Motion Detection and Object Tracking in Image Sequences

Ph.D. Thesis

Zoran Živković

June 5, 2003

Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school.
ASCI dissertation series number: 91

Cover design by Sayaka Honsho, shonsho@hotmail.com

ISBN: 90-9016982-2, printed by FEBODRUK B.V., Enschede.

MOTION DETECTION AND OBJECT TRACKING IN IMAGE SEQUENCES

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 5 juni 2003 om 16.45 uur

door

Zoran Živković
geboren op 13 juli 1975
te Zaječar, Servië

Dit proefschrift is goedgekeurd door:

prof. dr. ir. P.P.L. Regtien, PROMOTOR
dr. ir. F. van der Heijden, ASSISTENT-PROMOTOR

Preface

Four years I have been here in Enschede, the Netherlands. It seems like a liftime now. My thesis is finished and I am grateful to the members of the PhD committee: promotor P.P.L. Regtien, assistant-promotor and my daily supervisor F. van der Heijden, F.C.A. Groen, B. Kovačević, J. van Amerongen, W. Jonker for their time and comments. There are many people who were a part of my life during the last four years. Many of them I met here. They are all directly and indirectly involved in the accomplishment of this thesis. I want to mention some of them:

- my family
- Tanja
- all the members of the Laboratory for Measurement and Instrumentation, University of Twente: my supervisors Ferdi and Paul, the secretary Joan, the technicians Alfred and Egbert, the rest of the crew: Maarten, Nanno, Kees, Zweitze, Gloria and Ciska.
- the colleague PhD students: Jaap, Nicolai, Jianbo, Valer. Jaap was my roommate and he was always there to help me and to answer any question I had.
- the students I supervised: Rogier, Rudy and René and all other students that passed through the Laboratory MI.
- all the players from the basketball courts. Most of all, the superstars from the 3HA3 group: Boki, Boris, Dule, Žika, Marko, Ljuba, Petko, Raša, Vojkan, Toma, Mihajlo, Igor, Saša, Vojin, Goran G., Goran P., etc.
- some other friends from Enschede: Blaza, Jelena, Darja, Caroline, Irina, Olga, Tijana, Marija, Marko (also known as Hidden Marko), Nataša V., Nataša J., Nataša G., Katarina, Biljana, Dessy, Richard, Tomaso, Laura, etc.

- I have moved five times during last four years and I must mention my flatmates: Jelmer, Remco, Jaap, Jikke, Jikkenien, Annegien, Guy, Dragana, Katja, Dagmar, the last member of the family Perez - my (Karen's) goldfish and Katja's dog Aris.
- all the athletes from the absolutely best gymnastics club 'Linea Recta' and the trainer Job.
- the students from the Dutch Art Institute who are now Masters of Art: Sayaka (she designed the cover of the thesis and I am very grateful to her), Brana, Jasper, Karen, Katie, Kerstin, etc.
- the miracle called internet and www.google.com.
- all the people from the swimming pool (Karen and Katie belong also here)
- the people from the volleyball club Harambee and my teammates from the teams: '3-meter Sessies' and 'Heren Nijntje'.
- my Dutch language class at the ROC institute and my best pals there: Aardy and Atilla.
- all other people I met during my last four years living in the Netherlands and all my old friends from Serbia.

Thank you.

Zoran Živković Enschede, May 2003.

Contents

Chapter 1

Introduction

Artificial intelligence is an important topic of the current computer science research. In order to be able to act intelligently a machine should be aware of its environment. The visual information is essential for humans. Therefore, among many different possible sensors, the cameras seem very important. Automatically analyzing images and image sequences is the area of research usually called 'computer vision'. This thesis is related to the broad subject of automatic extraction and analysis of useful information about the world from image sequences. The focus in this thesis is on a number of basic operations that are important for many computer vision tasks. These basic steps are analyzed and improvements are proposed. The thesis is divided into three parts: statistical modeling, motion detection and motion measurements. Each part corresponds to one of the basic tasks that were considered. The parts are described separately next in this chapter. Beside proposing the new solutions, the new algorithms are applied to a number of practical problems and working demonstrational systems were built. From the huge number of possibilities the attention given to the applications usually named 'looking at people' where the goal is to detect, track and more generally to interpret human behavior. Although in the 80's it was considered among the hardest areas of computer vision and one of the least likely to have a quick success, the 'looking at people' has become a central topic of the computer vision research today. The applications that receive particular attention in this thesis are: surveillance/monitoring and visual user interfaces.

The outline of the thesis is given next. The three parts of the thesis and the corresponding chapters are shortly introduced. The chapter 8, the last one, brings some final conclusions and some personal views that resulted from this work.

1.1 Statistical modeling

One of the reasons for the early success of the computer vision systems is in proper application of the well-established pattern recognition and statistics techniques. A basic general task is to estimate the probability density function from the data. An intelligent system should be able to gather data from the environment and use this data to adapt and to learn on-line. The first part of this thesis analyzes the problem of recursive (on-line) probability density estimation. Furthermore, finite mixtures and in particular the mixtures of Gaussian distributions are used. A finite mixture is a probability density function that is presented as a weighted sum of simpler base density functions. The components of the Gaussian mixture are the Gaussian distributions. The Gaussian mixture is a flexible model appropriate for many situations and used very often in the computer vision area.

Chapter 2 presents an algorithm for recursive estimation of the parameters of a finite mixture. A stochastic approximation can be used to get the recursive equations and that is standard theory. However, the recursive equations suffer from the common problem that the estimation highly depends on proper initial parameter values. Furthermore, we also need to specify the number of components of the mixture a priori. Choosing the appropriate number of components for the given data is also a standard topic from the literature. Based on some recent results and some approximations, an algorithm is proposed to choose the appropriate number of components and estimate the parameters of the mixture in an recursive procedure. The algorithm starts with a large number of components. Then the unimportant components are identified and discarded. Simultaneously the parameters are estimated. The new algorithm is also less sensitive to the initial parameter values.

Chapter 3 brings a further elaboration on the problem. In chapter 2 the data was coming from a stationary process. A more general problem is considered here. The data is now coming from a process that can have some stationary periods but also some sudden or gradual changes in data statistics are possible. An extension of the algorithm from chapter 2 is presented. The new algorithms can adapt to the changes. Both the parameter values and the number of components are adapted. The algorithm can be essential for many practical on-line systems to quickly get an up to date compact model for the data.

1.2 Motion detection

The scene analysis often starts with segmenting the foreground objects from the background. This basic image sequence processing step is the topic of the second part of this thesis. The focus is on the real-time surveillance systems. The foreground segmentation algorithm should be robust and able to adapt to difficult and changing conditions. Furthermore, only the common case is analyzed when the camera is mostly static.

Chapter 4 presents an analysis of the common pixel-based background foreground/ background segmentation. The assumption is that the images of the scene without the intruding objects exhibits some regular behavior and that the scene can be described by a probability density function for each pixel in the image. If the statistical model of the scene is available the foreground objects are detected by spotting the parts of the image that don't fit the scene model. The main problem is updating and adapting the scene model. An efficient algorithm that has an adaptive Gaussian mixture for each image pixel is developed using the results from the previous chapters. Further, another efficient algorithm using a non-parametric k nearest neighbors approach is proposed. The algorithms are evaluated and compared.

Chapter 5 briefly presents two practical scene analysis applications where the foreground/ background segmentation was the basic image processing step. The first application is a traffic monitoring problem. The algorithms from the previous chapter were directly applied since the camera was static. Final demonstrational system was able to automatically extract some important traffic parameters and detect some traffic events of interest. The second application was a more challenging case of tennis game matches. Using the extracted silhouette of a tennis player, the movements of the player are recognized using an appropriate set of features. Two interesting and timely problems of a practical nature are considered and the results could be of interest to many professionals in the field including archivists, broadcasters and the law enforcement sector. Although very specific, the two applications have many elements that are important for any surveillance/monitoring system.

1.3 Measuring motion

Detecting objects was analyzed in the previously described chapters. Tracking objects is another basic operation a computer should perform in order to understand the environment. The image motion or 'optical flow' can be defined as the movement of the image patterns in an image sequence. This basic motion is important for many computer vision tasks and closely related to the object tracking problem. Measuring the motion of a single point in the image presents an 'ill-posed' problem. However, it is usually reasonable to assume that the points from a small image neighborhood have some similar motion. The movement is then calculated for a small image patch by searching the next image from the sequence for a similar patch. In a similar way an object can be tracked. A larger part of the image is considered then and therefore a more elaborate model is needed to model the possible transformation from one image to another. This type of object tracking is usually known as 'template matching'. The third part of the thesis presents some improvements for the basic image motion problem and a simple 3D object tracking scheme using template matching.

Chapter 6 gives an analysis of the problem of choosing the points in an image for calculating the image movement. These points are usually called 'feature points'. Not every point from an image is suitable for computing the optical flow. This problem is known as 'the aperture problem'. Consider for example an area of uniform intensity in an image. The movement of a small patch within the area would not be visible and the calculated optical flow will depend on noise. There are some standard procedures for selecting suitable points. This chapter points out that most feature point selection criteria are more concerned with the accuracy, rather than with the robustness of the results. A way of estimating the 'region of convergence' is proposed. The size of the 'region of convergence' can be used as a measure of feature point robustness.

Chapter 7 presents a simple heuristic for efficient object tracking based on the template matching. The focus in this chapter is on face tracking but the results are generally applicable. In a tracking scheme an object is first detected and then tracked. We use a simple generic 3D model to describe the transformations between the initial object appearance and the subsequent images. However there are many deformations and changes not covered by this model. Standard approach is to use a database of different appearances of the object to model the possible changes statistically. However this is not generally applicable since in many situations we don't know a priori what kind of object we are going to track. We propose a simple generally applicable heuristic that updates the initial object appearance using new images.

Part I

Statistical modeling

Chapter 2

Recursive Unsupervised Learning

There are two open problems when finite mixture densities are used to model multivariate data: selecting the number of components and initialization. In this chapter an on-line (recursive) algorithm is proposed to simultaneously estimate the parameters of the mixture and select the number of components. The new algorithm is not sensitive to the initialization. A prior is used as a bias for maximally structured models. A stochastic approximation recursive learning algorithm is proposed to search for the maximum a posteriori solution.

2.1 Introduction

The finite mixture probability density models have been analyzed many times and used extensively for modeling multivariate data [16, 8]. In [3] an efficient heuristic was proposed to select compact models for the available data. A certain type of prior is introduced that presents, in a way, a bias for more structured models. This can be used for the finite mixtures to simultaneously estimate the parameters and select the appropriate number of components. This prior has no closed form solution and an incremental search procedure was proposed. A similar heuristic and another prior wes used in [6], but only as a step in a standard model-selection scheme (for standard model selection schemes see section 2.1).

The work we present here is inspired by the mentioned ideas from [3] and [6]. Our contribution is in developing an on-line version. We use a stochastic approximation procedure to estimate the parameters of the mixture recursively. We propose a way to use the mentioned prior from [6] in the recursive equations. Consequently, we can select the number of components of the mixture on-line. In addition, the new algorithm can arrive at the similar results as the previously reported algorithms but much faster.

In section 2.2 we introduce the notation and discuss the standard problems with the finite mixtures. In section 2.3 we describe the previously mentioned idea to simultaneously estimate the parameters of the mixture and select the number of components. Further, in section 2.4 we develop an on-line version. The final practical algorithm we used in our experiments is described in section 2.5. In section 2.6 we demonstrate how the new algorithm performs for a number of standard problems.

2.2 Problem definition

In this section we introduce the finite mixtures and the related problems.

2.2.1 Estimating the parameters

Let us denote the probability density function of a stochastic vector variable $\vec{x}$ as $p(\vec{x};\vec{\theta})$ where the vector $\vec{\theta}$ contains the function parameters. A mixture density with M components can be written as:

$$p(\vec{x};\vec{\theta}(M)) = \sum_{m=1}^{M} \pi_m p_m(\vec{x};\vec{\theta}_m) \text{ with } \sum_{m=1}^{M} \pi_m = 1 \tag{2.1}$$

where $\vec{\theta}(M) = \{\pi_1,..,\pi_M,\vec{\theta}_1,..,\vec{\theta}_M\}$. The number of parameter depends on the number of components M we decide to use. The mixing densities are denoted by $p_m(\vec{x};\vec{\theta}_m)$ where $\vec{\theta}_m$ are the parameters. The mixing weights denoted by π_m are positive. An example is the mixture of the three Gaussian distributions given in table 2.1 and further discussed later. The finite mixtures are a powerful tool for clustering purposes (see for example the 'Iris' data-set we used in our experiments). However, the finite mixtures are also very effective as a general tool for modeling multivariate data (for example the 'Shrinking spiral' data-set analyzed later, table 2.2).

Let us denote the set of the available data samples by $\mathcal{X} = \{\vec{x}^{(1)},...,\vec{x}^{(t)}\}$. The standard tool for estimating the parameters of a mixture model from the data is the 'Expectation Maximization' (EM) algorithm [4]. The EM algorithm can be used to perform an incremental local search for the maximum likelihood (ML) estimate:

$$\widehat{\vec{\theta}} = \max_{\vec{\theta}}(\log p(\mathcal{X}; \vec{\theta}))$$

Apart from its simple implementation and its global convergence to a local maximum, one of the serious limitations of the EM algorithm is that it can end up in a poor local maximum if not properly initialized. The selection of the initial parameter values is still an open question that was studied many times. Some recent efforts are reported in [17, 18].

2.2.2 Selecting the number of components

Note that in order to use the EM algorithm we need to know the appropriate number of components M. Too many components will 'over-fit' the data. Choosing an appropriate number of components is important. Sometimes, for example, the appropriate number of components can reveal some important existing underlying structure that characterizes the data. Full Bayesian approaches sample from the full a posteriori distribution with the number of components M considered unknown. This is possible using Markov chain Monte Carlo methods as reported recently [11, 10]. However, these methods are still far too computationally demanding. Most of the practical model selection techniques are based on maximizing the following type of criteria:

$$J(M, \vec{\theta}(M)) = \log p(\mathcal{X}; \vec{\theta}(M)) - P(M) \tag{2.2}$$

Here $\log p(\mathcal{X}; \vec{\theta}(M))$ is the log-likelihood for the available data. This part can be maximized using the EM. However, introducing more mixture components always increases the log-likelihood. The balance is achieved by introducing the increasing function $P(M)$ that penalizes complex solutions. Some examples of such criteria are: 'Akaike Information Criterion' [1], 'Bayesian Inference Criterion' [14], 'Minimum Description Length' [12], 'Minimum Message Length' (MML) [19] etc. For a detailed review see for example [8].

2.3 Solution using MAP estimation

Suppose that we have a prior for the mixture parameters $p(\vec{\theta}(M))$ that has similar properties as the function $P(M)$ from (2.2) that penalizes complex solutions. Instead of (2.2) we could use:

$$\log p(\mathcal{X}; \vec{\theta}(M)) + \log p(\vec{\theta}(M)) \tag{2.3}$$

The procedure is then as follows. We start with a large number of components M. Maximizing (2.3) is equal to searching for the 'maximum a posteriori' (MAP) solution. This could be done using for example the EM algorithm. The prior presents, in a way, a bias for more compact models. While searching for the MAP solution, the prior drives the irrelevant parameters to extinction. In this way we simultaneously estimate the parameters and reduce the number of components M until the balance is achieved.

The mixing weights define a multinomial distribution which presents a building block for the mixture densities (see for example [7], Chapter 16). The recently proposed algorithm [6] is using the Dirichlet prior for the underlying multinomial distribution:

$$p(\vec{\theta}(M)) \propto \exp \sum_{m=1}^{M} c_m \log \pi_m \tag{2.4}$$

If the parameters c_m are negative (improper prior), the Dirichlet prior has some interesting properties. It can be shown that the standard MML criterion (mentioned in section 2.2) in the standard form (2.2) when written for the finite mixtures has a part that has the form (2.3). The prior introduced in this way is the Dirichlet prior with the coefficients c_m equal to $-N/2$ where N presents the number of parameters per component of the mixture. See [6] for details. This gives the theoretical background for using the improper Dirichlet prior and provides a way to choose the parameters c_m.

The parameters c_m have a meaningful interpretation. For a multinomial distribution, c_m presents the prior evidence (in the MAP sense) for a certain class m (number of samples a priori belonging to that class) . Negative evidence means that we will accept that the class m exists only if there is enough evidence for the existence of the class. In this sense the presented linear connection between c_m and the number of parameters N, that follows from the MML criterion, seems very logical. Consequently, while searching for the MAP solution, when a mixing weight π_m becomes negative the component can be removed. This also ensures that the mixing weights stay positive.

2.4 Recursive (on-line) solution

Let all the parameters of the Dirichlet prior (2.4) be the same $c_m = -c$ (for the finite mixtures we use $c = N/2$ as it was mentioned in the previous section). The Dirichlet prior is the conjugate prior for the multinomial distribution (see [7]) and we get the following simple MAP solution:

$$\hat{\pi}_m = \frac{1}{K}(\sum_{i=0}^{t} o_m(\vec{x}^{(i)}) - c) \tag{2.5}$$

where $K = \sum_{m=1}^{M}(\sum_{i=0}^{t} o_m(\vec{x}^{(i)}) - c) = t - Mc$ (since $\sum_{m=1}^{M} o_m = 1$) is the normalization constant that takes care that the weights sum up to one. For a stand-alone multinomial distribution, the 'ownerships' o_m have values 0 or 1 indicating which class m the sample belongs to. For the mixture densities we get the 'soft' ownerships given by (2.6). The given equation (2.5) is used as one of the steps of the EM algorithm in [6] and we are going to analyze here a recursive version.

A recursive procedure uses a current estimate $\hat{\vec{\theta}}^{(t)}$ for the first t data samples and the new data sample $\vec{x}^{(t+1)}$ to get the new estimate $\hat{\vec{\theta}}^{(t+1)}$. For very large data-sets this can greatly reduce the computation costs since in this way we run through the data only once, sequentially. Without a natural conjugate prior for the mixture densities we need some stochastic approximation procedure. We use the general stochastic approximation from [13]. The connection with the EM algorithm is described in [15]. See also appendix A. However, the procedure is also very sensitive to the initial conditions and in the beginning (small t) the equations tend to be very unstable. The modifications proposed later in this paper overcome these problems.

This stochastic approximation procedure leads to simple and intuitively clear recursive version of (2.5) given by:

$$\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + (1 + t - Mc)^{-1}(o_m^{(t)}(\vec{x}^{(t+1)}) - \hat{\pi}_m^{(t)})$$

with the 'ownerships':

$$o_m^{(t)}(\vec{x}) = \hat{\pi}_m^{(t)} p_m(\vec{x}; \hat{\vec{\theta}}_m^{(t)}) / p(\vec{x}; \hat{\vec{\theta}}^{(t)}) \tag{2.6}$$

However, it is not clear how this equation could be used since $1 + t - Mc$ is negative in the beginning (for small t).

Let us now take one more look at the non-recursive version of the update equations. After dividing (2.5) by t, we get:

$$\hat{\pi}_m = \frac{\hat{\Pi}_m - c/t}{1 - Mc/t}$$

where $\hat{\Pi}_m = \frac{1}{t}\sum_{i=0}^{t} o_m(\vec{x}^{(i)})$ is the standard ML estimate and the bias from the prior is introduced through the c/t. The equation holds if we assume that none of the components was discarded (we need at least $Mc/t < 1$).

The bias from the previous equation decreases for larger data sets (larger t). For a recursive procedure, the data set is constantly getting larger. However, if a small bias is acceptable we can keep it constant by fixing the c/t to some constant value $c_T = c/T$ with some large T. This means that the bias will be always the same as if it would have been for a data sample with T samples. The fixed bias leads to the following well behaved and easy to use recursive update equation:

$$\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + (1+t)^{-1}\left(\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{1-Mc_T} - \hat{\pi}_m^{(t)}\right) - (1+t)^{-1}\frac{c_T}{1-Mc_T} \tag{2.7}$$

Here T should be sufficiently large to make sure that $Mc_T < 1$. Furthermore, simply as in the non-recursive version, when $\hat{\pi}_m^{(t+1)} < 0$ we can discard this component while working on-line.

The most commonly used mixture is the Gaussian mixture. A mixture component $p_m(\vec{x};\vec{\theta}_m) = \mathcal{N}(\vec{x};\vec{\mu}_m, C_m)$ has the mean $\vec{\mu}_m$ and the covariance matrix C_m as the parameters. For the Gaussian mixture we get also simple equations for the rest of the mixture parameters:

$$\widehat{\vec{\mu}}_m^{(t+1)} = \widehat{\vec{\mu}}_m^{(t)} + (t+1)^{-1}\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{\hat{\pi}_m^{(t)}}\vec{\delta}_m \tag{2.8}$$

$$\hat{C}_m^{(t+1)} = \hat{C}_m^{(t)} + (t+1)^{-1}\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{\hat{\pi}_m^{(t)}}(\vec{\delta}_m\vec{\delta}_m^T - \hat{C}_m^{(t)}) \tag{2.9}$$

where $\vec{\delta}_m = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}$. See for example [15] for details.

2.5 A simple practical algorithm

For an on-line procedure it is reasonable to fix the influence of the new samples by replacing the term $(1+t)^{-1}$ from the recursive update equations (2.8), (2.9) and (2.7) by $\alpha = 1/T$. There are also some practical reasons for using a fixed small constant α. This reduces the problems with instability of the equations for small t. Furthermore, fixed α helps in forgetting the out-of-date statistics (random initialization and component deletion) more rapidly.

For the sake of clarity we present here the whole algorithm we used in our experiments. We start with a large number of components M and with a random initialization of the parameters (see next section for an example). We have $c_T = \alpha N/2$. Further, we use Gaussian mixture components with full covariance matrices. Therefore, if the data is d-dimensional, we have $N = d + d(d+1)/2$ (the number of parameters for a Gaussian with full covariance matrix). The on-line algorithm is then given by:

- **Input:** new data sample $\vec{x}^{(t+1)}$, current parameter estimates $\widehat{\vec{\theta}}^{(t)}$
- calculate 'ownerships': $o_m^{(t)}(\vec{x}^{(t+1)}) = \hat{\pi}_m^{(t)} p_m(\vec{x}^{(t+1)}; \widehat{\vec{\theta}}_m^{(t)}) / p(\vec{x}^{(t+1)}; \widehat{\vec{\theta}}^{(t)})$
- update mixture weights: $\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + \alpha(\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{1-Mc_T} - \hat{\pi}_m^{(t)}) - \alpha\frac{c_T}{1-Mc_T}$
- check if there are irrelevant components: if $\hat{\pi}_m^{(t+1)} < 0$ discard the component m ($M = M - 1$) and renormalize the remaining mixing weights
- update the rest of the parameters:
  - $\widehat{\vec{\mu}}_m^{(t+1)} = \widehat{\vec{\mu}}_m^{(t)} + w\vec{\delta}_m$ (where $w = \alpha\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{\hat{\pi}_m^{(t)}}$ and $\vec{\delta}_m = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}$)
  - $\hat{C}_m^{(t+1)} = \hat{C}_m^{(t)} + w(\vec{\delta}_m\vec{\delta}_m^T - \hat{C}_m^{(t)})$ (tip: limit the update speed $w = \min(20\alpha, w)$)

  Output: new parameter estimates $\widehat{\vec{\theta}}^{(t+1)}$

This simple algorithm can be implemented in only a few lines of code. The recommended upper limit 20α for w simply means that the updating speed is limited for the covariance matrices of the components representing less then 5% of the data. This was necessary since $\vec{\delta}\vec{\delta}^T$ is a singular matrix and the covariance matrix may become singular if updated too fast.

2.6 Experiments

In this section we demonstrate the algorithm performance on a few standard problems. We show the results of 100 trials for each data set. For the real-world data-sets we randomly sample from the data to generate longer sequences needed for our sequential algorithm. For each of the problems we present in table 2.2 how the selected number of components of the mixture was changing with sequentially adding new samples. Further, we present one of the common solutions from the trials. Finally, the number of components that was finally selected is presented in the form of a histogram for the 100 trials.

The random initialization of the parameters is the same as in [6]. The means $\widehat{\vec{\mu}}_m^{(0)}$ of the mixture components are initialized by some randomly chosen data points. The initial covariance matrices are a fraction (1/10 here) of the mean global diagonal covariance matrix:

$$C_m^{(0)} = \frac{1}{10d} trace\left(\frac{1}{n}\sum_{i=1}^{n}(\vec{x}^{(i)} - \widehat{\vec{\mu}})(\vec{x}^{(i)} - \widehat{\vec{\mu}})^T\right) I$$

where $\widehat{\vec{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\vec{x}^{(i)}$ is the global mean of the data and I is the identity matrix with proper dimensions. We used first $n = 100$ samples. It is possible to estimate this initial covariance matrix recursively. Finally, we set the initial mixing weights to $\hat{\pi}_m^{(0)} = 1/M$. The initial number of components M should be large enough so that the initialization reasonably covers the data. We used here the same initial number of components as in [6].

2.6.1 The 'Three Gaussians' data set

First we analyze a three-component Gaussian mixture (see table 2.1). It was shown that the EM algorithm for this problem was sensitive to the initialization. A modified version of the EM called 'deterministic anealling EM' from [17] was able to find the correct solution using a 'bad' initialization. For a data-set with 900 samples they needed more than 200 iterations to get close to the solution. Here we start with $M = 30$ mixture components. With random initialization we performed 100 trials and the new algorithm was always able to find the correct solution simultaneously estimating the parameters of the mixture and selecting the number of components. Similar batch algorithm from [6] needs about 200 iterations to identify the three components (on a 900 sample data-set). From the plot in table 2.2 we see that already after 9000 samples the new algorithm is usually able to identify the three components. The computation costs for 9000 samples are approximately the same as for only 10 iterations of the EM algorithm on a 900 sample data-set. Consequently, the new algorithm for this data set is about 20 times faster in finding the similar solution as the previously mentioned algorithms. In [9] some approximate recursive versions of the EM algorithm were compared to the standard EM algorithm and it was shown that the recursive versions are usually faster. This is in correspondence with our results.

Empirically we decided that 50 samples per class are enough and used $\alpha = 1/150$. In table 2.1 we present the final estimates from one of the trials (also presented in table 2.2 by the '$\sigma = 2$ contours' of the Guassian components). We also used the last 150 samples and a properly initialized EM algorithm to find the ML estimates. The results are very similar (table 2.1).

Parameters $\vec{\theta}$	True values	Final estimate	ML estimate (last150 samples)
π_1	0.33	0.35	0.33
π_2	0.33	0.30	0.29
π_3	0.33	0.35	0.37
$\vec{\mu}_1$	$\begin{bmatrix} 0 & -2 \end{bmatrix}^T$	$\begin{bmatrix} 0.15 & -1.99 \end{bmatrix}^T$	$\begin{bmatrix} 0.25 & -2.06 \end{bmatrix}^T$
$\vec{\mu}_2$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$	$\begin{bmatrix} 0.10 & -0.02 \end{bmatrix}^T$	$\begin{bmatrix} -0.01 & -0.01 \end{bmatrix}^T$
$\vec{\mu}_3$	$\begin{bmatrix} 0 & 2 \end{bmatrix}^T$	$\begin{bmatrix} 0.09 & 1.99 \end{bmatrix}^T$	$\begin{bmatrix} 0.02 & 1.95 \end{bmatrix}^T$
C_1	$\begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 1.83 & -0.08 \\ -0.08 & 0.23 \end{bmatrix}$	$\begin{bmatrix} 1.59 & -0.13 \\ -0.13 & 0.25 \end{bmatrix}$
C_2	$\begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 2.17 & 0.03 \\ 0.03 & 0.22 \end{bmatrix}$	$\begin{bmatrix} 2.30 & 0.00 \\ 0.00 & 0.20 \end{bmatrix}$
C_3	$\begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 2.51 & 0.12 \\ 0.12 & 0.20 \end{bmatrix}$	$\begin{bmatrix} 2.80 & 0.09 \\ 0.09 & 0.22 \end{bmatrix}$

Table 2.1: The three Gaussians data set

2.6.2 The 'Iris' data set

We disregard the class information from the well-known 3-class, 4-dimensional 'Iris' data-set [2]. From the 100 trials the clusters were properly identified 81 times. This shows that the order in which the data is presented can influence the recursive solution. The data-set had only 150 samples (50 per class) that were repeated many times. We can expect that the algorithm would perform better with more data samples. We used $\alpha = 1/150$. The typical solution in table 2.2 is presented by projecting the 4-dimensional data to the first two principal components.

2.6.3 The 'Shrinking Spiral' data set

This data-set presents a 1-dimensional manifold ('shrinking spiral') in the three dimensions with added noise:

$$\vec{x} = \begin{bmatrix} (13 - 0.5t)\cos t & (0.5t - 13)\sin t & t \end{bmatrix} + \vec{n}$$

with $t \sim Uniform[0, 4\pi]$ and the noise $\vec{n} \sim \mathcal{N}(0, I)$. The modified EM called 'SMEM' from [18] was reported to be able to fit a 10 component mixture in about 350 iterations. The batch algorithm from [6] is fitting the mixture and selecting 11, 12 or 13 components using typically 300 to 400 iterations for a 900 samples data set. From the graph in table 2.2 it is clear that we achieve similar results but much faster. About 18000 samples was enough to arrive at a similar solution. Consequently, again the new algorithm is about 20 times faster. There are no clusters in this data-set. It can be shown that fixing the influence of the new samples by fixing α has as the effect that the influence of the old data is downweighted by a exponential decaying envelope $S(k) = \alpha(1-\alpha)^{t-k}$ (for $k < t$). For comparison with the other algorithms that used 900 samples we limited the influence of the older samples to 5% of the influence of the current sample by $\alpha = -\log(0.05)/900$. In table 3.1 we present a typical solution by showing for each component the eigen-vector corresponding to the largest eigen-value of the covarance matrix.

2.6.4 The 'Enzyme' data set

The 1-dimensional 'Enzyme' data-set has 245 data samples. It was shown in [11] using the MCMC that the number of components supported by the data is most likely 4, but 2 and 3 are also good choices. Our algorithm arrived at similar solutions. In the similar way as before we used $\alpha = -\log(0.05)/245$.

2.7 Conclusions and discussion

The new algorithm was able to solve difficult problems and to arrive at similar solutions as other much more elaborate algorithms. Compared to the previously proposed algorithms the new algorithm is also faster. However, the most important advantage of the new algorithm is that the data is processed sequentially. When the data also arrives sequentially, as for many on-line working systems, a recursive procedure may be essential to give 'quick' up-to-date parameter estimates.

Introducing a prior and using the MAP estimation to select compact models is an efficient heuristic. However, this leaves some open questions (see also the discussion in [3]). The influence of the prior we used is fixed by taking $c = N/2$ (see sections 3 and 4). This follows from the MML criterion, but it is only an approximation of the MML. Furthermore, we presented an on-line procedure and the influence of the prior needs to be balanced with the influence of the data. The influence of the data is controlled by setting the constant α. Larger influence of the prior leads to more biased and more compact models.

It should be noted that the constant α also controls the speed of updating the parameters and the accuracy of the results. However, the parameters of the algorithm have a clear meaning and it is not difficult to choose them and to arrive at some compact representation of the data.

Appendix - Recursive parameter estimation

The general stochastic approximation procedure we used is given by:

$$\widehat{\vec{\theta}}^{(t+1)} = \widehat{\vec{\theta}}^{(t)} + (t+1)^{-1} I(\widehat{\vec{\theta}}^{(t)})^{-1} \vec{g}(\vec{x}^{(t+1)}, \widehat{\vec{\theta}}^{(t)})$$

with

$$\vec{g}(\vec{x}, \vec{\theta}) = \nabla_{\vec{\theta}} \log p(\vec{x}; \vec{\theta}) = \frac{\nabla_{\vec{\theta}} p(\vec{x}; \vec{\theta})}{p(\vec{x}; \vec{\theta})}$$

and where $I(\vec{\theta}) = E(\vec{g}(\vec{x}; \vec{\theta}) \vec{g}(\vec{x}; \vec{\theta})^T)$ is the Fisher information matrix corresponding to one observation. Under some regularity conditions [5] for $t \to \infty$ we have consistency and asymptotic efficiency: $(\widehat{\vec{\theta}}^{(t)} - \vec{\theta}^{*}) \to \mathcal{N}(; 0, (tI(\vec{\theta}^{*}))^{-1})$ in distribution [13]. However, it turns out that calculating and inverting the Fisher information matrix $I(\vec{\theta})$ becomes quite complicated for most of the practical cases. Therefore many alternatives were proposed.

For the finite mixtures, in the same way as when using the EM algorithm, it turns out that it is useful to introduce a discrete unobserved indicator vector $\vec{y} = [y_1 ... y_M]^T$. The indicator vector specifies the mixture component from which each particular observation is drawn. The new joint density function can be written as a product:

$$p(\vec{x}, \vec{y}; \vec{\theta}) = p(\vec{y}; \pi_1, .., \pi_M) p(\vec{x} | \vec{y}; \vec{\theta}_1, .., \vec{\theta}_M) = \Pi \pi_m^{y_m} p_m(\vec{x}; \vec{\theta}_m)^{y_m}$$

where exactly one of the y_m from $\vec{y}$ can be equal to 1 and the rest are zeros. The indicator vectors $\vec{y}$ have a multinomial distribution defined by the mixing weights $\pi_1, .., \pi_M$. The observed data $\vec{x}$ and unobserved indicators $\vec{y}$ form together the complete observation. The matrix $I(\vec{\theta})$ can be replaced by its upper bound $I_c(\widehat{\vec{\theta}}^{(t)})$ -the Fisher information matrix corresponding to one observation but now for the complete data. The efficiency is lost but under certain conditions we still get consistent estimates [5].

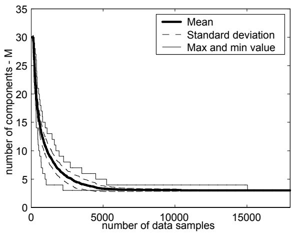
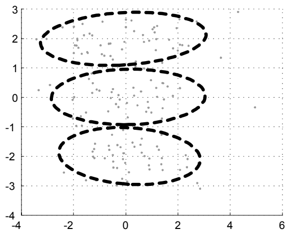
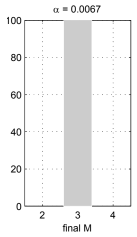
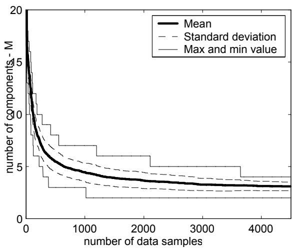
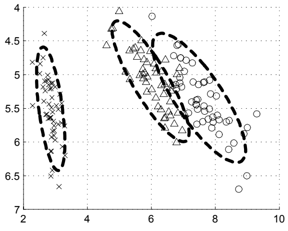
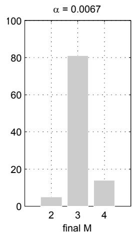
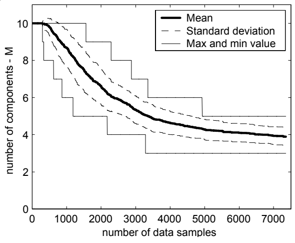
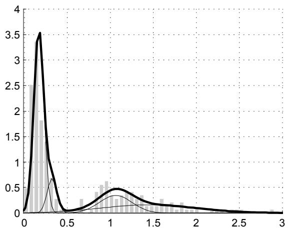
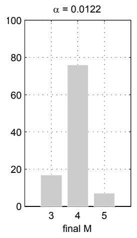
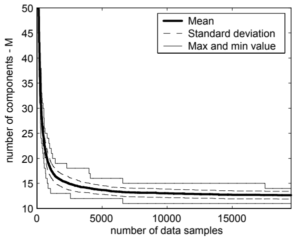
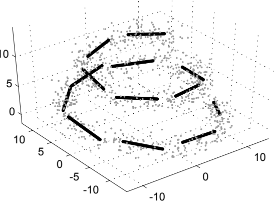
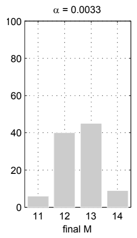
data-set	M with new data	a typical solution	histogram
Three Gaussians			
Iris			
Enzyme			
Shrinking Spiral			

Table 2.2: Results

Bibliography

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716 – 723, 1974.

[2] E. Anderson. The irises of the gaspe peninsula. *Bulletin of the American Iris Society*, 59, 1935.

[3] M.E. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation Journal*, 11(5):1155–1182, 1999.

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

[5] V. Fabian. On asymptotically efficient recursive estimation. *Annals of Statistics*, 6:854–866, 1978.

[6] M. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[7] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, 1995.

[8] G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley and Sons, 2000.

[9] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. *In, M. I. Jordan editor, Learning in Graphical Models*, pages 355–368, 1998.

[10] C. Rasmussen. The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems*, 12:554–560, 2000.

[11] S. Richardson and P. Green. On Bayesian analysis of mixture models with unknown number of components. *Journal of the Royal Statistical Society, Series B (Methodological)*, 59(4):731–792, 1997.

[12] J. Rissansen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B (Methodological)*, 49(3):223–239, 1987.

[13] J. Sacks. Asymptotic distribution of stochastic approximation procedures. *Annals of Mathematical Statistics*, 29:373–405, 1958.

[14] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[15] D.M. Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society, Series B (Methodological)*, 2(46):257–267, 1984.

[16] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley and Sons, 1985.

[17] N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11:271–282, 1998.

[18] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[19] C. Wallace and P. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society, Series B (Methodological)*, 49(3):240–265, 1987.

Chapter 3

Adaptive Recursive Unsupervised Learning

We propose a set of simple on-line equations for fitting a finite mixture model to sequential data. The algorithm can adapt to some sudden or gradual changes of the data statistics. Both the parameters of the mixture and the number of components are simultaneously adapted. A simple outlier detection rule is used to generate new components if the data statistics changes. A prior is used as a bias for maximally structured models. The prior helps in discarding the insignificant components.

3.1 Introduction

The finite mixture probability density model is a powerful tool for modeling multivariate data [15, 10]. The 'Expectation Maximization' (EM) algorithm [5] is a standard method for fitting a mixture model to the data. Many modifications of the EM algorithm have been proposed in the literature, for example [16, 17]. In some situations, as for many on-line working systems, a recursive procedure is essential to give 'quick' up-to-date parameter estimates. Approximate recursive (on-line) versions of the EM algorithm were discussed in [14, 11]. For the given data, it is also important to select a finite mixture model having an appropriate number of components to avoid 'under fitting' or 'over fitting'. Often, the appropriate number of components can reveal an underlying structure that characterizes the data. In [3] and [6] an efficient heuristic was used to select compact models for the available data. A certain type of prior is introduced that presents, in a way, a bias for more structured models. This can be used for the finite mixtures to simultaneously estimate the parameters and select the appropriate number of components. Inspired by the mentioned ideas, in the previous chapter we developed an on-line version to recursively update the parameters and select the number of components.

In this chapter we consider a general situation when the data comes from a process that is stationary for some periods but some sudden or gradual changes can occur from time to time. Automatic detection of various types of data changes was analyzed many times in the literature [3]. See also the 'novelty detection'[4] and the closely related 'outlier detection' [1]. In the case of on-line data, instead of detecting the changes we could simply constantly update the parameters and in that way we could always have a density estimate that describes the current situation. It is important to note here that also the number of components should be adapted. In [12] a simple on-line procedure was proposed to adapt to the changes. They proposed to add new components to the mixture every time there comes a new data sample not well described by the current model. The result was a non-parametric procedure with constantly increasing number of components. This is not appropriate for practical use. In this paper we add new components in a similar way, but based on the results from the previous chapter of this thesis we also detect and discard the out-of date components. The result is an on-line algorithm that constantly updates the model parameters and the number of components of the mixture to reflect the most current distribution. In addition, when the data comes from a stationary process the new algorithm can estimate the parameters and select the appropriate number of components to arrive at similar results as some previously reported algorithms but much faster. In this paper we focus on the Gaussian mixture, but the results could be extended to other mixture types.

In section 3.2 we introduce the notation and discuss the on-line parameter estimation. Further, in sections 3.3 and 3.4 we describe the additional component deletion and the component generation rule for on-line updating the number of mixture components. The final practical algorithm we used in our experiments is described in section 3.5. In section 3.6 we demonstrate how the new algorithm performs for a number of problems.

3.2 Problem definition

Lets denote the probability density function of a stochastic vector variable $\vec{x}$ as $p(\vec{x};\vec{\theta})$ where the vector $\vec{\theta}$ contains the function parameters. A mixture density with M components can be written as:

$$p(\vec{x};\vec{\theta}(M)) = \sum_{m=1}^{M} \pi_m p_m(\vec{x};\vec{\theta}_m) \text{ with } \sum_{m=1}^{M} \pi_m = 1 \tag{3.1}$$

where $\vec{\theta}(M) = \{\pi_1, .., \pi_M, \vec{\theta}_1, .., \vec{\theta}_M\}$. The mixing densities are denoted by $p_m(\vec{x};\vec{\theta}_m)$ where $\vec{\theta}_m$ are the parameters. The mixing weights denoted by π_m are positive. A recursive procedure uses a current estimate $\widehat{\vec{\theta}}^{(t)}$ for the first t data samples and the new data sample $\vec{x}^{(t+1)}$ to get the new estimate $\widehat{\vec{\theta}}^{(t+1)}$. The stochastic approximation from [13] gives:

$$\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + (1+t)^{-1}(o_m^{(t)}(\vec{x}^{(t+1)}) - \hat{\pi}_m^{(t)}) \tag{3.2}$$

with the 'ownerships':

$$o_m^{(t)}(\vec{x}) = \hat{\pi}_m^{(t)} p_m(\vec{x};\widehat{\vec{\theta}}_m^{(t)})/p(\vec{x};\widehat{\vec{\theta}}^{(t)}) \tag{3.3}$$

where the new data sample $\vec{x}^{(t+1)}$ (the $t+1$-th sample) is used to update the current estimate denoted by $\hat{\pi}_m^{(t)}$ and to get the new estimate $\hat{\pi}_m^{(t+1)}$. The most commonly used mixture is the Gaussian mixture. A mixture component $p_m(\vec{x};\vec{\theta}_m) = \mathcal{N}(\vec{x};\vec{\mu}_m, C_m)$ has the mean $\vec{\mu}_m$ and the covariance matrix C_m as the parameters. For the Gaussian mixture we get:

$$\widehat{\vec{\mu}}_m^{(t+1)} = \widehat{\vec{\mu}}_m^{(t)} + (t+1)^{-1} \frac{o_m^{(t)}(\vec{x}^{(t+1)})}{\hat{\pi}_m^{(t)}} (\vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}) \tag{3.4}$$

$$\hat{C}_m^{(t+1)} = \hat{C}_m^{(t)} + (t+1)^{-1} \frac{o_m^{(t)}(\vec{x}^{(t+1)})}{\hat{\pi}_m^{(t)}} (\vec{\delta}_m \vec{\delta}_m^T - \hat{C}_m^{(t)}) \tag{3.5}$$

where $\vec{\delta}_m = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}$. See [14] for details.

If the data statistics is changing, the simplest approach to adapt the parameter estimates is just to rely more on the newest samples by replacing $(t+1)^{-1}$ from the recursive update equations (3.4), (3.5) and (3.6) above with a small constant $\alpha = 1/T$. This also reduces the problems with instability of the equations for small t. In the next sections two additional update rules are described: the component deletion rule and the component generation rule. The component deletion rule detects the non-important components and discards them. The component generation rule should generate new components when needed. Together this two rules take care that also the number of components M is adapted to the situation. Furthermore, both the EM and the recursive versions can end up in a poor solution if not properly initialized. The additional rules make the procedure also much less sensitive to the initialization.

3.3 Deletion rule

The component deletion rule is based on the results from the previous chapter. The equation for recursive update of the mixing weights (3.2) is modified to include the influence of a prior that presents, in a way, a bias toward more compact solutions and drives the unimportant components to extinction:

$$\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + (1+t)^{-1}\left(\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{1 - Mc_T} - \hat{\pi}_m^{(t)}\right) - (1+t)^{-1}\frac{c_T}{1 - Mc_T} \tag{3.6}$$

The prior we used is the Dirichlet prior for the mixing weights [8], but with negative coefficients as suggested in [6]. The influence of the prior is controlled by the constant $c_T = c/T$. This prior is related to an approximation of the standard 'Minimum Message Length' (MML) [19] model selection criterion and we get $c = N/2$ where N presents the number of parameters per component of the mixture. As mentioned before, to rely more on the latest data we fix the influence of the new samples by replacing the $(1+t)^{-1}$ from (3.6) by a small constant $\alpha = 1/T$. Here T should be sufficiently large to make sure that $Mc_T < 1$. Weak components are discarded simply when $\hat{\pi}_m^{(t+1)} < 0$. This also ensures that the mixing weights stay positive.

3.4 Generation rule

When the data statistics changes the components should be also generated to be able to fully adapt to the changes. We propose here a "component generation rule". Our approach is based on two previous results: the 'adaptive kernel' from [12] and the recent recursive mixture learning we analyzed in the previous chapter.

As mentioned, the recursive equations described in section 2 are highly sensitive to initialization. Similar problems occur also with sudden changes in the data statistics. In order to overcome this problems but also to avoid specifying the number of components M, it was proposed in [12] to start with a single component and add new components whenever there is new data not described well by the current model. The new components are added simply by:

$$\begin{aligned}
\hat{\pi}_m^{(t+1)} &= (1-\alpha)\hat{\pi}_m^{(t)},\ \hat{\pi}_{M+1}^{(t+1)} = \alpha && (3.7)\\
\hat{\vec{\mu}}_{M+1}^{(t+1)} &= \vec{x}^{(t+1)} && (3.8)\\
\hat{C}_{M+1}^{(t+1)} &= C_0 && (3.9)\\
M &= M+1 && (3.10)
\end{aligned}$$

where C_0 is an appropriate initial covariance matrix. Note that if we add a new component for each new data sample we get a simple non-parametric kernel based approximation (the mixture components from (3.1) can then be considered as kernels). The non-parametric approaches are not sensitive to initialization and are consistent under very weak conditions. However, the non-parametric approaches would be unpractical for an on-line procedure since M is constantly growing and the model would soon be too computationally and memory intensive. If we use some reasonable rule to decide when to add a component and when to only update the mixture, we get a sort of 'adaptive kernel' approach with the number of components M still growing but at much slower rate. By choosing the decision rule to generate new components more or less often we balance between the non-parametric kernel based approach and a parametric recursive finite mixture fitting. When a new component is added the dimensionality of the likelihood surface changes and this allows the estimator to proceed towards a "good" solution and escape the local maxima. Furthermore, as noted in [17], the problems with mixture models often arise when there are unpopulated areas in the parameter space. The algorithms using a constant number of components M, usually have difficulties in moving the components to the unpopulated areas through the positions that have a lower likelihood. Here simply new components are generated for this areas.

The 'adaptive kernel' approach, analyzed in detail in [12], is using only the "component generation rule". Their approach still has a constantly growing number of components and can not be used for a practical on-line procedure. However we added here the described "component deletion rule" to select the compact models for the data and keep the number of components bounded. For a stationary process, in the previous chapter of this thesis we start with a large number of components and a random initialization. A similar algorithm using only the described delete rule was analyzed. The weakly supported components are discarded and the algorithm was able to arrive at a compact model for the data. It was demonstrated that this scheme is not very sensitive to the initialization. We could conclude that the results of the new algorithm, that is using the same deletion rule, should not be very sensitive to the way the new components are generated by the component generation rule. Therefore, we choose to add new components whenever it seems that there are changes in the data statistics. If it was a false alarm (for example just an outlier) the delete rule should take care that the unnecessary added components are removed eventually.

To decide when to add a component we need some sort of clustering criterion. There are many possibilities for this. The simplest, as in [12], is to check the Mahalanobis distance of the new sample from the current mixture components:

$$D_m(\vec{x}^{(t+1)}) = (\vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)})(\hat{C}_m^{(t)})^{-1}(\vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)})$$

If the minimum of this distances exceeds a threshold

$$D(\vec{x}^{(t+1)}) = \min_m(D_m(\vec{x}^{(t+1)})) > R_c$$

then the point is "far" from the existing components (in a way not well described by the current mixture) and a new component should be created. The threshold R_c can be used to control the generation of the components. In our experiments we used $R_c = 4$ that implies creating a new component for any observation that is at least two standard deviations away from all the existing mixture components. Another way could be to create components stochastically, for instance, with probability inversely proportional to the $D(\vec{x}^{(t+1)})$. This also has some similarities with simulated annealing.

In the previous chapter of this thesis (also in [6]), the initial covariance matrices that were used for the random initialization were a fraction (1/10 was used) of the mean global diagonal covariance matrix:

$$C_0^{(t)} = \frac{1}{10d} trace\left(\frac{1}{t}\sum_{i=1}^{t}(\vec{x}^{(i)} - \widehat{\vec{\mu}}_0)(\vec{x}^{(i)} - \widehat{\vec{\mu}}_0)^T\right) I$$

where $\widehat{\vec{\mu}}_0 = \frac{1}{t}\sum_{i=1}^{t}\vec{x}^{(i)}$ is the global mean of the data and I is the d-dimensional identity matrix. The data is d-dimensional. The recursive versions are:

$$\begin{aligned}
\widehat{\vec{\mu}}_0^{(t+1)} &= \widehat{\vec{\mu}}_0^{(t)} + (1+t)^{-1}(\vec{x}^{(t+1)} - \widehat{\vec{\mu}}_0^{(t)}) \\
\hat{C}_0^{(t+1)} &= \hat{C}_0^{(t)} + (1+t)^{-1}\left(\frac{1}{10d}(\vec{x}^{(t+1)} - \widehat{\vec{\mu}}_0^{(t)})^T(\vec{x}^{(t+1)} - \widehat{\vec{\mu}}_0^{(t)})I - \hat{C}_0^{(t)}\right)
\end{aligned}$$

We use here this covariance matrix to initialize the new components. We calculate the up-to-date covariance matrix recursively again using the small constant α instead of $(1+t)^{-1}$.

3.5 A simple practical algorithm

For the sake of clarity we present here the whole algorithm we used in our experiments. First we wait for some time (here we used 100 samples) to get a reasonable estimate for the global initialization matrix $C_0^{(t)}$. Then, with the first next sample we can start with a single component. We have $c_T = \alpha N/2$. Further, we use Gaussian mixture components with full covariance matrices. Therefore, if the data is d-dimensional, we have $N = d + d(d+1)/2$ (the number of parameters for a Gaussian with full covariance matrix). The on-line algorithm is then given by:

- **Input:** new data sample $\vec{x}^{(t+1)}$, current parameter estimates $\widehat{\vec{\theta}}^{(t)}$
- update the initialization covariance matrix:
  - $\widehat{\vec{\mu}}_0^{(t+1)} = \widehat{\vec{\mu}}_0^{(t)} + \alpha\vec{\delta}_0$ (with $\vec{\delta}_0 = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_0^{(t)}$)
  - $\hat{C}_0^{(t+1)} = \hat{C}_0^{(t)} + \alpha(\frac{1}{10d}\vec{\delta}_0^T\vec{\delta}_0 I - \hat{C}_0^{(t)})$
- calculate Mahalanobis distances $D_m(\vec{x}^{(t+1)}) = \vec{\delta}_m^T(\hat{C}_m^{(t)})^{-1}\vec{\delta}_m$ (where $\vec{\delta}_m = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}$)
- calculate 'ownerships': $o_m^{(t)}(\vec{x}^{(t+1)}) = \hat{\pi}_m^{(t)} p_m(\vec{x}^{(t+1)}; \widehat{\vec{\theta}}_m^{(t)})/p(\vec{x}^{(t+1)}; \widehat{\vec{\theta}}^{(t)})$
- update mixture weights: $\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + \alpha(\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{1-Mc_T} - \hat{\pi}_m^{(t)}) - \alpha\frac{c_T}{1-Mc_T}$
- *deletion rule:* check if there are irrelevant components: if $\hat{\pi}_m^{(t+1)} < 0$ discard the component m ($M = M - 1$) and renormalize the remaining mixing weights
- update the rest of the parameters:
  - $\widehat{\vec{\mu}}_m^{(t+1)} = \widehat{\vec{\mu}}_m^{(t)} + w\vec{\delta}_m$ (where $w = \alpha\frac{o_m^{(t)}(\vec{x}^{(t+1)})}{\hat{\pi}_m^{(t)}}$ and $\vec{\delta}_m = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}$)
  - $\hat{C}_m^{(t+1)} = \hat{C}_m^{(t)} + w(\vec{\delta}_m\vec{\delta}_m^T - \hat{C}_m^{(t)})$ (tip: limit the update speed $w = \min(20\alpha, w)$)
- *generation rule:* if $\min_m(D_m(\vec{x}^{(t+1)})) > R_c$ generate a new component $\hat{\pi}_m^{(t+1)} = (1-\alpha)\hat{\pi}_m^{(t)}$, $\hat{\pi}_{M+1}^{(t+1)} = \alpha$, $\widehat{\vec{\mu}}_{M+1}^{(t+1)} = \vec{x}^{(t+1)}$, $\hat{C}_{M+1}^{(t+1)} = \hat{C}_0^{(t+1)}$, $M = M + 1$

  Output: new parameter estimates $\widehat{\vec{\theta}}^{(t+1)}$

This simple algorithm can be implemented in only a few lines of code. The recommended upper limit 20α for w simply means that the updating speed is limited for the covariance matrices of the components representing less then 5% of the data. This is recommended since $\vec{\delta}\vec{\delta}^T$ is a singular matrix and the covariance matrix may become singular if updated too fast. Singular covariance matrix is a common problem also for the standard EM algorithm and this could lead to some difficulties. However, with this small modification we never encountered any problems. The weak components are simply discarded before they can become singular.

3.6 Experiments

The properties of the presented algorithm are demonstrated and analyzed in this section using a number of examples. Since the algorithm is intended for on-line usage, we focus on the large-sample properties and computational complexity. We start with comparing the algorithm with the two related previously reported algorithm from [12] and the algorithm from the previous chapter.

3.6.1 Only the generation rule

First, we compare the new algorithm to the related 'adaptive kernel' from [12]. Our algorithm has a similar component generation rule but we added an additional component deletion rule. In order to demonstrate the differences, we use the static 1D Gaussian mixture example from [12]: $0.5\mathcal{N}(-2, 0.5) + 0.5\mathcal{N}(0.5, 1.5)$. We present the results from 100 trials and we set $\alpha = 0.01$.

In figure 3.1a we present how the average number of components M changes with increasing number of samples. The 'adaptive kernel' is constantly adding new components and after some time the number of components is, although much lower than for a standard kernel based approach, too high to be used in a practical on-line procedure. However, in a static case, as noted in [12], most of these components are not significant. This effect is strongly present here since we use a rather large constant $\alpha = 0.01$ and the not often updated components are quickly dying off. Figure 3.1c presents a typical solution after 15000 samples. We can see that only 4 components were significant. Still, there was no principled rule to discard the non-significant components. In this paper we added the component deletion rule. The solution for the same data using the new algorithm is presented in figure 3.1d. We observe that the solution looks quite similar but correctly only 2 large components are present.

The influence of the prior from the section 3 is clearly visible. The obsolete components are suppressed and eventually discarded. In the graph 3.1a we show also the average number of components for the new algorithm. The component deletion rule takes care that the number of used components stays bounded. The number of components seems to converge to a steady state. The simple component generation rule will add new components for any new far away data sample. Even in a stationary case the far away samples will occur from time to time. This means that even with the new algorithm there will be a few insignificant nuisance components. On the other hand, these components will allow adaptation if the data statistics changes. To avoid this insignificant components when using the mixture estimate from the new algorithm, we could for example consider only the components with $\hat{\pi}_m > \alpha$. In figure 3.1b we show again the average number of the components for the new algorithm but now we show only the significant components. We observe that the average number of components converges to the correct number 2. Further through the experiments we will always report only the number of components that have $\hat{\pi}_m > \alpha$.

The two typical solutions of the two algorithms seem to be very similar (figure 3.1c and d). In figures 3.1e,f,g and h we present the average L1 and L2 errors with respect to the true distribution. Because of using a fixed constant α we can also observe that the errors seem to converge to a fixed value. However, the asymptotic convergence to zero is not relevant for an on-line adaptive procedure. The convergence rate was discussed in [12]. The new algorithm performs slightly better with much less parameters. This is because the new algorithm was able to correctly identify the two components.

3.6.2 Only the deletion rule

Another closely related method is the recursive mixture learning algorithm we presented in the previous chapter of this thesis where only the component deletion rule was used. The algorithm needs a random initialization using a large number of components. Eventually the algorithm discards the obsolete components and arrives at a compact model for the data. The new algorithm has an automatic start (using one component). The component generation rule takes care that in the beginning the data gets well covered by the initial components of the mixture model. Furthermore, the important advantage of adding the component generation rule is, of course, that the new algorithm can adapt to the data statistics changes by adapting also the number of components.

To compare the two algorithms we analyze the three-component Gaussian mixture from table 3.1. It was shown that the standard EM algorithm for this problem was sensitive to the initialization. A modified version of the EM called 'deterministic annealing EM' from [16] was able to find the correct solution using a 'bad' initialization. For a data-set with 900 samples they needed more than 200 iterations to get close to the solution. In previous chapter we start with $M = 30$ mixture components (as in [6]). With random initialization we performed 100 trials and the algorithm was always able to find the correct solution simultaneously recursively estimating the parameters of the mixture and selecting the number of components. In figure 3.2a we present how the average number of components changes with more samples. We performed the same test using the new algorithm, figure 3.2b. In the beginning a large number of components is automatically generated. The maximum average number was $M = 16$ and then the number of components decreased to the correct 3 components. Because of the smaller and better distributed (compared to the random initialization we used previously) number of components at the beginning, the new algorithm can identify the 3 components a bit faster than the previous algorithm. Furthermore, the similar batch algorithm from [6] needs about 200 iterations to identify the three components (on a 900 sample data-set). From the plot in table 3.2b we see that already after 9000 samples the new algorithm is usually able to identify the three components. The computation costs for 9000 samples are approximately the same as for only 10 iterations of the EM algorithm on a 900 sample data-set. Consequently, the new algorithm for this data set is about 20 times faster in finding the similar solution than the previously mentioned algorithms and similar to the algorithm we proposed in chapter 2. In [11] some approximate recursive versions of the EM algorithm were compared to the standard EM algorithm and it was shown that the recursive versions are usually faster. This is in correspondence with our results. Because of the discussed effects of the generation rule we also observe some occasional nuisance components occuring at the end when the 3 Gaussians are already identified. However, as we noted, this components will be useful when the data statistics changes. This is demonstrated in the experiments that follow.

Empirically we decided that 50 samples per class are enough and used $\alpha = 1/150$. In table 3.1 we present the final estimates from one of the trials. Only the three largest components are presented. The components are also presented in figure 3.2c by the '$\sigma = 2$ contours'. We also used the last 150 samples and a properly initialized EM algorithm to find the ML estimates. The results are very similar (table 3.1).

3.6.3 Adaptation

The important property of the new algorithm is that it can automatically adapt to the changes of the data statistics. This is demonstrated on a few examples. First we use the previously described three-component Gaussian mixture (see table 3.1). After 9000 samples we add one more component having the mean $\mu_4 = [\begin{array}{cc} 0 & 4 \end{array}]^T$ and the same covariance matrix as the other three components. All the mixing weights are changed at that moment to be equal to 0.25. In figure 3.3a we show how the average number of components changed for 100 trials. First, the three components are identified and the number of components remains almost constant with occasional nuisance components as discussed before. When the data statistic changes, the component generation rule adds a number of components. After some time the number of components again converges to a constant; this time 4. In figure 3.3b we show a typical solution after 9000 data samples, just before the data statistics changes. In figure 3.3c a typical final estimate after 18000 samples is presented. The algorithm has an automatic start and it can adapt to the changes in data statistics.

Another example is the 'shrinking Spiral' data set. This data-set presents a 1-dimensional manifold ('shrinking spiral') in three dimensional space with added noise:

$$\vec{x} = [\begin{array}{ccc} (13-0.5t)\cos t & (0.5t-13)\sin t & t \end{array}] + \vec{n}$$

with $t \sim Uniform[0, 4\pi]$ and the noise $\vec{n} \sim \mathcal{N}(0, I)$. After 9000 samples we exchange the sin and cos from the above equation. This gives a spiral that is spinning in the other direction. In figure 3.3e we show a typical solution after 9000 data samples, just before the data statistics changes and in figure 3.3f we show a typical final estimate after 18000 samples. For each component ($\pi_m > \alpha$) we show the eigen-vector corresponding to the largest eigen-value of the covariance matrix. The algorithm was able to automatically fit the mixture to the spiral and choose an appropriate number of components. The sudden change of the data statistic presented no problem.

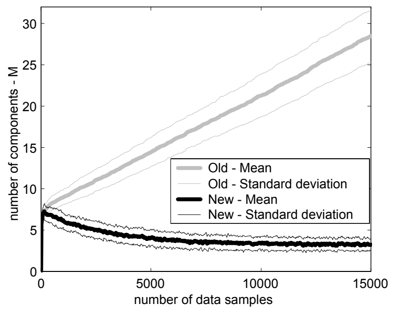
The modified EM called 'SMEM' from [17] was reported to be able to fit a 10 component mixture in about 350 iterations. The batch algorithm from [6] is fitting the mixture and usually selecting 11, 12 or 13 components using typically 300 to 400 iterations for a 900 samples data set. From the graph 3.3d it is clear that we achieve similar results but much faster. After 9000 samples we arrive at an appropriate solution but with some more components. We tested the algorithm also for a static case and we observed that about 18000 samples was enough to arrive at a similar solution as the previously mentioned algorithms (see also [ziv]). Again the new algorithm is about 20 times faster and similar to the related algorithm from chapter 2.

There are no clusters in this data-set. It can be shown that fixing the influence of the new samples by fixing α has as the effect that the influence of the old data is downweighted by a exponential decaying envelope $S(k) = \alpha(1-\alpha)^{t-k}$ (for $k < t$). For comparison with the other algorithms that used 900 samples we limited the influence of the older samples to 5% of the influence of the current sample by $\alpha = -\log(0.05)/900$. From the graph 3.3d we also observe that the number of components is much less stable than in the previous cases. This is because the Gaussian mixture is just an approximation of the true data model. Furthermore, there are no clusters to clearly define the number of components.

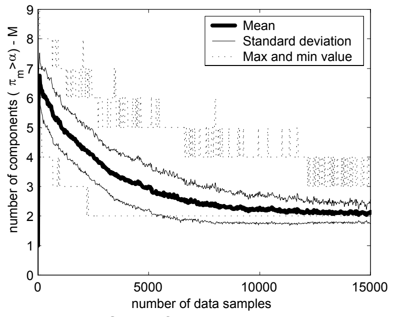
3.7 Conclusions and discussion

We proposed an adaptive recursive algorithm. The important advantage of the new algorithm is that can adapt to possible data statistics changes. Both the parameters and the number of components are adapted. As a consequence the algorithm has also an automatic start. Further, the new algorithm was able to solve difficult problems and to arrive at similar solutions as other much more elaborate algorithms. Compared to the previously proposed algorithms the new algorithm is also faster.

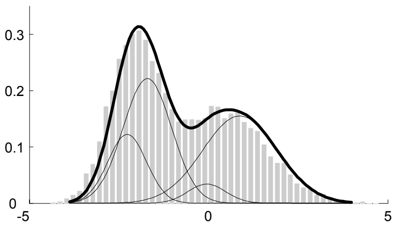
Two rules were introduced: the generation rule to create the components when needed and the deletion rule to discard the nuisance components. These two rules are used to adapt to the changes in the data statistics but also to avoid local maxima of the likelihood surface and proceed to a 'good' solution. When a component splits into two close components or similarly when a local maximum of the likelihood is achieved because two close components are modelled as one bigger component, the two rules used here are not very effective. It would be appropriate to check the goodness of fit from time to time in some way ([9], Chapter 15). The modified EM from called 'SMEM' from [17] performs split and merge steps to avoid the local maxima. Another interesting reference is the recent 'greedy' EM from [18] where adding new components to avoid these situations is discussed. Developing an on-line rule for splitting the components is a possible topic for further research.



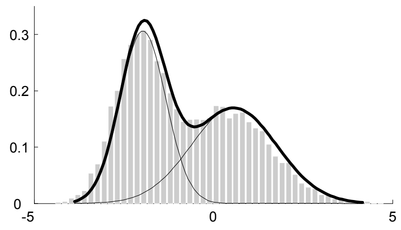
a) total number of components - Old(only generation rule)



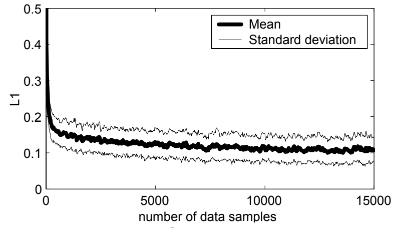
b)number of significant components - New



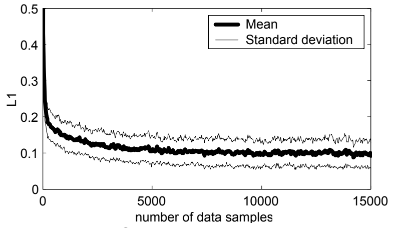
c) a typical solution - Old(only generation)



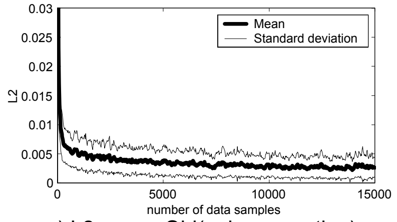
d) a typical solution - New



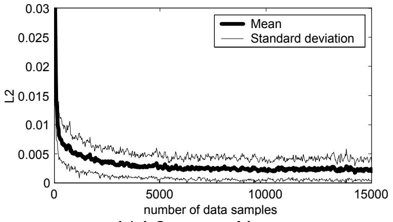
e) L1 error - Old(only generation)



f) L1 error - New



g) L2 error - Old(only generation)



h) L2 error - New

Figure 3.1: Comparison to only generation rule

Parameters $\vec{\theta}$	True values	Final estimate (3 largest components)	ML estimate (last 150 samples)
π_1	0.33	0.32	0.31
π_2	0.33	0.33	0.35
π_3	0.33	0.33	0.33
$\vec{\mu}_1$	$\begin{bmatrix} 0 & -2 \end{bmatrix}^T$	$\begin{bmatrix} 0.14 & -1.95 \end{bmatrix}^T$	$\begin{bmatrix} 0.35 & -1.91 \end{bmatrix}^T$
$\vec{\mu}_2$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$	$\begin{bmatrix} 0.00 & 0.08 \end{bmatrix}^T$	$\begin{bmatrix} -0.11 & 0.12 \end{bmatrix}^T$
$\vec{\mu}_3$	$\begin{bmatrix} 0 & 2 \end{bmatrix}^T$	$\begin{bmatrix} -0.26 & 2.02 \end{bmatrix}^T$	$\begin{bmatrix} -0.40 & 2.03 \end{bmatrix}^T$
C_1	$\begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 2.08 & -0.05 \\ -0.05 & 0.17 \end{bmatrix}$	$\begin{bmatrix} 1.66 & -0.10 \\ -0.10 & 0.20 \end{bmatrix}$
C_2	$\begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 1.99 & -0.02 \\ -0.02 & 0.20 \end{bmatrix}$	$\begin{bmatrix} 2.20 & -0.01 \\ -0.01 & 0.21 \end{bmatrix}$
C_3	$\begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 1.75 & 0.04 \\ 0.04 & 0.18 \end{bmatrix}$	$\begin{bmatrix} 1.98 & 0.12 \\ 0.12 & 0.16 \end{bmatrix}$

Table 3.1: The three Gaussians data set

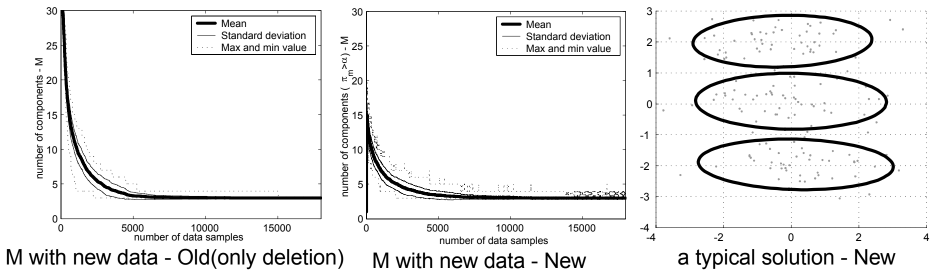


Figure 3.2: Comparison to only deletion rule

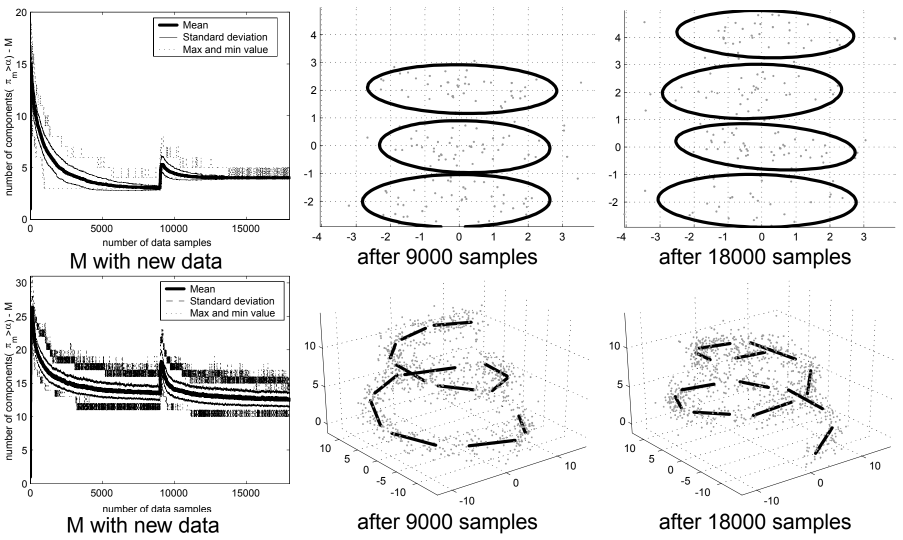


Figure 3.3: Adaptation examples

Bibliography

[1] V. Barnett and T. Lewis. *Outliers in statistical data.* Wiley, 1984.

[2] M.E. Brand. Structure learning in conditional probability models via an eutropic prior and parameter extinction. *Neural Computation Journal*, 11(5):1155–1182, 1999.

[3] I. V. Cadez and P. S. Bradley. Model based population tracking and automatic detection of distribution changes. *In, T. G. Dietterich and S. Becker and Z. Ghahramani, editors, Advances in Neural Information Processing Systems 14*, 2002.

[4] C. Campbell and K.P. Bennett. A linear programming approach to novelty detection. *In, T. K. Leen, T.G. Dietterich and V. Tresp, editors, Advances in Neural Information Processing Systems 13*, pages 389–401, 2001.

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

[6] V. Fabian. On asymptotically efficient recursive estimation. *Annals of Statistics*, 6:854–866, 1978.

[7] M. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[8] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, 1995.

[9] E. Kreyszig. *Introductory Mathematical Statistics.* John Wiley and Sons, 1970.

[10] G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley and Sons, 2000.

[11] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. *In, M. I. Jordan editor, Learning in Graphical Models*, pages 355–368, 1998.

[12] C.E. Priebe and D.J.Marchette. Adaptive mixture density estimation. *Pattern Recognition*, 26(5):771–785, 1993.

[13] J. Sacks. Asymptotic distribution of stochastic approximation procedures. *Annals of Mathematical Statistics*, 29:373–405, 1958.

[14] D.M. Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society, Series B (Methodological)*, 2(46):257–267, 1984.

[15] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley and Sons, 1985.

[16] N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11:271–282, 1998.

[17] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[18] J.J. Verbeek, N. Vlassis, and B. Krose. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 15(1), 2003.

[19] C. Wallace and P. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society, Series B (Methodological)*, 49(3):240–265, 1987.

Part II

Motion detection

Chapter 4

Adaptive background modeling

Background maintenance and subtraction is a common computer vision task. We analyze the usual pixel-level approach. First, the contributions from the literature are summarized and some basic principles and requirements are extracted. Further, based on the presented principles, some standard theory and some recent results, we develop two efficient adaptive algorithms. The first algorithm is using the parametric Gaussian mixture probability density. Recursive equations are used to constantly update the parameters and to select the appropriate number of components for each pixel. The second algorithm is using the non-parametric k nearest neighbor based density estimate. Finally, the two algorithms are analyzed and compared.

4.1 Introduction

A static camera observing a scene is a common case of a surveillance system [12, 30, 8, 23]. Detecting intruding objects is an essential step in analyzing the scene. An usually applicable assumption is that the images of the scene without the intruding objects exhibit some regular behavior that can be well described by a statistical model. If we have a statistical model of the scene, an intruding object can be detected by spotting the parts of the image that don't fit the model. This process is usually known as "background subtraction".

Usually a simple bottom-up approach is applied and the scene model has a probability density function for each pixel separately. A pixel from a new image is considered to be a background pixel if its new value is well described by its density function. For example for a static scene the simplest model could be just an image of the scene without the intruding objects. The next step would be, for example, to estimate appropriate values for the variances of the pixel intensity levels from the image since the variances can vary from pixel to pixel. This was used for example in [30]. However, pixel values often have complex distributions and more elaborate models are needed. In this paper, we consider two popular models: the parametric Gaussian mixture and the non-parametric k nearest neighbors (k-NN) estimate.

The scene could change from time to time (sudden or slow illumination changes, static objects removed etc.). The model should be constantly updated to reflect the most current situation. The major problem for the background subtraction algorithms is how to automatically and efficiently update the model. In this paper we summarize the results from the literature and extract some basic principles (section 4.2). Based on the extracted principles we propose, analyze and compare two efficient algorithms for the two models: Gaussian mixture and k-NN estimate. The Gaussian mixture density function is a popular flexible probabilistic model [14]. A Gaussian mixture was proposed for background subtraction in [7]. One of the most commonly used approaches for updating the Gaussian mixture model is presented in [22] and further elaborated in [10]. A Gaussian mixture having a fixed number of components is constantly updated using a set of heuristic equations. Based on the results from the previous chapter of this thesis and some additional approximations we propose a set of theoretically supported but still very simple equations for updating the parameters of the Gaussian mixture. The important improvement compared to the previous approaches is that at almost no additional cost also the number of components of the mixture is constantly adapted for each pixel. By choosing the number of components for each pixel in an on-line procedure, the algorithm can automatically fully adapt to the scene. Another simple probabilistic model from the literature is the kernel based estimate and it was the base for the background subtraction algorithm [5]. We propose an efficient algorithm based on the more appropriate non-parametric k-NN based model. Both the Gaussian mixture and the k-NN based algorithm are designed starting from some general principles given in section 4.2. The both algorithms have similar parameters with a clear meaning and that are easy to set. We also suggest some typical values for the parameters that work for most of the situations. Finally, we analyze and compare the two proposed algorithms.

The paper is organized as follows. First, in section 4.2 we analyze the common pixel-based background subtraction problem and extract some basic principles. In section 4.3 we develop an efficient algorithm using the Gaussian mixture density model. In section 4.4, an efficient algorithm is proposed using the non-parametric k-NN density model. Finally, in the last section, the algorithms are evaluated and compared.

4.2 Problem analysis

The simple pixel-based background subtraction can be described as follows. We denote the vector describing the value of a pixel at time t by $\vec{x}^{(t)}$. The elements of the vector are usually the RGB pixel values. Gray values or some other color space values are also possible. The probability density function modeling a pixel is $p_b(\vec{x})$. The pixel is marked as background if its new value $\vec{x}^{(t+1)}$ is well described by its density function, usually:

$$p_b(\vec{x}^{(t+1)}) > T_b$$

where T_b is a chosen threshold value.

The problem is estimating the density function (the pixel model) from the data. There are various standard ways that can be divided into the parametric and the non-parametric ones. However, there are some requirements specific for the background subtraction task.

- **time and memory efficient** - Background subtraction is in practice just the starting video processing step in a system that is usually supposed to work in real-time. Therefore, it is important to make this step both time and memory efficient. For a parametric approach, this leads to the fast recursive algorithms. Recursive algorithms are also memory efficient since only the parameters of the model are saved per pixel and updated for every new data sample.

- **adaptive** - The difficult part of the background subtraction task is the maintenance of the background model. The illumination in the scene could change gradually (daytime and weather conditions in an outdoor scene) or suddenly (switching light in an indoor scene). A new object could be brought into the scene or a present object removed from it. The background subtraction module should be able to adapt to these changes. The density estimates should rely more on the recent data. We use an exponential decaying window. At time t the influence of the older samples at time k is downweighted by $(1 - \alpha)^{t-k}$ $(k \leqslant t)$ where the constant α describes how fast the influence is decreasing. The exponential window is often used in practice because it has a simple recursive implementation.

- **automatic learning** - We don't always have some clear images without the intruding objects to learn the model. The model should be automatically constructed and updated. We assume that the intruding objects do not cover a pixel for a long time. Therefore, if we estimate the model using data from a longer period, the influence of the intruding objects should be minimal and the estimated density function should be close to that of the background. As an improvement we could decide to update the model of a pixel only for the pixel values when the pixel is detected as the background:

  $$\text{if } (p_b(\vec{x}^{(t+1)}) > T_b) \text{ update } p_b \text{ using } \vec{x}^{(t+1)}$$

  In this way we get into problems with adaptation to the possible changes of the background scene. The model could still adapt to some slow gradual changes. However for some sudden changes, for example, if a new object is brought into the scene or a present object removed from it, the background model would never be updated anymore for the corresponding pixels. The heuristic that can be derived from the standard approaches like [22] and [5] is a compromise. The values of the foreground pixels are remembered in some way, for example using an additional density function per pixel $p_{b+f}(\vec{x})$ that is always updated. If the foreground pixels exibit some stabile behavior, here simply $p_{b+f}(\vec{x}^{(t+1)}) > T_f$ this means that after some sudden change again a stabile situation occurred. Therefore we can use:

  $$\text{if } ((p_b(\vec{x}^{(t+1)}) > T_b) \text{ or } (p_{b+f}(\vec{x}^{(t+1)}) > T_f)) \text{ update } p_b \tag{4.1}$$

  This introduces an additional delay before a static object can become a part of the background. The delay is controlled by the additional threshold T_f. The objects that don't stay static long enough will not be included in the background model. Some specific solutions are presented later.

The described simple pixel-based background modeling is analyzed here. The main problem is estimating the density function for the pixels according to the described requirements. As said before, from the many possible parametric models we use the standard flexible parametric Gaussian mixture model. From the possible non-parametric models we use the k-NN based approach that is appropriate for our purpose because of its simplicity.

4.3 Gaussian mixture background model

Here, we analyze the Gaussian mixture model and its application for the background subtraction.

4.3.1 Model

A Gaussian mixture density with M components can be written as:

$$p(\vec{x};\vec{\theta}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\vec{x};\vec{\mu}_m, C_m), \text{ with } \sum_{m=1}^{M} \pi_m = 1 \tag{4.2}$$

and $\vec{\theta} = \{\pi_1, ..., \pi_M, \vec{\mu}_1, ..., \vec{\mu}_M, C_1, ..., C_M\}$. Where $\vec{\mu}_1, ..., \vec{\mu}_M$ are the means and $C_1, ..., C_M$ are the covariance matrices describing the Gaussian distributions. The mixing weights denoted by π_m are positive. Parameter estimates at time t will be denoted as $\widehat{\vec{\theta}}^{(t)}$. Adaptive recursive estimation of both the parameters and the number of components is discussed in chapter 2. The parameters are updated recursively according to the stochastic approximation procedure from [21]. Two rules, component generation and component deletion, are added to adapt also the number of components and choose compact models for the data. The generation rule is inspired by the 'adaptive kernel' approach from [18]. The deletion rule is inspired by the recent results from [3, 6].

4.3.2 Modifications for background subtraction task

We develop a practical background subtraction algorithm using the recursive update equations from chapter 2.

Time and memory efficient

Although the update equations from presented in chapter 2 are very simple, some additional approximations can significantly speed up the algorithm. The recursive equations for updating the Gaussian mixture parameters are presented in [24]. In the previous chapter of this thesis the equation for recursive update of the mixing weights is modified to include the influence of a prior that presents, in a way, a bias toward more compact solutions and drives the unimportant components to extinction. A simplified version since we don't usually have many component (M is small), is given as:

$$\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + \alpha(o_m^{(t)}(\vec{x}^{(t+1)}) - \hat{\pi}_m^{(t)}) - \alpha c_T$$

with the ownerships:

$$o_m^{(t)}(\vec{x}^{(t+1)}) = \hat{\pi}_m^{(t)} \mathcal{N}(\vec{x}; \hat{\vec{\mu}}_m^{(t)}, \hat{C}_m^{(t)}) / p(\vec{x}^{(t+1)}; \hat{\vec{\theta}}^{(t)}) \tag{4.3}$$

The constant c_T describes the influence of a prior [6]. According to the MML criterion [27] $c_T = \alpha N/2$ where N is the number of parameters per component of the mixture. The constant α controls the previously discussed exponential envelope that is used to limit the influence of the old data. Further, we are going to assume that the Gaussians from the mixture are not close to each other (if there are two close components they can be modelled as a single larger component). Under this assumption, if a new data sample $\vec{x}^{(t+1)}$ is 'close' to the m-th component, the ownership $o_m^{(t)}(\vec{x}^{(t+1)})$ for the new sample will be almost 1 and almost 0 for the other components. Therefore, instead of expensive calculation of the ownerships we do the following. We can define that a sample is 'close' to a component if the Mahalanobis distance from the component is for example less than three standard deviations. Then we set the $o_m^{(t)}(\vec{x}^{(t+1)}) = 1$ for the largest 'close' component and 0 for the others. Furthermore, also for computational reasons we assume the covariance matrix in the form of $C_m = \sigma_m^2 I$. The distance of a new sample from the m-th component is calculated as: $D_m^2(\vec{x}^{(t+1)}) = \frac{1}{\sigma_m^2}(\vec{x}^{(t+1)} - \hat{\vec{\mu}}_m^{(t)})^T(\vec{x}^{(t+1)} - \hat{\vec{\mu}}_m^{(t)})$.

Finally, instead of calculating the whole density function for the background subtraction as an approximation we use the calculated Mahalanobis distances from the components. This allows an efficient implementation since it turns out that in this way we never actually calculate the density function. An additional advantage is that it is easy to choose a resonable threshold value. We use $T_b^{GM} = 4$ that corresponds to four standard deviations distance from the components. In total, this will lead to a sort of 'on-line k-means' clustering algorithm similar to the heuristic from [22] with the important difference that also the number of components is selected on-line.

Adaptive

As discussed in chapter 2 the two mentioned rules, component generation and deletion, update the number of components and choose compact models for the data. The rules also help in automatically adapting to some sudden changes in data statistics and the algorithm is not very sensitive to the initial conditions. The generation rule simply adds a new component whenever there is a new sample that is not well described by the current distribution. We define 'close' to a component, as mentioned above, by being within three standard deviations from the component's mean. If a sample is not 'close' to any of the components a new component is generated:

$$\min D_m(\vec{x}^{(t+1)}) \geqslant T_g$$

We use $T_g = 3$, as mentioned. When a new component is generated it is initialized by: $\hat{\pi}_{M+1}^{(t+1)} = \alpha$, $\hat{\vec{\mu}}_{M+1}^{(t+1)} = \vec{x}^{(t+1)}$, $\hat{\sigma}_{M+1}^2 = \sigma_0^2$. We need some reasonable value for σ_0^2. We use a fraction (0.2 is used here, see also previous chapter of this thesis) of the mean variation of the pixel values. We use a simple robust estimate by calculating the median med of the differences $\left\|\vec{x}^{(t)} - \vec{x}^{(t-1)}\right\|$ for all the pixels between two successive frames and for a number of pairs of images:

$$\sigma_0 = 0.2\frac{med}{0.68\sqrt{2}} \tag{4.4}$$

The deletion rule is added to discard the obsolete components. A component is deleted if its weight becomes negative $\hat{\pi}_m^{(t+1)} < 0$. This also ensures that the weights stay positive. Adapting the number of components is influenced by the constant c_T. If the data is d-dimensional and we use the simplified covariance matrix, the total number of parameters is $d+1$ is and we have $c_T = \alpha N/2 = \alpha(d+1)/2$. Using a larger c_T means increasing the influence of the prior and stronger preference toward compact solutions. In chapter 3 we used $\alpha = 0.01$ for a data set with two components which is close to a typical situation for the background data. Here α is usually smaller since we want to consider a longer time period. In order to keep the low number of components we fix the influence of the prior by using $c_T = 0.01(d+1)/2$.

Automatic learning

It would be impractical to estimate one more distribution to be able to directly implement the logic we discussed in section 4.2. We observe that the algorithm performs on-line clustering. We can assume that the intruding objects will be represented by some additional clusters with small weights π_m. As in [22], instead of using two distributions per pixel we can simply select the first B largest clusters to be the background model:

$$p_b(\vec{x}) \sim \sum_{m=1}^{B} \pi_m \mathcal{N}(\vec{x}; \vec{\mu}_m, C_m)$$

and all the components make the $p_{b+f}(\vec{x})$. If the components are sorted to have descending weights we have:

$$B = \arg\min_b \left(\sum_{m=1}^{b} \pi_m > (1 - T_f^{GM}) \right)$$

where T_f^{GM} is a measure of the maximum portion of the data that can belong to the foreground objects without influencing the background model. For example, if a new object comes into a scene and remains static for some time it will probably generate an additional stabile cluster. Since the old background is occluded the weight π_{B+1} of the new cluster will be constantly increasing. If the object remains static long enough, its weight becomes larger than T_f^{GM} and it can be considered to be part of the background. The object should be static for approximately (we don't consider the c_T) $\log(1 - T_f^{GM}) / \log(1 - \alpha)$ frames. For example for $T_f^{GM} = 0.1$ and $\alpha = 0.001$ we get 105 frames.

4.3.3 Practical algorithm

The whole practical algorithm can be summarized as:

Input: new data sample $\vec{x}^{(t+1)}$, current parameter estimates $\widehat{\vec{\theta}}^{(t)}$ - components sorted to have decreasing weights π_m, initially $M = 0$.

- background subtraction:
  - find $B = \arg\min_b \left(\sum_{m=1}^{b} \pi_m > (1 - T_f^{GM}) \right)$(we use $T_f^{GM} = 0.1$)
  - if $D_m < T_b^{GM}$ and $m \leqslant B$ it is a background pixel. Here $D_m^2 = \vec{\delta}_m^T \vec{\delta}_m / (\widehat{\sigma}_m^{(t)})^2$(with $\vec{\delta}_m = \vec{x}^{(t+1)} - \widehat{\vec{\mu}}_m^{(t)}$) is the Mahalanobis distance of the new sample from the m-th mode (we use $T_b^{GM} = 4$)
- background update:
  - for the first (the largest) 'close' mode ($D_m < T_g$) update the parameters (we use $T_g = 3$):
    * $\hat{\pi}_m^{(t+1)} = \hat{\pi}_m^{(t)} + \alpha(1 - \hat{\pi}_m^{(t)}) - \alpha c_T$ (where $c_T = 0.01(3+1)$ for 3-dimensional data)
    * $\widehat{\vec{\mu}}_m^{(t+1)} = \widehat{\vec{\mu}}_m^{(t)} + (\alpha / \hat{\pi}_m^{(t)}) \vec{\delta}_m$
    * $(\widehat{\sigma}_m^{(t+1)})^2 = (\widehat{\sigma}_m^{(t)})^2 + (\alpha / \hat{\pi}_m^{(t)})(\vec{\delta}_m^T \vec{\delta}_m - (\widehat{\sigma}_m^{(t)})^2)$
    * sort the mode so that we again have decreasing weights π_m
  - for all other modes:

- * $\hat{\pi}_m^{(t+1)} = (1-\alpha)\hat{\pi}_m^{(t)} - \alpha c_T$
  * *deletion rule:* if $\hat{\pi}_m^{(t+1)} < 0$ discard the component m ($M = M-1$)
- *generation rule:* if not 'close' to any of the components ($\min(D_m) \geqslant T_g$) generate a new component $\hat{\pi}_{M+1}^{(t+1)} = \alpha$, $\widehat{\vec{\mu}}_{M+1}^{(t+1)} = \vec{x}^{(t+1)}$, $(\widehat{\sigma}_{M+1}^{(t+1)})^2 = \sigma_0^2$, $M = M+1$

Output: new parameter estimates $\widehat{\vec{\theta}}^{(t+1)}$

Two subparts of the algorithm are: the background update and the background subtraction. Additionally from time to time we need to update the initialization variance σ_0^2, equation (4.4) (a histogram can be used to find the median). We also need to specify the maximum number of components M and reserve the memory for the $\hat{\pi}_m$-s, $\widehat{\vec{\mu}}_m$-s and $\widehat{\sigma}_m$-s. In practice a maximum of 4 components is enough.

4.4 Non-parametric k-NN background model

Here, we analyze the nonparametric density estimation techniques for the background subtraction.

4.4.1 Model

The non-parametric methods estimate density function without making assumptions about the form of the density function. A popular non-parametric method is the kernel based estimation [20, 16]. The kernel estimate is constructed by centering a kernel $\mathcal{K}$ at each observation. To suppress the influence of the old data we again use the exponentially decaying envelope and for practical reasons we disregard the old samples having little influence:

$$p(\vec{x}) \sim \sum_{n=t-K}^{t} \pi_n \mathcal{K}(\vec{x}; \vec{x}^{(n)}, D), \text{ with } \pi_n = (1-\alpha)^{t-n} \tag{4.5}$$

If we decide to throw away the samples that have less than 10% influence with respect to the influence of the newest sample $\vec{x}^{(t)}$ we get:

$$K = [\log(0.1)/\log(1-\alpha)]$$

where [.] is the 'round-to-integer' operator. The model is presented by the set of K samples from the recent history. The influence of a sample depends on its weight π_n and the space where the influence of a sample is present is controlled by D. Too big D leads to an oversmoothed solution where the close structures are merged together. Too small D leads to a very noisy and spiky estimate of the density function. The parameter D is called the smoothing factor or the bandwidth. While in practice the kernel form $\mathcal{K}$ has little influence, the choice of D is critical [1, 28]. However, it is easy to avoid this problems by using another standard non-parametric approach: the k nearest neighbors (k-NN) density estimate. The logic of the k-NN estimate is in a way opposite to the logic of the kernel estimates. The k-NN estimate is obtained by increasing the D until certain amount of data is covered by the kernel. In this way we get larger kernels in the areas with small number of samples and smaller kernels in the densely populated areas which is a desirable property. Usually the uniform kernel is used. The needed size of the kernel defined by its diameter D is used as the density estimate:

$$p(\vec{x}) \sim 1/D^d$$

where d is the dimensionality of the data. Using the k-NN estimates as density estimates leads to some practical problems, but using the k-NN estimates for the classification purposes, as needed here, is straightforward and often used [2]. The parameter we need to set is the number k of the nearest samples. One nearest neighbor is often used. To be more robust to some outliers we use $k = [0.1K]$.

4.4.2 Modifications for background subtraction task

We propose a practical background subtraction algorithm here based on the non-parametric k-NN density estimate.

Time and memory efficient

The number of samples K in the model will be usually much larger than the number of clusters when the Gaussian mixture is used. For example for a moderate $\alpha = 0.001$ we need $K = 2301$ samples which is far too much since it means keeping all this samples for each image pixel. However, if we don't include samples from every incoming frame we could still have samples from a wide time span with a reasonable total number of samples. Following this idea we also note that instead of changing the weights π_n it is possible to approximate the exponential envelope by appropriately changing the frequency of adding new samples to the model. In figure 4.1 we present an approximation with three steps for $\alpha = 0.001$. The samples from the past are divided into

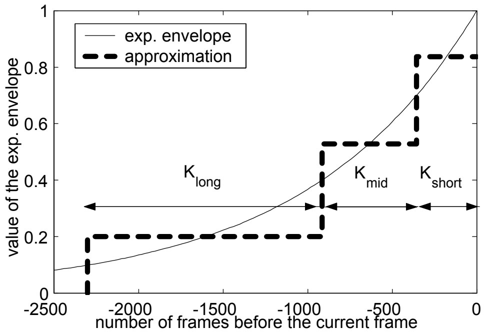


Figure 4.1: Approximation of the exponential envelope

three groups. Each group stands for 30% of the total area under the envelope (the last 10% - the oldest samples - are not considered as mentioned before). The total number of frames per group can be calculated as:

$$\begin{aligned}
K_{short} &= [\log(0.7)/\log(1-\alpha)] \\
K_{mid} &= [\log(0.4)/\log(1-\alpha) - K_{short}] \\
K_{long} &= [\log(0.1)/\log(1-\alpha) - K_{mid} - K_{short}]
\end{aligned}$$

If we decide to use for example $N = 10$ samples per group we will have a total of $K = 3N = 30$ samples. A new sample is added to the short-term group after every K_{short}/N frames. The mid-term group gets the oldest one from the previous short-term group every K_{mid}/N frames. The long-term group gets a new sample (the last one from the previous mid-term group) every K_{long}/N frames. In this way we get the recent history most densely sampled but we still keep some of the old samples in the mid and long-term groups.

The background subtraction is now implemented in a simple way. For a specified threshold T_b^{NP} , the new sample $\vec{x}^{(t)}$ belongs to the background:

$$\left(\sum_{n=1}^{K} D_n > T_b^{NP}\right) \geqslant k$$

where squared distance of the new sample $\vec{x}^{(t+1)}$ from the model sample $\vec{x}^{(n)}$ is calculated by $D_n^2 = (\vec{x}^{(n)} - \vec{x}^{(t+1)})^T(\vec{x}^{(n)} - \vec{x}^{(t+1)})$. The notation we used assumes that the result of the operation $D_n > T_b^{NP}$ is equal to 1 if it is true and 0 if it is false. It is important to note that as soon as the sum is equal to k we can stop calculating the sum. Usually many pixels belong to the background and in this way we don't need to always go through all the samples.

Adaptive

Again, the influence of the old data is controlled by the constant α. The approximation with three groups of samples is described previously. The parametric approaches are usually sensitive to the initial conditions. On the other hand the non-parametric approaches don't have these problems.

Automatic learning

The heuristic described in section 4.2 used to suppress the influence of the intruding objects on the background model can be implemented in the following way. Besides the samples $\vec{x}^{(1)}, ..., \vec{x}^{(K)}$ from the model, we keep also a set of corresponding indicators $b^{(1)}, ..., b^{(K)}$. The indicator $b^{(n)}$ indicates if the saved sample $\vec{x}^{(n)}$ belongs to the background model. All the samples give the p_{b+f} estimate and only the samples with $b^{(k)} = 1$ give the background model:

$$\left(\sum_{n=1}^{K} b^{(k)} (D_n > T_b^{NP}) \right) \geqslant k$$

When we add a new sample $\vec{x}^{(t+1)}$ to the model the corresponding indicator is set to:

$$\begin{aligned} \text{if } \left(\sum_{n=1}^{K} b^{(k)} (D_n > T_b^{NP}) \right) &\geqslant k \text{ or } \left(\sum_{n=1}^{K} D_n > T_b^{NP} \right) \geqslant T_f^{NP} \\ \text{then } b &= 1, \text{ otherwise } b = 0 \end{aligned}$$

If a new object remains static it will take some time before it can become the part of the background. Since the background is occluded, the number of samples in the model corresponding to the object will be increasing. We could expect the new samples to be close to each other. If the neighborhood defined by the T_b^{NP} is large enough we could use $T_f^{NP} = [0.1K]$ as a corresponding value to the value $T_f^{GM} = 0.1$ discussed for the Gaussian mixture. In a similar way, approximately after $\log(1 - T_f^{NP}/K)/\log(1 - \alpha)$ frames, the threshold T_f^{NP} will be exceeded and the new samples start to be included into the background model (the b-s are 1). In the Gaussian mixture case the new object is usually presented by an additional cluster that is immediately included into the background model. However, since the data is not clustered for the non-parametric method it will take additional time $\log(1-k/K)/\log(1-\alpha)$ before the pixels corresponding to the object start to be recognized as the background.

For $\alpha = 0.001$, $T_f^{NP} = [0.1K]$ and $k = [0.1K]$ it will take approximately $2 \cdot 105 = 210$ frames.

4.4.3 Practical algorithm

The whole practical algorithm is given by:

Input: new data sample $\vec{x}^{(t+1)}$, the samples from the model $\vec{x}^{(1)}, ..., \vec{x}^{(K)}$, corresponding indicators $b^{(1)}, ..., b^{(K)}$, for initialization we put values from the first frames into the model and the b-s to 1.

- background subtraction:
  - $k_b = 0, k_{b+f} = 0$
  - for $n = 1$ to $K(= 3N)$ (go through all the samples)
    - $k_{b+f} = k_{b+f} + (D_n > T_b^{NP})$, with $D_n^2 = (\vec{x}^{(n)} - \vec{x}^{(t)})^T(\vec{x}^{(n)} - \vec{x}^{(t)})$ (we suggest $T_b^{NP} = 4\sigma_0$)
    - if $(b^{(k)})$ $k_b = k_b + (D_n > T_b^{NP})$
    - if $k_b \geqslant k$ it is a background pixel (we can stop further calculation) (we use $k = [0.1K]$)
- background update:
  - after every K_{long}/N frames remove the oldest sample from the long-term group and add the oldest sample from the mid-term group
  - after every K_{mid}/N frames remove the oldest sample from the mid-term group and add the oldest sample from the short-term group
  - after every K_{short}/N frames remove the oldest sample from the short-term group and add the new sample $\vec{x}^{(t+1)}$. The corresponding indicator b is set to 1 if $((k_b \geqslant k)$ or $(k_{b+f} \geqslant T_f^{NP}))$, otherwise $b = 0$ (we use $T_f^{NP} = [0.1K]$)

Output: updated set of samples $\vec{x}^{(1)}, ..., \vec{x}^{(K)}$ and the corresponding indicators $b^{(1)}, ..., b^{(K)}$.

We suggest the value $T_b^{NP} = 4\sigma_0$ as a reasonable threshold value. Here σ_0 is the mean variation of the pixel values mentioned previously, equation (4.4). Initially we need to select the number of samples per group N and reserve the memory for the $K = 3N$ samples. For the Gaussian mixture we needed to set the upper limit for the number of components but it had no important influence on the results. However, the number of samples K is an important parameter here. More samples lead to more accurate results. However this also leads to slower performance and a large amount of memory is needed for the model.

4.5 Evaluation and comparison

Evaluating and comparing computer vision algorithms is a difficult task. We analyze the evaluation problem and present some experiments to evaluate and compare the two proposed algorithms.

4.5.1 Fitness/cost measures

Usually it is not possible to capture the algorithm's effectiveness in a single measure (except for some very specific applications). Background subtraction can be regarded as an image segmentation algorithm. Many different fitness measures are used [26, 31]. The three important fitness/cost measures for our problem are:

- true positives - percentage of the pixels that belong to the intruding objects that are correctly assigned to the foreground.
- false positives - percentage of the background pixels that are incorrectly classified as the foreground.
- processing time - the reported processing time is measured on a Pentium 4 at 2GHz and normalized to the time needed for an 320×240 image.

4.5.2 Performance analysis

To illustrate and later also analyze the performance of the algorithms we used a difficult scene. There was a monitor with rolling interference bars in the scene. The plant from the scene was swaying because of the wind. The image sequence we used contains the typical situations an adaptive background subtraction algorithm should deal with [25]. In figure 4.2 we show some interesting frames from the sequence and the results of the background/foreground segmentation for both algorithms. We used the typical parameter values we suggested before. To get comparable results we used $K = 21$ for the non-parametric model ($N = 7$ samples for each of the three groups described in section 4.4). First, in figure 4.2a we show that the object (the hand) remaining still for some short time does not influence the background model. The hand was not moving for about 70 frames and at frame 405 it was still not included into the background. We used $\alpha = 0.001$ (see the discussion in section 4.4). In figure 4.2b we show the situation at frame 440 just after the cup was removed from the scene. The new background from behind the cup is classified as foreground in the beginning. However, after enough time has passed, at frame 640, this problem is solved as shown in figure 4.2c. In figure 4.2d we show the last frame of the sequence that is used for evaluation later. We can conclude that both algorithms can deal with common background subtraction problems in a proper way.

In order to better understand the performance of the algorithms we also show the estimated background models for some characteristic pixels at some characteristic frames in figure 4.3. For the Gaussian mixture, we plot a sphere for each component of the mixture that is part of the background model (first B components, see section 4.3). The points within the presented spheres are classified as background for the suggested threshold $T_b^{GM} = 4$. For the non-parametric model, we present the corresponding $T_b^{NP} = 4\sigma_0$ spheres around the samples that are part of the model (the indicators $b = 1$, section 4.4). Since $k = [0.1 \cdot K] = 2$, a data sample is classified as background if it is included in at least two of the presented spheres (two nearest neighbors). First, in figure 4.3a we show both models at frame 405 for a pixel from the area of the monitor that was occluded by the hand. Because of the interference bars on the monitor we notice a clear bi-modal form well captured by both models. We observe that the new pixels values from the hand (the cluster of points left-below in the graphs) had not influenced the background model. The non-parametric model had included some samples from this group but they were marked with b-s equal to 0. The Gaussian mixture had an additional Gaussian to model these points with $\pi_3 = 0.069$ at this point of time which was not enough to be a part of the background (B was 2). Further, in figure 4.3b we show the models for a pixel from the area of the removed cup at frame 640. The new background is now a part of both models (the upper group of pixel values in the graph) and there is a part (the lower cluster) of the both models describing the old pixel values while the cup was in the scene. Finally, in figure 4.3c we show the models for a pixel from the area of the waving plant. The GM model again used two Gaussians to model this complex distribution. We notice that the GM with the simple covariance matrix is sometimes a crude approximation of the true distribution.

We used the last frame from the sequence, figure 4.2d, to further analyze the performance of the algorithms. The frame is manually segmented to generate the ground truth. The result of the segmentation for both algorithms is compared to the ground truth by counting the true positives and the false

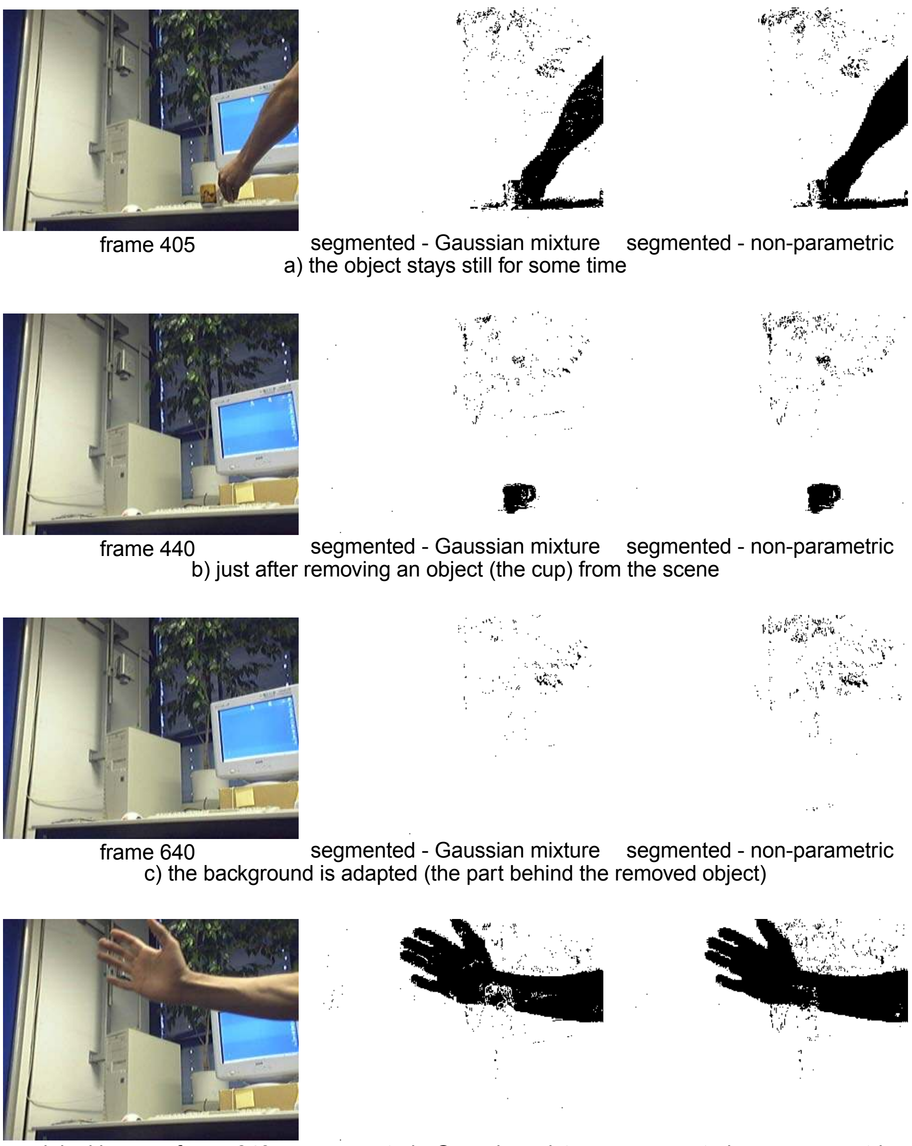
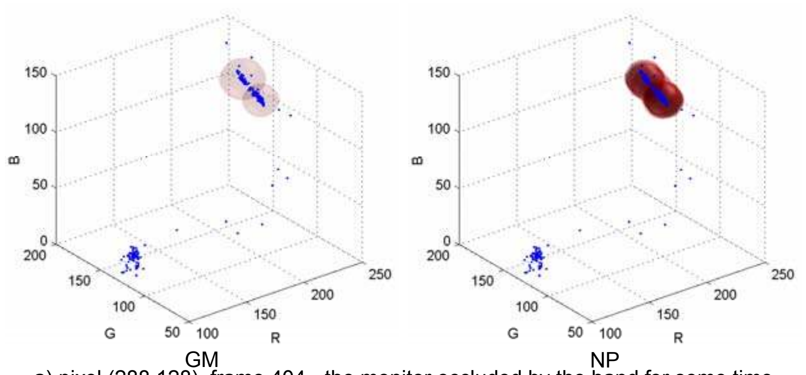
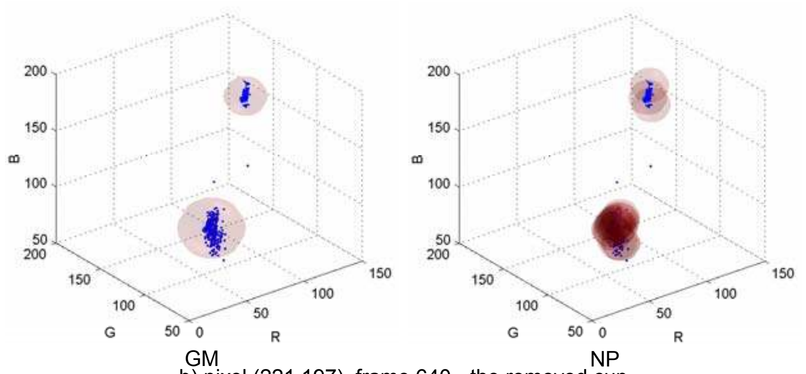

frame 405
segmented - Gaussian mixture
segmented - non-parametric
a) the object stays still for some time
frame 440
segmented - Gaussian mixture
segmented - non-parametric
b) just after removing an object (the cup) from the scene
frame 640
segmented - Gaussian mixture
segmented - non-parametric
c) the background is adapted (the part behind the removed object)
original image - frame 840
segmented - Gaussian mixture
segmented - non-parametric
d) object in the scene without shadow - used for evaluation

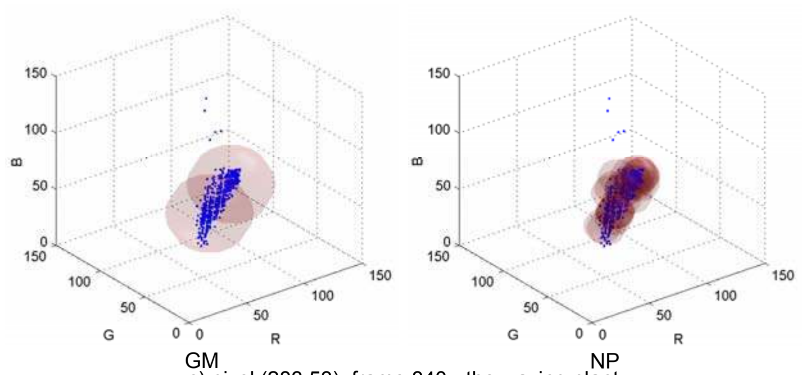
Figure 4.2: Typical situations



a) pixel (288,128), frame 404 - the monitor occluded by the hand for some time



b) pixel (221,197), frame 640 - the removed cup



c) pixel (283,53), frame 840 - the waving plant

Figure 4.3: Example models

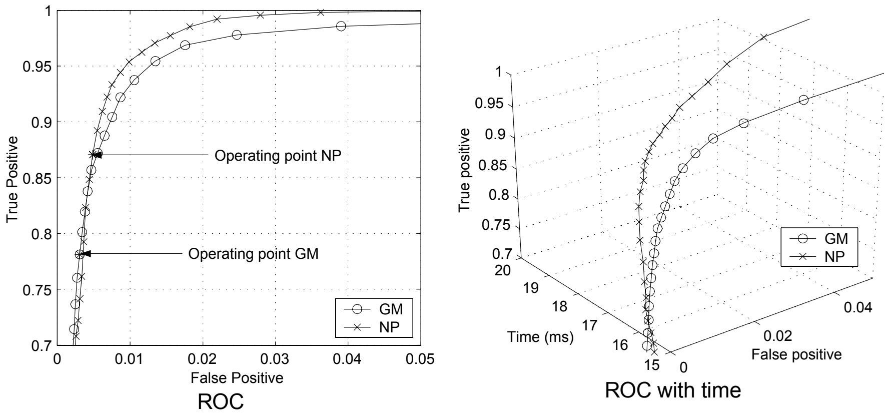


Figure 4.4: ROC curve

positives. This is done for different values of the thresholds T_b^{GM} and T_b^{NP}. The results are plotted as receiver operating characteristic (ROC) curves in figure 4.4. The ROC curve is a standard way to present and analyze the discriminative power of an algorithm [4]. We see that both algorithms are performing similarly. However, since as we noted (see for example figure 4.3c) the NP model is more flexible and therefore often much closer to the true distribution, for small thresholds (upper part of the curves) the NP method is performing better. The common problem of the k-NN approach is that the model tends to be too specific if it is based on a small number of samples as here. Therefore we observe somewhat better suppression of the noise (better generalization) for the simpler GM model for larger threshold values (GM curve is more to the left). In the paper we suggested some reasonable values for the thresholds T_b^{GM} and T_b^{NP}. The operating points selected in this way, as indicated on the ROC curves (figure 4.4), present a reasonable initial choice. However, the optimal choice for the thresholds will depend on the application and the needs of the further processing.

Since the processing time is important we plot the ROC curves with the corresponding measured time, figure 4.4. We observe that the processing time of the NP model depends on the threshold value. For a background pixel we stop calculations as soon as we find its k nearest neighbors. However, for smaller threshold values (upper part of the curve) we need to check more of the samples from the model searching for the k nearest neighbors. Similarly, the increase of the processing time is also present when there is a large foreground object in the scene. In figure 4.5 we present the measured processing time for the whole sequence. We observe that also the GM model needs more time when a foreground object is present (in the middle of the sequence and at the end) since new components are generated to model the foreground. We observe a larger variation of the processing time for the NP model. These variations for the NP model are even larger if we use more samples as a model.

4.5.3 Parameters and comparison

The algorithms usually have a number of parameters which need to be set by the user. For different parameter values the algorithm performs differently. The final background threshold T_b (here T_b^{GM} and T_b^{NP}) is an example of a parameter that has a great influence on the results. Plotting the false positives and the true positives for different thresholds as a ROC curve is a standard way to present the discriminative power of an algorithm free of the threshold value. A generalization is the Pareto front [15] that was used recently in [13] for evaluating and comparing the segmentation algorithms. The Pareto front for an algorithm is a surface in the fitness/costs space that corresponds to the best possible results for all possible parameter values. The fitness/cost measures we use are: the true positives, the true negatives and the processing time, as presented at the beginning of this chapter. Extensive search of the parameter space should be done. For the presented algorithms we construct a simple sketch of the Pereto front. We suggested the values for most of the parameters of both algorithms. Except the final threshold T_b, a free parameter for both algorithms is the parameter α that controls how fast the algorithms can adapt to new situations. To simplify the procedure, and since the meaning of the parameter α is clear and it has the same influence on the performance of both algorithms, we use fixed typical values $\alpha = 0.001$ for slow changing scenes and $\alpha = 0.005$ for more dynamic ones. The frames near the end of the sequence are used for analysis to avoid the influence of the transitional processes.

Since there are no free parameters that directly influence the processing time for the GM model, the Pareto front for our sequence looks like it is presented in figure 4.6a. On the other hand, for the NP model we have to choose the number of samples N per group. The number of samples has a direct influence on the processing time and on the accuracy. Therefore, for the NP model we measure the true positives, the false-positives and the processing time for different threshold values T_b^{NP} and for different N (from 1 till 15). The results are summarized in figure 4.6b. Finally, to compare the algorithms we plot the both Pareto fronts on a same graph in figure 4.6c. The two surfaces are close to each other, but the NP method mostly outperforms the GM method.

In figure 4.7 we show the results on two other sequences. Since the object in the sequences are much more dynamic we used $\alpha = 0.005$. The results illustrate that the performance is highly dependent on the situation. First, a standard traffic scene is used. This was a rather simple background scene without any dynamic elements. However, it was hard to distinguish the car left-down from the road because of the similar colors. This led to worse results than for the previous much more difficult scene (the monitor and the waving plant). For this scene the GM performs a bit better than the NP method as illustrated in 4.7a. The rather crude GM approximation seems to be good enough for this situation. Usually, a single Gaussian is used to model each pixel. This also leads to less processing time. The second sequence is the very difficult waving trees sequence used in [5]. The results are much worse than for the previous sequences, but both algorithms can deal with the dynamic nature of the scene and it is possible to track the person walking behind the trees. However, because of the complex pixel value distributions we observe that the NP is now performing much better, figure 4.7b. Because of the dynamic nature of the scene the processing time is also much higher. The GM needed 19ms per frame on average. The performance of the algorithms depends on the situation. In general for more challenging scenes where the pixels have complex distributions the NP approach seems to be a better choice.

4.6 Related work

We analyzed the common simple pixel-based background subtraction. There are many aspects of the background subtraction task that are not considered here. We list some possible improvements:

- spatial correlation - A pixel value is not independent of the values from its neighboring pixels. Taking this into account could lead to big improvements. Usually this fact is used in some way in some postprocessing steps (disregarding the isolated pixels or some other morphological operations on the segmented images etc.). Some examples could be found in [5].
- temporal correlation - A pixel value is not independent of the values the pixel had previously. An approach taking this into account is presented in [25]. There, the pixel value distribution over time is modelled as an autoregressive process. In [11] the Hidden Markov Models [19] are used.
- higher level knowledge - The background subtraction module is just a part of a larger system. The results from the background subtraction

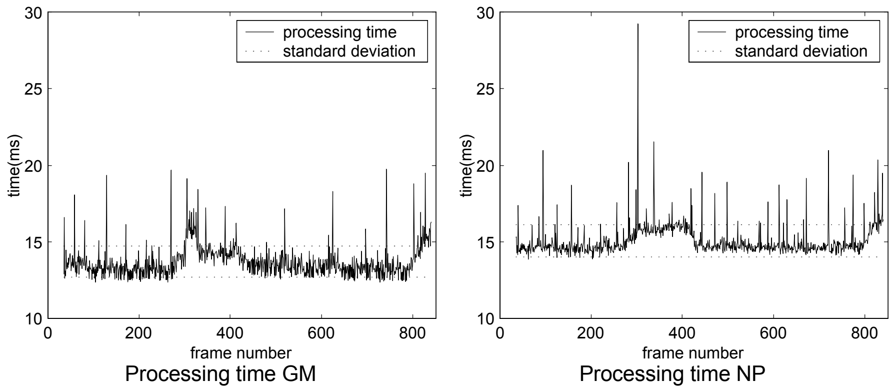


Figure 4.5: Time analysis

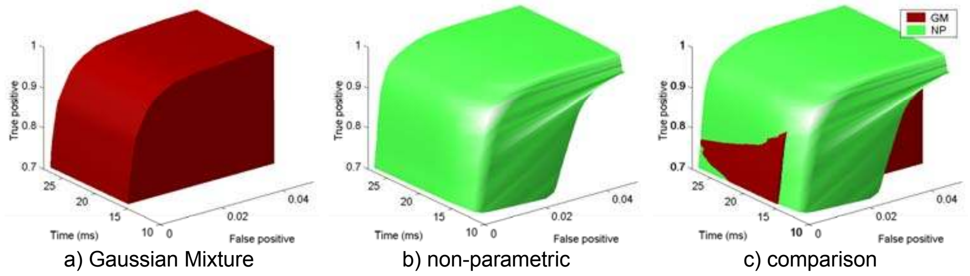
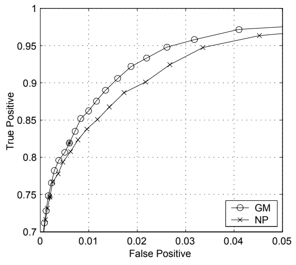
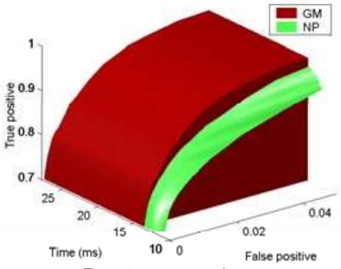


Figure 4.6: Constructing Pareto front for the 'cup' sequence

frame 573



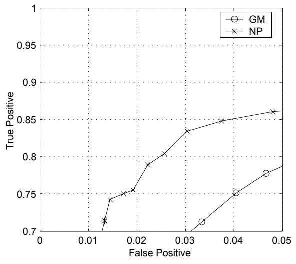
ROC curves



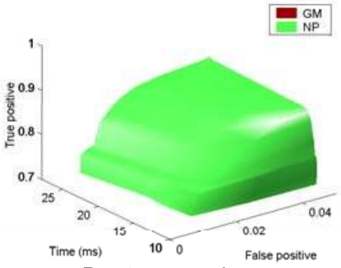
Pareto comparison

a)Traffic scene

frame 573



ROC curves



Pareto comparison

b)Waving trees scene

Figure 4.7: Pareto front comparison

are propagated to the higher level modules. Use of this higher level knowledge could greatly improve the object detection. For example, while tracking an object we could build a model of the object. This is discussed for example in [9] and [29]. Furthermore, for certain situations some appropriate higher level change detection schemes can be added to boost the adaptation process (see [25] for example).

- shadow detection - The intruding object can cast shadows on the background. Usually, we are interested only in the object and the pixels corresponding to the shadow should be detected [17].

For various applications some of the described additional aspects might be important. Taking them into consideration could lead to some improvements but, this is out of scope of this paper. Anyhow, many of the described improvements could be implemented as appropriate postprocessing steps. Therefore, the results presented can be used as an initial step for many of the presented practical background subtraction schemes.

4.7 Conclusions

Two efficient methods are presented for constructing and updating the density function of the background scene for each pixel. Some general principles for the pixel-based background subtraction are first given. This principles were used to derive the algorithms using some standard theory and some recent results. The performance of the algorithms is analyzed in detail. Both algorithms can deal with most of the standard problems of the background subtraction task. Furthermore, the influence of the few parameters of the algorithms is also studied.

Acknowledgements

We would like to thank our student René Willemink for valuable ideas and discussions and for the extensive testing of the algorithms.

Bibliography

[1] A. Berlinet and L. Devroye. A comparison of kernel density estimates. *Publications de l'Institut de Statistique de l'Universit de Paris*, 38(3):3–59, 1994.

[2] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[3] M.E. Brand. Structure learning in conditional probability models via an eutropic prior and parameter extinction. *Neural Computation Journal*, 11(5):1155–1182, 1999.

[4] J.P. Egan. *Signal detection theory and ROC analysis.* Academic Press, New York, 1975.

[5] A. Elgammal, D. Harwood, and L. S. Davis. Non-parametric background model for background subtraction. *In Proceedings 6th European Conference of Computer Vision*, 2000.

[6] M. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[7] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. *In Proceedings Thirteenth Conf. on Uncertainty in Artificial Intelligence*, 1997.

[8] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[9] M. Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-Gaussian background models. *In Proceedings of the European Conference on Computer Vision*, 2002.

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

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

[12] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. *In Proceedings of the 12th Int'l Conference on Pattern Recognition*, pages 126–131, 1994.

[13] H. Muller M. R. Everingham and B. T. Thomas. Evaluating image segmentation algorithms using the pareto front. *In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, Proceedings of the 7th European Conference on Computer Vision (ECCV2002), Part IV (LNCS 2353)*, 2002.

[14] G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley and Sons, 2000.

[15] V. Pareto. *Manual of political economy.* A.M. Kelley, New York (Original in french 1906), 1971.

[16] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[17] A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara. Detecting moving shadows: Formulation, algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, to appear.*

[18] C.E. Priebe and D.J.Marchette. Adaptive mixture density estimation. *Pattern Recognition*, 26(5):771–785, 1993.

[19] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *In Proceedings of the IEEE*, 77(2):257 – 285, 1989.

[20] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, 1956.

[21] J. Sacks. Asymptotic distribution of stochastic approximation procedures. *Annals of Mathematical Statistics*, 29:373–405, 1958.

[22] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *In Proceedings CVPR*, pages 246–252, 1999.

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

[24] D.M. Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society, Series B (Methodological)*, 2(46):257–267, 1984.

[25] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. *In Proceedings of the International Conference on Computer Vision*, 1999.

[26] J.K. Mui W.A. Yasnoff and J.W. Bacus. Error measures for scene segmentation. *Pattern Recognition*, (9):217–231, 1977.

[27] C. Wallace and P. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society, Series B (Methodological)*, 49(3):240–265, 1987.

[28] M.P. Wand and M.C. Jones. *Kernel Smoothing*. Chapman and Hall, London, 1995.

[29] P. J. Withagen, K. Schutte, and F. Groen. Likelihood-based object tracking using color histograms and EM. *In Proceedings of the International Conference on Image Processing, Rochester (NY), USA*, pages 589–592, 2002.

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

[31] Y.J. Zhang. A survey on evaluation methods for image segmentation. *Pattern Recognition*, (29):1335–1346, 1996.

Chapter 5

Two Video Analysis Applications

This chapter addresses two applications. The first application involves detection and tracking of moving vehicles along a highway whose traffic is being monitored using cameras. The second application is automatic analysis of tennis game matches. The both applications have the background/foreground segmentation as the basic video processing step.

5.1 Introduction

A huge amount of video material is produced daily: television, movies, surveillance cameras etc. As the amount of the available video content grows, higher demands are placed on video analysis and video content management. A general review of the image based content indexing is given in [1]. The video indexing is reviewed for example in Brunelli et al [2].

Probably the most frequently solved problem when videos are analyzed is segmenting a foreground object from its background in an image. After some regions in an image are detected as the foreground objects, some features are extracted that describe the segmented regions. These features together with the domain knowledge are often enough to extract the needed high-level semantics from the video material. In this paper we present two automatic systems for video analysis and indexing. In both systems the segmentation of the foreground objects is the basic processing step. The extracted features are then used to solve the problem. The first system (described in the next section) is a traffic video analysis system. The foreground objects that need to be detected are the vehicles on a highway. Usually, there is a huge gap ("semantic gap") between the low-level features extracted from the foreground objects and the high-level concepts. However, for this domain it was possible to manually map the extracted features to the events that need to be detected (high-level concepts) in a simple way. The second system (section 3 of this paper) analyzes videos of tennis games. It is difficult to manually generate the mapping from the features to the high-level concepts. Therefore we exploited the learning capability of Hidden Markov Models (HMMs) to extract high-level semantics from the raw video data automatically.

Although very specific, the two applications have many elements that are important for any surveillance/ monitoring system.

5.2 Traffic videos

One of the many tasks of the Dutch Ministry of Transport is construction and maintenance of dual carriage highways in The Netherlands. AVV is an advisory organ that among others gathers statistical data on road usage for traffic management and flow control. Nowadays, the video camera equipment is mainly used for dynamic goals, i.e. keeping control on what is going on. The recorded videotapes are usually re-used every 24 hours. Workers in the traffic control centers do not have time and capability to analyze the video material for gathering the statistical data.

5.2.1 Problem definition

In the project "Secure Multimedia Retrieval" (SUMMER) attention has been given to the possibilities of automatically extracting statistical data from traffic video material. A number of events of interest for the AVV were defined. The task was to automatically detect these events.

5.2.2 Object detection and feature extraction

A video camera was mounted above a highway, somewhere around the middle of one side of the highway (see figure 5.1). One camera monitors one side of the highway. The camera points in the direction of the traffic. The difficulties with the vehicles occluding each other are reduced if the camera is placed very high. However, mounting the camera too high was not practical. Therefore, the position of the camera was a compromise. The angle of the camera with the respect to the road was also a compromise. When the camera is for example parallel to the road we can see a long part of the road but the accuracy is decreased and there is a large amount of occlusion between the vehicles. The view angle of the camera lens is another parameter that needs to be chosen. The detailed analysis of the choices we made is given in [4].

Since the camera was static and the background scene was not changing rapidly we used a background subtraction method described in the previous chapter and also in [6]. See figure 5.1d and 5.1e for the results.

The camera intrinsic parameters (focal length, lens distortion parameters etc.) were estimated using a calibration object and the technique from [5]. Some standard dimensions of the road were known. The 3D model was fitted to the road to estimate the camera extrinsic parameters (position and orientation). See figure 5.1a and 5.1b. The 3D model is also used to segment the image into the driving lanes – figure 5.1c. We assumed that the road is a flat surface which is usually true for a small part of a highway.

After the background subtraction, it is straightforward to detect the vehicles that are entering the scene. Furthermore we also estimated the speed of the vehicles. The bumpers of the vehicles were detected and tracked, figure 5.1f. Tracking techniques are the topic of the next chapters. We assumed that the bumper is approximately at the same height from the road surface for all the vehicles. Therefore, since the camera was calibrated, we could estimate the speed of the vehicles. For details see [4].

5.2.3 Results

The following events are detected: unnecessarily left lane driving, slow traffic on left lane, slow traffic on right lane, fast traffic on left lane, fast traffic on right lane, left lane passing, right lane passing, lane change, vehicle on emergency lane, truck on left lane.

The clips containing the events are automatically cut out from the raw material and compressed. The clips and the statistical data have been stored in a database and a user interface has been build to access the data and the video images in combination with administrative data AVV has stored in separate databases, such as data on accidents, traffic jams, road details, etc.

The traffic monitoring was previously analyzed a number of times, for example in [3]. One of our contributions is in solving the occlusion problem by carefully choosing the camera parameters and position. Further, a demonstration system is built to show the integration of different information sources from different databases with the results from the traffic video analysis system. There was a predefined set of events to detect. In the preliminary experiments the system was able to detect the events from 4 hours of video under different weather and traffic conditions. A demonstration and some further details about the project are at the moment available at

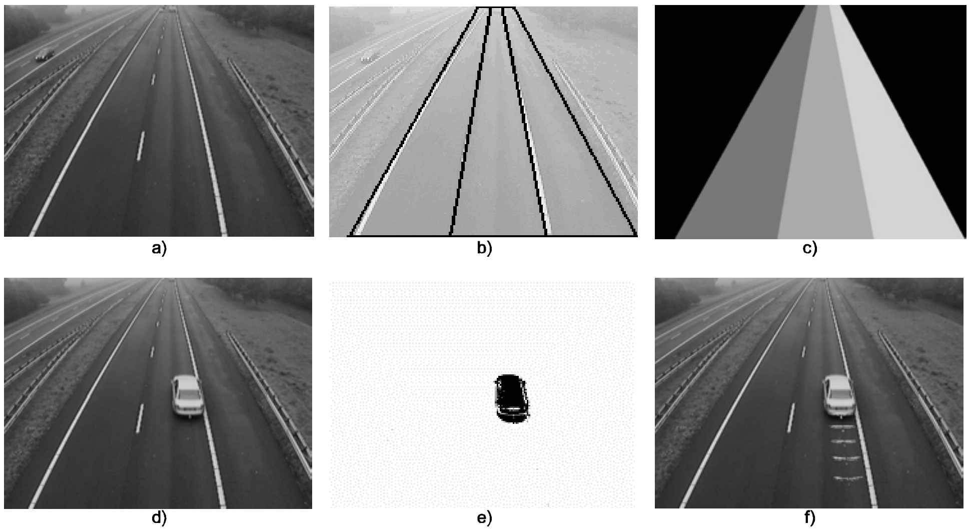
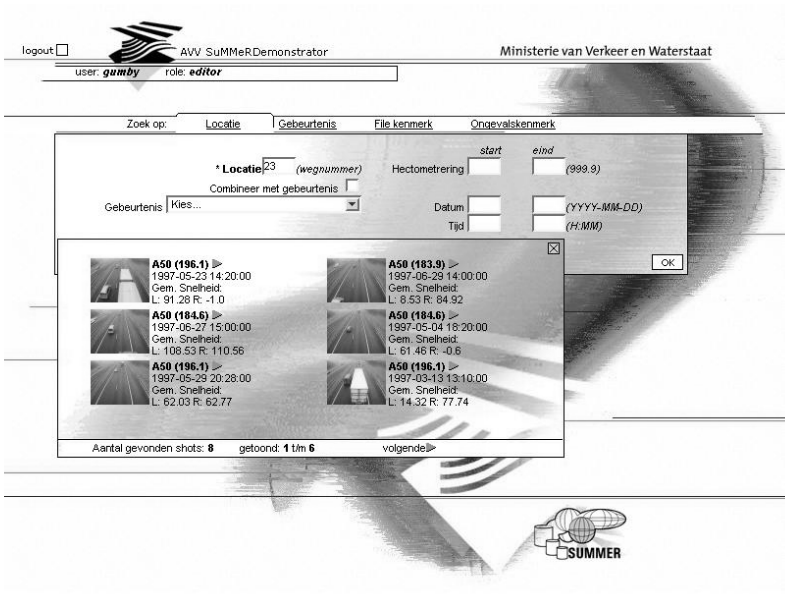


Figure 5.1: Vehicle segmentation and feature extraction: (a) Original image; (b) Fitted 3D model; (c) Segmented road; (d) Original image and a vehicle; (e) Segmented foreground; (f) Detected and tracked bumper of the vehicle.



user: **gumby** role: **editor**

* **Locatie** 23 (wegnummer)
Combineer met gebeurtenis
Gebeurtenis Kies...
Hectometrering start eind (999.9)
Datum (YYYY-MM-DD)
Tijd (H:MM)

A50 (196.1)
1997-05-23 14:20:00
Gem. Snelheid:
L: 91.28 R: -1.0

A50 (183.9)
1997-06-29 14:00:00
Gem. Snelheid:
L: 8.53 R: 84.92

A50 (184.6)
1997-06-27 15:00:00
Gem. Snelheid:
L: 108.53 R: 110.56

A50 (184.6)
1997-05-04 18:20:00
Gem. Snelheid:
L: 61.46 R: -0.6

A50 (196.1)
1997-05-29 20:28:00
Gem. Snelheid:
L: 62.03 R: 62.77

A50 (196.1)
1997-03-13 13:10:00
Gem. Snelheid:
L: 14.32 R: 77.74

Aantal gevonden shots: **8** getoond: **1** t/m **6** volgende

Figure 5.2: Demo user interface

: `www.cs.utwente.nl/~summer`. A screen shot of the final interface is given in figure 5.2.

5.3 Tennis game videos

The project "Digital media warehouse system" (DMW) aims to advance scalable solutions to content-based retrieval technique in large multi-media databases (see: `www.cs.utwente.nl/ dmw`).

5.3.1 Problem definition

A case study is done and the limited domain of tennis game videos was analyzed to demonstrate the extraction and querying of high-level concepts from raw video data. The aim was to recognize different tennis strokes from the ordinary TV broadcast tennis videos.

5.3.2 Object detection and feature extraction

From the whole video of a tennis game we use the game shots when the camera is observing the whole field as in figure 5.3a. These shots can be automatically extracted from the video using a number of global image features and some heuristics.

First step is to segment the player from the background. The camera was not always static so the problem was solved in a different way. The initial segmenting is by detecting the dominant color of the scene, figure 5.3b. The remaining lines are removed using a 3D model of the scene that was fitted to the scene automatically, figure 5.3c. An example of the final segmentation is given in figure 5.3d. Details about the segmentation are given in [8].

Further, we extract some specific features characterizing the shape of the segmented player's binary representation. The moving light display experiments from Johansson [10] demonstrated that people are able to recognize human activities provided by relatively little information (motion of a set of selected points on the body). Additional motivations for our approach are some recent results in recognizing human activities from their binary representations Fujiyoshi and A. Lipton [11] and Rosales and Sclaroff [12]. Besides the standard shape features such as orientation (f_1), and eccentricity (f_2), we extract the following features:

- The position of the upper half of the mask with respect to the mass center (f_{3-4}), its orientation (f_5), and the eccentricity (f_6). These features describe the upper part of the body that contains most of the information.

- A circle is centered at the mass center as shown in figure 5.3e. For each segment of the circle, we count the number of pixels in the mask (f_{7-14}). This can be seen as a general approximate description.

- The sticking-out parts (f_{15-16}) are extracted by filtering and finding local maximums of the distance from a point on the contour to the mass center. Only certain angles are considered as indicated in figure 5.3f.

The player position in the image was not used since it was not leading to any improvement according to the experiments.

5.3.3 Results

Parts of video showing different tennis strokes were manually extracted. Automatic extraction is also possible using the audio, see Petkovic (14) for further elaboration. We used first order left-to-right discrete HMMs with 4 to 48 states. We used the k-means algorithm to divide the feature space into the

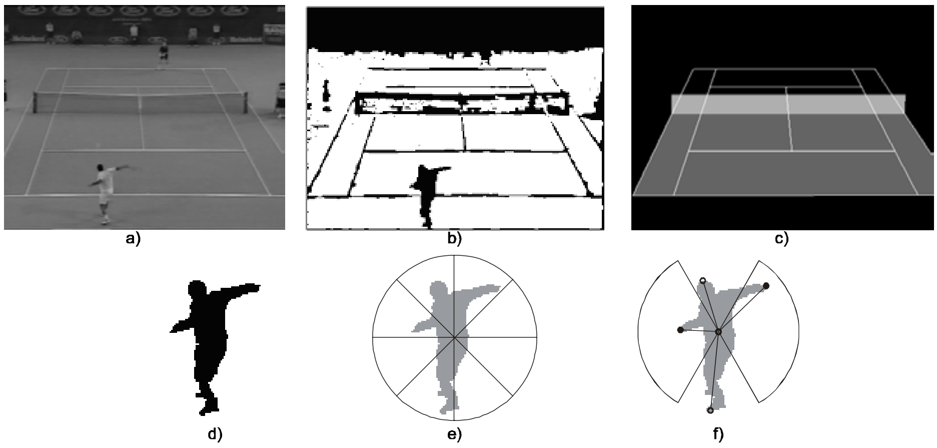


Figure 5.3: Player segmentation and feature extraction: (a) Original image; (b) Initial segmentation; (c) Fitted 3D model; (d) Final segmentation; (e) Pie features; (f) Skeleton features.

Features\ Experiment	1a	1b	2
f_{1-4}	82	79	76
f_{1-6}	85	82	80
$f_{1-2,5-6}$	81	78	76
$f_{1-2,15-16}$	89	88	87
f_{1-16}	86	82	79
$f_{2-4,15-16}$	91	89	88
f_{7-14}	85	78	78
f_{7-16}	93	87	86

Table 5.1: Recognition results (%)

discrete states - codebook. We tried various codebook sizes in the range of $8-80$ symbols.

The first experiment we conducted had two goals: (1) determine the best feature set and (2) investigate the person independence of different feature sets. Hence, we have performed a number of experiments with different feature combinations. In order to examine how invariant they are on different players (both male and female ones), two series of experiments have been conducted: 1a and 1b. In the series 1a, we used the same player in the training and evaluation sets, while in 1b HMMs were trained with one group of players, but strokes performed by other players were evaluated. In both cases, the training and the evaluation set contained 120 different sequences. To be able to compare our results with Yamato at al (13), we selected the same six events to be recognized: forehand, backhand, service, smash, forehand volley and backhand volley. In each experiment, six HMMs were constructed - one for each type of events we would like to recognize. Each stroke sequence was evaluated by all HMMs. The one with the highest probability was selected as the result (parallel evaluation). In order to find the best HMM parameters, a number of experiments with different number of states and codebook sizes were performed for each feature combination. The recognition accuracies in table 5.1 (% of correctly classified strokes using parallel evaluation) show that the combination of pie and skeleton features (f_{7-16}) achieved the highest percentage in the experiment 1a. The recognition rates dropped in experiment 1b as expected, but the combination of eccentricity, the mass center of the upper part, and skeleton features ($f_{2-4,15-16}$) popped up as the most person independent combination, which is nearly invariant on different player constitutions. The optimal result with this combination of features was achieved with the codebook size of 24 symbols and HMMs with 8 states. For this case and for our data set we present the number of true strokes and how they were classified as a matrix in table 5.2 (confusion matrix). We observe that the errors usually occur between the similar strokes as backhand volley and backhand, forehand volley and forehand and also forehand and service. Compared to (13), the improvement is around 20% (experiment 1b) ant it is mostly due to improved, more informative, and invariant features (in the first place the novel skeleton features and then the pie features). The improvement we achieved is certainly more significant taking into account that we used TV video scenes with a very small player shape compared to the close-ups used in (13).

In the second experiment, we investigated recognition rates of different feature combinations using 11 different strokes: service, backhand slice, backhand

true\rec.	backh.	foreh.	service	backh.vol	foreh.vol	smash
backh.	19	1				
foreh.		22	3			
service		3	43			
backh.vol	2			8		
foreh.vol		1	1		8	
smash			2			8

Table 5.2: An example confusion matrix

spin, backhand spin two-handed, forehand slice, forehand spin, smash, forehand volley, forehand half-volley, backhand volley, and backhand half-volley. The training and the evaluation set remained the same as in experiment 1b, only the new classification was applied. Although some strokes in this new classification are very similar to each other (for example volley and half-volley or backhand slice and spin), the performance (table 5.1, last column) dropped only slightly.

5.4 Conclusions

In general, to be able to completely understand the video material, computers need to achieve visual competence near the level of a human being. This is still far beyond the state of the art. Nevertheless, for particular applications it is possible to design systems that create the appearance of high-level understanding. A basic video analysis step is segmentation of the foreground objects from the background. The usefulness and importance of this step is illustrated in this paper. For two domains, traffic videos and tennis game videos, we presented here two video indexing systems that automatically extract the high-level concepts from the video using the segmented images.

5.5 Acknowledgements

The SUMMER project was a national project carried out with a subsidy from the Dutch Ministry of Economic Affairs. The DMW project was sponsored by the Dutch Telematica Institute. We would like to thank all the people participating in the both projects.

Bibliography

[1] A.W.M.Smeulders, M.Worring, S.Santini, A.Gupta, and R.Jain, 2000, "Content based image retrieval at the end of the early years", IEEE Tr.PAMI, 22(12), 1349-80

[2] R.Brunelli, O. Mich, C. M. Modena, 1999, "A Survey on Video Indexing", J. of Visual Communication and Image Representation, 10, 78-112

[3] D.J. Beymer, P. McLauchlan, B. Coifman and J. Malik, 1997, "A real time computer vision system for measuring traffic parameters", In Proc. CVPR

[4] R.J van Mierlo, 2002, "Video Analysis for Traffic Surveillance", MSc. Thesis, University of Twente

[5] Z. Zhang, 2000, "A flexible new technique for camera calibration", IEEE Tr.PAMI, 22(11), 1330-1334

[6] Z.Zivkovic, R. Willemink, 2002, "Adaptive background subtraction", Technical report MI028M02, University of Twente.

[7] Z.Zivkovic, F.van der Heijden, 2002, "Better Features to Track ", In Proc. ICPR, Canada.

[8] Z.Zivkovic, F.van der Heijden, M.Petkovic, W.Jonker, 2001, "Image processing and feature extraction for recognizing strokes in tennis game videos", In Proc. 7thASCI Conference, The Netherlands

[9] M. Petkovic, Z. Zivkovic, W. Jonker, 2001, "Recognizing Strokes in Tennis Videos Using Hidden Markov Models", In Proc. IASTED Int. Conf. Visualization, Imaging and Image Processing, Spain

[10] G. Johansson, 1973, "Visual perception of biological motion and a model for its analysis". Perception and Psychoph. 14(2), 210-211

[11] H. Fujiyoshi and A. Lipton, 1998, "Real-time Human Motion Analysis by Image Skeletonization", In Proc. IEEE Workshop on Applic. of Comp. Vis., 15-21

[12] R.Rosales and S.Sclaroff, 2000, "Inferring Body Pose without Tracking Body Parts", In Proc. CVPR

[13] J. Yamato, J. Ohya, K. Ishii, 1992, "Recognizing Human Action in Time-Seuential Images using Hidden Markov Model", In Proc. CVPR, 379-385

[14] M.Petkovic, "Content-based Video Retrieval Supported by Database Technology", PhD thesis, University of Twente.

Part III

Measuring motion

Chapter 6

Basic Image Motion

The problem considered here is how to select the feature points (in practice small image patches are used) in an image from an image sequence, such that they can be tracked well further through the sequence. Usually, tracking is performed by some sort of local search methods searching for a similar patch in the next image from the sequence. Therefore, it would be useful if we could estimate 'the size of the convergence region' for each image patch. It is less likely to erroneously calculate the displacement for an image patch with a large convergence region than for an image patch with a small convergence region. Consequently, the size of the convergence region can be used as a proper goodness measure for a feature point. For the standard Lucas-Kanade tracking method we propose a simple and fast method to approximate the convergence region for an image patch. A 'scale-space' point of view is also mentioned. In the experimental part we test our hypothesis on a large set of real data.

6.1 Introduction

The term "feature point" denotes a point in an image that is sufficiently different from its neighbors (L-corner, T-junction, a white dot on black background etc.). The position of a feature point is well defined and this is useful in many applications [17]. An important example is the simple 'optical flow' problem [10, 2] where the task is to find, for a feature point from one image, the corresponding point in the next image from a sequence. Usually it is assumed that some small neighborhood is also moving together with the point and therefore a small image patch around the point can be considered. When the displacements are small, the Lucas-Kanade algorithm [15] is commonly used to search for the similar patch from the next image. Furthermore, it is often useful to track the feature points further through the sequence. The positions of the tracked feature points are used for example in [3, 20] or in the 'structure and motion' algorithms [22, 1, 9].

The errors that occur during tracking should be detected. The task of 'monitoring', i.e. checking whether the points from a sequence that are found still look similar to the original feature point, is discussed in [21, 18] and further elaborated in [6, 11]. Furthermore, the false measurements can also be detected on a higher level of the processing chain, for example when the measurements are combined into 3D structure and motion estimates (see [9]). The problem considered in this paper is how to select the feature points from the initial image that are less likely to lead to false measurements (therefore suitable for tracking). Feature point selection strategies are analyzed and evaluated many times [17]. However, the selection in the important tracking context was not often analyzed previously. In [23] there is a tracking evaluation experiment for a few corner detectors. In [21] the Harris corner operator [8] is analyzed in connection with the accuracy of the matching (summarized in section 6.2). Standard feature point operators (usually corner detectors) give a numerical value, the so-called interest response (IR), at a pixel location based on the intensity values from the local image neighborhood. The points with high IR are the possible feature point candidates. The IR of the standard feature point detectors is related to the accuracy of the matching. However, tracking involves also some other factors. The practical tracking is performed by some sort of local search which might not converge to the correct solution. Furthermore, similar structures in the neighborhood can lead to mismatching that is hard to detect. With larger movements in the image (low temporal sampling) we can expect the mismatching problem to occur often. We propose an additional goodness measure, 'the size of the convergence region' (SCR), for the selected points which can help to identify and discard the point candidates that are likely to be unreliable. In section 6.3, for the Lucas-Kanade tracker we propose a simple method for estimating the SCR for a feature point. We show how this can improve the standard feature point detectors. We use two common, simple and fast corner detectors: the Harris corner operator (many times evaluated the best) and the recently often used SUSAN corner detector [19] (which is based on quite different principles). For the selected corners we estimate the SCR and show that the points with small SCR are usually the points that are erroneously tracked. For evaluation we use a large set of data with ground truth. In section 6.6, we further analyze the problem and propose yet another simple method that can lead to improvements. In this paper we considered the common Lucas-Kanade tracker. However, the main idea could be useful, if appropriately applied, for numerous other tracking/matching schemes.

6.2 Image Motion

The simplest and often used approach for calculating the movement of a small image patch from an image I_0 is to search the next image I_1 for a patch that minimizes the sum of squared differences [10, 2]:

$$J(\vec{d}) = \iint_W [I_1(\vec{x}_{im}) - I_0(\vec{x}_{im} + \vec{d})]^2 d\vec{x}_{im} \tag{6.1}$$

where W is the window of the feature (interest) point under consideration, $\vec{x}_{im} = [\ x_{im} \quad y_{im}\ ]^T$ presents the 2D position in the image plane and $\vec{d}$ is the displacement between the two frames. In practice the integration denotes simply summing over all the image pixels within the patch.

If we use a truncated Taylor expansion approximation in (6.1), we can find $\vec{d}$ that minimizes the sum of squared differences by solving:

$$\begin{aligned} Z\vec{d} &= \overrightarrow{e}, \text{ with} && (6.2)\\ Z &= \iint_W \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} d\vec{x}_{im} && (6.3)\\ \overrightarrow{e} &= \iint_W (I_0 - I_1) \begin{bmatrix} g_x & g_y \end{bmatrix}^T d\vec{x}_{im} && (6.4) \end{aligned}$$

Here, $g_x(\vec{x}_{im})$ and $g_y(\vec{x}_{im})$ are the derivatives of I_0 in the x_{im} and the y_{im} direction at the image point $\vec{x}_{im}$. The dependence on $\vec{x}_{im}$ is left out for simplicity.

The Lucas-Kanade procedure [15, 14] minimizes (6.1) iteratively. The solution of the linearized system (6.2) is used to warp the new image I_1 and the procedure is repeated. This can be written as:

$$\vec{d}(k+1) = \vec{d}(k) + Z^{-1}\overrightarrow{e}(k), \text{ with } \vec{d}(0) = 0 \tag{6.5}$$

where $\vec{d}(k)$ presents the estimated displacement at the k-th iteration. Equation (6.4) with the image I_1 warped using $\vec{d}(k)$ gives us $\overrightarrow{e}(k)$ (linear interpolation is usually used). The described algorithm is the *Gauss-Newton minimization procedure* (see [5], chapter 6).

The image derivatives and the matrix Z are calculated only once [7]. System (6.2) is solved in each iteration using the same matrix Z. Therefore the matrix Z should be both above the noise level and well-conditioned. This means that the eigenvalues λ_1, λ_2 of Z should be large and they should not differ by several orders of magnitude. Since the pixels have a maximum value, the greater eigenvalue is bounded. In conclusion, an image patch can be accepted if for some predefined λ we have:

$$IR_{Harris} = \min(\lambda_1, \lambda_2) > \lambda \tag{6.6}$$

The presented formulation is given in [21] (the Harris corner detector was originally implemented as $\det(Z) - \alpha \text{trace}(Z)^2$ where α is a constant).

6.3 Estimating the convergence region

We denote the true displacement by $\vec{d}^*$ and define $\vec{x}(k) = \vec{d}^* - \vec{d}(k)$. In the ideal case (no noise and no deformations) the minimized function (6.1) can be locally approximated by $J(\vec{x}) \approx \vec{x}^T Z \vec{x}$. This is another way to interpret the Harris operator given by (6.6). Here we introduce the notion of 'the convergence region' for a selected point, which is more global in nature.

The iteration equation (6.5) can be rewritten as:

$$\vec{x}(k+1) = \vec{x}(k) - Z^{-1}\vec{e}(k), \text{ with } \vec{x}(0) = \vec{d}^* \tag{6.7}$$

First we define:

$$V(\vec{x}) = \|\vec{x}\| \tag{6.8}$$

Successful tracking would mean that

$$V(\vec{x}(k)) = \|\vec{x}(k)\| \to 0 \text{ for } k \to \infty \tag{6.9}$$

The convergence region is the domain where for each initial displacement $\vec{x}(0)$ the tracking process converges. The size of this region would be an appropriate criterion to define how well the feature point could be tracked.

Suppose that we can find a domain S with the following properties:

$$\forall \vec{x}(k) \in S, \dot{V}(\vec{x}(k)) < 0 \text{ and } \vec{x}(k+1) \in S \tag{6.10}$$

with $\dot{V}(\vec{x}(k)) = V(\vec{x}(k+1)) - V(\vec{x}(k))$. Convergence is guaranteed within S since what we state is simply that we want to always move closer to the solution. Our function $V(\vec{x})$ is symmetric and monotonously increasing with

$\|\vec{x}\|$. If we find the point $\vec{x}_c$ closest to the origin for which $\dot{V}(\vec{x}_c) \geqslant 0$, the region $\|\vec{x}\| < \|\vec{x}_c\|$ will have the mentioned properties. The distance $\|\vec{x}_c\|$ can be used to describe the size of the estimated convergence region and consequently it is a proper feature point goodness measure denoted further as IR_{SCR}.

In figure 6.1 we present an illustrative example. We selected 30 'corner-like' feature points (7×7 pixels image patches). After a circular camera movement some of the feature points were erroneously tracked (black boxes). From the scatter diagram we observe that the radius of the estimated convergence region (x-axis,IR_{SCR} in pixels) discriminates the well tracked and the lost feature points. We also see that the smaller eigenvalue does not carry this information (y-axis, relative IR_{Harris} value with respect to the largest).

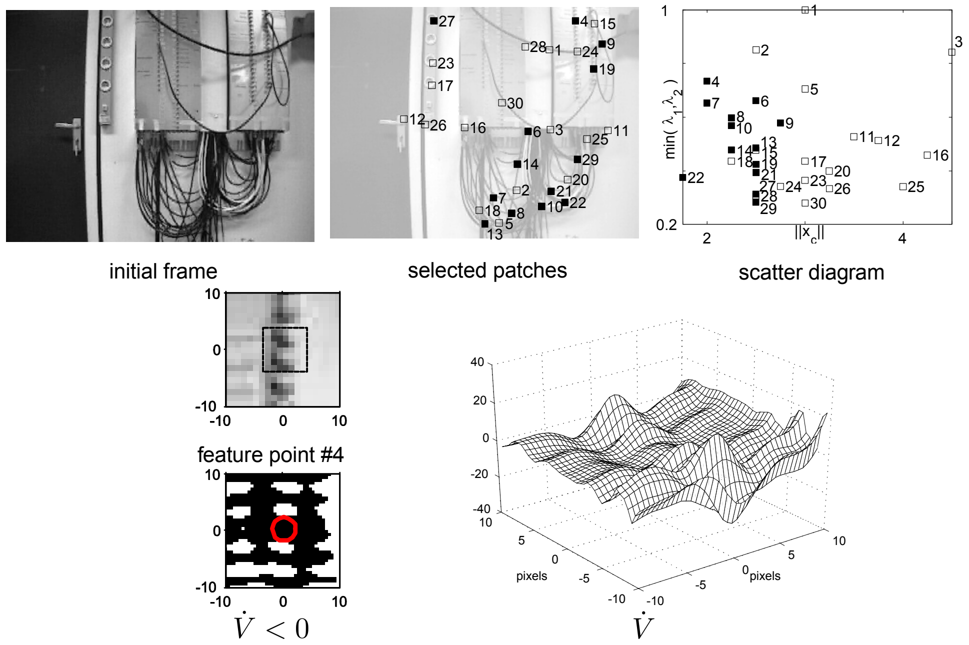


Figure 6.1: An illustrative experiment

The theory presented here is inspired by the nonlinear system analysis methods [24] and in this sense $V(\vec{x})$ corresponds to *the Lyapunov function.*

6.4 Implementation

The highly nonlinear function $\dot{V}(\vec{x})$ depends on the local neighborhood of the feature point. As an example see figure 6.1 (feature point 4). The function

$\dot{V}(\vec{x})$ is presented using a 0.5 pixel grid. We show also the estimated convergence region $\|\vec{x}\| < \|\vec{x}_c\|$ (indicated by the circle).

In practical implementation, for each feature point we compute $\dot{V}(\vec{x})$ for some discrete displacements around the feature point till we find the first $\dot{V}(\vec{x}) \geqslant 0$:

```
Input: $I_0$, $g_x$, $g_y$, $W$ , $Z^{-1}$, $SS$ (an array of 2D displacements $\vec{d}$ with non-
    decreasing $\left\|\vec{d}\right\|$- we use 8 points (angular sampling every $45^o$) on con-
    centric circles with radiuses increasing in 0.5 pixel steps starting from
    initial 0.5 pixel radius)

  1. $\vec{x}(0) = (\vec{d}^* =)SS(i)$
  2. Calculate $\overrightarrow{e}$ (window $W$ from $I_1$ simulated using $W$ shifted for $\vec{d}^*$ from
     $I_0$)
  3. $\vec{x}(1) = \vec{x}(0) - Z^{-1}\overrightarrow{e}$ (one Lucas-Kanade iteration step)
  4. If $\|\vec{x}(1)\| \geqslant \|\vec{x}(0)\|$ (equivalent to $\dot{V} \geqslant 0$) return $\|\vec{x}_c\| = \|\vec{x}(0)\|$
     else $\{i = i + 1$; go to 1$\}$

Output: $IR_{SCR} = \|\vec{x}_c\|$
```

The computational cost for a feature point is comparable with the computations needed for calculating the movement of the point. In our case, the average number of iterations (that are similar to the Lucas-Kanade iterations) is 8(because of every 45^o)$\cdot$2(because of 0.5 pixel sampling steps)$\cdot$ average$\|\vec{x}_c\|$. Increasing the number of the angular or the radial sampling steps does not lead to significant changes in the results we present in the next section while decreasing the number of sampling steps degrades the results. Further, in our experiments the algorithm was modified to stop when we find $\dot{V} \geqslant 0$ for the third time (step 4 from above is modified) and for IR_{SCR} we used the average of the three distances $\|\vec{x}_c\|$. This leads to small improvements and also less coarsely sampled results.

6.5 Experiments

The initial frames from the image sequences we used are presented in figure 6.4. The sequences are from the CMU VASC Image Database. The sequence "marbled-block" used in [16] is added (complex motion - both camera and an object are moving). The sequences exhibit a variety of camera movements, object textures and scene depth variations. Most of the sequences have 512×480 format. The chosen sequences have very small displacements between the consecutive frames. Therefore, it was possible to select and track the feature points (7x7 pixel patches) for some short time. This was used as the ground truth. To generate more difficult situations and some errors we start again from the initial frame and calculate, for the selected feature points, the displacement between the initial and i-th frame in the sequence (skipping the frames in between) using the Lucas-Kanade procedure with fixed 20 iterations. We choose i so that per sequence for at least 20% of the feature points the displacement is erroneously calculated.

First we selected 'corner-like' points having $IR_{Harris} > 0.05$. From initial 2143 points 754 lead to false measurements. We select the same number of feature points using the SUSAN corner detector and get 876 'bad' points. The worse performance of the SUSAN detector in the tracking context is in correspondence with [23]. In our experiments we use the 3×3 Sobel operator for the image derivatives. For the SUSAN corner detector we use usual 3.5 pixels radius circular neighborhood for the feature points (giving a mask W containing 37 pixels). If I_C is the intensity value at the center pixel the response function is $IR_{SUSAN} = 37/2 - \sum_W \exp(-(I(\vec{x}_{im}) - I_C)/t)^6)$. Negative values are discarded. For additional details see [19]. For our data, we have empirically chosen $t = 15$. For both SUSAN and Harris detectors the feature points are the local maxima but constrained to be at a minimum of 15 pixel distance from each other.

During the selection we need to set a threshold and discard the features point candidates having IR below the threshold. If we plot the results for different thresholds we get a 'receiver operator characteristic' (ROC) curve that shows the discriminative power of the IR. For our data set with the ground truth (total of 2143 feature points selected from the sequences) we plot the empirical ROC curves (linear interpolation is used between the points on the curve). A feature point belongs to the true-positives if it was selected and it was well tracked. The false-positives are the points selected but lost. Relative values are used (divided by the total number of the well tracked and the 'bad' ones respectively). The ROC curves in figure 6.2 show clearly the large improvements when the new defined IR_{SCR} is used.

6.6 Feature points and scale

The problems with tracking usually occur when the points are selected in the areas where we can find similar structures in the neighborhood. An example is the feature point 4 presented in figure 6.1. Similar structures in the vertical direction make this point highly unreliable (properly detected using the SCR). If we blur the image (convolve with a Gaussian kernel with standard deviation σ), the close similar structures will merge and become indistinguishable [12]. For example the feature point 4 will be in an area that after blurring looks like a vertical line and therefore not interesting anymore. For the problematic point 4 and the isolated and well tracked point 12, we show in figure 6.3 how the IR_{Harris} changes with blurring the image. Although initially the point 4 was ranked more promising, after some blurring the point 12 is ranked correctly as the better one. In conclusion, another simple method for detecting and discarding the weak points is to blur the images and check if the points can still be selected. The results using IR_{Harris} with blurring (for different σ) are illustrated in figure 6.3 by plotting the ROC curves for our data set. The optimal result for our data set was achieved using $\sigma = 2.5$ and in figure 6.3 we show that it is similar to the result using SCR. Although, as presented, the feature point 4 can be correctly identified as unreliable by both proposed criteria, $IR_{Harris(\sigma)}$ (Harris with blurring) and IR_{SCR} do not describe the same effects. Further improvement can be achieved by combining them. In figure 6.3 we show also the ROC curve for the empirical combination $IR_{SCR} + \log(IR_{Harris(\sigma=2.5)})$. Finding the optimal combination is beyond the scope of this paper. In table 6.1 we present the values for the area under ROC curve (AUC) [4] for all the presented experiments.

We propose to use the new defined IR_{SCR} and $IR_{Harris(\sigma)}$ as an additional check for the feature point candidates selected initially using a standard corner detector (IR_{Harris}). Using only the IR_{SCR} to directly select the points is not advised since it does not consider the accuracy of the matching. Besides, computing IR_{SCR} for all the image pixels is computationally expensive. One might suggest to use the blurred $IR_{Harris(\sigma)}$ to directly select the points. However, blurring images changes the positions of the detected corners (local maxima) [12, 13]. The corners detected in the blurred images might not have the desired properties described by IR_{Harris} in the original images where the tracking is actually done. For example, detecting corners in the blurred images using $\sigma = 2.5$ leads to a large number of points that are highly unreliable when we look the original images. The best result for our data set obtained with $\sigma = 1.5$ (figure 6.3 and table 6.1) is still quite poor compared to the other proposed methods.

6.7 Conclusions and recommendations

The problem of estimating the motion of a feature point has two aspects: the accuracy of the result and the convergence of the tracker. The accuracy is well addressed by the standard feature point detectors. The corner-like points can be accurately matched. The Harris corner operator is a nice example. A well conditioned matrix Z assures low sensitivity to the noise but only if the tracking converges. Our new goodness measures are related to the convergence of the tracker. Consequently, the new measures should be used as an additional check to improve the selection. Practically, the effect is that the 'clean' corners, like L-corners from a structure in man made environment, that are usually fairly stable during tracking are preferred. The feature points from a local area with rapidly changing and inconsistent gradients are usually unreliable and discarded by the new criteria. An important case are the areas with repetitive structures (an example is discussed in the previous section). Tracking can easily switch to a similar structure in the neighborhood and this errors would be difficult to detect.

Having no prior knowledge about the scene we select small patches (here it was 7×7 pixel) for tracking and assume no deformation between successive frames. For the feature point candidates (local maxima of the IR_{Harris} or possibly IR_{SUSAN}) we compute the additional IR_{SCR} or $IR_{Harris(\sigma)}$ (blurred $IR_{SUSAN(\sigma)}$ is also possible) and discard the problematic ones. By blurring the images we consider a larger neighborhood of a feature point. When calculating IR_{SCR} we use only the initial image and therefore also consider a larger neighborhood. Although the 'no deformation' assumption between frames has less chance to be valid, using the larger neighborhood in this way seems to be useful in practice. The AUC results (table 6.1) show large improvements for the Harris detector IR_{Harris} with the additional check using the IR_{SCR} or the blurred $IR_{Harris(\sigma)}$. The empirical combination of the new measures leads to even better results for our data set.

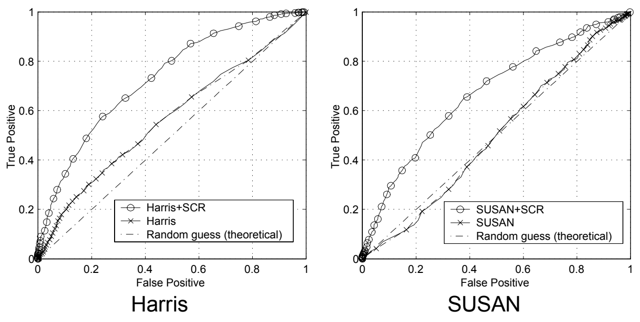


Figure 6.2: Improvement using SCR

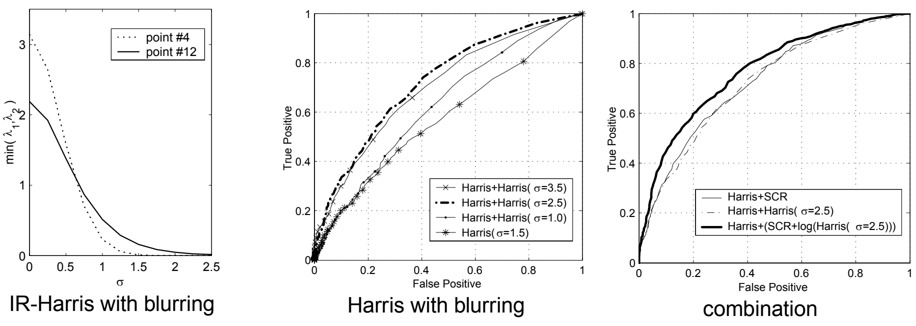


Figure 6.3: Improvement with blurring

Corner Selection + Additional check	AUC (Standard Error)
Harris + (SCR + log(Harris($\sigma = 2.5$)))	0.77(0.010)
Harris + SCR	0.73(0.011)
Harris + Harris($\sigma = 2.5$)	0.72(0.011)
Harris + Harris($\sigma = 3.5$)	0.70(0.011)
SUSAN + SCR	0.67(0.012)
Harris + Harris($\sigma = 1.0$)	0.63(0.012)
Harris($\sigma = 1.5$)	0.58(0.013)
Harris	0.56(0.013)
SUSAN	0.50(0.013)

Table 6.1: Area under ROC curve- comparison

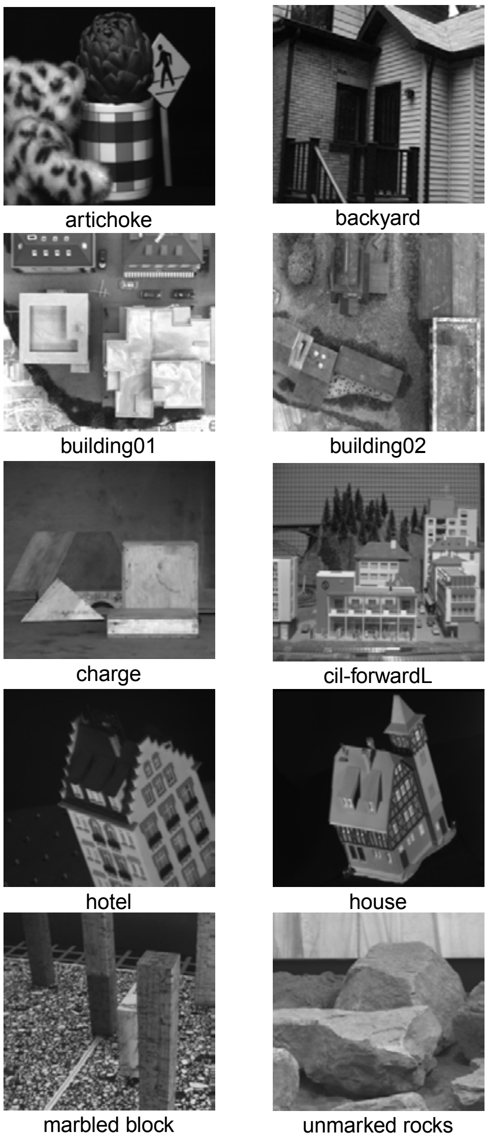


Figure 6.4: Image sequences

Bibliography

[1] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 6(17), 1995.

[2] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, September 1995.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. *In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[4] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[5] R. Fletcher. *Practical Methods of Optimization.* J. Wiley, 1987.

[6] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving feature tracking with robust statistics. *Pattern Analysis and Applications*, 2(4):312–320, 1999.

[7] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[8] C. Harris and M. Stephens. A combined corner and edge detector. *In Proceedings of 4th Alvey Vision Conference*, pages 147–151, 1988.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[10] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow: a retrospective. *Artificial Intelligence*, 59(1-2):81–87, January 1993.

[11] H. Jin, P. Favaro, and S. Soatto. Real-time feature tracking and outlier rejection with changes in illumination. *In proceedings of ICCV*, pages 684–689, 2001.

[12] A. Kuijper and L.M.J. Florack. Understanding and modeling the evolution of critical points under gaussian blurring. *In Proceedings of the 7th European Conference on Computer Vision*, pages 143–157, 2002.

[13] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998.

[14] B.D. Lucas. *Generalized Image Matching by the Method of Differences.* PhD Thesis, Dept. of Computer Science, Carnegie-Mellon University, 1984.

[15] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *In Proceedings IJCAI81*, pages 674–679, 1981.

[16] Michael Otte and Hans-Hellmut Nagel. Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced image sequences. *Artificial Intelligence*, 78(1/2):5–43, November 1995.

[17] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[18] J. Shi and C. Tomasi. Good features to track. *In Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[19] S.M. Smith and J.M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[20] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion models. *In, T. G. Dietterich and S. Becker and Z. Ghahramani, editors, Advances in Neural Information Processing Systems 14*, 2002.

[21] C. Tomasi and T. Kanade. Detection and tracking of point features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

[22] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 2(9):137–154, 1992.

[23] M. Trajkovic and M. Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998.

[24] M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice-Hall, 1993.

Chapter 7

3D Object Tracking

A simple method is presented for 3D head pose estimation and tracking in monocular image sequences. A generic geometric model is used. The initialization consists of aligning the perspective projection of the geometric model with the subjects head in the initial image. After the initialization, the gray levels from the initial image are mapped onto the visible side of the head model to form a textured object. Only a limited number of points on the object is used allowing real-time performance even on low-end computers. The appearance changes caused by movement in the complex light conditions of a real scene present a big problem for fitting the textured model to the data from new images. Having in mind real human-computer interfaces we propose a simple adaptive appearance changes model that is updated by the measurements from the new images. To stabilize the model we constrain it to some neighborhood of the initial gray values. The neighborhood is defined using some simple heuristics.

7.1 Introduction

The reconstruction of 3D position and orientation of objects in monocular image sequences is an important task in the computer vision society. This paper concentrates on 3D human head tracking. The applications we have in mind are: model-based coding for video conferencing, view stabilization for face expression recognition and various possible human-computer interface applications. Anyway, the approach proposed here can be applied in general for rigid object tracking in 3D.

In the initialization procedure we align our generic geometric head model with the observed subject's head. This can be done manually, or automatically by using some other algorithm [12]. For new images in the sequence, tracking consists of estimating the human head pose with respect to this initial pose. Because of the perspective projection of standard cameras it is possible to estimate the 3D pose from the 2D image data. We use an initially aligned generic geometric 3D head model. Therefore, as described later, the 3D pose is estimated only up to a scaling factor. However, this is of no importance for the applications we are considering.

The paper is organized as follows. Related work is presented in the next section. Then the geometric part of our model based approach is described. The adaptive radiometric model is presented in section 7.6. Finally, the whole algorithm is described and some experimental results are discussed.

7.2 Related work

One of the big problems in tracking algorithms is the object appearance change caused by movement under realistic light conditions. These effects are usually very hard to model. In almost all realistic situations light conditions are complex and unknown.

Many 3D head tracking methods start from tracking some distinctive feature points on the head (for example eyes, nose, mouth corners etc.) in the 2D image plane [13]. The appearance changes caused by movement in realistic light conditions are addressed by choosing appropriate similarity norms for tracking the selected feature points. A generic 3D model is then fitted to these 2D measurement to estimate the 3D head pose. The biggest drawback is that features can be lost because of occlusions or some other not modeled effects. Knowledge about the 3D object geometry can be used to predict feature point occlusion and to recover it if it appears again. An attempt is reported in [11] where they also used the 2D feature trajectories in a structure from motion algorithm to update the generic 3D model geometry.

Another way is to use the generic 3D model geometry directly. This is usually done by forming a textured 3D head model using the initial image [1]. This textured model is then fitted to the data from the new images. We also use the textured 3D model in this paper. In [10] the so called 'backward procedure' is proposed for fitting the model to the data where the roles are switched and a new image is actually fitted to the model. This allows efficient implementation since the needed derivatives (see section 7.7) are calculated only once for the initial image. However, this is not possible for the adaptive procedure we propose here. An efficient solution is obtained here by using only a limited number of points on the object and the response of a Gaussian filter at these points as a model with reduced size.

In practice, because of complex light conditions, head movements introduce large changes in the texture of the previously described textured 3D head model. An approach is to form a large image database of the object under various light conditions. Then a model should be constructed from the data. This is usually done by finding a representative orthogonal linear subspace using *principal component analysis* (PCA) [14] [9]. This subspace is used to represent the whole database (all possible appearances of the object when it moves in realistic light conditions). How much is a new image "face like" is calculated by measuring the distance from this subspace. This "brute force" method needs a long and hard to perform preparation procedure, which is highly unpractical for real user interface applications. A textured cylindrical model with PCA appearance changes modeling is presented in [7]. We search here for other simpler and more appropriate solutions. In a typical situation we have only one image of the object - the initial image. Then, using some heuristics we define some neighborhood around the initial gray values to constrain possible object appearance changes.

No big appearance changes are expected for small movements between two consecutive images. We then try to use the gray levels from the new images to update the 3D objects texture. This is somewhat similar to the methods that use *optical flow* (movement of gray level patterns in the images) as their input [3]. Because of error accumulation these methods were not able to deal with longer image sequences. Some solutions were proposed trying to prevent this 'drift away' [8] [15]. Our method constrains the texture appearance to the neighborhood of the initial values and in this way prevents the 'drift away'.

7.3 Model based approach

We use a model-based approach where we try to find the best fit of the model to the images in the sequence. The parameters we want to estimate are contained in the vector:

$$\vec{q} = [\ x \quad y \quad z \quad \alpha \quad \beta \quad \gamma\ ]^T \tag{7.1}$$

where x, y, z describe the position and α, β, γ are the Euler angles describing the head orientation in the camera coordinate system.

If we don't take into account the previous history of the head movement and we consider all the image pixel measurements equally important, the problem can be formulated as follows:

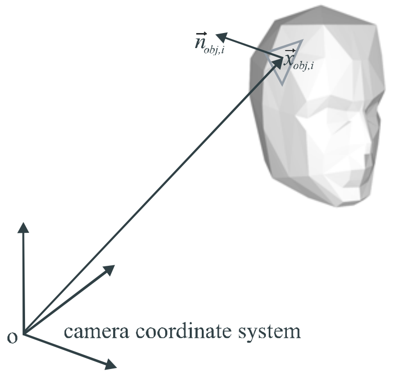


Figure 7.1: The Geometric Model

$$\widehat{\vec{q}} = \underset{\vec{q}}{argmin}\left(\int_{image} (model(\vec{q}) - currentimage)^2\right) \tag{7.2}$$

where integration is done over the whole image area and $\widehat{\vec{q}}$ presents the estimated pose parameters. Here, *model*($\vec{q}$) presents the model generated image and *currentimage* is the current image from the camera.

In practice, it is not feasible to have the complete model of the imaging process. Therefore, we are bound to use a number of approximations. We divide the model into two parts: a geometric part and a radiometric part. These two parts are described in the next sections.

7.4 The geometric model

There are various ways to describe the geometry of 3D objects [2]. We use a triangular mesh, the common representation supported by fast graphics hardware. The mesh we use (Figure 7.1) is generated as an attempt to represent the 3D geometry of a human head.

Let $\vec{x}_{obj,i} = [\ x_{obj,i} \quad y_{obj,i} \quad z_{obj,i}\ ]^T$ present the position of a fixed point i on the object's surface in the camera coordinate system. This position, of course, depends on the head pose $\vec{x}_{obj,i} = \vec{x}_{obj,i}(\vec{q})$. For simplicity of notation we will further often omit $\vec{q}$.

We assume that the camera is calibrated. Therefore we know the perspective projection function $\vec{x}_{im,i} = p(\vec{x}_{obj,i})$ of the camera lens system that projects the 3D point $\vec{x}_{obj,i}$ to the 2D image plane point $\vec{x}_{im,i}$. If the camera doesn't introduce any distortions we have:

$$\vec{x}_{im,i} = \begin{bmatrix} x_{im,i} \\ y_{im,i} \end{bmatrix} = p(\vec{x}_{obj,i}) = f \cdot \begin{bmatrix} \frac{x_{obj,i}}{z_{obj,i}} \\ \frac{y_{obj,i}}{z_{obj,i}} \end{bmatrix} \tag{7.3}$$

where f presents the focal length of the camera lens.

A generic geometric human head model is used. The size of the subject's head is unknown and we don't want to complicate the initialization procedure. Therefore, the initial position (contained in $\vec{q}_0$) is known only up to a scaling factor. As a consequence 3D head position is estimated only up to the scaling factor. As mentioned before, this doesn't present a problem for the applications we are considering.

7.5 Problem definition

We define a set of test points on the object $\vec{x}_{obj,i}$. For our triangular mesh model we choose the centers of the triangles in the mesh as shown in Figure 7.1. Our proposal is to check the fit of the model only at these object defined points and not to try to reconstruct the image. This can heavily reduce the amount of data to be processed and speed up the tracking algorithm. Therefore, our problem (7.2) for the k-th image can be redefined as:

$$\begin{aligned} \widehat{\vec{q}_k} &= \arg\min_{\vec{q}_k} [\frac{1}{\sum_i w(i, \vec{q}_k)} \\ &\quad \sum_i w(i, \vec{q}_k) \rho(M_k(i) - I_k(p(\vec{x}_{obj,i}(\vec{q}_k))))] \end{aligned} \tag{7.4}$$

where the summing is done over all test points. Here, $M_k(i)$ presents the model-predicted gray value to be observed when $\vec{x}_{obj,i}$ is projected to the image plane and $I_k(p(\vec{x}_{obj,i}))$ is the actual observed value at that position in the current image. We search for the pose parameters $\widehat{\vec{q}_k}$ that give the best fit.

The measurements are weighted by:

$$\begin{aligned} & w(i, \vec{q}_k) = \\ & \quad \begin{cases} A(i) \cdot \vec{x}_{obj,i} \cdot \vec{n}_{obj,i}, & \text{for } \vec{x}_{obj,i} \cdot \vec{n}_{obj,i} < 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{7.5}$$

A point $\vec{x}_{obj,i}$ corresponds to a triangular patch i of the object surface as defined by the generic geometric model. The size of the patch is denoted by

$A(i)$. The current normal of the patch is described by $\vec{n}_{obj,i} = \vec{n}_{obj,i}(\vec{q}_k)$. In total, the weight $w(i, \vec{q}_k)$ presents the visible size of the triangular patch i. Note that occlusion by other triangular patches is not included in this model. However we don't expect such situations to occur often.

Because of many not modeled effects some measurements can contain unexpectedly high errors. Therefore, instead of the standard quadratic norm we use the less sensitive Geman & McClure robust error norm:

$$\rho(x) = \frac{x^2}{1 + x^2/\sigma^2} \tag{7.6}$$

Here, σ controls the difference beyond which a measurement is considered as an outlier [4].

7.6 The radiometric model

The radiometric model describes which gray value $M_k(i)$ is expected to be observed in the k-th image when object point $x_{obj,i}$ is projected onto the image plane. In general this depends on the local surface radiometric properties, local surface orientation and light conditions. Approximate radiometric models exist [6] and theoretically $M_k(i)$ should then be written as: $M_k(i, \vec{x}_{obj,i}(\vec{q}_k), \vec{q}_k)$. However, the local surface properties are unknown. Also the lighting conditions in real scenes are very complex in general and we are forced to use a number of approximations.

7.6.1 Approximate radiometric models

After initial alignment of the 3D object with the first image ($k = 0$) in the sequence we can obtain the values $M_0(i)$ for the test points $\vec{x}_{obj,i}$ visible for that head model pose:

$$M_0(i) = I_0(f(\vec{x}_{obj,i}(\vec{q}_0))) \tag{7.7}$$

where the $\vec{q}_0$ presents the parameters selected to align the generic model with the subjects head in the initial image.

The simplest approximate model is the so called *constant brightness assumption* that predicts the gray value in the k-th image as:

$$M_k^{cb}(i) = M_0(i) \tag{7.8}$$

This model is correct for Lambertian surfaces and with only ambient light present, which is far from realistic. A simple relaxation is to allow global

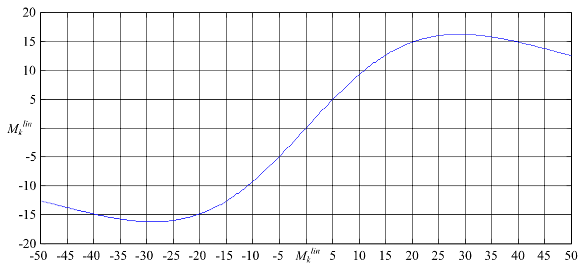


Figure 7.2: Function $\eta(I_k)$ for $\beta = 0.0004$

brightness changes by adding a constant a to all points gray values. Further approximation is to include linear brightness changes in the image plane over the object.[5]. This crude model can be written as:

$$M_k^{lin}(i) = M_0(i) + a + [\ b \quad c\ ] \cdot \vec{x}_{im,i} \tag{7.9}$$

where we have a dot product of the vector $[\ b \quad c\ ]$ and image projection of the i-th object point, vector $\vec{x}_{im,i}$. The parameters a, b, c should be estimated for each new image k.

We use this additive model to describe some global illumination changes. Although this model does not need any preparation procedure, the changes in the appearance of the human face are to complex to be well approximated in this way. We introduce an adaptive model in the next section which allows floating around this model.

7.6.2 Adaptive radiometric model

For small object movements between two consecutive images we don't expect large changes in appearance and the *constant brightness* model can still be used. Then, an adaptive model can be formed. After model fitting on the new image using the *constant brightness assumption* between two images, the measurements from the new image can be used to update the model. The predicted value for the next $k+1$-th image becomes:

$$M_{k+1}^{adaptive}(i) = M_k^{adaptive}(i) + \alpha \cdot (innovation) \tag{7.10}$$

$$innovation = I_k(p(\vec{x}_{obj,i}(\widehat{\vec{q}}_k))) - M_k^{adaptive}(i) \tag{7.11}$$

here constant α encodes our assumption that the gray value is not supposed to change rapidly by taking into account the previous values with exponentially decreasing weights (1st order AR filtering). With this kind of *innovation* we have the error accumulation problem but now low-pass filtered. For $\alpha = 1$ this is similar to some optical flow approaches, and for $\alpha = 0$ we get the *constant brightness assumption model.*

The initially obtained values $M_0(i)$ contain the gray values for certain head pose and illumination. We can try to use this measurements too to form the *innovation*. A crude approximation of the appearance changes from this initial values is the linear model described by (7.9). Our assumption is that the gray values are not going far away from this model. We incorporate this in the *innovation* by using the following combination of the current measurement $I_k(p(\vec{x}_{obj,i}))$ and linear model $M^{lin}(i)$ which is based on the initial measurements $M_0(i)$:

$$\begin{aligned}&\eta(I_k(p(\vec{x}_{obj,i}))) = \\ &\quad \frac{I_k(p(\vec{x}_{obj,i}))-M_k^{lin}(i)}{(1+\beta\cdot(I_k(p(\vec{x}_{obj,i}))-M_k^{lin}(i))^2)^2} + M_k^{lin}(i)\end{aligned} \tag{7.12}$$

The function η compresses the measured values $I_k(p(\vec{x}_{obj,i}))$ to some neighborhood of the simple model $M_k^{lin}(i)$ as presented in Figure 7.2. This is controlled by the constant β. Note that this function has actually the form of the derivative (*influence function*) of the robust norm introduced in (7.6).

Finally we define our adaptive model with:

$$innovation = \eta(I_k(p(\vec{x}_{obj,i}))) - M_k(i) \tag{7.13}$$

This simple model encodes our two assumptions. First, the gray values are not changing rapidly, controlled by the parameter α. Second, we approximate changes from the initial values $M_0(i)$ by a linear model $M_k^{lin}(i)$ and assume that the gray values remain in the neighborhood of this simple model , controlled by the parameter β. Only initial alignment of the 3D model is needed to form the model.

7.7 Algorithm

For each new image we have to find the optimal head pose vector $\widehat{\vec{q}}$ according to (7.4). We already need the initial alignment of the 3D model. Afterwards, we assume that there are no large changes in head pose between two successive images and for each new image we use the previous head pose as the starting position. Than we search for the nearest local minimum using Gauss-Newton iterative procedure. For determining image derivatives we use Gaussian kernels. Our measurements are also done with Gaussian blurred image at the same scale.

Note that (7.4) has also the weights $w(i, \vec{q}_k)$ described by (7.5) that depend on the current pose $\vec{q}_k$. For simplicity, we don't include this in derivatives for the Gauss-Newton iterative procedure. Anyway, this is included in the line search part after we determine the search direction. Also, the robust norm is included only as a weight factor in every iteration forming in total an iteratively reweighted least square (IRLS) minimization procedure.

Further, the parameters a, b, c for the linear approximate radiometric model should also be estimated. This could be done together with $\widehat{\vec{q}}$. For simplicity we do this separately. Since (7.9) is linear with respect to its parameters this is done in a single iteration. The same weights $w(i, \vec{q}_k)$ described by (7.5) and the same robust norm (7.6) are used.

Finally the whole algorithm can be described as follows:

1. initialization

   input: initial image I_0 and pose $\vec{q}_0$

   output: initial texture $M_0(i)$

   - obtain $M_0(i)$ for the visible points according to the initial pose $\vec{q}_0$

2. tracking -

   input: current image I_k, current texture $M_k(i)$ and predicted pose $\overline{\vec{q}}_k (= \vec{q}_{k-1}$ in our case, we don't use any temporal model for head movement in this paper)

   output: $M_{k+1}(i), \vec{q}_k, \overline{\vec{q}}_{k+1} = \vec{q}_k$

   - constant brightness assumption, find optimal $\vec{q}_k$ according to (7.4)
   - fit the approximate linear model, find a, b, c
   - update model, according to (7.10) and (7.12)

7.8 Experiments

Various experiments were conducted. Our unoptimized test version of the algorithm was able to work at standard PAL 25 frames/second even on a low-end Pentium Celeron 500MHz. The time needed for an IRLS algorithm iteration was less than 10ms. We used three iterations per image and the

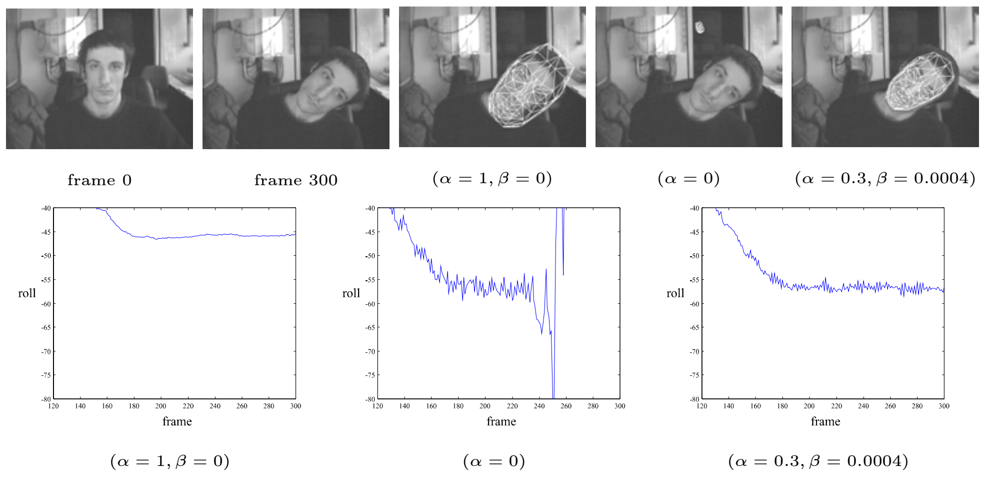


Figure 7.3: Adaptive model and estimated angle γ (roll rotation)

rest of the time was used for visualization. A cheap web camera is used that gives 320x240 pixel images with a large amount of noise. Using few images of known objects we approximately determined the camera focal length in a simple experiment. The camera pixels are assumed to be squares. Smoothing and differentiation is done with Gaussian kernels (with standard deviation =2). For the robust norm $\sigma = 100$ is used.

7.8.1 Experiment 1

To illustrate the operation of the adaptive model we constructed an experiment where the subject has rotated his head parallel the image plane (roll rotation) and than remained in that position. We wanted to investigate only the influence of the appearance changes. This kind of movement is chosen because it doesn't suffer from the geometric model errors. The light conditions were chosen to be not too difficult (no specular reflections and only small global brightness changes). For better comparison instead of the linear approximate model $M^{lin}(i)$ in (7.12) we used only the *constant brightness* model $M^{cb}(i)$. The adaptive algorithm (here only around $M^{cb}(i)$) for $\alpha = 0.3$ and $\beta = 0.0004$ could handle the changes but they were to big for the pure constant brightness approach ($\alpha = 0$) which diverged after some time. It was also quite instable before it diverged (see Figure 7.3). For parameters $\alpha = 1$ and $\beta = 0$ the adaptive model can drift away similar to some optical flow approaches. As it can be observed in Figure 7.3, after this short movement the model was already not fitting the target properly.

7.8.2 Experiment 2

We conducted a series of experiments in typical office conditions at various locations. Some captures from the tests are presented in Figure 7.4. Difficult light conditions caused large appearance changes. The movements were of normal speed. Rapid movements can also be handled except for large out the plane rotations (pitch and jaw rotations). Out the plane rotations of up to approximately 35 degrees can be handled. This, however, depends on the camera focal length and object -camera distance. Web cameras have usually very small focal length and for this angle we could almost see only one half of the head (see the figures). For the parameters α and β we always used $\alpha = 0.3$ and $\beta = 0.0004$ and that appeared to work good for various situations. In future we plan to obtain ground truth data in order to investigate the precision of the algorithm and the influence of the parameters α and β. For the moment the results were checked only visually by backprojecting the 3D mesh head model over the images. For bigger α the tracker relied too much on the new measurements and tended to float away sooner. The parameter β describes how much the appearance can change. Too small β (big changes possible) allows the model to float away with time. At least for the initial pose (initial image) we would like to have the neighborhood defined by β small enough that the model can not float away. This can then be used as a criterion for choosing an appropriate β.

7.9 Conclusions

A real-time 3D head tracking algorithm is presented. A simple heuristic model is used to describe the appearance changes caused by movement in realistic light conditions. The algorithm was able to operate in various realistic conditions using cheap low-end equipment. Together with an automatic initialization procedure and reinitialization when the target is lost, the algorithm seems to be a promising solution for a number of applications. The algorithm heavily relies on the initial image. Therefore, small movements around the initial head pose were handled the best. However, for many human-computer interaction applications this would be exactly the way the system would be used.

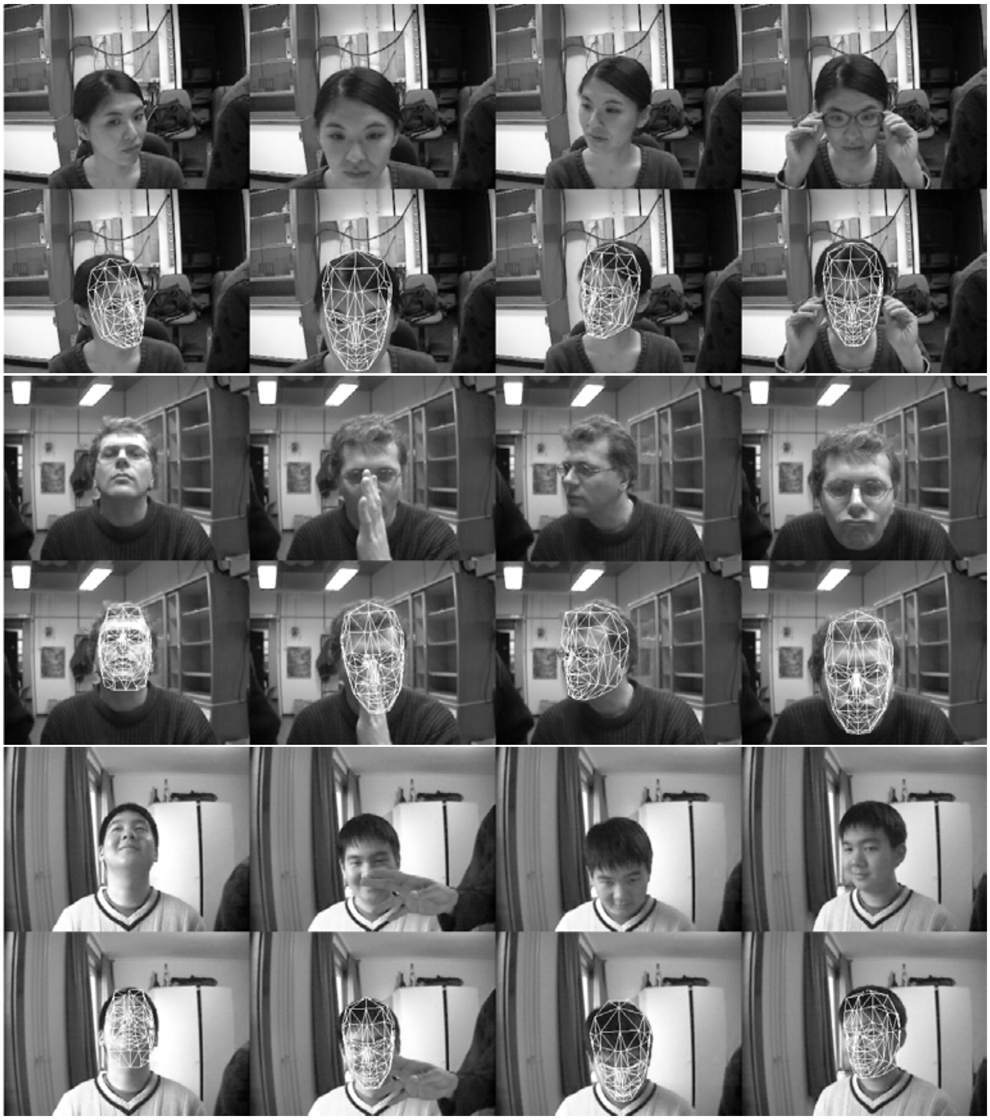


Figure 7.4: Real time tracking

Bibliography

[1] I. E. Arno Schodl, Antonio Haro. Head tracking using a textured polygonal model. *In Proceedings of Workshop on Perceptual User Interfaces*, November 1998.

[2] N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, 1993.

[3] S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head tracking. *In Proceedings of Intl. Conf. on Pattern Recognition*, 1996.

[4] M. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Comput. Vis. Image Understanding*, 63(1):75–104, 1996.

[5] M. Black, D. J. Fleet, and Y.Yacoob. Robustly estimating changes in image appearance. *Comput. Vis. Image Understanding*, 78(1):8–31, 2000.

[6] P. Bui-Thong. Illumination for computergenerated pictures. *Communications of the ACM*, 18(6), 1975.

[7] M. L. Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. *IEEE Transactions on PAMI*, 22(4):322–336, 2000.

[8] D. DeCarlo and D. Metaxas. Deformable modes-based shape and motion analysis from images using motion residual error. *In Proceedings of ICCV*, pages 113–119, 1998.

[9] P. Hallinan. A low-dimensional lighting representation of human faces for arbitrary lighting conditions. *In Proceedings of Computer Vision and Pattern Recognition*, pages 995–999, 1994.

[10] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[11] T. Jebara and A. Pentland. Parametrized structure from motion for 3d adaptive feedback tracking of faces. *In Proceedings of Computer Vision and Pattern Recognition*, 1997.

[12] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features *In Proceedings of Computer Vision and Pattern Recognition*, 2001.

[13] C. Tomasi and J. Shi. Good features to track. *In Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[14] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[15] Y. Zhang and C. Kambhamettu. Robust 3d head tracking under partial occlusion. *In Proceedings of Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

Chapter 8

Conclusions

This chapter brings some final conclusions and some recommendations for further research. Also some personal views and worries are presented.

8.1 Conclusions and recommendations

A number of important basic computer vision tasks are analyzed in this thesis. Some improvements are proposed and a number of practical applications is addressed. The focus was on the vivid area of the current computer vision research called 'looking at people' [4] and the related applications. Each of the chapters of the thesis has its own set of conclusions. Some final remarks are listed here.

The first part of the thesis (chapters 2 and 3) analyzed the problem of recursive probability density function estimation. The finite mixtures as a common flexible probabilistic model were used to model the incoming data on-line. The proposed simple recursive algorithms seem to be able to efficiently solve very difficult problems. Being able to automatically obtain a compact statistical model of the data is of great importance for the development of the intelligent systems. There are many possibilities for extending the work from this part of the thesis. First, the result could be extended to other hierarchical models (e.g. Hidden Markov Models). Furthermore, there is a huge number of possible applications of the algorithm (for the moment only the problem of background scene modelling is analyzed in chapter 4). Note that the algorithms are based on a number of approximations and the accuracy of the results can not be expected to be better than the standard well-established and more elaborate methods. However, the presented fast recursive solution can be essential for many real-time working systems. If better accuracy is required the algorithms could be still important to generate a reasonable starting point for further refinement.

The second part of the thesis (chapters 4 and 5) studied the foreground/ background segmentation, a standard problem in many automatic scene analysis applications. Efficient solutions are proposed and analyzed for the standard pixel-based foreground/background segmentation. Furthermore, the analysis and the solutions for two practical problems are presented. The results are of interest for the development of the monitoring/surveillance systems.

The third part of the thesis (chapters 6 and 7) is about image motion and object tracking, also standard problems in the automatic scene analysis applications. The object tracking is in practice realized by some local search algorithm. Chapter 6 points out the importance of 'the region of convergence' of the local search for the tracking problem. If the convergence region is large we can expect robust results. Small convergence region is an indication of possible unreliable tracking results. The principle is generally applicable but it might be hard to estimate the convergence region. A simple solution is presented for the simple but important problem of feature point tracking.

Because of the high complexity, the models that are used in computer vision are very often learned from the data rather than hand generated. However, for many practical problems it seems that the human knowledge and the hand generated models are indispensable. As an illustration of these two paradigms, consider how one might represent a dancer. One extreme would be to construct an articulated 3D model with texture-mapped limbs and hand-specified degrees of freedom. On the other hand, we could use a large number of images of the dancer in M different poses from N vantage points, and some general statistical model could be learned from this data. What are the inherent strengths and weaknesses of these two approaches and how to combine them are the important issues of the computer vision research of today. Throughout this thesis the choices are made and the human generated and learned models are combined on different processing levels. Two examples are the solutions from chapter 5 for the two applications : traffic monitoring and tennis game analysis. Chapter 7 analyzed some advantages of the simple human generated models and proposed an object tracking method. The tone in this chapter was in favor of the human generated models. On the other hand, the first part of the thesis proposed an efficient on-line learning algorithm which might be applied for many purposes and could make the hand-generated models less needed. The following joke is related to this situation:

"A mathematician, a physicist, and a statistician are shooting on a target using only one gun. The mathematician quickly calculates the distance, the velocity, the angle, etc. and shoots. Well, he misses by an inch to the LEFT! The physicist takes the gun. The physicist also takes into consideration the gravity, air friction, and such things... and fires! Well, he misses by an inch to the RIGHT! The statistician takes the gun, sets the gun position somewhere in between the previous two attempts, fires and HITS THE TARGET!"

Although the statistician seems the smartest of the three, if there were not the previous two close approximate shots, it would probably have been a long time shooting around until he could hit the target.

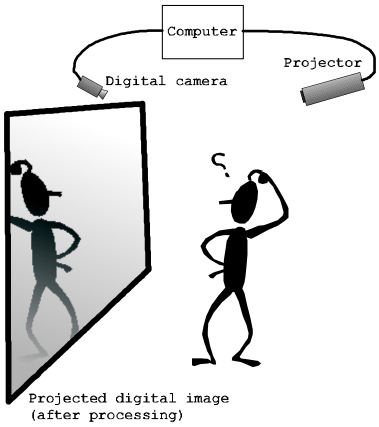
8.2 Some general open issues and recommendations

The general goal of the computer vision research is to make the computer aware of its environment. There are many application that could help people in various ways. However, when many such perceptually aware machines, that can be present everywhere, are connected together, we get a possibility to concentrate information about people and a situation which closely resembles George Orwell's dark vision of a government that can monitor and control your every move [3]. Every computer scientist should be aware of this and the related issues and the possible applications of the developed techniques. Some of the results from this thesis were used to design two interactive demos. The demos can be shown on computer screen or using a projector as presented in figure 8.2a. It is simple to create an illusion of a mirror in this way and provide natural interaction. Such installations are also known as 'camera-projector' systems. For some more elaborate technical solutions see [2, 5].

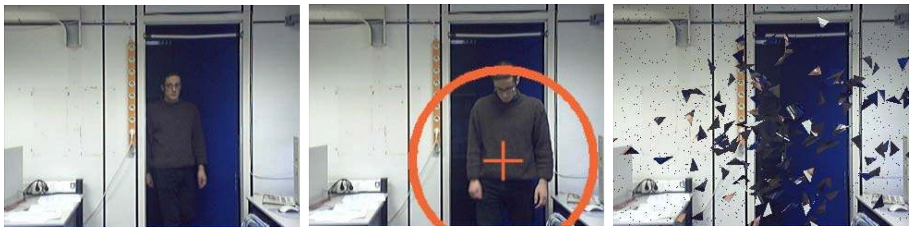
The first demo is using the techniques from the part two of this thesis to detect and track people and 'shoot' on them, figure 8.2b. Different scenarios are possible. For example the goal of the game could be to perform some tasks (move from one side to the other, seize some objects etc.) in front of the camera moving fast and hiding from the camera to avoid being spotted. A standard surveillance algorithm is presented in an amusing way but possible misusages of such system are made very clear.

The second demo is related to the third part of this thesis. Motion is used to play a shape sorting game. Additional image transformations are performed as awards or when the game is over, figure 8.2c. The game is a tribute to the Lucas-Kanade motion estimation technique [1] and the 'peg-in-hole' problems from computer vision. The human motion controls the game but the computer also provokes the human motion and in a way also controls it.

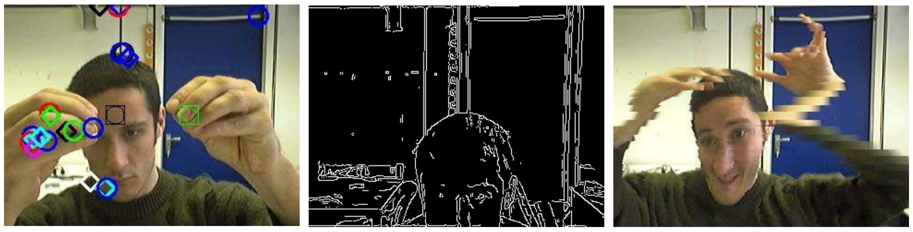
The two games were presented to many people. The result was many smiling faces. Hopefully, the future computer vision research will aim mostly at achiving similar results.



a) simple virtual mirror



intruder detection and tracking intruder destroyed
b) object detection and tracking game



shape sorting game over transformation as award
c) shape sorting game based on optical flow

Bibliography

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework part 1: The quantity approximated,the warp update rule, and the gradient descent approximation. *International Journal of Computer Vision*, 2003.

[2] T. Darrell, G.Gordon, M.Harville, J.Woodfill, H. Baker, and A.Hertzmann. Magic morphin mirror/ mass hallucinations. *An interactive art and technology installation, The Tech Museum of Innovation and other sites (1997-1999).*

[3] G. Orwell. *Nineteen Eighty-Four.* Signet Classic (original 1945), 1990.

[4] A. Pentland. Looking at people: Sensing for ubiquitous and wearable computiong. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 2000.

[5] R. Yang and Z. Zhang. Eye gaze correction with stereovision for video tele-conferencing. *In Proc. 7th European Conference on Computer Vision (ECCV2002)*, pages 479–494, 2002.

Summary

"Motion Detection and Object Tracking in Image Sequences",
Zoran Živković, Ph.D. thesis, University of Twente, Enschede, June 5, 2003.

The visual information is essential for humans. Therefore, among many different possible sensors, the digital video cameras seem very important for making machines that are aware of their environment and are able to act intelligently. This thesis is related to the subjects: computer vision and artificial intelligence. The focus in this thesis is on a number of basic operations that are important for many computer vision tasks. The thesis is divided into three parts: statistical modeling, motion detection and motion measurements. Each part corresponds to one of the basic tasks that were considered. From the huge number of possibilities the attention here is on the applications usually named 'looking at people' where the goal is to detect, track and more generally to interpret human behavior.

One of the reasons for the early success of the computer vision systems is in proper application of the well-established pattern recognition and statistics techniques. An intelligent system should be able to constantly adapt and learn from the data it gathers from the environment. The first part of this thesis analyzes the problem of recursive (on-line) probability density estimation. The 'finite mixtures' and in particular on the mixtures of Gaussian distributions are used. An algorithm is proposed that estimates the parameters and the appropriate number of components in a mixture simultaneously. The new algorithm is also less sensitive to the initial parameter values. A modified version of the algorithm that can adapt to changes in data statistics is also presented. The algorithm can be essential for many real-time systems to quickly get an up to date compact statistical model for the data.

The scene analysis often starts with segmenting the foreground object from the background. This basic image sequence processing step is the topic of the second part of this thesis. If the statistical model of the scene is available the foreground objects are detected by spotting the parts of the image that don't fit the scene model. The main problem is updating and adapting the scene

model. Two efficient pixel-based foreground/background segmentation algorithms are presented. Furthermore, two practical applications are analyzed: a traffic monitoring application and automatic analysis of the tennis game matches. The results could be of interest to many professionals in the field including archivists, broadcasters and the law enforcement sector. Although very specific, the two applications have many elements that are important for any surveillance/monitoring system.

Beside detecting the objects, tracking the objects is another basic scene analysis task. The third part of the thesis considers first the related basic image motion problem. The image motion or the 'optical flow' can be defined as the movement of the image patterns in an image sequence. A common problem is how to choose points in an image where the image movement can be calculated reliably. There are some standard procedures to select the suitable points. Most of them are more concerned with the accuracy, rather than with robustness of the results. A way for estimating the 'region of convergence' for the image points is proposed. The size of the 'region of convergence' can be used as a measure of feature point robustness. Further in the third part of the thesis, a simple heuristic for efficient object tracking based on the template matching is presented. The attention is on face tracking but the results are generally applicable. In a tracking scheme an object is first detected and then tracked. We use a simple generic 3D model to describe the transformations between the initial object appearance and the subsequent images. However there are many deformations that not described by this model. We propose a simple generally applicable heuristic that updates the initial object appearance with the new images.

Samenvatting

"Het Detecteren van Beweging en Volgen van Objecten in Beeldreeksen", Zoran Živković, proefschrift, Universiteit Twente, Enschede, 5 juni, 2003.

De visuele informatie is essentieel voor mensen. Daarom, te midden van alle mogelijke sensoren, zijn de digitale video camera's belangrijk voor het maken van intelligente en omgevingbewuste machines. Het onderwerp van dit proefschrift hoort tot het computer visie en artificiële intelligentie onderzoek. Een aantal van de belangrijke basisoperaties is behandeld. Het proefschrift heeft drie onderdelen: statistische modellering, het detecteren van beweging en het meten van beweging. Elke onderdeel van het proefschrift behandelt een van de basisoperaties. Verder wordt er gefocust op de applicaties genoemd 'looking at people', waar het doel is om mensen te detecteren, volgen en uiteindelijk hun gedrag te interpreteren.

Een reden voor het succes van computer visie systemen is de toepassing van patroonherkenning en statistische methoden. Een intelligent systeem zou constant moeten kunnen leren en zich aanpassen aan de hand van de informatie vanuit zijn omgeving. Het eerst deel van dit proefschrift behandelt het probleem van het recursief (on-line) schatten van de kansdichtheid functies. De 'finite mixtures' en Gaussische distributies zijn gebruikt. Een algoritme is ontwikkeld om de parameters en het aantal componenten in een mixture tegelijk te schatten. Het algoritme is ook minder gevoelig voor initialisatie. Verder is er ook een modificatie van het nieuwe algoritme geanalyseerd om on-line aan de veranderingen in de data statistiek aan te kunnen passen. Het algoritme kan heel nuttig zijn voor alle real-time systemen om snel een compact statistisch model van de data te maken.

De analyse van de scène begint vaak met segmentatie van het beeld in voorgrond objecten en achtergrond. Deze segmentatie is het onderwerp van het tweede onderdeel van dit proefschrift. Als er een statistisch model van de scène beschikbaar is, worden de voorgrond objecten gedetecteerd zijnde die delen van het beeld die niet aan het model van de scène voldoen. Een belangrijk punt is hoe zo'n model te maken en hoe het daarna aan te passen. Twee efficiënte algoritmen zijn voorgesteld. Verder worden er twee praktische

toepassingen behandeld: een automatisch verkeer analyse systeem en een systeem voor het automatisch analyseren van video's van tenniswedstrijden. De resultaten kunnen van belang zijn voor sectoren zoals justitie en de omroep. De twee applicaties zijn specifiek, maar toch bevatten ze een aantal elementen die voorkomen in elk surveillance systeem.

Behalve detecteren is het volgen van de objecten een andere belangrijke operatie voor de automatische analyse van beeldreeksen. Het derde onderdeel van dit proefschrift behandelt eerst het hieraan gerelateerde basisonderwerp aangaande het berekenen van beeldbewegingen. De beeldbeweging of 'optical flow' is gedefinieerd als de beweging van de beeldpatronen in een beeldreeks. Het vinden van de punten in een beeld die geschikt zijn om beweging goed te kunnen schatten, is een standaard probleem. De standaard procedures houden de nauwkeurigheid van het resultaat in de gaten, maar er wordt geen rekening gehouden met de robuustheid van de resultaten. Er is een nieuwe methode voorgesteld om het convergentiegebied voor de beeldpunten te schatten. De grootte van het convergentiegebied kan als maat voor de robuustheid worden gebruikt. Verder brengt het derde onderdeel van het proefschrift een simpele efficiënte heuristiek om objecten te volgen op basis van 'template matching'. De aandacht is gericht op het volgen van menselijke gezichten, maar de resultaten zijn ook toepasbaar voor andere situaties. Een object wordt eerst gedetecteerd en dan gevolgd. Een 3D model wordt gebruikt om de transformatie als gevolg van de beweging te modelleren. Toch zijn er veel van de transformaties die niet goed worden beschreven door dit model. Er is een eenvoudige heuristiek voorgesteld om het initiële radiometrische model van het object te updaten met de informatie vanuit de nieuwe beelden.

Related publications

Refereed journal papers

- Z.Zivkovic, F.van der Heijden, "Adaptive density estimation for foreground/background segmentation", submitted.
- Z.Zivkovic, F.van der Heijden, "Adaptive recursive unsupervised learning of finite mixture models ", to be submitted.
- Z.Zivkovic, F.van der Heijden, "Recursive unsupervised learning of finite mixture models ", submitted.
- Z.Zivkovic, F.van der Heijden, "Improving the selection of feature points for tracking", submitted.
- Z.Zivkovic, A.Schoute, F.van der Heijden, "Updating the global position of an autonomous vehicle by combining a known environment map and sensor data", submitted.

Refereed conference papers

- Z.Zivkovic, M.Petkovic, R.van Mierlo, M.van Keulen, F.van der Heijden, W.Jonker, E. Rijnierse, "Two video analysis applications using foreground/background segmentation", IEE Visual Information Engineering Conference, UK, 2003.
- Z.Zivkovic, F.van der Heijden, "Better Features to Track by Estimating the Tracking Convergence Region", ICPR, Canada, 2002.
- Z.Zivkovic, A.Schoute,F.van der Heijden, "Combining a priori knowledge and sensor information for updating the global position of an autonomous vehicle", The 8th Mechatronics Forum International Conference, Netherlands, 2002.
- Z.Zivkovic, F.van der Heijden, "A stabilized adaptive appearance changes model for 3D head tracking", ICCV'01: Second International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems,Vancouver, Canada, 2001.

- M.Petkovic, Z.Zivkovic, W.Jonker, "Recognizing Strokes in Tennis Videos Using Hidden Markov Models", IASTED International Conference Visualization, Imaging and Image Processing, Marbella, Spain, 2001.
- Z.Zivkovic, F.van der Heijden, M.Petkovic, W.Jonker, "Image processing and feature extraction for recognizing strokes in tennis game videos", 7th Annual Conference of the Advanced School for Computing and Imaging (ASCI), the Netherlands, 2001.
- Z.Zivkovic, F.van der Heijden, Z.Houkes, "A finite element model for articulated object tracking - human motion capture", 6th Annual ASCI Conference, Belgium, 2000.

Abstract based oral presentations

- Z.Zivkovic, "Adaptive recursive probability density estimation for background modeling ", The Dutch Society for Pattern Recognition and Image Processing (NVPHBV) - Spring meeting, 2003.
- Z.Zivkovic, "Adaptive recursive learning of finite mixture models", NVPHBV - Fall meeting, 2002.
- Z.Zivkovic, "Video tracking", Invited speech, TNO Physics and Electronics Laboratory, Den Haag, 2002.
- Z.Zivkovic, "Real-time 3D head tracking", NVPHBV - Spring Meeting, 2001, similar talks at: ASCI 2001 conference, Mechatronica cotactgroepdag 2002, etc.

Demos

- Z.Zivkovic, "Virtual mirrors - Shape sorting game and shooting game", Presented at: IEEE ICME 2003, University of Twente open-day and Minor-markt, after NHPHBV Fall meeting 2002 etc.
- R.van Mierlo and Z.Zivkovic, "A traffic monitoring system", Presented at: ICT knowledge congress 2002, Mechatronics Forum International Conference 2002, SUMMER workshop etc.
- Z.Zivkovic, "Real-time 3D head tracking", Presented at: ICCV 2001, NVPHBV- Spring Meeting 2001, ASCI conference 2001, Mechatronica cotactgroepdag 2002, etc.