

University of Nevada, Reno

**A Non-Parametric Framework for Object Tracking in Videos with  
Quasi-Stationary Backgrounds**

A dissertation submitted in partial fulfillment of the  
requirements for the degree of Doctor of Philosophy in  
Computer Science and Engineering

by

Alireza Tavakkoli

Dr. Mircea Nicolescu/Dissertation Advisor

May, 2009



University of Nevada, Reno  
Statewide • Worldwide

THE GRADUATE SCHOOL

We recommend that the dissertation  
prepared under our supervision by

**ALIREZA TAVAKKOLI**

entitled

**A Non-Parametric Framework For Object Tracking In Videos  
With Quasi-Stationary Backgrounds**

be accepted in partial fulfillment of the  
requirements for the degree of

**DOCTOR OF PHILOSOPHY**

Dr. Mircea Nicolescu, Advisor

Dr. George Bebis, Committee Member

Dr. Monica Nicolescu, Committee Member

Dr. Zong Tian, Committee Member

Dr. Michael Webster, Graduate School Representative

Marsha H. Read, Ph. D., Associate Dean, Graduate School

May, 2009

# Abstract

The ability to automatically detect and track objects of interest in a video sequence is an essential feature of many high-level vision-based applications. In this dissertation we propose a non-parametric computational framework which detects and tracks foreground image regions (typically corresponding to moving objects such as people or vehicles), while estimating their trajectories for further processing stages such as activity recognition.

The contribution of this dissertation extends along two main directions: background modeling for object detection and appearance-based object tracking. In many applications such as vision-based surveillance or traffic monitoring it is typically assumed that the camera is static. Even with this assumption, quasi-stationary backgrounds (that change due to moving tree branches, waving flags, rain or water surfaces) pose significant challenges in detecting and tracking the actual objects of interest while ignoring such changes. Due to the diverse nature of vision-based applications, it has been a main concern for researchers to design a scene-independent system that consistently handles these difficult situations. In the object detection stage, we first propose a novel adaptive statistical method as a base-line system that addresses the issue of scene-independent background modeling. After investigating its performance, we introduce a universal statistical technique which aims to overcome the weaknesses of its predecessor in modeling slow changes in the background. Furthermore, a new analytical technique is proposed that addresses the limitations of statistical techniques

which are bound to the probability density estimation accuracy. The performance of each proposed method is studied, while investigating scenarios where each technique leads to better performance.

In the object tracking stage, photometric and geometric appearance models are built for the detected objects, then used to predict their locations in subsequent images by employing a novel spatio-spectral tracking technique. The proposed tracker is shown to be particularly robust to local illumination changes, while also being able to detect and resolve the temporary overlapping of tracked objects.

We support our claims with extensive experimental results and comparisons between the proposed techniques and other methods for detection and tracking. Finally, we describe a robotic application which successfully employs the proposed vision-based tracking framework in order to infer the intentions of agents in the monitored environment, before their actions are finalized.

# Dedication

I would like to dedicate this to my wonderful parents, Ashraf and David who showed me the path to the vast universe of science, showed me how to climb the stairs to success and supplied my life with their love, care and understanding. I would specially like to dedicate this to my lovely partner, Keith for his unconditional love, constant help and patient understanding.

# Acknowledgments

During my studies in the University of Nevada, Reno, I had been fortunate to work with incredible scientists and researchers within the department of Computer Science and Engineering and the Computer Vision Laboratory. First and foremost, I would like to thank my supervisor Dr. Mircea Nicolescu who helped me learn about computer vision and pattern recognition and patiently supported me in conducting this research. I would like to thank my committee members Dr. George Bebis, Dr. Monica Nicolescu, Dr. Zhong Tian and Dr. Michael Webster for their useful comments which improved the quality of this dissertation.

I would also like to thank all of the members of Computer Vision group, especially Dr. Gholamreza Amayeh and Mr. Amol Amberdekar for their invaluable and helpful comments.

I would like to offer my deepest appreciations to my wonderful family; my parents Davood and Ashraf, my brother Abdolreza, and partner Keith, for their constant help, understanding, support and love.

This research was supported in part by the NSF-EPSCoR Ring True III award EPS0447416, the Office of Naval Research award N00014-06-1-0611, the University of Nevada Junior Faculty Research Grant Fund and by NASA under grant # NCC5-583.

# Table of Contents

<b>Abstract</b>	<b>i</b>
<b>Dedication</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>iv</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivations . . . . .	1
1.2 Objectives . . . . .	3
1.3 Contributions . . . . .	4
1.3.1 The Object Detection Component . . . . .	5
1.3.2 The Object Tracking Component . . . . .	8
1.4 Overview . . . . .	9
<b>2 The Object Detection Framework</b>	<b>10</b>
2.1 Goals and motivation . . . . .	10
2.2 Previous work . . . . .	12
2.3 Contributions . . . . .	16
2.4 The AKDE Method . . . . .	17

2.4.1	The theory . . . . .	17
2.4.2	The algorithm . . . . .	19
2.4.3	Performance evaluation . . . . .	24
2.5	The RM Method . . . . .	28
2.5.1	The theory . . . . .	28
2.5.2	The algorithm . . . . .	32
2.5.3	Performance evaluation . . . . .	35
2.6	The SVDDM Method . . . . .	37
2.6.1	The theory . . . . .	38
2.6.2	The algorithm . . . . .	42
2.6.3	Performance evaluation . . . . .	48
<b>3</b>	<b>The Object Tracking Framework</b>	<b>53</b>
3.1	Goals and Motivation . . . . .	53
3.2	Previous Work . . . . .	54
3.3	Contributions . . . . .	58
3.4	Spatio-Spectral Connected Component Tracker . . . . .	59
3.4.1	The visual tracking algorithm . . . . .	61
3.4.2	Performance evaluation . . . . .	70
<b>4</b>	<b>Comparison and Evaluation</b>	<b>73</b>
4.1	Object Detection . . . . .	74
4.1.1	Rapidly fluctuating backgrounds . . . . .	75
4.1.2	Low contrast videos . . . . .	76
4.1.3	Slowly changing backgrounds . . . . .	77
4.1.4	Hand-held camera . . . . .	79
4.1.5	Non-empty backgrounds . . . . .	80

4.1.6	Convergence speed . . . . .	81
4.1.7	Sudden global changes . . . . .	82
4.1.8	Other difficult examples . . . . .	83
4.1.9	Quantitative evaluation . . . . .	84
4.1.10	Synthetic data sets . . . . .	85
4.1.11	Comparison summary . . . . .	91
4.2	Object Tracking . . . . .	93
4.2.1	Frame rate and tracking speed . . . . .	93
4.2.2	Tracking performance in the presence of collision . . . . .	95
4.2.3	Other challenging experiments . . . . .	97
<b>5</b>	<b>A Robotic Application</b>	<b>99</b>
5.1	Introduction . . . . .	100
5.2	Context-Based Intent Recognition . . . . .	102
5.2.1	Hidden Markov Models for Representing Activities . . . . .	102
5.2.2	Context Models for Inferring Intent . . . . .	102
5.2.3	Intention-Based Control . . . . .	104
5.3	Experimental Validations . . . . .	104
5.3.1	The results . . . . .	106
<b>6</b>	<b>Conclusions</b>	<b>112</b>
6.1	Summary . . . . .	112
6.2	Conclusions . . . . .	115
6.3	Future work . . . . .	116
	<b>Bibliography</b>	<b>118</b>

## List of Tables

2.1	Per-pixel memory requirements for the AKDE method. . . . .	27
2.2	Per-pixel computational cost for the AKDE method. . . . .	28
2.3	Per-pixel memory requirements for the RM method. . . . .	36
2.4	Per-pixel computational cost for the RM method. . . . .	37
2.5	Per-pixel memory requirements for the SVDDM method. . . . .	49
2.6	Speed comparison of the incremental, online and canonical SVDD. . .	50
2.7	The number of support vectors retained. . . . .	51
4.1	Quantitative evaluation . . . . .	84
4.2	Comparison of FRR and RR for different classifiers . . . . .	86
4.3	Need for manual optimization and number of parameters. . . . .	87
4.4	Comparison of memory requirements for different classifiers. . . . .	87
4.5	comparison of incremental, online, and batch SVDD training . . . . .	88
4.6	Qualitative comparison of various classifiers . . . . .	91
4.7	Comparison of methods . . . . .	91
4.8	Scenarios where each method appears to be particularly suitable. . .	92
5.1	Quantitative Evaluation . . . . .	107

# List of Figures

1.1	Computational Vision as a part of visual processing applications . . .	2
1.2	Overview . . . . .	3
2.1	Examples of challenges in quasi-stationary backgrounds . . . . .	11
2.2	Diagonal vs full kernel matrix . . . . .	18
2.3	The proposed AKDE modeling algorithm. . . . .	20
2.4	Adaptive threshold map . . . . .	22
2.5	Enforcing spatial consistency . . . . .	24
2.6	Effect of the number of training samples . . . . .	25
2.7	Recursive modeling . . . . .	29
2.8	The RM algorithm . . . . .	32
2.9	Addressing complex description with SVDD . . . . .	40
2.10	The SVDDM algorithm . . . . .	43
2.11	Inequality and equality constraints on the two Lagrange multipliers .	46
2.12	Speed and the number of support vectors for the three algorithms . .	51
3.1	The object tracking overview . . . . .	59
3.2	The spatio-spectral object tracking algorithm . . . . .	63
3.3	Objects photometric appearances . . . . .	65
3.4	The collision resolution algorithm . . . . .	68
4.1	Rapidly fluctuating background . . . . .	75

4.2	Low contrast videos . . . . .	76
4.3	Slowly changing background . . . . .	77
4.4	Hand-held camera . . . . .	78
4.5	Non-empty background . . . . .	80
4.6	Convergence speed. . . . .	81
4.7	Sudden global changes . . . . .	83
4.8	Other difficult examples . . . . .	84
4.9	Comparison between different classifiers on a synthetic data set . . . . .	86
4.10	Incremental SVDD Comparison with its Online counterpart . . . . .	89
4.11	Incremental, online, and batch SVDD comparison . . . . .	90
4.12	Visual object tracking comparison . . . . .	94
4.13	Visual object tracking comparison under collision . . . . .	96
4.14	Challenging condition for visual object tracker . . . . .	97
5.1	Object tracking frameworks for static and dynamic cameras . . . . .	100
5.2	Simple intention recognition. . . . .	105
5.3	Similar looking activities. . . . .	109
5.4	The theft scenario. . . . .	109
5.5	The unattended bag scenario. . . . .	110

# Chapter 1

## Introduction

In many applications researchers and operators need to process visual information, presented as images and video frames, in order to perform high-level tasks such as the diagnosis of a particular situation. In video surveillance systems, for example, human operators try to detect possible security threats by looking at multiple screens. Scientists in bio-medicine use microscopic images and videos to diagnose particular anomalies in tissues or to detect specific cells or microbes. Understanding the context of the visual information while watching lengthy videos, multiple screens, and blurred or fuzzy images, becomes a tedious process which makes human operators vulnerable to making mistakes. Since the computational power of microprocessors has greatly improved in the recent years, many scientists have begun investigating the possibility of performing the low-level processing of images and video frames with the help of these computers.

### 1.1 Motivations

Computers perform redundant and repetitive tasks with far greater precision than humans. By developing computational frameworks which process the raw information encoded within image pixels and generating relevant mid-level information, the

accuracy, efficiency, and robustness of the over-all visual application will be improved. Computer vision, as a subdivision of the discipline known as Artificial Intelligence, presents various mathematical and algorithmic frameworks which generate reliable and accurate understanding of relevant contextual information by processing raw images and videos.

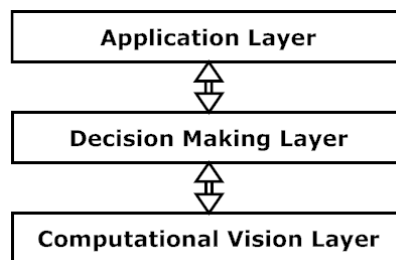


Figure 1.1: Different layers in any application which uses visual information.

Figure 1.1 shows a three-layer model for applications which uses visual information as the input data to perform specific tasks. In this model the top layer, the *Application Layer*, is responsible for the visualization and/or preparation of the final results of the process in a manner useful to human operators. The middle layer of the model takes mid-level information related to scene interpretation and contextual information generated by the lower level i.e.the *Computational Vision Layer*. This layer is responsible for processing raw pixel values and generating meaningful and relevant mid-level data required for the upper layers.

In this dissertation a computational vision framework is designed and implemented which looks at the detection and tracking of objects of interest in videos. In conjunction with the *Decision Making Mechanism* and the *Application Layer* implementation, the proposed object tracking framework can be employed in a wide range of applications from video surveillance to robotics. As evidence of its efficiency, the proposed algorithm is currently used in a robotic application which employs robots capable of inferring the intent of the people they survey.

## 1.2 Objectives

When processing video frames there are several crucial questions. The answers to these questions define the behavior of the system. One of the most basic and important component of any video processing mechanism is the detection of objects of interest in the scene. In many applications the background of the video may undergo changes. However, not all changes represent foreground objects. In addition to detection of these objects, a reliable and robust video processing mechanism should be able to generate models for the detected objects and to track them throughout the video.

While all object tracking algorithms consider both components, some concentrate more heavily on one of these components, while others consider the importance the two components more evenly. The *object detection* mechanism, deals with the dynamics of the scene and addresses the learning of the priors of the background. The *object tracking* part of the system maintains these models and searches in new frames to find the suitable matches for the models. Each of these components may employ different approaches to address their corresponding problems. The overview of a general video tracking framework, with its components, is shown in Figure 1.2.

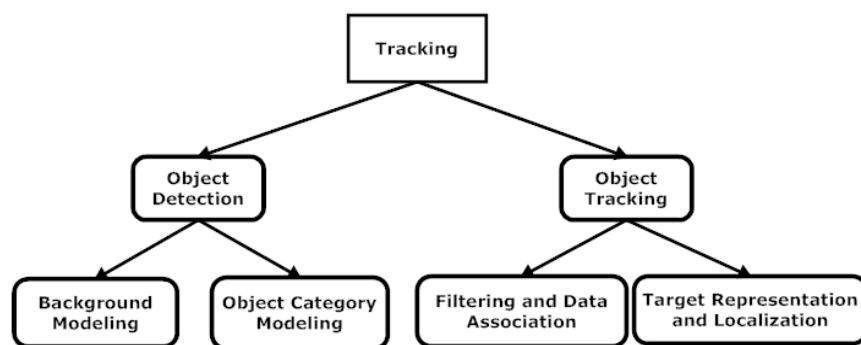


Figure 1.2: The general overview of different components of a general object tracking framework .

In order for a reliable object tracking framework to be successful, it has to address

a number of issues and challenges within its two components. In the object detection stage, although the camera can be assumed to be stationary, there may be inherent changes in the background itself, such as fluctuating monitors, waving trees, rain or snow, etc. These changes require sophisticated pixel modeling in order to account for the dynamics of the scene. The target tracking part of the system should be robust and reliable to the possibility of local illumination changes, efficient tracking of multiple objects, and the possibility of resolving the collision and partial occlusions.

### 1.3 Contributions

As discussed in the previous section, most object tracking algorithms divide the problem into two components; the object detection and the object tracking stages. In this project we developed an object tracking mechanism which treats these two components equally. It is a reasonable expectation that by employing a reliable object detection algorithm the performance of the tracking component of the system can be improved.

The proposed object detection component aims at a robust and efficient detection of objects in videos with backgrounds which might have inherent changes present within the field of view, such as waving flags, rain or snow, flickering screens, etc. In the literature, these background are referred to as quasi-stationary. These background changes make the detection of objects of interest particularly difficult.

The second component of the proposed technique is responsible for building appearance models of the detected objects, robustly maintaining these models, and accounting for the possibility of multiple object collisions as well as partial or complete occlusion. Since the proposed object tracking algorithm is to be employed as the computational vision layer of other visual processing applications, both object tracking and detection components of the proposed framework have to be fast and to

approach real-time requirements of the system. In the following, the implementation of the two proposed components are highlighted.

### 1.3.1 The Object Detection Component

Because of the aforementioned issues in detecting foreground regions in videos with quasi-stationary backgrounds researchers have proposed alternative approaches. By surveying the existing methods in the literature, it appears that each of the existing systems addresses issues in specific scenarios. In the present research, our focus is to find a common ground that would cover a general scenario for background modeling. The contributions of this dissertation can be summarized as follows:

1. *Find an appropriate approach to the problem of detecting foreground regions in videos with quasi-stationary background.*

This approach should address the multi-modality of the background and the scene-independence issue. Our proposed solution is based on a non-parametric framework that addresses the issues in the literature. This is the base-line system and is called the Adaptive Kernel Density Estimation (AKDE).

2. *Investigate the efficiency of the base-line system and derive a more universal framework upon this system.*

The proposed universal method is called Recursive Modeling (RM). This technique addresses the issue of limited number of training data, which compromises the accuracy of the system.

3. *Find limits on the accuracy of the proposed universal model and devise an alternative system which is not bound to these limitations.*

This alternative technique is the Support Vector Data Description Modeling (SVDDM) which explicitly addresses the single-class classification of background

samples. Since this technique does not estimate the pixel probability density function, it is not limited to the model probability estimation error.

In the following a brief overview of each method is presented. Their details and performance evaluations are presented in Sections 2.4, 2.5, and 2.6, of Chapter 2, respectively.

### **The AKDE**

The theory behind the AKDE algorithm is to estimate the probability of each pixel being background, based on a number of samples in its history. One advantage of the AKDE method over existing traditional density estimation techniques is that in the proposed algorithm instead of using a single threshold for all pixels in the scene a different threshold for each pixel is used. By employing these localized thresholds the system works efficiently on different video scenes and is more robust to local changes in the same scene. Also the AKDE method exploits the dependency between pixel color components, thus leading to a more accurate background model.

### **The RM**

The RM method is a recursive counterpart for the AKDE technique which uses pixel intensity/color values in new frames to update the background model at that pixel location. Since the update process is performed continuously, the background model converges to the actual model as more frames are processed. This gives the RM method the ability to detect foreground regions in situations during which the background changes occur slowly and do not fit in a small number of training frames. Also in videos without a set of empty background frames the RM technique has the ability to generate a clear background model. The proposed RM method uses a schedule for learning, which makes the background model converge to the actual model and

to recover from the expired model faster. It should be noticed that a non-parametric recursive modeling scheme has not been investigated in the literature.

### **The SVDDM**

Unlike the two statistical modeling methods – the AKDE and RM – the SVDDM method does not use probability of pixels for classification purposes. Instead, it generates an analytical description for the boundary of each pixel value, based on its history. The main advantage of this approach is that it explicitly addresses the single-class classification problem. Before samples of foreground appear in the scene, training samples belong to the background class. Therefore the SVDDM is not limited to the accuracy of the estimation of probability density function and the lack of foreground samples. Another desirable property of the SVDDM technique is its memory efficiency. Unlike the AKDE method, the SVDDM does not store all of the training samples to perform the classification which makes it quite memory efficient as an approach to foreground region detection, particularly when compared to other existing techniques.

### **The Incremental SVDDM**

Although the SVDD modeling is an elegant method which addresses the single class classification problem of background modeling, its major draw back is the issue of training of the SVDD – which is a quadratic programming (QP) problem. A novel method is investigated and developed to efficiently train a SVDD for each background pixel. The advantages of this technique, called Incremental SVDD, are its low memory requirements and its efficiency in terms of speed. The proposed method runs in constant time with respect to the size of the training data set, since its retraining is performed only on the support vector working set.

### 1.3.2 The Object Tracking Component

After the objects of interest are detected, the problem of tracking them is addressed. This stage of the process deals with the tracking of objects and generating their tracking trajectories. Several issues in visual object tracking applications are discussed. After investigating the limitations of the object tracking mechanisms in the literature, the result of object detection are used to improve the tracking efficiency and its robustness. The proposed object tracking algorithm uses spatio-spectral connected component processing in conjunction with a statistical correspondence matching.

#### The Spatio-Spectral Connected Component Tracker

Since the proposed object tracking mechanism first uses a background modeling approach to detect foreground object efficiently, it is reasonable to take advantage of this information. Employing the detected objects from the detection phase of the algorithm drastically enhances and improves the performance of our object tracking technique.

The object tracking algorithm follows a *target representation and localization* approach. This technique has been shown to perform faster than the techniques which employ *data association and filtering* approaches. In order to perform efficiently, the objects photometric and geometric appearances are modeled in a spatio-spectral connected component processing mechanism. The existing techniques perform a localization search over a window around the previous location of the object to find potential target matches. Instead, we perform the search for potential target matches over the newly detected connected components in new frames. As a result of this process the proposed tracking algorithm performs more efficiently compared to that of the state-of-the-art.

One major issue in visual object tracking is the presence of partial or complete

occlusion. Since, many tracking algorithms should work in cluttered environments and also be able to track multiple moving objects, addressing the occlusion issue efficiently becomes very important. While multiple objects are moving in the scene a particular type of occlusion is inevitable when one object occludes one or more other moving objects. This scenario is referred to as a *collision*. We propose a sub-module in our object tracking algorithm which detects the possibility of collision between two or more objects. Once a collision is detected our algorithm takes the appropriate steps to resolve the collision while still tracking the objects independently.

## 1.4 Overview

The rest of this dissertation is organized as follows. Chapter 2 discusses our approaches in robustly detecting objects of interest. In this chapter, Section 2.4 discusses the theory behind the AKDE method and its implementation and gives a detailed analysis of its performance. In Section 2.5, both the underlying theory and a detailed algorithm of the RM technique are presented. Its performance is evaluated. Section 2.6 presents our incremental SVDDM method together with a detailed assessment of its performance. Chapter 3 presents the proposed object tracking mechanism and discusses its performance briefly. Experimental results of the proposed techniques are presented in Chapter 4. Sections 4.1 and 4.2 evaluate the performance of our proposed object detection and tracking algorithms, respectively. In these sections we compare the performances of our algorithms with existing techniques for a variety of difficult scenarios. In Chapter 5 we present a robotic application which employs the proposed visual object tracking algorithm to recognize the intents of people and robots. Finally, Chapter 6 concludes this study as future directions for the discipline are discussed.

## Chapter 2

# The Object Detection Framework

In this chapter we define the problem of foreground segmentation in videos with quasi-stationary backgrounds, discuss its application and explore its difficulties. Existing techniques that aim to solve this problem are briefly described and their limitations are investigated. Finally, the approach taken in this research to overcome limitations of currently existing methods is discussed.

### 2.1 Goals and motivation

Detecting foreground regions in videos is one of the most important tasks in high-level video processing applications. There are generally two approaches in treating the object detection problem in video sequences. The top-down approach detects moving regions in videos sequences by finding the areas which do not belong to the background. This methodology first models the background and its inherent changes in order to find the objects of interest. In the bottom-up approach, the photometric and/or geometric models for objects of interest should be generated. These models are then used to perform the search for potential instances of these objects in video frames. In contrast with the top-down approach, this framework models the foreground objects categories instead of the background of the video.

One of the major issues in detecting foreground regions using the top-down approach is the presence of inherent changes in the background such as; fluctuations in monitors and fluorescent lights, waving flags or trees, water surfaces, etc. In these cases the background may not be completely stationary. Furthermore, the background may not appear empty in any image across the sequence, making the background modeling even more problematic. These difficult situations are illustrated in Figure 2.1.



Figure 2.1: Examples of challenges in quasi-stationary backgrounds: (a) Fluctuating monitors. (b) Rain/Snow. (c) Waving tree branches. (d) Non-empty background.

In the presence of these types of backgrounds, referred to as quasi-stationary, a single background frame is not enough to accurately detect moving regions. Therefore the background of the video has to be modeled in order to detect foreground regions which are the objects newly introduced to the scene.

There is also great diversity in scenarios where the background modeling techniques are used to detect foreground regions. Applications vary from indoor scenes to outdoor, from completely stationary to dynamic backgrounds, from high quality videos to low contrast scenes and so on. Therefore, designing a single system that addresses all possible situations is one of the main issues to address in background modeling.

The purpose of this project is to propose a non-parametric approach to detection of foreground regions in videos with quasi-stationary backgrounds. In order for the system to handle quasi-stationary backgrounds and overcome the uncertainty in the

background values for each pixel a non-parametric approach has been sought and is proposed.

In this dissertation, three different techniques based on the non-parametric solution are proposed. The goal of these algorithms is to segment a video into foreground/background regions without the need to specify and fine-tune their parameters. The proposed methods should also be scene-independent in the scenarios suitable for its design.

## 2.2 Previous work

As previously discussed there are two approaches in addressing the overall object detection problem; namely the top-down and the bottom-up approaches. First, we explore the bottom-up approach in the literature. In order to detect moving objects in videos the regions of interest could be directly detected if their photometric and geometric models are learned a priori. These models should be affine covariant and robust representatives of the object of interest category. Such regions can be categorized; as Harris-Affine [54], Hessian-Affine [53], Maximally Stable Extremal Regions (MSER) [49], Intensity Based Regions [94], Edge Based Region [95], and Salient Regions [35].

Various methods have been recently introduced to detect object categories of images in general and humans in particular. In [19], the authors used a constellation model [20] to represent object categories by maintaining a probabilistic representation of object parts. SIFT features [47] are employed in the "Bag of Keypoints" method proposed by Csurka *et al.* in [9]. These techniques however are useful to detect any object category in images given the category is previously learned by the system. Sivic and Zisserman in [73] proposed a method to retrieve videos from covariant features they share with a specific image by combining Harris-Affine points and Maximally Stable Regions.

In order to detect humans in videos, several methods are proposed based on both geometric and photometric invariant features unique to the human body. Dalal and Triggs in [10] proposed a method to detect human bodies in image by employing a Histogram of Oriented Gradients (HoG). This method uses overlapping windows of various sizes and employs a voting mechanism to generate a histogram map of gradients of the image from different directions. A linear SVM is used to train the system for the category of human bodies and the classifier is able to detect human bodies in different conditions in images. They enhanced their HoG method to account for motion flow and the appearance of people in videos to detect humans [11].

Ramanan *et al.* in [65] proposed a geometric approach to detect human bodies in video sequences. They assumed that a human body is composed of a torso which can be approximated by a rectangle after detecting edges in the video frames. Once rectangles are found in the image, a probabilistic model for different body parts is shaped to further improve the detection rate.

Since many visual surveillance applications, are required to run in real-time, the computational complexity of the tracking mechanism is critical. One of the main drawbacks of the top down approaches is their computation complexity. The use of object category models in the top down approaches, require intensive training over a large data base of object categories. This makes these methods unlikely candidates for real-time applications. Another issue with object category modeling is its dependence on certain poses by objects and different viewpoints. For example a system trained to detect humans in an upright position may not be able to reliably detect other human poses, unless it is properly trained.

In most visual surveillance systems, stationary cameras are typically used. However, as mentioned earlier, the background may not be completely stationary. As a result, using a single background frame is not appropriate for detecting moving

regions. Pless *et al.* [62] evaluated different models for dynamic backgrounds. Typically, background models are defined independently on each pixel, and depending on the complexity of the problem employ the expected pixel features (i.e. colors) [16], [17], [18], [70] or consistent motion [61], [100]. They also may employ pixel-wise information [99] or regional models of the features [93], [28] and [50]. To improve robustness to noise, spatial [57] or spatio-temporal [46] features may be used.

In [99] a single 3-D Gaussian model for each pixel in the scene is built, in which the mean and covariance of the model are learned in each frame. This system tried to model the noise and used a background subtraction technique to detect those pixels whose probabilities are smaller than a threshold. However, the system failed to label a pixel as foreground or background when it has more than one modality due to fluctuations in its values, such as a pixel belonging to a fluctuating monitor.

Kalman filtering [43], [36], [37] is also used to update the model while linear prediction using Wiener filtering is presented in [93]. These background models were unable to represent multi-modal situations.

A mixture of Gaussians modeling technique was proposed in [75], [74] and [22] to address the multi-modality of the underlying background. In this technique background pixels are modeled by a mixture of a number of Gaussian functions. During the training stage, parameters of each Gaussian and their weights are trained and used in the background subtraction where the probability of each pixel is generated, using the mixture of Gaussians. The pixel is labeled as foreground or background based on its probability.

There are several shortcomings for mixture learning methods. First, the number of Gaussians need to be specified. Second, this method does not explicitly handle spatial dependencies. Also, even with the use of incremental expectation maximization, the parameter estimation and its convergence is noticeably slow where the Gaussians

adapt to a new cluster. The convergence speed can be improved by sacrificing memory as proposed in [51] and [52], limiting its applications where mixture modeling is pixel-based and over long temporal windows.

A recursive filter formulation is proposed by Lee in [45] to speed up the convergence. However, the problem of specifying the number of Gaussians as well as the adaptation in later stages still exists. Also this model does not account for situations in which the number of Gaussians changes due to occlusion or uncovered parts of the background.

In [18], El Gammal *et al.* proposed a non-parametric kernel density estimation method (KDE) for pixel-wise background modeling without making any assumption about its probability distribution. Therefore, this method can easily deal with multimodality in background pixel distributions without specifying the number of modes in the background. However, there are several issues to be addressed using non-parametric kernel density estimation.

These methods are memory and time consuming since for each pixel in each frame the system has to compute the average of all kernels centered at each training sample. Also, the size of the temporal window used as the background model needs to be specified. Too small a window increases speed, while it does not incorporate enough history for the pixel, resulting in a less accurate model. Also the adaptation will be problematic by using small window sizes. Increasing the window size improves the model accuracy but at the cost of higher memory requirements and slower convergence. Finally, the non-parametric KDE methods are pixel-wise techniques and do not use spatial correlation of pixel features. In order to adapt the model a sliding window is used in [55]. However the model convergence is problematic in situations where the illumination suddenly changes.

In order to update the background for scene changes such as moved objects, parked

vehicles or opened/closed doors, Kim *et al.* in [41] proposed a layered modeling technique. This technique needs an additional model called "cache" and assumes that the background modeling is performed over a long period of time. It should also be used in a post-processing stage after the background is modeled.

Another approach to model variations in the background is to represent these changes as different states, corresponding to different environments; such as lights on/off, night/day, sunny/cloudy. For this purpose Hidden Markov Models (HMM) have been used in [66] and [76]. However these techniques suffer from slow model training speed and are sensitive to model selection and initialization.

## 2.3 Contributions

By evaluating the existing methods for the background modeling in the literature it becomes clear that each of the existing algorithms addresses issues differently in specific scenarios. In the present research, our focus is to find a common ground that would cover a general scenario for background modeling. Our major contributions to the object tracking field are to find an appropriate approach to the problem of detecting foreground regions in videos with quasi-stationary background. This approach should address the multi-modality of the background as well as its scene-independence.

We proposed a base-line solution based on a non-parametric framework called the Adaptive Kernel Density Estimation (AKDE). We investigated the efficiency of this base-line approach and derived a more universal framework. The proposed universal method is called Recursive Modeling (RM). This technique addresses the issue of limited number of training data, which compromises the accuracy of the system.

This research found limits on the accuracy of the proposed universal model and devise an alternative system which is not bound to these limitations. This alterna-

tive technique is the Support Vector Data Description Modeling (SVDDM) which explicitly addresses the single-class classification of background samples. Since this technique does not estimate the pixel probability density function, it is not limited to the model probability estimation error.

Finally, we compared the performance of the proposed techniques with the existing methods in the literature and investigated scenarios suitable for each of the three proposed methods. Based on the scenario in which the background modeling is to be used we are able to choose the most appropriate algorithm from the proposed techniques.

## 2.4 Adaptive Kernel Density Estimation (AKDE)

In this section a technique based on Adaptive Kernel Density Estimation (AKDE) is presented. A detailed description of the research on AKDE has also been published in [81], [79], and [86]. This method seeks to classify pixels as foreground or background based on a statistical approach. It estimates the probability density function of the background at each pixel location. The estimated probabilities of pixels in new frames are then compared to trained thresholds to label them as foreground or background. First, the theory behind this technique is presented in Section 2.4.1. Then the algorithm is discussed in detail in Section 2.4.2 where the three stages of the system are explained. Section 2.4.3 presents a performance evaluation of this method in terms of memory and computation cost.

### 2.4.1 The theory

In the AKDE method a non-parametric kernel density estimation model for each pixel is generated and its classifier is trained. It uses the history of pixel values as training

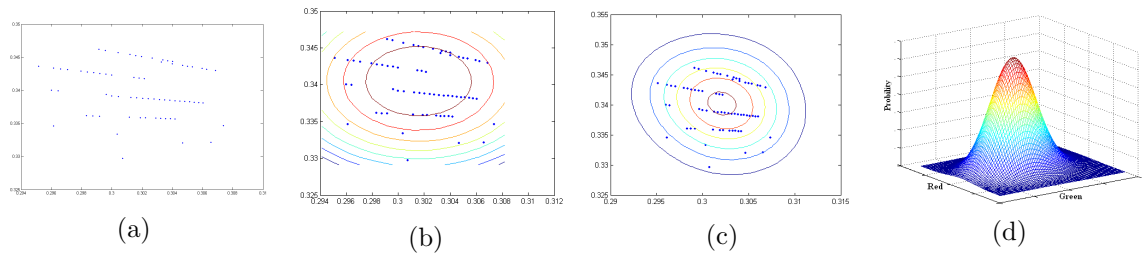


Figure 2.2: Diagonal vs full kernel matrix: (a) Pixel chrominance scatter plot ( $c_r, c_g$ ). (b) Probability constant contours using diagonal covariance matrix. (c) Probability constant contours using full covariance matrix. (d) Probability density using full covariance matrix.

samples and estimates the probability of each pixel being background in new frames to label them as foreground or background. The estimated probability for each pixel in the new frames is computed by:

$$P_t(Bg|\mathbf{x}_t) = \frac{1}{N2\pi|\boldsymbol{\Sigma}|^{1/2}} \sum_{i=1}^N e^{[-\frac{1}{2}(\mathbf{x}_t - \mathbf{x}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_t - \mathbf{x}_i)]} \quad (2.1)$$

where  $\mathbf{x}_t$  is the pixel feature vector at time  $t$  and  $\mathbf{x}_i, i = 1, 2, \dots, N$ , are its values in the training sequence.  $\boldsymbol{\Sigma}$  is a positive definite symmetric kernel bandwidth covariance matrix and  $N$  is the number of frames used to train the background model. In order to capture dependencies between features for each pixel,  $\boldsymbol{\Sigma}$  has to be a full (non-diagonal) matrix.

The effect of using a full covariance matrix can be observed in Figure 2.2. By using a full covariance matrix ( $\boldsymbol{\Sigma}$ ) in equation (2.1) we do not impose an assumption of feature independence on our estimation. If we assumed that features for each pixel are independent then a simplified version of equation (2.1) could be used, where the covariance matrix is diagonal. However as it can be seen from Figure 2.2 by using chrominance features, the independence assumption is not valid and the full covariance matrix results in a more accurate density estimation, as opposed to the diagonal covariance matrix proposed in [18].

Since in the AKDE method no assumptions are made on the covariance matrix  $\Sigma$ , any features can be used for pixels. Because color is the easiest and one of the most reliable feature to extract we use chrominance values for each pixel as its features. That is, given color values in *RGB* space, we determine red ( $c_r$ ) and green ( $c_g$ ) chrominance values by:

$$\begin{aligned} c_r &= \frac{R}{R + G + B} \\ c_g &= \frac{G}{R + G + B} \end{aligned} \quad (2.2)$$

Thus the feature vector for each pixel at a given time  $t$  is defined by:

$$\mathbf{x}_t = \left[ c_r(t) \quad , \quad c_g(t) \right]^T \quad (2.3)$$

Due to limited memory and computational power we need to store a rather short term memory of the background frames as training samples. This makes the non-parametric kernel density estimation dependent on the choice of its kernel bandwidth. In order to achieve an accurate and scene-independent background model which is adaptive to the spatial information in the scene such as different changes in the background the kernel bandwidth matrix needs to be trained.

### 2.4.2 The algorithm

Figure 2.3 shows the pseudo-code for the AKDE algorithm, consisting of three major stages: training, classification and update. In the training stage the background model is generated, and for each pixel its model values are used to estimate the probability of that pixel to be background in new frames. The proposed method detects foreground regions by solving a classification problem. However, it should be noted that we only have samples of the background class, before any foreground

```

N: size of training buffer
For each frame at time t
  1. Training stage
  for each pixel (u,v)
    - Calculate kernel covariance  $\Sigma(u,v)$  and threshold  $th(u,v)$ .
  2. Classification stage
  for each pixel (u,v)
    - Compute median of probability in neighborhood:  $Med(u,v)$ 
    - if  $Med(u,v) \leq th(u,v)$ 
      then  $FG_t(u,v) = 1$  % (Foreground)
      else  $FG_t(u,v) = 0$  % (Backgrounds)
  3. Update stage
    - if  $size(FG) \geq 0.5 \text{ Image\_Size}$ 
      then % Global sudden change detected
        for each pixel (u,v) % Replace oldest frame (OF)
           $OF(u,v) \leftarrow I_t(u,v)$  % with current frame
        else % Gradual change
          for each pixel (u,v) % Replace background pixels in OF
             $OF(u,v | FG_t(u,v) = 0) \leftarrow I_t(u,v | FG_t(u,v) = 0)$ 

```

Figure 2.3: The proposed AKDE modeling algorithm.

object appears in the scene. Therefore, one can argue that this method deforms a bi-class classification and makes it into a single-class classification problem.

The only parameter in kernel density estimation is the kernel bandwidth. In theory, as the number of training samples reaches infinity the estimated density converges to the actual underlying density regardless of the kernel bandwidth value [12], [69]. Thus we refer to kernel density estimation as non-parametric, as opposed to the mixture of Gaussian techniques, in which a parametric form is assumed for the underlying probability density.

The AKDE technique has three parameters, the kernel covariance matrix  $\Sigma$ , localized thresholds  $th$  and the number of training samples  $N$ . Among these parameters, the covariance matrix and thresholds are trained in the training stage of the algorithm. The number of training frames is the only parameter that needs to be selected heuristically.

### Training stage

For each pixel the training samples are vectors  $\mathbf{X}_N = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ , from consecutive frames. The successive deviation of the above vectors is a matrix  $\Delta_X$  whose columns are defined by:

$$[\mathbf{x}_i - \mathbf{x}_{i-1}]^T \quad \text{with } i = 2, 3, \dots, N \quad (2.4)$$

For each pixel, the kernel bandwidth matrix is defined such that it represents the temporal scatter of training samples. Thus, the kernel bandwidth is defined by:

$$\Sigma = \text{cov}(\Delta_X) \quad (2.5)$$

From equations (2.4) and (2.5) it can be seen that for pixels with more feature changes through time, such as flickering pixels, the kernel bandwidth matrix has larger elements, while for pixels that do not change much, its elements are smaller. Also notice that the kernel bandwidth is drawn from the training samples without any assumption on features and their underlying probability density function. The estimated probability density function by using this adaptive kernel bandwidth is accurate, even with a small number of background training frames. Finally, since the kernel bandwidth matrix is computed using successive deviations in equation (2.4) it accounts for temporal dependencies in pixel feature vectors.

In the traditional foreground detection techniques, usually the foreground regions are detected by comparing the value or model of each pixel with its value or model in the background, and if this deviation is larger than a heuristically selected threshold it is selected as a foreground region. If we estimate the probability of each pixel in all of the background frames, given that all of these pixels are background, their probabilities should have large values, close to 1. Because of noise and the inherent

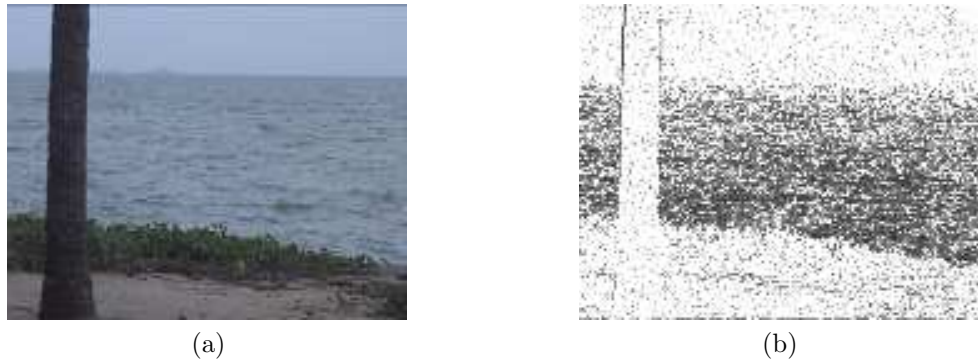


Figure 2.4: Adaptive threshold map: (a) An arbitrary frame. (b) Threshold map.

background changes however, pixels do not take a single value and therefore their probabilities become smaller. The probability of a pixel to be background is related to the amount of change that its features undergo in time. Therefore a single global threshold does not work well because pixels in different locations experience different amounts of change.

These threshold values need to be trained for each pixel during the training stage to build an accurate and automatic classifier. For each pixel the threshold value ( $th$ ) is selected such that its classifier results in 5% false reject rate. That is, 95% of the time the pixel is correctly classified as belonging to background:

$$\sum_{\substack{i=1 \\ P(Bg|x_i) \leq th}}^N P(Bg|x_i) \leq 0.05 \sum_{i=1}^N P(Bg|x_i) \quad (2.6)$$

This can be seen in Figure 2.4, where (a) shows an arbitrary frame of a sequence containing a water surface and (b) shows the trained threshold map for this frame. Darker pixels in Figure 2.4(b) represent larger threshold values and lighter pixels correspond to smaller threshold values. As it can be observed, the thresholds in the areas that tend to change more, such as the water surface, are lower than in those areas with less amount of change, such as the sky. This is because for pixels which change all the time, the certainty about the background probability values is less.

### Classification stage

In the training stage, the kernel bandwidth matrix  $\Sigma$  and its classification decision criterion  $th$  are determined for each pixel. The probability of each pixel in the new frame is then estimated using (2.1). If we directly apply the trained threshold of each pixel to its estimated probability then, due to strong noise, pixels may be erroneously classified. This happens when one pixel has been affected by noise and when the amount of noise is so strong it that changes its feature vector  $\mathbf{x}_t$  such that its estimated probability affects the decision results.

One of the properties of this type of noise is that, if strong noise affects a pixel, it is less likely to affect its neighborhood with the same strength. If a pixel in a region belonging to the background produces a fairly small probability because of noise, its neighboring pixels are expected to produce larger probabilities. Thus, using the median of estimated probabilities in a region around a pixel enforces the spatial consistency of its neighborhood. After estimating the probability of each pixel in the new frame, the median of probabilities in its 8-connected neighborhood is compared with its threshold to make the classification decision:

$$\text{Label}_t = \begin{cases} \text{Foreground} & \text{if } \text{median}(\text{Prob}_t) \leq th \\ \text{Background} & \text{otherwise} \end{cases} \quad (2.7)$$

Figure 2.5 shows the effect of enforcing spatial consistency using the median of probabilities in foreground region detection. As it can be seen, by applying the threshold on the median of estimated probabilities of pixels in a neighborhood, most of the noise can be suppressed.

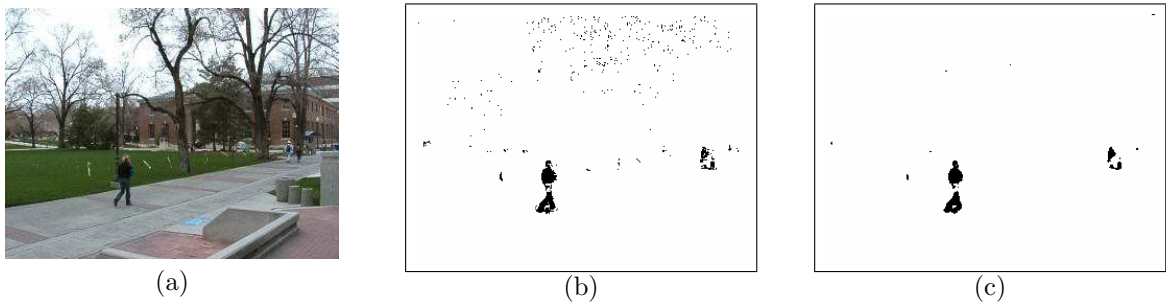


Figure 2.5: Enforcing spatial consistency: (a) Original frame (b) Detected foreground regions by applying thresholds directly on the estimated probability. (c) Detected foreground regions by applying thresholds on the median of probabilities in a neighborhood.

### Update stage

In the proposed AKDE method we use two different types of adaptation. To make the system adaptive to *gradual changes* in illumination, we replace pixels in the oldest background frame with those pixels belonging to the current background mask. In case of *sudden changes* in the illumination, the area of the detected foreground objects is checked. Once sudden change is detected in their area, the classification stage of the algorithm is suspended and new frames replace frames in the background training buffer.

Because the training stage of the algorithm is time consuming, the updating stage is actually performed every few frames, depending on the rate of changes and the processing power. In the current implementation the updating stage is performed every 10 frames.

### 2.4.3 Performance evaluation

In this section we evaluate the performance of the AKDE method in terms of memory requirements and computational cost.

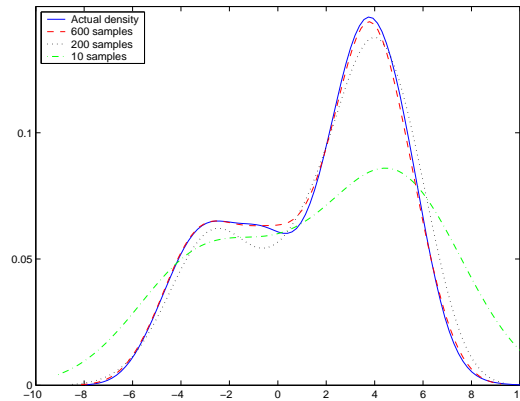


Figure 2.6: Effect of the number of training samples on the estimated density function.

## Parameters

In kernel density estimation methods the goal is to estimate the probability density function based on the training samples without making any assumptions on the underlying distribution. One important parameter in these methods is the number of training samples used to estimate the probability density. Other issues in this technique are the type of kernel, its bandwidth and the classification threshold. The kernel type is not very critical and usually Gaussian kernels are preferred. Other parameters such as the threshold and the kernel bandwidth matrix are trained during the training stage, discussed earlier in this chapter.

In Figure 2.6 the actual probability function of a randomly distributed population is shown by the solid line. The estimated probability density function converges to the underlying density by increasing the number of training samples as it can be seen in Figure 2.6. However, there is a trade-off between the number of training samples, memory requirements and the convergence speed of the algorithm. Our experiments show that using more than 250 samples makes the algorithm converge too slowly and requires too much memory to store the model. This issue is discussed later in this section.

## Memory requirements

- *Using only intensity values.*

Assume there are  $n$  1-D training samples given by  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ . The probability of a new sample  $\mathbf{x}_t$  is given by:

$$P(\mathbf{x}_t) = \frac{1}{\sqrt{2\pi\sigma n}} \sum_{i=1}^n e^{-\left(\frac{\mathbf{x}_t - \mathbf{x}_i}{\sqrt{2}\sigma}\right)^2} \quad (2.8)$$

where  $\sigma$  is the bandwidth of the Gaussian Kernel.

The system needs to store all the training samples in order to estimate the probability of a new sample. If only pixel intensity values are to be employed for each pixel,  $n$  values need to be stored. Given that these values range between 0 to 255, each intensity value is stored in 1 byte, resulting in  $n$  bytes per-pixel memory requirement. Also the system needs to store the kernel bandwidth and the thresholds for each pixel, which result in 2 floating numbers. Considering that each floating number can be stored on 4 bytes, 8 bytes per pixel are needed to store the kernel bandwidth and the threshold. This results in  $n + 8$  bytes memory requirement per pixel.

- *Using chrominance values.*

Similarly if there are  $n$  training samples composed of chrominance values for each pixel, the system needs to store  $8 \times n$  bytes per pixel. The kernel covariance matrix needs  $4 \times 4$  bytes and the threshold needs only 4 bytes per pixel, respectively. Thus the per-pixel memory requirements is  $8n + 20$ .

Table 2.1 shows the per-pixel memory requirements using only intensity values, red and green chrominance values and their combinations. From the above discussion we can conclude that the asymptotic memory requirement for the system is  $O(n)$ .

Table 2.1: Per-pixel memory requirements for the AKDE method.

Memory Req.	Intensity	Chrominance	Intensity+Chrominance
Bytes per pixel	$n+8$	$8n+20$	$9n+40$

That is, if the number of training samples reaches infinity, the memory requirements of the system grow linearly.

### Computation cost

- *Using only intensity values.*

If we only use pixel intensity values for  $n$  training samples per pixel according to equation (2.8) we need 2 additions and 2 multiplications for each training sample, resulting in  $2n$  addition and  $2n$  multiplication operations. Given the optimal implementation of the exponential function using look-up tables, its cost is equal to a memory indexing. This can be assimilated to a single addition operation:

$$\text{Per-pixel cost} = 5 \times n \quad (2.9)$$

- *Using chrominance values.*

Similarly if there are  $n$  training samples composed of red/green chrominance values for each pixel as in equation (2.3), the system requires 5 addition and 6 multiplication operations. With the optimal exponential function implementation, the computational cost of the system per pixel is:

$$\text{Per-pixel cost} = 13 \times n \quad (2.10)$$

Table 2.2 shows the per-pixel computational cost of the system using only intensity values or red/green chrominance values for each pixel. Given the optimal imple-

Table 2.2: Per-pixel computational cost for the AKDE method.

Operations	Addition	Multiplication	Asymptotic
Intensity	$2n$	$2n$	$O(n)$
Chrominance	$5n$	$6n$	$O(n)$

mentation of the exponential function and multiplication operations, the asymptotic per-pixel computational cost is  $O(n)$ . Note that this is the optimal asymptotic computational cost per pixel. The actual frame rate of the current implementation of the AKDE method is about 5 fps.

## 2.5 Recursive Modeling (RM)

This chapter describes a technique called Recursive Modeling (RM) for foreground region detection in videos. The theory behind this approach is to generate a histogram of the data samples, with the hope that when a large number of training samples are processed, the histogram estimates the actual probability of the underlying data. System details and its theory are explained in the following. We have published this technique in [80], [83], and [86].

### 2.5.1 The theory

Let  $x_t$  be the the intensity value of a pixel at time  $t$ . The non-parametric estimation of the background model that accurately follows its multi-modal distribution can be reformulated in terms of recursive filtering [86]:

$$\hat{\theta}_t^B(x) = [1 - \beta_t] \cdot \theta_{t-1}^B(x) + \alpha_t \cdot H_\Delta(x - x_t) \quad \forall x \in [0, 255] \quad (2.11)$$

$$\sum_{x=0}^{255} \theta_t^B(x) = 1 \quad (2.12)$$

where  $\theta_t^B$  is the background pixel model at time  $t$ , normalized according to (2.12).

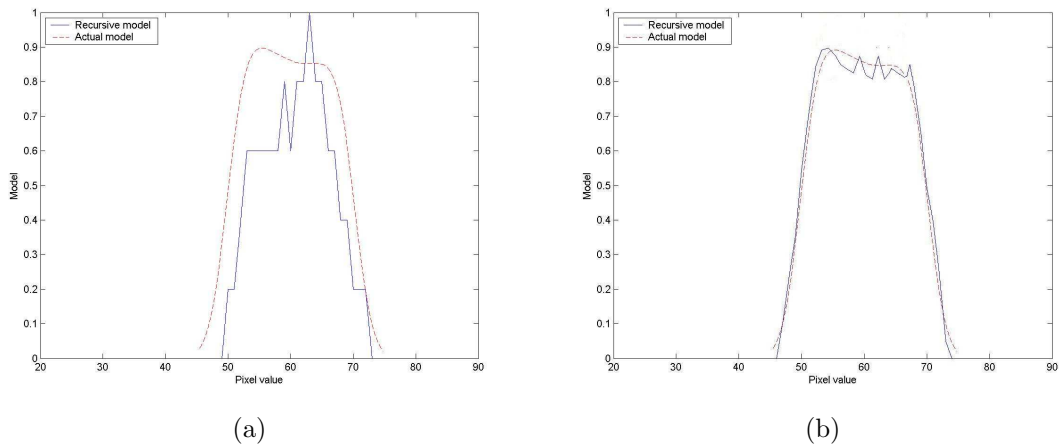


Figure 2.7: Recursive modeling: (a) Model after 10 frames.(b) after 200 frames.

$\hat{\theta}_t^B$  is updated by the local kernel  $H(\cdot)$  with bandwidth  $\Delta$  centered at  $x_t$ . Parameters  $\alpha_t$  and  $\beta_t$  are the learning rate and forgetting rate schedules, respectively. The kernel  $H$  should satisfy the following conditions:

$$\begin{aligned} \sum_x H_\Delta(x) &= 1 \\ \sum_x x \times H_\Delta(x) &= 0 \end{aligned} \quad (2.13)$$

These conditions should be satisfied to ensure that the kernel is normalized, symmetric and positive definite in case of multivariate kernels. Note that in this context there is no need to specify the number of modalities of the background representation at each pixel. In our implementation of the RM method we use a Gaussian kernel which satisfies the above conditions.

Figure 2.7 shows the updating process using our proposed recursive modeling technique. It can be seen that the trained model (solid line) converges to the actual one (dashed line) as new samples are introduced. The actual model is the probability density function of a sample population and the trained model is generated by using the recursive formula presented in equation (2.11).

In existing non-parametric kernel density estimation methods, the learning rate  $\alpha$

is selected to be constant and has small values. This makes the pixel model convergence slow and keeps its history in the recent temporal window of size  $L = 1/\alpha$ . The window size in non-parametric models is important as the system has to cover all possible fluctuations in the background model. That is, pixel intensity changes may not be periodic or regular and consequently do not fit in a small temporal window. In such cases larger windows are needed, resulting in higher memory and computational requirements to achieve accurate, real-time modeling.

Another issue in non-parametric density estimation techniques is that the window size is fixed and is the same for all pixels in the scene. However, some pixels may have less fluctuations and therefore need smaller windows to be accurately modeled, while others may need a much longer history to cover their fluctuations.

### Scheduled learning

In order to speed up the modeling convergence and recovery we use a schedule for learning the background model at each pixel based on its history. This schedule makes the adaptive learning process converge faster, without compromising the stability and memory requirements of the system. The learning rate changes according to the schedule:

$$\alpha_t = \frac{1 - \alpha_0}{h(t)} + \alpha_0 \quad (2.14)$$

where  $\alpha_t$  is the learning rate at time  $t$  and  $\alpha_0$  is a small target rate which is:

$$\alpha_0 = 1/256 \times \sigma_\theta \quad (2.15)$$

where  $\sigma_\theta$  is the model variance. The function  $h(t)$  is a monotonically increasing function:

$$h(t) = t - t_0 + 1 \quad (2.16)$$

where  $t_0$  is the time at which a sudden global change is detected. At early stages the learning occurs faster ( $\alpha_t = 1$ ), then it monotonically decreases and converges to the target rate ( $\alpha_t \rightarrow \alpha_0$ ). When a global change is detected  $h(t)$  resets to 1. In Chapter 4 we discuss the effect of this schedule on improving the convergence and recovery speed.

The forgetting rate schedule is used to account for removing those values that have occurred long time ago and no longer exist in the background. In the current implementation we assume that the forgetting rate is a portion of the learning rate  $\beta_t = l \cdot \alpha_t$ , where  $l \leq 1$ . In the current implementation  $l = 0.5$  is employed in all experiments. This accounts for those foreground objects that cover some parts of the background and after a sufficiently small period uncover that part of the background. This keeps the history of the covered background in short-term.

### **Incorporating color information**

In Section 2.5.1, we described the recursive learning scheme in 1-D where the background and foreground models are updated using the intensity value of pixels at each frame. To extend the modeling to higher dimensions and incorporate color information, one may consider each pixel as a 3 dimensional feature vector in  $[0, 255]^3$ . The kernel  $H$  in this space is a multivariate kernel  $H_\Sigma$ . In this case, instead of using a diagonal matrix  $H_\Sigma$  a full multivariate kernel can be used. The kernel bandwidth matrix  $\Sigma$  is a symmetric positive definite  $3 \times 3$  matrix. Given  $N$  pixels,  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ , labeled

<ol style="list-style-type: none"> <li>1. Initialization; <math>\Delta</math>, <math>\alpha_0</math>, <math>\beta</math>, <math>\kappa</math> and <math>th</math></li> <li>2. For each frame <ul style="list-style-type: none"> <li>For each pixel <ol style="list-style-type: none"> <li>2.1. <b>Training stage</b> <ul style="list-style-type: none"> <li>- Update <math>\alpha_t = \frac{1-\alpha_0}{h(t)} + \alpha_0</math> and <math>\Delta</math></li> <li>- Update <math>\theta_t^B = (1 - \beta_t)\theta_{t-1}^B + \alpha_t \cdot H_\Delta</math></li> <li>- If <math>\theta_t^B \leq th</math> then update <math>\theta_t^F = (1 - \beta_t)\theta_{t-1}^F + \alpha_t \cdot H_\Delta</math></li> </ul> </li> <li>2.2. <b>Classification stage</b> <ul style="list-style-type: none"> <li>- If <math>\ln(\text{med}(\theta_t^F)/\text{med}(\theta_t^B)) \geq \kappa</math> then label pixel as foreground.</li> </ul> </li> <li>2.3. <b>Update stage</b> <ul style="list-style-type: none"> <li>- Update <math>\kappa</math> and <math>th</math></li> </ul> </li> </ol> </li> </ul> </li> </ol>
--

Figure 2.8: The RM algorithm.

as background, their successive deviation matrix is a matrix  $\Delta_X$  whose columns are:

$$[\mathbf{x}_i - \mathbf{x}_{i-1}]^T \quad \text{with } i = 2, 3, \dots, N \quad (2.17)$$

The bandwidth matrix is defined such that it represents the temporal scatter of training samples:

$$\Sigma = \text{cov}(\Delta_X) \quad (2.18)$$

However, in the current implementation only red and green chrominance values are used. Also in order to decrease the memory requirements of the system we assumed that the two chrominance values are independent. Making this assumption results in a significant decrease in memory requirements while the accuracy of the model does not decay drastically. The red/green chrominance values are quantized into 256 discrete values.

## 2.5.2 The algorithm

The proposed method, in pseudo-code, is shown in Figure 2.8. There are three major steps in the RM method: training, classification and update stages, respectively. The role and results of each stage along with its details are presented in the following.

### Training stage

Before new objects appear in the scene, at each pixel all the intensity values have the same probability of being foreground. However, in each new frame the pixel background models are updated according to equation (2.11), resulting in larger model values ( $\theta^B$ ) at the pixel intensity value  $x_t$ . In essence, the value of the background pixel model at each intensity  $x$  is:

$$\theta_t^B(x) = P(\text{Bg}|x) \quad x \in [0, 255] \quad (2.19)$$

In order to achieve better detection accuracy we introduce the foreground model which in the classification stage is compared to the background model to make the decision on whether the pixel belongs to background or foreground. This foreground model is defined by:

$$\hat{\theta}_t^F(x) = [1 - \beta_t^F] \cdot \theta_{t-1}^F(x) + \alpha_t^F \cdot H_\Delta(x - x_t) \quad \forall x \in [0, 255] \quad (2.20)$$

$$\sum_{x=0}^{255} \theta_t^F(x) = 1 \quad (2.21)$$

Once the background model is updated, it is compared to the threshold  $th$ . If its value is less than this threshold the foreground model for that pixel value is updated according to (2.20) and (2.21).

### Classification stage

For each pixel at time  $t$  we use a function  $\theta_t^B$  for the background model and  $\theta_t^F$  for the foreground. The domain of these functions is  $[0, 255]^N$ , where  $N$  is the dimensionality of the pixel feature vector. For simplicity assume the one dimensional case again, where  $\theta_t$  is the background/foreground model whose domain is  $[0, 255]$ . From equation

(2.20), each model ranges between 0 to 1 and its value shows the amount of evidence accumulated in the updating process (i.e., the estimated probability). For each new intensity value  $x_t$  we have the evidence of each model as  $\theta_t^B(x_t)$  and  $\theta_t^F(x_t)$ . The classification uses a *maximum a posteriori* criterion to label the pixel as foreground:

$$\ln \left( \frac{\theta_t^B}{\theta_t^F} \right) \leq \kappa \quad (2.22)$$

### Update stage

In many applications with dynamic or quasi-stationary backgrounds, we need adaptive classification criteria. Because not all pixels in the scene follow the same changes, the decision threshold  $\kappa$  should be adaptive and independent for each pixel and has to be derived from the history of that pixel. Figure 2.4 explains this issue. The argument is similar to the issue of adaptive, localized threshold map discussed in chapter 2.4.

From the algorithm shown in Figure 2.8 it can be observed that there are two set of thresholds  $th$  and  $\kappa$ . Thresholds  $th$  for each pixel should adapt to a value where:

$$\sum_{x:\theta_t^B(x) \geq th} \theta_t^B(x) \geq 0.95 \quad (2.23)$$

For the other set of thresholds  $\kappa$ , we similarly use a measure of changes in the intensity at each pixel position. Therefore the threshold  $\kappa$  is proportional to the logarithm of the background model variance:

$$\kappa \approx \ln \left\{ \sum_{x=0}^{255} \left( \theta_t^B(x) - \text{mean}[\theta^B(x)] \right) \right\} \quad (2.24)$$

This ensures that for pixels with more changes, higher threshold values are chosen for classification, while for those pixels with fewer changes smaller thresholds are em-

ployed. It should be mentioned that in the current implementation of the algorithm, the thresholds are updated every 30 frames (which are kept as known background buffer and used to perform the adaptation process).

### 2.5.3 Performance evaluation

In this section the RM method performance is evaluated. As it will be discussed later the RM method memory requirements and computation cost are independent of the number of training samples. This property makes the RM method a suitable candidate to be used in scenarios where the background changes are very slow.

#### Parameters

In the RM method there are 5 parameters: the learning and forgetting rate  $\alpha$  and  $\beta$ , thresholds  $th$  and  $\kappa$ , and the bandwidth  $\Sigma$ . As described earlier in this chapter these parameters are trained and estimated from the data to generate an accurate and robust model. The reason that the RM technique is robust is that it uses most of the information in the data set and there is no limit on the number of training samples. With all parameters being automatically updated, the system performance does not depend on heuristically (and therefore scene dependent) values for these parameters.

#### Memory requirements

- *Using only intensity values.*

Since the model is a 1-D function representing the probability mass function of the pixel, and pixel intensity values range from 0 to 255, it only needs  $256 \times 4$  bytes per pixel to be stored. Notice that in this case, for each pixel the intensity (gray scale) values are integer numbers. Therefore, the histogram bin size is 1. If the memory of the system is scarce larger bin sized can be used to decrease

Table 2.3: Per-pixel memory requirements for the RM method.

Memory Req.	Intensity	Chrominance	Intensity+Chrominance
Bytes per pixel	1024	2048	3072

the memory requirements of the system.

- *Using chrominance values.*

In this case the model is 2-D and needs  $256^2 \times 4$  bytes in memory. The current implementation of the RM method uses a simple assumption of independence between color features which results in  $8 \times 256$  bytes memory requirements [83]. As mentioned in Section 2.4.1, color components are not independent. However, assuming that they are independent helps decreasing memory needs drastically while the accuracy does not decrease significantly.

Table 2.3 shows the memory requirements in bytes per pixel for the RM method, using intensity, chrominance values and their combinations, respectively. In conclusion the asymptotic memory requirement of the RM algorithm is constant  $O(1)$ .

### Computation cost

- *Using only intensity values.*

If we only use pixel intensity values for pixels according to equation (2.11) we need 256 addition and  $2 \times 256$  multiplication operations. This is because the model is updated by multiplying the kernel to the learning rate and adding this value to the old model from the previous frame. Both the kernel and the model range from 0 to 255.

- *Using chrominance values.*

Similarly, if we use 2-D chrominance values as pixel features and use the independence assumption discussed earlier, the system requires only  $2 \times 256$  addition

Table 2.4: Per-pixel computational cost for the RM method.

Operations	Addition	Multiplication	Asymptotic
Intensity	256	512	$O(1)$
Chrominance	512	1024	$O(1)$

and  $4 \times 256$  multiplication operations to update the model.

Table 2.4 summarizes the per-pixel computational cost of the RM algorithm using only intensity values or red/green chrominance values for each pixel. The asymptotic computation cost for this system is constant,  $O(1)$ , since the updating process merely consists of adding two 1-D functions. Note that this technique does not need to compute the exponential function and acts as an incremental process, updating the model at each frame using the kernel and the previous model. The algorithm is inherently fast and an efficient implementation runs in real-time reaching frame rates of 15 frames per second (fps).

## 2.6 Support Vector Data Description Modeling (SVDDM)

In this chapter a powerful technique in describing one class of known data samples, called Support Vector Data Description Modeling is presented. We have published this method in [82] and as a journal paper in [84]. Single class classifiers, also known as novelty detectors are investigated in the literature, [2], [3] and [33]. Our method trains single class classifiers for each pixel in the scene as their background model. The backbone of the proposed method is a theory based on describing a data set using their support vectors [90], [88], [89], [72]. In the following, details of the SVDDM and the algorithm which detects foreground regions based on this technique are presented.

### 2.6.1 The theory

A normal data description is a description which gives a closed boundary around the data. A simple normal data description can be considered as a sphere with center  $\mathbf{a}$  and radius  $R > 0$ , which encloses all of the training samples  $\mathbf{x}_i$ . The data description is achieved by minimizing the error function:

$$F(R, \mathbf{a}) = R^2 \quad (2.25)$$

subject to the constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2, \quad \forall i \quad (2.26)$$

In order to allow for outliers in the training data set, the distance of each training sample  $\mathbf{x}_i$  to the center of the sphere  $\mathbf{a}$  should not be strictly smaller than  $R^2$ . However, large distances should be penalized. Therefore, after introducing slack variables  $\epsilon_i \geq 0$  the minimization problem becomes:

$$F(R, \mathbf{a}) = R^2 + C \sum_i \epsilon_i \quad (2.27)$$

subject to the new constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \epsilon_i, \quad \forall i \quad (2.28)$$

where  $C$  controls the trade-off between the sphere volume and the description error.

In order to solve the minimization problem in equation (2.27), the constraints of equation (2.28) are introduced to the error function using Lagrange multipliers:

$$L(R, \mathbf{a}, \alpha_i, \gamma_i, \epsilon_i) = R^2 + C \sum_i \epsilon_i - \sum_i \alpha_i \left[ R^2 + \epsilon_i - (\|\mathbf{x}_i - \mathbf{a}\|^2) \right] - \sum_i \gamma_i \epsilon_i \quad (2.29)$$

where  $\alpha_i \geq 0$  and  $\gamma_i \geq 0$  are Lagrange multipliers.  $L$  should be maximized with respect to  $\alpha_i$  and  $\gamma_i$  and minimized with respect to  $R$ ,  $\mathbf{a}$  and  $\epsilon_i$ .

$$\frac{\partial L}{\partial R} = 0 : \quad \sum_i \alpha_i = 1 \quad (2.30)$$

$$\frac{\partial L}{\partial \mathbf{a}} = 0 : \quad \mathbf{a} = \frac{\sum_i \alpha_i \mathbf{x}_i}{\sum_i \alpha_i} = \sum_i \alpha_i \mathbf{x}_i \quad (2.31)$$

$$\frac{\partial L}{\partial \gamma_i} = 0 : \quad C - \alpha_i - \gamma_i = 0 \quad (2.32)$$

From equations (2.30) - (2.32) and the fact that all of the Lagrange multipliers are not negative, when we put  $0 \leq \alpha_i \leq C$ , Lagrange multipliers  $\gamma_i$  can be safely removed. By replacing results of (2.30)-(2.32) into (2.29) we have:

$$L = \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad \forall \alpha_i : 0 \leq \alpha_i \leq C \quad (2.33)$$

A set of  $\alpha_i$  values can be achieved by maximizing equation (2.33). If a sample  $\mathbf{x}_i$  satisfies the inequality in equation (2.28) its corresponding Lagrange multiplier will be zero ( $\alpha_i = 0$ ). For all the training samples for which the equality in equation (2.28) is satisfied Lagrange multipliers become greater than zero ( $\alpha_i > 0$ ). These are the possible scenarios for a given sample  $\mathbf{y}_i$ :

$$|\mathbf{y}_i - \mathbf{a}|^2 < R^2 \rightarrow \alpha_i = 0, \gamma_i = 0 \quad (2.34)$$

$$|\mathbf{y}_i - \mathbf{a}|^2 = R^2 \rightarrow 0 < \alpha_i < C, \gamma_i > 0 \quad (2.35)$$

$$|\mathbf{y}_i - \mathbf{a}|^2 > R^2 \rightarrow \alpha_i = C, \gamma_i = 0 \quad (2.36)$$

Note that from equation (2.31), the center of the sphere is a linear combination of the training samples. Only those training samples  $\mathbf{x}_i$  which satisfy the equality of equation (2.28) are needed to generate their description since their coefficients are

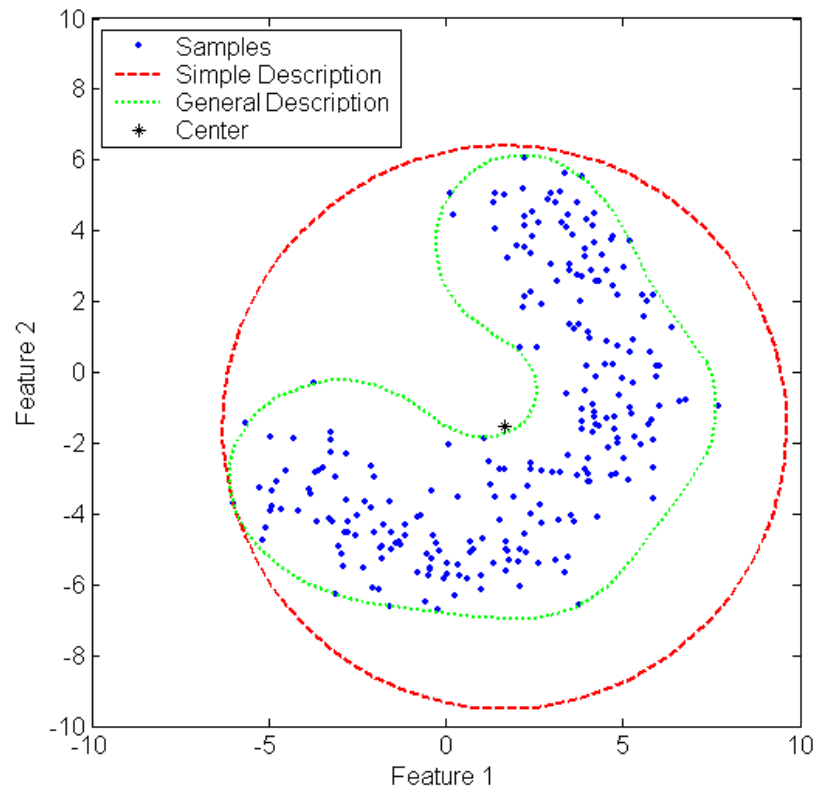


Figure 2.9: Description of a general, complex data set. The blue dots are the samples drawn from a general data set with complex distribution sketched in the 2-D feature space. The red circle shows the description if the simple dot product is used to answer the Lagrange optimization problem. The green description boundary is achieved by employing a kernel in the Lagrange equation.

not zero. Therefore these samples are called *Support Vectors*.

The above theory describes the normal data description which is the smallest sphere surrounding the training data. However, this simple normal description is not enough for more complex data points which do not fit into a sphere, i.e. their description needs more complex boundaries.

To achieve a more flexible description, instead of a simple dot product of the

training samples  $(\mathbf{x}_i \cdot \mathbf{x}_j)$ , we can perform the dot product using a kernel function:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (2.37)$$

This is done by using a mapping function  $\Phi(\cdot)$  which maps the data into another (higher dimensional) space. By performing this mapping any complicated boundary (description of data) in low dimension can be modeled by a hyper-sphere in higher dimension. The kernel function in (2.37) automatically performs the mapping through its parameters – without the need for finding the mapping function  $\Phi(\cdot)$ . Therefore, we do not need to explicitly identify the mapping function  $\Phi(\cdot)$ . Several kernel functions have been proposed in the literature [97],[71] and [97], out of which the Gaussian kernel gives a closed data description:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \quad (2.38)$$

After applying the kernel function on the data points, the Lagrange optimization of (2.33) becomes:

$$L = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \quad \forall \alpha_i : 0 \leq \alpha_i \leq C \quad (2.39)$$

Figure 2.9 shows samples of a data-set drawn from a complex distribution (blue dots). If we use the simple Lagrange optimization equation (2.33) without the application of kernels on the samples the smallest circle encompassing the data (the red circle) will be achieved as the description of this data-set. However, by using the kernel function in (2.39) a more accurate boundary for the data is achieved (the green curve).

According to the above theory the proposed SVDDM method generates a sup-

port vector data description for each pixel in the scene using its history. These descriptions are then used to classify each pixel in new frames as a background or a novel/foreground pixel. In the following the actual implementation of the system is presented.

### 2.6.2 The algorithm

The methodology described in section 2.6.1 is used in our technique to build a descriptive boundary for each pixel in the background training frames to generate its model for the background. Then these boundaries are used to classify their corresponding pixels in new frames as background and novel (foreground) pixels. There are several advantages in using the Support Vector Data Description (SVDD) method in detecting foreground regions:

- Unlike existing statistical modeling techniques, the proposed method explicitly addresses the single-class classification problem. Existing statistical approaches try to estimate the probability of a pixel being background, and then use a threshold for the probability to classify it into background or foreground regions. The disadvantage of these approaches is in the fact that it is impossible to have an estimate of the foreground probabilities, since there are no foreground samples in the training frames.
- The proposed method has less memory requirements compared to non-parametric density estimation techniques, in which all the training samples for the background need to be stored in order to estimate the probability of each pixel in new frames. The proposed technique only requires a very small portion of the training samples, *support vectors*, to classify new pixels.
- The accuracy of our method is not limited to the accuracy of the estimated

```

1. Initialization;  $C$  , Trn_No,  $\sigma$ 
2. For each frame  $t$ 
  For each pixel  $x(i,j)$ 
    2.1. Training stage % OC(i,j) = 1- class classifier for pixel (i,j)
        SVD(i,j) ← Incrementally train( $x_t(i,j)$ ) % SVD: The Description
    2.2. Classification stage % Desc(i,j) = classification values
        Desc(i,j) ← Test( $x_t(i,j)$ , OC(i,j))
        Label pixel based on Desc(i,j).
    2.3. Update stage
        Re-train classifiers every 30 frames

```

Figure 2.10: The SVDDM algorithm.

probability density functions for each pixel. Also, since there is no need to assume any parametric form for the underlying probability density of pixels, this gives the proposed method superiority over the parametric density estimation techniques, i.e. mixture of Gaussians.

- The efficiency of our method can be explicitly measured in terms of false reject rates. The proposed method considers a goal for false positive rates, and generates the description of data by fixing the false positive tolerance of the system. This helps in building a robust and accurate background model.

Figure 2.10 shows the proposed algorithm in pseudo-code format<sup>1</sup>. The only critical parameter is the number of training frames (**Trn\_No**) that needs to be initialized. The support vector data description confidence parameter  $C$  is the target false reject rate of the system. This is not a critical parameter and accounts for the system tolerance. Finally the Gaussian kernel bandwidth,  $\sigma$  does not have a particular effect on the detection rate as long as it is not set to be less than one, since features used in our method are normalized pixel chrominance values. For all of our experiments

---

<sup>1</sup>The proposed method is implemented in MATLAB 6.5, using Data Description toolbox [92].

we set  $C = 0.1$  and  $\sigma = 5$ . The optimal value for these parameters can be estimated by a cross-validation stage.

### **Training stage**

In order to generate the background model for each pixel the SVDDM method uses a number of training frames. The background model in this technique is the description of the data samples (color and or intensity of pixels). The data description is generated in the training stage of the algorithm. In this stage, for each pixel a SVDD classifier is trained using the training frames, detecting support vectors and the values of Lagrange multipliers that maximize equation (2.33).

The support vectors and their corresponding Lagrange multipliers are stored as the classifier information for each pixel. This information is used for the classification step of the algorithm. The training stage can be performed off-line in cases where there are not global changes in the illumination or can be performed in parallel to the classification to achieve efficient foreground detection.

### **The Incremental SVDD Training Algorithm**

We published the proposed training algorithm for the SVDD in [85], and [87]. Our incremental training algorithm is based on the theorem proposed by Osuna *et al.* in [56]. According to this theorem a large QP problem can be broken down in to series of smaller sub-problems. The optimization on these sub-problems converges when new samples are added as long as at least one sample violates the KKT conditions. These conditions are shown in the following equations:

$$f(x_t) = \frac{(x_t \cdot x_t) - 2 \sum_i (x_t \cdot x_i) + \sum_{ij} (x_i \cdot x_j)}{R^2} = \begin{cases} < 1 & \alpha_t = 0 \\ = 1 & 0 < \alpha_t < C \\ > 1 & \alpha_t = C \end{cases} \quad (2.40)$$

In effect the KKT conditions, if satisfied, mean that if a sample's corresponding Lagrange multiplier is zero the sample is inside the boundary describing the data set. If the sample is on or outside the boundary its corresponding Lagrange multiplier should be non-zero. For the samples outside the boundary the Lagrange multipliers will be equal to the trade-off parameter ( $C$ ).

In the incremental learning scheme at each step we add one sample to the training working set. The training working set only consists of the support vectors. This is a direct result of the above theorem. Assume we have a working set which minimizes the current SVDD objective function for the current data set. If a new sample belongs to the description then it satisfies the KKT conditions. This means that its inclusion to the working set does not minimize the currently minimum objective function. Thus it will be discarded. If the KKT conditions do not hold for this sample the SVDD optimization is solved for the new working set which includes the new sample. Since the working set contains only support vectors of the data-set its size is considerably smaller than the actual data-set and the optimization can be performed efficiently.

From (2.30) it can be observed that Lagrange multipliers have a linear relationship. In order to further increase the optimization efficiency, we propose to solve the smallest possible subproblem [59] which consists of only two samples. Since only the new sample violates the KKT conditions, at every step, our incremental learning algorithm chooses one sample from the working set along with the new sample and solves the optimization on this two sample sub-set.

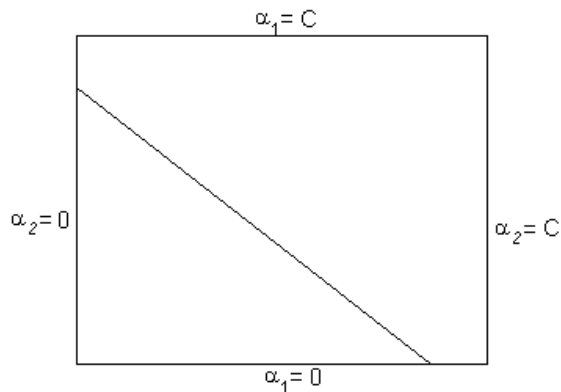


Figure 2.11: The two Lagrange multipliers should satisfy the inequality constrain (2.33) and the linear equality (2.41).

Solving the QP problem for two Lagrange multipliers can be done analytically. The two Lagrange multipliers should satisfy the inequality constrain in (2.33) and the following linear equality constrain (Figure 2.11):

$$\alpha_1 + \alpha_2 = \gamma \quad : \quad \gamma \leq 1 \quad (2.41)$$

The main component of our incremental learning algorithm is based on an analytical method to solve for the two Lagrange multipliers. We first compute the constrains on each of the two multipliers. From Figure 2.11 the two Lagrange multipliers should lie on a diagonal line (equality constrain) within the rectangular box (inequality constrain). By expressing the two ends of this line we can easily find bounds for one of the two multipliers and from there proceeds the optimization process. Without loss of generality we consider that the algorithm starts with finding the upper and lower bounds on  $\alpha_2$  which are  $H = \min(C, \alpha_1^{old} + \alpha_2^{old})$  and  $L = \max(0, \alpha_1^{old} + \alpha_2^{old})$ , respectively. The new value for  $\alpha_2^{new}$  is computed by finding the maximum along the direction of the linear equality constraint:

$$\alpha_2^{new} = \alpha_2^{old} + \frac{E_1 - E_2}{K(x_2, x_2) + K(x_1, x_1) - 2K(x_2, x_1)} \quad (2.42)$$

where  $E_i$  is the error in evaluation of each multiplier in equation (2.44). The denominator in (2.42) is a step size (second derivative of objective function along the linear equality constraint). Next, we determine whether the new value for  $\alpha_2^{new}$  has exceeded the bounds and needs to be clipped. We call this  $\hat{\alpha}_2^{new}$ . Lastly, the new value for  $\alpha_1$  is computed using the linear equality constrain:

$$\alpha_1^{new} = \alpha_1^{old} + \alpha_2^{old} - \hat{\alpha}_2^{new} \quad (2.43)$$

### Classification stage

In this stage for each frame, its pixels are used and evaluated by their corresponding classifier to label them as background or foreground. To test each pixel  $\mathbf{z}_t$ , the distance to the center of the description hyper-sphere is calculated:

$$\|\mathbf{z}_t - \mathbf{a}\|^2 = (\mathbf{z}_t \cdot \mathbf{z}_t) - 2 \sum_i \alpha_i (\mathbf{z}_t \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (2.44)$$

A pixel is classified as a background pixel if its distance to the center of the hyper-sphere is less than or equal to  $R^2$ :

$$\|\mathbf{z}_t - \mathbf{a}\|^2 \leq R^2 \quad (2.45)$$

$R^2$  is the distance from the center of the hyper-sphere to its boundary. This is also equal to the distance of each support vector to the center of the hyper-sphere and all the support vectors which fall outside the boundary are disregarded:

$$R^2 = (\mathbf{x}_k \cdot \mathbf{x}_k) - 2 \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_k) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (2.46)$$

Note that in the implementation of the algorithm, since the boundaries of the

data description are more complicated than a hyper-sphere, a kernel is used to map the training samples into a higher dimension. As the result the mapped samples in the higher dimension can be described by a high dimensional hyper-sphere and the above discussion can be used.

### 2.6.3 Performance evaluation

In this section the SVDDM performance in terms of memory requirements and computation cost is discussed. The key to evaluate the performance of this technique is to analyze the optimization problem solved by the system to find support vectors. The issues in detail are discussed as follows.

#### Parameters

As described earlier in this chapter, the support vector data description (SVDD) is an elegant technique to describe one class of data samples without any information about data belonging to other (novel) classes. In order to generate the data description, a hyper-sphere of minimum size which contains most of the training samples is constructed which represents the boundary of the known class. There are parameters involved with the construction of the class boundary (i.e. hyper-sphere), namely, the number of training samples  $N$  and the trade off factor  $C$  in equation (2.27) and the bandwidth of Gaussian kernel  $\sigma$  in (2.38). As mentioned in Section 2.6.2 for all of the experiments the values for  $C$  and  $\sigma$  are taken 0.10 and 5, respectively. This leaves the system with only the number of frames as a scene-dependent parameter.

#### Memory requirements

It is not easy to answer how many data samples are required to find a sufficiently accurate description of a target class boundary. It not only depends on the complexity

Table 2.5: Per-pixel memory requirements for the SVDDM method.

Memory Req.	Intensity	Chrominance	both	asymptotic
Bytes per pixel	$f(C, \sigma) \times 5 \geq 10$	$f(C, \sigma) \times 8 \geq 24$	$f(C, \sigma) \geq 32$	$O(1)$
No. of SVs	$f(C, \sigma) \geq 2$	$f(C, \sigma) \geq 3$	$f(C, \sigma) \geq 4$	$O(1)$

of the data itself but also on the distribution of the outlier (unknown) class. However, there is a trade-off between the number of support vectors to describe the data set and the driven description accuracy. In that sense, a lower limit can be found for the number of samples that can describe the data which corresponds to the rigid hyper-sphere containing most of the data samples with the target error goal.

In theory, only  $d + 1$  support vectors in  $d$  dimensions are sufficient to construct a hyper-sphere and their corresponding Lagrange multipliers ( $\alpha_i$ ) sum to 1. The center of the sphere lies within the convex hull of these support vectors.

- *Using only intensity values.*

Since by using intensity for each pixel there is only one feature value, the support vectors are 1-D and therefore the minimum number of support vectors required to describe the data will be 2. For each support vector 2 bytes are required to store the intensity and 8 bytes to store the Lagrange multipliers, resulting in at least 10 bits per pixel memory requirements.

- *Using chrominance values.*

By using red and green chrominance values,  $c_r$  and  $c_g$ , the minimum of 3 support vectors are needed to be used. This results in at least 24 bytes per pixel memory requirements.

The above reasoning provides a lower limit on the number of support vectors. Obviously, in practical applications this lower limit is far from being useful for implementation. The only fact that can be used from the above discussion is that the number of support vectors required to sufficiently describe a data set is related to

Table 2.6: Speed comparison of the incremental, online and canonical SVDD.

Training Set Size	Incremental[87] SVDD	Online[91] SVDD	Canonical[90] SVDD
100	0.66	0.73	1.00
200	1.19	1.31	8.57
500	2.19	2.51	149.03
1000	4.20	6.93	1697.2
2000	8.06	20.1	NA
$n$	$O(1)$	$\Omega(1)$	$O(n)$

the target accuracy of the description. Therefore, the memory requirement of the SVDDM method is independent of the number of training frames.

Table 2.5 shows memory requirements in bytes per pixel for the SVDDM method using intensity, chrominance values and their combinations, respectively. In conclusion, the asymptotic memory requirements of the SVDDM algorithm are constant  $O(1)$  since they are independent of the number of training frames.

### Computation cost

Training the SVDDM system for each pixel needs to maximize (2.33), which is a quadratic programming (QP) optimization problem. The most common technique to solve the above QP is the Platt's algorithm (sequential minimal optimization) [60], [58], which runs in polynomial time  $O(n^k)$ .

In order to show the performance of the proposed incremental training method and its efficiency we compare the results obtained by our technique with those of the online SVDD [91] and canonical SVDD [90]. We compare the speed of the algorithms as well as several error values for these techniques using different number of training samples and different data sets.

**The SVVD Training Speed.** In this section we compare the speed of incremental SVDD against its online and canonical counterparts. The experiments are conducted in Matlab 6.5 on a P4 Core Duo processor with 1GB RAM. The reported

Table 2.7: The number of support vectors retained.

Training Set Size	Incremental[87] No. of SV's	Online[91] No. of SV's	Canonical[90] No. of SV's
100	12	16	14
200	14	23	67
500	16	53	57
1000	19	104	106
2000	20	206	NA
$n$	$O(1)$	$O(n)$	$O(n)$

training times are in seconds. Table 2.6 Shows a report the training speed of our incremental SVDD, online and canonical versions on various sizes of data set. As seen, the proposed SVDD training technique runs faster than both canonical and online algorithms and its asymptotic speed is linear with the data set size. The online SVDD runs in linear time but for larger data sets its training time is more than the proposed method. Our observation showed that this is due to the fact that online SVDD retains more un-necessary support vectors than the proposed technique. As expected, both online and our SVDD training methods are considerably faster than the canonical training of the classifier. Notice that the training time of a canonical SVDD for 2000 training points is not available because of its slow speed.

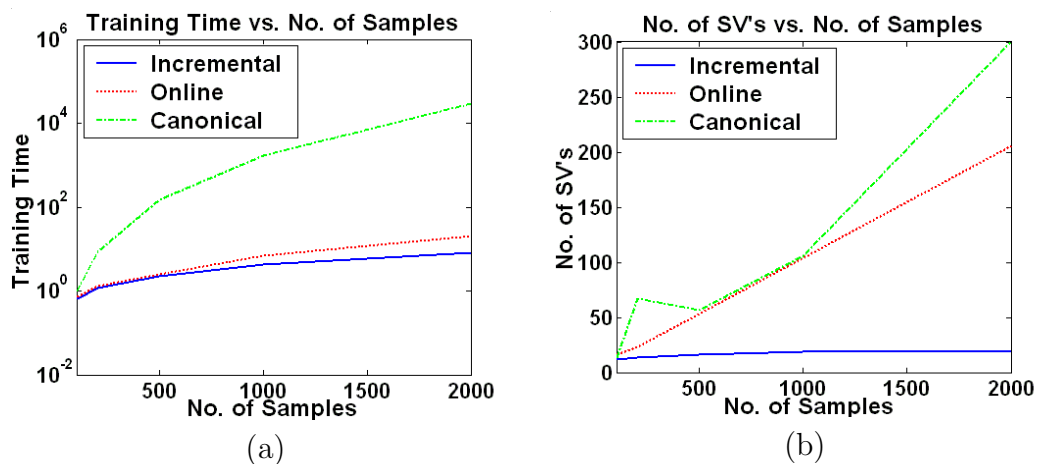


Figure 2.12: Comparison of the (a) training speed and (b) number of retained support vectors between the canonical learning ( $\cdot-$  curve), the online learning ( $- -$  curve), and the proposed method ( $-$  line).

**Number of Support Vectors.** A comparison of the number of retained support vectors for our technique, canonical, and online SVDD learning methods is presented in Table 2.7. In this experiment the parameters of the SVDD system are set as  $C = 0.1$  and  $\sigma = 5$  with a Gaussian kernel for all three classifiers. As can be observed, both the online and canonical SVDD training algorithm increase the number of support vectors as the size of the data set increases. However, our method keeps almost a constant number of support vectors. This can be interpreted as mapping to the same higher dimensional feature space for any given number of samples in the data set.

Notice that by increasing the number of training samples the proposed SVDD training algorithm requires less memory than both online and canonical algorithms. This makes the proposed algorithm very suitable for applications in which the number of training samples increase by time, i.e. in the case of growing data sets. Since the number of support vectors is inversely proportional to the classification speed of the system in (2.44), the classification time of a classifier trained by the proposed method is constant by the number of samples compared with the canonical and the online methods.

Figure 2.12 (a) and (b) compare the training speed and the number of retained support vectors, respectively, of the proposed incremental training algorithm as well as the online and canonical training methods.

## Chapter 3

# The Object Tracking Framework

In this chapter we address the second part of our visual tracking framework which deals with tracking of objects and generating their tracking trajectories. We discuss its application and explore its difficulties. Existing techniques that aim to solve this problem are briefly described and their limitations are investigated. Finally, the approach taken in this research to overcome limitations of the existing methods is discussed.

### 3.1 Goals and Motivation

Tracking of humans and other objects of interest within video frames is a very important task in many applications such as video surveillance [24], [40], perceptual user interfaces [5], smart environments [99], and driver assistance [26], just to name a few.

The tracking trajectories of the objects provide information about the objects interaction with the scene and each other. In general the problem of object tracking breaks down to a maintenance and search problem. Any reliable and robust object tracking mechanism includes two components. Different object tracking methods give different weights to these components to make the framework robust to a specific application. The first component is responsible for generating and maintaining a

model for the objects while the second process searches for potential new locations for these objects in the new frames.

The target model generation can be considered as a target representation and data association process which deals with the dynamics of the tracked objects, learning of the scene priors and the evaluation of multiple hypotheses. The search components of the visual tracking mechanism mostly deals with the target representation localization and changes in the target appearance [8].

In this chapter we investigate both components of the visual tracking from the literature and propose a robust, efficient, and reliable object tracking mechanism which addresses the shortcomings of the existing techniques.

## 3.2 Previous Work

As discussed earlier visual tracking mechanisms may employ two different approaches to predict the new location of objects of interest in new frames, based on either the targets appearance or their dynamics. However, since in real-world applications the computational resources are limited, it is desired that each of these components be designed as efficiently as possible.

Shalom in [1] presents the filtering and data association process through a state space approach. A discrete-time state sequence  $\{x_n\}$  represents the target characteristics. Two noise sequences  $\{v_n\}$  and  $\{q_n\}$  are considered to affect the target dynamics and the available measurement sequence  $\{z_n\}$ , respectively. The state sequence evolution is governed by the target dynamic equation  $x_n = \mathbf{f}_n(x_{n-1}, v_n)$  while the measurements are taken from the measurement equation  $z_n = \mathbf{h}_n(x_n, q_n)$ . In order to track the objects the goal is to estimate, at time  $t$ , the state  $x_t$  given all the measurements up to that time,  $z_{1:t}$ . This can be achieved by finding the posterior probability density function  $p(x_t|z_{1:t})$  (PDF).

The tracking given the setting from the above state space approach is performed by an iterative Bayesian filtering approach [8]. The prior PDF of the current frame  $p(x_t|z_{1:t-1})$  is evaluated from the posterior PDF of the previous frame  $p(x_{t-1}|z_{1:t-1})$  and the target dynamic equation. Then the posterior PDF of the current frame  $p(x_t|z_{1:t})$  is estimated by the estimated prior PDF and the likelihood pdf  $p(z_t|x_t)$  which is governed by the measurement equation.

Different assumptions can be made about the dynamic and measurement noise sequences  $\{v_t\}$  and  $\{q_t\}$  as well as their respective equations  $\mathbf{f}$  and  $\mathbf{h}$ , according to any specific tracking scenario. Kalman filter results in optimal solution if both noise sequences are Gaussian and the equations are linear [1]. This optimal solution provides that the posterior probability function to become a Gaussian distribution. However, if the  $\mathbf{f}$  and  $\mathbf{h}$  functions are non-linear the posterior can be obtained by linearization through the Extended Kalman Filters (EKF) [8], [1] resulting in Gaussian posterior distribution.

Although the Kalman filters and the EKFs result in reasonable tracking performance in simple and clutter free environments, they fail when applied to scenes with more clutter or when the background contains instances of the tracked objects. This is due to the fact that the tracking contains multi-modalities which can not be modeled by Gaussian distributions. In such cases a discrete state propagation process is considered without any assumptions made about the densities and the dynamic and measurement functions. Through Monte Carlo based integration methods the particle filters [42] and the bootstrap filters [23] were proposed. Also in discrete state cases the Hidden Markov Models are used for tracking purposes in [64].

In visual tracking applications, a Kalman filter has been employed to track vehicles [4] while in [67] Extended Kalman Filters are used to extract 3D object trajectories from 2D images. Isard and Blake in [30] and [32] introduced the conditional density

estimation algorithm based on particle filtering to track object contours in a cluttered environment. Chen *et al.* in [7] introduced the Hidden Markov Model filtering combined with the Joint Probabilistic Data Association Filters [1] for visual tracking mechanisms.

The aforementioned top-down mechanisms are statistical frameworks which are limited to the accuracy of the probabilistic modeling and estimation of priors of the states. The uni-modal representation techniques such as Kalman filtering do not result in robust tracking performances in cases where the optimal answer consists of complex distributions. These methods also in general do not provide reliable tracking results for non-rigid objects and deformable contours. Particle filters however have proven to be more robust in tracking non-rigid objects within complex and cluttered scenes. Unfortunately, these filters require probability density propagation through sequential importance sampling algorithm. This algorithm is a sequential Monte Carlo mechanism which is computationally intensive and does not satisfy the real-time requirements of many applications.

As mentioned earlier, the other component of object tracking algorithms is the target representation and localization. This bottom up approach generates and maintains the target models and searches in new frames for their potential locations. These methods mainly establish a likelihood function and try to maximize it [8]. To this end the target localization between two frames is very similar to the image registration problem. However, in target localization the amount of changes in the location and appearance of the target is assumed to be small. A real-time tracking algorithm was proposed by Hager and Belhumeur in [25] in which both geometric and illumination changes are modeled.

In [21], Ferrari *et al.* proposed an affine tracker which recovers an affine transformation between corresponding planar regions in image sequences. Recently, Tuzel *et*

*al.* in [96] proposed a new non-statistical method under the target representation and localization. In their approach the objects appearances are learned and Lie algebra is employed to detect and track them under significant pose changes. However, tracking non-rigid objects such as human bodies is a complicated problem which many image registration frameworks are not able to handle. Non-rigid, 3D motion analysis and tracking has been introduced in [27] and [98]. However, explicitly tracking of complex models for human bodies is very time consuming.

In order to efficiently track non-rigid objects Comaniciu *et al.* proposed a statistical kernel-based method to build probabilistic appearances of the target objects [8]. Their algorithm uses Bhattacharyya coefficient, a spatially smooth similarity function, to measure the similarity of the model and the target histograms. This measure is used tracking objects in the video by performing a mean-shift process. However, the kernel-based tracking mechanism has a poor scalability due to the mean-shift process and the need for maintaining target and model histograms.

Loza *et al.* in [48] present a structural similarity approach to object tracking in the video sequences. Their method uses a similarity measure which takes into account the luminance, contrast and the structural similarity of the two images. They used this measure within a particle filtering framework [31] to track objects of interest in video frames.

The bottom up mechanisms which perform target representation and modeling require the definition of several similarity measures and algorithms to perform target matching. These statistical methods are computationally intensive. In cases where several object should be tracked over a long period of time, these methods fail to perform efficiently. Since our system has a object detection stage, the results of the detection phase can be used to improve the object tracking efficiency.

Another issue in object tracking in video sequences is the ability to resolve oc-

clusion. In visual surveillance systems, this issue becomes more critical since people occlude each other when they are passing in any crowded situation. When this happens we consider it as a collision. Our approach is able to resolve this collision by building a reliable and robust model for each human detected in the scene. Once the collision happens the system only updates the person which is in front and thus the model for the occluded person remains intact. After the occlusion is over, both occluding and occluded persons are detected and tracked accordingly.

### 3.3 Contributions

The proposed algorithm is composed of two main stages:

- **Appearance correspondence mechanism.**

Once detected, photometric appearance based models are generated for the objects of interest. These models are considered to be the first degree estimation of the probability distribution of pixel colors. These models are employed in the spatio-spectral connected component tracking mechanism to solve a correspondence problem between the set of detected connected components and the set of target appearance models.

- **Occlusion resolution and update stage.**

An occlusion resolution stage is used to halt the update process in case two or multiple people occlude each other. For the purpose of tracking the occluding object, a mean-shift tracker is used. The tracking for the occluded object is suspended until the situation is resolved.

It should be noted that the proposed tracker solves a set of correspondence problems between target and models. Since our algorithm does not use any specific optimization method such as kernel based trackers [8], particle filters [30], Extended

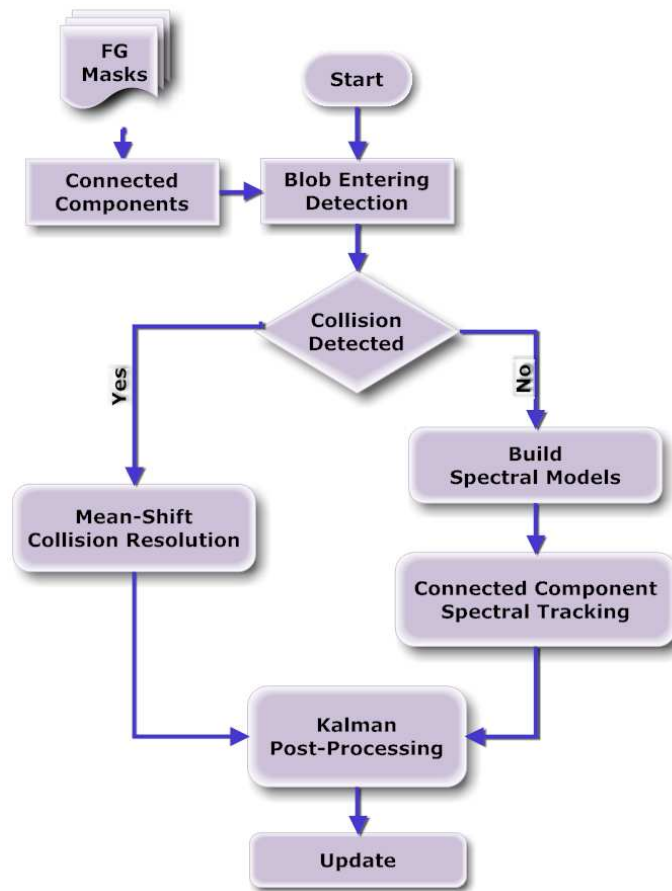


Figure 3.1: The overview of the visual tracker using a spatio-spectral tracking mechanism.

Kalman filters [1] and [4], etc., its speed and scalability is enhanced over the traditional methods.

### 3.4 Spatio-Spectral Connected Component Tracker

In this section we provide the details of our object tracking mechanism which facilitate the processing of tracking trajectories and information about each object. As the appearance of these objects are generally not known a priori, the only visual cue that can be used for detecting and tracking them is image motion. Although it is possible to perform segmentation from an image sequence that contains global motion, such

approaches – typically based on optical flow estimation [14]– are not very robust and are time consuming. Therefore our approach uses more efficient and reliable techniques, based on background modeling and segmentation, as discussed in the previous chapter.

Figure 3.1 shows an overview of our proposed visual tracking algorithm. As observed, the system uses the foreground masks detected during the object detection stage to generate connected components for each of the objects of interest. These connected components are then used to generate blobs for the objects. Each blob contains spatial and spectral information about its corresponding object. These information include the height and width of the object, its center of mass, and the first degree statistical estimation of its photometric appearance. The algorithm looks for the possibility of objects occluding each other. We call this event a collision. If no collision is occurring in the scene, the spectral data association process is performed to solve the blob correspondence problem and track individual objects.

If the algorithm detects the possibility of the collision, a multi-hypothesis data association is performed to find the occluded object(s). Since the visible photometric information for current frame does not represent the occluded object, its tracking process is suspended and its model will not be updated until the collision has been resolved. Since there will be only one occluding object, a kernel based tracking process will be used to track it. Notice that kernel based tracking for only one object is efficient and does not compromise the scalability issue discussed earlier. We retain the blob information for this object and update it for tracking purposes after the collision has been resolved.

In the final step of the algorithm, a simple Kalman filter is performed on the center of the blobs. Employing a Kalman filter in order to track individual points is an efficient process and does not complicate the computation requirements of the

algorithm. This step helps refine the objects' tracking trajectories and remove the jitters in the trajectories that might occur from the object detection stage on the foreground region centers.

### 3.4.1 The visual tracking algorithm

We propose an efficient Spatio-Spectral Tracking module (SST) to track objects of interest in the video sequences. The major assumption is that the camera is static. However, we do not make any further restrictions on the background composition, thus allowing for local changes in the background such as; fluctuating lights, water fountains, waving tree branches, etc.

The proposed approach uses models of the background pixel changes using the object detection module discussed in the previous section. The foreground regions are processed further by employing a connected component processing in conjunction with a blob detection module to find objects of interest. These objects are tracked by their corresponding statistical models which are built from the objects' spectral (color) information. A laser-based range finder is used to extract the objects' trajectories and relative angles from their 2-D tracking trajectories and their depth in the scene. However, the spatio-spectral coherency of tracked objects may be violated in cases when two or more objects occlude each other. This situation is called a collision in the literature.

A collision resolution mechanism is devised to address the issue of occlusion of objects of interest. This mechanism uses the spatial object properties such as their size, the relative location of their center of mass, and their relative orientations to predict the collision.

### **Blob detection and object localization**

In the blob detection module, the system uses a spatial connected component processing to label foreground regions from the previous stage. However, to label objects of interest a blob refinement framework is used to compensate for inaccuracies in physical appearance of the detected blobs due to unintended region split and merge, inaccurate foreground detection, and small foreground regions. A list of objects of interest corresponding to each detected blob is created and maintained to further process and track each object individually. This raw list of blobs corresponding to objects of interest is called the spatial connected component list.

Spatial properties about each blob such as its center and size are kept in the spatial connected component list. The list does not incorporate individual objects' appearances and thus is not solely useful for tracking purposes. The process of tracking individual objects based on their appearance in conjunction with their corresponding spatial features is carried out in the spatio-spectral tracking mechanism.

### **Spatio-spectral tracking algorithm**

A technique which tracks moving objects (such as humans) requires a model for individual objects. In our approach, these "appearance model" are employed to search for correspondences among the pool of objects detected in new frames. Once the target for each individual has been found in the new frame they are assigned a unique ID. In the update stage the new location, geometric and photometric information for each visible individual are updated. This helps recognize the objects and recover their new location in future frames.

Listing in Figure 3.2 show the pseudo-code of the proposed object tracking algorithm. Our proposed appearance modeling module represents an object with a set of statistical representation for the appearance of the object. In the spatio-spectral

```

Maintain the list of tracking objects: O-Lt-1[1 : n]
For new frame t containing the foreground masks
  1. Detect the connected components
  2. Perform morphological smoothing to detect contingent objects
  3. Detect collision
     if no-collision:
  4. Maintain the new object list: CC-Lt[1 : k]
  5. if k > n determine if new objects are to be added to the new list
     if new objects then create = 1
     for i = 1 : k
       5.1. Generate the following:
           CC-L[i].Center
           CC-L[i].width
           CC-L[i].height
           CC-L[i].appearance
     for all unassigned O-Lt-1 list objects
       5.2. find object O-Lt-1[j] : argmax [mean (p(CC-Lt[i]|O-Lt-1))]
       5.3. if probability is larger than threshold
           Assign: O-Lt[j] ← CC-Lt[i]
           Make object O-Lt[j] visible
         else
           Make object O-Lt[j] invisible
       5.4. if ( create = 1 )
           Assign: O-Lt[n + 1] ← CC-Lt[k]
           Make object O-Lt[n + 1] visible
  6. if collision:
     Maintain colliding object list: CO-Lt[1 : k]
     for colliding objects:
       6.1. find CO-Lt[j] : argmax [mean (p(CO-Lt[i]|O-Lt-1))]
       6.2. find maximum probability among colliding list objects
           suspend update for all the other objects in colliding list
           perform mean-shift tracking on the occluding object
  7. perform Kalman filter on the centers of visible objects

```

Figure 3.2: The spatio-spectral object tracking algorithm.

tracking module a list of known objects of interest is maintained. This list represents each individual object and its corresponding spatial and color information along with its unique ID. During the tracking process the system uses the raw spatial connected component list as the list of observed objects and uses a statistical correspondence matching to maintain the ordered objects list and track each object individually. The

tracking module is composed of three components, appearance modeling, correspondence matching, and model update.

### ***Appearance modeling***

Once each connected component is detected and processed according to algorithm shown in Figure 3.2 their appearance models are generated. These appearance models along with the objects location and first order geometric approximation produce an extended blob structure for the detected objects.

In order to produce the geometric appearance of the detected objects, we use their corresponding connected components and geometric moments analysis. According to Hu [29] the 2-D geometric moments of a region  $R$  are:

$$m_{pq} = \iint_R x^p y^q f(x, y) dx dy \quad (3.1)$$

where  $f(x, y)$  is the function over the region  $R$  (i.e. intensity) and  $p + q$  is the order of the moment. Some moments such as  $m_{00}$ ,  $m_{10}$ , and  $m_{01}$  are particularly important since they encode relevant visual and simple geometric features of a shape. In order to use these moments in computing geometric features of the objects in this research, we use the connected components of these objects. Therefore the function  $f(x, y) = 1$  for the region occupied by the object. Since the moments for the connected components are computed on discrete images, the integrals in equation (3.1) become summations. Thus, the 2-D geometric moments for our objects become:

$$m_{pq} = \sum_R x^p y^q \quad (3.2)$$

where  $R$  is the connected component region of each object. According to equation (3.2), the zeroth order geometric moment,  $m_{00}$ , becomes the object area and the first

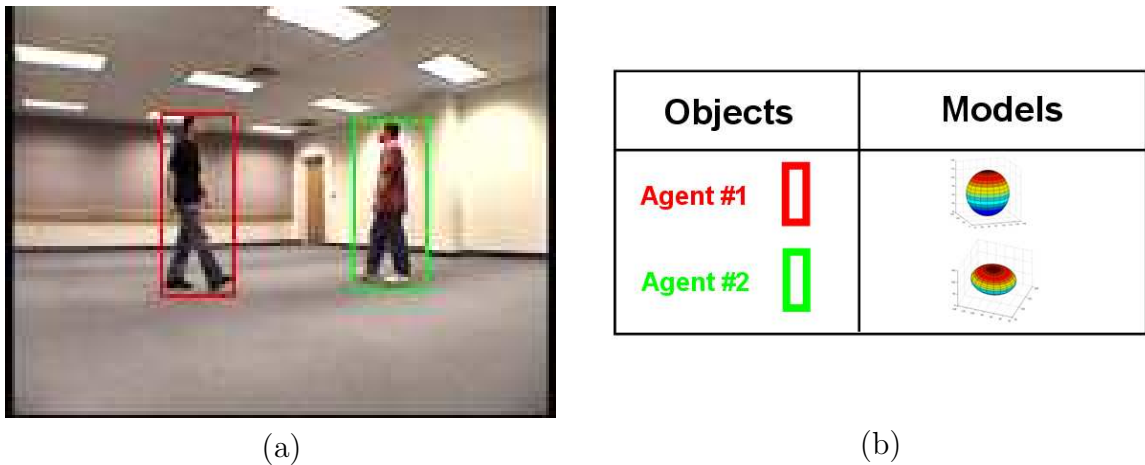


Figure 3.3: Photometric appearance models for objects: (a) an arbitrary frame of the video with objects being tracked. (b) the visualization of the object appearance models in the RGB feature space.

order geometric moments  $m_{10}$ , and  $m_{01}$  are the spread of the object along  $x$  and  $y$  axes, respectively. From these moments the center of mass of the object can be defined as:

$$\begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} \frac{m_{10}}{m_{00}} \\ \frac{m_{01}}{m_{00}} \end{bmatrix} \quad (3.3)$$

Along with these geometric features for the objects we extract orientation, major and minor axis lengths which represent the height and width of the objects according to its orientation. The objects' centers and their width are used in the process of collision detection which is discussed later.

The other component of the models of the objects in our algorithm is their photometric representation. Our current photometric appearance models are the first order statistical estimation of the probability density functions of pixel colors within the object. The first order representation is modeled as a normal distribution function of the pixels belonging to the object:

$$\forall i : O_i^t = \mathcal{N}(\mu_i, \sigma_i) \quad (3.4)$$

where  $O_i^t$  is  $i$ th object model at time  $t$  and  $\mu_i$  and  $\sigma_i$  are the estimated mean and covariance matrix of the RGB color values of object pixels, respectively. Figure 3.3 shows the visualization of this first degree statistical estimation of the object models in the RGB color space.

### *Correspondence matching*

After the models are generated and objects are tracked in the previous frame at time  $t - 1$ , a correspondence matching mechanism is employed in the current frame to keep track of the objects at time  $t$ . This technique is the main component of our spatio-spectral object tracking framework. Unlike many target representation and localization methods such as kernel based object tracking [8] and particle filters based approaches [48], our mechanism takes advantage of the object detection stage of the algorithm. The traditional approaches usually ignore the foreground objects and search in a neighborhood of the object in the previous frame to find the local probability maxima for the presence of the object in that neighborhood.

Foreground objects generated using the connected component process from the foreground image which is produced from the background segmentation process populate a finite list of un-assigned objects in the current frame. We call this list  $\{\text{CC-L}\}_t$  and the list of object appearance models from the previous frame  $\{\text{O-L}\}_{t-1}$ .

The idea in our correspondence matching algorithm is to search on the un-assigned list of objects in the current frame for their corresponding blob (appearance model) from the previous frame. Notice that in our algorithm instead of a spatial search over a window around each object and finding the best target match, we perform the search over the object list in the new frame. This decreases the computational cost of the algorithm compared to the traditional methods. Let's denote the  $i$ th object from the connected component list in the current frame as;  $\text{CC-L}_t(i)$ . The algorithm

goes through the object models from the  $\{\text{O-L}\}_{t-1}$  list and finds the model which maximizes the likelihood of representing the  $\text{CC-L}_t(i)$ . If such model exists and is denoted by  $\text{O-L}_{t-1}(j)$  then:

$$\text{O-L}_{t-1}(j) = \arg \max_k [\text{mean}(P(\text{C-L}_t(i)|\text{O-L}_{t-1}(k)))] \quad : \forall k \quad (3.5)$$

Once such correspondence is found between objects from the two lists, the object in the new frame is assigned the appropriate ID, and its appearance model and tracking trajectory is updated. If all the object models are assign and there are still unassigned connected components, new models and blob are created. The appearances of each new objects are then updated using the geometric and photometric modeling presented earlier.

### Collision resolution

In order for the system to be robust to collisions – when individuals get too close to each other so that they occlude each other – the models for the occluded individual may not reliable for tracking purposes. Our method uses the distance of detected objects as a means of detecting a collision. After a collision is detected we match each of the individual models with their corresponding representatives. The one with the smallest matching score is considered to be occluded. The occluded object’s model will not be updated but its new position is predicted by a Kalman filter. The position of the occluding agent is updated and tracked by a mean-shift algorithm. After the collision is over the spatio-spectral tracker resumes its normal process for these objects.

In this case we match each of the individual models with their corresponding representatives. The one with the largest matching score is considered to occlude other objects with which its mask is colliding. The update process for all occluded objects

```

For new frame t
1. Calculate the speed of the objects
2. for each object pair
   2.1. predict the new object centers in the
       next frame using Kalman filter
   2.2. if ( the two objects overlap ) then Collision = 1
   2.3. else Collision = 0
3. return Collision

```

Figure 3.4: The collision resolution algorithm.

is suspended and their appearances are not updated accordingly. This accounts for the situations in which multiple people are in the scene and it is likely that collision would occur. This had been a major drawback for some of the well known appearance based tracking systems such as the mean-shift algorithm.

Figure 3.4 shows the algorithm which performs the collision detection and resolution in the proposed object tracking technique. The collision detection algorithm assumes that the center of the object and their velocity within consecutive frames are linearly related and the observation and measurement noises are normal with zero mean. Therefore, a Kalman filter can be used to predict the new locations of object centers. These information along with the width and height of the objects are used to predict the possibility of collision between multiple objects.

In our approach we assume that the discrete state of the objects is represented by their center of mass and its velocity. Therefore, we have:

$$\mathbf{x}(k) = [C_x(k), \dot{C}_x(k), C_y(k), \dot{C}_y(k)]^T \quad (3.6)$$

where  $[C_x(k), C_y(k)]$  is the center of the object at frame  $k$  and:

$$\begin{aligned} \dot{C}_x(k) &= C_x(k) - C_x(k-1) \\ \dot{C}_y(k) &= C_y(k) - C_y(k-1) \end{aligned} \quad (3.7)$$

Since the measurements are taken every frame at discrete time-steps with the rate of 30 frames per second, it is important to be able to predict whether the objects will collide given the observation and measurement parameters in the current frame. We assume that the object centers undergo a constant acceleration from frame to frame with unknown rates. We also assume that between each time-step the acceleration is randomly distributed with a normal probability density function with zero mean and an unknown covariance matrix. The governing equation that rules the relationship of consecutive states is given by:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}a_k \quad (3.8)$$

where  $a_k$  is the constant acceleration from time  $k$  to  $k+1$ ,  $\mathbf{G} = [1/2 \ 1]^T$  is the acceleration vector and  $\mathbf{F}$  is the the velocity matrix:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.9)$$

Since we assume that the acceleration is drawn from random white zero mean noise the state equation will become a linear function  $\mathbf{h}$  affected by noise  $\mathbf{n}$ . Also we assume that the measurements are subject to a normal noise  $\mathbf{v}$  which is independent from the observation noise. Therefore:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{h}(\mathbf{x}(k)) + \mathbf{n} \\ \mathbf{z}(k) &= \mathbf{f}(\mathbf{x}(k)) + \mathbf{v} \end{aligned} \quad (3.10)$$

where  $n = \mathcal{N}(\mathbf{0}, \mathbf{Q})$  and  $v = \mathcal{N}(\mathbf{0}, \mathbf{R})$ . Since the state and measurement equations

are linear and the noise is Gaussian Kalman filter can be used to predict the location of the object centers in the new frames [6]. After finding the new center for objects,

$\mathbf{C}_t^{new} = [C_x^{new} \quad C_y^{new}]^T$ , the following steps are performed to detect collision:

- find the list of object which are close
- for each pair of the objects from the above list determine whether the collision is about to occur – if any of the following is true:

$$\left\{ \begin{array}{l} C_1^{new} \cdot x < C_2^{new} \cdot x \Rightarrow C_1^{new} \cdot x + O_1 \cdot \frac{w_1}{2} \geq C_2^{new} \cdot x - O_2 \cdot \frac{w_2}{2} \\ C_1^{new} \cdot y < C_2^{new} \cdot y \Rightarrow C_1^{new} \cdot y + O_1 \cdot \frac{h_1}{2} \geq C_2^{new} \cdot y - O_2 \cdot \frac{h_2}{2} \\ C_1^{new} \cdot x > C_2^{new} \cdot x \Rightarrow C_1^{new} \cdot x - O_1 \cdot \frac{w_1}{2} \leq C_2^{new} \cdot x + O_2 \cdot \frac{w_2}{2} \\ C_1^{new} \cdot y > C_2^{new} \cdot y \Rightarrow C_1^{new} \cdot y - O_1 \cdot \frac{h_1}{2} \leq C_2^{new} \cdot y + O_2 \cdot \frac{h_2}{2} \end{array} \right. \quad (3.11)$$

where  $C_1$  and  $C_2$  are the center coordinates of each object pair and  $h$  and  $w$  is their respective height and width.

### 3.4.2 Performance evaluation

In this section we evaluate the performance of the proposed visual object tracking framework. We explore the scalability of the proposed method with the techniques in the literature. As mentioned in the previous sections our object tracking algorithm is a spatio-spectral data association approach. The photometric and geometric appearances of the objects are maintained through the video. The tracking algorithm associates pairs of objects whose appearances are generated from the previous frames and the objects detected in the new frame.

Our approach is different from target localization methods such as the kernel-based object tracking in that the search to find the suitable target which represents the object model is not done spatially in the frame. The proposed algorithm uses

the results of object detection process to find the best target match from the list of detected objects. Notice that this approach reduces the two dimensional search space to one dimension of detected object lists. As a result the object tracking speed is increased compared to techniques which require spatial search. Another advantage of the proposed method is in its memory requirements compared to kernel-based techniques. Our algorithm maintains a first order statistical approximation of the photometric object appearance instead of a kernel based histogram.

In order to quantitatively compare the computational cost of our method with kernel-based tracking mechanisms we first analyze our algorithm. By looking at our tracking algorithm it is obvious that the asymptotic cost of our algorithm is  $\Omega(n)$  for tracking  $n$  objects at the same time. By closely examining the two loops in the algorithm the asymptotic cost of our method is considered  $O(n^2)$ .

Notice that since neither our algorithm nor the kernel-based method are designed to track infinite number of objects at the same time we are not interested in asymptotic costs. The inner loop of our algorithm searches for those objects in the previous modeled object list which are not already associated with those in the connected component list. Therefore, the computational cost of our algorithm can be calculated using arithmetic series:

$$\text{Cost}_{SSC} = \sum_{i=n}^1 (i) = \frac{n(n-1)}{2} \quad (3.12)$$

In the kernel based tracking algorithms the matching target is found by searching for the closest histogram to the model in a window around the object using the mean-shift algorithm. Although the generation of the histograms and the mean-shift algorithm (through the Bhattacharyya coefficient) are more computationally expensive than our probabilistic comparison, we assume their costs are similar for the sake of our comparison. If the kernel-based algorithm looks for a  $W \times H$  window around the object, then for each object the number of histograms to be evaluated by

mean-shift algorithm is equal to  $W \times H$ . This means that for tracking  $n$  objects at the same time the computational cost of the kernel-based approach is:

$$\text{Cost}_{MS} = (W \times H)n \quad (3.13)$$

If we want the two approaches to have the same computational complexity we should have:

$$\begin{aligned} \text{Cost}_{SSC} = \text{Cost}_{MS} &\Rightarrow \frac{n(n-1)}{2} = (W \times H)n \\ &\Rightarrow (n-1) = 2(W \times H) \\ &\Rightarrow n = 2(W \times H) + 1 \end{aligned} \quad (3.14)$$

From (3.14) we conclude that the proposed algorithm runs faster than the kernel-based approaches for typical applications. For example, if the search window size in the kernel-based algorithm is only  $10 \times 10$ , our algorithm runs 100 times faster tracking one object. In other words, the proposed tracking algorithm is capable of tracking 201 objects at the same time, running with the same speed as its kernel-based counterpart.

## Chapter 4

# Comparison and Evaluation

In this chapter the performance of our visual object tracking framework on a number of challenging videos is discussed and its results are compared with those of the methods in the literature. As mentioned in the previous sections, the visual object tracking is composed of two components, namely object detection and object tracking. In order to have a comprehensive evaluation and comparison, we compare each of the components within our algorithm with the respective state-of-the-art counterparts.

This chapter first evaluates the object detection mechanism developed in this dissertation and compares it with various existing object detection algorithms. A number of particularly challenging scenarios are presented to the algorithms and their ability to handle issues are evaluated. The comparisons of methods are performed both qualitatively and quantitatively on these scenarios.

Finally, the chapter evaluates the performance of the proposed object tracking algorithm and discusses its final frame rate, its strengths over the existing visual trackers, and its ability to handle collision. The proposed method is compared with the existing techniques in as well as outside its category within the literature. We also combined a few methods from the literature to assess the performance of the proposed approach with a hybrid technique based on the existing algorithms.

## 4.1 Object Detection

This section compares the performance of proposed techniques using several real video sequences that pose significant challenges. Also their performances are compared with the mixture of Gaussians method [75], the spatio-temporal modeling presented in [46] and the simple KDE method [18]. We use different scenarios to test the performance of the proposed techniques and to discuss where each method appears to be particularly suitable.

By comparing Table 2.1, Table 2.3 and Table 2.5 we notice that the memory requirements of the AKDE technique are lower than those of the RM method if the number of training frames is less than 250. That is, for situations where about 250 frames can cover most of the changes that occur in the background, by using a sliding window of size 250 the AKDE method needs less memory than the RM technique. Note that SVDDM needs even less memory than the AKDE and like the RM its memory requirements are independent of the number of training samples. However if the changes in the background are very slow and not periodic, such as the water surface or slowly waving flags, 250 frames are not enough to accurately model the background. This results in erroneously detecting some parts of the changing background as foreground. In such situations the RM technique is superior because its memory requirements are independent of the number of frames. When the number of training frames increases the SVDDM becomes less efficient because of its computational cost, not of its memory requirements.

Similarly, by comparing Table 2.2, Table 2.4 and section 2.6.3 we notice that if the number of training frames is more than a certain value the computational cost of the SVDDM becomes more than that of the AKDE and the RM methods. This shows that in order for the AKDE and SVDDM methods to reach the same frame-per-second rate as the RM, a sliding window of reasonably small size should be used as



Figure 4.1: Rapidly fluctuating background: (a) *Handshake* video sequence. Detected foreground regions using (b) AKDE. (c) RM. (d) SVDDM.

the background training buffer. Also it should be noted that the SVDDM technique uses a sequential minimal optimization method for solving Lagrange multipliers. This optimization technique is slow and computationally extensive. Although the SVDDM technique is quite accurate and requires less memory than the AKDE method, it takes more processing power to train and to retain its accurate model for background pixels.

#### 4.1.1 Rapidly fluctuating backgrounds

As described above, for videos with rapidly changing backgrounds, the AKDE method has a better performance in terms of memory requirements and speed. Our experiments showed that for videos where possible fluctuations in the background occur in about 10 seconds, the AKDE technique needs less memory and works faster compared to the RM and SVDDM methods.

Figure 4.1 shows the detection results of the AKDE, RM and the SVDDM algorithms on the *Handshake* video sequence. As it can be seen from this figure, capturing dependencies between chrominance features results in more accurate foreground regions (Figure 4.1(b)). This shows that AKDE performs better than both the RM and the SVDDM particularly in this case. Note that in this particular frame the color of foreground objects is very close to the background in some regions. The SVDDM technique results in very smooth and reliable foreground regions because this technique uses the confidence factor to eliminate false positives in the classification. This



Figure 4.2: Low contrast videos: (a) *Handshake* video sequence. Detected foreground regions using (b) AKDE. (c) RM. (d) SVDDM.

may lead to missing some parts of the foreground which are very close in color to the background. Also notice that in all methods the fluctuations seen on monitors are completely modeled as a part of background and not detected as foreground regions.

#### 4.1.2 Low contrast videos

To clarify the accuracy of the SVDDM technique in low contrast video sequences and to compare its results with those of the AKDE technique, the experiment is performed on the *Handshake* video sequence. Figure 4.2 shows a frame where the background and foreground colors are reasonably different. In this experiment we intentionally decreased the quality of the images in video sequence by down-sampling them by a factor of 4. The accuracy of the foreground region detected using the SVDDM technique is clearly better than that of the AKDE method. The reason is that the SVDDM method fixes the false reject rate of the classifier and generates a discrimination boundary independent of the estimated probability density function of the single known class, i.e. the background class. On the other hand, the AKDE method uses a complicated training scheme to make a statistical binary classifier into a single-class classifier introducing a statistical Bayes error to the model.

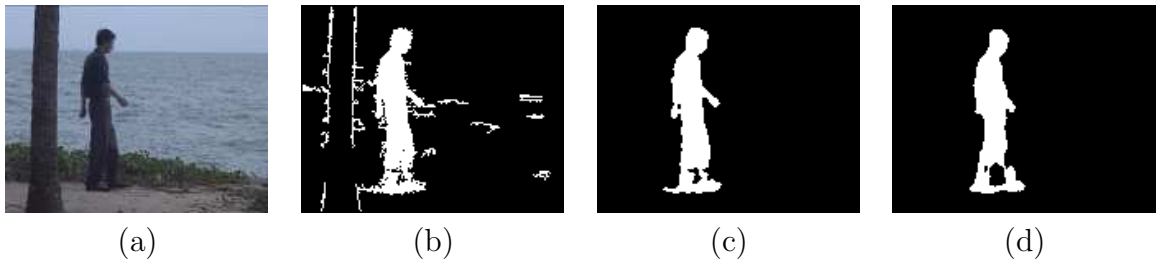


Figure 4.3: Slowly changing background: (a) *Water* video sequence. Detected foreground region using (b) AKDE. (c) RM. (d) SVDDM.

### 4.1.3 Slowly changing backgrounds

For videos with slowly changing backgrounds or backgrounds whose changes are not periodic, the AKDE method needs more training frames to generate a good model for the background. This increases the system memory requirements and drastically decreases its speed. In these situations the RM technique is a very good alternative, since its performance is independent of the number of training frames.

Figure 4.3(a) shows an arbitrary frame of the *Water* video sequence. In this figure the detection results of the AKDE, the RM and the SVDDM methods are presented. This example is particularly difficult because waves do not follow a regular motion pattern and their motion is slow. As can be seen from Figure 4.3(b), using the AKDE method without any post-processing results in many false positives. Figure 4.3(c) shows the detection results of the RM method while Figure 4.3 (d) shows the results of the SVDDM.

From this figure we can conclude that the RM method has a better performance compared to both the AKDE and the SVDDM in situations in which the background has slow and irregular motion. The reason is that in the AKDE there is a sliding window of limited size which may not cover all changes in the background, while the model is continuously updated in the RM method therefore keeping most of the changes that occurred in the past. However the SVDDM method performs better than the AKDE technique in this scenario because the model built in the SVDDM uses

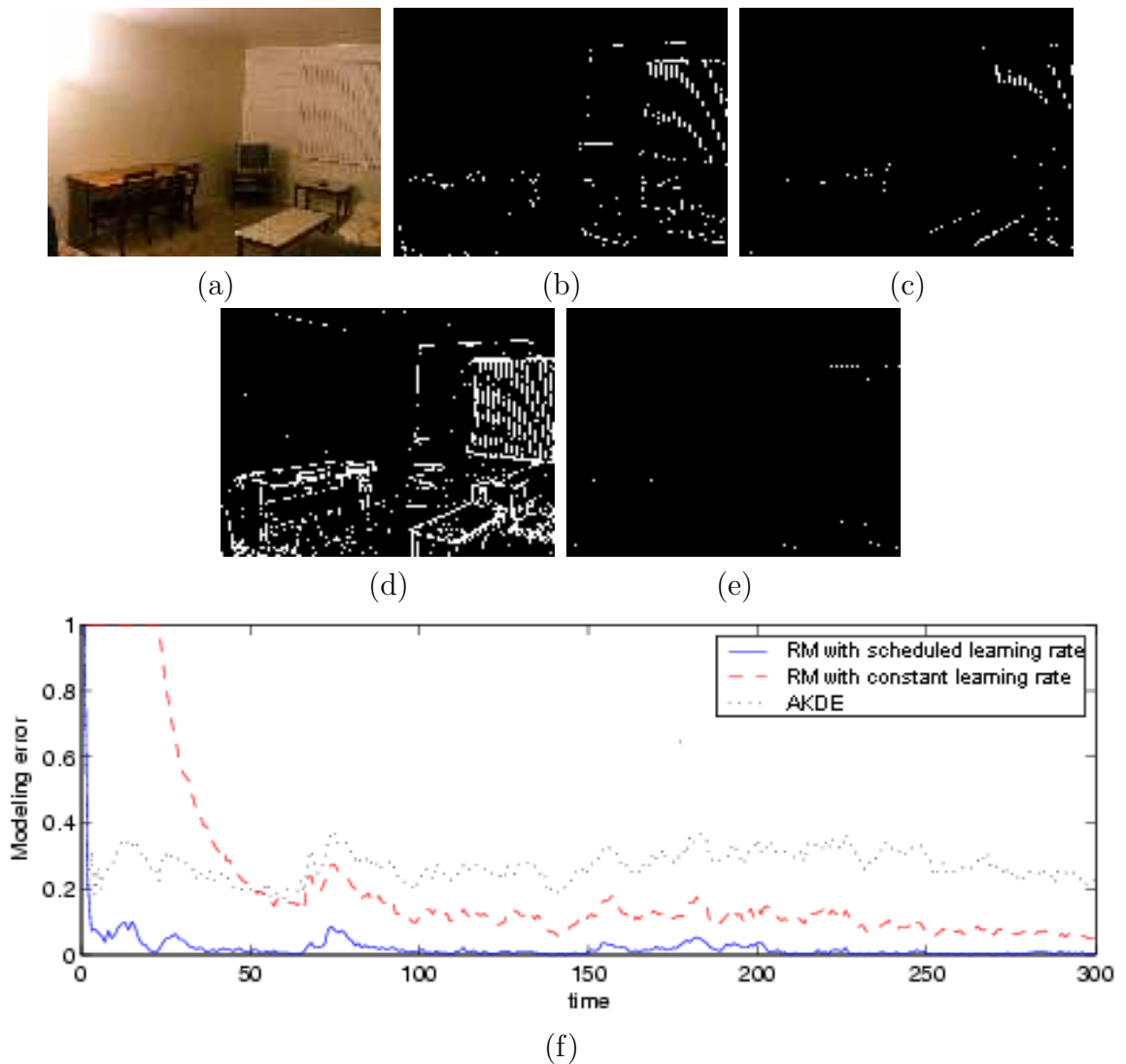


Figure 4.4: Hand-held camera: (a) *Room* video sequence. (b) Modeling error in a hand-held camera situation using different methods. (c) False positives after 2 frames using the AKDE method. (d) False positives after 247 frames using the AKDE method. (e) False positives after 2 frames using the RM method. (f) False positives after 247 frames using the RM method.

the decision boundaries of the single training class instead of bounding the training accuracy to the accuracy of the probability estimation.

#### 4.1.4 Hand-held camera

In situations when the camera is not completely stationary, such as the case of a hand-held camera, the AKDE and the current batch implementation of the SVDDM methods are not suitable. In these situations there is a consistent, slow and irregular global motion in the scene, which can not be modeled by a limited size sliding window of training frames. In such cases the RM method is particularly preferable and appropriate.

Figure 4.4 shows the modeling error of the RM method in the *Room* video sequence. In Figure 4.4(a) an arbitrary frame of this video is shown. Figure 4.4(b) compares the modeling error using different techniques. As can be seen, the modeling error using a constant window size in the AKDE (the dotted line) is between 20%-40%, and it does not decrease with time. This shows that the system using the AKDE method with a constant sized sliding window never converges to the actual model. The dashed line shows the modeling error using the RM method with a constant learning rate, and the solid line shows the modeling error of the RM with scheduled learning. We conclude that the model generated by the RM technique eventually converges to the actual background model and its error goes to zero. Figure 4.4(c) and (d) show misclassified regions using the AKDE method after 2 and 247 frames respectively and Figure 4.4(e) and (f) show the false positives using the RM method after 2 and 247 frames into the video. As it can be seen, the amount of false positives decreases with time as the system accumulates most changes observed in the history of the scene using the RM method, whereas for the AKDE it does not converge to zero.

Results of the SVDDM technique on this video sequence are very similar to those of the AKDE and therefore are not shown in Figure 4.4 since there are very few visual differences in the results. This is expected since both techniques use a finite number of

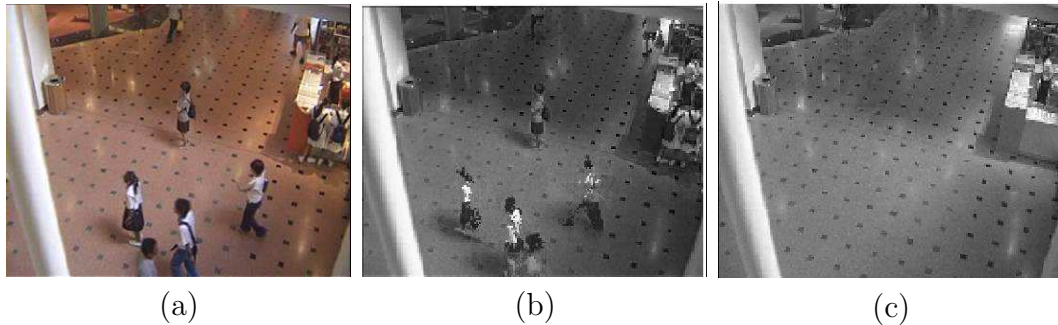


Figure 4.5: Non-empty background: (a) *Mall* video sequence. (b) Background model after 5 frames using the RM method. (c) Background model after 95 frames using the RM method.

frames to train the background model. Also note that there is no foreground object present in the scene in order for us to quantitatively measure the accuracy of the background modeling.

#### 4.1.5 Non-empty backgrounds

In situations in which the background of the video is not empty (that is, there is no clear background at any time in the video sequence), the AKDE and SVDDM methods fail to accurately detect the foreground regions. In these situations the RM technique has to be used to generate an accurate empty background model.

Figure 4.5 shows the background model in the *Mall* video sequence in which the background is almost never empty. In this situation the AKDE and the SVDDM methods fail unless a post-processing on the detected foreground regions is performed to generate models for uncovered parts of the background. The reason is that the system considers the foreground objects present in the background training window as a part of background and when those objects move, their empty position is detected as a foreground region. In the RM method however, the background model is being updated every frame and from the beginning of the video. When an object moves, the new pixel information is used to update the background model and converges

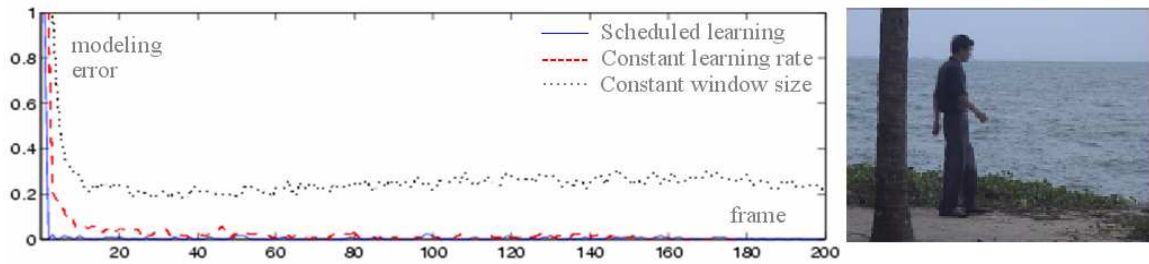


Figure 4.6: Convergence speed.

to the new one. Figure 4.5(b) shows the background model after 5 frames from the beginning of the video and Figure 4.5(c) shows the model after 95 frames into the scene. As it can be seen the model converges to the empty background since moving people do not add enough evidence to the model and the background itself contributes more to consistently it.

#### 4.1.6 Convergence speed

An important issue in the recursive learning is the convergence speed of the system (how fast the model converges to the actual background). Figure 4.6 illustrates the convergence speed of our approach with scheduled learning rate, compared to constant learning and kernel density estimation with constant window size. In this figure the modeling error of the RM with scheduled learning and constant learning rate as well as the AKDE modeling error are plotted against frame number. An arbitrary frame of the video which was processed by the methods is shown as well.

As seen from Figure 4.6, the AKDE modeling error (the black  $(-\cdot)$  curve) drops to about 20% after about 20 frames – the training window size. The modeling error for this technique does not fall below a certain value since the constant window size does not cover all of the slow changes on in the background. In contrast, the error for an RM approach decreases as more frames are processed. This is due to the recursive nature of this algorithm and the fact that every frame contributes to the generation

and update of the background model.

The effect of the scheduled learning proposed in section 2.5 can be observed by comparing the modeling error of the RM algorithm with a constant learning rate (the red dashed curve) and with the scheduled learning rate (the solid blue curve). Although the error decreases in both cases, the scheduled learning rate makes the decrease in the error rate much faster. This fact shows that the scheduled learning rate help speed up the convergence speed of the algorithm.

#### 4.1.7 Sudden global changes

In situations where the video background suddenly changes, such as lights on/off, the proposed RM technique with scheduled learning recovers faster than the AKDE method. Generally, with the same speed and memory requirements, the RM method results in faster convergence and lower model error.

Figure 4.7 shows the comparison of the recovery speed from an expired background model to the new one. This happens in the *Lobby* video sequence when the lights go off (Figure 4.7(a)) or they go on (Figure 4.7(b)), or when a new object is permanently added to or removed from the background. In our example, lights go from on to off through three global, sudden illumination changes at frames 23, 31 and 47 (Figure 4.7(c)). Figure 4.7(c) shows that the scheduled learning RM method (solid curve) recovers the background model after these changes faster than non-scheduled RM and AKDE with constant window size. The constant, large learning rate recovers much slower (dashed curve) and the AKDE technique (dotted curve) is not able to recover even after 150 frames. A similar situation with lights going from off to on through three global, sudden illumination changes is shown in Figure 4.7(d). Note that the mixture learning algorithms are even slower in convergence and recovery. A typical mixture learning technique [75] needs at least 1000 frames to converge.

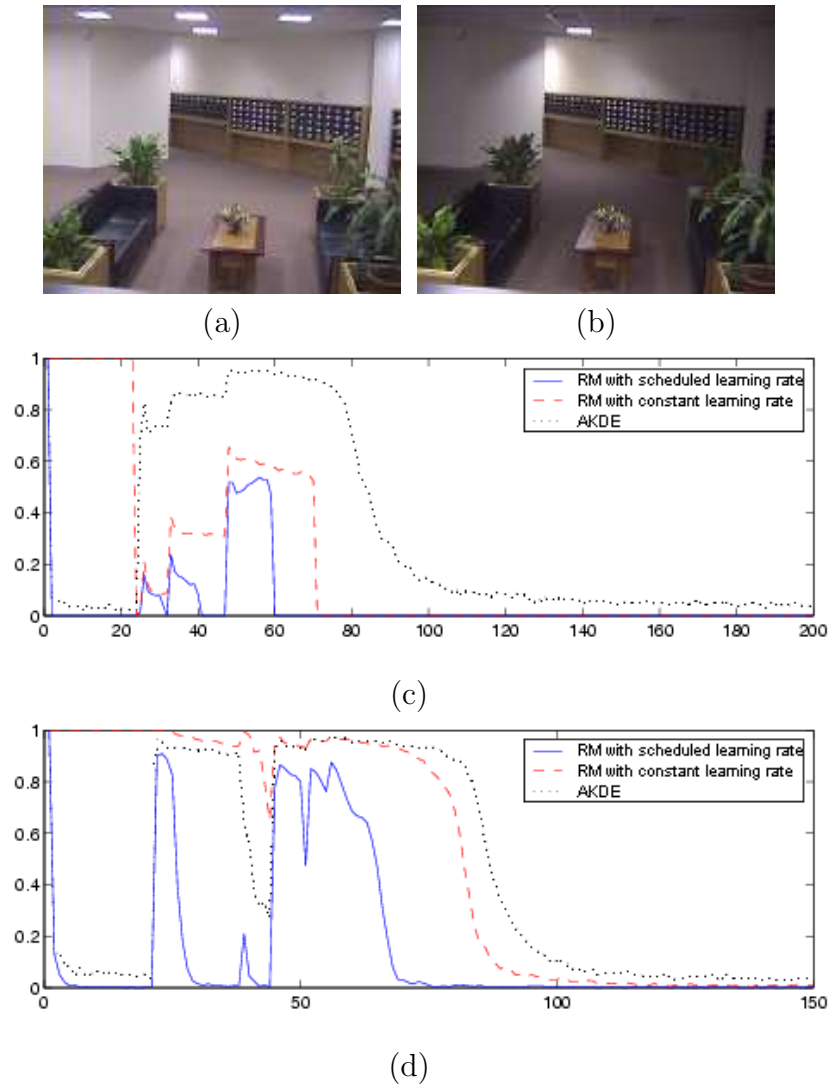


Figure 4.7: Sudden global changes in the background: (a) the *Lobby* video sequence with lights on. (b) Lights off. (c) Recovery speed comparison in lights turned off scenario. (d) Recovery speed comparison in lights turned on scenario.

#### 4.1.8 Other difficult examples

Figure 4.8 shows three video sequences with challenging backgrounds. In column (a) the original frames are shown; while columns (b), (c), and (d) show the results of the AKDE, the RM and the SVDDM methods, respectively. In this figure, from top row to the bottom; heavy rain, waving tree branches, and the water fountain pose significant difficulties in detecting accurate foreground regions.

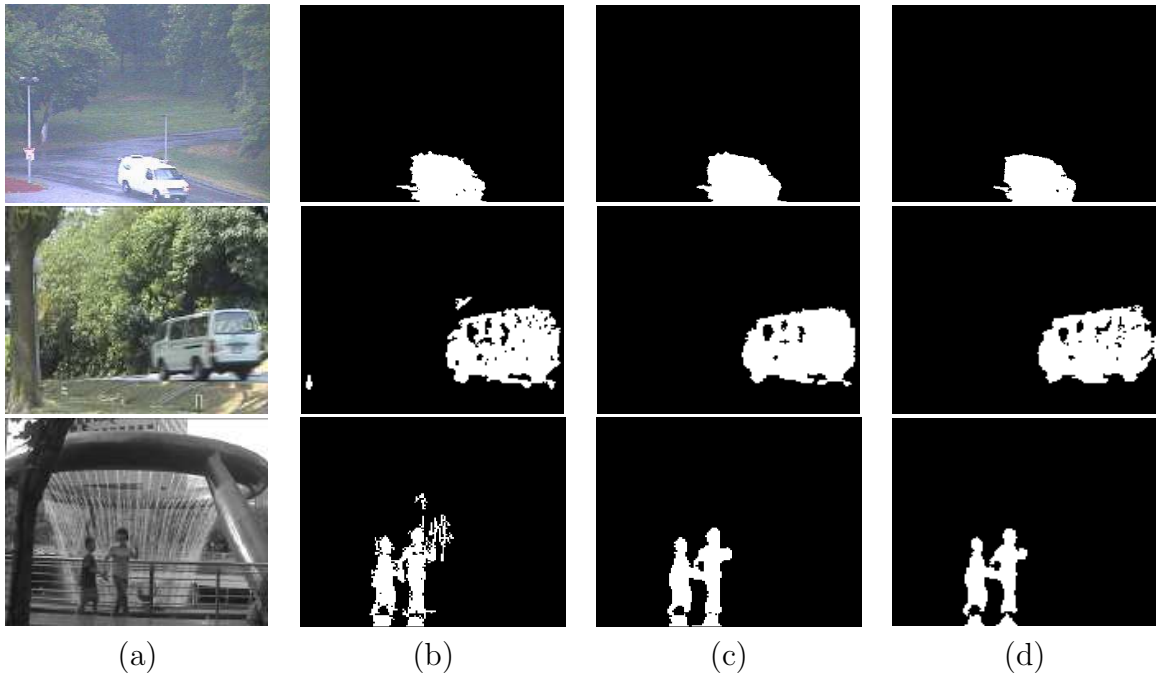


Figure 4.8: Other difficult examples: (a) Original frame. Detected foreground region using (b) AKDE. (c) RM. (d) SVDDM.

Table 4.1: Quantitative evaluation and comparison. The sequences are *Meeting Room*, *Lobby*, *Campus*, *Side Walk*, *Water* and *Fountain*, from left to right from [46].

Method	Video	MR	LB	CAM	SW	WAT	FT	Avg. $\mathcal{S}(\mathcal{A}, \mathcal{B})$
AKDE[81]		0.74	0.66	0.55	0.52	0.84	0.51	0.64
RM[86]		0.92	0.87	0.75	0.72	0.89	0.87	0.84
SVDDM[85]		0.84	0.78	0.70	0.65	0.87	0.80	0.77
Spatio-Temp[46]		0.91	0.71	0.69	0.57	0.85	0.67	0.74
MoG[75]		0.44	0.42	0.48	0.36	0.54	0.66	0.49

#### 4.1.9 Quantitative evaluation

Performances of our proposed methods, AKDE, RM, and SVDDM are evaluated quantitatively on randomly selected samples from different video sequences, taken from [46].

To evaluate the performance of each method a value “called similarity” measure is used. The similarity measure between two regions  $\mathcal{A}$  (detected foreground regions)

and  $\mathcal{B}$  (ground truth) is defined by [46]:

$$\mathcal{S}(\mathcal{A}, \mathcal{B}) = \frac{\mathcal{A} \cap \mathcal{B}}{\mathcal{A} \cup \mathcal{B}} \quad (4.1)$$

This measure increases monotonically with the similarity between detected masks and the ground truth, ranging between 0 and 1. By using this measure we report the performance of the AKDE method, the RM method, the SVDDM, the spatio-temporal technique presented in [46], and the mixture of Gaussians (MoG) in [75]. By comparing the average of the similarity measure over different video sequences in Table 4.1, we observed that the RM and the SVDDM methods outperform other techniques. This also shows that the AKDE, RM and SVDDM methods work consistently well on a wide range of video sequences. Also note that AKDE, RM, and SVDDM methods unlike the existing techniques, are automatic and do not need a large number of parameters to be fine-tuned for different scenes.

However, from this table one might argue that AKDE does not perform better than the method presented in [46]. The reason is that in [46] the authors used a morphological post-processing stage to refine their detected foreground regions. Note that the results shown for AKDE are the raw detected regions. We performed a morphological post-processing on the results obtained by the AKDE and the average similarity measure increased to 0.74 – which is similar to that of [46].

#### 4.1.10 Synthetic data sets

In this section we use a synthetic data set, which represents randomly distributed training samples with an unknown distribution function (*Banana* data set). Figure 4.9 shows a comparison between different classifiers. This experiment is performed on 150 training samples using the support vector data description (SVDDM), the

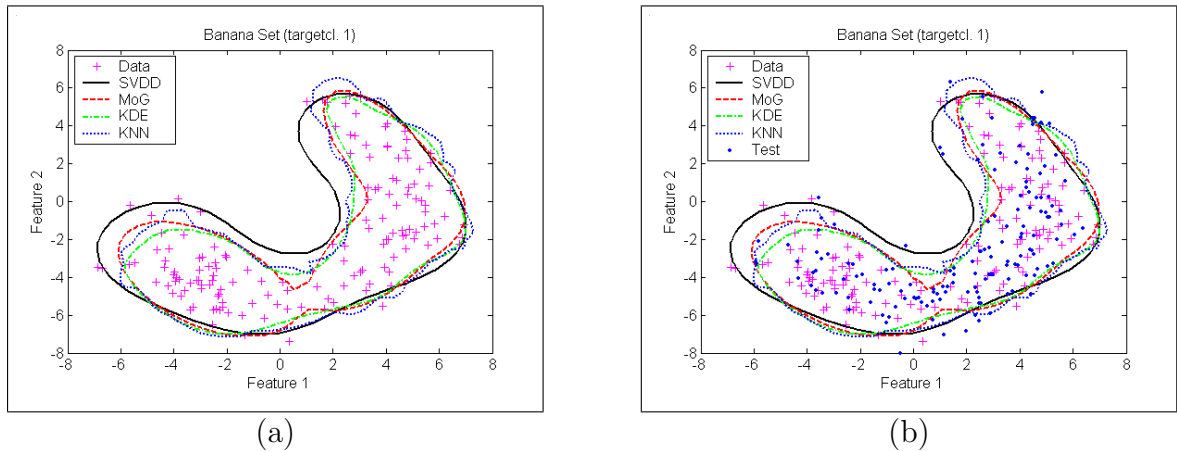


Figure 4.9: Comparison between different classifiers on a synthetic data set: (a) Decision boundaries of different classifiers after training. (b) Data points (blue dots) outside decision boundaries are false rejects.

mixture of Gaussians (MoG), the kernel density estimation (AKDE) and a k-nearest neighbors (KNN).

Parameters of these classifiers are manually determined to give a good performance. For all classifiers the confidence parameter is set to be 0.1. In MoG, we used 3 Gaussians. Gaussian kernel bandwidth in the AKDE classifier is considered  $\sigma = 1$ . For the KNN we used 5 nearest neighbors. And, for the SVDDM classifier the Gaussian kernel bandwidth is chosen to be 5.

Figure 4.9(a) shows the decision boundaries of different classifiers on 150 training samples from the *Banana* dataset. As it can be seen from Figure 4.9(b), SVDDM generalizes better than the other three classifiers and classifies the test data more accurately. In this figure the test data is composed of 150 samples drawn from the same probability distribution function as the training data. Therefore this should be classified as the known class.

Table 4.2: Comparison of False Reject Rate and Recall Rate for different classifiers.

Method	SVDDM	MoG	AKDE	KNN
FRR	0.1067	0.1400	0.1667	0.1333
RR	0.8933	0.8600	0.8333	0.8667

Here we need to define the False Reject Rate (FRR) and Recall Rate (RR) for a quantitative evaluation. By definition, FRR is the percentage of missed targets, and RR is the percentage of correct prediction (True Positive rate). These quantities are given by:

$$\text{FRR} = \frac{\text{\#Missed targets}}{\text{\#Samples}} \quad \text{RR} = \frac{\text{\#Correct predictions}}{\text{\#Samples}} \quad (4.2)$$

Table 4.2 shows a quantitative comparison between different classifiers. In this table, FRR and RR of classifiers are compared after training them on 150 data points drawn from an arbitrary probability function and tested on the same number of samples drawn from the same distribution. From the above example, the FRR for SVDDM is less than that of the other three, while its RR is higher. This proves the superiority of this classifier in case of single class classification over the other three techniques.

Table 4.3: Need for manual optimization and number of parameters.

Method	SVDDM	MoG	AKDE	KNN	RM
No. of parameters	1	4	2	2	2
Need manual selection	No	Yes	Yes	Yes	Yes

Table 4.3 compares the number of parameters that each classifier has and the need for manually selecting these parameters for a single class classification problem. The only method that automatically determines data description is the SVDDM technique. In all of the other classification techniques there is at least one parameter that needs to be manually chosen to give a good performance.

Table 4.4: Comparison of memory requirements for different classifiers.

Method	SVDDM	MoG	AKDE	KNN	RM
Memory needs (bytes)	1064	384	4824	4840	1024

Table 4.4 shows memory requirements for each classifier. Since in SVDDM we do

Table 4.5: Comparison of the incremental SVDD training algorithm with, online and batch methods on *Banana*, *Ellipse* and *Egg* data sets of size 1000.

Training	Data Set	Error	$F_1$	No. SV's	Time
Banana	Proposed	0.005	0.997	19	4.2
	Online	0.075	0.961	104	6.9
	Canonical	0.085	0.956	106	1697
Ellipse	Proposed	0.013	0.993	6	3.72
	Online	0.100	0.947	105	4.1
	Canonical	0.110	0.994	108	2314
Egg	Proposed	0.065	0.966	8	3.85
	Online	0.095	0.950	101	3.7
	Canonical	0.128	0.932	87	1581

not need to store all the training data, as can be seen from the table, it requires much less memory than the KNN and KDE methods. Only the MoG and the RM methods need less memory than the SVDDM technique. However in MoG methods we need to manually determine the number of Gaussians to be used, which is not practical when training one classifier per pixel in real video sequences.

**Error Evaluation.** Table 4.5 compares the classification error, the  $F_1$  measure, the number of the support vectors, and the learning time for the three learning methods. The experiments are performed on three data sets ('*Banana*', '*Ellipse*', '*Egg*') with 1000 training samples and 1000 test samples. The  $F_1$  measure combines both the recall and the precision rates of a classifier:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (4.3)$$

**Classification Comparison.** Table 4.6 compares the classification error, the  $F_1$  measure and the training and the classification asymptotic time for various classifiers. As seen, the incremental training of the SVDD's reaches very good classification rates compared to other methods. The SVDD's trained with batch and online training techniques give good classification accuracy – but are outperformed by our method both in terms of accuracy and efficiency. In all systems, the trade-off parameter is

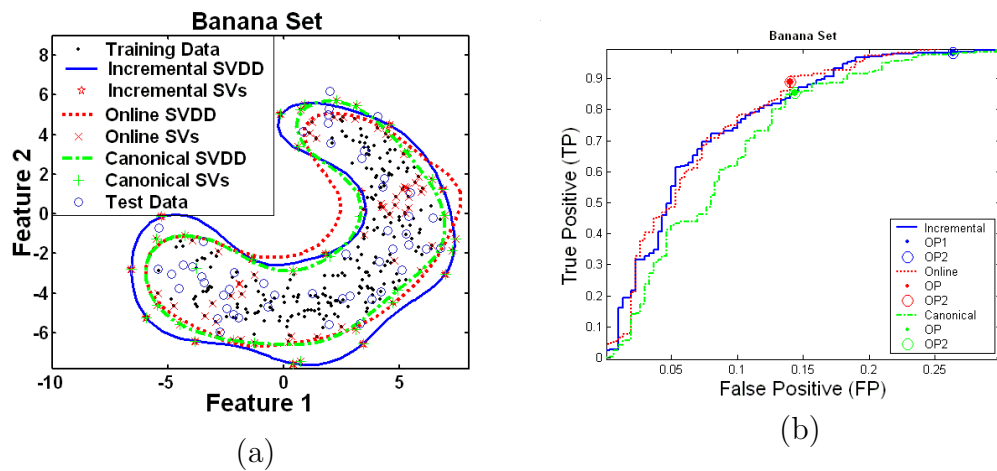


Figure 4.10: Comparison of incremental with canonical and online SVDD: (a) Classification boundaries . (b) Receiver Operating Curve (ROC).

set to be  $C = 0.1$ . Kernel bandwidth for the three SVDD methods and the Parzen window is  $\sigma = 3.8$ .  $K = 3$  is selected for the number of Gaussians in the MoG and number of nearest neighbors in the K-NN method.

**Classification Boundaries and Receiver Operating Curves.** In Figure 4.10 (a) the classification boundaries of the three SVDD training algorithms are shown. In this figure the blue dots are the training samples drawn from the *Banana* data set and the circles represent the test data set drawn from the same probability distribution function.

The  $\star$ ,  $\times$ , and  $+$  symbols are the support vectors of the Incremental, Online and Canonical SVDD training algorithms, respectively. As it can be observed the proposed incremental learning had fewer support vectors compared to both online and canonical training algorithms. From Figure 4.10 (a) it can be observed that the decision boundaries of the classifier trained using the Incremental algorithm (solid curve) is objectively more accurate than those trained by Online (dotted curve) and Canonical (dashed curve) methods.

Figure 4.10 (b) shows the comparison between the Receiver Operating Curve (ROC) of the three algorithms. The solid curve is the ROC of the Incremental

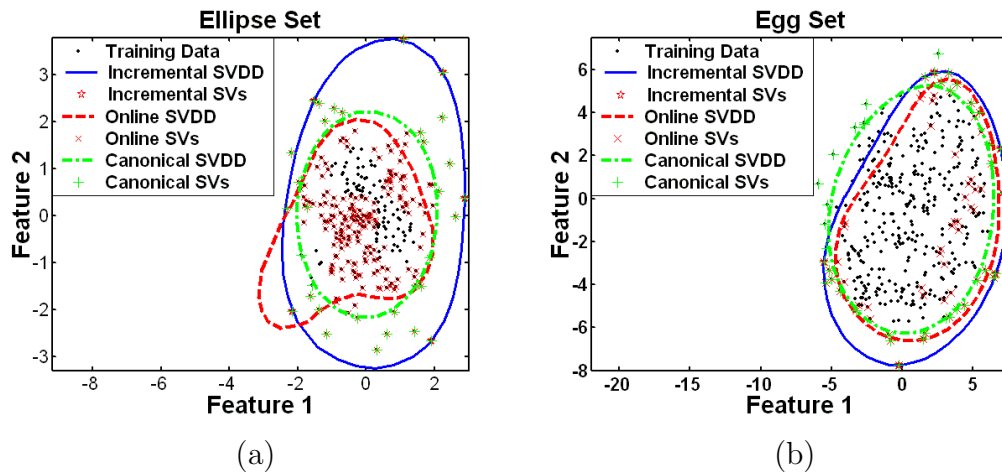


Figure 4.11: Comparison of incremental with online and canonical SVDD: (a) Normal data set. (b) Complex (egg) data set.

learning while dotted and dashed curves belong to Online and Canonical learning algorithms, respectively. Notice that the true positive rate is higher for small false positive rates in the case of the proposed incremental learning algorithm compared to both canonical and online learning. This can be expected since the proposed method explicitly uses the small trade-off parameter ( $C$ ) into account by learning the support vectors incrementally.

In Figure 4.10 (b) the operating point (OP) of the three ROC's (for the given trade-off value) are represented by the circle and dot symbols. As can be seen from this figure, the true positive rate for the proposed method is higher than both the Online and the Canonical methods. This shows that the proposed method, under the same conditions and with the same parameters, has higher precision and recall rates. We will investigate this claim quantitatively later.

Figure 4.11 shows a comparison of the classification boundaries, and the support vectors between the three SVDD training algorithms. Figure 4.11(a) shows the result of the classification on a 2-D normal distribution and Figure 4.11 (b) is the experiment on a data set drawn from a more complex distribution function in 2-D (egg shape). As seen from the figure the proposed incremental SVDD results in more accurate

Table 4.6: Comparison of the classification error,  $F_1$  measure, and asymptotic speeds with various classifiers on a *complex data set* of size 1000.

Classifier	Error	$F_1$	Training	Classification
Proposed	0.015	0.992	$O(1)$	$O(1)$
Batch SVD	0.100	0.947	$O(N)$	$O(N)$
Online SVD	0.103	0.945	$O(N)$	$O(N)$
KDE(Parzen)	0.114	0.940	$O(N)$	$O(N)$
MoG	0.143	0.923	$O(1)$	$O(1)$
K-means	0.150	0.919	$O(1)$	$O(1)$

Table 4.7: Comparison between the proposed methods and the traditional techniques.

	AKDE	RM	SVDDM	[18] <sup>1</sup>	[46] <sup>2</sup>	[75] <sup>3</sup>	[93] <sup>4</sup>
Automated	Yes	Yes	Yes	No	No	No	No
Post proc.	No	No	No	No	Yes	No	No
Classifier	Bayes	MAP	SVD	Bayes	Bayes	Bayes	K-means
Memory req.*	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Comp. cost*	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$

\* : Per-pixel memory requirements or computational cost

$n$ : number of training frames or training features used per pixel

1 : KDE

2 : Spatio-Temporal

3 : MoG

4 : Wallflower

classification boundaries than both online and canonical versions. Notice that the proposed method keeps a smaller number of support vectors to describe both data sets when compared to the other two methods.

#### 4.1.11 Comparison summary

Table 4.7 summarizes this study and provides a comparison between different traditional methods for background modeling in the literature and our proposed methods. The comparison includes; the classification type, the memory requirements, the computation cost and the type of parameter selection.

As can be seen from Table 4.7 the RM method uses a MAP decision criterion whereas other systems – except the SVDDM – use a Bayes classifier. The only method

Table 4.8: Scenarios where each method appears to be particularly suitable.

	AKDE	RM	SVDDM	[18] <sup>1</sup>	[46] <sup>2</sup>	[75] <sup>3</sup>	[93] <sup>4</sup>
Low contrast video	S*	NS**	S	S	NS	NS	NS
Close Bg/Fg colors	S	NS	S	NS	NS	NS	NS
Slowly changing background	NS	S	S	NS	S	S	S
Rapidly changing background	S	S	S	S	S	NS	S
Sudden global changes	NS	S	NS	NS	S	S	NS
Non-empty backgrounds	NS	S	NS	NS	S	S	S
Hand-held camera	NS	S	NS	NS	NS	NS	NS

\* : Suitable

\*\* : Not suitable

1 : KDE

2 : Spatio-Temporal

3 : MoG

4 : Wallflower

which explicitly deals with the single class classification is the SVDDM technique, which fits the description of data belonging to the background class in its rather simple training stage. Other methods shown in the table use a binary classification scheme and use heuristics or a more sophisticated training scheme to make it useful for the single-class classification problem of background modeling.

Table 4.8 shows different scenarios and illustrates where each method appears to be particularly suitable for foreground region detection. As expected the RM method is suitable for a wide range of applications except when the contrast of images in the video is low, however not as satisfactory as the results of the AKDE and the SVDDM methods in this scenario.

In case of low contrast videos, the AKDE and the SVDDM methods work reasonably well in detecting foreground regions. This is due to the fact that these techniques aim to build an accurate background model from a finite number of background training frames. Moreover, the accuracy of the background model is limited to the false reject rate of the classifier in the SVDDM technique and to the probability density estimate in the AKDE. This makes the SVDDM method unable to detect foreground

regions in those locations where the background color is very close to that of the foreground. On the other hand, the AKDE is more sensitive to noise compared to the SVDDM.

As discussed earlier and as seen in Table 4.8, the only method suitable for the hand-held camera scenario is the RM technique. The other methods fail to build a very long term model for the background model because of the fact that their cost grows linearly with the number of training background frames.

## 4.2 Object Tracking

In this section we compare the performance of the proposed technique in object tracking using several real video sequences that pose significant challenges. Also our algorithm's performance is compared with that of kernel-based tracking using Bhattacharyya Coefficient [8] with and without foreground detection and the structural-similarity measure of [48]. We use different scenarios to test the performance of the proposed techniques. Our performance evaluation consists of an evaluation of the frame rates of the systems. We have depicted their ability to resolve collision and their robustness to changes in the objects' appearances and slight lighting changes.

### 4.2.1 Frame rate and tracking speed

As discussed in the previous sections the computational complexity of the proposed method is less than the existing object tracking algorithms in the literature. By limiting the search for potential targets to the linear list of detected objects in new frame, we decreased the search space. The advantage of this mechanism is the increased speed while tracking multiple objects. Figure 4.12 shows the performance of our method in terms of frame rate in comparison with two kernel based approaches

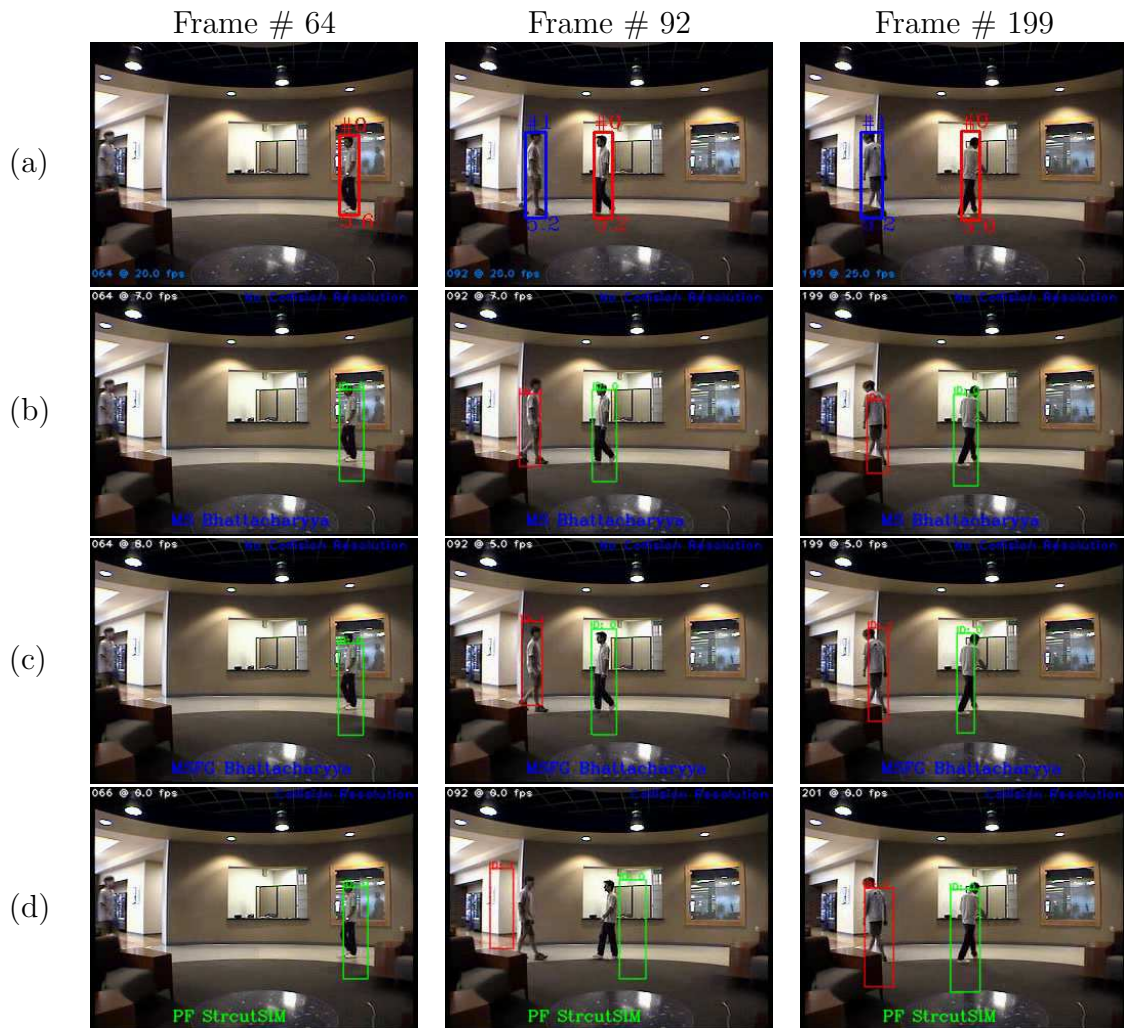


Figure 4.12: Comparison of our visual object tracking algorithm with several traditional methods: (a) our method, (b) mean-shift algorithm, (c) mean-shift algorithm using foreground masks, (d) particle filter using structural similarity measure.

as well as a particle filter tracking algorithm.

Figure 4.12 (a) shows the results of our proposed spatio-spectral connected component tracking algorithm while Figure 4.12 (c)-(d) present the tracking results of the mean-shift algorithm [8], a mean-shift algorithm which uses the detected foreground object masks, and a particle filter with structural similarity measure [48].

From Figure 4.12 we can easily conclude that the proposed approach works faster than the traditional algorithms. Also, the processing speed does not drastically de-

crease by introducing more objects to the scene. In the leftmost column of Figure 4.12 the systems track the only object which has been reliably detected. Our approach tracks the object with real-time performance of 20-25 frames per second (fps) while the single object tracking speed for the mean-shift algorithms is 7-10 fps (Figure 4.12 (b) and (c)).

By introducing the second object to the scene (the center and the rightmost columns in Figure 4.12), the tracking speed of the mean-shift algorithms drop to 4-7 fps compared to 20-25 fps in our algorithm. This shows that speed of the proposed tracking framework is not considerably affected by the presence more than one objects in the scene. Notice that the time required for the particle filtering approach to track objects was more than one second (4.12 (d)). Therefore, this algorithm is not suitable for applications which require real-time processing speed. In addition to its low processing speed, the particle filtering approach missed the objects in some frame (middle column in Figure 4.12 (d)).

### 4.2.2 Tracking performance in the presence of collision

One of the issues that many visual object tracking algorithms need to consider, especially in cases in which multiple moving objects appear at the same time within the scene, is the possibility of collision. As discussed in Section 3, a collision occurs when one object partially or totally occludes one or more other objects. Unfortunately, the appearance of the occluded objects become unreliable and this fact results in loss of tracking trajectories or even loss of the object appearance.

Figure 4.13 presents the results of the proposed algorithm, the two mean-shift based method, and the particle filter based on a video with two successive collisions, one occurring from frame 311 through frame 325 and the other from frame 373 to frame 385. The figure show two frames per collision, one taken before the collision and

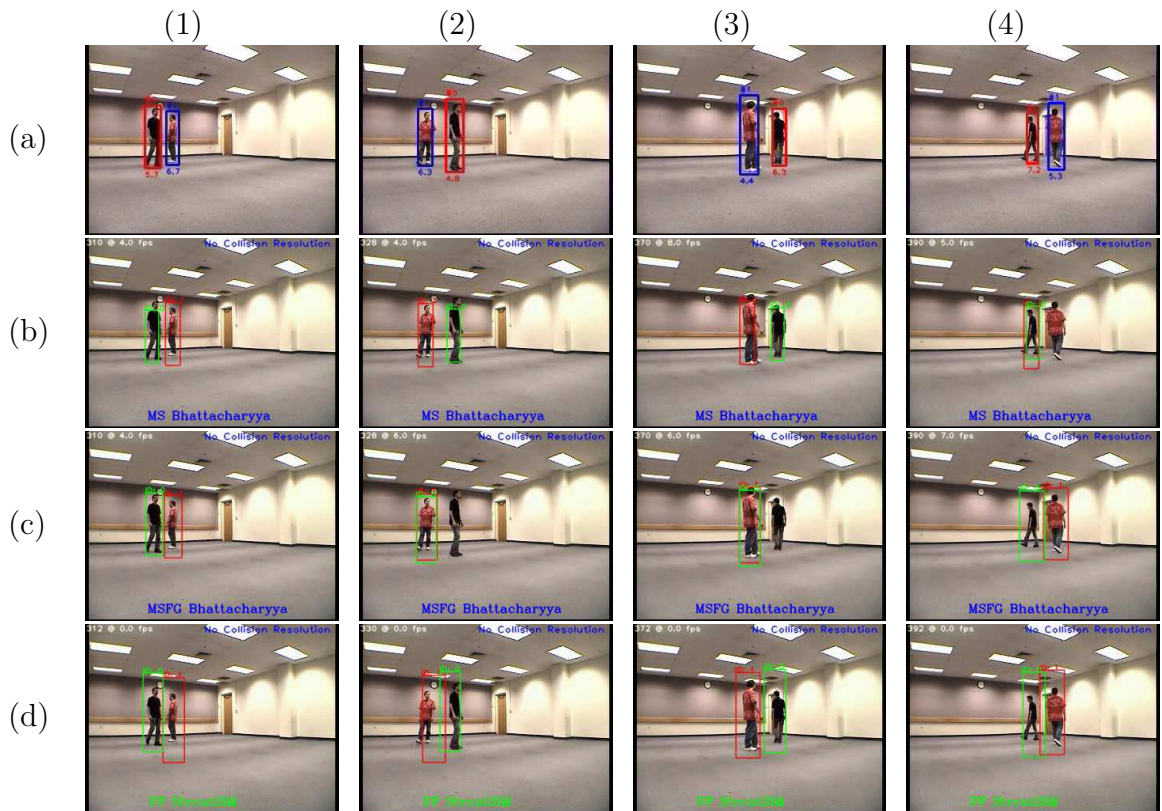


Figure 4.13: Comparison of our visual object tracking algorithm with several traditional methods in the presence of collision: (a) our method, (b) mean-shift algorithm, (c) mean-shift algorithm using foreground masks, and (d) Particle Filter using structural similarity measure.

the other afterwards. In this figure rows (a)-(d) represent the results of our approach, the mean-shift algorithm, the mean-shift algorithm with the foreground masks, and the particle filter technique, respectively.

Columns (1) and (2) in Figure 4.13 show the pre and post collision frames for the first collision. From Figure 4.13 (a) it can be observed that the proposed collision resolution mechanism in our algorithm was able to effectively detect the collision and resolve it accordingly without the loss of object tracks. In this case the mean-shift and particle filter based methods (Figure 4.13 (b) and (d)) could also keep the tracking trajectories of the objects while the mean-shift algorithm which used the foreground regions (Figure 4.13 (c)) lost the track of occluding object.

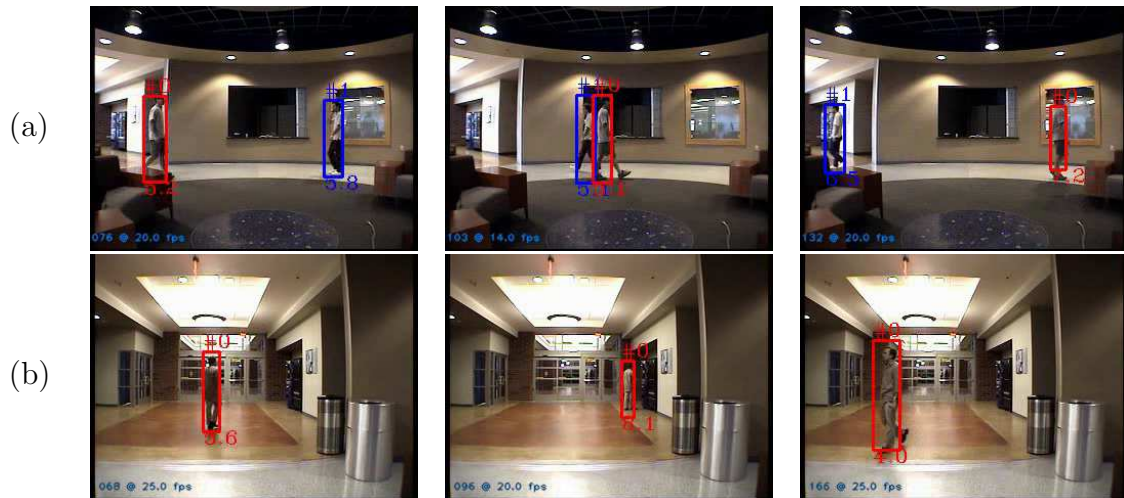


Figure 4.14: The tracking results of the proposed visual object tracker under challenging conditions: (a) illumination changes, and (b) reflective surfaces.

By examining columns 3 and 4 of Figure 4.13 we confirm that the proposed collision resolution mechanism in our approach was able to handle the second collision as well (Figure 4.13 (a)). However, the mean-shift algorithm in Figure 4.13 (b) lost the track of the occluding object. In columns 3 and 4 of Figure 4.13 (c) the collision caused the occluding object to lose one of its tracks. Notice that in columns (1) and (2) of Figure 4.13 (c) one of the objects acquired the appearance and tracks of the other and lost them during the second collision. Figure 4.13 (d) shows that the particle filter approach was robust to both collisions. However, as noted earlier this approach is very slow compared to our proposed technique and its tracking results are not as accurate as our algorithm.

### 4.2.3 Other challenging experiments

Figure 4.14 shows two challenging scenarios for visual object tracking algorithms which deal with target representation and localization. In Figure 4.14 (a) three frames of a video taken in a dark lobby with different local lightings is shown. Since lighting changes across the lobby the objects' appearances change in differ locations. The

proposed algorithm shows robustness to this particular challenge as it is able to keep track of objects while their appearances change. Notice that the algorithm was also able to resolve the collision while the persons were passing by each other.

Two difficult scenarios were tested in Figure 4.14 (b). The video in this figure was taken in a bright corridor with flickering lights and bright, shiny surfaces. In addition to the reflective surface of the corridor the object of interest appearance (clothing colors) was close to that of the background. As seen from Figure 4.14 (b) the proposed object tracking algorithm robustly tracks the object without losing its accurate appearance model.

## Chapter 5

# A Robotic Application for the Proposed Framework

Recognizing the intentions of others is an important part of human cognition, especially in activities in which collaboration or competition play an important role. To achieve autonomy, robots will have to have similar capabilities. This will be especially critical in tasks requiring substantial human-robot interaction.

Any system that uses visual clues to infer the intent of active agents in a scene requires a robust mechanism to track these agents at pixel level, to accurately generate and to maintain their trajectories at the world level, and simultaneously to pass this world-level information to higher level stages of the process. In this chapter, we present a robotic system that performs these tasks and is robust to illumination changes, background clutter, and to occlusion. The proposed visual object tracking algorithm is successfully employed in this system while running in real-time. We collaborated with the robotics research lab to apply our visual tracker in the robotic application. The details and variant implementations of this system are published in [38] and [39]. Details about the visual capabilities of the system are published in [77] and [78].

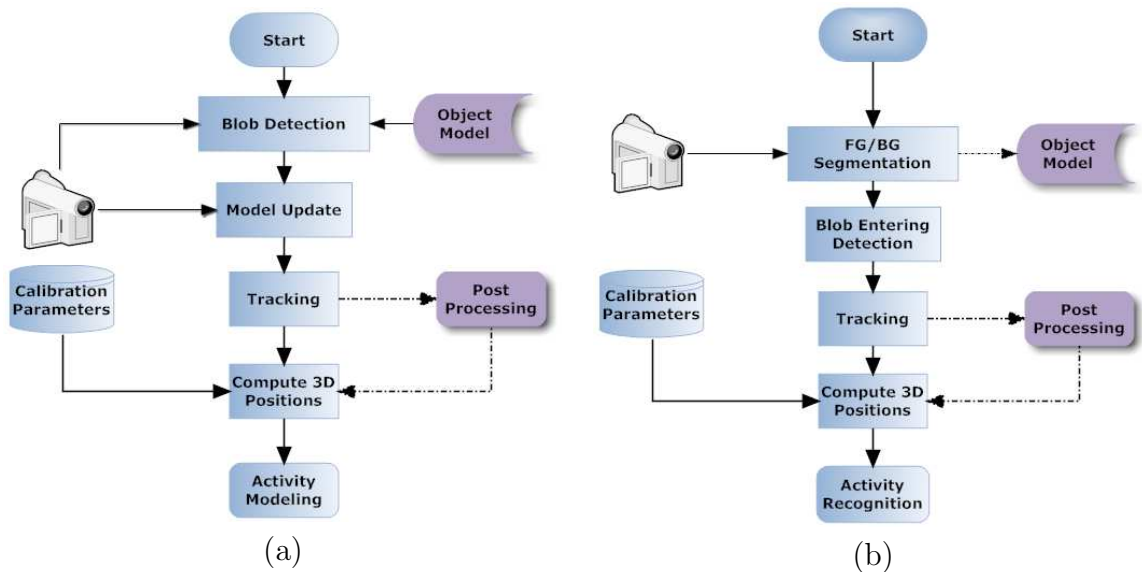


Figure 5.1: The two object tracking frameworks for (a) *activity modeling* using a modified mean-shift tracker and (b) *intent recognition* using a spatio-spectral tracker.

## 5.1 Introduction

We presented our object tracking framework as a component of the robotic system to provide it with visual capabilities that facilitate the modeling and recognition of actions carried out by other agents. As the appearance of these agents is generally not known a priori, the only visual cue that can be used for detecting and tracking them is image motion. Although it is possible to perform segmentation from an image sequence that contains global motion, such approaches – typically based on optical flow estimation [14] – are not very robust and are time consuming. Therefore, our approach uses more efficient and reliable techniques from real-time surveillance, based on background modeling and segmentation:

- During the *activity modeling* stage, the robot is moving while performing various activities. The appearance models of other mobile agents, necessary for tracking, are built in a separate and earlier prior process in which the static robot observes each agent that will be used for action learning. The robot uses

an enhanced mean-shift tracking method to track the foreground object.

- During the *intent recognition* stage, the static robot observes the actions carried out by other agents. This allows the use of a foreground-background segmentation technique to build appearance models on-line, and to improve the speed and robustness of the tracker. The robot is stationary for efficiency reasons. If the robot moves during intent recognition we can use the approach from the modeling stage.

Figure 5.1 shows the block diagram of the proposed object tracking frameworks. The main difference between the two algorithms is the absence of global motion in the background in the *intent recognition* stage. In this stage the system reliably detects moving objects and segments them from the background clutter or local motion by employing our object detection algorithm described in Chapter 2. After objects are detected the proposed object tracking technique, Chapter 3, is used to generate tracking parameters such as distances, angles, etc. for the robotic component of the system.

In the *activity learning* stage the robot performs the activities. Therefore, the background of the videos contain global motion. Since the background modeling will not be useful in this case, we acquire the object appearance models in a separate process. In this process the robot is static and observes the trainers who will train it for different activities. The object detection algorithm proposed in this research is employed to detect the trainers and build their appearance models. These models are used in an enhanced mean-shift algorithm to track objects while the robot is moving.

According to the mean-shift object tracking algorithm [8] the histogram of the object model and the target are matched against each other. The Bhattacharyya coefficient is used as the distance measure to find the best target match. The original mean-shift algorithm performs the search for the target in a spatial window around

the object in the previous frame. We expand the search space to include the size of the new target as well as its location. This process ensures that the objects location as well as its accurate size will be found by employing the mean-shift algorithm.

Once the robot is trained for different activities and is able to robustly track objects in its field of view, it will be deployed to the observation location. The observer robot uses the *intent recognition* tracking algorithm to detect and track objects and infer from their tracking parameters conclusions about their intention.

## 5.2 Context-Based Intent Recognition

The information from the vision system is used by the intent recognition module. This module has two parts: a set of activity models and a set of context models. The overall architecture is inspired by the Theory of Mind [63].

### 5.2.1 Hidden Markov Models for Representing Activities

Our activity modeling approach uses hidden Markov models (HMMs). In our system, a single HMM models a single basic activity (“two people approach one another”). The hidden states of such a model correspond to “small-scale” intentions or components of a complex activity. The visible symbols of our models correspond to changes in the parameters produced as the output of the tracking module. The system is trained using the approach described above during the activity modeling stage and the Baum-Welch algorithm.

### 5.2.2 Context Models for Inferring Intent

Recent neuroimaging work [68] suggests that humans use the context in which an activity is performed to infer the intent of an agent. Our system mimics this capability

to improve intent recognition.

In our system, an *intention* consists of a name and an activity model, and a *context* consists of a name and a probability distribution over the possible intentions that may occur in the current context. The goal is to determine the most likely intention of an agent, given a sequence of observations of that agent's trajectory and the current context. Letting  $s$  be an intention,  $v$  be a sequence of visible symbols, and  $c$  be a context, we want to find

$$\arg \max_s p(s|v, c).$$

Applying Bayes' rule and simplifying, this is equivalent to finding  $s$  that maximizes

$$p(v|s, c)p(s|c).$$

Moreover, the above product can be further simplified to

$$p(v|s)p(s|c).$$

This follows from the definition of a context. Since we assume that a context only provides a distribution over intention names, knowing the current context tells us only about the relative likelihood of the possible intentions. A context says nothing about the hidden or observable states of any activity model, and so does not affect the independence assumptions of the underlying HMM.

Thus to find the most likely intention given a context  $c$  and observation sequence  $v$ , compute  $p(v|s)p(s|c)$  for each intention  $s$ , and select the one whose probability is largest. The value of  $p(v|s)$  can be calculated using the HMM that belongs to  $s$ , and  $p(s|c)$  is assumed available in the definition of the system's context.

### 5.2.3 Intention-Based Control

The proposed system is also able to dispatch behaviors on the basis of an observed agent's intentions. To do this the system performs its inference as above and examines both the current context and the decoded internal state of the activity model. On the basis of this state the robot performs a predefined action or set of actions.

## 5.3 Experimental Validations

To validate the approach, a set of experiments are performed using a Pioneer 2DX mobile robot with an onboard computer, a laser rangefinder, and a Sony PTZ camera. The robot is trained to understand three basic activities:

- *following* – in which one agent trails behind another,
- *meeting* – in which two agents approach one another directly, and
- *passing* – in which two agents move past each other without otherwise directly interacting.

The trained robot was placed in an indoor environment and observed the interactions of multiple human agents with each other, and with multiple static objects. Of particular interest was the performance of the system in two cases.

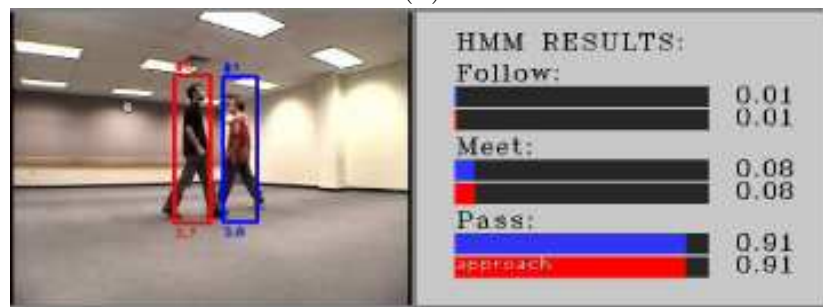
In the first case, we wanted to determine the performance of the system when a single activity could have different underlying intentions based on the current context (i.e. the activity of “moving ones hand toward a chess piece” could be interpreted as “making a move” during a game but as “cleaning up” after the game is over). This case deals directly with the problem that in some situations, two apparently identical activities may in fact be very different, although the difference may lie entirely in the contextually determined intentional component of the activity.



(a)



(b)



(c)

Figure 5.2: Local intention recognition results in: (a) following, (b) meeting, and (c) passing by scenarios.

In the second case of interest, we sought to determine the performance of the system in disambiguating two activities that were in fact different, but due to environmental conditions appeared superficially very similar. This situation represents one of the larger stumbling blocks of systems that do not incorporate contextual awareness.

### 5.3.1 The results

In order to show the performance of the system over simple intentions such as *following*, *passing*, and *meeting* the observer robot is given videos containing these local intentions. Figure 5.2 shows that the robot is able to infer the correct activity and intent for all the scenarios: the probability for the correct model rapidly exceeds the other models, which have very low likelihoods.

To test the system, the robot observed interactions among multiple human agents and multiple static objects in three sets of experiments. To provide a quantitative evaluation of intent recognition performance, two measures are used:

- *Accuracy Rate* = the ratio of the number of observation sequences, of which the winning intentional state matches the ground truth, to the total number of test sequences.
- *Correct Duration* =  $C/T$ , where  $C$  is the total time during which the intentional state with the highest probability matches the ground truth and  $T$  is the number of observations.

In all of the scenarios considered, the robot was able to effectively track the agents within its field of view and correctly infer the intentions of the agents that it observed.

The accuracy rate of the system is 100%: the system ultimately chose the correct intention in all of the scenarios in which it was tested. This shows that the outstanding result achieved are, in part, due to the accurate tracking information provided by the visual object tracking module of the system.

As mentioned earlier two sets of experiments were performed to validate the contextual information processing. In the first set of experiments, the same footage was given to the system several times, each with a different context, to determine whether the system could use the context alone to disambiguate agents' intentions.

Table 5.1: Quantitative Evaluation

Scenario (with Context)	Correct Duration [%]
Leave building (Normal)	96.2
Leave building (Evacuation)	96.4
Theater (Cleanup)	87.9
Theater (Movie)	90.9
Vending (Getting Drink)	91.1
Vending (Repair)	91.4
Meet (Unbiased) - Agent 1	65.8
Meet (Unbiased) - Agent 2	72.4
Meet (Biased) - Agent 1	97.8
Meet (Biased) - Agent 2	100.0

Three pairs of scenarios were considered:

- *leaving the building on a normal day/evacuating the building,*
- *getting a drink from a vending machine/repairing a vending machine,*
- *and going to a movie theater during the day/going to clean the theater at night.*

Quantitative results can be seen in Table 5.1.

The second set of experiments were performed in a lobby, and had agents meeting each other or passing each other, both with and without contextual information. The context in these scenarios accounts for the likelihood of each of these two activities. Results for the two agents involved are given in Table 5.1 for both the context-free and the context-assisted *meet* scenario.

The intention-based control is tested in two scenarios. In the first, the robot observes a human dropping a bag in hallway, recognizes this as a suspicious activity, and follows the human. In the second scenario, an agent enters an office and sets his bag on the table. Another agent enters the room and steals the bag. The observer robot recognizes the theft and then signals another robot in the hallway. The second robot then finds the thief, and follows him.

In both cases, the observer robot was able to correctly use intentional information to dispatch the context-appropriate behavior. Additionally, by using context to cut down on the number of intentions the robot has to consider in its maximum likelihood calculation, we improve the efficiency of our system.

### **One activity, many intentions**

The first six rows of Table 5.1, indicate the systems disambiguation performance. For example we see that in the case of the scenario, *Leave Building*, the intentions Normal vs. Evacuation are correctly inferred 96.2 and 96.4 percent of the time, respectively. We obtain similar results in two other scenarios in which the only difference between the two observed activities in question is the intentional information represented by the robots current context. We see therefore that the system is able to use this contextual information to correctly disambiguate intentions.

### **Similar-looking activities**

As we can see from the last four rows of Table 5.1, the system performs substantially better when using context than it does without contextual information. Because *meeting* and *passing* can – depending on the position of the observer – appear very similar. Without a context it may be hard for the autonomous machine to decide what the two agents are trying to do. With the proper contextual information it becomes much easier to determine the intentions of the agents in the scene.

Figure 5.3 shows the correct recognition of similar looking activities in the presence of contextual information. The visual object tracker prepares the tracking trajectories for the observable agents. However, these trajectories are very similar for intentions which share a great amount of similarities before their completion (i.e. meeting vs. passing by).

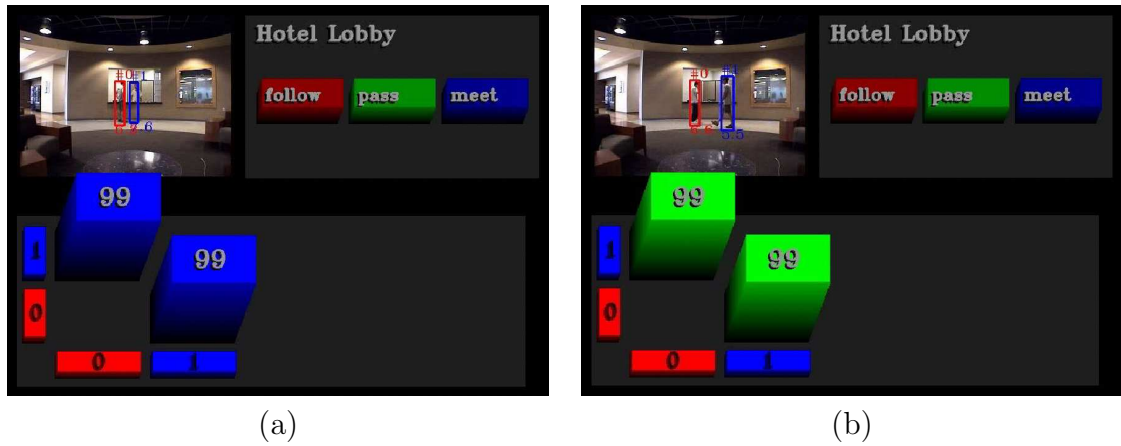


Figure 5.3: Recognition of similar-looking activities by context enforcement: (a) meeting and (b) passing by scenarios.

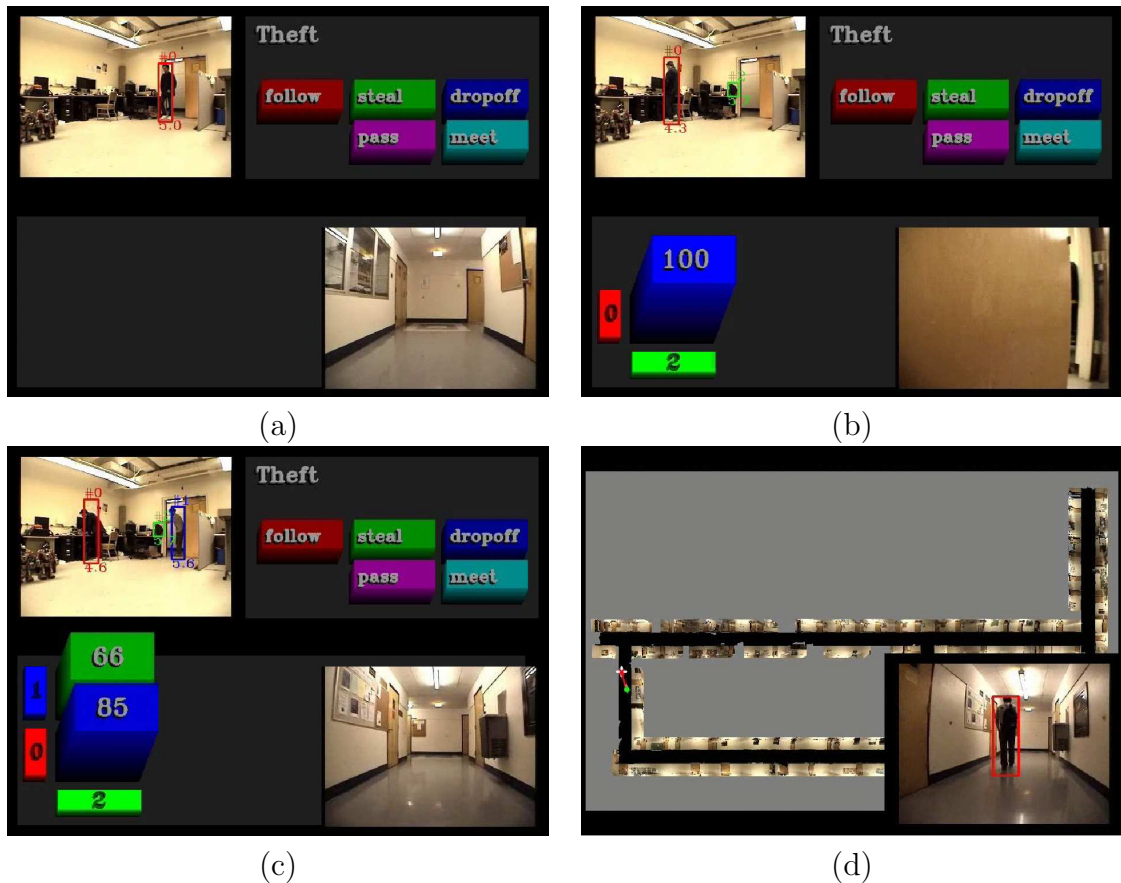


Figure 5.4: The results of visual tracking algorithm and the intent recognition in the theft scenario: (a) person detected, (b) the first person and the bag are being tracked, (c) the second person is approaching the bag, (d) the theft has been reported along with the coordinates of the event and the perpetrator's appearance to the patrol robot.

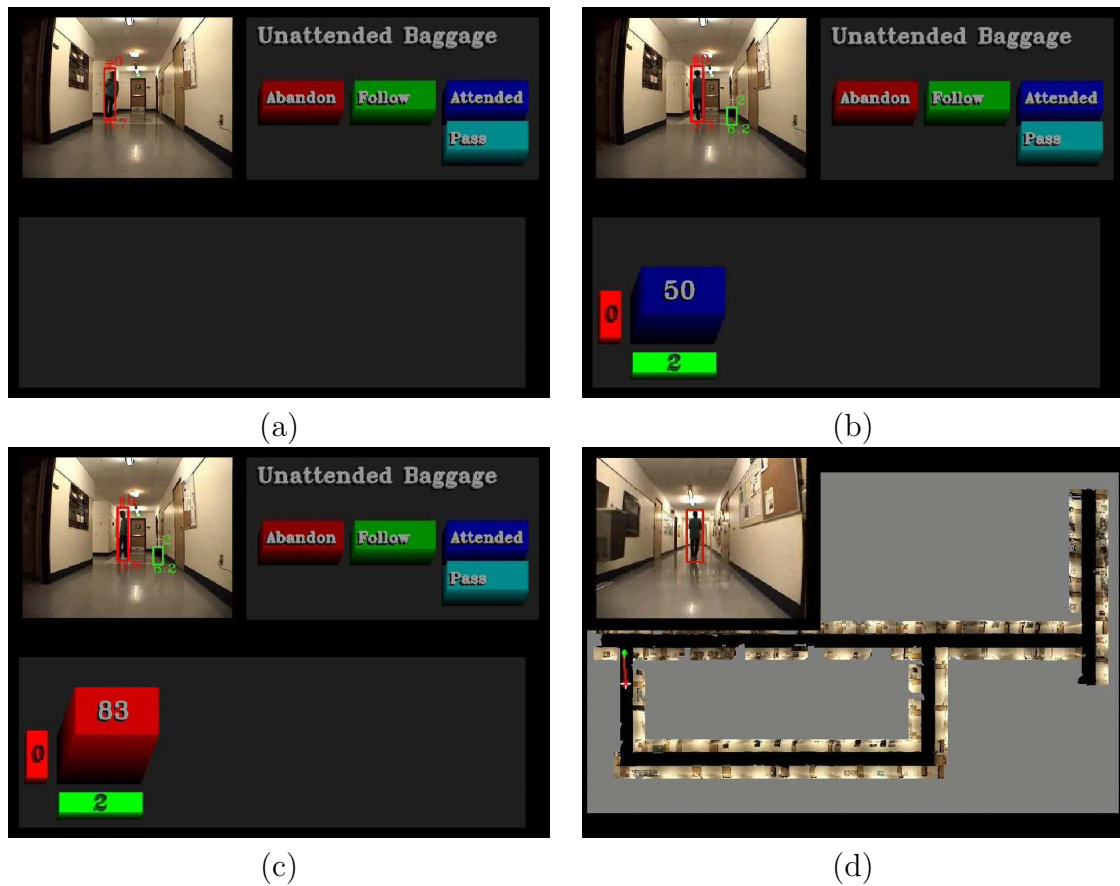


Figure 5.5: The results of visual tracking algorithm and the intent recognition in the unattended bag scenario: (a) person detected, (b) the person and the bag are being tracked, (c) the person leaves the bag unattended, (d) the robot tracks the person and follows him.

### Intent-based control

In both the scenarios we developed to test our intention-based control, our robot correctly inferred the ground truth intention, and correctly responded to the inferred intention.

In particular two scenarios were presented. In the first case –called the theft scenario– an observer robot with the visual tracking capabilities developed in this dissertation and learned set of activities watches an office, Figure 5.4. When a person enters the room, he/she is detected and tracked (Figure 5.4-a). Upon the detection of the bag its model is created and tracked (Figure 5.4-b). The system tracks the

new person along with the bag and its owner and generates information for the intent recognition module (Figure 5.4-c). At this point the intent recognition module with 66% confidence, reports the occurrence of theft to a patrol robot whose viewpoint is shown in the bottom-right corner of the Figure 5.4. The patrol robot moves toward the coordinate where the theft took place and tracks the perpetrator by his/her appearance (Figure 5.4-d).

In the second case – the unattended bag scenario – the goal is to detect whether a person who has dropped off a bag has the intention of leaving it unattended. An observer robot oversees a secure area (Figure 5.5). Once people appear in the scene the system tracks them (Figure 5.5-a) and keeps track of their bags or belongings that they have position (Figure 5.5-b). In this scenario the person intends to leave the bag unattended. At some point the robot gathers enough evidence of negative intent (Figure 5.5-c). When that threshold is reached it stops the process of intent recognition and follows the perpetrator while tracking his appearance.

## Chapter 6

# Conclusions and Future Work

Visual object tracking is a crucial and important step in many high-level video processing applications such as video indexing, video compression, video surveillance and activity recognition in videos. By analyzing the tracking trajectories of objects in a video, useful information can be produced about the actions and interactions of objects in the scene. Such information greatly enhances the performance of any application which requires the processing of visual data from video sequences.

### 6.1 Summary

In this dissertation, previous efforts in visual object tracking in the literature were studied and their limitations and weaknesses have been investigated. Based on the requirements of the real-time applications a non-parametric object tracking framework is proposed. This technique is composed of two major components.

The first part of the process deals with object detection in videos with quasi-stationary background. Three novel methods to detect foreground regions have been presented in this research and their performances were comprehensively studied, both theoretically and practically.

The first proposed method is a statistical adaptive technique (the AKDE) which

uses a set of background frames to generate an adaptive and accurate background model for each pixel in the scene. These models are used in a background subtraction stage (classification stage) to label each pixel in new frames as a background or a foreground pixel. As the theoretical investigation and qualitative experiments showed, this technique is particularly suitable for backgrounds whose changes occur fast and can be represented in a small temporal window. However, the number of background training frames (temporal window size) plays a critical role in the accuracy of this technique. If the changes in the background are slow and can not be covered in a small enough temporal window the accuracy of this method degrades. This algorithm is used as a base-line non-parametric background modeling technique. This base-line method is useful in evaluating the performance of more sophisticated approaches proposed in this dissertation.

The second technique proposed in this research (the RM) aims to address the issue of slowly changing backgrounds, which makes the AKDE method fail in some experiments. To solve this problem one should remove the dependency of the statistical background modeling to the number of training frames while preserving the complexity of the system. A recursive learning method is proposed which uses the background model for each pixel in the previous frame and recursively generates a new model as new frames appear. The research on this technique showed that the complexity of this system is independent of the number of training frames which proves that the system accuracy is independent of the number of background training frames.

The experiments revealed the fact that not only can this technique be used to address the issue of slowly changing backgrounds but it also opens the door to addressing several other issues such as hand-held camera, non-empty backgrounds and sudden illumination changes. However, since both the AKDE and the RM methods are statistical techniques their accuracy is limited to their estimation of the probability density

function for the background at each pixel. Furthermore, there is an additional issue with all statistical foreground detection techniques including the AKDE and the RM method. In statistical methods the assumption is that there are two classes, namely foreground and background, and the model is trained on background samples which are present during a short period of time (the AKDE) or from the beginning of the video (the RM). Note that until a foreground object is present in the scene, there is no information about the foreground class. This problem is addressed by using and training thresholds in the classification stage to label pixels as foreground or background.

To explicitly address the above issues –dependence on probability estimation and the single-class classification problem– a third non-statistical background modeling technique based on the Support Vector Data Description (SVDDM) is proposed. The SVDDM uses support vectors to generate a description for the known data; i.e., the background. An incremental learning algorithm is presented to train the classifiers for each pixel. These support vectors along with the classifier information for each pixel are stored and used in the classification stage to label pixels in new frames as foreground or background. The performance of this system is studied and its experimental results on both synthetic data and real video sequences are compared with other proposed methods and existing techniques in the literature.

The second component of the visual tracking approach deals with tracking the detected objects through the video. This dissertation proposed an approach which takes advantage of the object detection process for tracking purposes. The detected objects are used to generate photometric and geometric appearance models. These appearance models are employed to reduce the target localization search space to the list of visible objects in the scene. As discussed earlier, the experimental evaluation indicate that this technique is faster than kernel-based approaches and shows more

scalability. The performance of the proposed tracking algorithm is also compared to particle filter based methods. The results obtained from our method showed superior performance over the traditional techniques.

In addition, a collision detection and resolution mechanism is introduced to the object tracking framework. This module is responsible for predicting the possibility of collision between the foreground masks of two or more objects. Once a collision is detected the tracking algorithm stops for the occluded objects to maintain their pre-collision appearance. The collision resolution mechanism is tested in several scenarios and the results show significant improvement over the kernel-based methods.

Finally, a robotic application is presented in which the visual tracking framework is successfully employed. This robotic system looks are inferring the intents of moving objects in its field of view. The proposed tracking algorithm keeps track of objects of interest in the scene. The objects' tracking trajectories are maintained and passed to a mid-level process. This process uses the tracking trajectories for each object and produces interaction parameters for each object pair such as their relative distances, and angles. The robotic system takes advantage of this information to recognize the intentions of agents (also known as objects of interest).

## 6.2 Conclusions

The visual object tracking framework proposed in this dissertation is based on non-parametric representations of background and object models. This fact enables the models to assume any format dictated by the samples of background and object pixels observed within the field of view, regardless of any predefined parametric form. The non-parametric representation of pixel models addresses the issue of scene dependence in visual object tracking algorithms to a great extent.

Several experiments were conducted containing a number of challenging scenar-

ios. Three main approaches proposed in this dissertation for modeling background pixels and foreground objects were comprehensively evaluated and compared against each other as well as with the approaches in the literature. The experiments revealed the strengths and weaknesses of the proposed approaches under a variety of circumstances.

There is a trade off between the system memory requirements and the system accuracy in every scenario which employs visual object tracking. The proposed approaches in this dissertation proved to compliment each other addressing a wide range of scenarios without the need to tune them to any specific scene. Therefore, these approaches, if applied according to the system requirements provide a framework independent of any scene specification or type. The only criteria to consider while applying the suitable proposed non-parametric mechanisms are the amount of changes in the scene and the system memory requirements.

The successful implementation of the visual object tracking framework to a robotic system proves its efficiency in real life applications. The accuracy of the intent recognition in this robotic application requires the visual object tracker to possess two properties: to run in real-time, and to provide accurate tracking trajectories for objects of interest. As evaluated on several scenarios under different and challenging circumstances, the proposed visual object tracker maintained its efficiency in terms of speed and its accuracy of the tracking information.

### **6.3 Future work**

Future research on the visual tracking can be categorized along two directions. The first direction looks at the object detection component of the system. There is also the possibility of future research on the spatio-spectral object tracking component of the system as well.

This dissertation proposed two powerful object detection algorithms, each useful for a particular set of applications. In the recursive modeling approach the effectiveness of adaptive histogram bin sizes on decreasing the memory requirements of the system is currently under investigation. The impact of such approach on the foreground detection accuracy needs to be evaluated as well. The incremental support vector data description training mechanism resulted in an efficient detection technique. However, the incremental support vector training is an approximation to the optimal solution of the Lagrange optimization problem. I would like to investigate approaches from computational geometry which would yield fast and efficient solutions to the optimization problem while providing the optimal solution.

In the second component of the system, a spatio-spectral connected component tracking with a collision resolution mechanism is proposed. In the current implementation the photometric appearances of the objects are estimated by a single degree statistical model. In order to improve the accuracy and robustness of the tracking more complex and descriptive models for the appearances of the objects should be investigated.

Advances in visual object tracking mechanisms will result in more relevant, useful and efficient channels of information available to assist biomedicine, security, and service towards saving lives, protecting property, and improving quality of life.

# Bibliography

- [1] Y. Bar-Shalom. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.
- [2] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Walton Street, Oxford OX26DP, 1994.
- [3] C. Bishop. Novelty detection and neural network validation. *In IEE proceedings on Vision, Image and Signal Processing. Special Issue on Application of Neural Networks.*, 141(4):217–222, 1994.
- [4] Y. Boykov and D. Huttenlocher. Adaptive bayesian recognition in tracking rigid objects. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 697–704, 2000.
- [5] G. R. Bradski. Computer vision face tracking as a component of a perceptual user interface. *IEEE Workshop on Applications of Computer Vision*, pages 214–219, 1998.
- [6] T. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. volume 8, pages 90–99, 1986.
- [7] Y. Chen, Y. Rui, and T. Huang. Jpdaf-based hmm for real-time contour tracking. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 543–550, 2001.

- [8] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, 2003.
- [9] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bay. Visual categorization with bag of keypoints. *European Conference on Computer Vision*, 2004.
- [10] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. *International Conference on Pattern Recognition.*, pages pp. 886–893, 2005.
- [11] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *European Conference on Computer Vision.*, pages pp. 428–431, 2006.
- [12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2001.
- [13] E. Durucan and T. Ebrahimi. Change detection and background extraction by linear algebra. *In proceedings of IEEE*, 89(10):1368–1381, Oct. 2001.
- [14] J. Efors, A. Berg, G. Morri, and J. Malik. Recognizing action at a distance. In *Intl. Conference on Computer Vision*, 2003.
- [15] A. Elgammal and L. S. Davis. Probabilistic framework for segmenting people under occlusion. *In proceedings of the 8th IEEE International Conference on Computer Vision*, 2001.
- [16] A. Elgammal, R. Duraiswami, and L. S. Davis. Efficient computation of kernel density estimation using fast gauss transform with application for segmentation and tracking. *In proceedings of the 2nd IEEE International Workshop on Statistical and Computational Theories of Vision*, 2001.

- [17] A. Elgammal, R. Duraiswami, and L. S. Davis. Efficient non-parametric adaptive color modeling using fast gauss transform. *In proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
- [18] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *In proceedings of the IEEE*, 90:1151–1163., 2002.
- [19] L. Fei Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE PAMI*, 28(4):pp. 594–611, 2006.
- [20] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *Computer Vision and Pattern Recognition*, pages pp. 264–271, 2003.
- [21] V. Ferrari, T. Tuytelaars, and L. Van Gool. Real-time affine region tracking and coplanar grouping. volume 2, pages 226–233, Kauai, HI, 2001.
- [22] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. *Annual Conference on Uncertainty in Artificial Intelligence*, pages 175–181, 1997.
- [23] G. Gordon, D. Salmond, and A. Smith. A novel approach to non-linear and non-gaussian bayesian state estimation. *Proceedings of IEE*, 140:107–113, 1993.
- [24] M. Greiffenhagen, D. Comaniciu, H. Neimann, and V. Ramesh. Design, analysis and engineering of video monitoring systems: An approach and a case study. *Proceedings of the IEEE*, 89(10):1498–1517, 2001.
- [25] G.D. Hager and P.N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. pages 403–410, San Francisco, CA, Jun 1996.

- [26] U. Handman, T Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. Computer vision for driver assistance systems. *Proceedings of SPIE*, 3364:136–147, 1998.
- [27] I. Haritoglu, D. Harwood, and L. S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:809–830, 2000.
- [28] Y. Hsu, H. Nagel, and G. Rekers. New likelihood test methods for change detection in image sequences. *Computer Vision, Graphics and Image Processing*, 26:73–106, July 1984.
- [29] M. K. Hu. Visual pattern recognition by moment invariants. pages 179–187, 1962.
- [30] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the 4th European Conference on Computer Vision*, pages 343–356, London, UK, 1996.
- [31] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 343–356, London, UK, 1996. Springer-Verlag.
- [32] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal on Computer Vision*, 1(29):5–28, 1998.
- [33] N. Japlpwicz, C. Meyers, and M. Gluck. A novelty detection approach to classification. In *proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 518–523, 1995.

- [34] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. *In proceedings of IEEE Workshop on Motion Video Computing*, pages 22–27, Dec. 2002.
- [35] T. Kadir and A. Zisserman. An affine invariant salient region detector. *European Conference on Computer Vision*, pages pp. 345–457.
- [36] K. P. Karman and A. von Brandt. Moving object recognition using an adaptive background memory. *Time-varying Image Processing and Moving Object Recognition*, 1990.
- [37] K. P. Karman and A. von Brandt. Moving object segmentation based on adaptive reference images. *Signal Processing: Theories and Applications*, 1990.
- [38] R. Kelley, C. King, A. Tavakkoli, M. Nicolescu, M. Nicolescu, and G. Bebis. An architecture for understanding intent using novel hidden markov models. *International Journal of Humanoid Robotics- Special Issue on Cognitive Humanoid Robots*, 5(2):243–264.
- [39] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. Understanding human intent via hidden markov models in autonomous mobile robots. *In Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, 2008.
- [40] V. Kettanker and R. Zabih. Bayesian multi-camera surveillance. *International Conference on Computer Vision and Pattern Recognition*, pages 253–259, 1999.
- [41] K. Kim, D. Harwood, and L. S. Davis. Background updating for visual surveillance. *In proceedings of the International Symposium on Visual Computing*, 1:337–346, December 2005.

- [42] G. Kitagawa. Non-gaussian state-space modeling of nonstationary time series. *Journal of American Statistical Association*, 82:1032–1063, 1987.
- [43] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel. Towards robust automatic traffic scene analysis in real-time. *In proceedings of ICPR*, 1:126–131, October 1994.
- [44] C. Lambert, S. Harrington, C. Harvey, and A. Glodjo. Efficient on-line non-parametric kernel density estimation. *Algorithmica*, (25):37–57, 1999.
- [45] D. Lee. Effective gaussian mixture learning for video background subtraction. *IEEE Transactions on PAMI*, 27(5):827–832, May 2005.
- [46] L. Li, W. Huang, I.Y. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, November 2004.
- [47] D. Lowe. Object recognition from local scale-invariant features. *International Confernece on Computer Vision*, 1999.
- [48] A. Loza, L. Mihaylova, D. Bull, and N. Canagarajah. Structural similarity-based object tracking in multimodality surveillance videos. *Mach. Vision Appl.*, 20(2):71–83, 2009.
- [49] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Computer Vision Journal*, 22(10):pp. 761–767, 2004.
- [50] T. Matsuyama, T. Ohya, and H. Habe. Background subtraction for non-stationary scenes. *In proceedings of Asian Conference on Computer Vision*, pages 662–667, July 2000.

- [51] S.J. McKenna, Y. Raja, and S. Gong. Object tracking using adaptive color mixture models. *In proceedings of Asian Conference on Computer Vision*, 1:615–622, January 1998.
- [52] S.J. McKenna, Y. Raja, and S. Gong. Tracking color objects using adaptive mixture models. *Image and Vision Computing*, 17:223–229, 1999.
- [53] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. *7th European Conference on Computer Vision*, .(.):., . 2002.
- [54] K Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal on Computer Vision*., 60.(1.):pp. 63–86., 2004.
- [55] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. *In proceedings of CVPR*, 2:302–309, July 2004.
- [56] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. *In Proc. Neural Networks in Signal Processing*, 1997.
- [57] N. Paragios and V. Ramesh. A mrf-based approach for real-time subway monitoring. *IEEE Transactions on PAMI*, 1:1030–1040, December 2001.
- [58] J. Platt. *Advances in Kernel Methods - Support Vector Learning*. Editors: B. Schlkopf, C. Burges, A. J. Smola, MIT Press, 1998.
- [59] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*., MIT Press:185–208., 1998.
- [60] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Microsoft Research Technical Report MSR-TR-98-14*, 1998.

- [61] R. Pless, T. Brodsky, and Y. Aloimonos. Detecting independent motion: The statistics of temporal continuity. *IEEE Transactions on PAMI*, 22(8):68–73, 2000.
- [62] R. Pless, J. Larson, S. Siebers, and B. Westover. Evaluation of local models of dynamic backgrounds. *In proceedings of the CVPR*, 2:73–78, June 2003.
- [63] D. Premack and G. Woodruff. Does the chimpanzee have a theory of mind? In *Behavioral and Brain Sciences*, volume 1, pages 515–526, 1978.
- [64] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–285, 1989.
- [65] D. Ramanan, D. Forsyth, and A. Zisserman. Tracking people by learning their appearances. *IEEE PAMI*, 29(1):pp. 65–81, 2007.
- [66] J. Rittscher, J. Kato, S. Joga, and A. Blake. A probabilistic background model for tracking. *In proceedings of 6th European Conference on Computer Vision*, 1843:336–350, 2000.
- [67] R. Rosales and S. Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–123, 1999.
- [68] M. Schulte-Ruther, H. J. Markowitsch, G. R. Fink, and M. Piefke. Mirror neuron and theory of mind mechanisms involved in face-to-face interactions: A functional magnetic resonance imaging approach to empathy. *J. Cognitive Neuroscience*, 19(8):1354–1372, 2007.
- [69] D. W. Scott. *Multivariate Density Estimation*. Wiley Interscience, 1992.

- [70] Y. Sheikh and M. Shah. Bayesian object detection in dynamic scenes. *In proceedings of the CVPR*, 1:74–79, June 2005.
- [71] B. Sholkopf. *Support Vector Learning*. Ph.D. Thesis, Technischen Universitat Berlin, 1997.
- [72] B. Sholkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. *In proceedings of the 1st International Joint Conference on Knowledge Discovery and Data Mining*, pages 252–257, 1995.
- [73] J. Sivic and A. Zisserman. Video google: Efficient visual search of videos. *Toward Category-Level Object Recognition.*, pages pp. 127–144, 2006.
- [74] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *In proceedings of CVPR*, 2:246–252, 1999.
- [75] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on PAMI*, 22(8):747–757, August 2000.
- [76] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J.M. Buhmann. A probabilistic background model for tracking. *In proceedings of ICCV*, pages 294–301, July 2001.
- [77] A. Tavakkoli, R. Kelley, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. A vision-based approach for intent recognition. *In Proceedings of the 3rd International Symposium on Visual Computing*, 2007.
- [78] A. Tavakkoli, R. Kelley, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. A visual tracking framework for intent recognition in videos. *In Proceedings of the 4th International Symposium on Visual Computing*, 2008.

- [79] A. Tavakkoli, M. Nicolescu, and G. Bebis. Automatic robust background modeling using multivariate non-parametric kernel density estimation for visual surveillance. *In proceedings of the International Symposium on Visual Computing*, LNSC 3804:363–370, December 2005.
- [80] A. Tavakkoli, M. Nicolescu, and G. Bebis. An adaptive recursive learning technique for robust foreground object detection. *In proceedings of the International Workshop on Statistical Methods in Multi-image and Video Processing (in conjunction with ECCV06)*, May 2006.
- [81] A. Tavakkoli, M. Nicolescu, and G. Bebis. Automatic statistical object detection for visual surveillance. *In proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 144–148, March 2006.
- [82] A. Tavakkoli, M. Nicolescu, and G. Bebis. A novelty detection approach for foreground region detection in videos with quasi-stationary backgrounds. *In proceedings of the 2nd International Symposium on Visual Computing*, November 2006.
- [83] A. Tavakkoli, M. Nicolescu, and G. Bebis. Robust recursive learning for foreground region detection in videos with quasi-stationary backgrounds. *In proceedings of 18th International Conference on Pattern Recognition*, August 2006.
- [84] A. Tavakkoli, M. Nicolescu, and G. Bebis. A support vector data description approach for background modeling in videos with quasi-stationary backgrounds. *to appear in the International Journal on Artificial Intelligence Tools*, Accepted in December 2007.

- [85] A. Tavakkoli, M. Nicolescu, and G. Bebis. Efficient background modeling through incremental support vector data description. *In Proceedings of the 19th International Conference on Pattern Recognition*, 2008.
- [86] A. Tavakkoli, M. Nicolescu, G. Bebis, and M. Nicolescu. Non-parametric statistical background modeling for efficient foreground region detection. *International Journal of Machine Vision and Applications*, pages 1–16, 2008.
- [87] A. Tavakkoli, M. Nicolescu, M. Nicolescu, and G. Bebis. Incremental svdd training: Improving efficiency of background modeling in videos. *In Proceedings of the 10th IASTED Conference on Signal and Image Processing*, 2008.
- [88] D. Tax and R. Duin. Support vector domain description. *Pattern Recognition Letters*, (20):1191–1199, 1999.
- [89] D. Tax and R. Duin. Data description in subspaces. *In proceedings of the International Conference on Pattern Recognition*, 2:672–675, 2000.
- [90] D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.
- [91] D. Tax and P. Laskov. Online svm learning: from classification and data description and back. *Neural Networks and Signal Processing*, (1):499–508., 2003.
- [92] D.M.J. Tax. Ddtools, the data description toolbox for matlab, Augustus 2005. version 1.11.
- [93] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. *In proceedings of ICCV*, 1:255–261, September 1999.

- [94] T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. *British Machine Vision Conference*, pages 412–425, 2000.
- [95] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *International Journal on Computer Vision.*, 59(1):pp. 61–85., 2004.
- [96] O. Tuzel, F. Porikli, and P. Meer. Learning on lie groups for invariant detection and tracking. pages 1–8, Mineapolis, MN, June 2008.
- [97] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
- [98] S. Wachter and H.-H. Nagel. Tracking persons in monocular image sequences. *Computer Vision and Image Understanding*, 74(3):174–192, 1999.
- [99] C. Wern, A. Azarbayejani, T. Darrel, and A.P. Pentland. Pfinder: real-time tracking of human body. *IEEE Transactions on PAMI*, 19(7):780–785, July 1997.
- [100] L. Wixson. Detecting salient motion by accumulating directionary-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:774–780, August 2000.