ABSTRACT

Title of dissertation: ADAPTIVE KERNEL DENSITY APPROXIMATION AND ITS APPLICATIONS TO REAL-TIME COMPUTER VISION

Bohyung Han, Doctor of Philosophy, 2005

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

Density-based modeling of visual features is very common in computer vision research due to the uncertainty of observed data; so accurate and simple density representation is essential to improve the quality of overall systems. Even though various methods, either parametric or non-parametric, are proposed for density modeling, there is a significant trade-off between flexibility and computational complexity. Therefore, a new compact and flexible density representation is necessary, and the dissertation provides a solution to alleviate the problems as follows.

First, we describe a compact and flexible representation of probability density functions using a mixture of Gaussians which is called Kernel Density Approximation (KDA). In this framework, the number of Gaussians components as well as the weight, mean, and covariance of each Gaussian component are determined automatically by mean-shift mode-finding procedure and curvature fitting. An original density function estimated by kernel density estimation is simplified into a compact mixture of Gaussians by the proposed method; memory requirements are dramatically reduced while incurring only a small amount of error. In order to adapt to

variations of visual features, sequential kernel density approximation is proposed in which a sequential update of the density function is performed in linear time.

Second, kernel density approximation is incorporated into a Bayesian filtering framework, and we design a Kernel-based Bayesian Filter (KBF). Particle filters have inherent limitations such as *degeneracy* or *loss of diversity* which are mainly caused by sampling from discrete proposal distribution. In kernel-based Bayesian filtering, every relevant probability density function is continuous and the posterior is simplified by kernel density approximation so as to propagate a compact form of the density function from step to step. Since the proposal distribution is continuous in this framework, the problems in conventional particle filters are alleviated.

The sequential kernel density approximation technique is naturally applied to background modeling, and target appearance modeling for object tracking. Also, the kernel-based Bayesian filtering framework is applied to object tracking, which shows improved performance with a smaller number of samples. We demonstrate the performance of kernel density approximation and its application through various simulations and experiments with real videos.

ADAPTIVE KERNEL DENSITY APPROXIMATION AND ITS APPLICATIONS TO REAL-TIME COMPUTER VISION

by

Bohyung Han

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Commmittee:

Professor Larry Davis, Chair/Advisor
Professor Rama Chellappa
Professor Davis Jacobs
Professor Ramani Duraiswami
Professor Sung Lee

DEDICATION

To my wife and my parents.

ACKNOWLEDGMENTS

There are many people I should acknowledge for the completion of the dissertation.

Most of all, I thank my advisor, Prof. Larry S. Davis, for his support, patience and encouragement throughout my PhD study. Without his excellent supervision and valuable advice for my research, I could not have achieved my success in my PhD. I am also grateful to my thesis committee members for their time and comments.

During my internship in Siemens, I have got very good colleagues and friends. The discussion with Dr. Dorin Comaniciu was very helpful to my research, and I appreciate his brilliant idea and encouragement. Also, I would like to thank Ying for her support and advice, and the collaboration with her was a good opportunity to me. I cannot forget the hospitality of friends in Siemens: Anurag, Xean, and Kazunori. Also, special thanks to all my friends in our lab including Ser-Nam, Harsh, Mohammed, Changjiang, Shiv, Vinay, Son, Abhinav, and so on.

For the last five years in Maryland, I was so lucky to meet many good people and had a great time with them. I have spent a lot of time with KOSTAM members, and share good memory with them. I am afraid that I cannot list all the members, but I should not forget to thank Jungjun, Sukjoon, Hyoungsoo, Hyowook, Jongwon, Sungjoon, Hwansoo, Youngsoo, Hyuncheol, Jihoon, Choonghoon,

Hanny and their families. Also, I would like to thank many Korean friends in CS department: Kyongil, Bongwon, Hyunmo, Jihwang, Minho, Seungjoon, Jaeyong, Beomseok, Yoo-Ah, Jaeyoon, Seong-Wook, Kyungnam, Minkyoung and so on.

Finally, I really thank my parents and my grandmother for their love and guidance throughout my entire life. Also, this dissertation would not have been possible without the love and patience of my wife, Kyoungsun.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Recently, the development of computer hardwares and camera devices enable us to acquire and collect a large volume of video data. A substantial amount of resources such as CPU, memory and storage is generally required to handle such data, and efficient algorithms to facilitate processing are crucial. Even though various techniques have been developed and used in the wide range of computer vision problems, many traditional techniques are designed for batch processing and controlled environment. However, a variety of applications such as visual surveillance, driving assistant systems, vision-based interfaces, virtual reality, and image-based diagnosis require adaptive control to cope with unexpected variations and achieve real-time capability. Therefore, adaptive real-time algorithms for video data are becoming more important.

Typically, in a real-time environment, all the data are not available before the processing; they are obtained in a sequential manner. Also, it is not usually practical to store all previous data for later use. Instead, some kind of abstract methodology is employed to represent the history of data. Therefore, a powerful representation for data up to the current time step is required; it should be flexible enough to be updated for new data.

In the dissertation, we describe an on-line Kernel Density Approximation (KDA) algorithm which provides a compact and flexible methodology to represent and update a probability density function, and discuss a wide range of applications within computer vision area. Object detection and tracking, background modeling and subtraction, human motion analysis, and dynamic scene modeling are typical examples with real-time constraints, and we illustrate how our adaptive kernel density approximation provides a good theoretical and practical model for those applications.

1.1 Motivation

Visual features such as intensity, color, gradient, texture and motion are commonly modeled using probability density functions. Moreover, a number of real-time tasks such as background subtraction and modeling the appearance of a moving target require sequential density estimation, where new data is incorporated in the model as it becomes available. Nevertheless, current methods either lack flexibility by fixing the number of Gaussians in the mixture, or require large memory amounts by maintaining a non-parametric representation of the density. The underlying probability density of the feature may be described using a parametric (e.g., one Gaussian), semi-parametric (mixture of Gaussians) or non-parametric (e.g., histogram or kernel density-based) representation. These representations create a trade-off between the flexibility of the model and its data summarization property. For example, parametric and semi-parametric model are simple and easy to handle, but are unable or have difficulty to represent multi-modal density effectively. On the other

hand, non-parametric models are very flexible and can accommodate arbitrarily complex densities, but require a large amount of memory and huge computational cost for their implementation, especially for high-dimensional problems. Therefore, we need a new strategy to overcome the limitation of conventional parametric and non-parametric techniques. The following properties are required for a "good" representation.

- **Accuracy:** The density function should be constructed so that it simulates the true density function with incurring minimal error.
- **Flexibility:** An arbitrary density function should be representable without strong assumptions about the true density.
- **Compactness:** A simple density representation method is preferred to save memory space and computation time.
- **Adaptiveness:** For the sequential update of density function, the representation should provide a methodology to efficiently update the density function in an on-line manner given new data.

Kernel density approximation is the new technique we develop to approximate the density function constructed by kernel density estimation with a mixture of Gaussians. In this method, a single Gaussian kernel is assigned to each mode location automatically detected by mean-shift, and the covariance matrix of each Gaussian is estimated by curvature fitting around the mode. The accuracy of this technique

is comparable to kernel density estimation[1], and the density representation is very compact since it uses a small number of Gaussian components. Also, it provides a nice framework for sequential density estimation, so is suitable to be applied to real-time applications. The proposed density representation is memory efficient which is typical for mixture densities, and it inherits the flexibility of non-parametric methods by allowing the number of modes to be adaptive.

In this dissertation, we show how kernel density approximation can be applied effectively and efficiently to various real-time computer vision tasks such as background modeling and subtraction, target appearance modeling, and object tracking. The following sections discuss how density functions can be used in computer vision research and how current techniques can be improved by incorporating kernel density approximation.

1.1.1 Modeling of Visual Features

A Gaussian distribution has been widely used in computer vision applications, but it cannot model multi-modal distributions which are frequently observed in data for visual features. A mixture of Gaussians is a well known approach to describe a multi-modal density function, but it needs a pre-defined number of components as a parameter, which is very difficult to determine before observation. An alternative model, kernel density estimation, has the flexibility to describe the density function

[1]Since kernel density approximation estimates the covariance matrix using the curvature of mode location, the accuracy at the point far from every mode is relatively worse. Also, the approximation error increases when the underlying density function is not based on a Gaussian mixture.

with an arbitrary number of modes, but it suffers from huge memory requirements, especially for high dimensional data.

On the other hand, kernel density approximation is able to preserve all modes found in kernel density estimate, and has a significant advantage in the memory requirement since only a single Gaussian is assigned to each mode. Figure 1.1 illustrates the comparison of densities constructed by kernel density estimation, kernel density approximation, and an EM algorithm. The original density function is created as a mixture of Gaussians, and 200 samples are drawn from it. We reconstructed density functions using those three methods; three different parameters for the number of components are also tested for the EM algorithm. As seen in Figure 1.1, the density constructed by kernel density approximation is very close to the kernel density estimate as well as the original density. Even though the EM algorithm [16] is widely used for the implementation of a Gaussian mixture, this result shows that its performance is sensitive to number of components as well as the initial location of each component.

1.1.2 Modeling in State Space

Bayesian filtering [35, 36, 46] is a very popular framework for the state estimation in dynamic systems. Among many variations of Bayesian filtering methods, the particle filter [19, 31] is widely used in computer vision due to its robustness and simplicity. In the particle filter framework, the density function is represented with a set of samples and their weights, but it is difficult to describe the underlying density function precisely with only a limited number of samples. Moreover, it

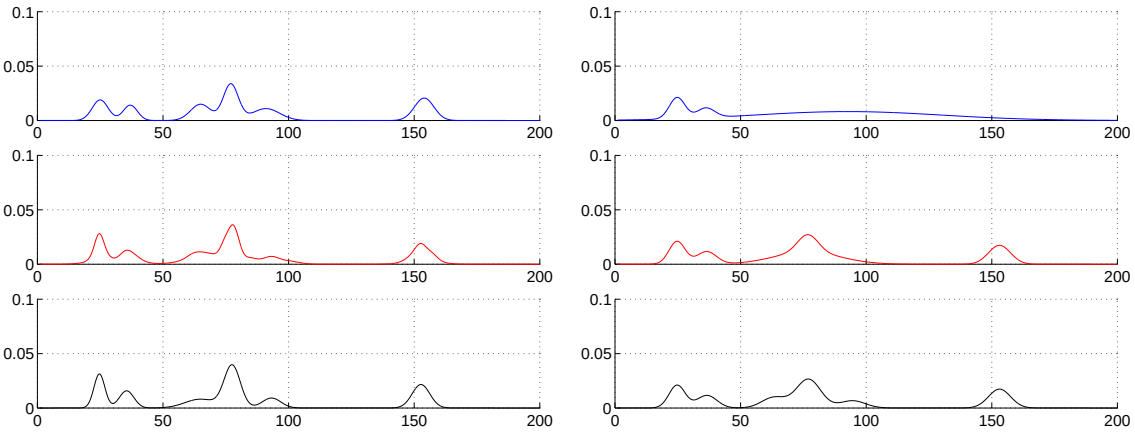


Figure 1.1: Comparison between kernel density approximation and other methods. (a) original density function (top) kernel density estimation (middle) kernel density approximation (bottom) (b) EM algorithm with 3, 6, and 9 components

often experiences sample depletion caused by the concentration of highly weighted samples in small areas, which is called the *degeneracy* problem [2, 18]. Because of this problem, particle filters generally need many samples or require occasional resampling steps. However, resampling frequently suffers from the *loss of diversity* problem, which also causes the waste of particles.

To overcome this problem, we employ a Gaussian mixture for every relevant density function in a Bayesian filtering. There have been several attempts to implement a Bayesian filtering for non-linear and non-Gaussian systems based on continuous density functions — usually Gaussian mixtures, but they could not provide any principled methodology to avoid the exponential increase of components in the mixture. In our method, kernel density approximation plays an important role to propagate the compact posterior density function to the next time step. A Bayesian

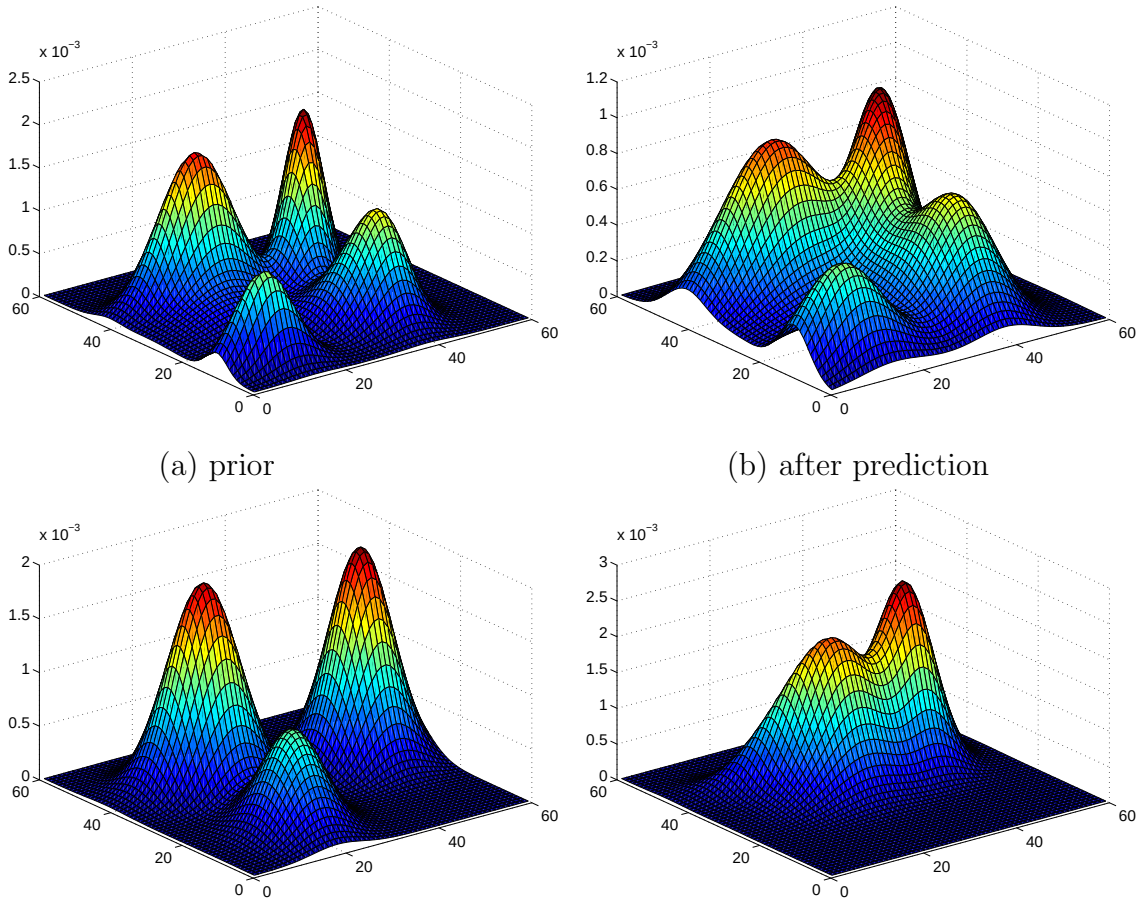
filter based on continuous density functions, which is referred to as a kernel-based Bayesian filter, has the following advantages.

- In a high dimensional state space, kernel density approximation is more efficient and effective in its ability to simulate true density.
- The number of particles for robust performance is reduced by employing kernel-based particles and continuous measurement function.

One-step kernel-based Bayesian filtering is illustrated in Figure 1.2 which shows the density modeling in the state space and its propagation to the next time step.

1.1.3 Sequential Model Update

A special class of learning techniques are needed for real-time processing tasks, where the data is included in the model as it becomes available in a recursive fashion. The updating strategy should accommodate gradual variations, while maintaining a memory efficient representation of the model. For these scenarios, batch processing is not a solution due to the induced processing delay and large amounts of data that need to be stored. Typical examples for this case are target model update for object tracking and adaptive background modeling. As illustrated in Figure 1.3, the initially constructed model for object tracking can quickly become invalid, because of the arbitrary transformation of object or changes of the environment. Therefore, for accurate and reliable modeling, sequential density update is a reasonable solution. For this purpose, a Gaussian mixture model [34, 73] can be employed, but still has the flexibility problem especially in real-time applications. The sequential version of



(a) prior

(b) after prediction

(c) observation

(d) posterior

Figure 1.2: Density functions in one-step of kernel-based Bayesian filter

kernel density estimation, which we call sequential kernel density estimation later, can be utilized, but the addition of new kernels in the current density function is not feasible due to the large memory requirement and tremendous processing time. These problems are alleviated by integrating a sliding window method [23], but it is not satisfactory solution and only has limited ability for maintaining an accurate modeling over time. On the other hand, kernel density approximation provides a convenient methodology to propagate the density over time, and we describe the use of this technique for sequential model update problems.

(a) Pose and lighting changes (frame 1 and 200)

(b) Severe scale changes (frame 1 and 400)

Figure 1.3: The requirement of adaptive modeling for object tracking.

1.2 Related Work

The simplest density-based modeling methods – a single Gaussian distribution – is incorporated for background subtraction and object tracking [83] and the Gaussian is updated sequentially [25, 52]. However, this method cannot handle multi-modal density functions effectively so the accuracy of modeling is severely damaged in real examples.

Mixture models with a fixed number of components have been utilized in various applications as follows. In [45], the target appearance model based on color is constructed by a mixture of Gaussians which is replaced in each frame. The same

density representation is employed for optical flow estimation by Jepson and Black [33]. Stauffer and Grimson [74] proposed a recursive update of Gaussian mixture model for background modeling, A more elaborate target model is described in [34], where a 3-component mixture for the stable process, the outlier data, and the wandering term is designed to capture rapid temporal variations in the model. This appearance model is adapted over time by an on-line EM algorithm.

The EM algorithm [16, 62] is the main statistical tool for learning density models from incomplete data. Although variants of incremental EM have been employed in practice to deal with real-time adaptation constraints, only recently have they received formal justification and many issues are still open [50]. Most of all, it is generally difficult to add or remove components in the EM framework in a principled way [61]. Therefore, most real-time applications rely on models with a fixed number of mixtures or apply ad-hoc strategies to adapt the number of mixtures in time.

However, all the methods presented above require knowledge of the number of components, which is generally unknown. Especially, it is not appropriate to estimate density with a fixed number of components if there are a large number of modes in the underlying probability density function or the number of modes changes frequently.

On the other hand, Cham and Rehg [5] introduce a piecewise Gaussian representation to specify the modes of the density characterizing a tracker state. The density at a certain location is determined by the Gaussian component providing the largest contribution at the location. This idea is applied to multiple hypothesis

tracking in a high dimensional space body tracker within the particle filter framework, but the sampling and the posterior computation are not straightforward and provide only a crude approximation to the density.

Kernel density estimation [22] is one of the most popular non-parametric techniques to model the density for various applications, because it is very flexible to represent multi-modal densities. But, its very high memory requirements and computational complexity inhibit the use of this method in real-time applications, even though there have been several attempts to improve the computational cost [21, 84].

1.3 Contribution Overview

1.3.1 Density Approximation

In contrast to previous approaches, this dissertation introduces a density approximation algorithm that is an alternative to kernel density estimation, but computationally as simple as parametric methods. We first detect local maxima (modes) based on variable-bandwidth mean-shift [12, 14]. The modes of a density function represent regions with higher local probability; hence, their preservation is important to maintain a low density approximation error. In the proposed algorithm, the density is represented with a weighted sum of Gaussians where each mode is represented with a single Gaussian. The number of Gaussians and the parameters for each Gaussian – weight, and mean – are automatically determined by mode finding procedure. Also, the covariance of each Gaussian is derived from curvature fitting using the Hessian matrix computed at the mode location. By this mode-based representation, the memory requirements are low, similar to all methods that

use mixture densities, but it is possible to add and delete Gaussian components in a principled way.

While the initial density approximation is performed based on a batch mode finding algorithm, where all the data should be processed at a time, we also describe a more efficient method – an incremental density approximation method. In addition, sequential kernel density approximation is proposed for on-line learning that relies on the modeling and propagation of density modes.

Density approximation and mode propagation are tested for the task of background modeling and subtraction, target appearance modeling, and object tracking.

1.3.2 Background Modeling and Subtraction

Background modeling and subtraction is one of the most critical steps for high-level vision applications. Most algorithms are based on pixel-wise density modeling even if there are some other approaches using codebook [38], eigenbackground [55, 77], subspace method [49], and global probability density function [71].

In order to model the variations caused by illumination changes, shadows and dynamic background (wave, rain, tree etc), density-based modeling methods are proposed varying from using a single Gaussian [30, 83], Gaussian mixture [73], to kernel density estimation [23, 48]. The pattern of visual features presented at each pixel is quite different, so the assignment of the same number of Gaussian components for each pixel either wastes memory resource or fails in modeling complicated backgrounds effectively. Since our kernel density approximation technique provides a methodology to determine the proper number of modes by mean-shift mode find-

ing procedure, the minimal number of Gaussian component is used for modeling a background by assigning a single Gaussian to each mode location. The estimated density function by the proposed method is very close to the kernel density estimate.

This strategy is utilized in a background modeling algorithm, and we show that the performance of our background subtraction is comparable to the results obtained by kernel density estimation, but using significantly less memory space.

1.3.3 Target Appearance Modeling for Object Tracking

Intensive research has been performed for the object tracking problem, but adaptive target appearance modeling is still a difficult problem. Most trackers employ a fixed appearance model [3, 6, 15, 21, 29, 54, 58], but this strategy is not appropriate for tracking objects over a long term or in conditions when the appearance of the target changes. Several methods to update the target appearance model are proposed: template update [44], histogram update [53], Bayesian filtering [37, 52], on-line EM algorithm [34, 86], incremental subspace update [43], and the adaptation by background/foreground statistics [10, 51]. However, these methods are heuristic, or assume uni-modal or multi-modal appearances with fixed number of components, so they have critical limitations to represent variable multi-modal data. We adopt a pixel-wise density-based modeling in which a variable number of Gaussian components is assigned to each pixel; sequential kernel density approximation is employed to implement this. Originally, sequential kernel density approximation is a quadratic algorithm, but it is improved to a linear-time algorithm which is proved by amortized analysis. Also, Haar-like rectangular feature as well as RGB color (or intensity)

are integrated together for target modeling, and some spatial information around each pixel is encoded. Our target appearance modeling by sequential kernel density approximation for color and rectangular feature improves the overall performance of general trackers, and it is illustrated with various examples.

1.3.4 Bayesian Filtering and Object Tracking

There are two different classes of tracking algorithm; one is deterministic and the other is probabilistic. Deterministic trackers are generally fast but the tracked object may be lost once tracking fails temporarily. On the other hand, probabilistic trackers consumes more system resources, but tracking can be recovered from the failure by multiple hypotheses. Markov Chain Monte Carlo (MCMC) [18] is a method to propagate probability density function to the next step which is usually represented by samples and their weights in the state space. Many variations have been introduced for many practical applications; the CONDENSATION algorithm [31] is the most popular in computer vision.

The MCMC method provides a good solution for non-Gaussian/non-linear model, but it has the following disadvantages. First of all, it suffers from degeneracy problem or loss of diversity in particle locations. This fact causes the waste of particles and requires a large number of samples to obtain reliable performance. Second, even without the degeneracy and lost of diversity problem, a huge number of samples is required in high dimensional problems. In order to address these problems, several methods have been suggested including the improvement of sampling quality [17, 59, 76] or the implementation of the variation of particle filter

[18, 46, 66]. We propose a more fundamental methodology as a solution, which is called Kernel-based Bayesian Filtering (KBF). In kernel-based Bayesian filtering, all the relevant probability density functions are continuous Gaussian mixtures, and particles are based on Gaussian kernels. By this strategy, we reduce the number of samples for the same quality of performance as conventional particle filters since each kernel-based particle has a large coverage in the state space and degeneracy (or loss of diversity) problems can be alleviated by adopting continuous proposal distributions.

We demonstrate the performance of our kernel-based Bayesian filtering by various simulations and object tracking in real videos.

1.4 Organization of Dissertation

The dissertation is organized as follows. In Chapter 2, the formal definition and properties of kernel density estimation are described and the method to approximate the density function constructed by kernel density estimation is discussed. Also, the incremental kernel density approximation method is presented. Chapter 3 and 4 describe background modeling and subtraction, and target appearance modeling for object tracking by sequential kernel density estimation. The kernel-based Bayesian filtering framework is described in Chapter 5, and we present how this framework is applied to visual tracking problems.

Chapter 2

Density Approximation

2.1 Kernel Density Estimation

2.1.1 Introduction

Density estimation is a fundamental concept in statistics and broadly used in computer vision research. By definition, *density estimation* is the construction of an estimate of an unknown density function from observed data, in other words, samples. One approach to density estimation uses parametric methods, whose objective is to fit a known probability density model to the data. Since parametric density estimation relies on model specification, it requires prior knowledge of the underlying density function, which is often unknown. Another problem is the flexibility and accuracy of parametrically estimated density functions. On the other hand, non-parametric methods, such as kernel density estimation, can represent any arbitrary density function since the estimated density is mainly dependent on the structure of data. However, techniques to mitigate the complexity of kernel density estimation are necessary for real-time processing. In this chapter, we first describe the most general non-parametric method, kernel density estimation, and then our proposed method to simulate a density function given by kernel density estimation, *kernel density approximation*, is discussed.

2.1.2 Kernel Estimator

One of the most general non-parametric density estimation method is kernel density estimation [72]. For a 1-dimensional random variable x, a kernel function K which satisfies the condition

$$\int_{-\infty}^{\infty} K(x)dx = 1 \tag{2.1}$$

is employed for each sample, and the kernel density estimation with kernel K is defined by

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{2.2}$$

where h is the window width, also called the smoothing parameter or bandwidth.

Since kernel density estimator asymptotically converges to any density function [20, 68], this technique is applicable to many problems where the underlying density function is not known. A variety of kernel functions with different properties have been used in the literature. Typically kernel functions are symmetric and unimodal functions that fall off to zero rapidly away from the center. The size of this window depends on the bandwidth. Table 2.1 shows some of the commonly used kernel functions for univariate data.

2.1.3 Bandwidth Selection

One of the main issues in using kernel density estimation is the choice of the proper bandwidth. Small kernel bandwidth results in a ragged density, while wide bandwidth yields a smoother density function as shown in Figure 2.1.

While, using a fixed bandwidth for all samples reduce the complexity of the

Table 2.1: Some examples of kernel functions

Kernel function	Equation
Uniform	$U(-1,1)$
Triangle	$1-\|x\|$
Epanechinikov	$\frac{3}{4}(1-x^2)$
Quadratic	$\frac{15}{16}(1-x^2)$
Gaussian	$\frac{1}{2\pi}\exp\left(-\frac{1}{2}x^2\right)$
Cosinus	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}x\right)$

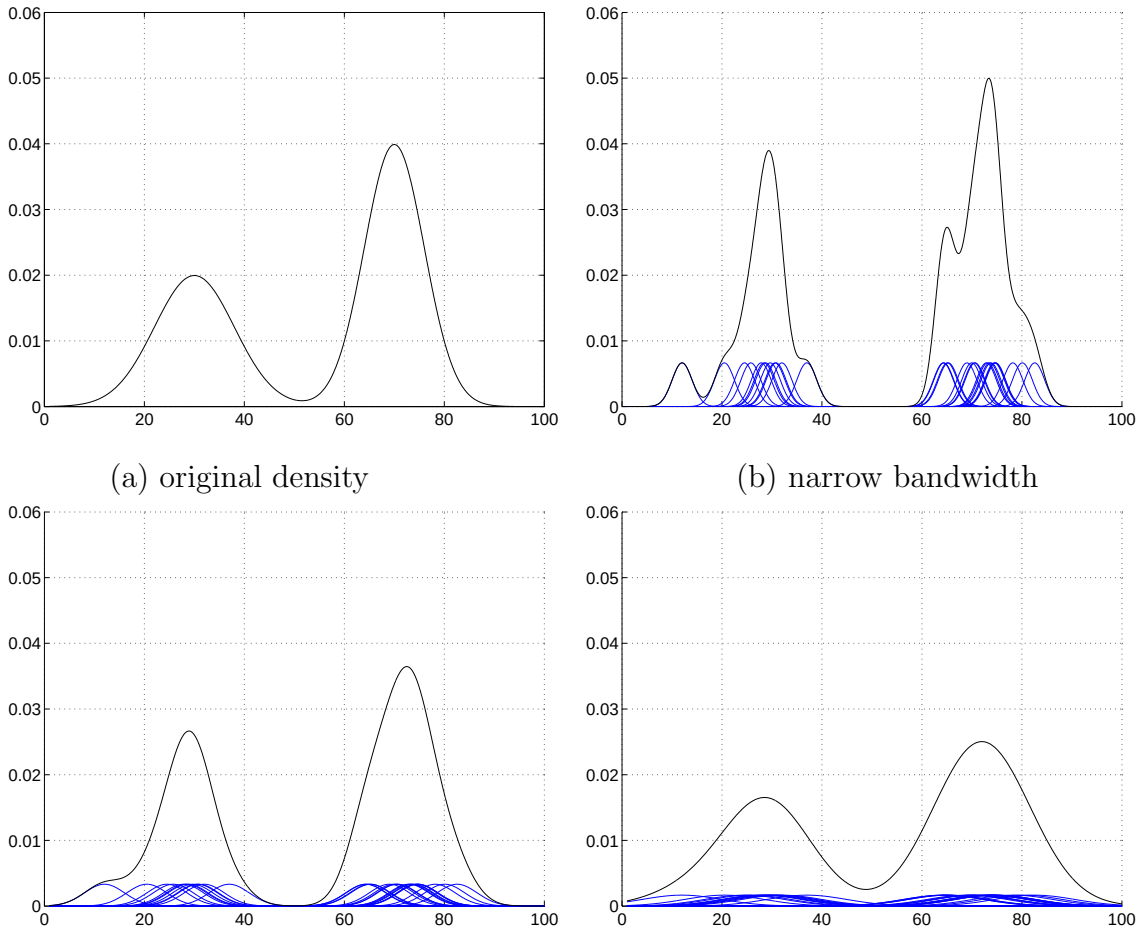
problem, this strategy causes spurious noise to appear in the tails of the estimates and degrades overall performance.

Obviously, the best choice of kernel bandwidth should depend on the number of samples, and it is reasonable that the bandwidth is inversely proportional to sample size. As the number of samples approaches infinity, the kernel should be as delta function and the bandwidth should approach zero.

For pointwise estimation, the classical measure of the similarity between the estimated density $\hat{f}$ and true density f is the Mean Squared Error (MSE), equal to the sum of the variance and squared bias

$$\begin{aligned} \text{MSE} &= E[\hat{f}(x) - f(x)]^2 \\ &= \text{Var}(\hat{f}(x)) + \text{Bias}(\hat{f}(x))^2 \end{aligned} \tag{2.3}$$

Using the Taylor theorem, the bias and the variance can be approximated [81], and the trade-off of bias versus variance is observed. The bias is proportional to h^2, which means that smaller bandwidth give a less biased estimator. On the other



(a) original density

(b) narrow bandwidth

(c) middle bandwidth

(d) wide bandwidth

Figure 2.1: Bandwidth selection and density estimates

hand, smaller h implies an increase in the variance which is proportional to $n^{-1}h^{-1}$. Thus, for a fixed bandwidth estimator, we should choose h that achieves an optimal compromise between the bias and variance over all x. Ideally, the optimal choice of kernel bandwidth can be achieved by minimizing the Mean Integrated Squared Error (MISE)

$$\text{MISE} = E\left(\int [\hat{f}(\mathbf{x}) - f(\mathbf{x})]^2 d\mathbf{x}\right) = \int E[\hat{f}(\mathbf{x}) - f(\mathbf{x})]^2 d\mathbf{x} \tag{2.4}$$

where $\hat{f}(\mathbf{x})$ and $f(\mathbf{x})$ are estimated density and true density, respectively. However,

the resulting bandwidth [72, 81] is of little practical use, since it depends on knowledge of the unknown true density function being estimated. Therefore, several data driven methods for bandwidth selection have been proposed, such as plug-in rule, least squares cross validation, and biased cross validation, but there is no generally good method to choose the optimal bandwidth.

2.2 Kernel Density Approximation

2.2.1 Mean-Shift Mode Finding

Suppose that we are given a density function by kernel density estimation based on Gaussian kernel. Denote by $\mathbf{x}_i$, $i = 1 \ldots n$ a set of points in R^d and assume that a symmetric positive definite $d \times d$ bandwidth matrix $\mathbf{P}_i$ is associated with each data point $\mathbf{x}_i$. The sample point density estimator with a d-variate normal kernel, computed at the point $\mathbf{x}$ is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}} \sum_{i=1}^{n} \frac{1}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \tag{2.5}$$

where

$$D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right) \equiv (\mathbf{x} - \mathbf{x}_i)^\top \mathbf{P}_i^{-1} (\mathbf{x} - \mathbf{x}_i) \tag{2.6}$$

is the Mahalanobis distance from $\mathbf{x}$ to $\mathbf{x}_i$. As one can see, the density at $\mathbf{x}$ is obtained as the *average of Gaussian densities* centered at each data point $\mathbf{x}_i$ and having the covariance $\mathbf{P}_i$.

In this section, we discuss an iterative procedure for mode detection based on the variable-bandwidth mean shift. Recall the density function $\hat{f}(\mathbf{x})$ made by kernel

density estimation where i-th kernel has the weight κ_i.

$$\hat{f}(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}} \sum_{i=1}^{n} \frac{\kappa_i}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \tag{2.7}$$

By taking the estimate of the density gradient as the gradient of the density estimate, the variable-bandwidth mean shift vector is defined by

$$\begin{aligned} \mathbf{m}(\mathbf{x}) &\equiv \mathbf{P}_h(\mathbf{x}) \sum_{i=1}^{n} \omega_i(\mathbf{x}) \mathbf{P}_i^{-1} \mathbf{x}_i - \mathbf{x} \\ &= \left(\sum_{i=1}^{n} \omega_i(\mathbf{x}) \mathbf{P}_i^{-1}\right)^{-1} \left(\sum_{i=1}^{n} \omega_i(\mathbf{x}) \mathbf{P}_i^{-1} \mathbf{x}_i\right) - \mathbf{x} \end{aligned} \tag{2.8}$$

where

$$\mathbf{P}_h^{-1}(\mathbf{x}) = \sum_{i=1}^{n} \omega_i(\mathbf{x}) \mathbf{P}_i^{-1} \tag{2.9}$$

and the weights

$$\omega_i(\mathbf{x}) = \frac{\kappa_i \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)}{\sum_{i=1}^{n} \kappa_i \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)} \tag{2.10}$$

satisfy $\sum_{i=1}^{n} \omega_i(\mathbf{x}) = 1$.

It can be shown that by iteratively computing the mean shift (2.8) and translating the location $\mathbf{x}$ by $\mathbf{m}(\mathbf{x})$, a mode seeking algorithm is obtained, which converges to a stationary point of the density (2.5). There are three kinds of stationary points in the density function: local maxima, local minima and saddle points. We are interested in finding mode locations (local maxima) to simplify the original density function, and a formal check for the maxima involves the computation of the Hessian matrix

$$\begin{aligned} \hat{\mathbf{H}}(\mathbf{x}) &\equiv (\boldsymbol{\nabla}\boldsymbol{\nabla}^{\top}) \hat{f}(\mathbf{x}) \\ &= \frac{1}{n(2\boldsymbol{\pi})^{d/2}} \sum_{i=1}^{n} \frac{\kappa_i}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \mathbf{P}_i^{-1} \left((\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^{\top} - \mathbf{P}_i\right) \mathbf{P}_i^{-1} \end{aligned} \tag{2.11}$$

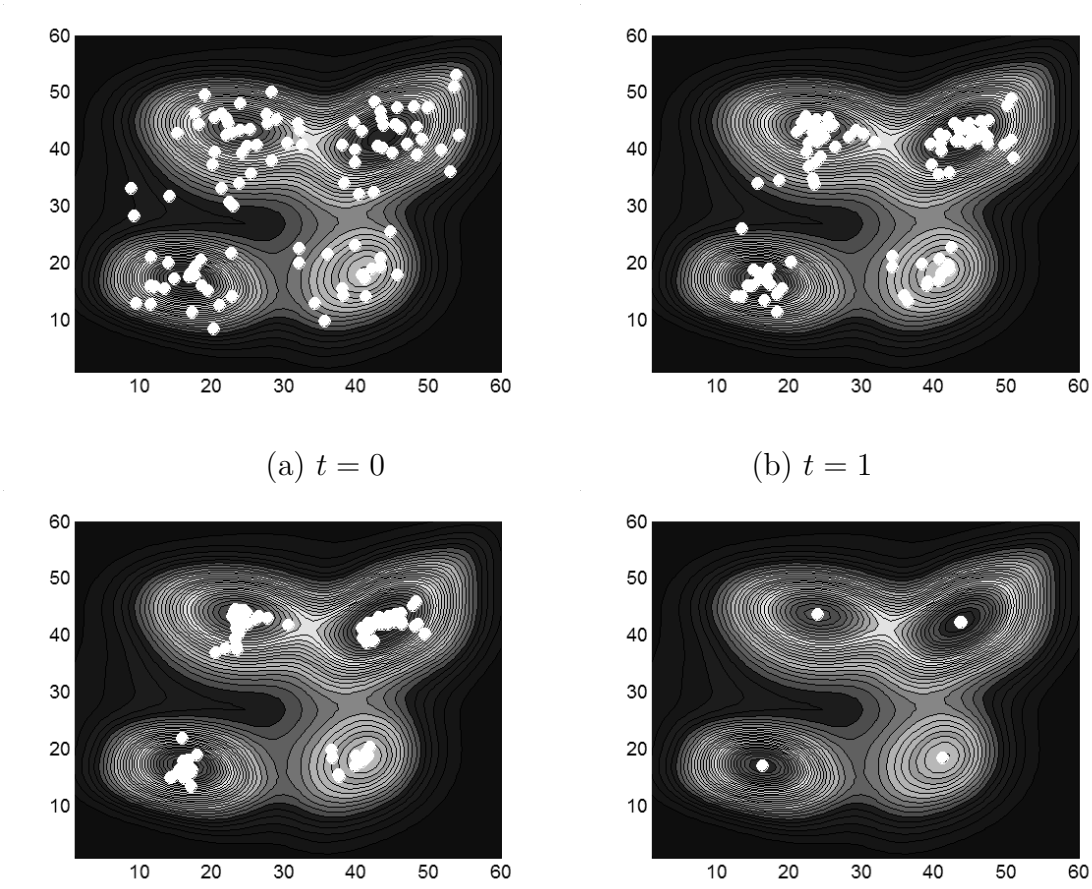
which should be negative definite (having all eigenvalues negative) at the mode location. If this condition is satisfied, all the sample points converged to a location should be merged with a single Gaussian centered at the convergence location. Otherwise, they should be left unchanged since density approximation creates too high error around the location.

Figure 2.2 shows the convergence to the local maximum of each sample by mode-finding algorithm. In each figure, the contour of a 2D density function constructed by KDE with 100 Gaussian kernels – all the weights are equal in this example – is presented, and white circles in Figure 2.2(a) means the initial location of each sample. As illustrated in the following figures, each sample is iteratively converged to a associated mode and four modes are finally detected at the 14th time step. The mean-shift procedure is terminated when the size of mean-shift vector is less than ϵ (negligibly small number).

2.2.2 Covariance Estimation

Suppose that the approximate density has m unique modes $\tilde{\mathbf{x}}_j$ $(j = 1, \ldots, m)$ with associated weights $\tilde{\kappa}_j$ after mode finding procedure; the weight of each mode $\tilde{\kappa}_j$ is equal to the sum of kernel weights converged to the mode. Then, the covariance matrix associated with each mode $\tilde{\mathbf{P}}_j$ is computed by curvature fitting in the neighborhood of the mode location.

The Hessian matrix $\hat{\mathbf{H}}(\mathbf{x}_j)$ at a mode $\tilde{\mathbf{x}}_j$ of the original density function in equation (2.7) is computed as in equation (2.11). Also, the Hessian matrix at the mean of a Gaussian distribution centered at $\tilde{\mathbf{x}}_j$ with weight $\tilde{\kappa}_j$ and covariance $\tilde{\mathbf{P}}_j$ is



(a) $t = 0$ (b) $t = 1$

(c) $t = 2$ (d) $t = 14$

Figure 2.2: Convergence by mode-finding algorithm

given by

$$\mathbf{H}_j(\mathbf{x}_j) = -\frac{\tilde{\kappa}_j}{(2\pi)^{d/2} \mid \tilde{\mathbf{P}}_j \mid^{1/2}} \tilde{\mathbf{P}}_j^{-1}. \tag{2.12}$$

Equalizing Hessian matrices in the original density and in the single Gaussian distribution, we solve for the estimated covariance matrix $\tilde{\mathbf{P}}_j$. Specifically, suppose that $\tilde{\mathbf{P}}_j$ is decomposed by Singular Value Decomposition (SVD) as $\tilde{\mathbf{P}}_j = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and that the Hessian matrix at $\tilde{\mathbf{x}}_j$ in equation (2.11) is represented with a similar form, $\hat{\mathbf{H}}(\mathbf{x}_j) = \mathbf{V}\mathbf{\Gamma}\mathbf{V}^\top$. Then, the following equation is obtained by the equalization of

two Hessian matrices.

$$\mathbf{V}\mathbf{\Gamma}\mathbf{V}^{\top} = -\frac{\tilde{\kappa}_j}{(2\pi)^{d/2} \mid \mathbf{\Lambda} \mid^{1/2}}\mathbf{U}^{\top}\mathbf{\Lambda}^{-1}\mathbf{U} \tag{2.13}$$

By assuming $\mathbf{U} = \mathbf{V}^{\top}$ and from equation (2.13), we can compute $\mid -\mathbf{\Gamma} \mid$ which is given by

$$\mid -\mathbf{\Gamma} \mid = -\frac{\tilde{\kappa}_j^d}{(2\pi)^{d^2/2}} \mid \mathbf{\Lambda} \mid^{-\frac{d+2}{2}}, \tag{2.14}$$

and $\mathbf{\Lambda}$ is derived from equation (2.13) and (2.14) as follows.

$$\mathbf{\Lambda} = \frac{\tilde{\kappa}_j^{\frac{2}{d+2}}}{\mid 2\pi(-\mathbf{\Gamma}^{-1}) \mid^{\frac{1}{d+2}}}(-\mathbf{\Gamma}^{-1}) \tag{2.15}$$

Therefore, the estimated covariance matrix is given by

$$\tilde{\mathbf{P}}_j = \frac{\tilde{\kappa}_j^{\frac{2}{d+2}}}{\mid 2\pi(-\hat{\mathbf{H}}_j^{-1}) \mid^{\frac{1}{d+2}}}(-\hat{\mathbf{H}}_j^{-1}), \tag{2.16}$$

and the approximated density is

$$\tilde{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{m} \frac{\tilde{\kappa}_i}{\mid \tilde{\mathbf{P}}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \tilde{\mathbf{x}}_{t+1}^i, \tilde{\mathbf{P}}_{t+1}^i\right)\right). \tag{2.17}$$

where $m(\ll n)$ is the number of detected modes. The approximation error $||\hat{f}(\mathbf{x}) - \tilde{f}(\mathbf{x})||$ can be evaluated straightforwardly.

2.3 Incremental Kernel Density Approximation

The density approximation technique described in Section 2.2 is accurate and memory efficient, but computationally expensive because the mode detection procedure for n components requires $O(n^2)$ time. Moreover, for each sample point, a large number of mean-shift iterations might be required to converge. To overcome this

computational costs, we suggest an alternative method, an incremental density approximation, described below.

Usually, a large number of samples are required to estimate the density correctly, but there are only several modes in the underlying density function. The incremental approximation algorithm is an empirical solution exploiting this fact. Suppose n samples are to be used for the density approximation. We will process samples, "one at a time." If a kernel associated with each sample can be merged incrementally with others in the same mode, then the time to compute the mean-shift vector will be decreased dramatically.

The algorithm proceeds as follows. When the Gaussian kernel for the next sample is added to the current density function, the density will be updated by the variable-bandwidth mean-shift. For example, if the component for the k-th sample is added to the current density function $\hat{f}_{(k-1)}$, the density after the insertion is given by

$$\begin{aligned}\hat{f}_k(\mathbf{x}) = \frac{\kappa_k}{K_k}\frac{1}{(2\pi)^{d/2} \mid \mathbf{P}_k \mid^{1/2}}\exp\left(-\frac{1}{2}D^2\left(\mathbf{x}, \mathbf{x}_k, \mathbf{P}_k\right)\right) + \\ \left(1-\frac{\kappa_k}{K_k}\right)\sum_{i=1}^{n_{k-1}}\frac{\hat{\kappa}_i}{(2\pi)^{d/2} \mid \hat{\mathbf{P}}_i \mid^{1/2}}\exp\left(-\frac{1}{2}D^2\left(\mathbf{x}, \hat{\mathbf{x}}_i, \hat{\mathbf{P}}_i\right)\right)\end{aligned} \tag{2.18}$$

where $K_k = \sum_{i=1}^{k}\kappa_i$, $\hat{\kappa}_i$ and $\hat{\mathbf{P}}_i$ are the weight and the covariance associated with $\hat{\mathbf{x}}_i$ in the current density respectively, and n_{k-1} is the number of modes after the $(k-1)$-th component insertion. In each step, the mode detection procedure and covariance estimation need to be applied, and the new density function $\hat{f}_k(\mathbf{x})$ is estimated. After n steps, the weight of each sample is adjusted to its original weight, and the

incremental density approximation can be obtained.

During the incremental procedure, two or more modes which are close to each other in the underlying density may be merged, and some of them may be lost by the final iteration. This situation should be avoided since it increases the approximation error. We avoid this problem by using a 2-stage algorithm. In the first stage, the incremental density approximation technique is used with a small bandwidth. This may result in several *spurious* modes which do not exist in the underlying density. After the final step, let each component in the approximate density be $N(\hat{\kappa}_i, \hat{\mathbf{x}}_i, \hat{\mathbf{P}}_i)$ $(i = 1 \ldots n_n)$ where $N(\cdot)$ is a Gaussian distribution having a (*weight, mean, covariance*) triple. In the second stage, the batch density approximation algorithm described in 2.2 is performed with the $\hat{\mathbf{x}}_i$'s as starting points. The correct mode locations and their covariance matrices can be computed accurately in the second stage.

The 2-stage incremental algorithm is very efficient since the intermediate and the final density function in the first stage have a small number of modes ($n_k \ll n$) in most cases.

2.4 Performance of Approximation

The accuracy of these approximations is demonstrated in Figure 2.3. From a one-dimensional distribution composed of five weighted Gaussians, 200 samples are drawn. Figure 2.3(a) shows the result of kernel density estimation. For the approximation, 200 samples are drawn from the original distribution – $N(0.2, 10, 2)$, $N(0.35, 17, 4)$, $N(0.15, 27, 8)$, $N(0.2, 50, 16)$, and $N(0.1, 71, 32)$. The results of batch

approximation with variable-bandwidth mean-shift are presented in Figure 2.3(b). The incremental approximation is presented in Figure 2.3(c) and the number of modes in each incremental step is shown in Figure 2.3(d).

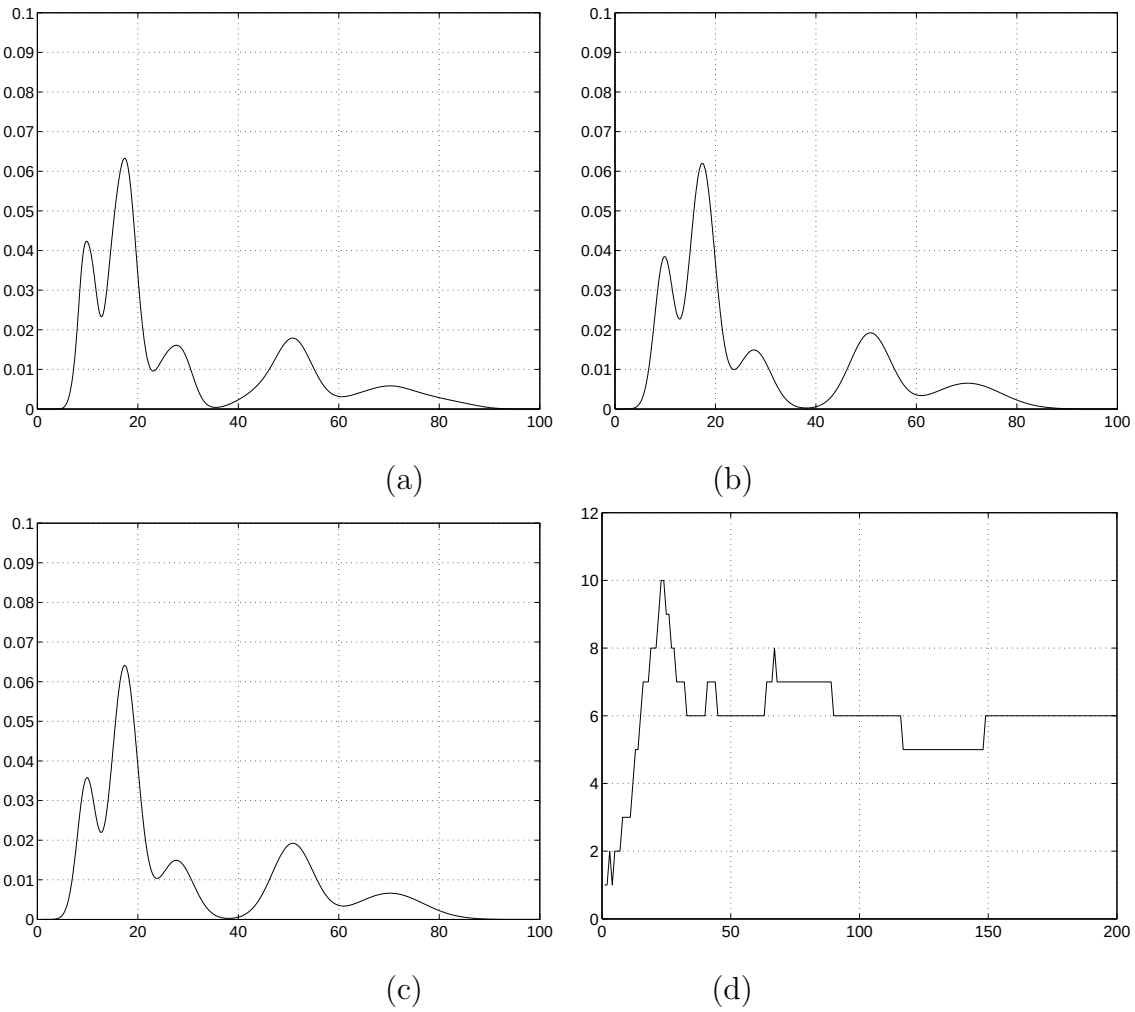


Figure 2.3: Comparisons between the kernel density estimation and its approximations (1D). (a) kernel density estimation (b) batch approximation (c) incremental approximation (d) number of modes in each incremental step

Table 2.2 compares accuracy and speed of the approximations. Three different cases are tested 20 times each, and Mean Integrated Squared Error (MISE) and execution time speedups are calculated. Denote by E_{kde} the error between the

Table 2.2: Performance comparison between batch and incremental approximation

case	MISE ($\times 10^{-5}$)			speedup
	E_{kde}	E_{bat}	E_{inc}	(batch/incremental)
1	5.0772	1.4512	3.1007	8.3502
2	2.2909	0.5323	1.2463	7.0119
3	1.0138	0.6900	1.7869	6.2597

- case 1: $N(0.2, 10, 2)$, $N(0.35, 17, 4)$, $N(0.15, 27, 8)$, $N(0.2, 50, 16)$, $N(0.1, 71, 32)$
- case 2: $N(0.15, 12, 5)$, $N(0.1, 50, 4)$, $N(0.35, 70, 8)$, $N(0.25, 90, 16)$, $N(0.15, 119, 32)$
- case 3: $N(0.15, 25, 10)$, $N(0.1, 37, 8)$, $N(0.15, 65, 16)$, $N(0.25, 77, 9)$, $N(0.15, 91, 30)$, $N(0.2, 154, 15)$

kernel density estimation and the original distribution, and by E_{bat} (E_{inc}) the error between the batch (incremental) approximation and the kernel density estimation. Both density approximations produce small errors comparable to kernel density estimation, and the incremental approximation is much faster with errors comparable to the batch approximation.

Figure 2.4 shows that both approximation methods are accurate enough to replace kernel density estimation in the multi-dimensional case. In 2D, the incremental approximation also has comparable accuracy to the batch approximation, but it is practically much faster (about 11 times) than the batch approximation when 400 samples are drawn.

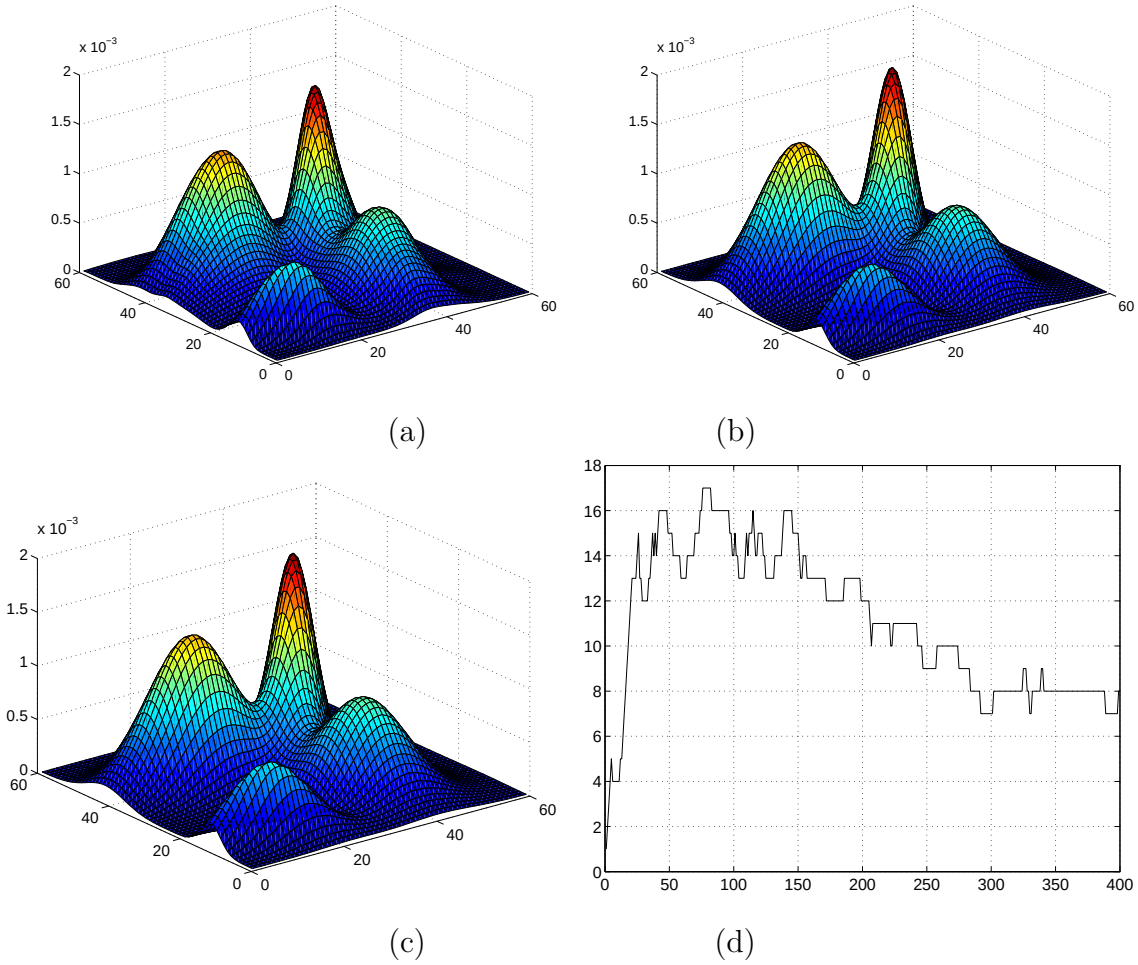


Figure 2.4: Comparison between the kernel density estimation and its approximations (2D). (a) kernel density estimation (b) batch approximation (MISE = 1.4889×10^{-8}) (c) incremental approximation (MISE = 1.2337×10^{-8}) (d) number of modes in each incremental step

Chapter 3

Background Modeling by Sequential Density Approximation

Automatic detection of objects is critical for security systems and video surveillance applications, since subsequent processing such as tracking, recognition, and high-level estimation techniques depends on the initial detection. In many vision based systems, such detection is carried out by background subtraction methods. These methods build a background model from the scene taken by stationary cameras, and detect the variation of the scene to find foreground regions.

A variety of methods have been proposed, but most of them employ density-based approaches in which visual features in a pixel or block are modeled by a certain density function. The performance of algorithms highly depends on the accuracy and flexibility of the density functions. Also, the management of the density functions is important to adapt to temporal changes of the background.

In this chapter, we show new kernel density approximation technique can be used to update the density function in a sequential manner to model the background adaptively. The proposed method is as flexible as kernel density estimation, and as compact as parametric methods, i.e. mixture of Gaussians. Background in each pixel is modeled by proposed method, and the performance of our background subtraction

is demonstrated.

3.1 Introduction

Stationary cameras are typically used to monitor activities in visual surveillance systems, and background subtraction is the process to detect moving objects by comparing each new frame with the constructed background model.

Background subtraction is the first stage for many high level applications, and it plays a very important roll for the further processing such as object tracking and event recognition.

Pixel-wise density-based modeling is very common for background subtraction vision, either by using non-parametric techniques or through representing the underlying density function as a weighted sum of Gaussians. Like a number of real-time tasks in computer vision area, background modeling requires sequential density estimation, where new data is incorporated in the model as it becomes available. Nevertheless, current methods for updating the density function either lack flexibility, by fixing the number of Gaussians in the mixture, or require large memory amounts, by maintaining a non-parametric representation of the density. This chapter presents an efficient method for recursive density approximation that relies on the propagation of the density modes. At each time step, the modes of the density are re-estimated and a Gaussian component is assigned to each mode. The covariance of each component is derived from the Hessian matrix estimated at the mode location. To detect the modes we employ the variable-bandwidth mean shift. While the proposed density representation is memory efficient (which is typical for

mixture densities), it inherits the flexibility of non-parametric methods, by allowing the number of modes to adapt in time. We show that the same mode propagation principle applies for subspaces derived from eigen analysis. Extensive experimental background modeling results demonstrate the performance of the method.

The chapter is organized as follows. In Section 3.2, various background modeling and subtraction methods are reviewed. Section 3.3 introduces the employed density representation and discusses mode detection using the variable-bandwidth mean shift. Sequential density approximation using mode propagation is introduced and tested in Section 3.4. Section 3.5 presents mode propagation in subspaces derived from eigen analysis. Background modeling experiments are given in Section 3.6.

3.2 Related Work

In most of background subtraction algorithms, pixel-wise modeling of visual features is most popular. Intensity or color are the most commonly used feature in background modeling, and a Gaussian distribution is the simplest density model. Wren et al. [83] models the color of each pixel using a single Gaussian, and Kalman update is attempted to adapt the variation of background [39, 63]. Also, more robust detection algorithm is proposed in [30].

The challenges of background modeling are often found in outdoor scenes due to illumination change and dynamic background. Due to variations, background modeling using a single component is not enough for the accurate estimation, the learning by multiple components is required. A mixture of Gaussians is proposed to model intensity or color distribution using a fixed number of components in

[26, 74]. The adaptation of the Gaussian mixture is achieved by incremental version of EM algorithm. The EM algorithm [16, 62] is the main statistical tool for learning the mixture model. Although variants of incremental EM have been employed in practice to deal with real-time adaptation constraints, it is generally difficult to add or remove components in the EM framework in a principled way [61]. Therefore, most real-time applications rely on models with a fixed number of mixtures or apply ad-hoc strategies to adapt the number of mixtures in time. Therefore, this technique may not be flexible enough to model complex background.

Non-parametric method is initially proposed by Elgammal [22] for more general background modeling, and arbitrary background distribution is simulated by kernel density estimation. Non-stationary background is directly addressed in kernel density estimation framework [48, 60]. Optical flow is modeled by kernel density estimation with color information in [48], and several pixel-wise models based on intensities and spatio-temporal derivatives are proposed in [60].

Another approach to model pixel-wise variations is to represent these variations as discrete states corresponding to modes of the environment. Hidden Markov Model (HMM) is used for this purpose in [64, 75]. In [75], the topology of the HMM representing global image intensity is learned while learning the background. Three state HMM is used to model the intensity of a pixel for traffic monitoring system in which each state corresponds to the background, shadow and foreground, respectively [64].

Also, there are several approaches to employ spatial coherency of given frame for background subtraction. A three-component system – pixel, region and frame

level – for background modeling is proposed in [78], and region and edge information is jointly integrated in [32]. On the other hand, several principal components obtained by eigenanalysis are used to model background in [55, 77]. Recently, on-line auto-regressive model to capture and predict the behavior of dynamic scenes are proposed in [49, 85].

3.3 Kernel Density Estimation and Mode Detection

In this section, we review kernel density estimation with adaptive bandwidth and an iterative procedure for mode detection based on the variable-bandwidth mean shift.

Denote by $\mathbf{x}_i$, $i = 1 \dots n$ a set of points in R^d and assume that a symmetric positive definite $d \times d$ bandwidth matrix $\mathbf{P}_i$ is associated with each data point $\mathbf{x}_i$. The sample point density estimator with a d-variate normal kernel, computed at the point $\mathbf{x}$ is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}} \sum_{i=1}^{n} \frac{1}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \tag{3.1}$$

where

$$D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right) \equiv (\mathbf{x} - \mathbf{x}_i)^\top \mathbf{P}_i^{-1} (\mathbf{x} - \mathbf{x}_i) \tag{3.2}$$

is the Mahalanobis distance from $\mathbf{x}$ to $\mathbf{x}_i$. As one can see, the density at $\mathbf{x}$ is obtained as the *average of Gaussian densities* centered at each data point $\mathbf{x}_i$ and having the covariance $\mathbf{P}_i$.

The mean-shift is a gradient ascent method to find a local maximum from a

given starting point. The variable-bandwidth mean-shift vector is defined by

$$\begin{aligned}\mathbf{m}(\mathbf{x}) &\equiv \mathbf{P}_h(\mathbf{x})\sum_{i=1}^{n}\omega_i(\mathbf{x})\mathbf{P}_i^{-1}\mathbf{x}_i - \mathbf{x} \\ &= \left(\sum_{i=1}^{n}\omega_i(\mathbf{x})\mathbf{P}_i^{-1}\right)^{-1}\left(\sum_{i=1}^{n}\omega_i(\mathbf{x})\mathbf{P}_i^{-1}\mathbf{x}_i\right) - \mathbf{x} \end{aligned} \tag{3.3}$$

where

$$\mathbf{P}_h^{-1}(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x})\mathbf{P}_i^{-1} \tag{3.4}$$

and the weights

$$\omega_i(\mathbf{x}) = \frac{\mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2}D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)}{\sum_{i=1}^{n} \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2}D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)} \tag{3.5}$$

satisfy $\sum_{i=1}^{n}\omega_i(\mathbf{x}) = 1$.

Mode seeking algorithm is obtained by iteratively translating the current location $\mathbf{x}$ by mean-shift vector $\mathbf{m}(\mathbf{x})$, and each sample point converges to a stationary point of the density (3.1). Since the maxima of the density are the only stable points of the iterative procedure, most of the time the convergence occurs at a mode of the underlying density. A formal check for the maximum involves the computation of the Hessian matrix

$$\begin{aligned}\hat{\mathbf{H}}(\mathbf{x}) &\equiv (\boldsymbol{\nabla}\boldsymbol{\nabla}^{\top})\hat{f}(\mathbf{x}) \\ &= \frac{1}{n(2\boldsymbol{\pi})^{d/2}}\sum_{i=1}^{n}\frac{1}{|\mathbf{P}_i|^{1/2}}\exp\left(-\frac{1}{2}D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \times \\ &\qquad \mathbf{P}_i^{-1}\Big((\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^{\top} - \mathbf{P}_i\Big)\mathbf{P}_i^{-1}\end{aligned} \tag{3.6}$$

which should be negative definite (having all eigenvalues negative) at the mode location.

Let each Gaussian receive a weight κ_i, with $\sum_{i=1}^{n} \kappa_i = 1$. The density (3.1) becomes

$$\hat{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{n} \frac{\kappa_i}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \tag{3.7}$$

which represents a mixture of Gaussian components. All the equations from above remain the same except those defining the weights ω_i

$$\omega_i(\mathbf{x}) = \frac{\kappa_i \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)}{\sum_{i=1}^{n} \kappa_i \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)} \tag{3.8}$$

where $\sum_{i=1}^{n} \omega_i(\mathbf{x}) = 1$.

Suppose that the approximate density has m unique modes $\tilde{\mathbf{x}}_j$ $(j = 1, \ldots, m)$ with associated weights $\tilde{\kappa}_j$ after mode finding procedure; the weight of each mode $\tilde{\kappa}_j$ is equal to the sum of kernel weights converged to the mode. Also, the covariance for each mode is defined by

$$\tilde{\mathbf{P}}_j = \frac{\tilde{\kappa}_j^{\frac{2}{d+2}}}{\mid 2\pi(-\hat{\mathbf{H}}_j^{-1}) \mid^{\frac{1}{d+2}}} (-\hat{\mathbf{H}}_j^{-1}) \tag{3.9}$$

as shown in Section 2.2.2.

Then, the approximated density is given by

$$\tilde{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{m} \frac{\tilde{\kappa}_i}{\mid \tilde{\mathbf{P}}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \tilde{\mathbf{x}}_{t+1}^i, \tilde{\mathbf{P}}_{t+1}^i\right)\right). \tag{3.10}$$

and $m(\ll n)$ is the number of detected modes.

In the following section, we will show how to use this framework to propagate the density modes in time.

3.4 Sequential Density Approximation

Assume that at time t the underlying density has n_t modes and that for each mode we have allocated a Gaussian $N(\kappa_i, \mathbf{x}_i, \mathbf{P}_i)$, according to equation (3.7). For the moment, select a learning rate α and assume that all incoming data become part of the model. Let $N(\alpha, \mathbf{x}_t, \mathbf{P}_t)$ be a new measurement. With the integration of the new measurement the density at time t is written as

$$\hat{f}_t(\mathbf{x}) = \frac{\alpha}{(2\pi)^{d/2}} \frac{1}{\mid \mathbf{P}_t \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_t, \mathbf{P}_t\right)\right) + \frac{1-\alpha}{(2\pi)^{d/2}} \sum_{i=1}^{n_t} \frac{\kappa_i}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \tag{3.11}$$

Starting now from the locations $\mathbf{x}_t$ and $\mathbf{x}_i$ with $i = 1 \ldots n_t$, we perform mean shift iterations and let $\mathbf{x}_t^c$ and $\mathbf{x}_i^c$, $i = 1 \ldots n_t$ be the convergence locations. We select first the convergence locations at which more than one procedure converged. Let $\mathbf{y}$ be a point in this set and let $\mathbf{x}_{i_j}$, $j = 1 \ldots m$ be the starting locations for which the mean shift procedure converged to $\mathbf{y}$. If the Hessian $\hat{\mathbf{H}}_t(\mathbf{y}) = (\boldsymbol{\nabla}\boldsymbol{\nabla}^\top)\hat{f}_t(\mathbf{y})$ is negative definite we associate with the mode $\mathbf{y}$ a Gaussian component defined by $N(u_y, \mathbf{y}, \mathbf{P}(\mathbf{y}))$ where the weight is

$$u_y = \sum_{j=1}^{m} u_{i_j} \tag{3.12}$$

and the covariance is defined by

$$\mathbf{P}(\mathbf{y}) = u_y^{\frac{2}{d+2}} \mid 2\pi(-\hat{\mathbf{H}}_t(\mathbf{y})^{-1}) \mid^{-\frac{1}{d+2}} (-\hat{\mathbf{H}}_t(\mathbf{y})^{-1}) \tag{3.13}$$

At time $t+1$ the Gaussian components located at $\mathbf{x}_{i_j}$, $j = 1 \ldots m$ will be substituted for by the new Gaussian denoted by $N(u_y, \mathbf{y}, \mathbf{P}(\mathbf{y}))$. It is obtained by fitting a

Gaussian in the neighborhood of the mode $\mathbf{y}$. The detail of the derivation of equation (3.13) is described in Section 2.2.2.

If the Hessian $\hat{\mathbf{H}}_t(\mathbf{y})$ is not negative definite (i.e., the location $\mathbf{y}$ is either a saddle point or a local minimum), all the components associated with $\mathbf{x}_{i_j}$, $j = 1 \ldots m$ are left unchanged since the Gaussian approximation in the neighborhood $\mathbf{y}$ would yield too high error.

For the convergence locations at which only one procedure converged, the weight, mean and covariance of the associated Gaussian component (as given in equation (3.11)) are also left unchanged.

Following the procedure above, we start from expression (3.11) of the density function $\hat{f}_t(\mathbf{x})$ and derive $\hat{f}_{t+1}(\mathbf{x})$. The number of Gaussians, n_{t+1}, can increase or decrease with respect to n_t, as a function of the increase/decrease in the complexity of the underlying density at time $t + 1$, represented by the number of modes.

In Figure 3.1 we show the mode propagation framework applied in one dimension. We used 500 measurements from a real video stream and a learning rate of $\alpha = 0.05$. We have represented in Figure 3.1(a)-(f) both the standard non-parametric estimate of the density and the density computed using mode propagation. Observe the very close resemblance of the two graphs in time, even for very complex density functions. The Mean Integrated Squared Error is drawn in Figure 3.1(g) while the number of modes is shown in Figure 3.1(h).

A two-dimensional experiment with 100 measurements is shown in Figure 3.3. Observe again the close resemblance between the standard kernel density estimate and the density computed through mode propagation, although the later has been

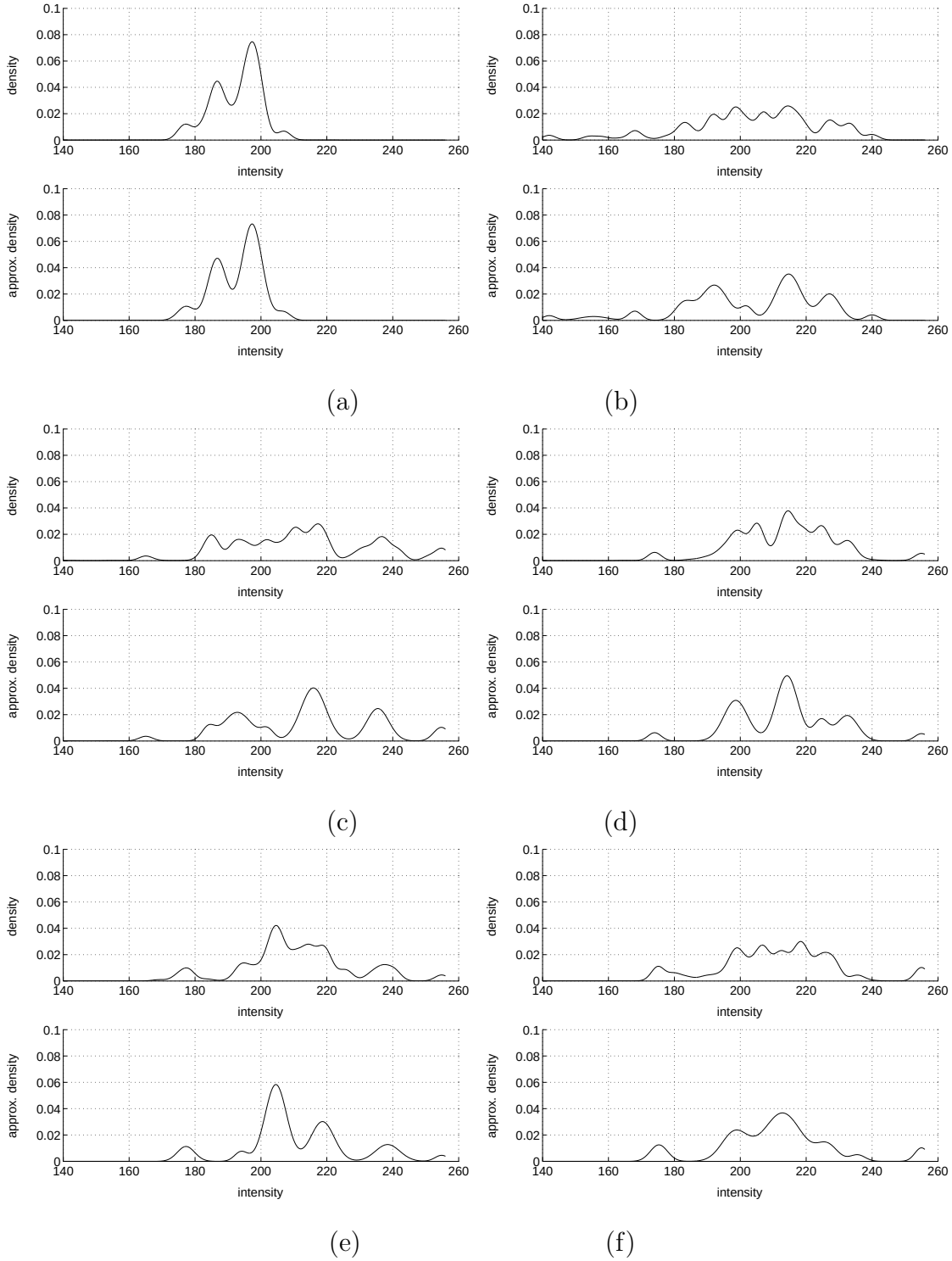
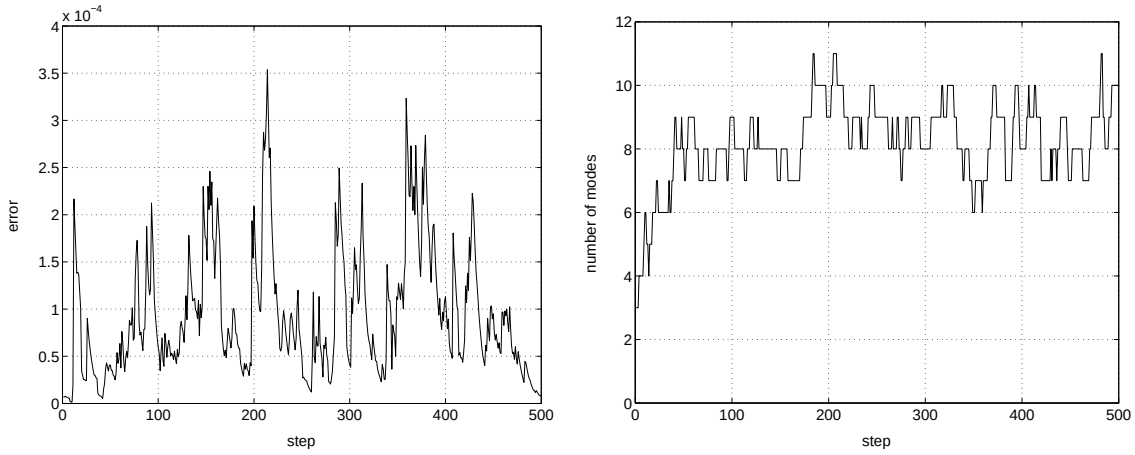


Figure 3.1: One dimensional mode propagation. (a)-(f) Standard non-parametric density estimation (above) vs. density computed through mode propagation (below). Steps 0, 60, 120, 180, 240, and 300 are represented.



(a) (b)

Figure 3.2: (a) Mean Integrated Squared Error. (b) Number of modes.

approximated with only 10 modes at the 60th step. The amount of memory required to represent this density function non-parametrically is much higher than in the case of mode propagation.

In the next section we extend the mode propagation framework to subspaces obtained from eigen analysis, while the eigenspace updating is implemented through Incremental PCA.

3.5 Propagation in Subspace

3.5.1 Batch PCA

Suppose that we have p-dimensional data points, denoted by $\mathbf{m}_i$, $i = 1, 2, \ldots, k$, having mean and covariance $\mathbf{m}$ and $\mathbf{P}$. Without loss of generality, we can assume $\mathbf{m} = \mathbf{0}$. Denote by $\mathbf{E}$ the matrix of eigenvectors of $\mathbf{P}$ which can be derived by the Singular Value Decomposition (SVD).

Select the q largest eigenvalues and the corresponding eigenvectors $\mathbf{e}_i$, $i =$

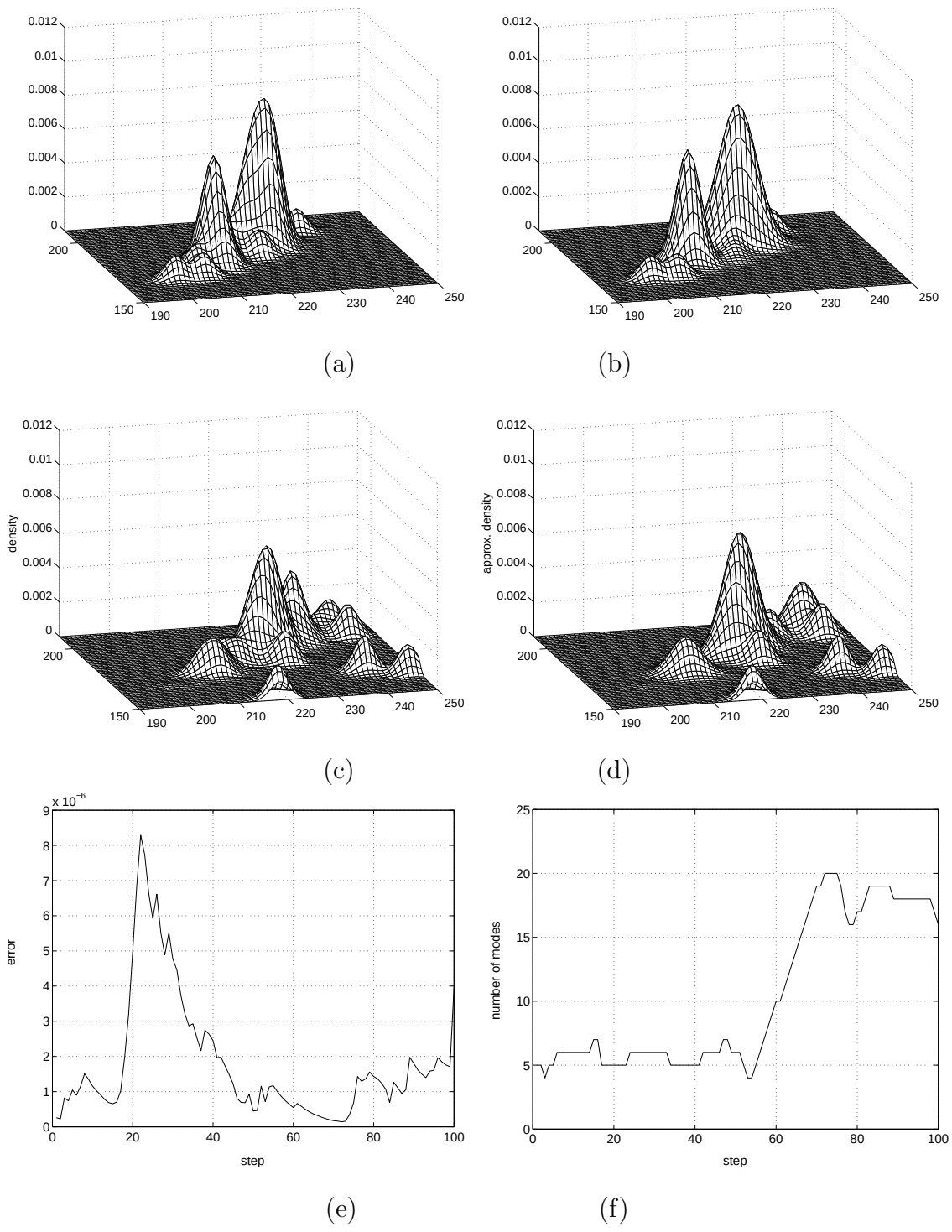


Figure 3.3: Two dimensional mode propagation. (a) and (c) Standard non-parametric density estimation. (b) and (d) Density computed through mode propagation. Steps 0 and 60 are represented. (e) Mean Integrated Squared Error. (f) Number of modes.

$1, 2, \ldots, q$, reducing the original p dimensional space to q dimensions ($q < p$). Define $\mathbf{E}_q$ to be a $p \times q$ matrix whose columns are eigenvectors, and project the data from the original space into the q dimensional subspace by the simple matrix-vector multiplication as $\mathbf{m}_i^q = \mathbf{E}_q^T \mathbf{m}_i$.

With these projected data $\mathbf{m}_i^q$, the mode finding algorithm is performed in the same fashion as in Section 3.3. Using the Hessian computed at the mode location we derive the covariance of each mode in the subspace. In the batch method, we need to perform the same procedure in each time step, so it is computationally expensive and it is not feasible to store all the history of data due to the limited memory space. Therefore, we consider incremental PCA algorithms [27, 82]. In this chapter, we use the method suggested in [82].

3.5.2 Incremental PCA (IPCA)

In the incremental PCA, we just need the small amount of additional information, the mean $\hat{\mathbf{m}}_i$ ($i = 1, 2, \ldots, n$) of each mode in the original space, other than the currently estimated density function and current basis vectors. Since there is not enough information to re-project all the data into the subspace, we have to formulate a method to transform the data representation in the current subspace to the next subspace. The information we have are the old and new eigenvectors ($\mathbf{E}_q$ and $\mathbf{E}'_q$, respectively), mean and covariance for the density in the old subspace, the new data $\mathbf{m}_{k+1}$, and the mean of each mode $\hat{\mathbf{m}}_i$ in the original space. Instead of projecting each mode from the old subspace to the new subspace, we would rather project $\hat{\mathbf{m}}_i$

into the new subspace as

$$\mathbf{m}_i^{'q} = \mathbf{E}_q^{'T} \hat{\mathbf{m}}_i \tag{3.14}$$

and the new data can be simply projected to the new subspace using

$$\mathbf{m}_{k+1}^{'q} = \mathbf{E}_q^{'T} \mathbf{m}_{k+1}. \tag{3.15}$$

With the means and covariances of re-projected density and the new data, the sequential density approximation algorithm is performed and the modes are propagated to the next step. At this time, we also have to update $\hat{\mathbf{m}}_i$ in the original space with the learning rate α. Figure 3.4 shows the mode propagation result in a two dimensional subspace.

3.6 Background Modeling

In this section, we describe the background modeling and subtraction application based on the kernel density approximation and mode propagation. We consider modeling the background based on each pixel or a fixed size block. The first step for background subtraction is to estimate the feature density of each unit (pixel or block); then, the background model of a unit is composed of a mixture of several Gaussians through the proposed algorithm. The covariance associated with a new component has been computed according to the method suggested in [22], using the median of the magnitude of differences between consecutive measurements. Each dimension is assumed to be independent of each other, and the initial covariance matrix is diagonal. The density function is updated selectively only if the new data is classified as background. We consider a unit as foreground if the feature of the unit

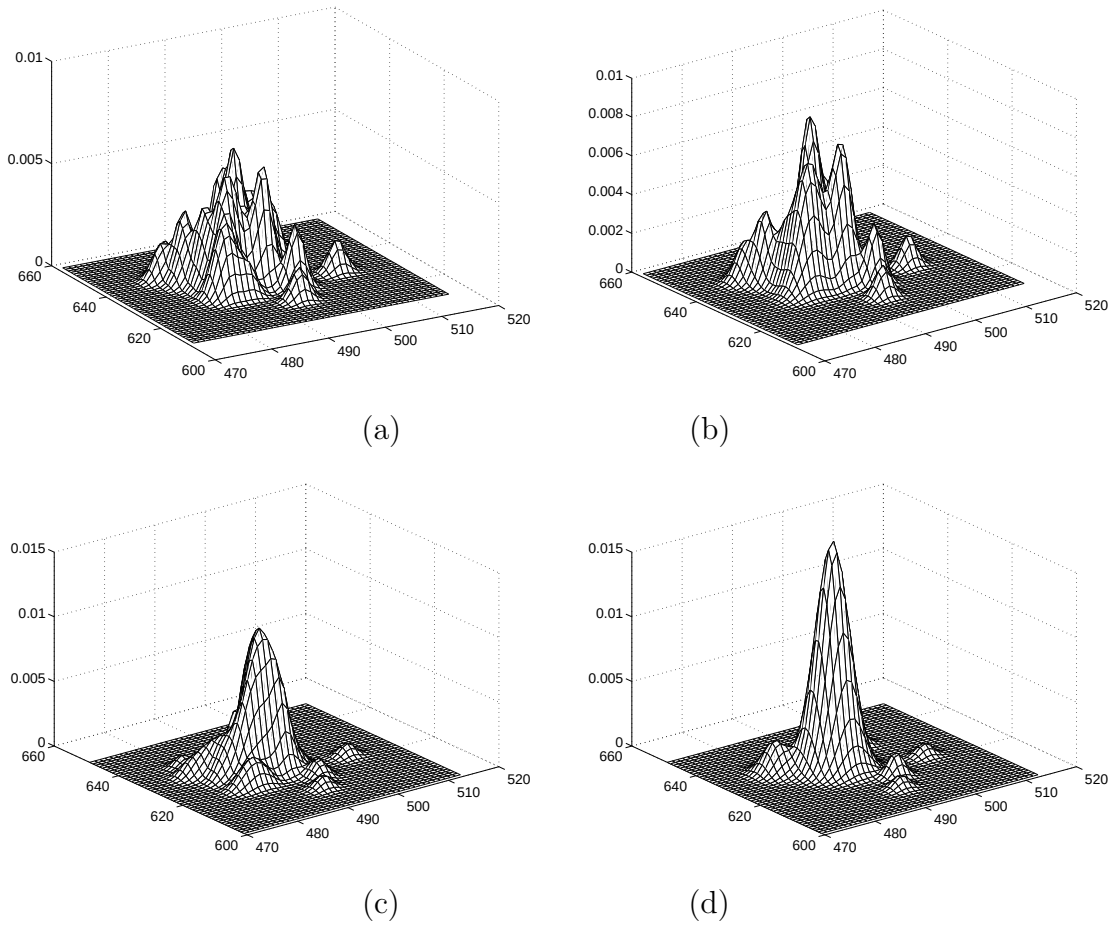


Figure 3.4: Two dimensional mode propagation in the subspace. (a) and (c) Standard non-parametric density estimation. (b) and (d) Density computed through mode propagation. Steps 0 and 50 are represented.

is far (e.g. outside 99.9% confidence ellipsoid) from every mode in the underlying density function. Only when the new data satisfies the background model do we update the model according to the procedure described in Section 3.4 and 3.5.

Two different sequences are used to test and compare the performance of the density approximation and mode propagation algorithm. The first one is from a subway station that contains a lot of pixelwise noise and the second is from a beach with large structural movements caused by waves. The experiments are performed

for gray and color images and for 3×3 blockwise modeling using the subspace method [69].

Figure 3.5 shows the result of background subtraction for color images with the gradual update of background.

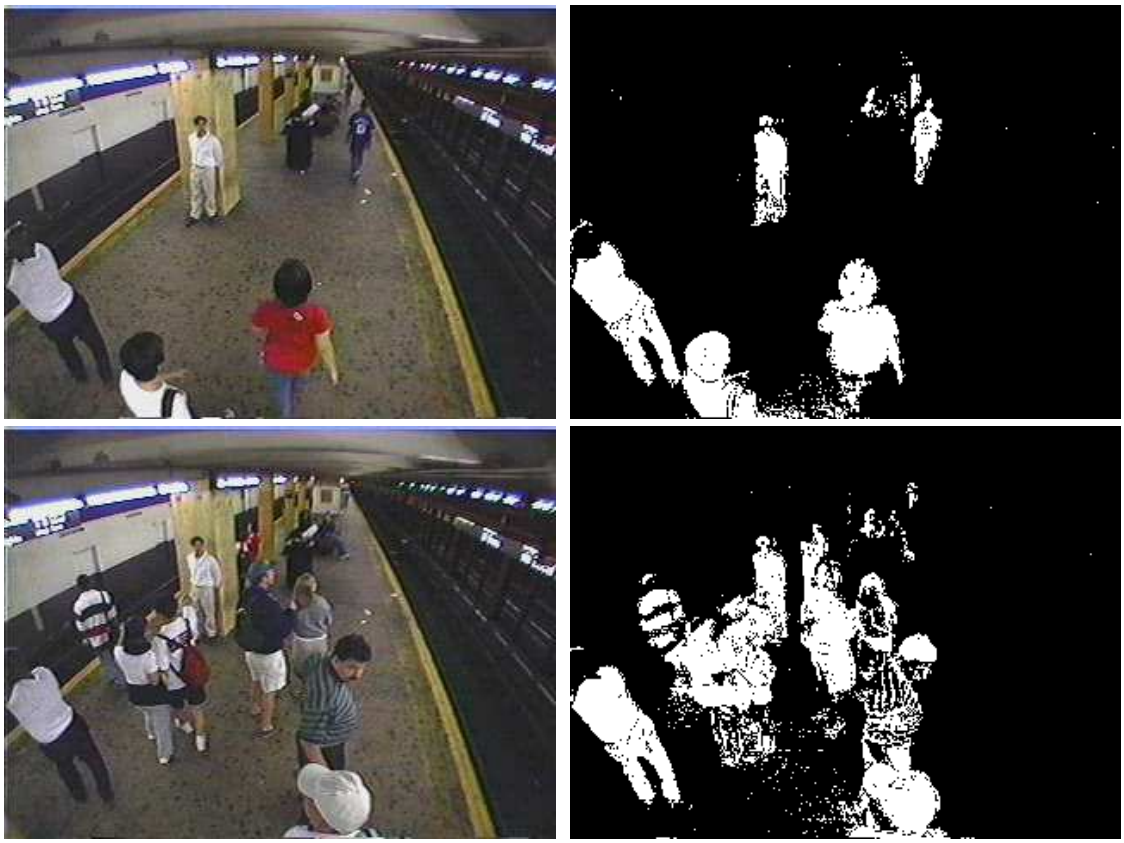
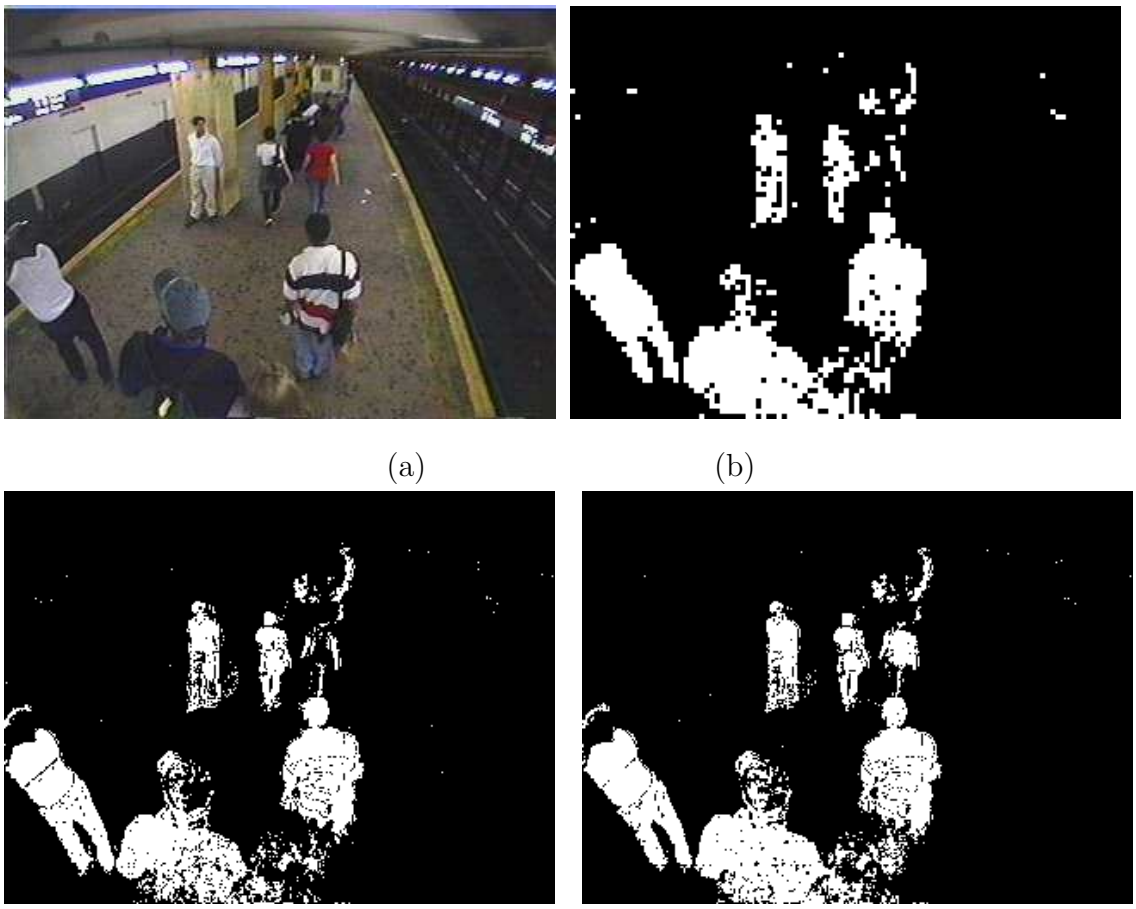


Figure 3.5: Background subtraction using mode propagation for the color density computed in the RGB space.

Figure 3.6 shows that background subtraction using color outperforms the gray scale image in both detection and false alarm. Also, the blockwise method in the subspace shows robust performance. Compare these results with those reported in [56], Figure 1.

A difficult sequence is one taken at the beach, where the waves constitute a



(a) (b)

(c) (d)

Figure 3.6: Background subtraction comparison. (a) Original frame. (b) Subspace. (c) Intensity modeling. (d) Color modeling.

dynamic background and the lighting condition changes a lot. Figure 3.7 shows several images used for initial background modeling containing a lot of movement due to the ocean waves.

Figure 3.8 shows the results for background subtraction in intensity, color and subspace. The large movements of the waves are accurately modeled as background, and the people in the scene are detected correctly.

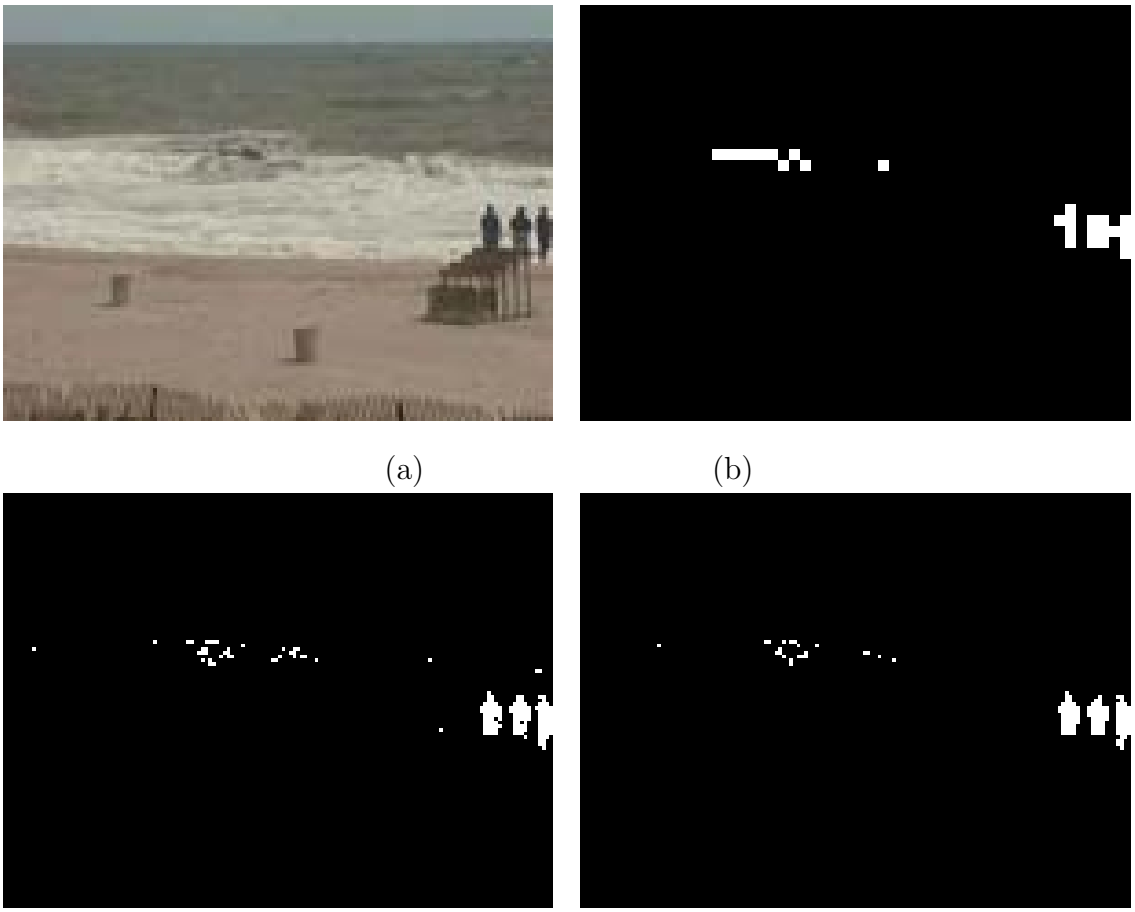
Another example of dynamic background modeling and subtraction in RGB

space is presented in Figure 3.9.

3.7 Conclusion

We have introduced in this paper mode propagation as a principled method for sequential density approximation. We show the correctness and the effectiveness of this algorithm by various simulations. The new framework has been successfully applied to background modeling and subtraction. We anticipate that our work will be useful for other computer vision applications, especially for tasks involving higher dimensional feature spaces.

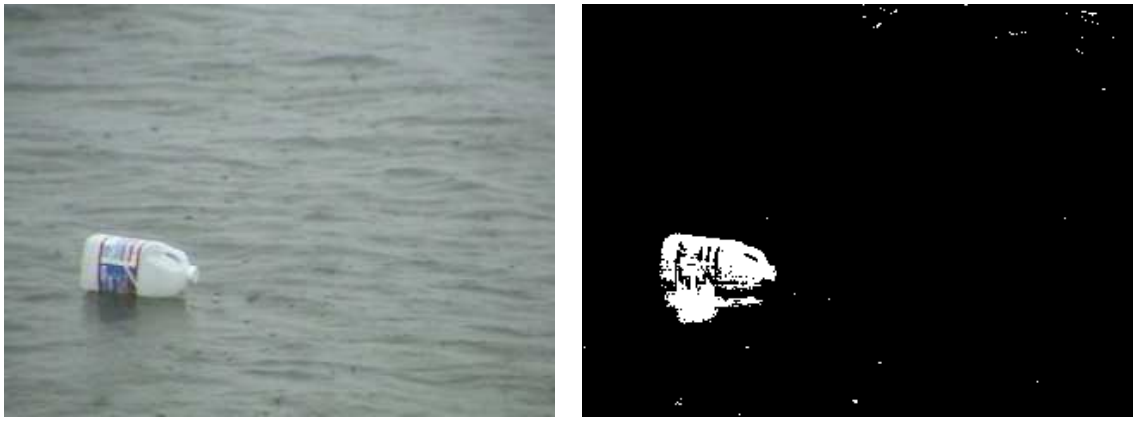
Figure 3.7: Sample frames used for background modeling.



(a) (b)

(c) (d)

Figure 3.8: Background subtraction comparison. (a) Original frame. (b) Subspace. (c) Intensity modeling. (d) Color modeling.



(a) (b)

Figure 3.9: Background subtraction for dynamic background. (a) Original frame. (b) Background subtraction in RGB space

Chapter 4

On-Line Density-Based Target Appearance Modeling

Object tracking is a challenging problems in real-time computer vision due to variations of lighting condition, pose, scale, and view-point over time. However, it is exceptionally difficult to model appearance with respect to all of those variations in advance; instead, on-line update algorithms are employed to adapt to these changes. We present a new on-line appearance modeling technique which is based on sequential density approximation. This technique provides accurate and compact representations using Gaussian mixtures, in which the number of Gaussians is automatically determined. This procedure is performed in linear time at each time step, which we prove by amortized analysis. Features for each pixel and rectangular region are modeled together by the proposed sequential density approximation algorithm, and the target model is updated in scale robustly. We show the performance of our method by simulations and tracking in natural videos.

4.1 Introduction

One of the most critical challenges in creating robust visual trackers is the development of adaptive appearance models that can accommodate unstable lighting

condition, pose variations, scale changes, view-point changes, and camera noise. Many tracking algorithms [13, 21, 58] are based on a fixed target model, and so are unable to track over long time intervals.

Some efforts have been made to overcome these problems. In [44], heuristics regarding the replacement of the target template are suggested; Nummiaro et al. [53] update the model by taking the weighted average of the current and new histograms. A view-based subspace model is implemented in EigenTracking [4], but it requires intensive offline learning before tracking. Recently, Ross et al. [65] propose an adaptive tracking algorithm that updates the models using an incremental update of eigenbasis.

Instead of using a template or a histogram for target modeling, parametric density representations have been used in many tracking algorithms. McKenna et al. [45] suggest Gaussian mixture models created by an EM algorithm for histogram-based trackers, but their method requires knowledge of the number of components, which may not be known in advance. Additionally, it is not appropriate if there are a large number of modes in the underlying density function or the number of modes changes frequently. In [25, 52], a pixel-wise target model based on Gaussian distribution is proposed, and it is updated during tracking. However, this method cannot model multi-modal density functions accurately. A more elaborate target model is described in [34], where a 3-component mixture for the stable process, the outlier data and the wandering term is designed to capture rapid temporal variations in the model. The implementation of that method allowed only a fixed number of components, so, for example, it cannot accommodate multiple stable components

since only one Gaussian is assigned to the stable component.

An important issue in target model update is the balance between adaptiveness to new observations and resistance to noise. Since the target model may drift away by undesirable updates, only target pixel observations should be integrated into the model. From this point of view, a probability density function of visual features is a good solution for target modeling, because frequently observed data contribute the most significant part while outliers can have limited effects on the integrity of the model.

In this chapter, we present an on-line density-based appearance model for object tracking which is more flexible than previously published parametric methods. The density function is composed of a mixture of Gaussians, where all of the parameters such as the number of modes, means, covariances, and weights are determined by a mean-shift algorithm [11, 12]. The method can represent the density function very accurately with a small amount of memory. Whenever a new observation is incorporated into the current model, the density function is updated in an on-line manner. This procedure is performed in linear time; the time complexity is proved by amortized analysis. Color features at each pixel are modeled by a Gaussian mixture, and color rectangular features are also integrated in the density function to encode the local spatial information of a pixel. We also describe a model update procedure in scale space using density approximation technique, so that the tracker can deal with scale changes robustly.

This chapter is organized as follows. Section 4.2 introduces the linear time sequential density approximation technique, and demonstrates its performance by

intensive simulation. Section 4.3 describes how to construct the appearance model adaptively, and experiments for object tracking in video are presented in Section 4.4.

4.2 On-line Density Approximation

Gaussian mixtures are frequently used to estimate density functions, and several heuristics to determine the number of components have been proposed [42, 61, 79]. However, it is extremely difficult to find the number of Gaussians in a principled way for on-line applications since previous data are not typically available at the current step due to memory constraint. In this section, we present an iterative procedure for sequential density approximation to manage a Gaussian mixture density function adaptively. Then, a more efficient algorithm is presented and its time complexity is discussed.

Section 4.2.1 presents the naive sequential density approximation method, while Section 4.2.2 presents a substantially faster algorithm. In Section 4.2.3, the performance of our sequential density approximation technique is investigated by various simulations.

4.2.1 Sequential Density Approximation

Assume that at time t the underlying density is a mixture of Gaussians having n_t modes and that for each mode we have allocated a Gaussian $N(\kappa_t^i, \mathbf{x}_t^i, \mathbf{P}_t^i)$, $i = 1, \ldots, n_t$, where $N(\cdot)$ is Gaussian distribution with the parameter (*weight*, *mean*, *covariance*). For the moment, select a learning rate α and assume that all incoming

data become part of the model. Let $N(\alpha, \mathbf{x}_t^{n_t+1}, \mathbf{P}_t^{n_t+1})$ be a new measurement. With the integration of the new measurement, the density at time $t+1$ is initially written as

$$\hat{f}_{t+1}(\mathbf{x}) = \frac{(1-\alpha)}{(2\pi)^{d/2}} \sum_{i=1}^{n_t} \frac{\kappa_t^i}{\mid \mathbf{P}_t^i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_t^i, \mathbf{P}_t^i\right)\right) + \frac{\alpha}{(2\pi)^{d/2} \mid \mathbf{P}_t^{n_t+1} \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_t^{n_t+1}, \mathbf{P}_t^{n_t+1}\right)\right) \tag{4.1}$$

where

$$D^2\left(\mathbf{x}, \mathbf{x}_t^i, \mathbf{P}_t^i\right) \equiv (\mathbf{x} - \mathbf{x}_t^i)^\top (\mathbf{P}_t^i)^{-1} (\mathbf{x} - \mathbf{x}_t^i) \tag{4.2}$$

is the Mahalanobis distance between $\mathbf{x}$ and $\mathbf{x}_t^i$.

To find the new mode locations in $\hat{f}_{t+1}(\mathbf{x})$, we perform mean-shift iterations until convergence for each $\mathbf{x}_t^i$, $i = 1 \ldots n_t + 1$. The variable-bandwidth mean-shift vector at location $\mathbf{x}$ is defined by

$$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^{n_t+1} \omega_t^i(\mathbf{x}) (\mathbf{P}_t^i)^{-1}\right)^{-1} \left(\sum_{i=1}^{n_t+1} \omega_t^i(\mathbf{x}) (\mathbf{P}_t^i)^{-1} \mathbf{x}_i\right) - \mathbf{x} \tag{4.3}$$

where the weights

$$\omega_t^i(\mathbf{x}) = \frac{\kappa_t^i \mid \mathbf{P}_t^i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_t^i, \mathbf{P}_t^i\right)\right)}{\sum_{i=1}^{n_t+1} \kappa_t^i \mid \mathbf{P}_t^i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_t^i, \mathbf{P}_t^i\right)\right)} \tag{4.4}$$

satisfy $\sum_{i=1}^{n_t+1} \omega_t^i(\mathbf{x}) = 1$.

We select first the convergence locations at which more than one procedure or $\mathbf{x}_t^{n_t+1}$ converged. Let $\mathbf{y}$ be a point in this set and let $\mathbf{x}_t^j$, $j = 1 \ldots m$ be the starting locations for which the mean shift procedure converged to $\mathbf{y}$. If the Hessian $\hat{\mathbf{H}}(\mathbf{y}) = (\boldsymbol{\nabla}\boldsymbol{\nabla}^\top)\hat{f}_{t+1}(\mathbf{y})$ is negative definite, we associate with the mode $\mathbf{y}$ a Gaussian component defined by $N(\kappa_y, \mathbf{y}, \mathbf{P}(\mathbf{y}))$ where κ_y is the sum of $\mathbf{x}_t^j$'s weights ($j =$

$1 \ldots m$) and the covariance matrix $\mathbf{P}(\mathbf{y})$ is given by

$$\mathbf{P}(\mathbf{y}) = \frac{\kappa_y^{\frac{2}{d+2}}}{\mid 2\pi(-\hat{\mathbf{H}}(\mathbf{y})^{-1}) \mid^{\frac{1}{d+2}}}(-\hat{\mathbf{H}}(\mathbf{y})^{-1}) \tag{4.5}$$

The basic idea of equation (4.5) is to fit the covariance using the curvature in the neighborhood of the mode. At time $t+1$, the Gaussian components located at $\mathbf{x}_t^j$ ($j = 1 \ldots m$) will be substituted for by the new Gaussian $N(\kappa_y, \mathbf{y}, \mathbf{P}(\mathbf{y}))$.

If the Hessian $\hat{\mathbf{H}}(\mathbf{y})$ is not negative definite (i.e., the location $\mathbf{y}$ is either a saddle point or a local minimum), all the components associated with $\mathbf{x}_t^j$, $j = 1 \ldots m$ are left unchanged since the Gaussian approximation in the neighborhood of $\mathbf{y}$ would yield too high an error. For the convergence locations at which only one procedure converged (except the convergence location for $\mathbf{x}_t^{n_t+1}$), the weight, mean and covariance of the associated Gaussian component are also left unchanged.

The new parameters in Gaussian mixture $\hat{f}_{t+1}(\mathbf{x})$ are determined by a mode finding algorithm based on mean-shift and covariance estimation method, and the updated density function is then given by

$$\hat{f}_{t+1}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{n_{t+1}} \frac{\kappa_{t+1}^i}{\mid \mathbf{P}_{t+1}^i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_{t+1}^i, \mathbf{P}_{t+1}^i\right)\right) \tag{4.6}$$

where κ_{t+1}^i, $\mathbf{x}_{t+1}^i$, and $\mathbf{P}_{t+1}^i$ are weight, mean, and covariance of each component at $t+1$, respectively.

4.2.2 Linear Time Algorithm for Sequential Approximation

The sequential density approximation technique described in the previous section takes $O(n_t^2)$ time in each step, where n_t is the number of modes at time step t. Now, we relax the constraint that the number of Gaussian components is equal to

the number of modes in the density function, and improve the time complexity to linear time. As a result, the number of Gaussian components may be slightly more than the compact representation introduced in Section 4.2.1, but the sequential density approximation is performed much faster asymptotically.

Recall equation (4.1). The previous algorithm runs the mean-shift procedure for all of $n_t + 1$ components, and finds convergence points for all of them. Then, it finds the mode associated with the new data, and updates that mode.[1] So, if we could identify those modes that would merge with the new data efficiently, then the execution time can be dramatically reduced. This technique is explained next.

Algorithm Description

Given the $n_t + 1$ modes in the $t + 1$st step, we first search for the convergence point $\mathbf{c}^{n_t+1}$ of $\mathbf{x}_t^{n_t+1}$ in the density $\hat{f}_{t+1}(\mathbf{x})$ of equation (4.1). Now, we have to find which other modes converge to $\mathbf{c}^{n_t+1}$ and should be merged with $\mathbf{x}_t^{n_t+1}$. The candidates to converge to $\mathbf{c}^{n_t+1}$ are determined by mean-shift, and this procedure is repeated until the candidate does not converge to $\mathbf{c}^{n_t+1}$ any more. The first candidate mode is the convergence point $\mathbf{x}_t^i$ $(i = 1 \dots n_t)$ from $\mathbf{x}_t^{n_t+1}$ in the density function $\hat{f}'_{t+1}(\mathbf{x}) = \hat{f}_{t+1}(\mathbf{x}) - N(\alpha, \mathbf{x}_t^{n_t+1}, \mathbf{P}_t^{n_t+1})$. Note that all the candidates are one of the components in the previous density function $\hat{f}_t(\mathbf{x})$. The mean-shift procedure is performed for $\mathbf{x}_t^i$ in $\hat{f}_{t+1}(\mathbf{x})$, and we check if the convergence point of $\mathbf{x}_t^i$ is equal to $\mathbf{c}^{n_t+1}$. If they are not equal, we conclude that there are no further merges with

[1] Rarely, some merges which do not include the new data may happen. It hardly affects the accuracy of the density functions, but leads to a difference in the number of components between the quadratic and the linear time algorithm.

$\mathbf{x}_t^{n_t+1}$ and create a Gaussian for the updated mode; otherwise, we check the next candidate, which is determined by finding the next convergence point of $\mathbf{x}_t^{n_t+1}$ in the density function $\hat{f}'_{t+1}(\mathbf{x}) = \hat{f}'_{t+1}(\mathbf{x}) - N(\kappa_t^i, \mathbf{x}_t^i, \mathbf{P}_t^i)$.

The covariance matrix and the weight of the merged mode should be also updated as proposed in Section 4.2.1. The formal description of this algorithm is given in algorithm 1. In algorithm 1, *MeanShiftModeFinding* is the function to detect the convergence location by the mean-shift algorithm from a point (the second argument) in the density (the first argument).

Algorithm 1 Linear-time sequential density approximation

1: $S = \{\mathbf{x}_t^{n_t+1}\}, \ \kappa = \alpha$
2: $\hat{f}'_{t+1}(\mathbf{x}) = \hat{f}_{t+1}(\mathbf{x})$
3: $\mathbf{c}^{n_t+1} = MeanShiftModeFinding(\hat{f}_{t+1}(\mathbf{x}), \mathbf{x}_t^{n_t+1})$
4: $\hat{f}'_{t+1}(\mathbf{x}) = \hat{f}'_{t+1}(\mathbf{x}) - N(\alpha, \mathbf{x}_t^{n_t+1}, \mathbf{P}_t^{n_t+1})$
5: **while** 1 **do**
6: $\quad \mathbf{x}_t^i = MeanShiftModeFinding(\hat{f}'_{t+1}(\mathbf{x}), \mathbf{x}_t^{n_t+1})$
7: $\quad \mathbf{c} = MeanShiftModeFinding(\hat{f}_{t+1}(\mathbf{x}), \mathbf{x}_t^i)$
8: $\quad$ **if** $\mathbf{c}^{n_t+1} \neq \mathbf{c}$ **then**
9: $\quad\quad$ **break**
10: $\quad$ **end if**
11: $\quad S = S \cup \{\mathbf{x}_t^i\}, \ \kappa = \kappa + \kappa_t^i$
12: $\quad \hat{f}'_{t+1}(\mathbf{x}) = \hat{f}'_{t+1}(\mathbf{x}) - N(\kappa_t^i, \mathbf{x}_t^i, \mathbf{P}_t^i)$
13: **end while**
14: merge all the modes in the set S and create $N(\kappa, \mathbf{c}, \mathbf{P_c})$ where $\mathbf{P_c}$ is derived by the same method in equation (4.5)

Analysis of Algorithm

The time complexity of this algorithm is $O(n_{max})$ by amortized analysis, where n_{max} is the maximum number of modes in all time steps, and a sketch of the proof is as follows.

Suppose that each of new data has $5n_{max} + 1$ credits, which is defined to be the reserved number of operations for the Gaussian component corresponding to the new data. For the search for the convergence point (line 3 in algorithm 1), at most $n_{max} + 1$ operations are performed and the new component consumes $n_{max} + 1$ credits since the function *MeanShiftModeFinding* takes linear time. $2n_{max}$ credits are required for two mean-shift iterations when the new component fails to merge (the last iteration of while loop). Also, we need $2n_{max}$ operations (line 6 and 7) whenever the new component is merged with the currently existing mode, but the existing mode is responsible for this cost. So, the remaining $2n_{max}$ credits are supposed to be used when another mode is merged with it later. After losing all credits, the mode finally disappears.

For K time steps, K new Gaussian components are entered and sequential density approximation is performed. Therefore, the number of operations for all K time steps is at most $O(Kn_{max})$, and the average time complexity in each step is $O(n_{max})$. The derived complexity is bounded by $O(n_{max})$, but practically it is faster than this since the number of components in each step is less than n_{max} in most cases. The number of components is slightly more than the previous quadratic algorithm, but the improvement of time complexity is the dominating factor for the

overall speed of algorithm.

4.2.3 Performance of Approximation

The linear time sequential density approximation algorithm, which we will refer to as the *fast approximation algorithm*, is a variant of the quadratic time algorithm. The performance of the fast approximation algorithm was tested through simulation, and compared with the quadratic algorithm as well as a sequential version of kernel density estimation.

Starting from the initial density function, a new data element is incorporated at each step, and Mean Integrated Squared Error (MISE) with sequential kernel density estimation is employed as a basis of comparison. The weighted Gaussian mixture – $N(0.15, 80, 15^2)$, $N(0.4, 122, 10^2)$, and $N(0.45, 122, 100^2)$ – is used as the initial density function, and the new data is sampled from another Gaussian mixture – $N(0.2, 52, 10^2)$, $N(0.5, 100, 10^2)$, and $N(0.15, 175, 10^2)$ – plus a uniform distribution in $[0, 255]$ with weight 0.15. In this experiment, we expect the initial density function to morph to the new density function from which data samples are drawn.

As seen in Figure 4.1, the fast approximation algorithm accurately simulates the sequential kernel density estimation; the final density function has three major modes which closely correspond to the Gaussian centers in the sampling function. The simulated density function using the quadratic algorithm is very similar to that of the fast approximation algorithm in most steps, so it is not presented separately in this figure. Note that the sequential kernel density estimation has more than 300 Gaussian components at the 300th time step, but the fast approximation algorithm

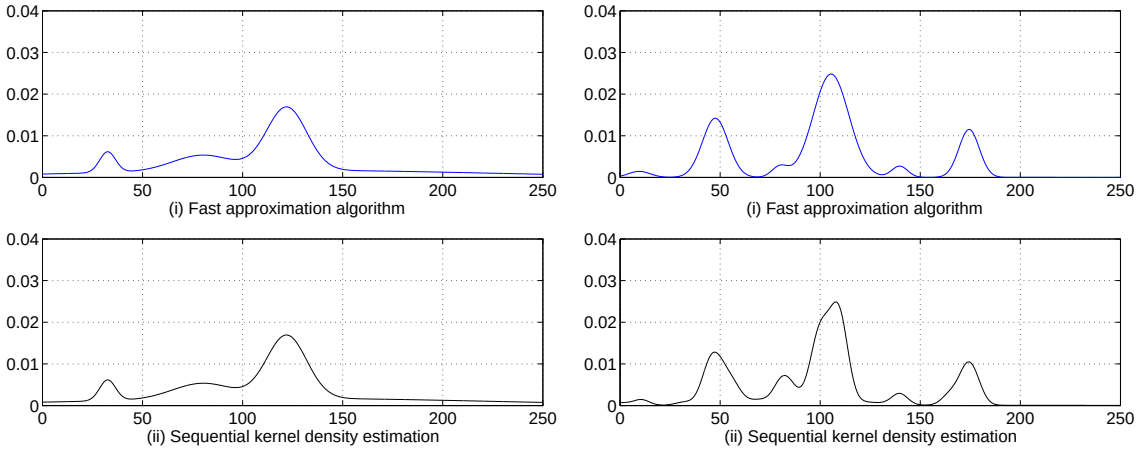
has only 7 components.

Figure 4.2 shows that the MISE of the fast approximation algorithm is comparable to the quadratic algorithm (and in repeated experiments, the new algorithm is often better) while the increase of the number of components using the fast algorithm is moderate.

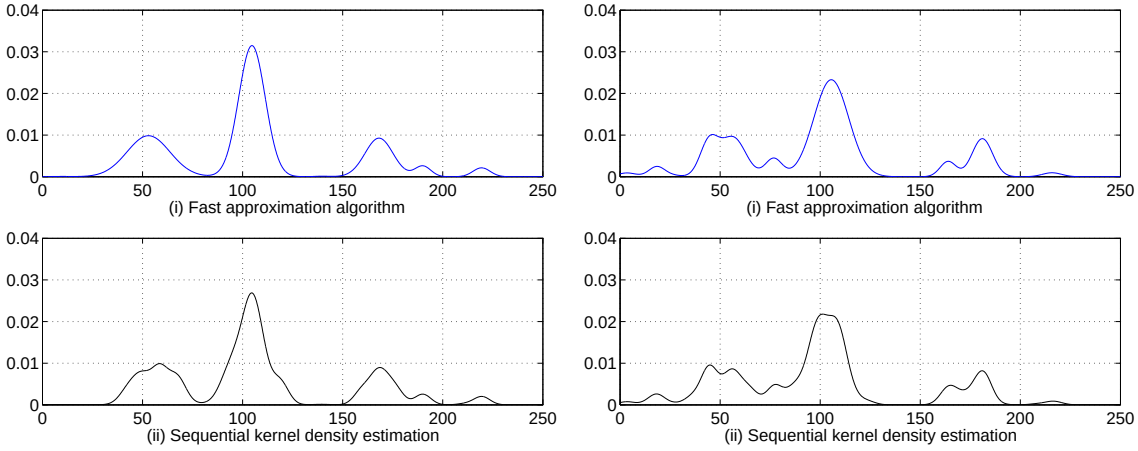
We performed the same experiment using the method suggested in [34] in order to compare the performance of our method with that one. The same initial density function is used except that the third Gaussian component with the large variance (100^2) is replaced by a uniform distribution in $[0, 255]$. The first and second Gaussians are assumed to be wandering and stable component, respectively. The sampling density function is identical, and the same input sequence is used. As mentioned earlier, the method from [34] cannot easily deal with complicated models because it requires a fixed number of components in the density and only one Gaussian is assigned to the stable component. During most time steps of the simulation, the number of modes is actually one, and the wandering component has negligible weight. In other words, the multi-modality of the density function is not well preserved, so the 3-component model might not be sufficient on real data. Figure 4.3 shows the sequence of density functions using the method of [34]; a significant difference of the estimated density function from Figure 4.1 can be observed.

Multi-dimensional simulations were also performed, and similar results were observed. Figures 4.4 and 4.5 present the simulation results, showing the comparisons of MISE and number of components, respectively.

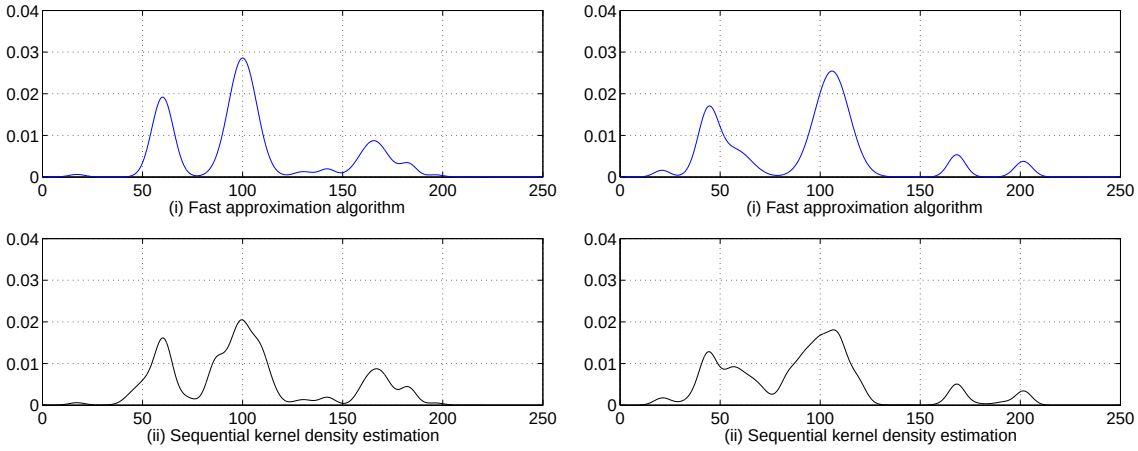
To compare the speed between the linear and quadratic time algorithms, the



(a) $t = 1$ (b) $t = 60$

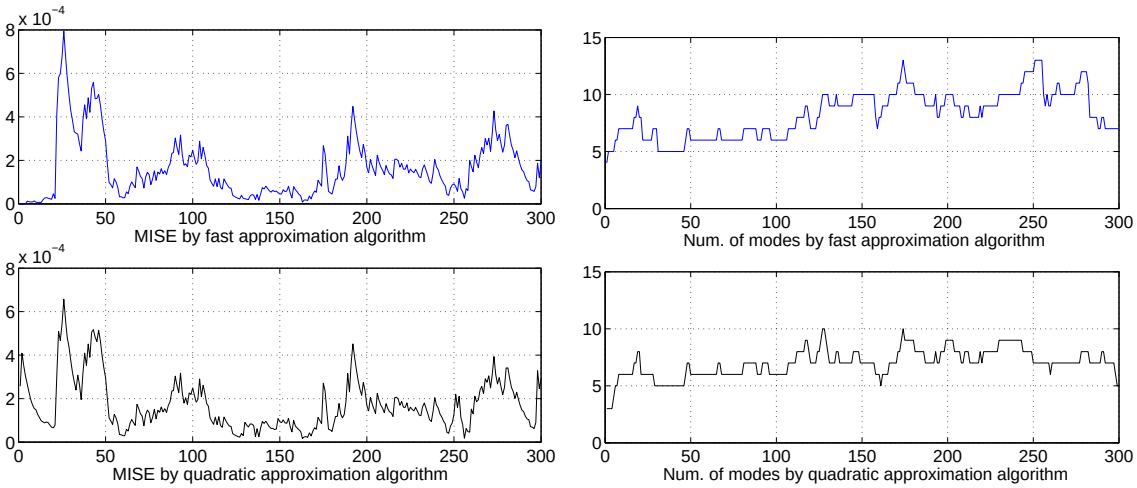


(c)$t = 120$ (d) $t = 180$



(e) $t = 240$ (f) $t = 300$

Figure 4.1: Simulation of fast approximation algorithm and comparison with sequential kernel density estimation. (a)-(f) Fast sequential density approximation (top) vs. sequential kernel density estimation (bottom)



(a) MISE (b) Number of components

Figure 4.2: Comparison of MISE and number of components in each step of 1D simulation between fast approximation algorithm (top) and quadratic time algorithm (bottom).

CPU time for the one-step sequential density approximation procedure was measured for density functions with different numbers of components. Since sequential density estimation contains matrix operations, it is also worthwhile to check the performance with respect to dimensionality. So, our experiments are performed by varying the number of modes and the dimensionality. We performed comparisons for two different dimensionalities (2D and 6D), and the results are presented in Figure 4.6. We observe that the running time of the fast approximation algorithm is significantly less than the quadratic algorithm.

Finally, we performed 100-step sequential density estimations in various dimensions, and computed the average CPU time and MISE[2]. Figure 4.7 illustrates that the fast approximation algorithm is faster and comparable in accuracy.

[2]The MISE is computed only at sample locations in this case to handle high dimensional examples.

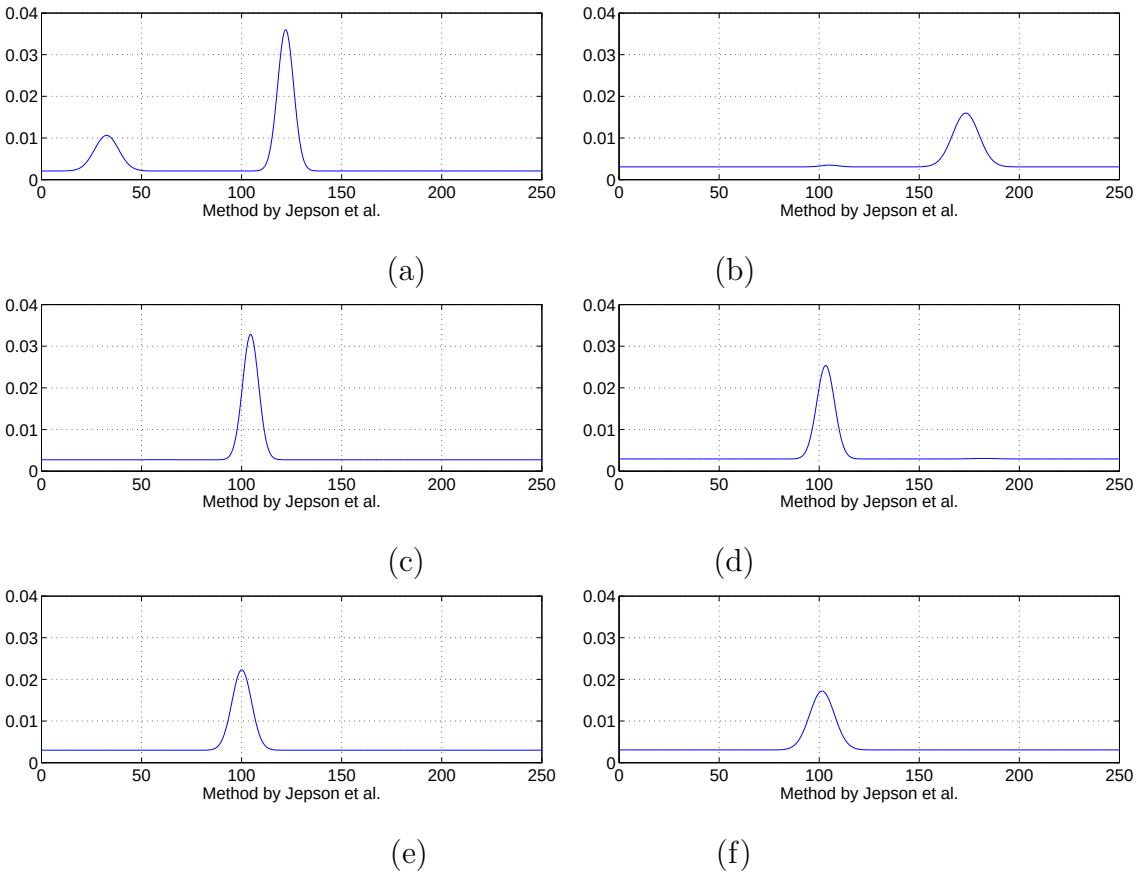


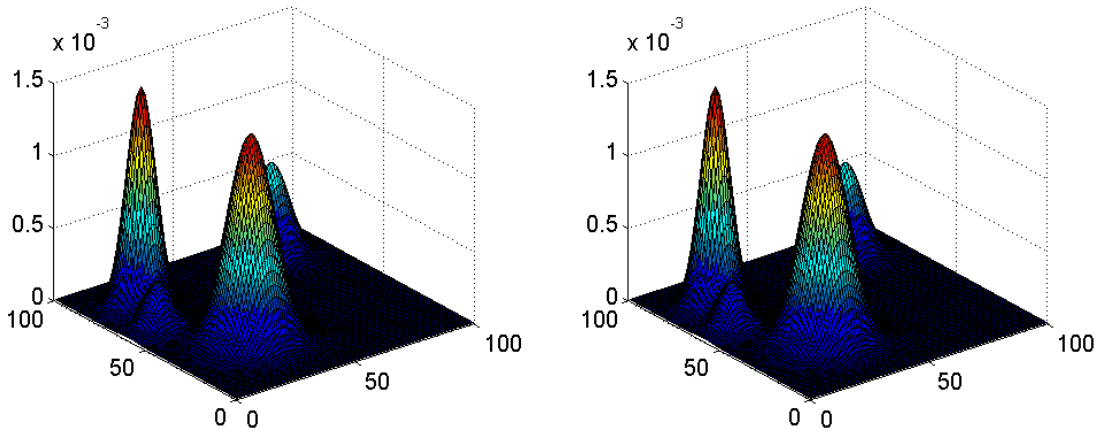
Figure 4.3: Simulation of Jepson et al.'s method [34] (a)-(f) Density function at time 1, 60, 120, 180, 240, and 300.

4.3 On-line Appearance Model

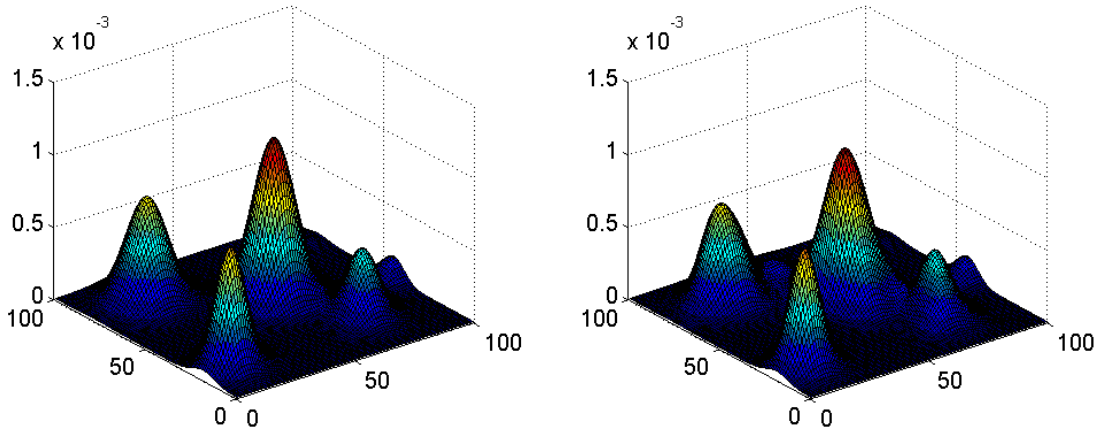
On-line appearance models are important for real-time object tracking, since the target model can change due to many factors. In this section, we present a density based target modeling and on-line model update algorithm to deal with changes in target appearance and size.

4.3.1 Target Modeling

We construct the model of features for each pixel in the target object with a mixture of Gaussians, so the target model is represented as a set of density functions.

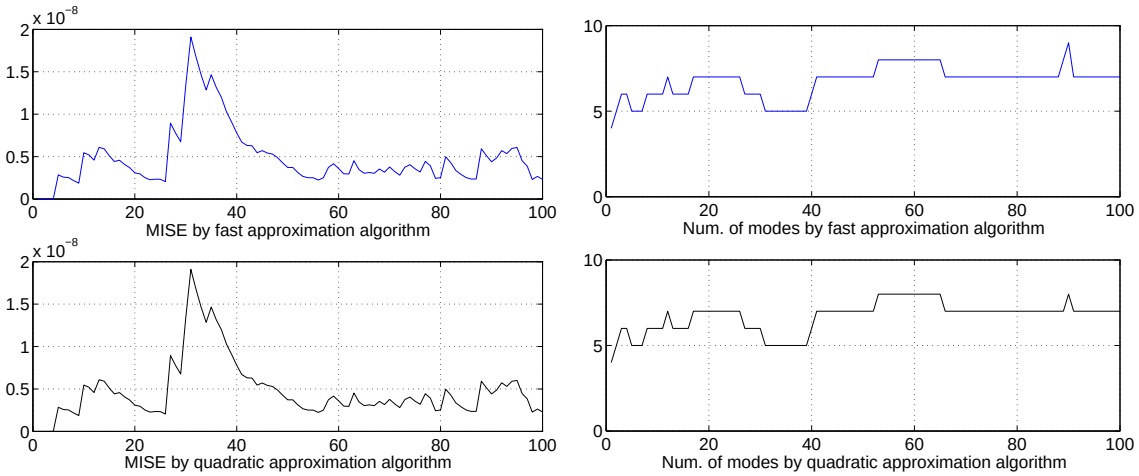


(a) $t = 1$



(b) $t = 100$

Figure 4.4: Simulation of fast sequential density approximation (left) and comparison with sequential kernel density estimation (right)



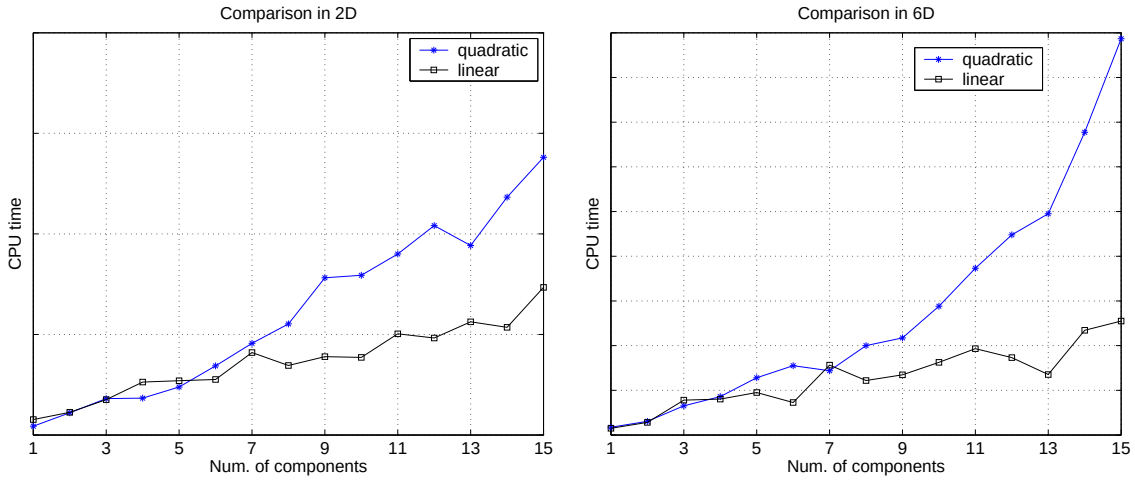
(a) MISE (b) Number of components

Figure 4.5: Comparison of MISE and number of components in each step of 2D simulation between fast approximation algorithm (top) and quadratic time algorithm (bottom).

Our representation can include an arbitrary number of Gaussian components in the density function, and describes the underlying density accurately.

Using pixel-wise color density modeling has the advantage of describing the details of target region, but does not capture any structural aspects of an object's appearance. So, we also incorporate rectangular features into our target model. These features are obtained by averaging the intensities of neighbors (e.g., 3×3 or 5×5) in each color channel (r, g, b) and are computed efficiently with *integral images* [80]. Since rectangular features encode the spatial information around a pixel, they ameliorate some problems caused by non-rigid motions of objects and pixel misregistrations. The performance of such features have previously been investigated in object tracking [24] and detection [80].

Initially, at time 0, the density function for each pixel (i, j) within a selected



(a) 2D (b) 6D

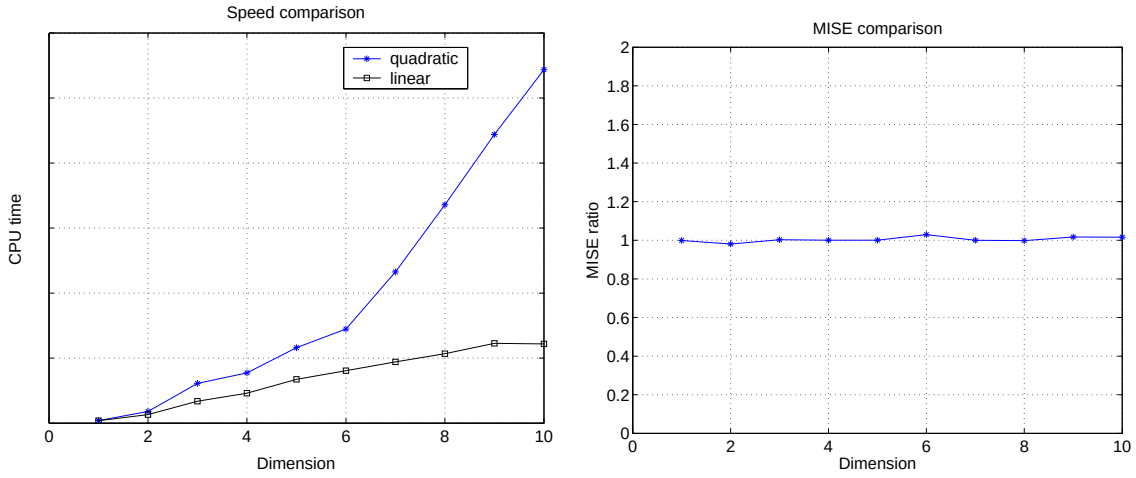
Figure 4.6: CPU time of fast approximation algorithm and quadratic time algorithm.

target region has a single Gaussian component $N(1, \mathbf{x}_0(i,j), \mathbf{P}_0(i,j))$ whose mean $\mathbf{x}_0(i,j)$ is the combination of color at (i,j) and average color of its neighborhood. In each time step t, the new data $\mathbf{x}_t(i,j)$ at the pixel location (i,j) is denoted as

$$\mathbf{x}_t(i,j) = (r, g, b, \bar{r}, \bar{g}, \bar{b}) \tag{4.7}$$

where $(\bar{r}, \bar{g}, \bar{b})$ denotes the average intensity of the neighborhood centered at (i,j) for each color channel. Note that $\mathbf{x}_t(i,j)$ would be a 2D vector composed of intensity and average intensity for gray scale images.

Whenever a new observation is integrated into the current density, the density function is updated as explained in Section 4.2.2. As time progresses, highly weighted Gaussian components are constructed around frequently observed data, and several minor modes may also develop. Using exponential updating, old information is removed gradually if no further observations are around it. So, the representation maintains a history of feature observations for any given pixel.



(a) (b)

Figure 4.7: CPU time and MISE with respect to dimensionality in sequential density approximation. (MISE ratio is error ratio of fast approximation algorithm to quadratic time algorithm.)

4.3.2 Update in Scale Space

The target model so constructed is robust to changes of feature values, but is not intended to handle scale change. Tracking may fail in sequences containing large scale changes of target objects, since the observations are severely affected by drastic up- or down-sampling. So, we update the size of the target model at every $\beta\%$ scale change as follows: the pixel location in the new target window is projected into the old target window, and the density function is computed by a weighted sum of neighborhood density functions as

$$\hat{f}^{\star}_{\mathbf{X}}(i,j) = \sum_{(u,v)\in N(i,j)} w(u,v)\hat{f}_{\mathbf{X}}(u,v) \tag{4.8}$$

where $N(i,j)$ is a set of pixels adjacent to (i,j)'s projection onto the old target window, $\hat{f}_{\mathbf{X}}(u,v)$ is an estimated density function for feature vector $\mathbf{x}$ at (u,v), and $w(u,v)$ is the normalized weight associated with each density function $\hat{f}_{\mathbf{X}}(u,v)$. The

new density function $\hat{f}^{\star}_{\mathbf{X}}(i,j)$ is also a mixture of Gaussians, and the batch version of our density approximation technique is applied to reduce the number of components for a compact representation.

The modeling error in scale change is fairly small because of the spatial coherence of the target. Also, since the rectangular features for adjacent pixels are based on highly overlapping areas, they are robust to updates in scale space. This appearance model update in scale space is simple, but performs well in experiments; the strategy plays an important role in the examples displaying significant scale change.

4.4 Object Tracking

In this section, the criterion to determine the optimal parameters for object tracking is described, and we present various tracking examples showing the effectiveness of on-line density-based appearance modeling. Comparisons with other appearance modeling methods are also performed.

4.4.1 Maximum Likelihood Parameter Optimization

Let $\mathbf{M}(\mathbf{x};\mathbf{p})$ denote the parameterized motion of location $\mathbf{x}$, where $\mathbf{p} = (v_x, v_y, s)$ is a vector for translations and scaling. Denote by $d(\mathbf{M}(\mathbf{x};\mathbf{p}))$ the observation at the image of $\mathbf{x}$ under motion $\mathbf{p}$. Now, the tracking problem involves finding the optimal parameter using the maximum likelihood method.

$$\mathbf{p}_t = \arg\max_{\mathbf{p}} \sum_{(i,j)\in R} \log(p(d(\mathbf{M}(\mathbf{x}^{(i,j)};\mathbf{p}))|\mathbf{p}_{t-1}, A_t^{(i,j)})) \tag{4.9}$$

where (i,j) is the relative location in the target region R, and $A_t^{(i,j)}$ the current appearance model at (i,j). A simple gradient-based tracking method is employed

to find the optimal $\mathbf{p}$, and the target model is updated as explained in Section 4.3.

4.4.2 Experiments

Various sequences are tested to illustrate the performance of our on-line appearance modeling technique. New data is integrated with the weight of $\alpha = 0.05$. Also, the target model is updated in scale space at every $\beta = 10\%$ change of size, and a slightly higher learning rate $\alpha = 0.1$ is used at this time. For the rectangular features, 5×5 neighborhood pixels are used.

Figure 4.8 shows the results on the *tank* sequence, in which the target has low contrast and changes its orientation during tracking. Our tracking algorithm succeeds in tracking for 940 frames even with transient severe noise levels due to dust (e.g., Figure 4.8 (b) and (d)).

The results on a *person* sequence are presented next. In this sequence, a human face is tracked with a large change of face orientation and lighting condition; the results are shown in Figure 4.9. Figure 4.9 (e) illustrates how the appearance of the face changes over time; Figure 4.9 (f) shows the average intensity of the target region over time.

In Figure 4.10, the tracking results for the head of a football player are shown. In this sequence, the target object moves fast in high clutter and is blurred frequently. Also, the changes in orientation and texture of the head make tracking difficult, so the density function for the target model must be able to accommodate those variations for accurate tracking. As shown in Figure 4.10 (f), the average number of modes per pixel (blue solid line) and the standard deviation (black dotted

(a) $t = 1$ (b) $t = 590$

(c) $t = 940$ (d) Target appearance changes

Figure 4.8: Tracking result of *tank* sequence

line) varies up to around 6, which suggests that a density function with a fixed number of components may produce a high tracking error compared with our method.

In Figure 4.11, a gray scale image sequence involving a large scale and illumination change is presented, and our appearance model update strategy adapts to this well.

In the last sequence, the appearance of the car changes frequently because of the shadow and its red lights. Also, a person passes in front of the car and the image

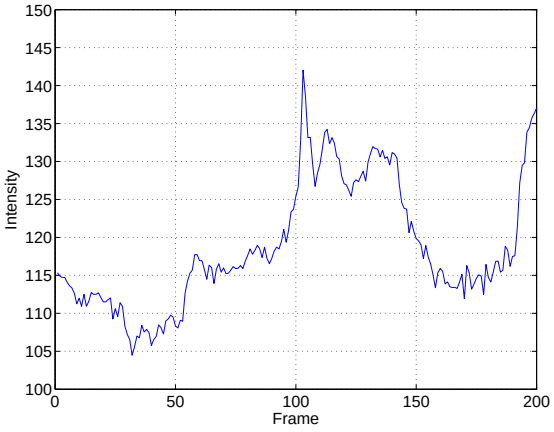
(a) $t = 1$

(b) $t = 36$

(c) $t = 150$

(d) $t = 205$

(e) Target appearance changes



(f) Intensity changes

Figure 4.9: Tracking results of *person* sequence

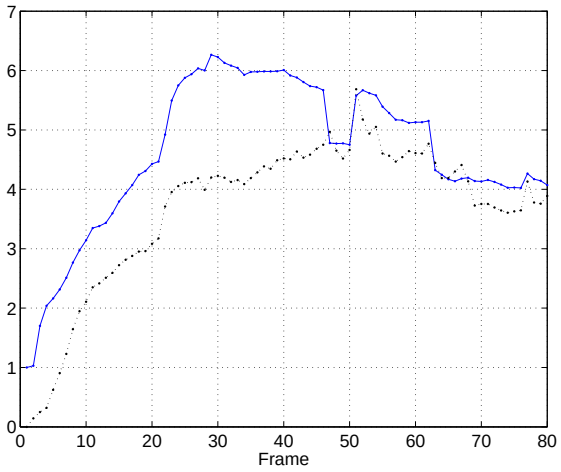
(a) $t = 1$

(b) $t = 22$

(c) $t = 47$

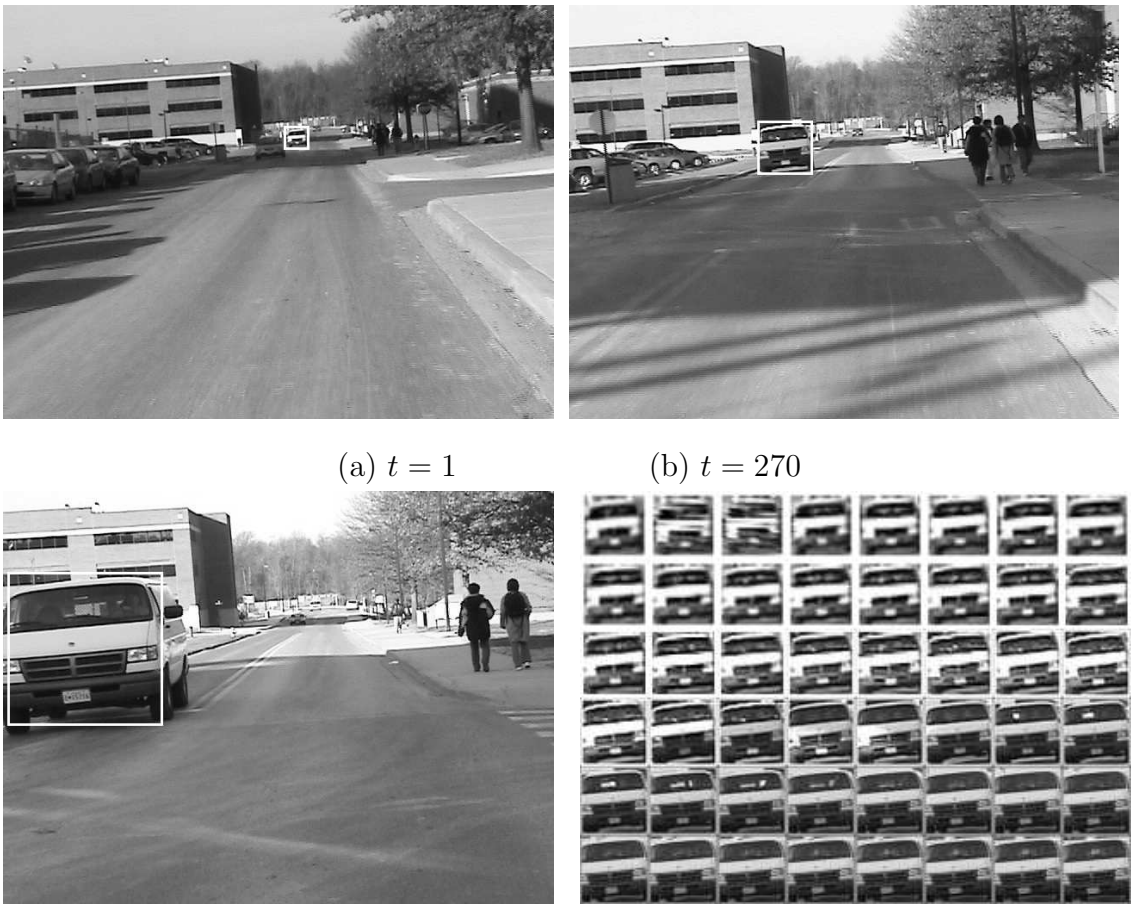
(d) $t = 67$

(e) Target appearance changes



(f) Num. of components

Figure 4.10: Tracking result of *football* sequence



(a) $t = 1$ (b) $t = 270$

(c) $t = 500$ (d) Target appearance changes

Figure 4.11: Tracking result of *car1* sequence

is severely shaken several times by camera movement. Figure 4.12 shows that our on-line density-based modeling is successful in spite of occlusion and appearance change.

4.4.3 Comparison with Other Methods

Two different appearance modeling methods are implemented for comparison; one is fixed modeling with a Gaussian distribution and the other is 3-component mixture model based on [34]. For each algorithm, two different feature sets – color only and color with rectangular feature for each pixel – are employed. Using the

(a) $t = 1$ (b) $t = 18$

(c) $t = 192$ (d) Target appearance changes

Figure 4.12: Tracking result of *car2* sequence

same gradient-based tracking, six different cases including two for our algorithm are tested, and the results are summarized in table 4.1. Also, some examples of failures are illustrated in Figure 4.13.

As table 4.1 illustrates, our on-line density approximation shows good performance compared with other parametric techniques. The only algorithm able to successfully track through both sequences was our method using both pixel-wise and rectangular features.

Table 4.1: Comparison of tracking results

Modeling method		*person* (205 frames)	*football* (80 frames)
Fixed Gaussian	C[a]	fail	fail
	C+R[b]	fail	fail
3-component mixture [34]	C	succeed	fail
	C+R	succeed	fail
Our method	C	succeed	fail
	C+R	succeed	succeed

[a]color feature

[b]color rectangular feature

4.5 Conclusions

We described a sequential density approximation algorithm in which the density function is represented with a mixture of Gaussians whose number, mean, covariance and weight are automatically determined. This algorithm has linear time complexity, and can be used in many real-time applications. We apply this technique to the adaptive target appearance modeling using pixel-wise and rectangular features for object tracking. The effectiveness of the fast approximation algorithm and combined feature is presented by various simulations and tracking results in natural videos.

Figure 4.13: Examples of tracking failure. Tracking for *person* sequence with the fixed Gaussian modeling (C+R, $t = 120$) (left) and for *football* sequence with the 3-component mixture modeling (C, $t = 59$) (right)

Chapter 5

Kernel-Based Bayesian Filtering

Particle filtering provides a general framework for propagating probability density functions in non-linear and non-Gaussian systems. However, the algorithm is based on a Monte Carlo approach and sampling is a problematic issue, especially for high dimensional problems. This chapter presents a new kernel-based Bayesian filtering framework, which adopts an analytic approach to better approximate and propagate density functions. In this framework, the techniques of *density interpolation* and *density approximation* are introduced to represent the likelihood and the posterior densities by Gaussian mixtures, where all parameters such as the number of mixands, their weight, mean, and covariance are automatically determined. The proposed analytic approach is shown to perform sampling more efficiently in high dimensional space. We apply our algorithm to real-time tracking problems, and demonstrate its performance on real video sequences as well as synthetic examples.

5.1 Introduction

Particle filtering is a Monte Carlo approach to solve the recursive Bayesian filtering problem. Although it provides tractable solutions to non-linear and non-Gaussian systems, it is faced with practical issues such as sample degeneracy and sample

impoverishment [2]. Moreover, to achieve reliable filtering, the sample size can grow exponentially as the dimension of the state space increases. To overcome these issues, we explore an analytic approach to approximate density functions and introduce a new kernel-based filtering scheme. The main idea of this work is to maintain an analytic representation of relevant density functions and propagate them over time. In this chapter, kernel-based density representation is adopted.

5.1.1 Related Work

There have been many parametric density representations proposed for various applications. In [45, 74], the authors suggest Gaussian mixture models, but their method requires knowledge of the number of components, which is difficult to know in advance. A more elaborate density representation is described in [34], where a 3-component mixture is used for target modeling in object tracking, but this approach cannot overcome the drawback of parametric methods. Kernel density estimation [22] is a widely used non-parametric approach in computer vision. Its major advantage is the flexibility to represent very complicated densities effectively. But its very high memory requirements and computational complexity inhibit the use of this method.

For Bayesian filtering, Cham and Rehg [5] introduce a piecewise Gaussian function to specify the tracker state, in which the selected Gaussian components characterize the neighborhoods around the modes. This idea is applied to multiple hypothesis tracking in a high dimensional space body tracker, but the sampling and the posterior computation are not straightforward. The closest work to ours is

[40] where the posterior is represented with a Gaussian mixture in a particle filter framework. However, this solution may not provide a compact representation for the posterior, and the prediction and the update steps are oversimplified.

5.1.2 Our Approach

In this chapter , we extend our previous work [28] which provides the framework of Kernel-based Bayesian filtering. We introduce density approximation and density interpolation to represent density functions efficiently and effectively. In both techniques, the density function is represented by a Gaussian mixture, where the number of mixands, their weights, means and covariances are automatically determined. The density approximation is based on a mode finding algorithm [12, 14] derived from variable-bandwidth mean-shift which provides the methodology to construct a compact representation with a small number of Gaussian kernels. A density interpolation technique is introduced to obtain a continuous representation of the measurement likelihood function. Unscented transformation (UT) [35, 46] is also adopted to deal with non-linear state transition models. These techniques are integrated into the Bayesian filtering framework. In the new kernel-based Bayesian filtering algorithm, the continuous representations of density functions are propagated over time.

The advantage of maintaining an analytic representation of density functions lies in efficient sampling. This is important for solving high dimensional problems. A multi-stage sampling strategy is introduced in density interpolation for accurate approximation of the measurement likelihood function. The new algorithm is applied

to real-time object tracking, and its performance is demonstrated through various experiments.

This chapter is organized as follows. Section 5.2 introduces the new density propagation technique in the Bayesian filtering framework. Section 5.3 and 5.4 explain the density approximation and the density interpolation method, respectively. Section 5.5 demonstrates its performance by various simulation results with synthetic examples. Finally, it is demonstrated in Section 5.6 how our algorithm can be applied to object tracking in real videos.

5.2 Bayesian Filtering

In this section, we introduce the new Bayesian filtering framework, where the relevant density functions are approximated by kernel-based representations and propagated over time.

5.2.1 Overview

In a dynamic system, the process and measurement model are given by

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{5.1}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{v}_t) \tag{5.2}$$

where $\mathbf{v}_t$ and $\mathbf{u}_t$ are the process and the measurement noise, respectively. The state variable $\mathbf{x}_t$ $(t = 0, \ldots, n)$ is characterized by its probability density function estimated from the sequence of measurements $\mathbf{z}_t$ $(t = 1, \ldots, n)$. In the sequential Bayesian filtering framework, the conditional density of the state variable given the

measurements is propagated through prediction and update stages,

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1} \tag{5.3}$$

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{1}{k}p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \tag{5.4}$$

where $k = \int p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})d\mathbf{x}_t$ is a normalization constant independent of $\mathbf{x}_t$. $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$ is the prior probability density function (pdf), $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ is the predicted pdf and $p(\mathbf{z}_t|\mathbf{x}_t)$ is the measurement likelihood function. The posterior pdf at time step t, $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, is used as the prior pdf in time step $t+1$.

At each time step, the conditional distribution of the state variable $\mathbf{x}$ given a sequence of measurements $\mathbf{z}$ is represented by a Gaussian mixture. Our goal is to retain such a representation through the stages of prediction and update, and to represent the posterior probability in the following step with the same mixture form.

The proposed filtering framework is described as follows. First, unscented transformation (UT) [35, 46] is used to derive a mixture representation of the predicted pdf $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$. Second, the density interpolation technique with multi-stage sampling is introduced to approximate the likelihood function with a mixture form. By multiplying two mixture functions, the posterior pdf is obtained through equation (5.4). To prevent the number of mixands from growing too large, an algorithm of density approximation based on mode finding is applied to derive a compact representation for the posterior pdf.

5.2.2 Prediction by Unscented Transform

Denote by $\mathbf{x}_t^i$ $(i = 1, \ldots, n_t)$ a set of means in R^d and by $\mathbf{P}_t^i$ the corresponding covariance matrices at time step t. Let each Gaussian have a weight κ_t^i with $\sum_{i=1}^{n_t} \kappa_t^i = 1$, and let the prior density function be given by

$$p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{n_{t-1}} \frac{\kappa_{t-1}^i}{\mid \mathbf{P}_{t-1}^i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_{t-1}^i, \mathbf{P}_{t-1}^i\right)\right) \tag{5.5}$$

The unscented transformation [35, 46] is a method for calculating the statistics of a random variable which undergoes a non-linear transformation.

$$\begin{aligned}
\mathcal{X}_{t-1}^{(i,0)} &= \mathbf{x}_{t-1}^i \\
\mathcal{X}_{t-1}^{(i,j)} &= \mathbf{x}_{t-1}^i - (\sqrt{(d+\lambda)\mathbf{P}_{t-1}^i})_j \;\; j = 1, \ldots, d \\
\mathcal{X}_{t-1}^{(i,j)} &= \mathbf{x}_{t-1}^i + (\sqrt{(d+\lambda)\mathbf{P}_{t-1}^i})_{j-d} \;\; j = d+1, \ldots, 2d \\
W_{(i,0)} &= \lambda/(d+\lambda) \\
W_{(i,j)} &= 1/2(d+\lambda) \quad j = 1, \ldots, 2d
\end{aligned} \tag{5.6}$$

where λ is a scaling parameter and $(\sqrt{(d+\lambda)\mathbf{P}_{t-1}^i})_j$ is the ith row or column of the matrix square root of $(d+\lambda)\mathbf{P}_{t-1}^i$. $W_{(i,j)}$ is the weight associated with the j-th sigma point where $\sum_{j=0}^{2d} W_{(i,j)} = 1$. These sigma vectors are propagated through the non-linear function,

$$\mathcal{X}_t^{(i,j)} = g(\mathcal{X}_{t-1}^{(i,j)}) \;\; i = 0, \ldots, 2d \tag{5.7}$$

and the mean and covariance for $\bar{\mathbf{x}}_t^i$ are approximated using a weighted sample mean and covariance of the posterior sigma points,

$$\bar{\mathbf{x}}_t^i = \sum_{i=0}^{2d} W_{(i,j)} \mathcal{X}_t^{(i,j)}$$

$$\bar{\mathbf{P}}_t^i = \sum_{i=0}^{2d} W_{(i,j)} (\mathcal{X}_t^{(i,j)} - \bar{\mathbf{x}}_t^i)(\mathcal{X}_t^{(i,j)} - \bar{\mathbf{x}}_t^i)^\top + \mathbf{Q} \tag{5.8}$$

where $\mathbf{Q}$ is the covariance matrix for the process noise.

For each mode in the prior, UT is applied independently and the density after prediction is as follows.

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{n_{t-1}} \frac{\bar{\kappa}_t^i}{\mid \bar{\mathbf{P}}_t^i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \bar{\mathbf{x}}_t^i, \bar{\mathbf{P}}_t^i\right)\right) \tag{5.9}$$

where $\bar{\kappa}_t^i = \kappa_{t-1}^i$. This non-linear transformation is guaranteed to be accurate up to the second order of the Taylor expansion.

5.2.3 Multi-stage Sampling and Interpolation of Measurement Likelihood

In contrast to various particle filters, we represent the measurement likelihood function in an analytic form. A continuous approximation of the likelihood function is interpolated from discrete samples. A multi-stage sampling scheme is introduced to improve the approximation progressively. The advantage of the analytic representation is that it provides a global view of the landscape of the likelihood function and thus enables efficient sample placement.

Multi-stage sampling

Unlike the SIR algorithm [31] which uses the predicted pdf as the proposal distribution, we employ the multi-stage sampling strategy and progressively update the proposal function based on observations. The predicted pdf is used as the initial proposal distribution q_0.

$$q_0(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \tag{5.10}$$

Assume that, in total, N samples are to be drawn to obtain measurement data. In our multi-stage sampling scheme, N/m samples are drawn in the first stage from the initial proposal distribution (5.10), where m is the number of sampling stages. An initial approximation of the likelihood function $p_1(\mathbf{z}_t|\mathbf{x}_t)$ is obtained through surface interpolation with Gaussian kernels. Details of the density interpolation algorithm are provided in Section 5.4. The proposal function is then updated by a linear combination of the initial proposal distribution and the current approximation of the likelihood function $p_1(\mathbf{z}_t|\mathbf{x}_t)$. We repeatedly approximate the likelihood function from available samples and update the proposal distribution.

$$p_j(\mathbf{z}_t|\mathbf{x}_t) = \sum_{\tau_i \neq 0} \tau_t^i \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}_t, \mathbf{x}_t^i, \mathbf{R}_t^i\right)\right) \tag{5.11}$$

$$q_j(\mathbf{x}_t) = (1-\alpha_j) q_{j-1}(\mathbf{x}_t) + \alpha_j \frac{p_j(\mathbf{z}_t|\mathbf{x}_t)}{\int p_j(\mathbf{z}_t|\mathbf{x}_t) d\mathbf{x}_t} \tag{5.12}$$

where $i = 1, \ldots, \frac{j}{m}N$, $j = 1, \ldots, m$, and $\alpha_j \in [0,1]$ is the *adaptation rate*.

Since the information of the observation is incorporated into the proposal distribution to guide sampling, the multi-stage sampling strategy explores the likelihood surface more efficiently than conventional particle filters. Thus, it is especially advantageous in dealing with a high dimensional state space.

Approximation of likelihood function

As discussed previously, the measurement likelihood is estimated through multi-stage sampling. With samples drawn from the improved proposal distributions, intermediate likelihood functions are constructed and used to update the proposal distributions. After m-step repetition of this procedure, the final measurement

distribution is obtained.

Algorithm 2 Measurement Step

1: $S_t = \phi$

2: $q_0(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$

3: **for** $i = 1$ to m **do**

4: draw sample s from proposal distribution

$S_t^i = \{s_i^{(j)} | s_i^{(j)} \sim q_{i-1}(\mathbf{x}_t), j = 1, \ldots, N/m\}$

5: $S_t = S_t \cup S_t^i$

6: assign mean and covariance for the element in S_t^i

$\mathbf{m}_{(i-1)\frac{N}{m}+j} = s_i^{(j)}$

$\mathbf{Q}_{(i-1)\frac{N}{m}+j} = c\,\mathrm{diag}(\mathrm{KNN}_1(k)\ldots\mathrm{KNN}_d(k))^2\,\mathbf{I}$ (5.25)

7: compute likelihood of each new sample

$l_{(i-1)\frac{N}{m}+j} = h(\mathbf{m}_{(i-1)\frac{N}{m}+j}, \mathbf{v}_t)$

8: compute $\mathbf{A}$ and $\mathbf{b}$ for every element in S_t

9: $\mathbf{w} = nnls(\mathbf{A}, \mathbf{b})$ (5.27)

10: $p_i(\mathbf{z}_t|\mathbf{x}_t) = \sum_{\tau_t^j \neq 0} N(\tau_t^j, \mathbf{x}_t^j, \mathbf{R}_t^j)$

where $\tau_t = \mathbf{w}$, $\mathbf{x}_t = \mathbf{m}$, and $\mathbf{R}_t = \mathbf{Q}$

11: $q_i(\mathbf{x}_t) = (1 - \alpha_i) q_{i-1}(\mathbf{x}_t) + \alpha_j \frac{p_i(\mathbf{z}_t|\mathbf{x}_t)}{\int p_i(\mathbf{z}_t|\mathbf{x}_t) d\mathbf{x}_t}$ (5.12)

12: **end for**

13: $p(\mathbf{z}_t|\mathbf{x}_t) = p_m(\mathbf{z}_t|\mathbf{x}_t)$

Algorithm 2 presents the complete procedure to compute the likelihood function, and the final measurement function with m_t Gaussians at time t is given by

$$p(\mathbf{z}_t|\mathbf{x}_t) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{m_t} \frac{\tau_t^i}{|\,\mathbf{R}_t^i\,|^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}_t, \mathbf{x}_t^i, \mathbf{R}_t^i\right)\right) \tag{5.13}$$

where τ_t^i, $\mathbf{x}_t^i$ and $\mathbf{R}_t^i$ are the weight, mean and covariance matrix of the i-th kernel.

5.2.4 Update

Since both the predicted pdf and the measurement functions are represented by Gaussian mixtures, the posterior pdf, as the product of two Gaussian mixtures, can also be represented by a Gaussian mixture. Denote the Gaussian components of the predicted pdf and the likelihood function by $N(\bar{\kappa}_t^i, \bar{\mathbf{x}}_t^i, \bar{\mathbf{P}}_t^i)$ $(i = 1, \ldots, n_{t-1})$ and $N(\tau_t^j, \mathbf{x}_t^j, \mathbf{R}_t^j)$ $(j = 1, \ldots, m_t)$ respectively, the product of the two distributions is as follows.

$$\left(\sum_{i=1}^{n_{t-1}} N(\bar{\kappa}_t^i, \bar{\mathbf{x}}_t^i, \bar{\mathbf{P}}_t^i)\right)\left(\sum_{j=1}^{m_t} N(\tau_t^j, \mathbf{x}_t^j, \mathbf{R}_t^j)\right) = \sum_{i=1}^{n_{t-1}}\sum_{j=1}^{m_t} N(\bar{\kappa}_t^i \tau_t^j, \mathbf{m}_t^{ij}, \mathbf{\Sigma}_t^{ij}) \tag{5.14}$$

where

$$\mathbf{m}_t^{ij} = \mathbf{\Sigma}_t^{ij}((\bar{\mathbf{P}}_t^i)^{-1}\mathbf{x}_t^i + (\mathbf{R}_t^j)^{-1}\mathbf{x}_t^j) \tag{5.15}$$

$$\mathbf{\Sigma}_t^{ij} = ((\bar{\mathbf{P}}_t^i)^{-1} + (\mathbf{R}_t^j)^{-1})^{-1} \tag{5.16}$$

The resulting density function in (5.14) is a weighted Gaussian mixture. However, the exponential increase in the number of components over time could make the whole procedure intractable. In order to avoid this situation, a density approximation technique is proposed to maintain a compact yet accurate density representation even after density propagation through many time steps. Details of the density approximation algorithm is given in Section 5.3.

After the update step, the final posterior distribution is given by

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{1}{(2\pi)^{d/2}}\sum_{i=1}^{n_t}\frac{\kappa_t^i}{\mid \mathbf{P}_t^i \mid^{1/2}}\exp\left(-\frac{1}{2}D^2\left(\mathbf{x}, \mathbf{x}_t^i, \mathbf{P}_t^i\right)\right) \tag{5.17}$$

where n_t is the number of components at time step t.

5.3 Density Approximation

In this section, we review an iterative procedure of mode detection derived from variable-bandwidth mean-shift [14], and density approximation using the mode detection technique [28].

5.3.1 Mode Detection and Density Approximation

Suppose that $N(\kappa_i, \mathbf{x}_i, \mathbf{P}_i)$ $(i = 1 \dots n)$ is a Gaussian kernel with weight κ_i, mean $\mathbf{x}_i$, and covariance $\mathbf{P}_i$ in the d-dimensional state space, where $\sum_{i=1}^{n} \kappa_i = 1$. Then, we define the sample point density estimator computed at point $\mathbf{x}$ by

$$\hat{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{n} \frac{\kappa_i}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \tag{5.18}$$

where

$$D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right) \equiv (\mathbf{x} - \mathbf{x}_i)^\top \mathbf{P}_i^{-1} (\mathbf{x} - \mathbf{x}_i). \tag{5.19}$$

Our purpose is to obtain a compact representation of the density function which is a Gaussian mixture. The mode location and its weight are found by a mean-shift algorithm, and the covariance matrix associated with each mode is computed using the Hessian matrix.

To find the gradient ascent direction at $\mathbf{x}$, the variable-bandwidth mean-shift vector at $\mathbf{x}$ is given by

$$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^{n} \omega_i(\mathbf{x}) \mathbf{P}_i^{-1}\right)^{-1} \left(\sum_{i=1}^{n} \omega_i(\mathbf{x}) \mathbf{P}_i^{-1} \mathbf{x}_i\right) - \mathbf{x} \tag{5.20}$$

where the weights

$$\omega_i(\mathbf{x}) = \frac{\kappa_i \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)}{\sum_{i=1}^{n} \kappa_i \mid \mathbf{P}_i \mid^{-1/2} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right)} \tag{5.21}$$

satisfy $\sum_{i=1}^{n} \omega_i(\mathbf{x}) = 1$. By computing the mean-shift vector $\mathbf{m}(\mathbf{x})$ and translating the location $\mathbf{x}$ by $\mathbf{m}(\mathbf{x})$ iteratively, a local maximum of the underlying density function is detected. A formal check for the maximum involves the computation of the Hessian matrix

$$\hat{\mathbf{H}}(\mathbf{x}) = \frac{1}{(2\boldsymbol{\pi})^{d/2}} \sum_{i=1}^{n} \frac{\kappa_i}{\mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i\right)\right) \times \mathbf{P}_i^{-1}\Big((\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^{\top} - \mathbf{P}_i\Big)\mathbf{P}_i^{-1} \tag{5.22}$$

which should be negative definite. If it is not negative definite, the convergence point might be a saddle point or a local minimum. In this case, kernels associated with such modes should be restored and considered as separate modes for further processing.

The approximate density is obtained by detecting the mode location for every sample point $\mathbf{x}_i$ and assigning a single Gaussian kernel for each mode. Suppose that the approximate density has n' unique modes of $\tilde{\mathbf{x}}_j$ $(j = 1 \ldots n')$ with associated weight $\tilde{\kappa}_j$ which is equal to the sum of the kernel weights converged to $\tilde{\mathbf{x}}_j$. The Hessian matrix $\hat{\mathbf{H}}_j$ of each mode is used for the computation of $\tilde{\mathbf{P}}_j$ as follows.

$$\tilde{\mathbf{P}}_j = \frac{\tilde{\kappa}_j^{\frac{2}{d+2}}}{\mid 2\pi(-\hat{\mathbf{H}}_j^{-1}) \mid^{\frac{1}{d+2}}}(-\hat{\mathbf{H}}_j^{-1}) \tag{5.23}$$

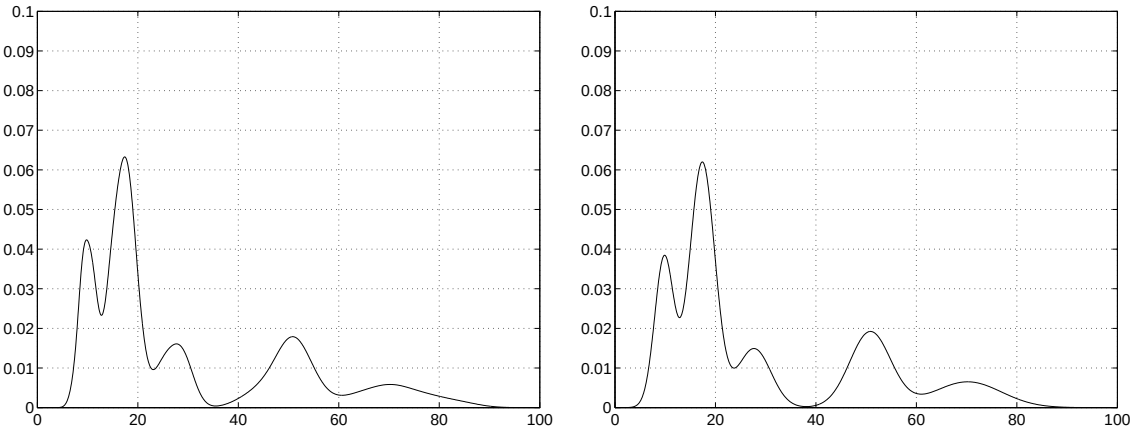
The basic idea of equation (5.23) is to fit the covariance using the curvature in the neighborhood of the mode. The final density approximation is then given by

$$\tilde{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{n'} \frac{\tilde{\kappa}_i}{\mid \tilde{\mathbf{P}}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2\left(\mathbf{x}, \tilde{\mathbf{x}}_{t+1}^i, \tilde{\mathbf{P}}_{t+1}^i\right)\right) \tag{5.24}$$

and $n' \ll n$ is satisfied in most cases. The approximation error $||\hat{f}(\mathbf{x}) - \tilde{f}(\mathbf{x})||$ can be evaluated straightforwardly.

5.3.2 Performance of Approximation

The accuracy of the density approximation is demonstrated in Figure 5.1. From a one-dimensional distribution composed of five weighted Gaussians, 200 samples are drawn, and the scale parameter is assigned as discussed in Section 5.4.1. Mean Integrated Squared Error (MISE) between the original and the approximated densities is calculated for the error estimation.

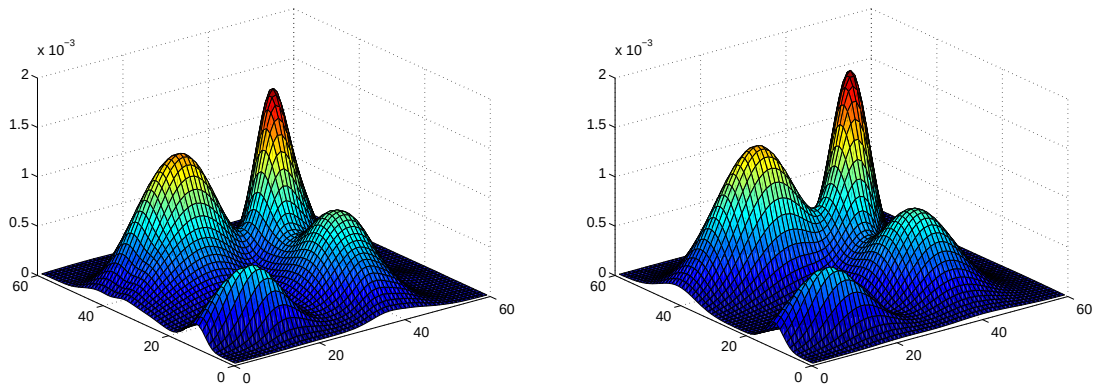


(a) (b)

Figure 5.1: Comparisons between kernel density estimation and density approximation (1D). For the approximation, 200 samples are drawn from the original distribution – $N(0.2, 10, 2^2)$, $N(0.35, 17, 4^2)$, $N(0.15, 27, 8^2)$, $N(0.2, 50, 16^2)$, and $N(0.1, 71, 32^2)$. (a) kernel density estimation (b) density approximation (MISE = 5.3234×10^{-5})

The result in Figure 5.1 shows that the mode finding based on mean-shift and the covariance estimation using the Hessian are very accurate.

Figure 5.2 shows the performance of the density approximation which is accurate enough to replace kernel density estimation in the multi-dimensional case.



(a) (b)

Figure 5.2: Comparison between kernel density estimation and density approximations (2D). (a) kernel density estimation (b) density approximation (400 samples, MISE = 1.5237×10^{-8})

5.4 Density Interpolation

The density approximation presented in Section 5.3 is an algorithm to find a compact representation when the mean, the covariance and the weight for each kernel are given. In the measurement step of Bayesian filtering, the likelihood values are known for a set of samples. In this case, the likelihood surface can be interpolated from sample likelihood. In this section, we describe the density interpolate algorithm.

5.4.1 Initial Scale Selection

One of the limitations of kernel-based algorithms is that they involve the specification of a scale parameter. Various research has been performed for the scale selection problem [1, 57, 70], but it is very difficult to find the optimal scale in general. Below, we explain a strategy to determine the scale parameter for the density estimation based on *nearest neighbors*.

The basic idea of this method is very simple, and similar approaches are discussed in [8, 9]. Each sample is intended to cover the local region around itself in the d-dimensional state space with its scale. For this purpose, k-nearest neighbors (KNN) is used, and the kernel bandwidth (scale) is determined by the distance to the k-th nearest neighbor of a sample. Define $\mathrm{KNN}_j^i(k)$ $(1 \leq j \leq d)$ to be the distance to k-th nearest neighbor from sample i in the j-th dimension, and then the covariance matrix $\mathbf{P}_i$ for i-th sample is given by

$$\mathbf{P}_i = c\,\mathrm{diag}(\mathrm{KNN}_1^i(k)\;\mathrm{KNN}_2^i(k)\ldots\mathrm{KNN}_d^i(k))^2\,\mathbf{I} \tag{5.25}$$

where c is a constant dependent upon the number of samples and the dimensionality, and $\mathbf{I}$ is a d-dimensional identity matrix.

By this method, samples in dense areas have small scales and the density will be represented accurately, but sparse areas convey only relatively rough information about the density function.

5.4.2 Interpolation

A Gaussian kernel is assigned to each sample for which mean and covariance corresponds to the sample location and the scale is initialized by the method in Section 5.4.1, respectively. When the likelihood value on each sample is given, the weight for each kernel can be computed by the Non-Negative Least Square (NNLS) method [41].

Denote $\mathbf{x}_i$ as the mean location and $\mathbf{P}_i$ as the covariance matrix for the i-th sample $(i = 1, \ldots, n)$. Also, suppose that l_i is the likelihood value of the i-th sample.

The likelihood at $\mathbf{x}_j$ induced by the i-th kernel is given by

$$p_i(\mathbf{x}_j) = \frac{1}{(2\pi)^{d/2} \mid \mathbf{P}_i \mid^{1/2}} \exp\left(-\frac{1}{2} D^2(\mathbf{x}_j, \mathbf{x}_i, \mathbf{P}_i)\right). \tag{5.26}$$

Define an $n \times n$ matrix $\mathbf{A}$ having an entry $p_i(\mathbf{x}_j)$ in (i, j), and an $n \times 1$ vector $\mathbf{b}$ having l_i in its i-th row. Then, the weight vector $\mathbf{w}$ can be computed by solving the following constrained least square problem,

$$\min_{\mathbf{w}} ||\mathbf{A}\mathbf{w} - \mathbf{b}||^2 \tag{5.27}$$

$$\text{subject to } \mathbf{w}_i \geq 0 \text{ for } i = 1, \ldots, n,$$

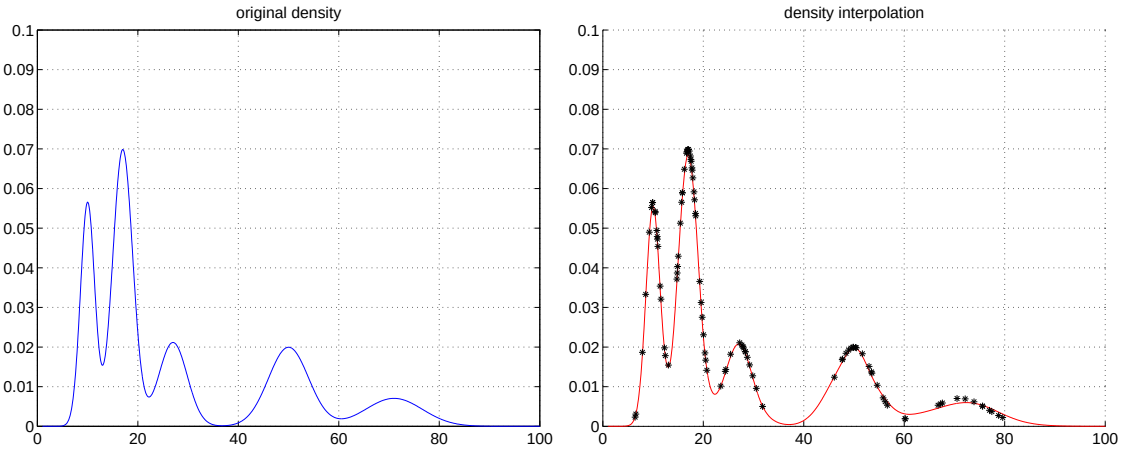
and it is denoted by $\mathbf{w} = nnls(\mathbf{A}, \mathbf{b})$. The size of matrix $\mathbf{A}$ is determined by the number of samples. When the sample size is large, sparse matrix operation methods can be used to solve $\mathbf{w}$ efficiently.

Usually, many of the weights will be zero and the final density function will be a mixture of Gaussians with a small number of components. The density interpolation simulates the heavy-tailed density function more accurately than the density approximation introduced in Section 5.3, while the density approximation generally produces a more compact representation.

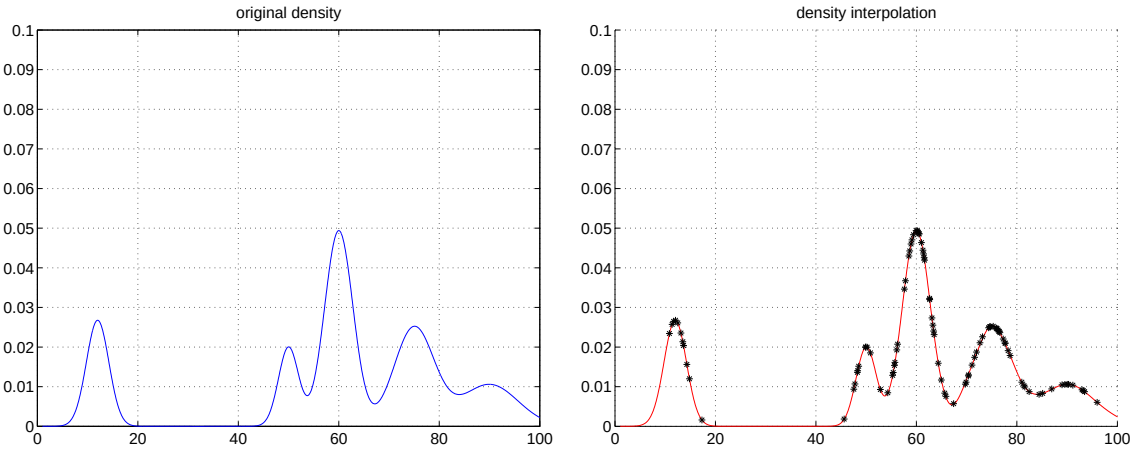
5.4.3 Performance of Interpolation

Figure 5.3 shows one-dimensional density interpolation results. For each case, 100 samples are drawn and the initial scale for each sample is given as explained in Section 5.4.1. The estimated density function approximates the original density very accurately as seen in Figure 5.3. Two different Gaussian mixtures –

$N(0.2, 10, 2^2)$ $N(0.35, 17, 4^2)$ $N(0.15, 27, 8^2)$ $N(0.2, 50, 16^2)$ $N(0.1, 71, 32^2)$ in example 1, and $N(0.15, 12, 5^2)$ $N(0.1, 15, 4^2)$ $N(0.35, 60, 8^2)$ $N(0.25, 75, 16^2)$ $N(0.15, 90, 32^2)$ in example 2 – are tested for the interpolation.



(a) example 1



(b) example 2

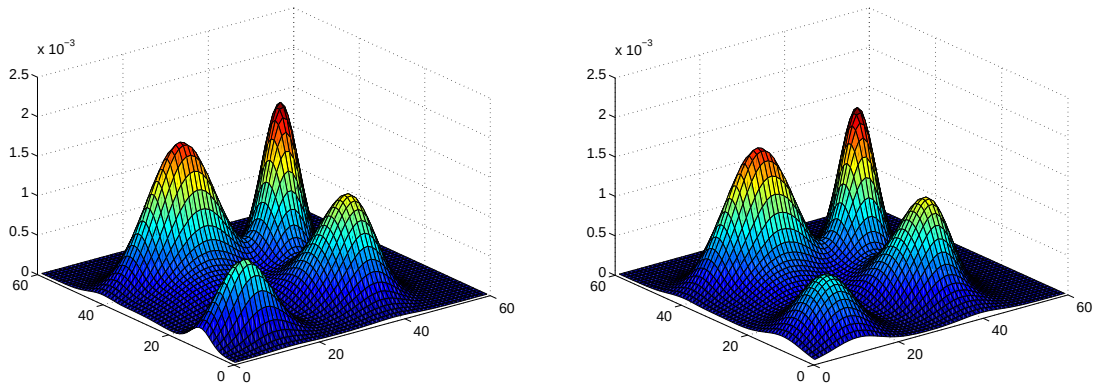
Figure 5.3: Two examples of original density functions and their interpolations. In the interpolation graphs (right), black stars represent the sample locations (100 samples). In case (a) and (b), 22 and 24 components have non-zero weights, respectively.

When 50 independent realizations are performed, MISE and its variance are very small for both examples as shown in Table 5.1.

Table 5.1: Error of density interpolation

	MISE	VAR
example 1	3.9479×10^{-5}	2.5613×10^{-9}
example 2	2.6871×10^{-5}	9.5103×10^{-10}

Also, a multi-dimensional density function is interpolated in the same manner, and its performance is discussed next. In Figure 5.4, the density interpolation produces a very accurate and stable result when 200 samples are drawn from the original density function (MISE $= 4.5467 \times 10^{-9}$, VAR $= 7.3182 \times 10^{-18}$ on average over 50 runs).



(a) (b)

Figure 5.4: Comparison between original density function and density interpolation (2D). (a) original density function (b) density interpolation with 30 non-zero weight components

These results show that surface interpolation is a sufficiently accurate method to approximate density function given samples and their corresponding likelihoods.

5.5 Simulation

In this section, synthetic tracking examples are simulated, and the performance of the kernel-based Bayesian filtering is compared with the SIR algorithm [31]. Two different process models – one linear and the other non-linear – are selected, and simulations are performed for various dimensions such as 2D, 3D, 5D, 10D, 12D and 15D. The accumulated Mean Squared Error (MSE) through 50 time steps is calculated in each run, and 50 identical experiments are made based on the same data for accurate error estimation.

The first process model is given by the following equation,

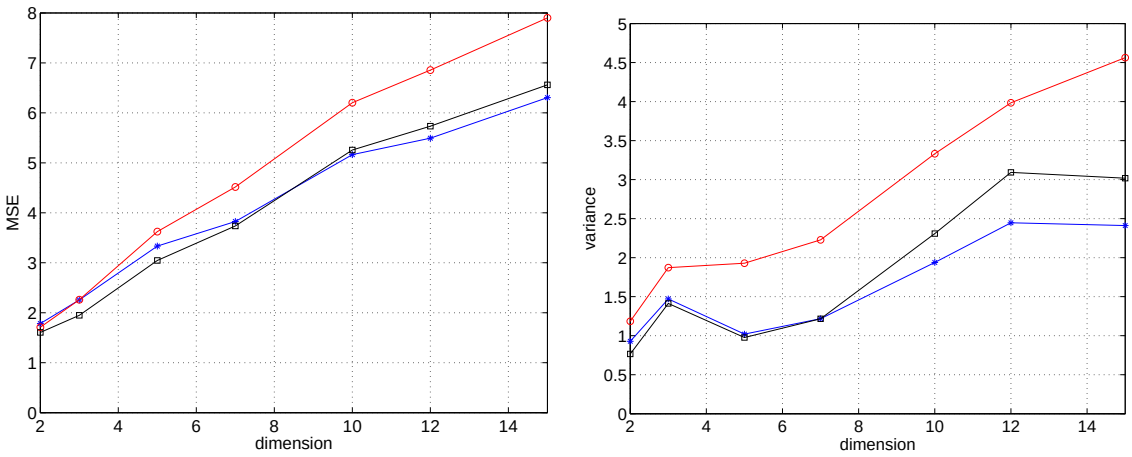
$$\mathbf{x}_t = \frac{\mathbf{x}_{t-1}}{2} + \frac{25\mathbf{x}_{t-1}}{1 + \mathbf{x}_{t-1}^T \mathbf{x}_{t-1}} + 8\cos(1.2(t-1))\mathbf{1} + \mathbf{u}_t \tag{5.28}$$

where $\mathbf{1}$ is the vector whose elements are all ones. The process noise $\mathbf{u}_t$ is drawn from a Gaussian distribution $N(1, \mathbf{0}, (\sqrt{2}\mathbf{I})^2)$ where $\mathbf{I}$ is the identity matrix. The measurement model is given by a non-linear function

$$\mathbf{z}_t = \frac{1}{2}\mathbf{x}_t^T \mathbf{x}_t + \mathbf{v}_t \tag{5.29}$$

where $\mathbf{v}_t$ is drawn from a Gaussian distribution $N(1, \mathbf{0}, \mathbf{I}^2)$. For the estimation of the measurement function, fifty particles (10 particles $\times$ 5 stages) are drawn, and the posterior is estimated and propagated through the time step t ($1 \leq t \leq 50$).

Figure 5.5 demonstrates simulation results by comparing MSE's and variances of both algorithms. According to our experiment with the first model, the SIR filter shows better or equivalent performance in low dimensions such as 2D and 3D, but our method starts to outperform in high dimensions – more than 5D.



(a) Error (b) Variance

Figure 5.5: MSE. Kernel-based Bayesian filtering with 50 particles (blue star), SIR with 50 particles (red circle), and SIR filter with 500 particles (black square) for model 1.

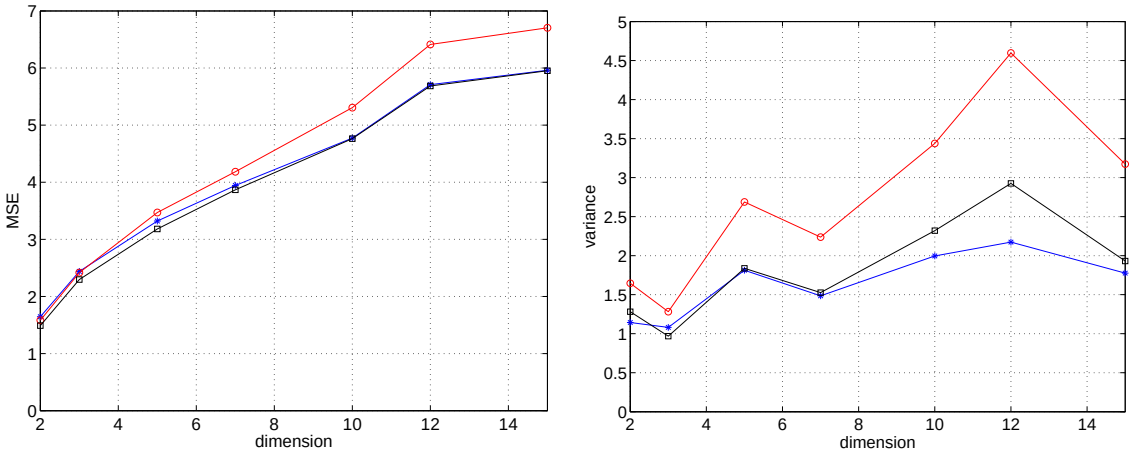
The second process model is a simple linear model given by

$$\mathbf{x}_t = \frac{\mathbf{x}_{t-1}}{2} + 2\cos(2(t-1))\mathbf{1} + \mathbf{u}_t \tag{5.30}$$

where $\mathbf{u}_t \sim N(1, \mathbf{0}, (\sqrt{2}\mathbf{I})^2)$. The same observation model as equation (5.29) is employed, and fifty samples are drawn for every simulation.

Kernel-based Bayesian filtering yields smaller error in the high dimension as in the previous case, and the detailed results are presented in Figure 5.6.

Two different process models produce almost similar results, and kernel-based Bayesian filtering shows better performance in high dimensional cases as expected. In order to demonstrate the benefit of kernel-based particles, we run the SIR algorithm with 500 samples, and compare the performance with our kernel-based Bayesian filtering with 50 samples. Surprisingly, the MSE's of the two cases are almost the same, and our algorithm has smaller variance of MSE than the SIR



(a) Error (b) Variance

Figure 5.6: MSE and variance of for MSE kernel-based Bayesian filtering with 50 samples (blue star), SIR filter with 50 particles (red circle), and SIR filter with 500 particles (black square) for model 2.

algorithm.

This result suggests that kernel-based Bayesian filtering can be applied effectively to high dimensional applications, especially when many samples are not available and the observation process is very time-consuming.

5.6 Applications to Tracking

Particle filtering provides a convenient method for estimating and propagating the density of state variables regardless of the underlying distribution and the given system in the Bayesian framework. Additionally, our kernel-based Bayesian filtering has an advantage of managing multi-modal density functions with a relatively small number of samples. In this section, we demonstrate the performance of the kernel-based Bayesian filtering by tracking objects in real videos.

5.6.1 1D Simulation

For this experiment, the process model is given by the following equation,

$$\mathbf{x}_t = 1 + \sin(w\pi(t-1)) + \phi_1 \mathbf{x}_{t-1} + \mathbf{u}_t \tag{5.31}$$

where $w = 4e - 2$, $\phi_1 = 0.5$, and $\mathbf{u}_t \sim N(1, 0, 2)$ is the random variable for the process noise. The measurement model is given by a non-linear function

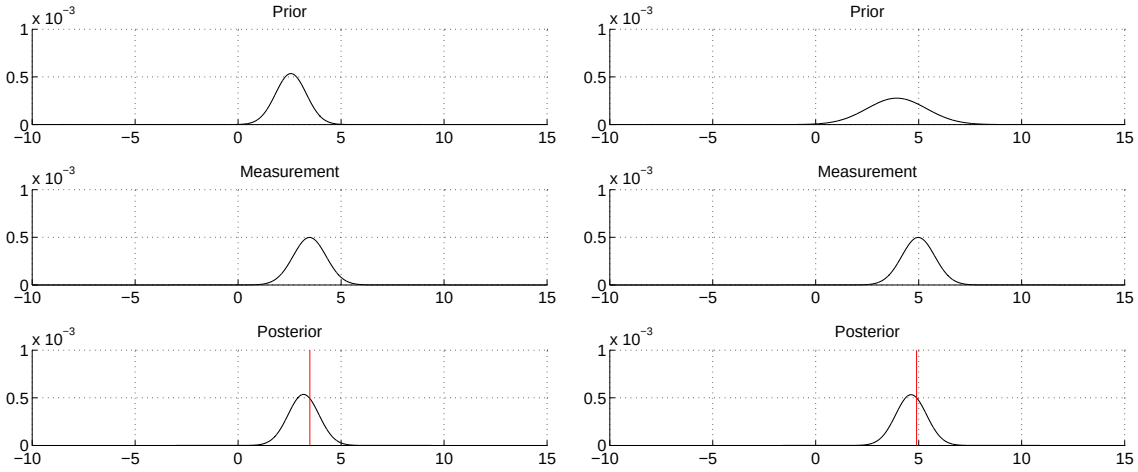
$$\mathbf{z}_t = \phi_2(\mathbf{x}_t^2 + \mathbf{x}_t) + \mathbf{v}_t \tag{5.32}$$

where $\phi_2 = 0.5$ and the observation noise $\mathbf{v}_t$ is drawn from a Gaussian distribution $N(1, 0, 0.1)$. One hundred particles are drawn by the quasi-random sampling method, and the density is estimated and propagated for each time step t $(1 \leq t \leq 150)$.

As seen in Figure 5.7, the multi-modal densities are effectively represented with the mixture of Gaussians, and the state density is propagated through the measurement and update stages. The same experiment was repeated 100 times, and the Mean Squared Error (MSE) between the true and the estimated target location was computed. The MSE and the variance of our algorithm are 0.284 and 0.136 respectively, which are better than the classical particle filter (MSE = 0.340, variance = 0.294).

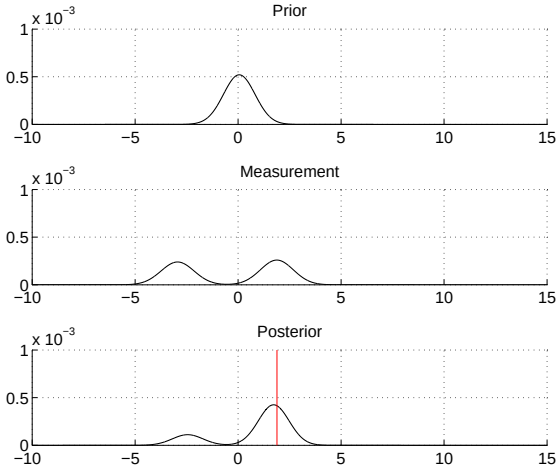
5.6.2 Visual Tracking

The overall tracking procedure is equivalent to what is described in Section 5.5, and we explain the process and the measurement models briefly.

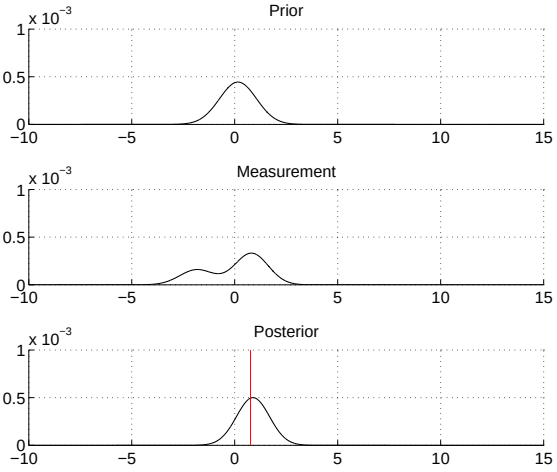


(a) t = 60

(b) t = 90



(c) t = 120



(d) = 150

Figure 5.7: The sequence of 1D tracking simulation. The top of each figure shows the prior probability, the second is the measurement function, and the last one is the posterior probability. In the posterior pdf, the (red) vertical bar denotes the true location of target.

A random walk is assumed for the process model since it is very difficult to describe the motion before the observation, even though our algorithm can accommodate the general non-linear function by unscented transformation described in Section 5.2.2. This assumption is natural for the motion of objects in video, and simple to manage because it is linear. So, the process model equation (5.1) can be rewritten as follows.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_t \tag{5.33}$$

where $\mathbf{v}_t$ is a zero mean Gaussian random variable.

5-stage sampling is incorporated as introduced in Section 5.2.3, and the likelihood of each particle is computed by comparing the target and the candidate histograms as suggested in [58]. Supposed that the histogram of the target is denoted by $c^\star(i)$ $(i = 1 \dots N)$, where N is the number of bins in the histogram and $\sum_{i=1}^{N} c^\star(i) = 1$. The Bhattacharyya distance in equation (5.34) is used to measure the similarity between two histograms

$$D[c^\star, c(\mathbf{x}_t)] = \left(1 - \sum_{i=1}^{N} \sqrt{c^\star(i)c(\mathbf{x}_t; i)}\right)^{1/2} \tag{5.34}$$

and the measurement function at time t is given by

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto \exp\left(-\lambda D^2[c^\star, c(\mathbf{x}_t)]\right) \tag{5.35}$$

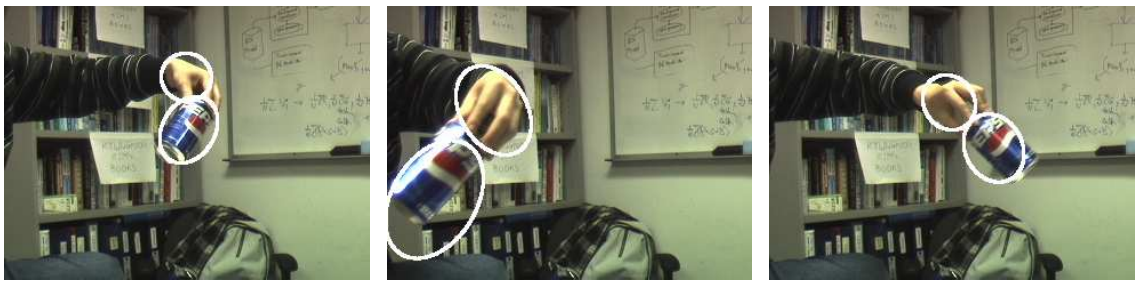
Based on the likelihood of each particle and the initial covariance matrix derived by the method in Section 5.4.1, the measurement density is constructed by density interpolation.

Two sequences are tested in our experiment. In the first sequence, two objects – a hand carrying a can – are tracked with 200 samples (40 samples $\times$ 5 stages).

The state space is described by a 10 dimensional vector, which is the concatenation of two 5 dimensional vectors representing two independent ellipses as follows.

$$(x_1, y_1, lx_1, ly_1, r_1, x_2, y_2, lx_1, ly_2, r_2) \tag{5.36}$$

where x_i and y_i $(i = 1, 2)$ are the location of ellipses, lx_i is the length of x-axis, ly_i is the length of y-axis, and r_i is the rotation variable. The tracking result is shown in Figure 5.8, and our algorithm successfully tracks two objects for the whole sequence except the period that the side of the can is completely occluded around the 470th frame.



(a) $t = 43$ (b) $t = 304$ (c) $t = 387$

(d) $t = 456$ (e) $t = 570$ (c) $t = 898$

Figure 5.8: Object tracking result of *can* sequence.

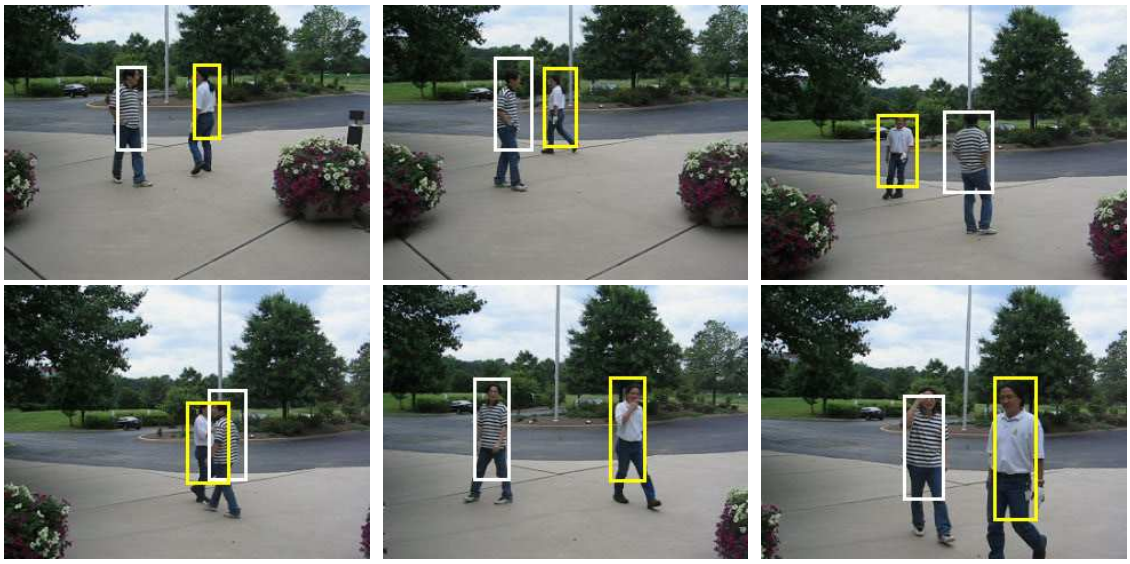
The bodies of two persons are tracked in the second sequence, in which one occludes the other completely several times. The state vector is constructed by the same method as in the *can* sequence, but two rectangles are used instead of ellipses.

A 8 dimensional vector – (x, y, w, h) for each rectangle – is used to describe the state, and 50 samples (25 samples $\times$ 2 stages) are used. Figure 5.9 (a) demonstrates the tracking results, and our algorithm shows good performance in spite of severe occlusions.

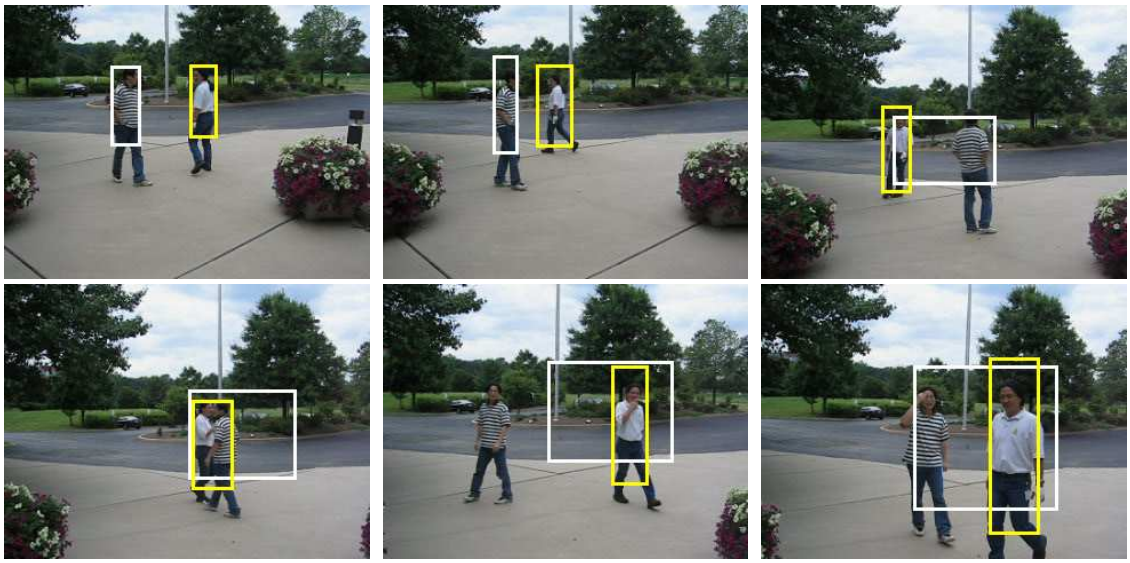
The tracker based on the SIR filter was also implemented, and compared with our algorithm. As seen in Figure 5.9 (b), the SIR algorithm shows unstable performance for the same sequence. According to experiments, one would need to run the SIR algorithm using about 200 particles to obtain a comparable result with our algorithm using 50 samples.

5.7 Discussion and Conclusion

In this chapter, we proposed a new Bayesian filtering framework where analytic representations are used to approximate relevant density functions. Density approximation and interpolation technique are introduced for density propagation. Various simulations and tests on object tracking in real videos show the effectiveness of our density approximation methods and the kernel-based Bayesian filtering. By maintaining analytic representations of the density functions, we can sample in the state space more effectively and more efficiently. This advantage is significant for high dimensional problems. In addition, the approximation error can be monitored and analyzed. Our future work is focused on analyzing the approximation error in the posterior distribution and its propagation over time.



(a) Result by our method



(b) Result by the SIR filter

Figure 5.9: Object tracking result of *person* sequence at $t = 1, 94, 140, 192, 236, 300$.

Appendix A

Robust Observations for Object Tracking

It is a difficult task to find an observation model that will perform well for long-term visual tracking. In this chapter, we propose an adaptive observation enhancement technique based on likelihood images which are derived from multiple visual features. The most discriminative likelihood image is extracted by Principal Component Analysis (PCA) and incrementally updated frame by frame to reduce temporal tracking error. In the particle filter framework, the feasibility of each sample is computed using this most discriminative likelihood image before the observation process. Integral image is employed for efficient computation of the feasibility of each sample. We illustrate how our enhancement technique contributes to more robust observations through demonstrations.

A.1 Introduction

Trackers are based on the some measurement of similarity between the target to be tracked and observations, and various observation methods are used to define this similarity. Intensity or color is natural to use in object tracking, and approaches based on templates [44] and histograms [13, 45, 53, 58] are very common. Also, edges [31] and filter responses [34, 51] are important features for object tracking. Various

observation strategies have been proposed, but there is no generally superior observation method for visual tracking algorithms. So, we instead propose an observation enhancement technique based on likelihood images, which can be incorporated into many tracking algorithms.

A likelihood image represents the contrast information between foreground (target region) and background (its surrounding); it is created by comparing histograms of both areas with respect to some features. It was originally suggested in [10] for tracking problems; there, the most discriminative feature selected from a set of likelihood images is directly used for mean-shift tracking. However, this approach may exhibit poor performance in clutter and can lose the target in spite of its visual salience. Also, the likelihood image can be significantly contaminated by temporary tracking errors. There have been some closely related works [7, 51], but they do not provide an adequate solution to these problems.

In this chapter, we present an observation enhancement technique using likelihood images obtained from two different feature spaces – RGB and normalized RGB (rgb). Six likelihood images are created, and a most discriminative likelihood image is extracted by PCA. In order to avoid pollution of the extracted likelihood image by tracking error, we update the subspace incrementally on the assumption that the scene around the target changes smoothly. The "brightness" in the most discriminative likelihood image delivers prior information about the target region, and it is employed to compute the *feasibility* of a sample (to be defined rigorously below) before the measurement step in the particle filter. The feasibility is obtained by the summation of values inside the region in the most discriminative likelihood

image, and an integral image [80] is employed for efficient computation. The final weight of each particle is determined by the product of the feasibility and the likelihood in observation. As a result, several independent features are merged through the likelihood images and the merged features are utilized for robust observation via the feasibility computation.

This chapter is organized as follows. We describe the likelihood images and the feature extraction in Section A.2 and A.3, respectively. In Section A.4, tracking in a particle filter framework and experimental results are demonstrated.

A.2 Likelihood Images

Likelihood images represent the distinctiveness of a target object from background with respect to a given feature or set of features. For the construction of likelihood images, log-likelihood ratios are obtained first from histograms of foreground and background pixels. Then, the salient region in foreground can be detected by identifying high likelihood ratios.

In detail, suppose the foreground is given and the background is regarded as a rectangular region surrounding the foreground. For a given feature, let $\phi_{fg}(i)$ and $\phi_{bg}(i)$ be the frequency of pixels with value i in the foreground and the background, respectively. The log-likelihood ratio for a feature value i is given by

$$L(i) = \max\left(-1, \min\left(1, \log\frac{\max\left(\phi_{fg}(i), \delta\right)}{\max\left(\phi_{bg}(i), \delta\right)}\right)\right) \tag{A.1}$$

where δ is a very small number. The likelihood image for each feature is created by backprojecting the ratio into each pixel in the image.

We construct a likelihood image for each color channel in the RGB and *rgb* color spaces, so that 6 different likelihood images are generated for feature extraction. Figure A.1 shows example likelihood images derived from each color channel.

A.3 Feature Extraction by PCA

A.3.1 Batch Method

Our objective is to identify the most discriminative likelihood image and measure the feasibility of each particle to improve the observation quality.

There are various feature extraction methods; here, PCA is employed to generate the most discriminative likelihood image. The linear discriminant method may not be appropriate since the histograms of the foreground and the background regions are often multi-modal in the original color image. However, the transformed likelihood image is likely to have positive value pixels in the foreground region while negative value pixels tend to be frequently observed in the background region. Even though the foreground and the background cannot be perfectly separated by a linear hyper-plane, we can expect that most pixels would be classified correctly by it. Also, linear methods are much faster than their non-linear counterparts such as Kernel LDA [47] and Kernel PCA [67].

Suppose that S_{fg} and S_{bg} are the set of n-dimensional vectors sampled from the foreground and background area of n likelihood images, and that $\mathbf{m}$ and $\mathbf{V}$ are the $n \times 1$ mean vector and $n \times n$ covariance matrix of these data, respectively. Let $\mathbf{e}_i$ $(i = 1, \ldots, n)$ be the eigenvectors associated with the eigenvalues λ_i which are sorted in non-increasing order. Once $\mathbf{e}_i$ are obtained, the value y projected from the

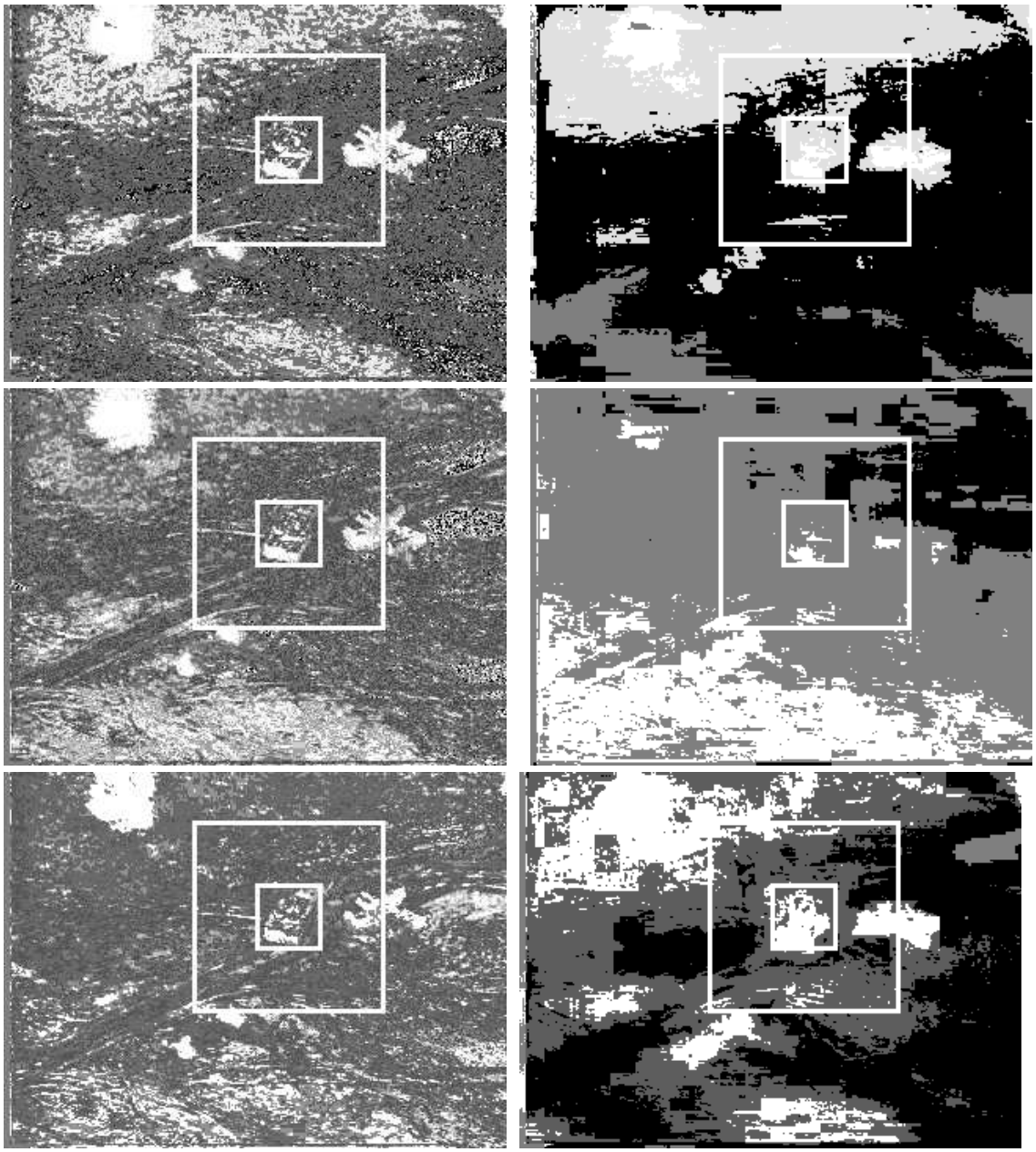
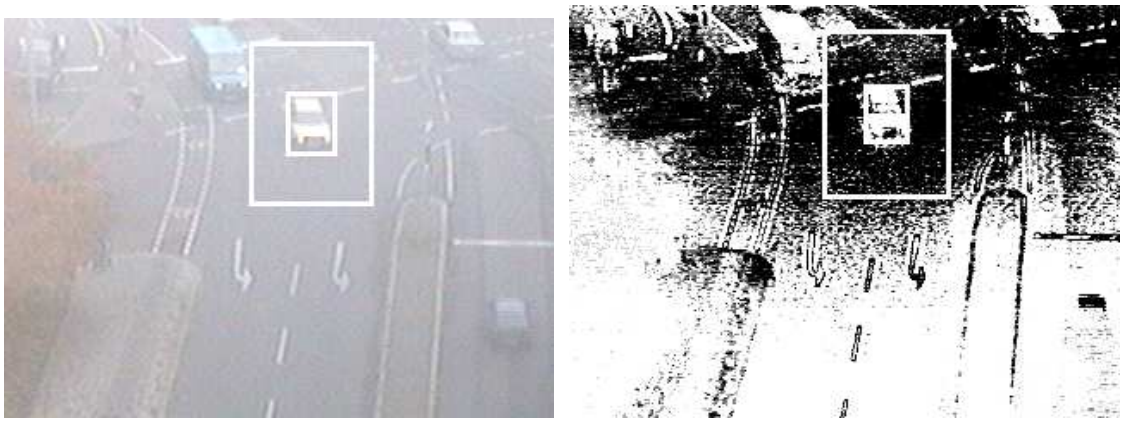


Figure A.1: Likelihood images in RGB (left) and *rgb* (right) color channel. For this image, the target in *rgb* space is more distinctive.

original vector $\mathbf{x}$ to the most discriminative feature space is given by $y = \mathbf{e}_1^T(\mathbf{x}-\mathbf{m})$. The following figure shows an example of the most discriminative likelihood image extracted by PCA.



(a) original image (b) extracted image

Figure A.2: Comparison between original image and the most discriminative likelihood image

The most discriminative likelihood image for the first frame is created by batch processing, but the subsequent ones are constructed by the following incremental method to include previous information.

A.3.2 Incremental Subspace Update

As described above, the distinctiveness information of the target in likelihood images can be significantly reduced by tracking errors. We alleviate this problem by updating the subspace incrementally rather than using a completely new subspace in each frame.

Since the data added in each frame has as many samples from the foreground or from the background region and the number of dimension is moderate in our

application, the standard incremental PCA [27, 82] is not required and a simpler (but accurate) method can be used. Instead of computing eigenvectors without consideration of the full covariance matrix and the matrix decomposition, we just compute the updated mean and covariance with new observations in the current frame and perform SVD to obtain eigenvectors.

Denote by $(\mathbf{m}_{old}, \mathbf{V}_{old})$ and $(\mathbf{m}, \mathbf{V})$ pairs of mean and covariance in the previous and current time step, respectively. Then, the updated mean and covariance $(\mathbf{m}_{new}, \mathbf{V}_{new})$ including the new observations are as follows.

$$\mathbf{m}_{new} = (1-\alpha)\mathbf{m}_{old} + \alpha\mathbf{m} \tag{A.2}$$

$$\mathbf{V}_{new} = (1-\alpha)\mathbf{V}_{old} + \alpha\mathbf{V} + \alpha(1-\alpha)(\mathbf{m}_{old} - \mathbf{m})(\mathbf{m}_{old} - \mathbf{m})^T \tag{A.3}$$

where α is the learning rate whose value is between 0 and 1. The derivations of the above equations are shown in equation (7) and (8) in [27].

In each time step, an incremental subspace update is performed to obtain the most discriminative likelihood image. This method is more efficient than the batch method since we do not need to store the data from the previous frames; instead, we only need the mean and covariance matrix.

A.4 Tracking by Particle Filtering

In this section we will show how to incorporate feature extraction technique for robust observations into the particle filter framework.

The particle filter [31] is a stochastic framework to propagate the conditional

density to the next step. The state variable $\mathbf{x}_t$ $(t = 0 \dots n)$ is characterized by its probability density function estimated from the sequence of measurements $\mathbf{z}_t$ $(t = 0 \dots n)$. The density function is represented with a set of samples and their weights which enable us to describe an arbitrary probability density function effectively.

In our experiments, the state variable is a 3-dimensional vector (x, y, s) where (x, y) is 2D location of an object and s is a scale parameter, and the target is represented with a rectangular region. A random walk is assumed for the process model since it is not desirable to assign any specific motion model before observation. Since we employ SIR filter, the weights of particles are equal until the prediction step; then they are updated twice by the feasibility and the likelihood in observation.

A.4.1 Feasibility for Particle

Feasibility is meant to capture how the region represented by a sample is salient with respect to the background, and it is computed by the summation of values inside the region in the most discriminative likelihood image.

Formally, suppose that the value at (x, y) in the most discriminative likelihood image is $MD(x, y)$. Since we use rectangular regions for the observation, the feasibility w_f is

$$w_f(x_i, y_i, s_i) = \sum_{x_i \le x \le x_i + w_i, y_i \le y \le y_i + h_i} MD(x, y) \tag{A.4}$$

where $x_i \le x \le x_i + w_i, y_i \le y \le y_i + h_i$ is the area associated with the i-th particle. The integral image (II) proposed in [80] is defined to be

$$II(x, y) = \sum_{x' \le x, y' \le y} MD(x', y'), \tag{A.5}$$

so that the feasibility can be computed by only 4 table look-up operations using the integral image. After computing the feasibility, the sample weight is updated with this value.

This strategy is reasonable since the regions containing many high likelihood-ratio pixels are selected target candidates based on multiple visual cues and the observation process described below can compensate for the disadvantage of likelihood images – poor performance in clutter.

A.4.2 Observation

Color-based tracking is employed in our experiments. The likelihood of each step is based on the similarity of the RGB histogram between the target and the candidates. The histogram of the target is denoted by $c^\star(i)$ $(i = 1 \dots N)$, where N is the number of bins in the histogram and $\sum_{i=1}^{N} c^\star(i) = 1$. The Bhattacharyya distance in equation (A.6) is used to measure the similarity between two histograms

$$D[c^\star, c(\mathbf{x}_t)] = \left(1 - \sum_{i=1}^{N} \sqrt{c^\star(i)c(\mathbf{x}_t; i)}\right)^{1/2} \tag{A.6}$$

and the final measurement function including feasibility at time t is given by

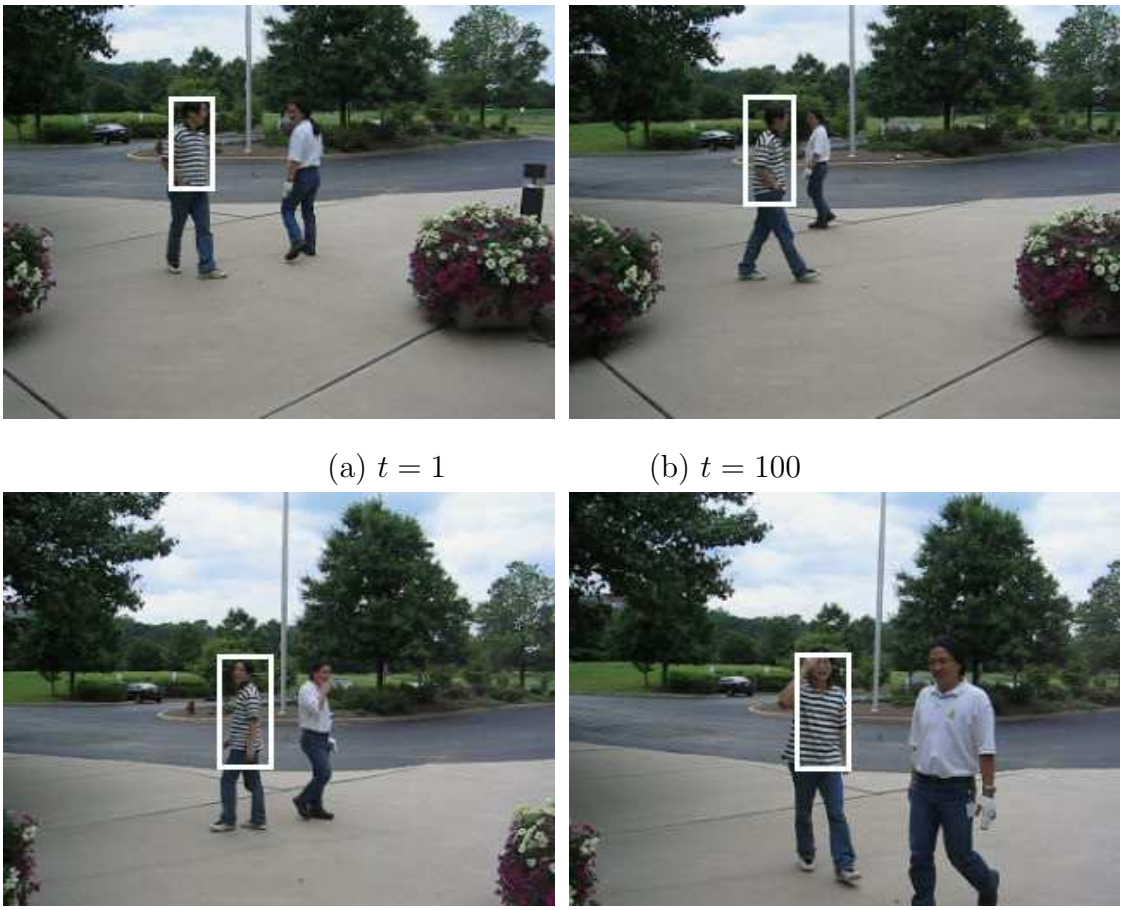
$$p(\mathbf{z}_t|\mathbf{x}_t) \propto w_f(\mathbf{x}_t)\exp\left(-\lambda D^2[c^\star, c(\mathbf{x}_t)]\right) \tag{A.7}$$

where λ is a constant.

A.4.3 Results

Two different video sequences were used to test the performance of our tracker – *people* and *vehicle* sequence. In the *people* sequence, the target is not so distinctive

in likelihood images due to clutter, so tracking only with likelihood images is not successful. However, the combination of feasibility and likelihood in observation can track a person for the whole sequence with 100 particles. The tracking results are shown in Figure A.3.



(a) $t = 1$ (b) $t = 100$

(c) $t = 210$ (d) $t = 300$

Figure A.3: Tracking results with *people* sequence

A car is moving in the severe fog in the second video, which is downloaded from Universität Kahlsruhe homepage (`http://i21www.ira.uka.de/image_sequences`). Even with white pixels due to the fog in the background, the white car in the foreground is identified clearly in the most discriminative feature space as seen in

Figure A.2, and tracking was also successful with 100 particles.

(a) $t = 1$ (b) $t = 25$

(c) $t = 67$ (d) $t = 95$

Figure A.4: Tracking results with *vehicle* sequence

A.5 Discussion

We described a method to improve the robustness of observations for object tracking using the most discriminative likelihood image. This likelihood image is obtained from the combination of multiple independent features and updated incrementally. This technique is incorporated into a particle filter, and tracking is performed based on the original image as well as the combined likelihood image.

In particle filter tracking , the quality of sampling is critical to its overall

performance. So, we can achieve better results with a small number of samples if the particles with low feasibility are rejected and new samples are used. Currently, only color information is used, and other visual features should be tested in our framework.

BIBLIOGRAPHY

[1] I. Abramson, "On bandwidth variation in kernel estimates - a square root law," *The Annals of Statistics*, vol. 10, no. 4, pp. 1217–1223, 1982.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–189, 2002.

[3] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Santa Barbara, CA, 1998, pp. 232–237.

[4] M. J. Black and A. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," in *Proc. European Conf. on Computer Vision,* Cambridge, UK, 1996, pp. 610–619.

[5] T. Cham and J. Rehg, "A multiple hypothesis approach to figure tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO, volume II, 1999, pp. 239–219.

[6] H. Chen and T. Liu, "Trust-region methods for real-time tracking," in *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, volume II, 2001, pp. 717–722.

[7] H. Chen, T. Liu, and C. Fuh, "Probabilistic tracking with adaptive feature selection," in Proc. of the 17th Intl. Conf. on Pattern Recognition Cambridge, UK, volume 2, 2004, pp. 736–739.

[8] W. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *J. Amer. Statist. Assn.*, vol. 74, pp. 829–836, 1979.

[9] W. Cleveland and C. Loader, "Smoothing by local regression: Principles and methods," *Statistical Theory and Computational Aspects of Smoothing*, pp. 10–49, 1996.

[10] R. Collins and Y. Liu, "On-line selection of discriminative tracking features," in *Proceedings of the 2003 International Conference of Computer Vision (ICCV '03)*, October 2003.

[11] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proc. 7th Intl. Conf. on Computer Vision,* Kerkyra, Greece, September 1999, pp. 1197–1203.

[12] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC, volume II, June 2000, pp. 142–149.

[14] D. Comaniciu, V. Ramesh, and P. Meer, "The variable bandwidth mean shift and data-driven scale selection," in *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, volume I, July 2001, pp. 438–445.

[15] D. Comaniciu, V. Ramesh, and P. Meer, "Kernl-based object tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 5, pp. 564–575, 2003.

[16] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Society B*, vol. 39, pp. 1–38, 1977.

[17] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC, 2000.

[18] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[19] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[20] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, second edition, 2001.

[21] A. Elgammal, R. Duraiswami, and L. Davis, "Probability tracking in joint feature-spatial spaces," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, Wisconsin, June 2003.

[22] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of IEEE*, vol. 90, pp. 1151–1163, 2002.

[23] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. European Conf. on Computer Vision,* Dublin, Ireland, volume II, June 2000, pp. 751–767.

[24] P. Fieguth and D. Teropoulos, "Color-based tracking of heads and other mobile objects at video frame rates," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Juan, Puerto Rico, 1997, pp. 21–27.

[25] B. Frey, "Filling in scenes by propagating probabilities through layers into appearance models," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC, volume I, 2000, pp. 185–192.

[26] N. Friedman and S. Russell, "Image segmenation in video sequences: A probabilistic approach," in *Proc. Thirteenth Conf. Uncertainty in Artificial Intell. (UAI)*, 1997.

[27] P. Hall, D. Marshall, and R. Martin, "Merging and splitting eigenspace models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 9, pp. 1042–1048, 2000.

[28] B. Han, D. Comaniciu, and L. Davis, "Sequential kernel density approximation through mode propagation: Applications to background modeling," in *Asian Conference on Computer Vision* Jeju Island, Korea, 2004.

[29] B. Han, D. Comaniciu, Y. Zhu, and L. Davis, "Incremental density approximation and kernel-based bayesian filtering for object tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington DC, 2004.

[30] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 809–830, 2000.

[31] M. Isard and A. Blake, "Condensation - Conditional density propagation for visual tracking," *Intl. J. of Computer Vision*, vol. 29, no. 1, 1998.

[32] O. Javed and M. Shah, "Tracking and object classification for automated surveillance," in *Proc. European Conf. on Computer Vision,* Copenhagen, Denmark, 2002, pp. 343–357.

[33] A. Jepson and M. Black, "Mixture models for optical flow computation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, 1993, pp. 760–761.

[34] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hawaii, volume I, 2001, pp. 415–422.

[35] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceedings SPIE*, volume 3068, 1997, pp. 182–193.

[36] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the Am. Soc. of Mechanical Eng., D, Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[37] Z. Khan, T. Balch, and F. Dellaert, "A rao-blackwellized particle filter for eigentracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington DC, 2004.

[38] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.

[39] 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 *Int. Conf. Pattern Recognition*, 1994.

[40] J. Kotecha and P. Djuric, "Gaussian sum particle filtering," *IEEE Trans. Signal Process.*, vol. 51, no. 10, pp. 2602–2612, 2003.

[41] C. L. Lauwon and B. J. Hanson, *Solving Least Squares Problems*. Prentice-Hall, 1974.

[42] J. Li and A. Barron, "Mixture density estimation," in *Advances in Neural Information Processing Systems 12*, MIT Press, 2000.

[43] J. Lim, D. Ross, R. Lin, and M. Yang, "Incremental learning for visual tracking," in *Neural Information Processing Systems*, 2004, pp. 793–800.

[44] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 6, pp. 810–815, 2004.

[45] S. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image and Vision Computing Journal*, vol. 17, pp. 223–229, 1999.

[46] R. Merwe, A. Doucet, N. Freitas, and E. Wan, "The unscented particle filter," Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, 2000.

[47] S. Mika, G. Rätsch, J. Western, B. Schölkoph, and K. Müller, "Fisher discriminant analysis with kernels," *Neural Networks for Signal Processing IX*, pp. 41–48, 1999.

[48] A. Mittal and N. Paragios, "Motion-based backgroudn subtraction using adaptive kernel density estimation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington DC, 2004.

[49] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, "Background modeling and subtraction of dynamic scenes," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, 2003.

[50] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models, M.I. Jordan, ed.*, 1998, pp. 355–368.

[51] H. Nguyen and A. Smeulders, "Tracking aspects of the foreground against the background," in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, May 2004.

[52] H. T. Nguyen, M. Worring, and R. van den Boomgaard, "Occlusion robust adaptive template tracking," in *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, October 2001.

[53] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99–110, 2003.

[54] K. Okuma, A. Taleghani, N. Freitas, J. Little, and D. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, May 2004.

[55] N. M. Oliver, B. Rosario, and A. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 831–843, 2000.

[56] N. Paragios and V. Ramesh, "A MRF-based approach for real-time subway monitoring," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, Hawaii, volume I, 2001, pp. 1034–1040.

[57] B. Park and J. Marron, "Comparison of data-driven bandwidth selectors," *J. of Amer. Stat. Assoc.*, vol. 85, pp. 66–72, 1990.

[58] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. European Conf. on Computer Vision,* Copenhagen, Denmark, volume I, 2002, pp. 661–675.

[59] V. Philomin, R. Duraiswami, and L. S. Davis, "Quasi-random sampling for condensation," in *ECCV (2)*, 2000, pp. 134–149.

[60] R. Pless, J. Larson, S. Siebers, and B. Westover, "Evaluation of local models of dynamic backgrounds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, Wisconsin, volume II, 2003, pp. 73–78.

[61] C. Priebe and D. Marchette, "Adaptive mixture density estimation," *Pattern Recog.*, vol. 26, no. 5, pp. 771–785, 1993.

[62] R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Review*, vol. 26, pp. 195–239, 1984.

[63] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and foreground detection using Kalman filtering," in *Proc. Int. Conf. Recent Advances Mechatronics*, 1995, pp. 193–199.

[64] J. Rittscher, J. Kato, S. Joga, and A. Blake, "A probabilistic background model for tracking," in *Proc. European Conf. on Computer Vision,* Dublin, Ireland, 2000, pp. 336–350.

[65] D. Ross, J. Lim, and M. Yang, "Adaptive probabilistic visual tracking with incremental subspace update," in *Proc. European Conf. on Computer Vision,* Prague, Czech, volume II, May 2004, pp. 470–482.

[66] Y. Rui and Y. Chen, "Better proposal distributions: Object tracking using unscented particle filter," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, Hawaii, volume II, 2001, pp. 786–793.

[67] B. Schölokph, A. Smola, and K. Müller, "'nonlinear component analysis as a kernel eigenvalue problems," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[68] D. W. Scott, *Multivariate Density Estimation.* Wiley, 1992.

[69] M. Seki, T. Wada, H. Fujiwara, and K. Sumi, "Background subtraction based on cooccurence of image variations," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, Wisconsin, 2003.

[70] S. Sheather and M. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *J. Royal Statist. Soc. B*, vol. 53, pp. 683–690, 1991.

[71] Y. Sheikh and M. Shah, "Bayesian object detection in dynamic scenes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA, 2005.

[72] B. W. Silverman, *Density Estimation for Statistics and Data Analysis.* Chapman & Hall, 1986.

[73] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO, 1999, pp. 246–252.

[74] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[75] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Buhmann, "Topology free hidden Markov models: Application to background modeling," in *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, 2001, pp. I: 294–301.

[76] P. Torma and C. Szepesvari, "Enhancing particle filter using local likelihood sampling," in *Proc. European Conf. on Computer Vision,* Prague, Czech Republic, 2004, pp. 16–27.

[77] F. D. L. Torre and M. Black, "A framework for robust subspace learning," *Intl. J. of Computer Vision*, vol. 54, no. 1-3, pp. 117–142, 2003.

[78] K. Toyoma, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. 7th Intl. Conf. on Computer Vision,* Kerkyra, Greece, 1999, pp. 255–261.

[79] J. J. Verbeek, N. Vlassis, and B. Kröse, "Efficient greedy learning of gaussian mixture models," *Neural Comput.*, vol. 15, no. 2, pp. 469–485, 2003.

[80] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, Hawaii, 2001, pp. 511–518.

[81] M. P. Wand and M. Jones, *Kernel Smoothing.* Chapman & Hall, 1995.

[82] J. Weng, Y. Zhang, and W. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 8, pp. 1034–1040, 2003.

[83] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, 1997.

[84] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation," in *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, volume I, 2003, pp. 464–471.

[85] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, Wisconsin, 2003, pp. 44–50.

[86] S. Zhou, R. Chellappa, and B. Moghaddam, "Visual tracking and recognition using appearance adaptive models in particle filters," *IEEE Trans. Image Process.*, vol. 11, pp. 1434–1456, 2004.