions, primary in the form of the appearance model. Nevertheless does this method has difficulties with occlusions as well. Lacking a good system of labeling objects as occluded or occluding (using the degraded appearance match alone is not sufficient), in combination with the absence of special measures (do not update the appearance model of an occluded object, try to match a part of the appearance model of an occluded object on the current frame) makes the tracking of objects through occlusions error prone. Tests done on artificial sets containing partly occlusions (not included), were handled correctly. The artificial sets had high contrast (grey background, red rectangle partly occluded by a blue one) between objects and background, making correct motion estimation more likely. Occlusions can be handled by the Bayesian method, but improvements have to be made to make it successful in real-world sequences.

Measure	Bayesian	Graph-based
E_{ocp}	4.80	2.00
E_{oa}	-418.02	0.00
F_{ti}	0.00	0.00
R_{fp}	0/2	0/2
R_{fn}	0/2	0/2
C_{mspf}	85189.5238	240.9434
C_{fps}	0.0117	4.1504

Table 5.9: Test results on the Cars test set (107 frames).

Measure	Bayesian	Graph-based
E_{ocp}	5.10	6.70
E_{oa}	-267.38	33.32
F_{ti}	0.00	0.00
R_{fp}	0/1	0/1
R_{fn}	0/1	0/1
C_{mspf}	47337.9167	91.1340
C_{fps}	0.0211	10.9728

Table 5.10: Test results on the Splash test set (98 frames).

Measure	Bayesian	Graph-based
E_{ocp}	4.10	2.70
E_{oa}	-955.30	-59.95
F_{ti}	0.00	0.00
R_{fp}	0/1	0/1
R_{fn}	0/1	0/1
C_{mspf}	48418.4483	106.9492
C_{fps}	0.0207	9.3502

Table 5.11: Test results on the Trailer test set (60 frames).

Measure	Bayesian	Graph-based
E_{ocp}	19.00	2.0000
E_{oa}	1588.23	0.0000
F_{ti}	0.0000	0.0000
R_{fp}	0/1	0/1
R_{fn}	0/1	0/1
C_{mspf}	43009.6610	523.8655
C_{fps}	0.0233	1.9088

Table 5.12: Test results on the Morph test set (120 frames).

Including rotation in the motion estimation step, is not useful in case of 3D scenes. It is unlikely an object turns exactly in the 2D plane perpendicular to the camera. Other rotations are too complex to be fathomed by the simple rotation estimation. Letting a fast appearance model update cope with appearance and shape changes seems useful, but enlarges the chance of background pixels becoming part of the appearance model and thus incorrect motion estimation.

The graph-based method beats the Bayesian method also on speed, for both artificial and real-world sequences. The artificial set containing aggressive changes in appearance was handled at a rate just above one frame per second, while both real-world sets were processed at 14 frames per second on average. The Bayesian method has a rate well below one frame per second for both types of sets.

Measure	Bayesian	Graph-based
E_{ocp}	2.70	2.70
E_{oa}	-18.39	-0.65
F_{ti}	0.00	0.00
R_{fp}	0/1	0/1
R_{fn}	0/1	0/1
C_{mspf}	51206.7135	19.1877
C_{fps}	0.0195	52.1168

Table 5.13: Test results on the Set 1 test set (358 frames).

Measure	Bayesian	Graph-based
E_{ocp}	4.00	10.20
E_{oa}	191.76	3290.31
F_{ti}	0.00	0.00
R_{fp}	0/2	0/2
R_{fn}	0/2	0/2
C_{mspf}	90416.9231	104.5714
C_{fps}	0.0110	9.5628

Table 5.14: Test results on the Set 2 test set (106 frames).

Measure	Bayesian	Graph-based
E_{ocp}	6.00	2.00
E_{oa}	8301.72	0.00
F_{ti}	0.00	0.00
R_{fp}	0/1	0/1
R_{fn}	0/1	0/1
C_{mspf}	47954.0196	563.3981
C_{fps}	0.0209	1.7749

Table 5.15: Test results on the Dawn test set (104 frames).

Measure	Bayesian	Graph-based
E_{ocp}	7.40	8.10
E_{oa}	175.78	45.61
F_{ti}	0.33	0.40
R_{fp}	0/3	1/3
R_{fn}	0/3	0/3
C_{mspf}	117989.5045	68.8991
C_{fps}	0.0085	14.5140

Table 5.16: Test results on the Set 3 test set (224 frames).

Measure	Bayesian	Graph-based
E_{ocp}	61.50	2.90
E_{oa}	27.28	114.19
F_{ti}	2.40	0.60
R_{fp}	0/7	0/7
R_{fn}	3/7	0/7
C_{mspf}	87536.02323	68.5417
C_{fps}	0.0114	14.5897

Table 5.17: Test results on the Set 4 test set (432 frames).

6 Discussion and Improvements

To compare the tracking performance of the Bayesian method of Tao *et al.* with the graph-based method developed by Conte, the former needed to be implemented. Test results showed that the Bayesian method handles occlusions better than the graph-based method, but does not always satisfactorily track single objects in real-world video sequences. This chapter will present a discussion trying to clarify why the Bayesian method fails in this situation. Furthermore suggestions are proposed to increase the computational efficiency, which is far from real-time, and tracking capability of the method.

Discussion As seen in the test results, the Bayesian method is not always able to track objects in real-world test sets. Objects in real-world sets have complex appearances and do not move with whole pixels at at the time, as is the case in artificial sets. The Bayesian method tries to track an object based on the internal representation it has of that object, in the form of the appearance model. This appearance model is updated every frame, to keep it close to the actual appearance of the object. The shape of the object (segmentation prior) is used in combination with how well its appearance model fits on the current frame (observation model) to form an ownership mask, which indicates which pixels belong to which objects. This ownership mask steers the shape of the appearance model, by indicating which pixels should be updated.

The appearance model is used during motion estimation and the construction of the observation model, thus playing an important role. Motion estimation is likely to fail when the appearance model is not resembling the current appearance enough. When motion estimation is incorrect, the ownership mask will contain high values for the wrong pixels, causing an update of the appearance model with pixels not belonging to the object. Here a circle emerges, pulling the estimation in the wrong direction further and further.

The appearance model is the weakness, but directly also the strength of the method. If the appearance model is good, the motion estimation will be correct. In case of an occlusion, the method has knowledge about the appearance and motion of the object before it was occluded. Those two pieces of information are valuable and can help the method to detect if an object *is* occluded and track it through the occlusion. Since the Bayesian method was developed for an aerial tracking system, its occlusion handling capabilities are not utilized completely. Even tough, results of the method on simple occlusion events were promising. Efforts in extending the occlusion capabilities of the Bayesian method will most likely be rewarding.

The author of the Bayesian method was able to reach tracking rates of ten frames per seconds for two objects and five frames per second with four. Test results do not come even close to those numbers, indicating a rate much lower than one frame per second for all test cases. This difference is most likely caused by the specialized hardware used by the author, but can also be caused by aspects of the implementation. To be able to use the method on cheap hardware platforms, improvements on its efficiency are essential. Suggestions for improvements are made at the end of this chapter.

The graph-based method is fast. With a good change blob detection layer it is able to do accurate, real-time tracking of objects in case they are not taking part in an occlusion. Objects close to each other do not form problems and also small objects are tracked correctly. The fast pyramid comparison technique to associate objects over time, fails in case of an occlusion. The tracker does not have enough information to be able to associate objects just before, during, and just after tracking. To be able to handle occlusions, the graph-based method needs a more sophisticated internal representation containing more information on the objects it is tracking.

Improvements While implementing and testing the Bayesian method, ideas for improvement were born. Unfortunately the amount of time available for the research did not permit to test and implement them all. Ideas are listed below.

- **Skip Frames** - To gain speed, a fixed number of frames can be skipped every n-th frame, thereby effectively lowering the frame rate. When the input video sequence has a high frame rate, objects tend to move and change at a low pace. Skipping frames will not affect the ability of a method to track objects, as long as the resulting average object motion (in pixels per frame) is in line with the motion estimation parameters. Appearance and shape estimation parameters have to be adjusted to accommodate for faster changes as well.

- **Limit Motion Search Area** - During motion estimation, the appearance model of an object is warped on to its currently known location p (thus at $t-1$) on the current frame and subsequently different combinations of translation and rotation are tried to determine how the object moved from the previous frame to the current. For every parameter combination, all pixels are regarded to form an appearance match value. This value is constructed by multiplying the ownership of a pixel with its difference in color in the appearance model and the current frame. Since the ownership value for pixels far away from the object center is very low, differences in color far away from the object center will not add significantly to the match value. Therefore a search window can be defined, which has to include the area for which the ownership values for the layer under review are high. For the object at p, this is exactly its bounding box. Motion estimation will translate and rotate the object with p as starting point, which makes an enlargement of the bounding box necessary. When the maximal translation is three pixels and the maximal rotation is two degrees, a bounding box enlarged by ten pixels in every direction can accommodate for all combinations. This improvement has been implemented.

- **Reinitialize Object on Border** - When an object enters or leaves the scene, its shape and appearance change rapidly. Motion estimation with an appearance model that does not resemble the appearance of its object in the current frame is likely to fail, causing a wrong appearance model update and wrong shape estimation, making future tracking even more unlikely to succeed successfully. To help the tracker under these circumstances, the appearance model, object center, and shape parameters are reinitialized when the change blob associated to the object is situated on the frame border. Once the object is fully visible in the frame, the tracker should be able to track it. Problems arise when an occlusion takes place on the frame border, since objects then will be reinitialized with a change blob containing all objects participating in the occlusion. This improvement has been implemented.

- **Steer Motion Estimation with Change Blob** - When an object is not participating in an occlusion, its change blob could be used to steer its motion estimation. The center of the

change blob associated to an object indicates the new center of that object in the current frame. The translation from $t-1$ to t can be derived directly from the currently known object center (that is at $t-1$) and the center of its associated change blob. By performing principal components analysis on the change blobs, the rotation of the object can also be derived. Because a change blob is situated over multiple objects in case of an occlusion, it cannot be used to steer motion estimation in this case.

- **Steer Shape Estimation with Change Blob** - As mentioned in chapter 5, the shape estimation step cannot cope with shrinking objects. When an object gets smaller, its appearance model is updated accordingly, while the surrounding pixels within the normalized prior distribution still have a high ownership value and are still seen as part of the object. The appearance model containing the smaller object surrounded by background, still has a high appearance match, so the shape prior does not shrink. In case an object is not occluded, its associated change blob can be used to reinitialize its shape parameters. This will result in a correct segmentation distribution and thus an ownership mask which will only contain high values for the correct pixels. The ownership mask will then let the appearance model shrink to its correct shape.

- **Occlusion Determination** - The Bayesian method features as weak occlusion detection mechanism, based solely on the appearance match of an object. If an object has a degraded appearance match, it is regarded as occluded. A stronger detection mechanism should be devised, using the knowledge that in case of an occlusion multiple objects are assigned to one change blob, and trying to identify which parts of the blob belong to which objects. Motion and appearance estimation can subsequently be done using only the part of the object that is visible, while maintaining its hidden part. This of course can only be done in case of a partly occlusion. Dealing with complete occlusion can be done by freezing the appearance model of an object that completely disappeared (there is no part of the to the object associated change blob, that can be assigned to the object), and zeroing its current motion. The association of change blobs to objects that are known to be occluded completely, should not only be based on the distance of their centers, but also (or completely) on the appearance match. This would cause an completely occluded object to be associated to the correct change blob again, as soon it is not part of the occlusion anymore. Even if it changed motion.

- **Downscale Frames** - The computational performance of the Bayesian method is directly related to the dimensions of a frame (section 4.2.3). When input frames are scaled down before being fed to the change blobs detection and Expectation Maximization steps, a speed up is gained. The down side of frame scaling is that objects might become too small to track, or even disappear. The objects of interest in traffic surveillance applications often are large enough to make down scaling a good possibility. This improvement has been implemented.

- **Motion Blobs** - The Bayesian method makes use of change blobs, detected by comparing the current frame to a background frame without any objects present. If change blobs are detected using multiple consecutive frames (section 3.1), stationary objects will not have such a *motion* blob. An advantage of this system is that objects lacking an associated motion blob do not need to participate in the motion estimation step. When their motion blob returns, their motion can be estimated again. When using change blobs, simply determining that an object is stationary and then leaving it out of motion estimation will not work. If no motion estimation is done for a stationary object, it will not be detected when it starts moving again.

- **Reinitialize Appearance** - A good appearance model is essential for correct motion estimation, which is essential for correct appearance and shape estimation. If an object is not occluded, its appearance model can be reinitialized every n-th frame, based on its associated change blob.

- **Pixel Comparison** - Pixels are compared using the average of their Red, Green, and Blue components. Although widely used, using the *RGB* color space for comparison has the disadvantage that its three color components can have one average value while representing visually different colors. The HSV (Hue Saturation Value) color space is fundamentally different from the RGB color space since it separates out the intensity (luminance) and shade variations from the color information (chromaticity). Using the color component from the HSV color space as dominant comparison factor, could yield better results. Normalized RGB also decouples intensity and color, thus could also be an alternative. The conversion of RGB to Normalized RGB is simpler that that of RGB to HSV.

- **Stationary Objects** - Let the appearance update of the background layer also update the background behind objects, but at a lower rate than the visible parts. This would cause stationary objects to eventually lose their change blobs, indicating they can be removed from the estimation process. Objects staying stationary for a long period of time, are unlikely to continue with moving around. Currently objects are kept till they leave the scene, meaning stationary objects are taking unnecessary computational time.

7 Conclusion and Future Work

This Master's Thesis described the three parts of the research project conducted at the MIVIA department, University of Salerno, Italy. The main goal was to compare the graph-based method, developed at the MIVIA department by Donatello Conte, to a promising object tracking method featuring a layered image representation, on their ability to track through an occlusion.

During the first part in which a layered image representation method had to be selected, the Bayesian method of Tao *et al.* [TSK02] seemed promising. It was able to process video sequences at real-time, and its dynamic layer representation offered tracking possibilities in case of occlusions.

The implementation part took, as can be expected, more time than initially reserved. The estimation of motion, shape, and appearance raised questions about the initialization of objects, the right moment to invoke the state machine, and the best way to implement the appearance model. Different approaches were tried, not always with the expected results.

For the testing part a database with artificial test sets was developed, to be able to test both methods on a specific basic and advanced events in a controlled environment. Also a real-world test set was used. Test results showed the Bayesian method indeed has promising tracking through occlusion possibilities, but is not able to handle the task in more complex situations. The computational performance of the method is not real-time. Improvements have been implemented to speed the method up, and suggestions have been made for further improvements. The graph-based method can process sequences above real-time, but cannot handle occlusions.

Future research should be focused on a hybrid solution, with the graph-based method as basis. To be able to handle occlusions, features of the Bayesian method can be used. As long as there is no occlusion present, the graph-based method in its current form is sufficient to accurately track objects at real-time. Besides the multi-resolution pyramid used to represent an object, its appearance model should be saved internally, like it is with the Bayesian method. This appearance model can be sampled every n-th frame and is based on the change blob associated to an object. As soon as an occlusion is detected, the shape prior for the occluded object can be derived from its appearance model, and object center and motion can be derived from the track laid by the object up to the last sampled situation before occlusion. Estimation of motion and appearance, as seen in the Bayesian method, is done using only the visible part of the object in the current frame and the equivalent part of its appearance model. Since more computational time is needed to perform those estimations, camera frames should be buffered during the occlusion. The graph-based method is fast enough to eliminate the gap with the real-time current frame once the occlusion is over.

An other option to invest future time in, is to optimize the detection of an object being occluded. Once the Bayesian method state machine can give this state to objects with certainty, motion estimation can be done in a faster way. In case object A is not occluded, translation estimation is done by directly calculating the difference between the center of the change blob associated to A and the center of A, its angle can be derived from the change blob as well (with principal component analysis). In case an association between an object and a change blob is not sufficiently

certain, the appearance model of the object can be used to find the correct blob. In case of an occlusion, the Bayesian method could try to estimate the motion and a part of the appearance, in the same manner suggested in the previous paragraph. The shape estimation step should be removed. It is a lot faster to calculate the shape parameters every frame from the change blob associated to an object, in case it is not participating in an occlusion. In case the assignment of the 'occluded' state to an occluded object fails, tracking goes wrong. Appearance models of occluded objects will be updates using the blob covering all objects taking part in the occlusion, making motion estimation using only the visible part of the object impossible.

Acknowledgments I would like to thank Professor Nikolai Petkov of the Intelligent Systems group of the Department of Computing Sciencea at the University of Groningen, the Netherlands, for giving me the opportunity to research and write my Master's Thesis abroad. The same gratitude needs to be expressed to Professor Mario Vento at the University of Salerno, Italy, whom I would also like to thank for defining the subject of my Thesis and helping me along the way. Help, conversations, and suggestions on the subject, given by Donatello Conte, who is currently working on his Ph.D. Thesis in the same field of research, were very much appreciated.

List of Tables

List of Figures

Bibliography

[BFBB94] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, February 1994.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[CFG+04] D. Conte, P. Foggia, C. Guidobaldi, A. Limongiello, and M. Vento. An object tracking algorithm combining different cost functions. *Lecture Notes in Computer Science*, 3212:614–622, 2004.

[CFJV05] D. Conte, P. Foggia, J. Jolion, and M. Vento. A graph-based, multi-resolution algorithm for tracking objects in presence of occlusions. In *Lecture Notes in Computer Science*, volume 3434, pages 193–202, January 2005.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, November 1977.

[GB98] M. Gelgon and P. Bouthemy. Determining a structured spatio-temporal representation of video content for efficient visualization and indexing. In *5th European Conference on Computer Vision*, Freiburg, June 1998.

[HS85] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, January 1985.

[IA99] M. Irani and P. Anandan. All about direct methods. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*. Springer-Verlag, 1999.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[MK96] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, New York, 1996.

[PET01] PETS'2001. http://www.cvg.rdg.ac.uk/vs/pets2001/. In *2nd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, December 2001.

[PTVF02] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 2002.

[RP66] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494, 1966.

[SDC04] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):479–494, April 2004.

[SHT+01] A. Senior, A. Hampapur, Y. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. In *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance, PETS 2001*, 2001.

[SK94] W. Skarbek and A. Koschan. Colour image segmentation — a survey. Technical report, Technical University Berlin, 1994.

[SS01] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[TSK02] H. Tao, H.S. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):75–89, January 2002.

[WA93] J.Y.A. Wang and E.H. Adelson. Layered representation for motion analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 361–366, New York, June 1993.

[WAB06] J. Wills, S. Agarwal, and S. Belongie. A feature-based approach for dense segmentation and estimation of large disparity motion. *International Journal of Computer Vision*, 68(2):125–143, June 2006.

[Wei97] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 520–527, 1997.

[Wik06] Wikipedia. http://en.wikipedia.org/, may 2006.

[ZL01] D. Zhang and G. Lu. Segmentation of moving objects in image sequence: A review. *Circuits, Systems, and Signal Processing*, 20(2):143–183, March 2001.