

**PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance**

This is to certify that the thesis/dissertation prepared

By Ka Ki Ng

Entitled

Background Subtraction and Object Tracking with Applications in Visual Surveillance


For the degree of Doctor of Philosophy

Is approved by the final examining committee:

- | | |
|---|----------|
| 1. <u></u> | 5. _____ |
| Chair | |
| 2. <u></u> | 6. _____ |
| 3. <u></u> | 7. _____ |
| 4. <u></u> | 8. _____ |

Format Approved by:

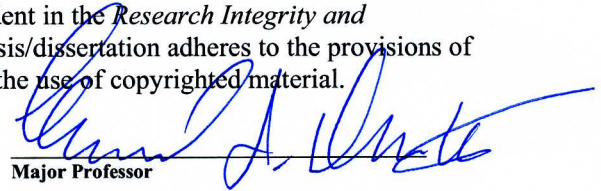
Chair, Final Examining Committee

or 
Department Thesis Format Advisor

is
This thesis is not to be regarded as confidential.


Major Professor

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.


Major Professor

Approved by:  12/6/11
Head of the Graduate Program Date

**PURDUE UNIVERSITY
GRADUATE SCHOOL**

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Background Subtraction and Object Tracking with Applications in Visual Surveillance

For the degree of Doctor of Philosophy

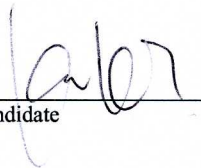
I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22, September 6, 1991, Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Ka Ki Ng

Printed Name and Signature of Candidate



Nov 21, 2011

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

BACKGROUND SUBTRACTION AND OBJECT TRACKING WITH
APPLICATIONS IN VISUAL SURVEILLANCE

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ka Ki Ng

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2011

Purdue University

West Lafayette, Indiana

Dedicated to my parents and my sisters.

ACKNOWLEDGMENTS

This is a great opportunity for me to share my tears of joy with loved ones and to express my immense gratitude to those who helped me on my academic journey. The memory of my first day at Purdue on June 6, 2002 is still very vivid for me. It is hard for me to believe that this long journey has finally come to an end.

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Edward J. Delp. His guidance and support helped me become independent in my scholarly endeavors. I truly appreciate the patience and time he devoted to revising my writings numerous times.

I am extremely grateful to all the members of my dissertation committee: Professor Mary Comer, Professor David Ebert, and Professor Zygmunt Pizlo for their advice and encouragement.

I am also grateful to Purdue University for providing me such an enabling environment for both of my undergraduate and graduate studies.

I would like to thank the U.S. Department of Homeland Security's VACCINE Center for funding this project and the Purdue University Police Department for supplying some of the video sequences used in my experiments.

I would like to thank all the members of VIPER laboratory: Dr. Golnaz Abdollahian, Murat Birinci, Marc Bosch, Dr. Ying Chen, Ye He, Dr. Nitin Khanna, Deen King-Smith, Dr. Liang Liang, Dr. Limin Liu, Kevin S. Lorenz, Ashok Mariappan, Anand Mariappan, Dr. Anthony Frank Martone, Aravind Mikkilineni, Albert Parra, Francisco Serrano, Dr. Satyam Srivastava, Chang (Joy) Xu, Meilin Yang, Bin Zhao, and Fengqing (Maggie) Zhu.

I would like to thank my entire extended family for always being so supportive throughout my graduate studies, especially my grandfather, who passed away while this thesis was being written.

Lastly and most importantly, I wish to thank my immediate family, especially my mother and my father, for their unconditional love and support. They selflessly endured the pain of being separated from their daughter by thousands of miles so that I could achieve my goals. This thesis would not be possible without them standing behind me.

This thesis is based upon work supported by the U.S. Department of Homeland Security's VACCINE Center under Award Number 2009-ST-061-CI0001.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABBREVIATIONS	xi
ABSTRACT	xii
1 INTRODUCTION	1
1.1 An Overview of Video Surveillance Systems	1
1.2 Thesis Contributions	3
1.3 Publications	5
2 VISUAL SURVEILLANCE	8
2.1 Thesis Overview	10
2.2 Literature Survey	14
2.2.1 Moving Object Detection	15
2.2.2 Object Tracking	21
2.2.3 Particle Filtering	23
2.2.4 Integral Image	29
2.2.5 Crowd Analysis	31
3 PROPOSED METHODS	34
3.1 Moving Object Detection	34
3.1.1 Background Model Initialization	35
3.1.2 Adaptive Threshold for Background Subtraction	36
3.1.3 Intensity-Based Foreground/Background Pixel Classification	36
3.1.4 Adaptive Pixel-Wise Background Model Update	37
3.1.5 Statistical-Based Foreground/Background Pixel Classification	41
3.1.6 Fast Implementation Using an Extension of Integral Images	46

	Page
3.2 Object Tracking	47
3.2.1 Object Tracking with Particle Filtering	49
3.2.2 Dynamic Parameter Setting for the Likelihood Model	58
3.2.3 Effect of Subsampling on Color Density Estimation	61
3.2.4 Fast Method For Constructing The Appearance Model	62
3.3 Crowd Analysis	65
3.3.1 Flow Estimation by Pixel Accumulation	66
3.3.2 Crowd Density Level Estimation with Texture Features	69
4 EXPERIMENTAL RESULTS	73
4.1 Moving Object Detection	73
4.1.1 Intensity-Based Background Subtraction	73
4.1.2 Computational Complexity Comparison	76
4.1.3 Statistical-Based Background Subtraction	77
4.1.4 Fast Implementation	80
4.2 Object Tracking	82
4.2.1 Effect of Subsampling on Histogram Construction	82
4.2.2 Experimental Results	84
4.2.3 Implementation on the Nokia N810 Internet Tablet	94
4.3 Crowd Analysis	96
4.3.1 Training and Testing for Crowd Density Estimation	96
4.3.2 Crowd Flow Results	98
5 CONCLUSIONS AND FUTURE WORK	103
5.1 Thesis Contributions	103
5.2 Future Work	104
5.3 Publications	105
LIST OF REFERENCES	108
VITA	117

LIST OF TABLES

Table	Page
4.1 Table of execution time of our method and a GMM-based method . . .	78
4.2 Table of Detection Rate and False Alarm Rate	80
4.3 Execution time in seconds	81
4.4 Similarity between histograms with $\sigma = 0.15$	83
4.5 Testing results for crowd flow estimation.	101

LIST OF FIGURES

Figure	Page
1.1 Locations of surveillance cameras in Chicago. Image from Eyeing Chicago [4]	2
1.2 Surveillance cameras have become an omnipresent reality in our daily lives. We are being monitored in places such as school campus, banks, airports, stores, and streets.	4
2.1 An overview of our proposed method for moving object detection.	12
2.2 A block diagram for particle filtering based object tracking.	13
2.3 An overview of our proposed method for crowd flow estimation.	14
2.4 One iteration of prediction and update of Bayesian recursive filtering. The goal is to find the posterior probability density function at time k	26
2.5 An illustration of how particles evolve in each iteration. Image from [51].	28
2.6 Integral Image $ii(x, y)$ used to determine the rectangle region sum of D .	30
3.1 Background subtraction.	34
3.2 Histogram of pixel values in frame $D_t(x, y)$. $Th_{ad} = 52$ is the adaptive threshold determined using the method described in Section 3.1.2. Consider Equation 3.9, $\sigma_1 = 10$ is empirically determined, and the plot of α_1 as $D_t(x, y)$ increases is shown.	40
3.3 An example of particle weight assignment using visual features. The candidate in red will be assigned a higher weight, and the candidates in yellow will be assigned lower weights.	51
3.4 Examples of edge orientation histogram of people.	53
3.5 Examples of edge orientation histogram of vehicles.	54
3.6 Observation likelihood function with $\epsilon = 0.1$ and $\sigma = 0.15$	59
3.7 The pixels marked by a cross are used to construct the histogram. Left: One half of the pixels are used. Right: One quarter of the pixels are used.	61
3.8 (a) An Epanechnikov kernel. (b) Approximated Epanechnikov Kernel using $N = 4$	63

Figure	Page
3.9 The shaded part represents the approximation error of the kernel function using a staircase function.	64
3.10 The approximation error with increasing N , the kernel function used to generate the figure is (a) Gaussian kernel and (b) Epanechnikov kernel.	65
3.11 An overview of our proposed method for crowd flow estimation.	67
3.12 An example of foreground mask generation by background subtraction.	68
3.13 Four spatial relationships for GLCM.	71
4.1 (a) Current frame (b) Binary foreground mask (c) Histogram of the difference image and the adaptive threshold	75
4.2 The 5 columns show frame 75, 85, 96, 110, 120 of a sequence. Row 1: original frames. Row 2: background models of our proposed method. Row 3: binary foreground masks of our proposed method. Row 4: GMM background models. Row 5: GMM foreground masks.	76
4.3 The 5 columns show frame 113, 115, 126, 130, 139 of a sequence. Row 1: original frames. Row 2: background models of our proposed method. Row 3: binary foreground masks of our proposed method. Row 4: GMM background models. Row 5: GMM foreground masks.	77
4.4 Wallflower dataset light switch sequence. Column 1: background models. Column 2: current frames. Column 3: generated foreground mask using our proposed method.	79
4.5 (a) Background model with average grayscale intensity = 156. (b) Current frame with average grayscale intensity = 55. (c) 3D Plots of H . (d) The pixels detected as foreground are white.	82
4.6 (a) Background model with average grayscale intensity = 157. (b) Current frame with average grayscale intensity = 32. Note that there is a person in front of the door. (c) 3D Plots of H . (d) The pixels detected as foreground are white.	83
4.7 Tracking results of a low-illumination sequence with a person walking, the tracked object is circled: frame number (a)45 (b)55 (c)65 (d)75 (e)85 (f)95.	87
4.8 Vehicle tracking results of a bus; frame number (a)30 (b)40 (c)50 (d)75 (e)102 (f)110.	88
4.9 Tracking results of an indoor sequence with a person walking, the tracked object: frame number (a)20 (b)40 (c)60 (d)80 (e)100 (f)120.	89

Figure	Page
4.10 Object tracking of an outdoor sequence with a person walking; Frame number (a)5 (b)10 (c)15 (d)20 (e)25 (f)30.	90
4.11 CAVIAR dataset with a person walking and raising his/her hand.	91
4.12 An example where the tracking method fails when there is a complete occlusion.	92
4.13 A sequence with multiple objects being tracked.	93
4.14 Left: Histograms of distances and likelihood function with 3 different σ s Right: Histograms of importance weights resulted from different σ s . . .	94
4.15 The Nokia N810 Internet Tablet Computer.	95
4.16 An example of the trip-wire and ROI.	96
4.17 A plot of a five-class texture classification on a test sequence. The “stable” periods are circled.	98
4.18 A block diagram illustrating the training process.	99
4.19 Examples of crowd density estimation on three video sequences. The estimated and true (in parentheses) density levels are (left to right) – Top: 3(3), 1(1), 2(2), Second: 1(1), 3(3), 4(4), Third: 2(2), 3(3), 1(1), Fourth: 1(1), 1(1), 2(2), Bottom: 1(1), 1(1), 1(1).	100

ABBREVIATIONS

BG	Background
BM	Background Model
CCTV	Closed-Circuit Television
FD	Frame Difference
FG	Foreground
GMM	Gaussian Mixture Model
KDE	Kernel Density Estimation
PDF	Probability Density Function
HOG	Histogram of Oriented Gradients
MSE	Mean Square Error
ROI	Region of Interest
SAD	Sum of Absolute Difference

ABSTRACT

Ng, Ka Ki Ph.D., Purdue University, December 2011. Background Subtraction and Object Tracking with Applications in Visual Surveillance. Major Professor: Edward J. Delp.

Visual surveillance has been a very active research topic in the last few years due to its growing importance in security, law enforcement, and military applications. More and more surveillance cameras are installed in security sensitive areas such as banks, train stations, highways, and borders. The massive amount of data involved makes it infeasible to guarantee vigilant monitoring by human operators for long periods of time due to monotony and fatigue. As a result, video feeds are usually archived for forensic purposes in the event suspicious activities take place. In order to assist human operators with identification of important events in videos an “intelligent” visual surveillance system can be used. Such a system requires fast and robust methods for moving object detection, tracking, and event analysis.

In this thesis, we investigate methods for moving object detection, tracking, and event analysis. We consider robustness and computational cost as the major design goals of our work. Our proposed method detects moving objects in indoor and outdoor environments under changing illumination conditions and in the presence of background dynamics. We also present a fast implementation of the method using an extension of integral images. We propose a method for robust tracking with dynamic parameter setting for the likelihood model of particle filtering. In addition, we propose a fast method to construct an appearance model for object tracking using a particle filtering framework. We also present a method for pedestrian “flow” estimation that counts the number of persons passing a detection line (trip wire) or a designated region over a period of time. The method is based on accumulated foreground pixel

count in the trip wire and texture features in an area enclosing the trip wire. The method is designed to be robust to varying pedestrian flow rate (crowdedness).

1. INTRODUCTION

1.1 An Overview of Video Surveillance Systems

The first generation of video surveillance systems (1980's) is the traditional analog closed-circuit television (CCTV) network. In the system, analog video cameras are connected by coaxial cables to surveillance screens for monitoring by human operators or the cameras could be connected to videotape recorders for archiving purposes. The second generation video surveillance (1990's) replaced the videotape recorder with a digital video recorder (DVR) with the data archived on hard drives. More recent systems have network connections so the video data can be stored on servers. The third generation is an IP network system, where the data is continuously being transmitted over the network. There are some excellent survey papers in the literature on video surveillance systems, see [1, 2].

The first CCTV system was developed as far back as the 1940's when it was used by the military in Germany to observe the launch of V2 rockets [2]. CCTV systems were also used by the US military to develop and test atomic weapons, as this allowed one to observe the tests from a safe distance. By the 1970's, CCTV had become common in the commercial world. Video surveillance was used in stores and banks to protect their properties and assets. Today, surveillance cameras are omnipresent, from security sensitive places, such as borders and military bases for mitigating terrorist activities, to private homes for burglary prevention [2].

It has been reported that the United Kingdom has more cameras per person than in any other country in the world [3]. Approximately 1.85 million surveillance cameras are installed in United Kingdom, the figure suggests that there is one CCTV camera for every 32 people [3]. Chicago is considered as the most watched city in the United States. Chicago has approximately 15,000 surveillance cameras throughout the city,

both privately and publicly owned. A study of the use of video surveillance has been conducted by Rajiv Shah [4]. Figure 1.1 shows the locations of surveillance cameras in Chicago from the project. Some typical scenes of video surveillance are also shown in Figure 1.2.



Fig. 1.1. Locations of surveillance cameras in Chicago. Image from Eyeing Chicago [4]

The massive amount of data involved makes it infeasible to guarantee vigilant monitoring by human operators for long periods of time due to monotony and fatigue. As a result, video feeds are usually archived for forensic purposes in the event suspicious activities take place. In order to assist human operators with identification of important events in videos, an “intelligent” visual surveillance system can be used. Such a system requires fast and robust methods for moving object detection, tracking, and event analysis [5, 6]. In this thesis, we propose methods for moving object detection and object tracking with applications in visual surveillance and consider robustness and computational cost as the major design goals of our work.

Moving object detection is the basic step for many video analysis tasks [7]. The performance of this step is particularly significant because subsequent processing of the video data is greatly dependent on this step. Moving object detection aims at extracting moving objects that are of interest in video sequences. The problems

with dynamic environmental conditions make moving object detection very challenging. Some examples of these problems are shadows, sudden or gradual illumination changes, rippling water, and repetitive motion from scene clutter such as waving tree leaves [8]. Commonly used techniques for moving object detection are background subtraction [9] [10], temporal frame differencing [11], and optical flow [12].

The next step in the video analysis is object tracking. This problem can be formulated as a hidden state estimation problem given available observations [13]. Another way to look at object tracking is the creation of temporal correspondence among detected object from frame to frame. Object tracking is an important component of many vision systems [14]. It is used not only for visual surveillance, but also for augmented reality, traffic control, medical imaging, gesture recognition, and video editing [15].

The final step of an “intelligent” visual surveillance system is to analyze the content of the video and to identify important events in a scene. This step provides human operators with high level analyses to assist them with making their decisions more accurately, effectively, and efficiently. Events in a scene can be defined by the behavior of an individual, several persons, or a crowd. Crowd analysis is the analysis of the behavior of a crowd, a typical problem is the number of persons walking pass a region of interest in a time interval. The problem is known as crowd flow estimation. A common application of crowd density estimation is automatic monitoring of the crowd density in public places for security control, such as crowd congestion detection and evacuation detection.

1.2 Thesis Contributions

This thesis describes our methods for background subtraction, object tracking, and crowd flow estimation for applications in visual surveillance. We consider robustness and computational cost as the major design goals of our work. and responding to them in a timely manner. The main contributions of the thesis are as follows.



Fig. 1.2. Surveillance cameras have become an omnipresent reality in our daily lives. We are being monitored in places such as school campus, banks, airports, stores, and streets.

- **Moving Object Detection** - In the area of moving object detection a technique robust to background dynamics using background subtraction with adaptive pixel-wise background model update is described. A foreground-background pixel classification method using adaptive thresholding is presented. Another technique that is robust to sudden illumination changes using an illumination model and a statistical test is presented. We also propose a fast implementation of the method using an extension of integral images.

- **Target Object Tracking** - Once a moving object is detected, the foreground object mask generated is used to initiate object tracking using particle filtering. We propose a method for robust tracking with dynamic parameter setting for the likelihood model of particle filtering. We investigate the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation. In addition, we propose a fast method to construct appearance model of particle filtering for object tracking application.
- **Crowd Flow Estimation** - We present a method for pedestrian “flow” estimation that counts the number of persons passing a detection line (trip wire) or a designated region over a period of time. The method is designed to be robust to varying pedestrian flow rate (crowdedness). We assume different level of crowdedness will result in different level of occlusion. We utilize texture feature to classify different levels of occlusion. Number of foreground pixels estimated from background subtraction is used as a complementary feature to estimate the pedestrian flow.

1.3 Publications

Ka Ki’s publications include:

Journal Articles:

1. **Ka Ki Ng** and Edward J. Delp, “Fast and Robust Object Detection and Tracking for Lightweight Visual Surveillance Systems,” *IEEE Transactions on Circuits and Systems for Video Technology*, in preparation.
2. **Ka Ki Ng** and Edward J. Delp, “Robust Crowd Flow Estimation Using Multiple Visual Features for Scenes with Changing Crowd Densities,” *Journal of Electronic Imaging*, in preparation.

Conference Papers

1. **Ka Ki Ng**, Satyam Srivastava, and Edward J. Delp, "Segmentation With Sudden Illumination Changes Using A Shading Model And A Gaussianity Test," *Proceedings of the International Symposium on Image and Signal Processing and Analysis*, Dubrovnik, Croatia, September 2011.
2. Satyam Srivastava, **Ka Ki Ng**, and Edward J. Delp, "Crowd Flow Estimation Using Multiple Visual Features for Scenes with Changing Crowd Densities," *Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance*, Klagenfurt, Austria, September 2011.
3. Satyam Srivastava, **Ka Ki Ng**, and Edward J. Delp, "Co-Ordinate Mapping and Analysis of Vehicle Trajectory for Anomaly Detection," *Proceedings of the IEEE International Conference on Multimedia and Expo*, Barcelona, Spain, July 2011.
4. Satyam Srivastava, **Ka Ki Ng**, and Edward J. Delp, "Color Correction for Object Tracking Across Multiple Cameras," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011.
5. **Ka Ki Ng**, Priyanka Agarwal, Nathan Mullen, Dzung Du, and Ilya Pollak, "Comparison of Several Covariance Matrix Estimators for Portfolio Optimization," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011.
6. **Ka Ki Ng**, and Edward J. Delp, "Background Subtraction Using A Pixel-Wise Adaptive Learning Rate For Object Tracking Initialization," *Proceedings of the SPIE Conference on Visual Information Processing and Communication*, vol. 7882, San Francisco, CA, January 2011.
7. **Ka Ki Ng**, and Edward J. Delp, "Object Tracking Initialization Using Automatic Moving Object Detection," *Proceedings of the SPIE Conference on Visual*

Information Processing and Communication, vol. 7543, San Jose, CA, January 2010.

8. **Ka Ki Ng**, and Edward J. Delp, “New Models For Real-Time Tracking Using Particle Filtering,” *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, vol. 7257, San Jose, CA, January 2009.
9. Fengqing Zhu, **Ka Ki Ng**, Golnaz Abdollahian, and Edward J. Delp, “Spatial and Temporal Models For Texture-Based Video Coding,” *Proceedings of the SPIE Conference on Video Communications and Image Processing*, vol. 6508, San Jose, CA, January 2007.

2. VISUAL SURVEILLANCE

Visual surveillance systems can be manual, partially autonomous, or fully autonomous [16]. Manual systems require video data to be monitored by human observers. Partially autonomous systems analyze video content automatically but still require human input for some monitoring analysis tasks. Fully autonomous systems are capable of performing surveillance task on different levels. This includes low-level task such as motion detection and high-level task such as event detection. With state-of-the-art techniques, the goal of present visual surveillance systems is to assist human operators with identification of important events in videos and to manage real-time alarms in a proactive manner [16, 17].

Visual surveillance has been a very active research topic in the last few years due to the growing importance in security, law enforcement, and military applications [18]. More surveillance cameras are installed in security sensitive areas such as banks, train stations, highways, and borders. The massive amount of data involved makes it infeasible to guarantee vigilant monitoring by human operators for long periods of time due to monotony and fatigue. As a result, video feeds are usually archived for forensic purposes in the event suspicious activities take place. In order to assist human operators with identification of important events in videos, an “intelligent” visual surveillance system can be used. Such a system requires fast and robust methods for moving object detection, tracking, and event analysis.

Moving object detection is the basic step for further video analysis tasks [19]. The performance of this step is particularly significant because subsequent processing of the video data is greatly dependent on this step. Moving object detection aims at extracting moving objects that are of interest in video sequences [7]. Commonly used techniques for moving object detection are background subtraction [9] [10], temporal frame differencing [11], and optical flow [12]. Background subtraction, in particular, is

a commonly used technique for foreground segmentation in static scenes because it has a very low computational cost [20]. The goal of background subtraction is to “remove” the background in a scene by describing an adequate model of the background, the result is that only “interesting objects” are left in the scene for tracking and further analysis [13]. The problems with dynamic environmental conditions make moving object detection very challenging. Some examples of these are shadows, sudden or gradual illumination changes, and repetitive motion from scene clutter such as waving tree branches and leaves [14].

The next step in the video analysis is object tracking, which can be considered as a hidden state estimation problem given available observations [21–24]. Object tracking is an important component of many vision systems [25]. It is used not only for visual surveillance, but also for augmented reality, traffic control, medical imaging, gesture recognition, and video editing [26]. Another way to look at object tracking is the creation of temporal correspondence among detected object from frame to frame [27].

The final step of an “intelligent” visual surveillance system is to analyze the content of video data and to annotate the data with semantic descriptions. The goal is to assist human operators with identification of important events in videos and to manage real-time alarms in a proactive manner. One popular problem of interest is “How many people are there in a crowd?” This problem is a typical problem of crowd analysis, specifically crowd flow estimation.

A common application on crowd density estimation is automatic monitoring of the crowd density in public places for security control, such as crowd congestion detection and evacuation detection. Other examples of applications include urban planning, resource management, tourist flow estimations, and advertising.

One main challenge of crowd density estimation is occlusion, this problem is inevitable in crowd scenes. Occlusion among people usually cause significant underestimation in the process because human body parts can be self-occluded or occluded by other people in the crowd. We present a method that takes occlusion into account for crowd density estimation and is robust to changing crowd density levels.

2.1 Thesis Overview

In this thesis we developed several methods for applications in visual surveillance:

- Moving object detection
 - Foreground-background pixel classification using adaptive thresholding
 - Adaptive pixel-wise background model update techniques
 - A technique that is robust to sudden illumination changes using a illumination model and a statistical test
 - A fast implementation of the methods using an extension of integral images
- Object tracking
 - Robust tracking with dynamic parameter setting for the likelihood model for particle filtering
 - Investigation of the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation
 - A fast method for construction of the appearance model for particle filtering for object tracking
- Crowd flow estimation
 - Estimation of the level of crowdedness in a scene
 - Estimation of the number of persons passing through a trip wire over a time period

We present a novel and simple method for moving object detection. The method is based on background subtraction. The first step, referred to as *background model initialization*, is to construct a background model using the initial frames. The step is based on temporal frame differencing because the background model is not available at the start of a sequence. A typical example is when a sequence starts with a

moving foreground object, part of the background model will be covered and hence the background model will not be available. Once the initial background model is constructed, with each subsequent frame, we detect if there is any sudden illumination change. If there is no illumination change, simple background subtraction can be used to find the foreground pixels. The threshold used for this process is determined adaptively according to the current frame. After thresholding the difference of the pixel intensities between the background model and the current frame, a foreground object mask is generated. In the presence of sudden illumination change, we use the illumination effect, which is expressed as a ratio of pixel intensities. We determine if the pixel is a foreground or background pixel by the statistics of the illumination effect of its neighboring pixels. A foreground object mask is generated according to this statistic. The process of obtaining the foreground mask is referred to as *foreground-background pixel classification*. After obtaining the foreground object mask, the background model should be updated because the scene dynamics are always evolving. We propose a technique that updates the background model according to two factors: how long a pixel has been a background pixel, and how large the value of the pixel is in the difference frame. This process is referred to as the *background model update*. An overview of the proposed method is shown in Figure 2.1.

Once the moving objects are detected, one can then use the information extracted from the detected moving objects to track their motion as they move. For object tracking, we propose a method that is based on the state-of-the-art object tracking technique known as particle filtering [28]. Particle filtering is a state estimation technique in a recursive Bayesian framework [24]. It can be used for the object tracking problem because the problem can be formulated as a hidden state estimation problem given available observations [24]. To track an object using particle filtering, an appearance model is required to represent the target object [29]. A popular choice for the appearance model is color distribution. Potential candidates (at different locations) in the current frame are evaluated according to how much they resemble the target object. The evaluation is done based on the distance between the two

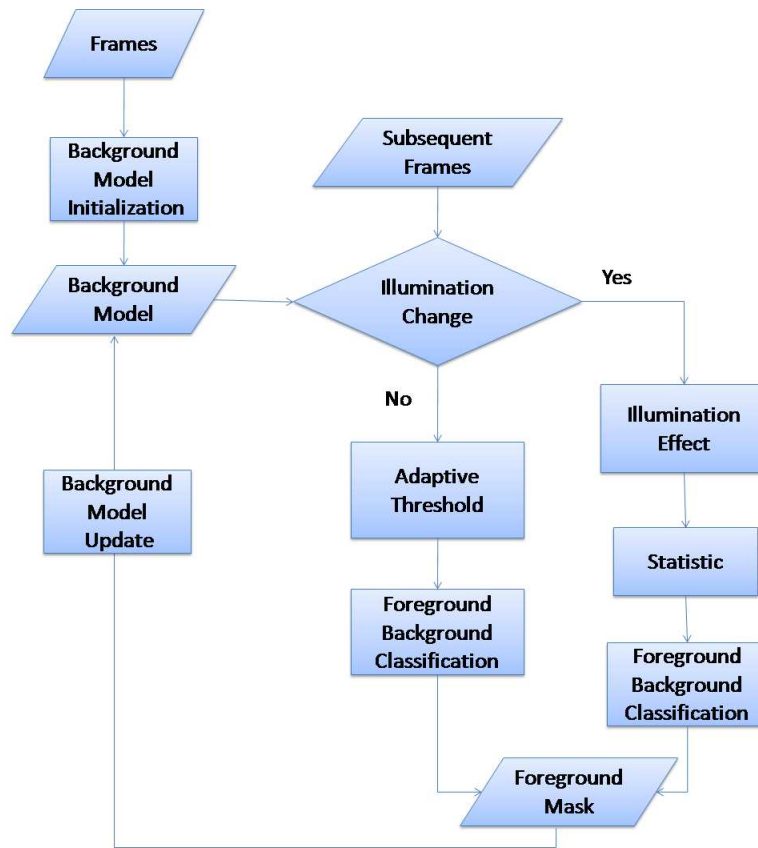


Fig. 2.1. An overview of our proposed method for moving object detection.

appearance models. For example, if color distribution is used, Bhattacharyya distance can be used as a metric to determine how much a candidate resembles the target object [29, 30]. With this distance measure, a likelihood model can be used to assign an importance weight to this candidate. The new state can then be estimated using this set of weighted candidates. We propose a dynamic parameter setting for the likelihood function used in particle filtering for more robust tracking performance.

We investigate the trade-off between object tracking performance and computational cost by subsampling the pixels in an area of interest used for color density estimation. We also propose a fast method to implement color density estimation using an extension of integral histograms. A block diagram that illustrates one iter-

ation of object tracking using particle filtering is shown in Figure 2.2. The blocks in red are the focus of our proposed methods.

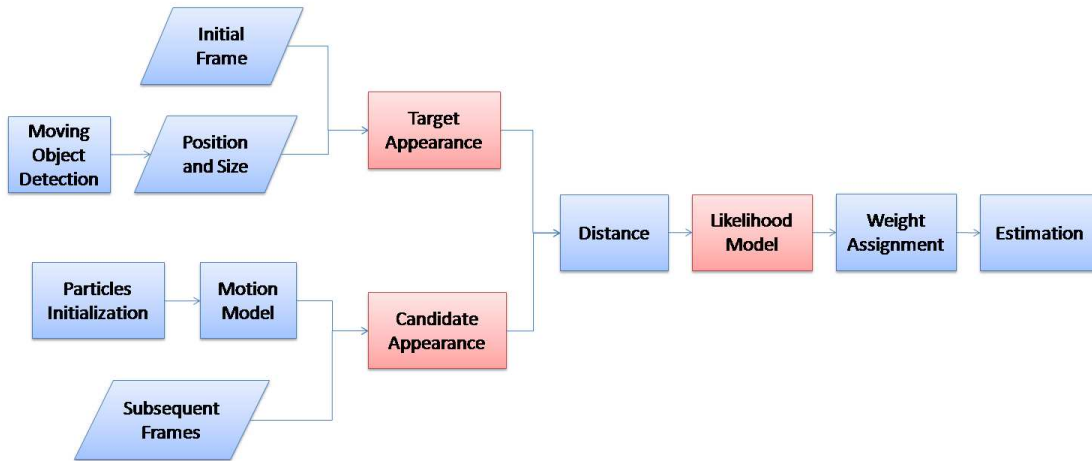


Fig. 2.2. A block diagram for particle filtering based object tracking.

For crowd analysis, we address the problem of accurate estimation of crowd flow using a single view. We propose a feature based method for crowd flow estimation. Most methods proposed in the literature can be categorized into two types: direct and indirect approaches. Direct approach is a detection based method that detects each individual person in a scene using segmentation or human detection. The total number of persons can then be determined easily. Another category known as the indirect approach is a map based method that maps some detected visual features to the number of people.

We propose an indirect method for crowd flow estimation based on two visual features: number of foreground object pixels and a texture measurement known as Gray Level Co-occurrence Matrix (GLCM). Our method is designed to be robust to significant change of crowd density levels. The first feature, foreground pixel count, is used based on the assumption that the number of foreground pixels is proportional to the number of persons. Of course, that case is only true when there are not any serious

occlusions among people. To take different levels of occlusion among individuals into account, we impose texture as a second feature. The choice of this feature is based on the fact that images with low density crowd tend to resemble coarse textures, and images of high density crowd tend to resemble fine textures. Figure 2.3 shows an overview of our crowd flow estimation method.

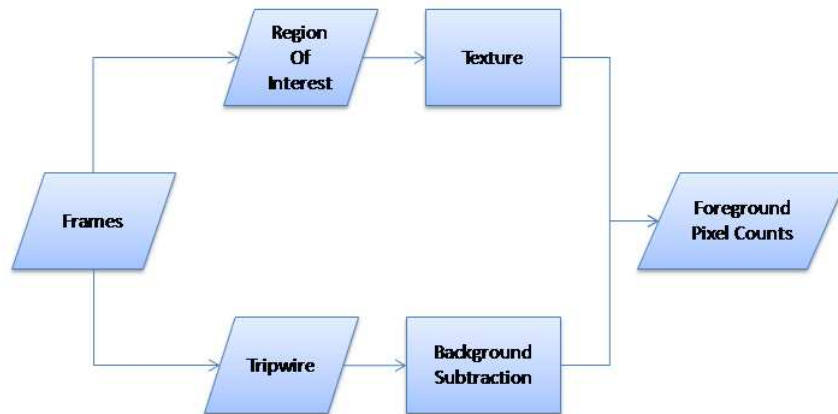


Fig. 2.3. An overview of our proposed method for crowd flow estimation.

2.2 Literature Survey

There are examples of visual surveillance systems in the literature that have been shown to work successfully. Some examples include the one that is developed under a U.S. government-funded program, which is known as Video Surveillance and Monitoring (VSAM) [31]. It is a system designed to detect moving objects, classify the moving objects as vehicles or humans, and track them. The VSAM system is also capable of detecting simple activities and interactions between human subjects. More recently, Shah *et al.* [5] propose a real-time automatic surveillance system known as KNIGHT. The KNIGHT system is capable of object detection and classification,

tracking, and activity recognition using single and multiple camera systems. The system also combines object tracking in different cameras with overlapping and non-overlapping field of views without requiring explicit calibration by human operators. The IBM Smart Surveillance System (S3) [6] detects and tracks moving objects, it also has the capability of identifying the activity of objects in a monitored area. Some examples include license plate recognition and face capture.

Many commercial surveillance systems target sensitive areas and utilize “hot spots” or “trip wires” in a business environment to identify activities of interest. Some of the systems are developed by: Acuity, Agent Vi, Avocado, Axis, AxonX, Cernium, Emza, ioimage, Mate, VideoIQ, Vidient, Vistascape(Siemens).

2.2.1 Moving Object Detection

Moving object detection aims at extracting moving objects that are of interest in video sequences. Some examples of “interesting objects” include people and vehicles, but not waving tree leaves. Many methods for moving object detection have been proposed in the literature, these include background subtraction [9] [10], temporal frame differencing [11], and optical flow [12].

The goal of background subtraction is to “remove” the background in a scene by describing an adequate model of the background. The result is that only “interesting objects” are left in the scene for tracking and further analysis. This technique generally has a low computational cost. However, one drawback is that it is vulnerable to scene dynamics and clutter. The most common methods are the intensity based background subtraction [9, 10], some other background subtraction proposed in the literature are edge based [32], texture based [7], a combination of color and edge based [30], and a combination of color and gradient based [33]. A hierarchical framework for background subtraction has also been proposed in the literature [33–36]. They typically have three distinct levels that construct the hierarchical framework, i.e. pixel level, region level, and frame level. Each level uses a single feature or

multiple features for moving object detection. Another challenge in moving object detection is the shadow interference which results in false detection of the foreground objects. Horprasert *et al.* [30] use a statistical color model (in RGB color space) and the ideas of brightness distortion and color distortion to develop a method that is invariant to illumination changes. In [37,38], Cucchiara *et al.* propose a method that can detect shadows, and the color representation in the HSV color space is used. The shadow regions are defined by a small variation of the hue component, and a diminution of the luminance and saturation components. Cucchiara *et al.* [39] provide an excellent survey on shadow detection.

Another challenging problem is sudden illumination changes due to the presence of photometric distortions. Many methods can address gradual illumination changes but remain susceptible to sudden changes. Some examples of sudden illumination changes are turning on/off light sources in a room for indoor scenes, and moving clouds for outdoor scenes. These situations alter the background model, and cause color or intensity-based subtraction methods to fail (having too many false positives, i.e. detecting background pixels as foreground pixels).

One of the popular approaches, the use of Gaussian Mixture Models (GMM), was proposed by Stauffer and Grimson [10] and is robust to gradual illumination changes as well as moving background regions. However, it fails under sudden illumination changes where by foreground objects can be integrated into the background model if they remain static for a long period of time. The GMM approach also has a relatively high computational cost. There are illumination-invariant foreground segmentation methods proposed in the literature. Vosters *et al.* [40] described the “eigenbackground” method combined with a statistical illumination model to address sudden illumination changes. The eigenbackground model is determined from training data, and the model is used to reconstruct the background image for each current frame which is the testing data. Xie *et al.* [41] suggested that the spatial ordering of pixel intensities is preserved even in the presence of large photometric distortions. They proposed a method using the Phong’s shading model [42], and under the assumption

of locally constant illumination, the sign of the difference between two pixel intensities is robust to sudden illumination changes.

Radke *et al.* [8], McIvor [20], and Piccardi *et al.* [43] provide some excellent surveys of moving object detection. Two commonly used methods (i.e. single Gaussian model and Gaussian mixture model) are described in detail below.

Pfinder - Single Gaussian Model

The single Gaussian method models each pixel of a background scene independently by a single Gaussian distribution. For each pixel in a frame, the model uses the previous n pixel values as samples for a Gaussian probability density function which is parametrized by a mean and a variance. If all of the previous n intensity values of each pixel for every frame have to be stored for the mean and variance computations, the memory requirement of the method would be very high. Wren *et al.* [9] solve the problem by using a running average approach which has a much lower memory requirement.

To accommodate background dynamics, the running average approach for the update of the background model B_t is described as follows:

$$B_t = (1 - \alpha)B_{t-1} + \alpha I_t \quad (2.1)$$

where α is the pre-determined “learning rate” (which will be defined in detail in Section 3.1.4), I_t is the current pixel intensity value and B_{t-1} is the previous running average of the background model.

The other parameter of the Gaussian model, the standard deviation σ , can be updated similarly using the same learning rate α :

$$\sigma_t = (1 - \alpha)\sigma_{t-1} + \alpha(I_t - B_t)^2 \quad (2.2)$$

Using this running average approach, the memory requirement is much lower. Only a mean and a variance are required to be stored for each pixel of the background

model. Simple thresholding is then used to classify if a pixel is a foreground or background pixel, and the results are represented using a binary foreground mask $FG(x, y)$,

$$FG(x, y) = \begin{cases} 1 & \text{if } |I_t - B_t| > k\sigma_t \\ 0 & \text{otherwise} \end{cases} \quad (2.3)$$

where k is a constant and it is set to be 2.5 [43], I_t is the pixel intensity of the current frame, B_t is obtained from Equation 2.1, and σ_t is obtained from Equation 2.2.

The single Gaussian method allows scene dynamics modeled by a single Gaussian distribution as shown in Equation 2.3. The method works under the assumption that the scene dynamics are smaller than the intensity change induced by the foreground object, as a result, pixels that have intensity values with larger deviations from the Gaussian mean are considered as foreground pixels. Although the single Gaussian method is robust to small or gradual variations in the background dynamics of a scene, the method fails when there are large or sudden changes in the background scene.

Gaussian Mixture Models

When the background scene involves large or sudden changes, a single Gaussian model is not adequate, and a multi-model distribution is needed to fully describe the scene dynamics. Stauffer and Grimson [10] proposed a Gaussian mixture model (GMM) to address this problem. GMM allows the representation of a background scene with multi-modal distributions. Here, multi-model distribution means multiple-Gaussian distribution, which essentially means a “multiple-surface” background representation of a pixel.

In GMM, each pixel is modeled parametrically by a mixture of K Gaussians as the intensity of each pixel evolves over time (temporally). The model is parametrized by a mean, a covariance matrix, a *a priori* probability for each of the K components.

Similar to the single Gaussian method, they can be implemented using a running average method. The parameters are updated for each frame and hence the method does not require a large buffer of video frames with high memory requirement.

GMM is a more general approach when compared to the single Gaussian model. For every pixel, each of the K Gaussian distributions corresponds to the probability of observing a particular intensity. The probabilities are then evaluated to determine which are most likely to be resulted from the background scene, and which are most likely to be resulted from the foreground objects [10].

The probability of observing the pixel intensity I_t in the current frame is:

$$P(I_t) = \sum_{i=1}^K \omega_{i,t} \eta(I_t, \mu_{i,t}, \Sigma_{i,t}) \quad (2.4)$$

where K is number of distributions, it is chosen to be usually $3 \dots 5$, $\omega_{i,t}$ is an estimate *a priori* probability (what portion of the data is accounted for by this Gaussian) of the i^{th} Gaussian at time t , $\eta(I_t, \mu_{i,t}, \Sigma_{i,t})$ is the i^{th} Gaussian model, with mean μ and covariance matrix Σ of the pixel intensities. For computational simplicity, the covariance matrix is assumed to be diagonal so that the inverse can be determined easily.

The distribution of recently observed values of each pixel in the scene is characterized by a mixture of Gaussians. A new pixel in the current frame is represent by one of the K components of the mixture model.

To update the background model, each new pixel in the current frame, is checked against the existing K Gaussian distributions, until a “match” is found. A match is defined as as a pixel value within 2.5 standard deviations of a distribution (out of the K components), and this matched component can be a background component or a foreground component which will be verified later. After determining the matched component, this component in the Gaussian model will be updated as follows:

$$\omega_{i,t} = (1 - \alpha)\omega_{i,t-1} + \alpha \quad (2.5)$$

$$\mu_{i,t} = (1 - \rho)\mu_{i,t-1} + \rho I_t \quad (2.6)$$

$$\sigma_{i,t}^2 = (1 - \rho)\sigma_{i,t-1}^2 + \rho(I_t - \mu_{i,t})^T(I_t - \mu_{i,t}) \quad (2.7)$$

where α is a pre-determined learning rate, and

$$\rho = \alpha \frac{P(i|I_t, \mu_{i,t}, \Sigma_{i,t})}{\omega_{i,t}} \quad (2.8)$$

with α defined earlier, is a learning rate for parameters update. Suggested by Power and Schoonees [44], an appropriate choice to approximate ρ would be:

$$\rho = \begin{cases} \frac{\alpha}{\omega_{i,t}} & \text{if the pixel matches the } i^{th} \text{ component} \\ 0 & \text{otherwise} \end{cases} \quad (2.9)$$

For the remaining components that I_t does not match, the μ and σ parameters of unmatched distributions remain the same while their *a priori* probability is reduced.

$$\omega_{i,t} = (1 - \alpha)\omega_{i,t-1} \quad (2.10)$$

$$\mu_{i,t} = \mu_{i,t-1} \quad (2.11)$$

$$\sigma_{i,t}^2 = \sigma_{i,t-1}^2 \quad (2.12)$$

If the pixel I_t does not match any of the K components, then the least likely (i.e. the component with the smallest $\frac{\omega_{i,t-1}}{\sigma_{i,t-1}}$) component is replaced with a new one which has $\mu_{i,t} = I_t$, large $\Sigma_{i,t}$, and low $\omega_{i,t}$. After the parameters of all Gaussian components have been updated, the $\omega_{i,t}$ are normalized so they sum up to 1:

$$\sum_{i=1}^K \omega_{i,t}$$

At this point, even though the pixel is classified as the component that it belongs to, whether or not it is a foreground or a background pixel is still undetermined. That means the new pixel that belongs to the i^{th} component can be a background or

foreground pixel, and it depends on if the i^{th} component is a background component or a foreground component. The information regarding which of the K components that the pixel belongs to is used to update the mixture model (details will be discussed below). For example, if $K = 3$, then 1 of the 3 components can be a foreground component, and the remaining 2 components are background components. But we can not determine (yet) which component is the foreground and which component is the background. It should be noted that they the background model is updated *before* the foreground pixel is detected.

In order to determine if the pixel I_t is foreground or background, we first need to determine the component that it belongs to is a background component or a foreground component. All components in the mixture are sorted in the order of decreasing $\frac{\omega}{\sigma}$. So higher importance gets placed on components with the most evidence and lowest variance, which are assumed to be the background. Let

$$B = \operatorname{argmin}_b \left(\sum_{i=1}^b \omega_{i,t} > T_B \right) \quad (2.13)$$

for some threshold T_B , the components $1 \dots B$ are assumed to be background, and the remaining components $B + 1 \dots K$ are assumed to be foreground. So if I_t does not match one of these components, the pixel is marked as foreground.

2.2.2 Object Tracking

Over the last few decades, many techniques have been described for object tracking in the literature. Visual object tracking is an important step in many applications such as human-computer interaction, medical imaging, video analytics, augmented reality, and gesture recognition [26]. Object tracking problems can be formulated as a hidden state estimation problem given available observations. One of the best known methods is the Kalman filter [45], it is an optimal recursive Bayesian estimator under linear and Gaussian assumptions. Non-linear variations of the Kalman filter include the extended Kalman filter [46] and the unscented Kalman filter [47]. More recently,

Kanade-Lucas-Tomasi (KLT) [48], mean-shift [49], and particle filtering based [50] tracking methods have been proposed and have been shown to be very successful for object tracking problems [14].

Object tracking methods can be classified into two categories [29]. The first one is deterministic methods that compares a target model with the current frame, searching for the most likely (probable) region, and selects it as the target. Two examples in this category are mean-shift tracking [49] and KLT tracking [48]. The second category is probabilistic methods which use the state space to model the underlying dynamics of the tracking process. An example of this method is particle filtering [24, 28, 51, 52].

In recent years, tracking methods based on particle filtering have been extensively studied [21–24]. Experimental results show that particle filtering outperforms other conventional state estimation methods such as Kalman filtering for non-linear and non-Gaussian systems [21–24]. Particle filters generalize the traditional Kalman filtering methods which only provide the optimal solution when the models have linear functions and Gaussian noise [24, 53]. Particle filtering will be described in more detail in the next section. To solve object tracking problem in a particle filtering framework, an appearance model is required to represent the target object visually. One of the most popular appearance models used for object tracking is color distribution because it has been shown to be robust to partial occlusion, scaling, and object deformation [29, 54]. The color distribution of a target object can be estimated by kernel density estimation which is a non-parametric method for probability density function estimation [26, 55].

Besides color features [56], contour [28], gradient direction [57], texture [58], or a combination of them (e.g. color and contour [59, 60]) have also been proposed for object tracking. In some cases, a single feature does not provide enough information about the object being tracked and multiple features can be combined for tracking to provide more information about the object [61]. Higher level feature descriptors can also be used for object tracking. Some examples of successful descriptors for visual

tracking include covariance matrix [62, 63] and Spatial-color Mixture of Gaussians (SMOG) [64]. A comprehensive survey on object tracking is provided by Yilmaz [14].

2.2.3 Particle Filtering

Filtering

The “filtering” problem is the process of estimating a system’s current state which is hidden, based on past and current observations [65]. This is represented by the probability density function $p(x_t|x_{t-1}, z_{0:t})$. Depending on the application, the state at time t , x_t and observation at time t , z_t can be different entities. For the case of visual tracking, the state can be position, velocity, orientation, or scale of an object, and the observation can be the color histogram, edge orientation histogram, or contours of the image data.

In general, the object tracking problem can be modeled by the state-space representation:

1. State Equation:

$$x_t = f_t(x_{t-1}, v_t) \quad (2.14)$$

2. Measurement Equation:

$$z_t = h_t(x_t, n_t) \quad (2.15)$$

where f_t is the function that describes how the state evolves (also known as transition function or dynamic function) at time t , and h_t is the function that describes the measurement of the state (also known as observation function) at time t . The functions are non-linear and time varying in general. The system noise and the measurement noise are denoted by v_t and n_t respectively.

Recursive Filtering

A recursive filter is a repeated application of a two-stage procedure [22] consisting of prediction and update. The first stage is the current probability density function being propagated to the future to produce the prediction density and is described by the system model. The second stage is finding the hidden state based on the only the observation can be described by the measurement model. If the recursive filter is Bayesian, the update step is performed using Bayes' rule.

Bayesian Recursive Filtering

The optimal estimator \hat{x}_t (optimal in the minimum variance sense) of the state x_t is provided by the conditional expectation $E[x_t|z_{1:t}]$. In order to determine $E[x_t|z_{1:t}]$, we need to estimate the posterior probability density function $p(x_t|z_{1:t})$ at each time instant t , and this estimation can be done by using Bayesian filtering which is a recursive filtering technique in a Bayesian framework.

The goal of Bayesian filtering is to construct the posterior probability density function $p(x_t|z_{1:t})$, of the state x_t at time t given all available observations $z_{1:t}$ up to time t (all available information, including the current observation). It is assumed that the initial probability density function of the state (prior probability density function) $p(x_0)$ is known. Then the posterior probability density function $p(x_t|z_{1:t})$ can be obtained recursively with a two-stage process: prediction and update. The process is described as follows:

- Prediction: Assume that the posterior probability density function $p(x_{t-1}|z_{1:t-1})$ is available, the prior probability density function $p(x_t|z_{1:t-1})$ can be determined by the prediction step using the Chapman-Kolmogorov integral equation:

$$p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1} \quad (2.16)$$

where x_t is the state at time t and $z_{1:t-1}$ is the observation sequence from time 1 to time $t - 1$. The probability density function $p(x_t|x_{t-1})$ expresses the state model (transition model).

- Update: Since the state is usually subject to unknown disturbances, the prediction step generally translates, deforms, and spreads the state probability density function. The update stage uses the latest observation of the current state to modify the prediction probability density function. At time t , the new observation z_t is available, and the prior probability density function $p(x_t|z_{1:t-1})$ is updated according to Bayes' rule:

$$p(x_t|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{\int p(z_t|x_t)p(x_t|z_{1:t-1})dx_t} \quad (2.17)$$

where x_t is the state at time t and $z_{1:t}$ is the observation sequence from time 1 to time t . The probability density function $p(z_t|x_t)$ expresses the observation likelihood function that describes the measurement model.

One iteration of this recursive process is shown in Figure 2.4. In general, Equation 2.16 and Equation 2.17 can not be evaluated in closed form. In some special cases, such as linear and Gaussian state-space models, which can be analyzed using a Kalman filter, closed form solutions are available. To solve this Bayesian filtering problem without the assumptions on the transition and observation models being non-linear and non-Gaussian, particle filtering is used to approximate the closed-form solution. Particle filter generalizes the traditional Kalman filtering methods which solves for the optimal solution only when the models have linear functions and Gaussian noise [24].

Particle Filtering

The Particle filter, also known as the condensation algorithm, bootstrap filter, Monte Carlo filter, sequential importance sampling, interacting particle approxima-

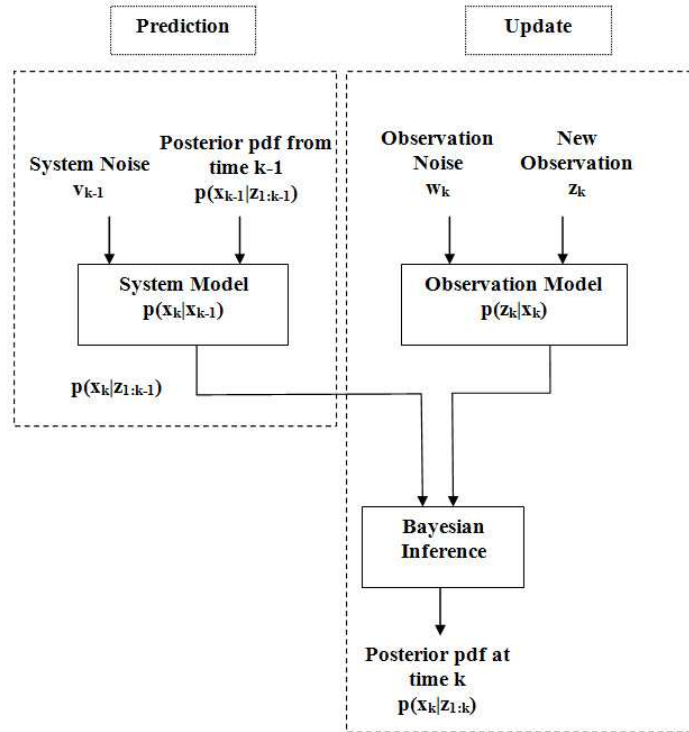


Fig. 2.4. One iteration of prediction and update of Bayesian recursive filtering. The goal is to find the posterior probability density function at time k .

tions, and survival of the fittest [21,24,52,66] is a sequential Monte Carlo method for state estimation problems within a Bayesian framework.

The key idea of particle filtering is to represent the posterior probability density function by a set of discrete samples known as particles. A sample is also referred to as a particle because of its discrete nature and its discrete representation by the probability density function. Each particle represents a hypothesis of the state and is randomly drawn from the prior density. After a particle is drawn, it is then propagated according to the transition model. Each propagated particle is verified by a weight assignment using the likelihood model. The weight characterizes the likelihood of a specific particle being the real state. A large weight will be assigned to a good particle, and a small weight will be assigned to a bad particle.

The posterior probability density function is constructed recursively by the set of weighted random samples $\{x_t^{(i)}, w_t^{(i)}; i = 1, \dots, N\}$, where N is the total number of particles. At each time t , the particle filtering method repeats a two-stage procedure: prediction and update:

- Prediction: Each particle $x_t^{(i)}$ evolves independently according to the state model, including the addition of random noise in order to simulate the unknown disturbance. This prediction step yields an *approximation* of the prior probability density function:

$$p(x_t) \approx \frac{1}{N} \sum_{i=1}^N \delta(x_t - x_t^{(i)}) \quad (2.18)$$

- Update: Each particle's weight is evaluated based on the latest measurement according to the measurement model (likelihood model). The posterior probability density function at time t in the form of a discrete approximation can be expressed as

$$p(x_t | z_{1:t}) \approx \sum_{i=1}^N w_t^{(i)} \delta(x_t - x_t^{(i)}) \quad (2.19)$$

where

$$w_t^{(i)} = \frac{\mathcal{L}(z_t | x_t^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)})}{q(x_t^{(i)} | x_{t-1}, z_t)} \quad (2.20)$$

where $q(\cdot)$ is the importance density, $\mathcal{L}(\cdot)$ is the observation likelihood function, and the set of importance weights satisfies

$$\sum_{i=1}^N w_t^{(i)} = 1. \quad (2.21)$$

An illustration showing how a set of particles and their corresponding weights evolve is shown in Figure 2.5. In each iteration, the set of unweighted particles is assigned a set of weights according to a likelihood model and the estimation is finished. The weighted particles are then resampled, i.e. a particle with a larger weight is “split” into many unweighted particles, and a particle with a smaller weight

is “split” into fewer number of particles. Then each of them is evolved (propagated) according to the state equation.

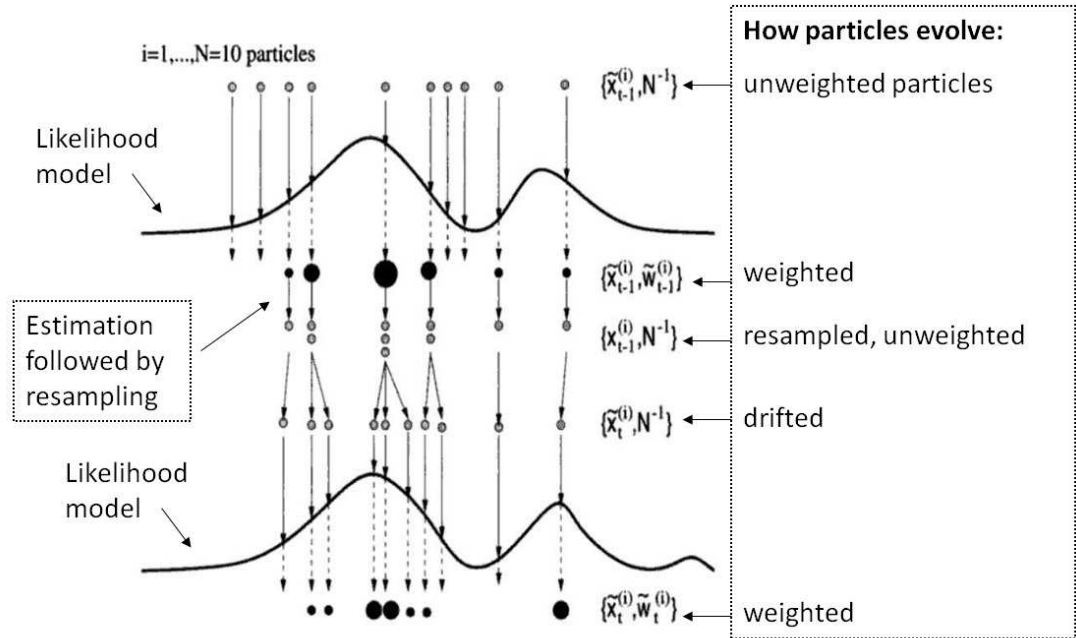


Fig. 2.5. An illustration of how particles evolve in each iteration. Image from [51].

Particle filter can solve non-linear and non-Gaussian state estimation problems and manage multi-model density function effectively [24]. It is a Bayesian sequential importance technique. The hidden state at time t denoted by X_t and observation at time t , Y_t . It estimates the posterior density, $p(X_t|Y_{1:t})$ recursively using a finite set of N weighted samples, $\{s_t^{(i)}, w_t^{(i)}; i = 1, \dots, N\}$. Each $s_t^{(i)}$ represents a hypothetical state at time t and it has a corresponding weight $w_t^{(i)}$ which is determined by the likelihood observation model.

2.2.4 Integral Image

Integral images were first proposed by Crow [67] for texture mapping in computer graphics. He called his method summed area tables. The method was then introduced to the image processing community by Viola and Jones [68]. They called their technique integral images and used it for fast computation of Haar wavelet coefficients for face detection. Integral images have been used for fast computation for various image analysis applications. Porikli [69] used this idea for feature histograms known as integral histograms. Feature histograms have been used for content based image retrieval [70], object segmentation [70] and object tracking [49]. Tuzel *et al.* [62] proposed region covariance [62] as a visual feature descriptor using integral images. Huang *et al.* [71] use integral images for fast implementation of eye localization in an automatic face recognition system. Zhu *et al.* [72] also used this idea for fast computation of the histogram of oriented gradients. The SURF descriptor proposed by Bay *et al.* [73] is based on sums of approximated 2D Haar wavelet coefficients using integral images. Instead of the typical rectangular sum, Lienhard and Maydt [74] propose an adoption of the integral image representation for 45 degree rotated rectangular features. Their method yields an area sum of a 45 degree rotated rectangular region. A three dimensional generalization, named integral video, is proposed by Ke *et al.* [75].

The integral image is an intermediate representation for the fast computation of the “sums of values” in a rectangular region of an image. We refer to this sum as the rectangular region sum. The original definition in [68] of an integral image $ii(x, y)$ at pixel location (x, y) is the sum of all pixels to the left and above the pixel (x, y) (inclusive) of the original image $i(x, y)$.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (2.22)$$

where $i(x, y)$ is the original image. This uses a single-pass recursive approach. The row cumulative sum is defined as,

$$s(x, y) = s(x, y - 1) + i(x, y) \quad (2.23)$$

The integral image can then be described as:

$$ii(x, y) = ii(x - 1, y) + s(x, y) \quad (2.24)$$

We can use $ii(x, y)$ to represent the rectangular region sum $S(x_1 : x_2, y_1 : y_2)$ bounded by four pixels (x_1, y_1) , (x_2, y_1) , (x_1, y_2) and (x_2, y_2) as:

$$S(x_1 : x_2, y_1 : y_2) = ii(x_2, y_2) - ii(x_2, y_1 - 1) - ii(x_1 - 1, y_2) + ii(x_1 - 1, y_1 - 1) \quad (2.25)$$

The intuition of Equation 2.25 is illustrated in Figure 2.6. The value of the integral image at (x_1, y_1) is the sum of all pixel values in region A; at (x_2, y_1) it is region A+B, at (x_1, y_2) it is region A+B+C, and at (x_2, y_2) it is region A+B+C+D. We use Equation 2.25 to obtain the sum of region D in the figure.

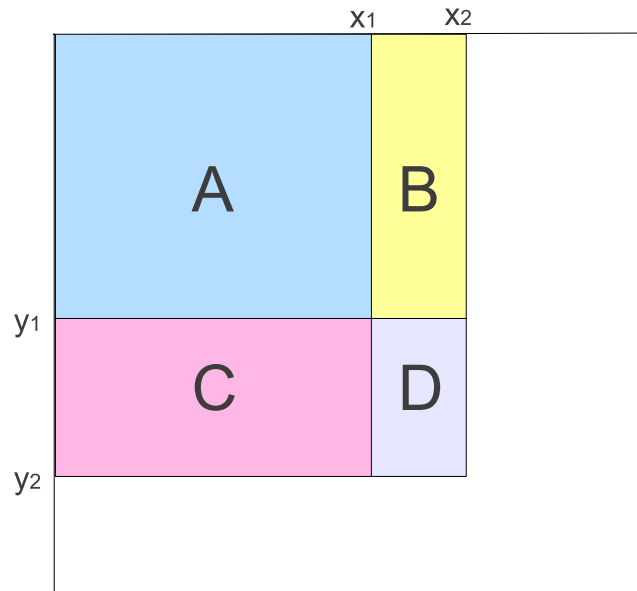


Fig. 2.6. Integral Image $ii(x, y)$ used to determine the rectangle region sum of D.

Computational Cost of Integral Image Construction Only two additions are required to determine a pixel of an integral image $ii(x, y)$ (except those pixels in the first row and first column). Hence for the entire $W \times H$ integral image, the computational cost is approximately $2WH$ additions using Equation 2.23 and 2.24. Compared to the naive way to determine a pixel (x', y') of an integral image $ii(x', y')$ which requires $(x'y') - 1$ additions. Hence, for the entire image, the total number of additions required is:

$$\sum_{x=1}^W \sum_{y=1}^H (xy - 1) \approx \frac{1}{4}W^2H^2,$$

which grows quadratically with the image size.

Computational Cost of Finding an Area Sum According to Equation 2.25, only three additions with four array references are required, regardless of the size of the region.

2.2.5 Crowd Analysis

Crowd monitoring and behavior understanding using visual methods is important in many surveillance applications. A number of methods for crowd analysis have been proposed in the literature [76]. Crowd density estimation is one of the most interesting problems in crowd analysis. A common application on crowd density estimation is automatic monitoring of the crowd density in public places for security control such as crowd congestion detection and evacuation detection. Other examples of applications include urban planning, resource management, tourist flow estimations, and advertising. Most methods proposed in the literature can be categorized into two types: direct and indirect approaches. Direct approach is a detection based method that detects each individual person in a scene using segmentation or human detection. The total number of persons can then be determined easily. Some of these methods involve explicit detection, tracking, and monitoring of individuals in the scene such as the use of histogram of oriented gradients (HOG) features for person

detection [77]. Another category known as the indirect approach is a map based approach that maps some detected visual features to the number of people. Marana *et al.* [78] show that different crowd density levels can be represented by different texture features. For example, images of low density crowds tend to resemble coarse textures; likewise, images of high density crowds tend to resemble fine textures. Ma *et al.* [79] determine the crowd density based on a linear relationship between the number of foreground pixels and number of persons. The method takes into account geometrical distortion from the ground plane to the image plane. It is assumed that the number of foreground pixels is proportional to the number of persons, which is only true when there are not serious occlusions between people. Kong *et al.* [80] use edge orientation and blob size histograms as feature, the relationship between the features and the number of people is related using a linear model. Normalization of the features is done by taking into account viewing geometry in the scene which makes the method viewpoint invariant. Kim *et al.* [81, 82] use number of foreground pixels and motion vectors to find the number of people passing through a gate. Feature normalization is used to make the method viewpoint invariant and robust to different moving speeds of pedestrians. Chan *et al.* [83] segment crowds with dynamic texture motion models and estimate pedestrian count using several geometric, edge, and texture features. They also point out that such indirect methods are better at preserving the privacy of observed individuals.

One of the biggest challenges of crowd density estimation is occlusion, and this problem is inevitable in crowd scenes. Occlusion among people usually cause significant underestimation in the process because human body parts can be self-occluded or occluded by other people in the crowd. Other challenges include perspective distortion. The problem comes from the optical equipment of the cameras, especially those with wide angle views. Images take with these cameras make people far from the camera appear smaller while the people closer to the cameras appear bigger. Some feature based methods can suffer more significantly from perspective distortion. For example if number of foreground pixels is used as a feature to estimate crowd density,

one person can be represented by 50 foreground pixels when he/she is close to the camera while the same person can be represented by 10 foreground pixels if he/she is far from the camera. Hence perspective distortion can deteriorate the accuracy of a method significantly. Another note is that it is highly desirable that a method can work with uncalibrated cameras with simple training processes.

There are some excellent surveys on behavior analysis in the literature, for example, recognition of human activity and motion is studied extensively by Aggarwal *et al.* [84, 85], and Turaga *et al.* [86].

3. PROPOSED METHODS

3.1 Moving Object Detection

Background subtraction is a widely used for detecting moving objects. The ultimate goal is to “subtract” the background pixels in a scene leaving only the foreground objects of interest. If one has a model of how the background pixels behave the “subtraction” process is very simple. Of course, in real world scenes, the situation will be more challenging.

Background subtraction usually consists of three attributes besides the basic structure of the background model: background initialization (construction of the initial background model), background maintenance (updating the background model to account for temporal changes in subsequent frames), and foreground/background pixel classification. The process is shown in Figure 3.1.

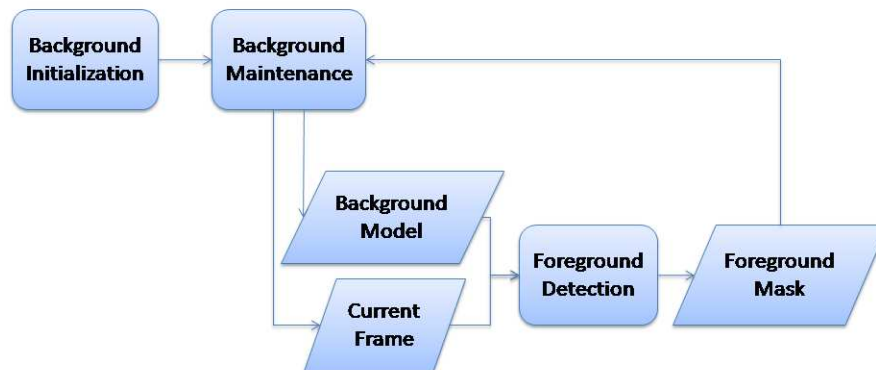


Fig. 3.1. Background subtraction.

3.1.1 Background Model Initialization

In order to avoid the assumption of a sequence starting in the absence of foreground objects, simple temporal frame differencing is used for the initial phase of background model initialization until the background pixels are “stable” (which will be defined later in this section). The temporal frame difference $FD_t(x, y)$ at time t is defined as:

$$FD_t(x, y) = |I_t(x, y) - I_{t-1}(x, y)| \quad (3.1)$$

where $I_t(x, y)$ is the intensity of pixel (x, y) in the frame at time t .

The foreground binary mask $FG_t(x, y)$ is determined by comparing $FD_t(x, y)$ to a threshold $T1$ which is empirically determined and set to be 20, a pixel is considered as having significant motion, and labeled as a foreground pixel, if the difference is greater than a threshold,

$$FG_t(x, y) = \begin{cases} 1 & \text{if } FD_t(x, y) > T1 \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

A pixel is considered as a “stable” background pixel if there is no significant motion detected (i.e. $FD_t(x, y) < T1$) for a certain number of frames (denoted by T_{fr}). Consider a frame count C_{fr} that is incremented by 1 each time $FD_t(x, y) < T1$, when this frame count $C_{fr} > T_{fr}$ (the consecutive background frame count T_{fr} is empirically determined and set to be 100), we can use this pixel in the current frame to construct the background model:

$$BM_t(x, y) = \begin{cases} I_t(x, y) & \text{if } C_{fr} > T_{fr}, \\ 0 & \text{otherwise.} \end{cases} \quad (3.3)$$

As soon as background information is available for a pixel (i.e. $BM_t(x, y)$ is constructed using Equation 3.3), the foreground mask is determined using background differencing instead of temporal frame differencing. The background difference frame (or image) is defined as follows:

$$D_t(x, y) = |I_t(x, y) - BM_{t-1}(x, y)| \quad (3.4)$$

The foreground binary mask is determined by comparing the background difference frame to a threshold $T2$:

$$FG_t(x, y) = \begin{cases} 1 & \text{if } D_t(x, y) > T2, \\ 0 & \text{otherwise.} \end{cases} \quad (3.5)$$

The threshold $T2$ is empirically determined and set to be 50 in our experiments. This background model initialization method assumes every pixel of the background will be uncovered at some time.

3.1.2 Adaptive Threshold for Background Subtraction

To obtain the adaptive threshold discussed above, an iterative technique proposed by Rilder and Calvar [87] is used. The histogram of the difference image $D_t(x, y)$ is segmented into two parts. The initial threshold is set to be the middle value of the range of intensities (for example, the middle value would be 127 for an 8-bit grayscale image), and then the image is thresholded with this middle value. For each iteration, the sample means of the intensities associated with the foreground pixels and the sample means of the intensities associated with the background pixels are determined. Then, a new threshold value is determined as the average of the two sample means. The iteration stops when the threshold stops changing.

This method for finding the threshold will be used in Equation 3.6 which is background subtraction for foreground/background pixel classification, the threshold is denoted as $Th_{Ad,t}$ in Equation 3.6.

3.1.3 Intensity-Based Foreground/Background Pixel Classification

After initializing the background model using the method described in Section 3.1.1, the next step is to use background subtraction for foreground/background pixel classi-

fication using the current frame and the background model. Background subtraction was described above in Equation 3.4.

A foreground mask representing regions of significant motion in the scene is produced by the threshold, $Th_{Ad,t}$. This threshold is determined adaptively for each frame using the current difference image, $D_t(x, y)$, using the method described previously in Section 3.1.2. If the difference between two pixel values is greater than the threshold, the movement of that pixel is considered as significant. A foreground binary mask $FG_t(x, y)$ is generated according to:

$$FG_t(x, y) = \begin{cases} 1 & \text{if } D_t(x, y) \geq Th_{Ad,t} \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

This mask indicates where moving pixels are located. This process is the same as the background subtraction described in Equation 3.5, except that above one assumes the entire background model frame is available, hence only background subtraction is used for every pixel in a subsequent frame. Also an adaptive threshold $Th_{Ad,t}$ is used instead of a pre-determined threshold $T2$.

3.1.4 Adaptive Pixel-Wise Background Model Update

After a background model is constructed using the initial frames, the background model evolves because of the existence of background dynamics. Hence a background model must be updated every frame in order to accommodate for background dynamics. For example, we have a video sequence of sunset, the initial background model frame that is constructed before sunset would be a lot brighter than the actual background after sunset. If the background model is not updated, there would be many false detections because of the intensity change. So the next question is, how “fast” do we want to update the background model? One can think of a simple solution of using the current frame as the background model frame. This method is the fastest way it can possibly be used to update the background model because the newest current frame is used. Consider the example of the sunset sequence again, one can use

the frame after sunset as a background model. It would work perfectly fine if there is no foreground object in the new after-sunset frame, however, if there is a foreground object, and this fast background update method is used, the foreground object would not be detected because it is “absorbed” into part of the background resulted from this fast update. One can update the background model in a more “gentle” manner (suggested by Stauffer and Grimson [10]), consider the equation:

$$BM_t(x, y) = \alpha I_t(x, y) + (1 - \alpha) BM_{t-1}(x, y),$$

The background model is updated using $\alpha\%$ of the intensity in the new current frame, and $(100 - \alpha)\%$ of the intensity from the previous background model. This α is referred to as the “learning rate” for the background model update [10]. One can observe the importance of the value of α and how it affects the performance on background subtraction. Consider another case when the change of background dynamics becomes smaller (for example the waving tree leaves move slower), this is also referred to as “background model is converging” [88]. The learning rate for the tree leaves pixels in this case can decrease to avoid absorbing foreground objects into the background model and results in missed detections. An example like this would require different learning rates in different regions of the background.

The background subtraction method proposed by Stauffer and Grimson [10] use the above background updating method with a constant learning rate for the entire sequence. In [88], an update method using two learning rates for the background update is described, one learning rate in the beginning of the sequence (before the L^{th} frame is available) and a different rate for the rest of the sequence (when the L^{th} frame is available) using a Gaussian mixture model (GMM). Using this method, the GMM learns faster and more accurately at the beginning, however it does not improve the convergence rate of the background model if the change of background dynamics becomes smaller.

We propose an effective scheme for updating the background and adaptively model dynamic scenes. Unlike the traditional methods that use the same learning rate for the entire frame or sequence, our method assigns a learning rate for each pixel in a

frame using two parameters. The first parameter depends on the difference between the pixel intensities of the background model and the current frame. The second parameter depends on the duration of the pixel being classified as a background pixel.

Each pixel in the frame has an adaptive learning rate, $\alpha_{ad,t}(x, y)$. The background mode is updated according to:

$$BM_t(x, y) = \alpha_{ad,t}(x, y)I_t(x, y) + (1 - \alpha_{ad,t}(x, y))BM_{t-1}(x, y), \quad (3.7)$$

where $0 \leq \alpha_{ad,t}(x, y) \leq 1$. The learning rate $\alpha_{ad,t}(x, y)$ depends on two weighted parameters, α_1 and α_2 (note that the subscript t and (x, y) are dropped for simplicity),

$$\alpha_{ad,t}(x, y) = w_1\alpha_1 + w_2\alpha_2, \quad (3.8)$$

where w_1 and w_2 are the weights empirically determined for α_1 and α_2 respectively.

The first parameter α_1 depends on the magnitude of $D_t(x, y)$ (from Equation 3.4). A larger value for α_1 is assigned for a smaller $D_t(x, y)$.

$$\alpha_1 = \begin{cases} e^{-\frac{1}{2} \frac{D_t(x,y)^2}{\sigma_1^2}} & \text{if } D_t(x, y) < Th_{ad}, \\ 0 & \text{otherwise,} \end{cases} \quad (3.9)$$

where $\sigma_1 = 10$ is empirically determined. This parameter ensures the group of pixels with the smallest $D_t(x, y)$ have the largest values for α_1 . Figure 3.2 shows the histogram of a difference frame $D_t(x, y)$, the threshold Th_{ad} associated with the histogram, the σ_1 is a pre-determined threshold, and the plot of α_1 which is a function of σ_1 .

The second parameter α_2 is obtained depending on the temporal duration of a pixel in the background. We assume the longer a pixel is in the background, the more stable and reliable a background pixel is. The stability and reliability is measured by the temporal background count C_{bg} , which is incremented by 1 each time when a pixel is classified as a background pixel *consecutively*. The parameter α_2 is determined according to:

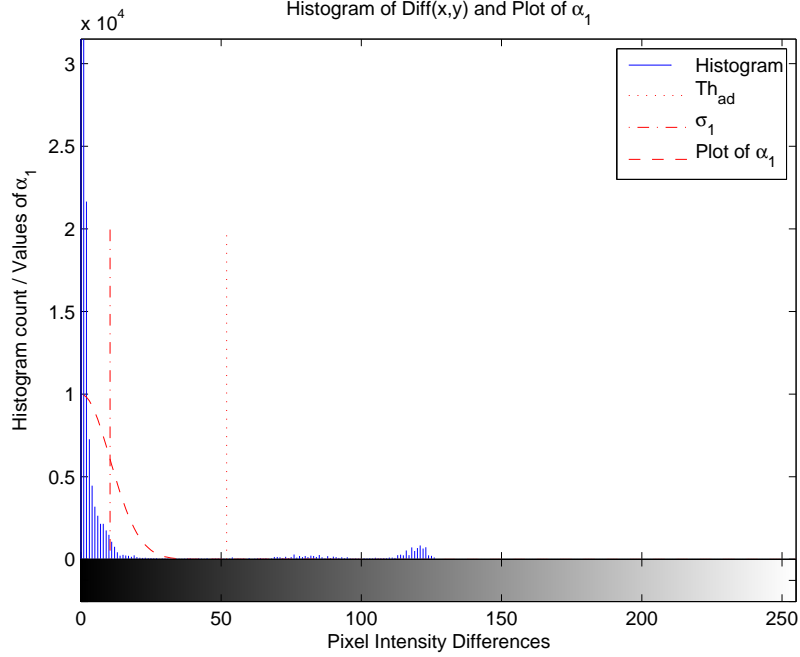


Fig. 3.2. Histogram of pixel values in frame $D_t(x, y)$. $Th_{ad} = 52$ is the adaptive threshold determined using the method described in Section 3.1.2. Consider Equation 3.9, $\sigma_1 = 10$ is empirically determined, and the plot of α_1 as $D_t(x, y)$ increases is shown.

$$\alpha_2 = \begin{cases} e^{-\frac{1}{2} \frac{(\zeta_{max} - C'_{bg})^2}{\sigma_2^2}} & \text{if } C_{bg} \geq \zeta_{min}, \\ 0 & \text{otherwise,} \end{cases} \quad (3.10)$$

where $\sigma_2 = 15$, $\zeta_{max} = 150$, and $\zeta_{min} = 30$ have been empirically determined, and $C'_{bg} = \min(\zeta_{max}, C_{bg})$. If a pixel remains as a background pixel for more than ζ_{min} frames, a non-zero α_2 is assigned to that pixel. The parameter α_2 increases with C_{bg} until C_{bg} is greater than ζ_{max} (i.e. C_{bg} is clipped at ζ_{max}), it has a small value for pixels of objects with small and repetitive motion such as waving tree leaves and water ripples (because these pixels have small values of C_{bg}). This works under the assumption that if the pixel switches between foreground and background states frequently (that is the pixel does not retain its background status for more than ζ_{min} frames), then it is not a “reliable” background pixel and hence it has a smaller

learning rate. Note that the temporal background count C_{bg} is reset to zero if the pixel is detected as a foreground pixel, it is because of the *consecutive* nature of the temporal background count defined above. The parameters α_1 and α_2 are used to update the background model frame using Equation 3.7 and Equation 3.8.

3.1.5 Statistical-Based Foreground/Background Pixel Classification

Many state-of-the-art approaches [10, 89] can handle gradual illumination changes well but remain vulnerable to sudden illumination changes. The assumption that the pixel intensities of background regions are Gaussian distributed does not hold because there is photometric distortion in the case of sudden illumination changes. The increase of intensities is not a constant even when the illumination change is global causing a drastic increase in false positive detections which in many cases the entire image is detected as foreground.

In this section, we describe a method for foreground/background pixel classification under sudden illumination change. The key differences between the method described previously in Section 3.1.3 and this approach is that the previous method uses the difference of pixel intensities as opposed to the illumination effect described later in this section. Another difference is that instead of thresholding the difference frame with an adaptive threshold, we “test” the statistical distribution of the neighboring pixels in the difference frame to determine if a pixel is a foreground or background pixel. This statistical test is referred to as “Gaussianity test” and will be described below in detail.

Gaussianity Test

A Gaussianity test determines if a set of samples are from a Gaussian distribution. The use of the test is inspired by Gurcan *et al.* [90, 91] who used it for detection of microcalcifications in mammogram images. This test was developed by Ojeda *et al.* [92] for causal invertible time series data. Gurcan *et al.* [90, 91] made the

assumption that when the microcalcification is absent, the difference image between the original image and the filtered image using a Least Mean Square adaptive filter are samples from a Gaussian distribution. If a region has a ‘‘Gaussianity test value’’ higher than a pre-determined threshold, then the region is classified as a region of microcalcification cluster. Otherwise, the Gaussianity test value (or *statistic*) should be close to zero. The test was also used by Chien *et al.* [93] for change detection under the assumption that camera noise is Gaussian distributed. We use this same assumption here.

Our proposed method is based on the assumption that the camera noise is *spatially* Gaussian. In addition, we assume that this Gaussian noise is temporally uncorrelated and hence, independent across the frames. Hence only Gaussian noise and foreground objects remain in the difference frame (because the sum of independent Gaussian random variables is Gaussian). Under these assumptions, the foreground pixels in the difference frame should be non-Gaussian distributed, and the background pixels in the difference frame should be Gaussian distributed.

Consider the moment generating function of a random variable X that is Gaussian distributed with mean μ and σ^2 :

$$M(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}. \quad (3.11)$$

The k^{th} order moment of distribution J_k , is defined in terms of the moment generating function as:

$$J_k = E[X^k] = \frac{d^k}{dt^k} M(t)|_{t=0} \quad (3.12)$$

If a set of samples is Gaussian distributed, the sample moments converge to their theoretical values (as shown below) as the sample size increases under the ergodicity assumption, i.e.

$$\begin{aligned} J_1 &\rightarrow \mu \\ J_2 &\rightarrow \sigma^2 + \mu^2 \\ J_3 &\rightarrow \mu^3 + 3\sigma^2\mu \\ J_4 &\rightarrow \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4 \end{aligned} \quad (3.13)$$

Similar to [90,91] we construct our Gaussianity test by defining the Gaussianity test statistic as:

$$H(J_1, J_2, J_4) = J_4 + 2J_1^4 - 3J_2^2 \quad (3.14)$$

If we substitute the limit values in Equation 3.13 into Equation 3.14, then:

$$\begin{aligned} H(J_1, J_2, J_4) &= (\mu^4 + 6\mu^2\sigma^2 + 3\sigma^4) + 2\mu^4 \\ &\quad - 3(\sigma^2 + \mu^2)^2 \\ &= 0 \end{aligned} \quad (3.15)$$

Hence, the Gaussianity test statistic is expected to be close to zero if the set of samples is Gaussian distributed.

The Gaussianity test can be used in a block-based manner in a frame to detect non-Gaussianity. Consider a $M \times M$ block centered at pixel (x, y) , the test is based on the sample estimates of the first four moments of some values x in a frame given by:

$$J_k(x, y) = \frac{1}{M^2} \sum_{m=-\frac{M-1}{2}}^{\frac{M-1}{2}} \sum_{n=-\frac{M-1}{2}}^{\frac{M-1}{2}} [x(x+m, y+n)]^k \quad (3.16)$$

for $k = 1, 2, 3, 4$. In our experiments, we use $M = 6$ for each non-overlapping block in a frame. Note that in the rest of the thesis, we will drop the pixel location (x, y) when representing the first four moments J_1, J_2, J_3, J_4 for simplicity.

If a set of samples in a block has a Gaussianity test statistic greater than a threshold τ (which is empirically determined), then the group of pixels in that block is considered as Gaussian distributed.

Sudden Illumination Change

To address the problem of sudden illumination change, we utilize the ‘‘illumination effect’’ of an image along with the Gaussianity test for robustness to sudden illumination change. Consider an image, we assume all surfaces in an image are Lambertian

(i.e. there is *no specular reflection* in the scene). The luminance depends only on the irradiance (light reaching the surface) $e(x, y)$ and its albedo (diffuse reflectance coefficient) $a(x, y)$ [94–97]. Irradiance $e(x, y)$ is a function of visible light sources and surface normal. If the surface is Lambertian (perfectly diffuse) [94, 95], the pixel intensity value can be expressed as:

$$I(x, y) = e_i(x, y)a_i(x, y).$$

Similarly, we have the same assumption for the background model:

$$BM(x, y) = e_m(x, y)a_m(x, y).$$

We express the illumination effect as a ratio of pixel intensities:

$$R(x, y) = \frac{I(x, y)}{BM(x, y)}, \quad (3.17)$$

Assuming the pixel is not occluded by a foreground pixel, with the same surface, two values of albedo will be equal:

$$a_m(x, y) = a_i(x, y)$$

Combining with the assumption of Lambertian surface above, Equation 3.17 becomes:

$$R(x, y) = \frac{I(x, y)}{BM(x, y)} = \frac{e_i(x, y)a_i(x, y)}{e_m(x, y)a_m(x, y)} = \frac{e_i(x, y)}{e_m(x, y)}. \quad (3.18)$$

The ratio $R(x, y)$ is independent of the albedo and only depends on the irradiance (i.e. light source and surface orientation). This ratio is expected to be constant at different locations in a frame as long as the irradiance at those locations is the same. If the irradiance is constant throughout the entire frame, then the ratios for every pixel are expected to be constant. The assumption implies the case when there is *no shadow* in the scene, it is because the existence of shadows implies a change in irradiance.

Gaussianity of Pixel Intensities Ratio

If the camera noise is assumed to be Gaussian distributed, then ratio of intensities $R(x, y)$ can also be assumed to be Gaussian under certain assumptions.

Let the pixel intensities of a current frame a_i and a background frame b_i with Gaussian noise be a_i+X and b_i+Y respectively, where the camera noise $X \sim N(0, \sigma_x^2)$ and $Y \sim N(0, \sigma_y^2)$ are Gaussian distributed. The ratio of pixel intensities is expressed as:

$$\frac{a_i + X}{b_i + Y} = \frac{\frac{a_i}{b_i} + \frac{X}{b_i}}{1 + \frac{Y}{b_i}} = \left(\frac{a_i}{b_i} + \frac{X}{b_i}\right)\left(\frac{1}{1 + \frac{Y}{b_i}}\right)$$

Let r be $\frac{a_i}{b_i}$ which is a constant under our assumptions on a Lambertian surface. Assuming the camera noise Y is smaller than the intensities of the pixels b_i :

$$\left\|\frac{Y}{b_i}\right\| < 1$$

then using the geometric series to expand the ratio of pixel intensities (with the higher order terms ignored):

$$\left(r + \frac{X}{b_i}\right)\left(\frac{1}{1 + \frac{Y}{b_i}}\right) = \left(r + \frac{X}{b_i}\right)\left[1 - \frac{Y}{b_i} + \left(\frac{Y}{b_i}\right)^2 + \dots\right] \approx r + \frac{X}{b_i} - \frac{rY}{b_i}$$

Since X and Y are Gaussian distributed and independent, $r + \frac{X}{b_i} - \frac{rY}{b_i}$ in the above expression is also Gaussian because the sum of Gaussian distributed random variables is also Gaussian.

Foreground/Background Pixel Classification

In our new technique, we combine the Gaussianity test together with the illumination effect for an illumination-invariant foreground pixel detection method. We determine the intensity ratio for each pixel of the current frame and the background model. It is assumed that when there is no foreground objects in the scene, the ratios of pixel intensity should be Gaussian distributed.

Under the assumption that camera noise is Gaussian distributed, i.e. the pixel intensity of a frame is Gaussian distributed. As shown in Section 3.1.5, the illumina-

tion effect which is a ratio of pixel intensities in two frames should also be Gaussian distributed.

We first determine the pixel intensity ratio frame using Equation 3.17. Then we determine the first four moments of the pixel intensity ratios $R(x, y)$ (similar to Equation 3.16):

$$\hat{J}_k(x, y) = \frac{1}{M^2} \sum_{m=-\frac{M-1}{2}}^{\frac{M-1}{2}} \sum_{n=-\frac{M-1}{2}}^{\frac{M-1}{2}} [R(x+m, y+n)]^k, \quad (3.19)$$

and we then construct the Gaussianity test by defining the test statistic as:

$$H(\hat{J}_1, \hat{J}_2, \hat{J}_4) = \hat{J}_4 + 2\hat{J}_1^4 - 3\hat{J}_2^2 \quad (3.20)$$

In our experiments, if a set of samples in a block has a Gaussianity test statistic greater than a pre-defined threshold τ , then the pixel is considered as a foreground pixel.

$$FG(x, y) = \begin{cases} 1 & \text{if } H(\hat{J}_1, \hat{J}_2, \hat{J}_4) > \tau \\ 0 & \text{otherwise.} \end{cases} \quad (3.21)$$

where τ is empirically determined and is set to be 50 in our experiments.

3.1.6 Fast Implementation Using an Extension of Integral Images

Determining $H(x, y)$ in Equations 3.13 and 3.14 for the Gaussianity test is computationally expensive. In this section we describe a fast method for the Gaussianity test based on integral images discussed in Section 3.1.5.

We employ the use of integral images for fast computation of the Gaussianity test. To determine $J_k(x, y)$ with Equation 3.19 for $H(x, y)$ in Equation 3.14 using a fast method, we define the k^{th} integral moment $IJ_k(x, y)$ of an image $i(x, y)$ as:

$$IJ_k(x, y) = \sum_{x' \leq x, y' \leq y} [i(x', y')]^k \quad (3.22)$$

To determine Equation 3.22, the single pass implementation described in Equations 2.23 and 2.24 for the integral image is used. Consider a $w_b \times h_b$ block centered at pixel (x, y) , the k^{th} moment of the pixel intensity differences expressed in Equation 3.14 is now given by,

$$\hat{J}_k(x, y) = IJ_k(x_2, y_2) - IJ_k(x_2, y_1 - 1) - IJ_k(x_1 - 1, y_2) + IJ_k(x_1 - 1, y_1 - 1)$$

for $k = 1$ to 4, where $x_2 = x_1 + w_b$ and $y_2 = y_1 + h_b$ in this case. Finally, Equation 3.20 and Equation 3.21 are used to determine $H(\hat{J}_1, \hat{J}_2, \hat{J}_4)$ for the Gaussianity test.

3.2 Object Tracking

We use moving object detection to obtain a binary foreground mask for every frame. This indicates where the moving objects are located. Morphological filters are used to post-process every binary foreground mask. Connected component labeling is used to determine the number of moving objects in a scene [98]. The centroid and size (which is modeled by the smallest rectangle that encloses the moving segment) are determined for each component, and the information is used to initialize object tracking.

A special case occurs when an object enters a scene from one of the four borders of a frame. In this case, it is very likely that there are frames where only part of an object appears in (has entered) the frames, in this case the foreground binary mask will be “touching” the frame boarder. Object tracking is only initialized when an object has completely entered a frame. In this case the foreground binary mask will “not be touching” the frame boarder. This “not touch” scenario is expressed in the following in terms of a binary foreground mask $FG_t(x, y)$ at time t , with frame width and height $W \times H$, which was defined previously in Equation 3.5. If the binary mask has a value of 1 a pixel is classified as a foreground pixel and 0 if it is classified as a background pixel.

$$FG_t(x, y) = 0, \text{ if } x = 0 \text{ or } x = W - 1 \text{ or } y = 0 \text{ or } y = H - 1$$

When an object completely enters a scene, object tracking is initialized. The position and size of the moving objects can be extracted easily. The position is determined as the centroid of the moving segment in the foreground mask, and the size is modeled by the smallest rectangle that encloses the moving segment in the foreground mask.

Once object tracking is initialized, moving object detection will terminate hence no further foreground object masks will be generated.

In recent years, tracking techniques based on particle filtering has been extensively studied [24]. Experimental results show that particle filtering outperforms other conventional state estimation method such as Kalman filter for non-linear and non-Gaussian systems. Particle filters generalize the traditional Kalman filtering methods which provide the optimal solution only when the models have linear functions and Gaussian noise [24].

As mentioned previously in Section 2.2.3, particle filtering is an iterative process and each step consists of three major components. The first is state transition using the motion model, the second is the computation of particle weights (likelihood estimation) using the observation likelihood function, and the third is resampling.

The focus of our study is based on likelihood estimation. We will describe in detail:

- which visual features we use for object tracking
- how we represent the features as density functions
- how we assign weights to particles using the likelihood function discussed in Section 3.2.1
- how we set parameters dynamically for the likelihood function discussed in Section 3.2.2

- how we investigate the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation in Section 3.2.3
- how to construct the appearance model (density functions) using the fast implementation discussed in Section 3.2.4.

3.2.1 Object Tracking with Particle Filtering

Particle filter is popularly used for object tracking because it has been shown to be very successful for non-linear and non-Gaussian dynamic state estimation problems and is still very reliable in cases such as clutter and occlusions [24]. Experimental results show that particle filtering outperforms other conventional state estimation method such as Kalman filter for non-linear and non-Gaussian systems [21–24].

Object tracking can be considered as a hidden state estimation problem given observations. The state can be the position, velocity, scale, or orientation of the interesting object. The observation can be visual features such as color features or edge features. Many techniques have been proposed for object tracking that rely on a single feature [26, 29, 50]. Color distributions have been widely used as the appearance model for tracking problems [26, 29, 50] because they are robust to partial occlusion and are rotation and scale invariant. The use of color is flexible in the types of object that they can be used to track, including rigid (e.g. vehicles) and non-rigid objects (e.g. people). Color has limitations in the area where the background has a similar color as the target object in that the tracker can be confused and lose track of the object. Color distributions also have poor performance when the illumination varies.

Other features have been proposed in the literature for object tracking including shape [28] and gradient direction [57] or a combination of multiple features such as shape and color [61]. A single feature does not provide enough information about the object being tracked and hence using multiple features can provide more information about the object. For our approach, we use color (details will be discussed in Section

3.2.1) and also introduce an edge feature (details will be discussed in Section 3.2.1) for the observation likelihood function. The edge feature is useful for modeling the structure of the object to be tracked. The edges are described using a histogram based on the estimated strength and orientation of the edges.

The features are represented as probability density functions. To do this, consider the case for color feature. We obtain a “weighted” color histogram using kernel density estimation (details will be discussed in Section 3.2.1). This “weighted” color histogram is known as the color density function. The target object to be tracked and a particle (candidate as described previously in Section 2.2.3) is represented by color density functions. The candidate will then be evaluated according to how much its color density function resembles the color density function of the target object. The Bhattacharyya distance is used for the metric of comparison, described in Section 3.2.1. According to the likelihood model, more weights are assigned to candidates with smaller Bhattacharyya distances (details are discussed in Section 3.2.1 and Section 3.2.2). To illustrate this idea of the likelihood model, consider the example in Figure 3.3, the candidate in red will be assigned a higher weight, and the candidates in yellow will be assigned lower importance weights. At the end of each iteration, the estimated state is the weighted average of all the candidates.

Color Features

This section describes how we model color features of a region R , where R can be a region surrounding the object to be tracked or region surrounding one of the hypothetical regions represented by a particle.

The color distribution is expressed by a m -bin histogram, whose components are normalized so that the total count of all bins equals one. For a region R in an image, given a set of n_R pixels in R , denoted by $\mathbf{S} = \{x_i, i = 1, 2, \dots, n_R\} \in R$, the m -bin color histogram $H(R) = \{h_j, j = 1, 2, \dots, m\}$ can be obtained by assigning each pixel, x_i to a bin, by the following equation:

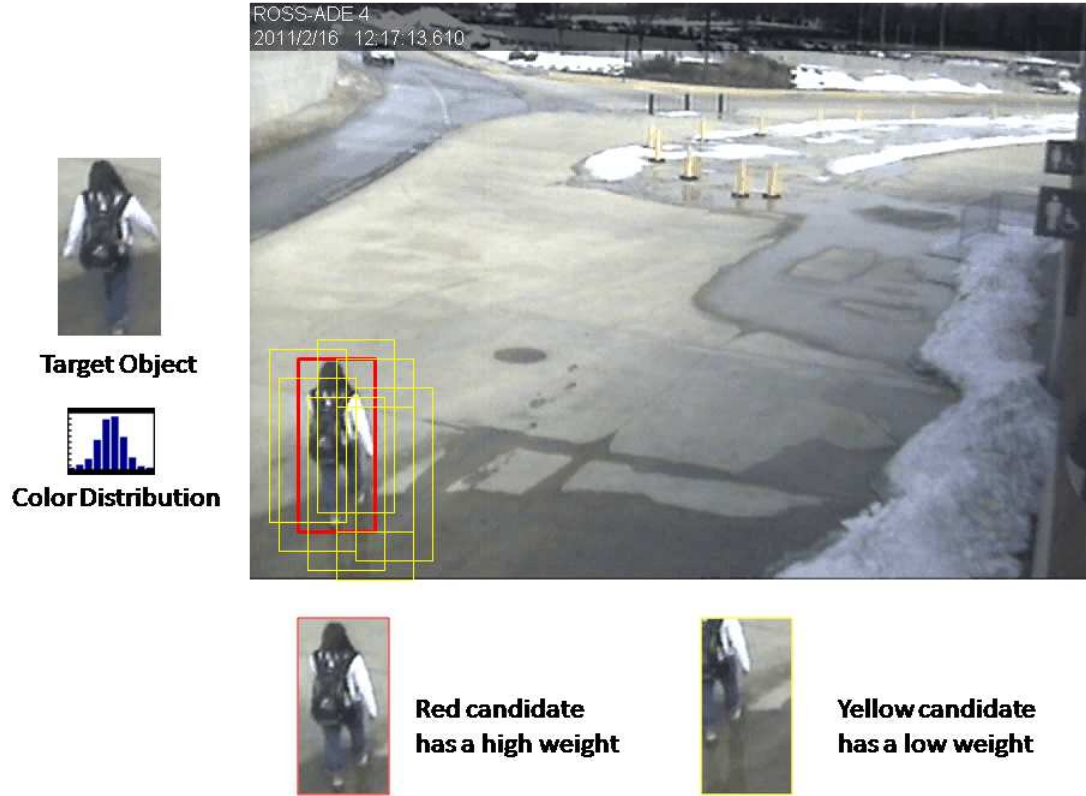


Fig. 3.3. An example of particle weight assignment using visual features. The candidate in red will be assigned a higher weight, and the candidates in yellow will be assigned lower weights.

$$h_j = \frac{1}{n_R} \sum_{(x_i) \in S} \delta_j[b(x_i)] \quad (3.23)$$

where $b(x_i)$ is the bin index where the color component at x_i falls into, and δ is the Kronecker delta function.

In our experiment, HSV color space is used to construct a $8 \times 8 \times 4$ -bin histogram for each region R . Less bins are used for V in order to reduce the sensitivity to lighting conditions [50].

Edge Features

Color is a powerful feature for target tracking, using only color features is sufficient for most tracking tasks, including cases such as occlusion and rotation. However, problems arise when the target object and background have similar color distribution. It may be difficult to distinguish the object from background. To solve this problem, multiple-feature tracking is used because it provides more information about the object. Edge features are introduced as a complementary feature to our appearance model because it is useful for modeling the structure of the object to be tracked. The edge features are used with a histogram based on the estimated strength and orientation of the edges. Figure 3.4 and Figure 3.5 show the edge orientation histograms of some persons and vehicles. The four examples for edge orientation histogram of vehicle have peaks at the 4th and the 5th bins. The four examples of person have significantly more counts at the 1st, 4th, 5th, and 8th bins. One can observe the consistency in performance and reliability of such a simple low level visual feature.

Edges are detected using a simple horizontal and vertical Sobel operators K_x and K_y , which are convolved with the original grayscale image $I(x, y)$. The gradient components of pixel x_i in horizontal and vertical orientations, represented as $G_x(x, y)$ and $G_y(x, y)$, have magnitudes:

$$G_x(x, y) = K_x * I(x, y) \quad (3.24)$$

$$G_y(x, y) = K_y * I(x, y) \quad (3.25)$$

where

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \quad (3.26)$$

The strength and the orientation of the edges are defined as:

1. Strength

$$S(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)} \quad (3.27)$$

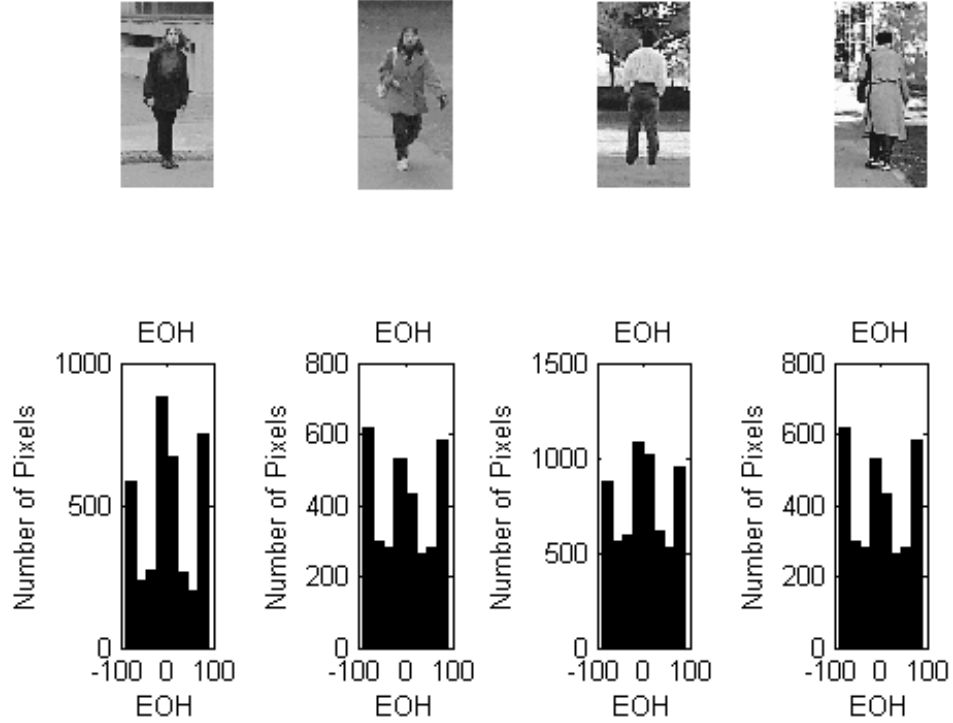


Fig. 3.4. Examples of edge orientation histogram of people.

2. Orientation

$$\theta(x_i) = \arctan(G_y(x, y)/G_x(x, y)) \quad (3.28)$$

A pre-determined threshold T_{edge} is used to remove noise, by discarding any edge orientation with strength less than a threshold for the histogram construction. The threshold is set to 100 in our experiments. The process is defined as:

$$\theta'(x_i) = \begin{cases} \theta(x_i) & \text{if } S(x, y) > T_{edge} \\ 0 & \text{otherwise} \end{cases} \quad (3.29)$$

$\theta'(x_i)$ instead of $\theta(x_i)$ is used to construct the histogram.

For a region R in an image, given a set of n_R pixels in R , denoted by $\mathbf{S} = \{(x_i), i = 1, 2, \dots, n_R\} \in R$, the k-bin edge orientation histogram $EOH(R) = \{e_j, j =$

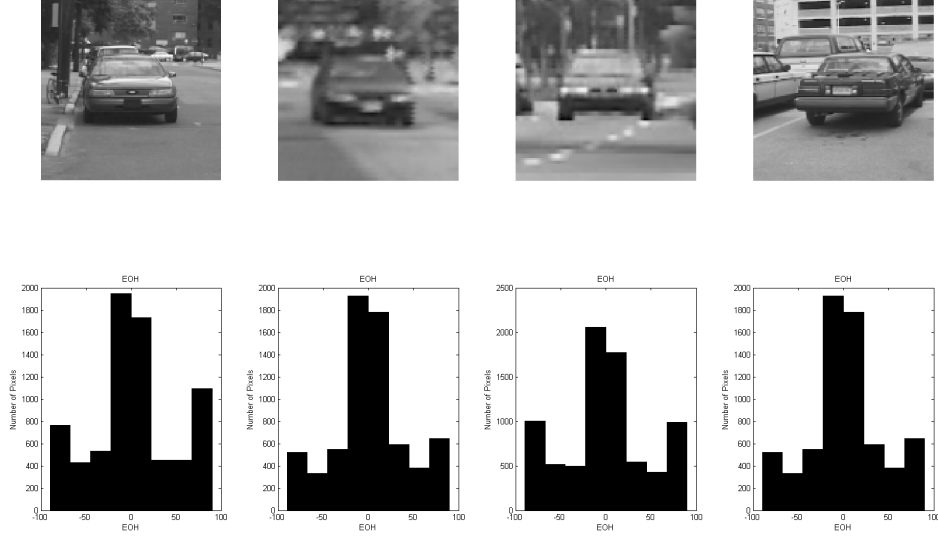


Fig. 3.5. Examples of edge orientation histogram of vehicles.

$1, 2, \dots, k$ is computed by assigning each pixel, (x_i) to a bin, by the following equation:

$$e_j = \frac{1}{n_R} \sum_{(x_i) \in S} \delta_j[b(x_i)] \quad (3.30)$$

where $b(x_i)$ is the bin where $\theta'(x_i)$ falls into.

Kernel Density Estimation

The probability density function of the visual features can be estimated using the set of pixels $\mathbf{S} = \{x_i, i = 1, 2, \dots, n_R\} \in R$ by kernel density estimation. Kernel density estimation is one of the most popular non-parametric methods for probability density function estimation [26, 55]. The features will be more unstable at peripheral points in the region than its center when the object is suffering from partial occlusions and background changes. To avoid this instability, assigning larger weights for

the central points of the region is achieved by applying a convex and monotonically decreasing function [50, 99, 100].

For a region R centered at pixel x_c , a density estimator with kernel $K(x_c)$ is defined as:

$$pr(x_c) = \frac{1}{n_R} \sum_{i=0}^{n_R} K\left(\frac{|x_i - x_c|}{h}\right) \quad (3.31)$$

where h is the size (also known as bandwidth) of the kernel, and $|\cdot|$ is the L2 norm. A popular choice for kernel $K(\cdot)$ (for object tracking application) is the Epanechnikov kernel, $K_e(\cdot)$, which is a convex and monotonic decreasing function, and it is defined as:

$$K_e(x) = \begin{cases} \frac{3}{4}(1 - |x|^2) & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.32)$$

We estimate the probability density function by using every pixel in the region R as a sample for the histogram (color histogram or edge orientation histogram) followed by using a $w_k \times h_k$ Epanechnikov kernel filter (a weighting scheme for the histogram construction). The probability of the quantized color vector (bin count of the kernel filtered histogram) u in the object model is given by:

$$q_u(x_c) = C \sum_{i=1}^{n_R} K\left(\frac{|x_c - x_i|}{\sqrt{w_k \times h_k}}\right) \delta[b(x_i)] \quad (3.33)$$

where C is the normalization factor

$$C = \frac{1}{\sum_{i=1}^{n_R} K\left(\frac{|x_c - x_i|}{\sqrt{w_k \times h_k}}\right)} \quad (3.34)$$

that ensures

$$\sum q_u(\cdot) = 1.$$

This “weighted histogram” obtained from the kernel density estimation is also referred to as the appearance model in the remaining context of this thesis.

Distance Measure

A target appearance model and a candidate appearance model are represented by edge or color distributions described previously. A distance metric known as Bhattacharyya distance is used to measure the dissimilarity. An observation likelihood function is then used to assign a weight associated to a specific particle (new observation) according to the Bhattacharyya distance. To evaluate the similarity between the appearance model of the target q , and the appearance model of the i^{th} particle at time t , denoted by $p(s_t^{(i)})$, we employ the Bhattacharyya coefficient $\rho(\cdot)$ which is defined as:

$$\rho[p(s_t^{(i)}), q] = \sum_{u=1}^m \sqrt{p_u(s_t^{(i)})q_u} \quad (3.35)$$

where m is the total number of quantized color vectors (histogram bins). Using the above function, the larger $\rho(\cdot)$ is, the more similar the two distributions are. For two identical normalized histograms we obtain $\rho(\cdot) = 1$, indicating a perfect match. To quantify the distance between two distributions, the distance $d_{feature}$ is defined as:

$$d_{feature}(p(s_t^{(i)}), q) = \sqrt{1 - \rho[p(s_t^{(i)}), q]} \quad (3.36)$$

which is known as the Bhattacharyya distance. The observation likelihood function uses this distance to assign a weight to each particle. A sample with small Bhattacharyya distance corresponds to a larger weight; similarly, a sample with large Bhattacharyya distance corresponds to a smaller weight. The weights are approximated using a Gaussian distribution of the Bhattacharyya distances.

The distance $d_{color}(\cdot)$ is determined from the color distribution, while $d_{edge}(\cdot)$ is determined from the edge orientation distribution. They are combined by:

$$d(p(s_t^{(i)}), q) = 0.5 \times d_{color}(p(s_t^{(i)}), q) + 0.5 \times d_{edge}(p(s_t^{(i)}), q)$$

Note that the distance and the importance weight of the i^{th} particle at time t , $d(p(s_t^{(i)}), q)$ and $w_t^{(i)}$ respectively, are represented as d and w for simplicity. After

d is determined, it can be used to assign a weight to the particle. An observation likelihood function that is typically used for this process is:

$$\mathcal{L}(z|x) \propto e^{-\frac{d^2}{2\sigma^2}} \quad (3.37)$$

The weights are assigned according to the likelihood function except that it must be normalized and sum to 1:

$$w = \frac{1}{c_w} e^{-\frac{d^2}{2\sigma^2}} \quad (3.38)$$

where c_w is the normalizing constant that ensures:

$$\sum_i w_t^{(i)} = 1,$$

and the value of the parameter σ will be discussed in detail in the next section.

Dynamic Model

As mentioned previously, one major component of the particle filtering iteration is the dynamic model, it is also known as the transition model. Let the i^{th} sample state of position (x_t, y_t) at time t be $s_t^{(i)}$, i.e. $s_t^{(i)} = [x_t, y_t]^T$. The transition model that describes the dynamics of the object is given by:

$$s_t^{(i)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot s_{t-1}^{(i)} + \begin{pmatrix} w_{xt} \\ w_{yt} \end{pmatrix} \quad (3.39)$$

where w_{xt} and w_{yt} are pre-determined Gaussian noise with zero mean and standard deviation σ_x and σ_y . This model is also known as a random walk model with fixed noise variance.

State Estimation

After determining the weights of all the samples, the position of the tracking object at time t is estimated as the weighted average of the samples:

$$E[S_t] = \frac{1}{\sum_{i=1}^N w_t^{(i)}} \sum_{i=1}^N w_t^{(i)} s_t^{(i)} \quad (3.40)$$

3.2.2 Dynamic Parameter Setting for the Likelihood Model

The observation likelihood function (or likelihood model) is used to determine the importance weights of particles. It is a function of distance between the target object appearance model and the particle appearance model. If a particle has features similar to the features of the target, this particle has a smaller distance from the object model, and it will be assigned with a larger weight according to the observation model. The observation likelihood function is expressed as $\mathcal{L}(z_t|x_t^{(i)})$ in Equation 2.20. In the case of the standard implementation, the importance density $q(x_t^{(i)}|x_{t-1}, z_t)$ is chosen to be the dynamic transition model, $p(x_t^{(i)}|x_{t-1}^{(i)})$. In this case, the observation likelihood function is the only function that contributes to a particle's weight. The observation likelihood function is usually assumed to be a zero-mean Gaussian parametrized its variance. The function is represented as follows:

$$\mathcal{L}(z|x) \propto e^{-\frac{d^2}{2\sigma^2}} \quad (3.41)$$

Because the observation likelihood function is used to assign importance weights to particles, the function has a significant impact on the tracking performance in two ways:

- The weights are used to estimate the state (i.e. location) of the target object
- Particles are re-sampled according to their weights, i.e. a high-weighted particle will be duplicated and a low-weighted particle will be eliminated for the next iteration.

The parameter σ of the observation likelihood model determines the “steepness” of the function, i.e. how fast the function decreases as the distance increases. A similar likelihood function is proposed in [57]:

$$\mathcal{L}(z|x) \propto e^{-\frac{d^2}{2\sigma^2}} + \epsilon \quad (3.42)$$

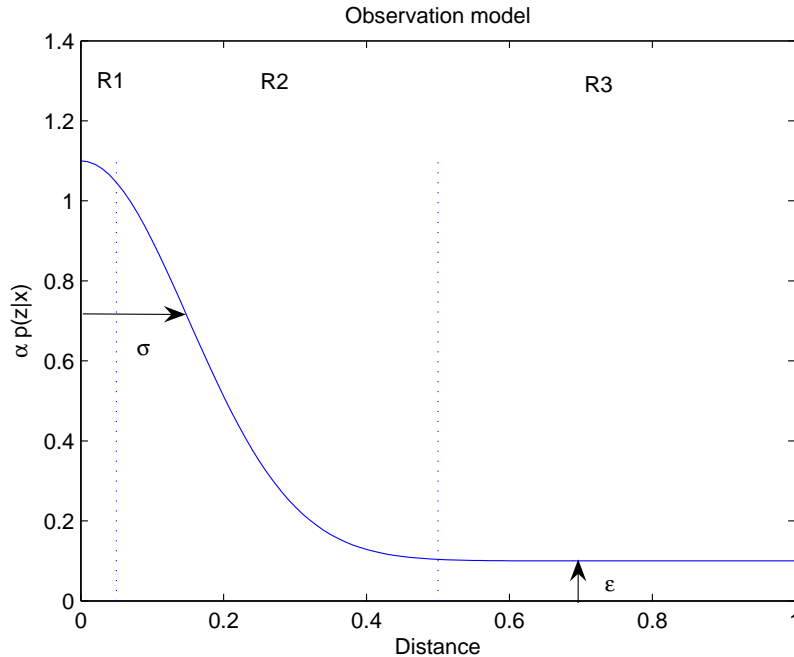


Fig. 3.6. Observation likelihood function with $\epsilon = 0.1$ and $\sigma = 0.15$.

where σ and ϵ are the model parameters. σ determines the steepness of the function, and ϵ is set to be a small constant. The function is shown in Figure 3.6, it is an example of the function with $\epsilon = 0.1$ and $\sigma = 0.15$. The figure is separated into 3 regions (R1, R2, and R3) for this discussion. For the good particles with distances close to zero (R1 in the figure), the model introduces some tolerance for the difference due to distortion. So particles fall into this region are assigned very high weights. The plummet in R2 discards the bad samples when there are good particles present in R1. The worst particles that fall onto R3 are weighted equally and are determined by ϵ . In [57], the importance of choosing the optimal model parameters of the observation likelihood function is investigated. The negative effects on the tracking performance when the model parameters are too large or too small are shown. However, suggestions on how to set the parameters are not provided.

We present a technique to set the parameters of likelihood function dynamically for each particle filtering iteration. From the likelihood function in Equation 3.41, we rearrange and solve for σ

$$\sigma = \frac{\sqrt{2}}{2} \sqrt{\frac{-d_{min}^2}{\log L}} \quad (3.43)$$

Where d_{min} is the distance of the most reliable particle (the particle with the minimum distance), and

$$\log(L) = -1$$

By using this σ , the “best” particle (i.e. the one with the minimum distance) is assigned the highest weight. An adaptive σ always ensures the best particles are assigned the highest weights and the worst ones are assigned a small value ϵ . For example, for the case of a fixed $\sigma = 0.15$ is used, as shown in Figure 3.6, if all particles have distances greater than 0.4, then all particles fall onto R3 and they are all assigned equally low weights ϵ . As a result the likelihood function will not differentiate the difference between particle with distance equals 0.4 and particle with distance equals 1. However, by adaptively setting σ according to d_{min} , there will be at least one particle that falls onto R1, and the worst particles will still fall onto R3, making the observation likelihood function more discriminative.

Consider the case where the particle that results in d_{min} is an outlier. In that case, the adaptive σ would still have a value that is too small, as a result, majority of the particles would have small weights using Equation 3.38. For this reason, we determine the value of σ using the standard deviation of the distances in this case when adaptive σ is too small. We detect this phenomenon (i.e. majority of the particles having small weights) by determining if at least a certain percentage of the particles have weights smaller than a threshold (e.g. 70% of the particles have weights less than 0.05). In this case, we use $\sigma = 2 \times \sigma_{dist}$ instead, where σ_{dist} is the standard deviation of the distances.

3.2.3 Effect of Subsampling on Color Density Estimation

In this section we investigate the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation. The motivation of this section is that each iteration of particle filtering based tracking system consists of three steps: state transition, likelihood estimation, and resampling. Generally, likelihood estimation requires the most computational cost among all these operations [101]. For example, it has been shown that computation of the color histograms at each iteration is a major bottleneck of the color-based particle filter object tracking [64]. The traditional way to compute the color density is to construct the color density using every pixel in the tracker. But in this thesis, we use a subset of the pixels inside the tracker for the color density construction e.g. subset of pixels that forms a checkerboard pattern. This pattern is shown in Figure 3.7. Only the pixels marked by crosses are used to construct the color distribution. This method takes advantage of the spatial redundancy of natural images since neighboring pixels are highly correlated.

x		x		x		x		x
	x		x		x		x	
x		x		x		x		x
	x		x		x		x	
x		x		x		x		x
	x		x		x		x	
x		x		x		x		x
	x		x		x		x	
x		x		x		x		x

x		x		x		x		x
x		x		x		x		x
x		x		x		x		x
x		x		x		x		x
x		x		x		x		x

Fig. 3.7. The pixels marked by a cross are used to construct the histogram. Left: One half of the pixels are used. Right: One quarter of the pixels are used.

In this thesis, we will investigate the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation.

3.2.4 Fast Method For Constructing The Appearance Model

Utilizing the concept of integral images, the integral histogram [69] is proposed as a fast method to construct histograms over rectangular regions. The idea is the same as integral image described in Section 2.2.4. To construct the integral histogram, similar to Equation 2.22, an integral histogram is defined as:

$$IH_j(x, y) = \sum_{x' \leq x, y' \leq y} \delta_j[b(x, y)] \quad (3.44)$$

where $b(x, y)$ is the bin index where the color component at (x, y) falls into, and δ is the Kronecker delta function. This is similar to the regular histogram construction in Equation 3.23 except that it sums every pixel for $x' \leq x, y' \leq y$ instead of the pixels in a rectangular region $(x, y) \in R$ which can be anywhere in a frame.

The integral histogram is propagated from the top left corner of an image and at each new visited pixel, one single bin of the integral histogram is updated. To find the histogram of a rectangular region using the integral histograms, the value of each histogram bin over any rectangular region R can be obtained with 3 arithmetic operations, similar to Equation 2.25, as show in the following:

$$h_j(x, y) = IH_j(x_2, y_2) - IH_j(x_2, y_1 - 1) - IH_j(x_1 - 1, y_2) + IH_j(x_1 - 1, y_1 - 1) \quad (3.45)$$

where j is the index of the histogram bin.

Despite the gain in the computational cost, the integral histogram only allows a fast implementation of uniform kernel filtering (also known as box filtering or uniform filter), i.e. it constructs a histogram of equally weighted pixels. However, it has been shown that convex and monotonically decreasing, e.g. Epanechnikov and Gaussian kernels [49] are required for better performance in object tracking. In this thesis, we present an extension of the integral histogram for non-uniform kernel filtering. It approximates filtering kernels using a weighted sum of uniform kernel filters. The

number of uniform filters used are $N = 3$ to 6. We will also refer to the weighted sum of uniform kernel filters as the “staircase function” in this thesis. Figure 3.8 shows how the non-uniform kernels are approximated by the staircase functions with $N = 4$.

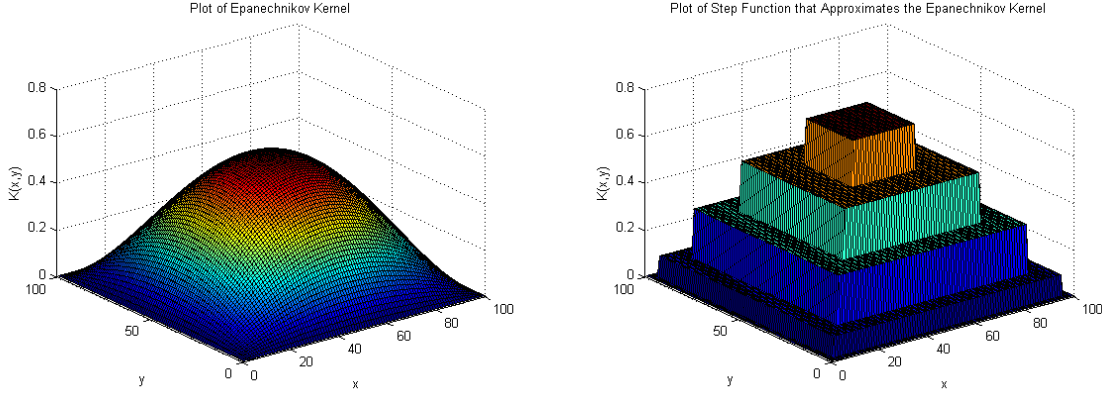


Fig. 3.8. (a) An Epanechnikov kernel. (b) Approximated Epanechnikov Kernel using $N = 4$.

The advantage of this method is that only one integral histogram has to be constructed (i.e. Equation 3.44 is determined once only) regardless of what the kernel function is. A larger N (number of stacked uniform filters) approximates the kernel function more accurately, with the trade-off between the approximation accuracy and the computational cost. The accuracy of the approximation also depends on the parameters of the staircase function, i.e. the widths, heights (box filter sizes), and weights (box filter height). In order to optimize the parameters in the minimum mean square error (MMSE) sense, we define the cost function:

$$C = \sum_{m=1}^{w_f} \sum_{n=1}^{h_f} (K(m, n) - \hat{K}(m, n))^2 \quad (3.46)$$

where $K(\cdot, \cdot)$ is the original $w_f \times h_f$ kernel function, $\hat{K}(\cdot, \cdot)$ is the staircase function that approximates $K(\cdot, \cdot)$. We use a method to optimize the parameters that utilizes a method that was developed in 1957 by Lloyd (also known as k-means method) [102, 103] for quantizer design.

Parameters Optimization

In order to find the parameters of the staircase function $\hat{K}(\cdot, \cdot)$ that minimize the cost function in Equation 3.46, let the mean square error (MSE) and the staircase function as:

$$MSE = \frac{1}{w_f \times h_f} \sum_{m=1}^{w_f} \sum_{n=1}^{h_f} (K(m, n) - \hat{K}(m, n))^2 \quad (3.47)$$

where

$$\hat{K}(m, n) = \sum_{i=1}^N a_i [u(m, n) - u(m - w_i, n - h_i)] \quad (3.48)$$

and $u(\cdot, \cdot)$ is the 2-dimensional step function, a_i is the height of the box filter, w_i and h_i is the size of the 2-dimensional box filter.

A 1-dimensional case of this problem is illustrated in Figure 3.9, it is the Epanechnikov kernel and the staircase function that is used to approximate it. The shaded part represents the approximation error which is the entity we want to minimize.

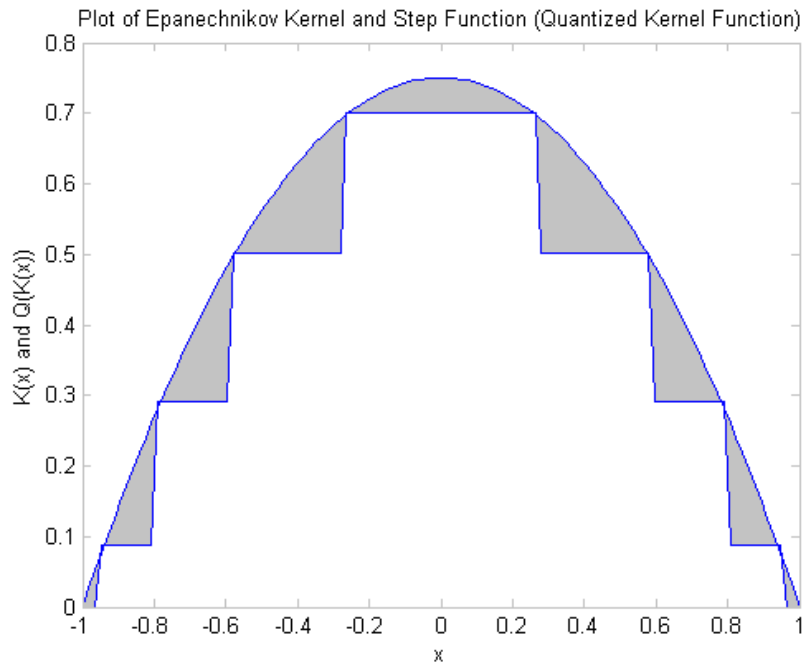


Fig. 3.9. The shaded part represents the approximation error of the kernel function using a staircase function.

As stated previously, a larger N approximates the kernel function more accurately, Figure 3.10 below shows the plots of the approximation error with increasing N using the Gaussian kernel and the Epanechnikov kernel.

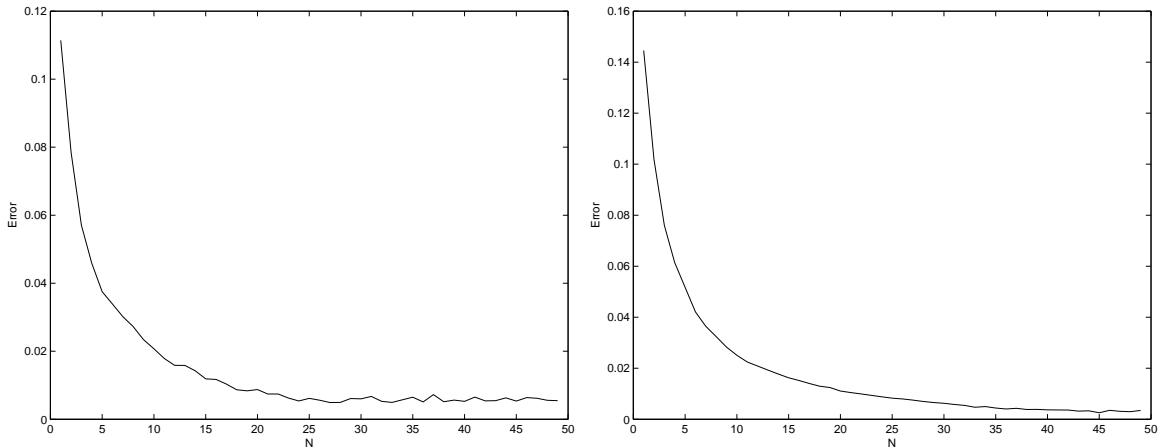


Fig. 3.10. The approximation error with increasing N , the kernel function used to generate the figure is (a) Gaussian kernel and (b) Epanechnikov kernel.

Using this method for color density construction, the weighted sum of histogram bins $h_u(x, y)$ in Equation 3.45 are used. An estimation of the color density is:

$$\tilde{p}_u(x, y) = \sum_{i=1}^N a_i [h_u(x, y)] \quad (3.49)$$

3.3 Crowd Analysis

The work presented in this section was done by the author jointly with Satyam Srivastava.

There are various research topics in video surveillance related to crowds. For example, crowd density estimation, face detection and recognition in crowds, crowd abnormal behavior detection. In our work, we focus on the problem of crowd flow estimation. We address the problem of accurate estimation of crowd flow using a single view. This analysis can help determine the average pedestrian traffic at a point

of interest, detect important ingress/egress points and possible bottlenecks, and also detect panic situations when the crowd motion patterns become anomalous. Our work extends the related work in crowd density [78] and flow estimation [81, 82]. There are two main new concepts described in our method – we use two unrelated features (foreground pixels and texture) to make the analysis more robust and we describe a complete process of training and deploying the system. Our method highlights the role of human user input for training which allows it to be flexible and work seamlessly even when crowd densities change significantly. We tested our methods with crowd videos from publicly available datasets and obtain good accuracies. Figure 3.11 shows an overview of our proposed method for crowd flow estimation for testing data. The training process will be described later in Section 4.3.1.

3.3.1 Flow Estimation by Pixel Accumulation

As stated earlier, our goal is to estimate the number of persons crossing a chosen area in a given time interval. This is referred to as “flow,” and the designated area as a virtual “trip wire.” A direct approach would consist of detecting and tracking every individual in the scene (at least those near the trip wire) and updating the count when any such tracked individual crosses the trip wire. However, robust simultaneous tracking of potentially hundreds of targets is not practical.

We use an indirect approach for flow estimation with the working hypothesis that certain low level image features are closely related to the presence and density of people in the scene. Specifically, our approach accumulates the number of foreground pixels in the trip wire region over a period of time. This cumulative pixel count is then scaled to produce an estimate of the flow. This scaling factor is the typical number of pixels accumulated when one person crosses the trip wire.

Our approach is based on the weighted pixel counting method described in [81]. In order to estimate the flow (persons per second), we consider a set of consecutive frames of duration T seconds. The frames are represented as

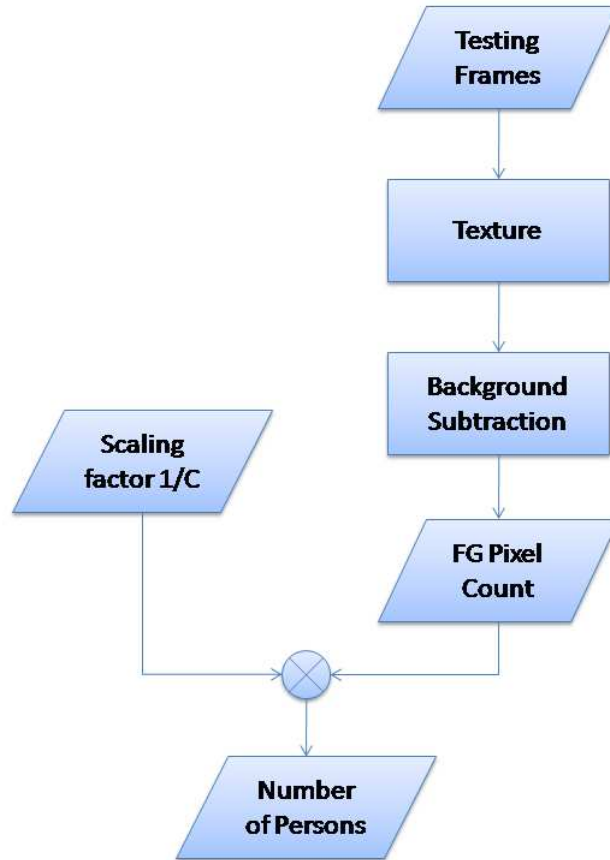


Fig. 3.11. An overview of our proposed method for crowd flow estimation.

$$F_{n_i} = f_{n_i}(x, y). \quad (3.50)$$

Here n_i is the frame number with $i = 0, 1, \dots, N - 1$ where N is the total number of frames in the segment (related to T through the frame rate). Finally, let the trip wire be represented as a set $\mathfrak{R} = \{f(x, y) | (x, y) \in \text{trip wire}\}$.

For flow estimation using pixel accumulation, we consider the RGB color of a frame at (x, y) to determine the foreground pixels. Background subtraction is a widely researched topic in image analysis [104]. We use a low complexity adaptive background subtraction technique [105] to classify the pixels in \mathfrak{R} . In this technique the RGB background model is constructed progressively and the decision threshold

is determined from the scene statistics. Note that the computational cost of background subtraction in only the trip wire region is much smaller than the cost for the entire frame. We can represent the foreground mask by an indicator function $I(x, y)$. Figure 3.12 shows an example of a video frame and the associated foreground mask in the trip wire region (indicated by red lines).

To classify the foreground pixels in \mathfrak{R} , the background subtraction method described in Section 3.1.3 is used to account for background dynamics in some of the sequences.

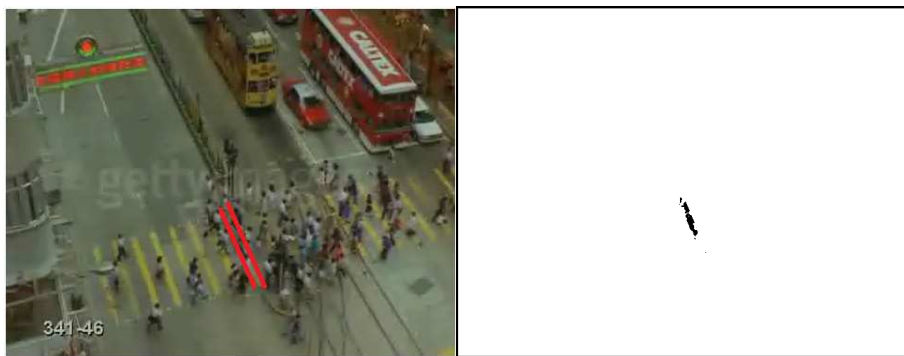


Fig. 3.12. An example of foreground mask generation by background subtraction.

The pixel count in trip wire region \mathfrak{R} is described as follows:

$$S_T = \sum_{i=0}^{N-1} \sum_{x,y \in \mathfrak{R}} I_{n_i}(x, y). \quad (3.51)$$

Note that some weighting scheme is needed to account for the effects of perspective, velocities, and direction of motion with respect to the normal to the trip wire. Methods to compensate for these effects were described in [81]. Here, we assume that the combined effect of these methods can be represented by a single weight factor $w_{n_i}(x, y)$. Thus, the weighted pixel summation is given by:

$$\tilde{S}_T = \sum_{i=0}^{N-1} \sum_{x,y \in \mathfrak{R}} I_{n_i}(x, y) \times w_{n_i}(x, y). \quad (3.52)$$

The second step in the estimation process involves scaling the *weighted* pixel count \tilde{S}_T by the number of weighted pixels obtained when an individual crosses the trip wire.

We represent this quantity by C . The final estimate of the number of persons crossing the trip wire in time T is:

$$\nu_T = \tilde{S}_T/C. \quad (3.53)$$

It is important to note that the scaling factor C is highly dependent on the sequence (size of individuals) and “crowdedness” of the scene. Crowdedness is related to the crowd density level, that is the number of persons present in the scene. A scene of high crowd density level has more occlusions with people. Thus, the factor C is obtained by a training process which is described in Section 4.3.1. Furthermore, multiple such factors are computed for a sequence and the most appropriate is automatically selected based on the estimated crowdedness. In [81] the variation was not considered resulting in a single C for a sequence. This approach would not be accurate in cases with occlusion. Later, an improvement was suggested to use foreground pixel count (in the full frame) for estimating the crowd density levels [82]. In this thesis, we propose the use of texture features to determine the crowd density level. This approach is better than the one using foreground pixel counting [82] because texture features are more robust to small variations in the background and lighting. We estimate the texture features in a subset of the frame known as the region of interest (ROI). We next describe the computation of the texture features (Section 3.3.2) used for density estimation. These features are associated with various levels of crowd densities through a training process described in Section 4.3.1.

3.3.2 Crowd Density Level Estimation with Texture Features

Crowd density level is closely related to the amount of occlusion of persons (by each other). In a flow estimation problem this becomes important because the number of pixels accumulated over a period of time does not scale linearly with the number of persons in presence of occlusion. Typically, an individual contributes more foreground pixels in a sparsely crowded scene than in a densely crowded scene. Therefore,

we propose using different scaling factors (C in Equation 3.53) according to crowd density.

Our approach to crowd density estimation is related to the work by Marana *et al.* [78]. This approach is based on the hypothesis that the presence and density of persons in a scene changes the texture of the scene. More specifically, a crowded scene typically results in a finely textured image. In contrast, the image of the same scene with fewer persons would be more coarse-textured or untextured. We characterize the texture content of the image with features extracted from the gray level co-occurrence matrix (GLCM).

The GLCM proposed by Harailck *et al.* [106] is a statistical method to describe texture feature. A GLCM is, essentially, a two-dimensional histogram of the number of times that pairs of intensity values occur in a given spatial relationship [106]. This spatial relationship is defined by a displacement vector, $\mathbf{d} = (d, \theta)$.

The GLCM $P_{\mathbf{d}}(i, j)$ is 256×256 (the image is 8-bit and not quantized) and each entry of the GLCM is the number of times that the intensity values pair i and j which are separated by a distance \mathbf{d} occurs:

$$P_{\mathbf{d}}(i, j) = |\{(u, v), (u, v) + \mathbf{d}\} | I(u, v) = i, I((u, v) + \mathbf{d}) = j\} | \quad (3.54)$$

where (u, v) are the pixel locations in the image $I(\cdot)$ and $|\cdot|$ denotes the cardinality of the set. The matrix should be normalized and sum to 1. Several descriptors can be obtained from GLGM:

1. Energy

$$f_{\mathbf{d}}^1 = \sum_{i,j=1}^{N_g} P_{\mathbf{d}}(i, j)^2 \quad (3.55)$$

2. Contrast

$$f_{\mathbf{d}}^2 = \sum_{i,j=1}^{N_g} P_{\mathbf{d}}(i, j)(i - j)^2 \quad (3.56)$$

3. Entropy

$$f_{\mathbf{d}}^3 = \sum_{i,j=1}^{N_g} -P_{\mathbf{d}}(i, j) \log P_{\mathbf{d}}(i, j) \quad (3.57)$$

4. Homogeneity

$$f_{\mathbf{d}}^4 = \sum_{i,j=1}^{N_g} \frac{P_{\mathbf{d}}(i,j)}{1 + |i - j|} \quad (3.58)$$

We construct the GLCM by considering pixel pairs differing by 1 intensity level. Further, we repeat the process for four spatial orientations corresponding to right (0°), top-right (45°), top (90°), and top-left (135°) neighbors. These spatial relationships are also represented in terms of the horizontal and vertical displacement as $(0,1)$, $(-1,1)$, $(-1,0)$, and $(-1,-1)$ and they are shown in Figure 3.13. For each of the four matrices $P_{0,1}, P_{-1,1}, P_{-1,0}, P_{-1,-1}$ we obtain four statistical features defined in above – energy, entropy, homogeneity, and contrast. We concatenate these features to represent the texture by a single vector τ in a 16 dimensional feature space.

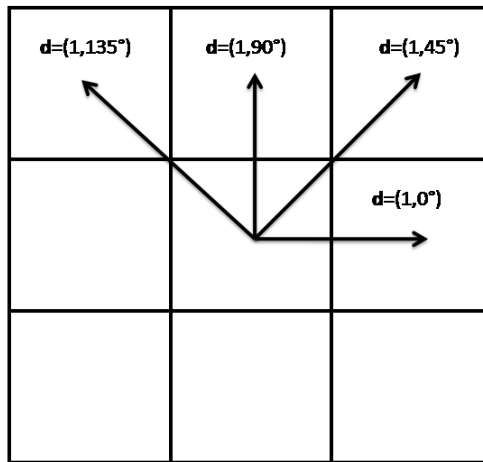


Fig. 3.13. Four spatial relationships for GLCM.

Consider a video sequence in which crowd densities occur at k levels such as “sparse,” “low,” and “very high.” We obtain the feature vectors $\tau_1, \tau_2, \dots, \tau_k$ through a training process detailed in Section 4.3.1. While classifying the density level for a

test frame, we estimate the distance between the test frame's feature vector τ_{test} and the k reference vectors. The density level is then determined by a nearest neighbor rule. Let l represent the crowd density level for a given frame. Then,

$$l = \arg \min_i d(\tau_{test}, \tau_i), \quad (3.59)$$

where $d(\cdot)$ is the distance function and $i \in \{1, 2, \dots, k\}$. We use the Euclidean distance after normalizing the values of each feature to approximately $[0.0, 1.0]$. The goal of this scaling process is to make the 16 dimensions comparable rather than enforce probability-like constraints. Thus, some values may lie outside of the range which is acceptable. The normalization is achieved using the maximum and minimum values of the features from the training frames. Since the scaling factor depends primarily on the crowd density near the trip wire, we estimate the texture features only in a subset G of the frame such that $G = \{f(x, y) | (x, y) \in \text{ROI}\}$. This region of interest (ROI) is specified during the training process and contains the trip wire ($\mathfrak{R} \subset G$).

Note that for a sequence with k crowd density levels, there should be k scaling factors C_1, C_2, \dots, C_k to be used for flow estimation. If the density level for frame n_i is $l(n_i)$, then the final flow estimation is obtained by combining the steps in Equation 3.52 and Equation 3.53 as follows:

$$\nu_T = \sum_{i=0}^{N-1} \sum_{x, y \in \mathfrak{R}} \frac{I_{n_i}(x, y) \times w_{n_i}(x, y)}{C_{l(n_i)}}. \quad (3.60)$$

The experimental results of our proposed methods are discussed in detail in the next chapter.

4. EXPERIMENTAL RESULTS

4.1 Moving Object Detection

In this section, we will use test sequences with background dynamics such as waving tree leaves and sudden illumination change to examine the performance of our moving object detection methods. Sources of the video sequences used in our experiments include publicly available datasets, video sequences obtained from the Purdue University Police Department, and sequences we acquired in the VIPER laboratory.

We will show qualitative results of moving background subtraction using the intensity based foreground/background pixel classification and its comparison with Gaussian Mixture Model (GMM) [10]. We also compare the execution time of our proposed intensity based foreground/background pixel classification method with GMM.

For sudden illumination change, we will use test sequences under extreme illumination change (including one of the the most challenging publicly available datasets for sudden illumination change “Wallflower - light switch”) for our proposed illumination model based statistical method. We also show the speed gain comparison of our proposed fast implementation of Gaussianity test with the naive implementation of Gaussianity test.

4.1.1 Intensity-Based Background Subtraction

Figure 4.1(a) shows a frame in a sequence taken by a surveillance camera installed in a building on the Purdue University campus. A corresponding binary foreground mask is shown in Figure 4.1(b). The mask is obtained by the intensity-based background subtraction method using Equation 3.6. A histogram of the difference image and the adaptive threshold determined using the method described in Section 3.1.4

are shown in Figure 4.1(c). Most of the pixel differences in the histogram are close to zero as a result of differences between the same background region. There is another peak near pixel value 110 which comes from the differences between the foreground pixels of the current frame and the background model. It takes 3 - 5 iterations on average to compute the threshold for each frame to determine the adaptive threshold.

Figure 4.2 and Figure 4.3 show the experimental results and comparisons with GMM. In both figures, the top row in the figure shows the original current frames, the second row shows the background models, the third row shows the binary foreground masks using our proposed method, the fourth row shows the background models using GMM, and the last row shows the binary foreground masks using the GMM method. The GMM method requires $1/\alpha$ number of frames to construct a background that does not have the foreground contamination. This means, for example, if the learning rate is 0.01, then GMM will need 100 frames to construct a background that does not have any “history” of a person passing by.

As shown clearly in Figure 4.2, the background model using our proposed method is not contaminated by foreground pixels because the learning rates for the background update for the uncovered region have very small values. This is because the pixels are just uncovered, which means the number of consecutive frames the pixel being a background pixel is very small, and hence α_2 in Equation 3.10 has a very small value. This value will increase as the uncovered background pixels become more stable over time.

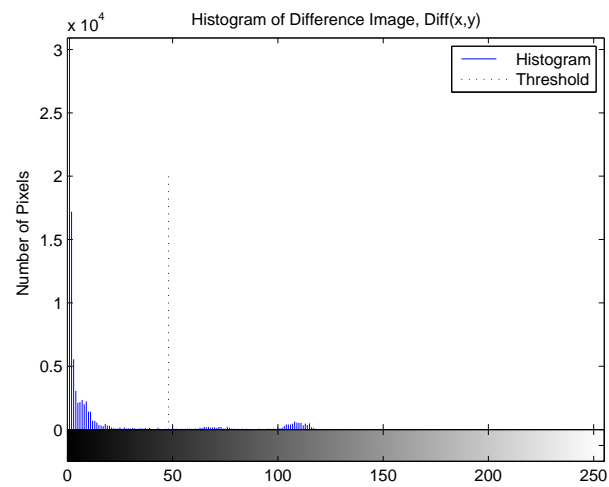
Figure 4.3 shows the result of a sequence taken in an outdoor scene, with many waving tree leaves and bushes. Our proposed temporal background pixel count eliminates most, but not all, of the false alarms (false positives) from the moving tree leaves, as shown in frame 139. This is because there are pixels of tree leaves have been static (remain as background pixels) for a long period of time and the α_2 has a high value in this case. The pixels are detected as foreground pixels when they start moving again.



(a)



(b)



(c)

Fig. 4.1. (a) Current frame (b) Binary foreground mask (c) Histogram of the difference image and the adaptive threshold

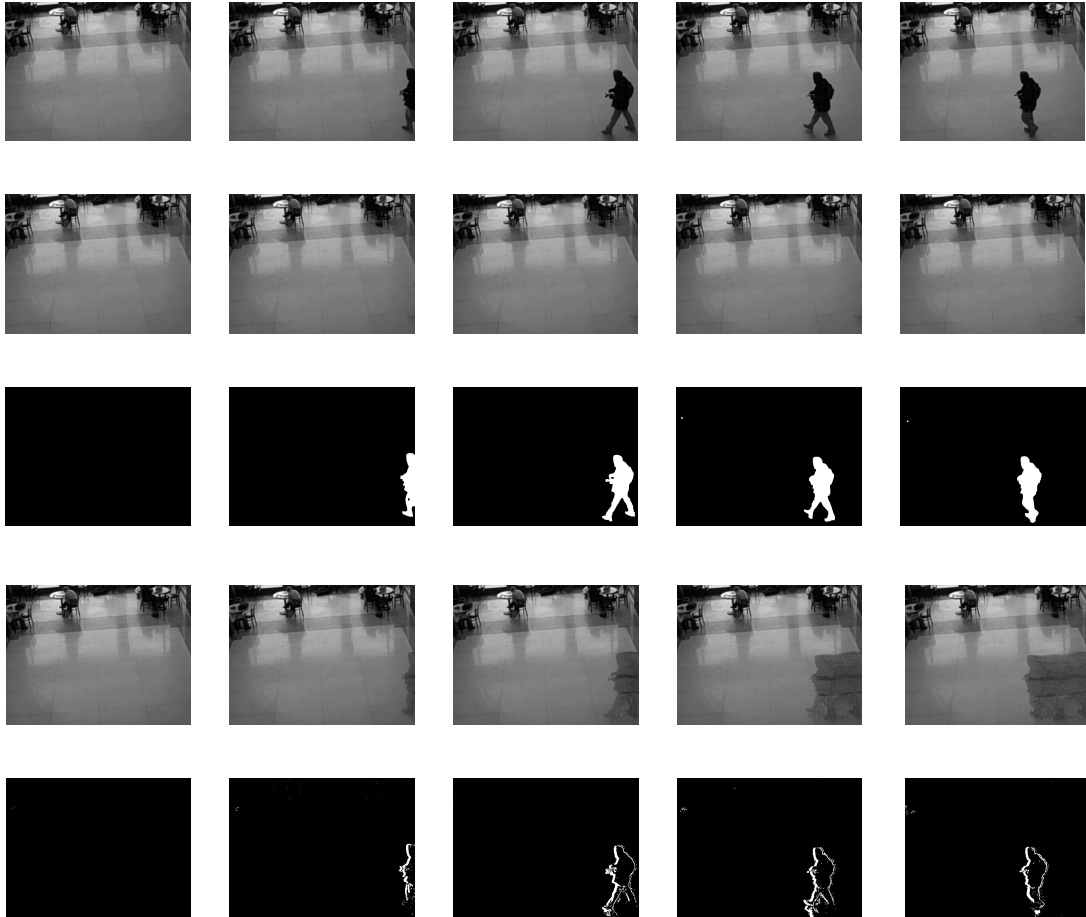


Fig. 4.2. The 5 columns show frame 75, 85, 96, 110, 120 of a sequence. Row 1: original frames. Row 2: background models of our proposed method. Row 3: binary foreground masks of our proposed method. Row 4: GMM background models. Row 5: GMM foreground masks.

4.1.2 Computational Complexity Comparison

We observe that our method has a much lower computational complexity compared to a typical GMM-based method [10]. Table 4.1 shows the execution time comparison of our method and a GMM-based method. Execution time is an estimate of computational complexity. However, it should be noted that while the actual time durations depend on the computing platform, their relative magnitudes provide a direct comparison of their complexity.

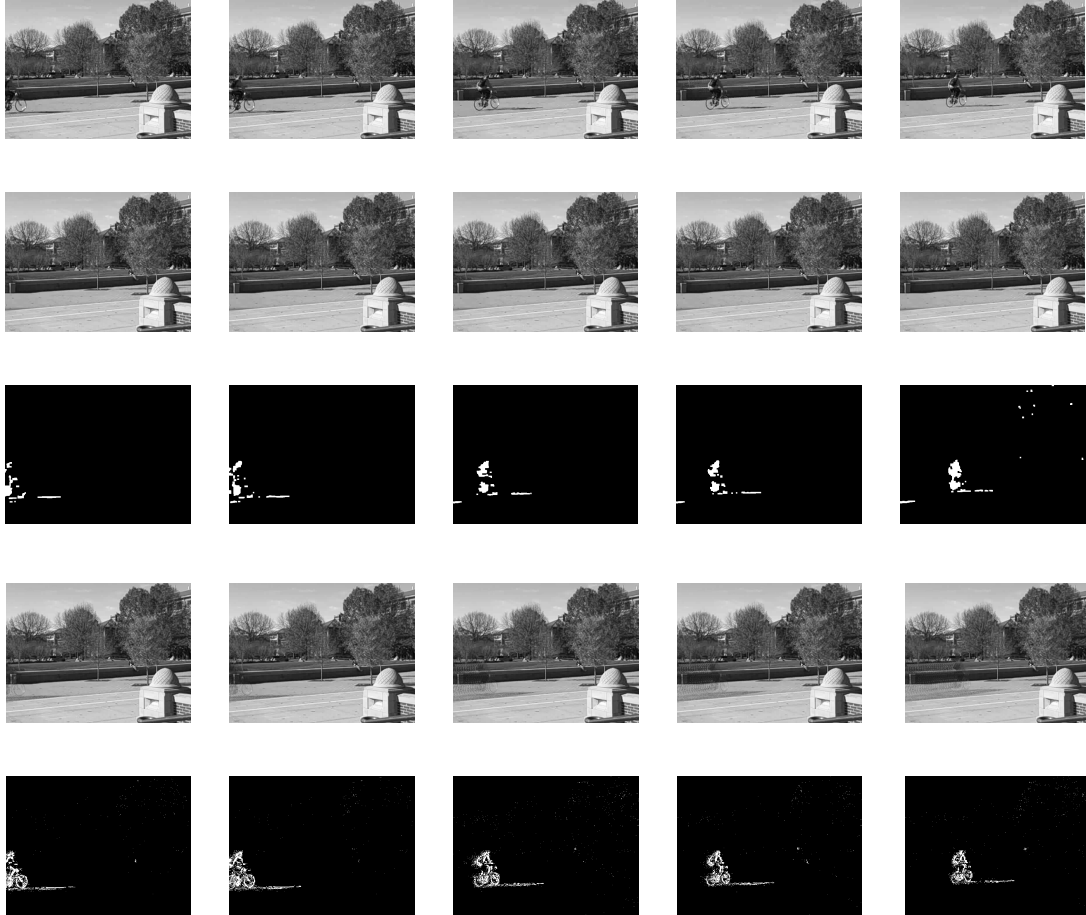


Fig. 4.3. The 5 columns show frame 113, 115, 126, 130, 139 of a sequence. Row 1: original frames. Row 2: background models of our proposed method. Row 3: binary foreground masks of our proposed method. Row 4: GMM background models. Row 5: GMM foreground masks.

4.1.3 Statistical-Based Background Subtraction

Figure 4.4 shows the results of 4 frames taken from the publicly available dataset Wallflower [35]. The sequence is known as “light switch” and it is known to be one of the most challenging sequences for sudden illumination change. The sequence has a person entering a dark room, switching on the light, and sitting down. The first column shows the results of our background model which is very dark, the second column shows the current frames and are taken from shots after the light is switched

Table 4.1
Table of execution time of our method and a GMM-based method

Sequence	GMM	Our Proposed Method
A	1008	191
B	983	189
C	1881	331
D	2150	395
E	2029	393

on. The third column shows the generated foreground mask using our proposed method. Now consider the first frame shown on the top row, the result has three sources of false detections, 1) the door slit is detected as a false positive, 2) the monitor is detected as foreground and is a false positive, and 3) the person's face is not detected and is an example of false negative. Our proposed method works based on the assumption that the surface is Lambertian (perfectly diffuse) and clearly this does not hold for non-Lambertian surface such as human skin. Another assumption is that the irradiance should be a constant for every pixel in a frame. This assumption also does not hold when there is light coming from the other room through the door slit. The computer monitor is a light source so again it is a non-Lambertian surface. Our proposed method fails when at least one of these assumptions does not hold.

The results of the wallflower sequence [35] is also compared with the ground truth that is segmented by hand. Two metrics for characterizing the Detection Rate (DR) and the False Alarm Rate (FAR) are used and they are defined as follows:

$$DR = \frac{TP}{TP + FN} \quad (4.1)$$

and

$$FAR = \frac{FP}{TP + FP} \quad (4.2)$$

where



Fig. 4.4. Wallflower dataset light switch sequence. Column 1: background models. Column 2: current frames. Column 3: generated foreground mask using our proposed method.

- TP (true positive): detected regions that correspond to moving objects
- FP (false positive): detected regions that do not correspond to a moving object, (also known as false alarms).
- FN (false negative): moving objects not detected. (also known as misses)

The results are shown in Table 4.2.

Table 4.2
Table of Detection Rate and False Alarm Rate

	Our Proposed Method	GMM
DR	0.8201	0.7006
FAR	0.0613	0.9425

Figures 4.5 and 4.6 show the results of two other sequences which have extreme sudden illumination change. Both sequences are acquired in the VIPER lab and both have grayscale intensity changes of more than 100. The sequence in Figure 4.6 is a very challenging test sequence. A person appears in the scene (as shown in Figure 4.6b) but it is so dark that even human observers can barely notice the person. Similar to the previous sequence, human skin in both frames are missed (not detected). Other false positive detections in the frames are errors due to specular reflections in the scene.

4.1.4 Fast Implementation

To investigate speed improvements for the Gaussianity test using an integral moment implementation, we use a test image of 512×512 , and determine $H(x, y)$ for every pixel of the image using block sizes ranging from 5 to 65. The execution time for both naive implementation and integral moments implementation, and the speed improvements are shown in Table 4.3. The speed improvement increase as the block size increases. This is due to the integral image approach always determines the sum of an area using four memory accesses regardless of the size of the area to be summed. However, with the naive implementation, the computational complexity increases as the size of the area to be summed increases.

Table 4.3
Execution time in seconds

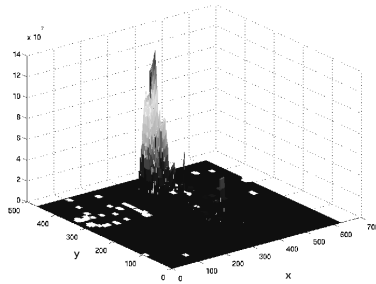
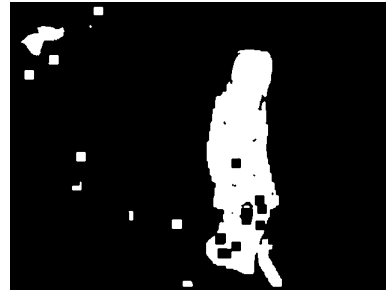
Block Size	Naive Implementation	Fast Implementation	Speed Improvement
5×5	1.667873	0.208484	8
9×9	4.376945	0.198952	22
17×17	9.464206	0.172412	55
33×33	29.345497	0.154037	191
49×49	47.419024	0.127780	371
65×65	86.129788	0.119752	719



(a) Background model .



(b) Current frame.

(c) 3D Plots of H .

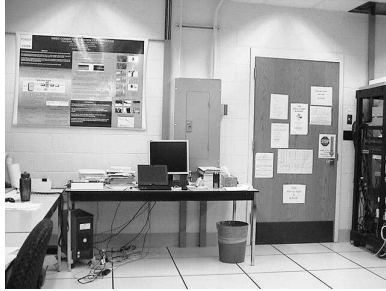
(d) Detection results.

Fig. 4.5. (a) Background model with average grayscale intensity = 156. (b) Current frame with average grayscale intensity = 55. (c) 3D Plots of H . (d) The pixels detected as foreground are white.

4.2 Object Tracking

4.2.1 Effect of Subsampling on Histogram Construction

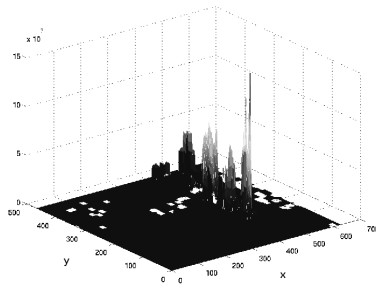
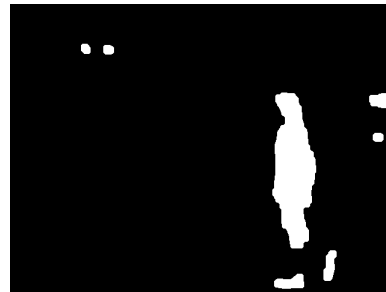
In order to test the effect of using only a subset of pixels in the color histogram construction on object tracking performance, we use a still image from a video sequence and define a region of interest in the image for histogram construction. The testing region of interest has a human body that is enclosed by an elliptical tracker. A color histogram constructed using all pixels in the ellipse is constructed and is used as a control for comparison. Two histograms using only a subset (i.e. one half and one quarter) of the pixels are constructed and are compared with the control histogram.



(a) Background model .



(b) Current frame.

(c) 3D Plots of H .

(d) SM detection results.

Fig. 4.6. (a) Background model with average grayscale intensity = 157. (b) Current frame with average grayscale intensity = 32. Note that there is a person in front of the door. (c) 3D Plots of H . (d) The pixels detected as foreground are white.

	All Pixels	Half of the Pixels	One Quarter
<i>execution time(sec)</i>	0.1304	0.0498	0.0252
ρ	1	0.9998	0.9993
<i>distance</i>	0	0.0157	0.0258
$\mathcal{L}(z x)$	1	0.9946	0.9853

Table 4.4
Similarity between histograms with $\sigma = 0.15$.

The comparison is done in terms of the Bhattacharyya coefficient ρ using Equation 3.35, Bhattacharyya distance using Equation 3.36, the observation likelihood using Equation 3.41, and computational time.

The results is shown in Table 4.4. For the case with only half of the pixels being used, the computational complexity was decreased by about 50%. With ρ and the distance between the histogram with all pixels being used and the histogram with only one half of the pixels being used were very close to 1 and 0 respectively. For the case with only one quarter of the pixels being used, the computational complexity was decreased by about 25%, with ρ and distance very close to 1 and 0 as well.

It is shown in our experiments that this approach works well and takes advantage of the spatial redundancy of natural images in that their neighboring pixels are highly correlated.

4.2.2 Experimental Results

In all our experiments, the number of particles N is set to be 500 for each frame. Sources of the video sequences used in our experiments include publicly available datasets (CAVIAR dataset [107] taken with one camera with ground truth provided), video sequences obtained from the Purdue University Police Department, and sequences we acquired in the VIPER lab. The dataset has a wide variety and it consists of:

- Different frame sizes
- Different frame rates
- Moving objects: human or vehicles
- Date time and night time
- Indoor or outdoor
- Scenes with high or low illumination
- Moving and static camera
- Different camera views (aerial, ground, rooftop)

- With and without occlusion
- Single or multiple objects

Figure 4.7 shows a person walking indoor from right to left in a school classroom. The sequence is taken with one camera which is static from the ground view. It is a color video with frame size 384×288 , and frame rate of 25 fps. The person is tracked very accurately from the frame he/she enters the scene until he/she leaves the scene.

Figure 4.8 shows a bus traveling on the road approaching the camera. The sequence is taken with one camera which also has slightly shaky camera movement from a rooftop view. It is a color video with frame size 384×288 and frame rate of 25 fps. The perspective is changing as the bus is approaching the camera (the scale is increasing). The vehicle is tracked very accurately from the frame it enters the scene until it leaves the scene.

Figure 4.9 shows the tracking of a person walking in an indoor scene. The sequence is taken with one camera which is static from the aerial view. It is a color video with frame size 640×480 and frame rate of 25 fps. The person is tracked very accurately from the frame he/she enters the scene until he/she leaves the scene.

Figure 4.10 shows a person walking in an outdoor scene during daytime it is from a real video surveillance dataset. The sequence is taken with one single camera which is static, from the rooftop view. It is a color video with frame size 704×480 , and frame rate of 25 fps. The person is tracked very accurately from the frame he/she enters the scene until he/she leaves the scene.

Figure 4.11 is a sequence from the publicly available dataset CAVIAR. The sequence shows a person walking in an indoor scene. The sequence is taken with one wide angle camera lens in the entrance lobby of the INRIA Labs at Grenoble, France. The camera is static and the sequence is from a rooftop view. It is a color video with frame size 384×288 and frame rate of 25 fps. The person is tracked very accurately from the frame he/she enters the scene until he/she leaves the scene.

Figure 4.12 shows an example where the tracking method fails when there is a complete occlusion. The test data is a sequence of an outdoor scene in a real video surveillance dataset. The sequence is taken with one camera which is static from a ground view. It is a color sequence with 352×240 (SIF) resolution and frame rate of 25 fps. The car to be tracked is a white sedan, it is then completely occluded by a black truck. During this period of occlusion, none of the particles have small Bhattacharyya distance because not even part of the car is visible. Low importance weights are assigned to all of the particles according to the likelihood function as a result of the occlusion. The estimate ended up getting “stuck” in the sky and loses track because the sky has a relatively similar appearance to the white car. This is a typical example of particle filter based object tracking failure.

Figure 4.13 shows an example with multiple object tracking. The test data is an outdoor scene during daytime. The sequence is taken with one camera which is static from a ground view. It is a color sequence with 640×480 resolution and a frame rate of 25 fps. Note that the experiment is set up using two individual particle filtering object trackers and the two trackers work independently.

The effect of the observation likelihood function using dynamic parameter setting in Equation 3.43 was also investigated. In Figure 4.14, the left column shows three histograms of distances and three likelihood functions using 3 different σ s are overlaid on the same figure. The three distributions of the importance weights result from the 3 likelihood functions are shown on the right column in the same Figure. The top row is an example of a likelihood function which has a value of σ which is too small, as a result, the majority of the particles are considered as bad and are assigned very low importance weights. The second row shows an example of a likelihood function which has a value of σ which is too high, as a result, majority of the particles are assigned with equally high importance weights and hence the likelihood is not discriminative enough. The bottom row shows the σ that is set dynamically using the method described in Section 3.2.2. A significant portion of particles are assigned with high

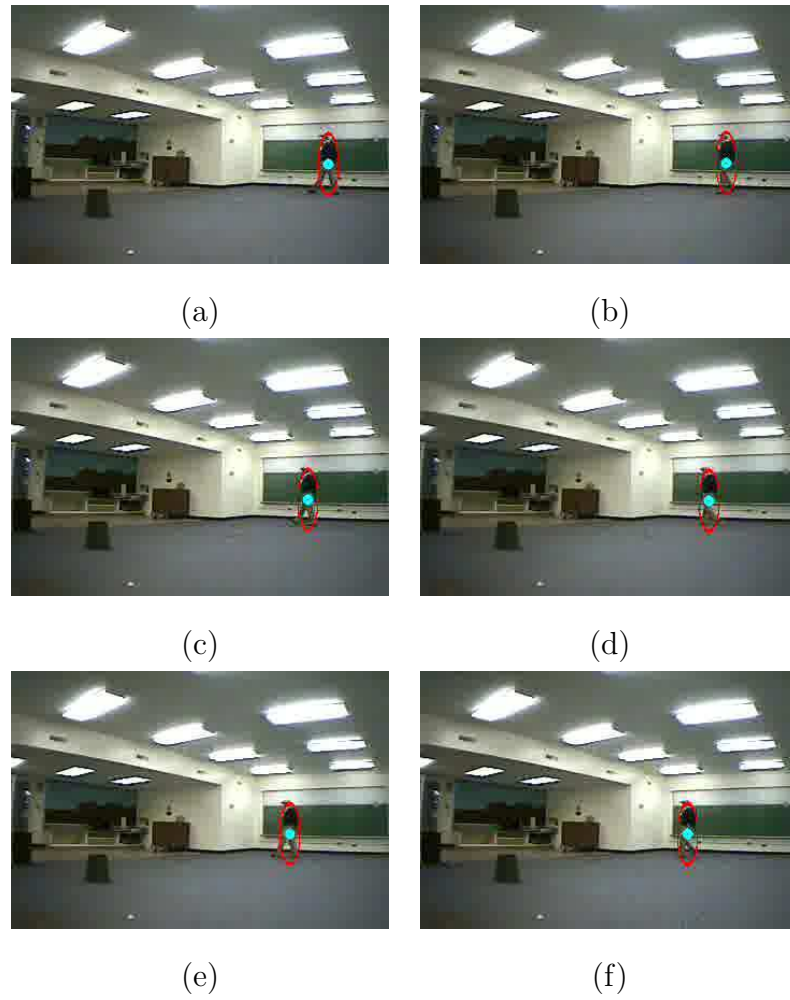


Fig. 4.7. Tracking results of a low-illumination sequence with a person walking, the tracked object is circled: frame number (a)45 (b)55 (c)65 (d)75 (e)85 (f)95.

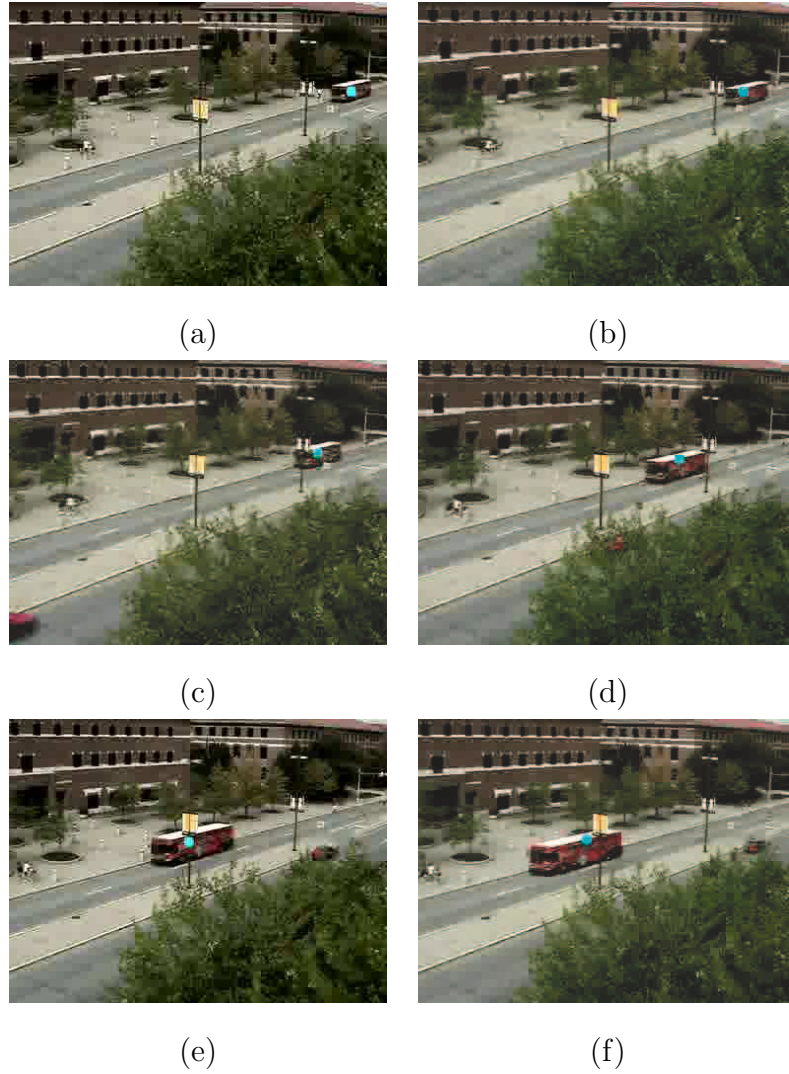


Fig. 4.8. Vehicle tracking results of a bus; frame number (a)30 (b)40 (c)50 (d)75 (e)102 (f)110.

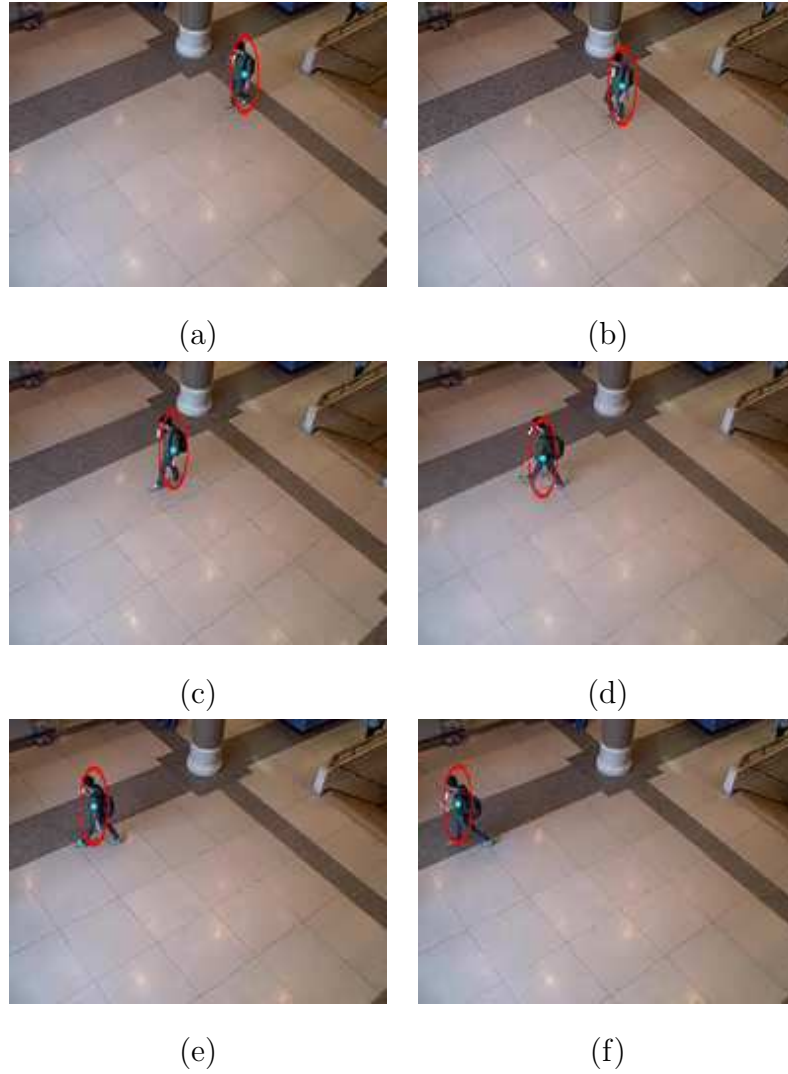


Fig. 4.9. Tracking results of an indoor sequence with a person walking, the tracked object: frame number (a)20 (b)40 (c)60 (d)80 (e)100 (f)120.

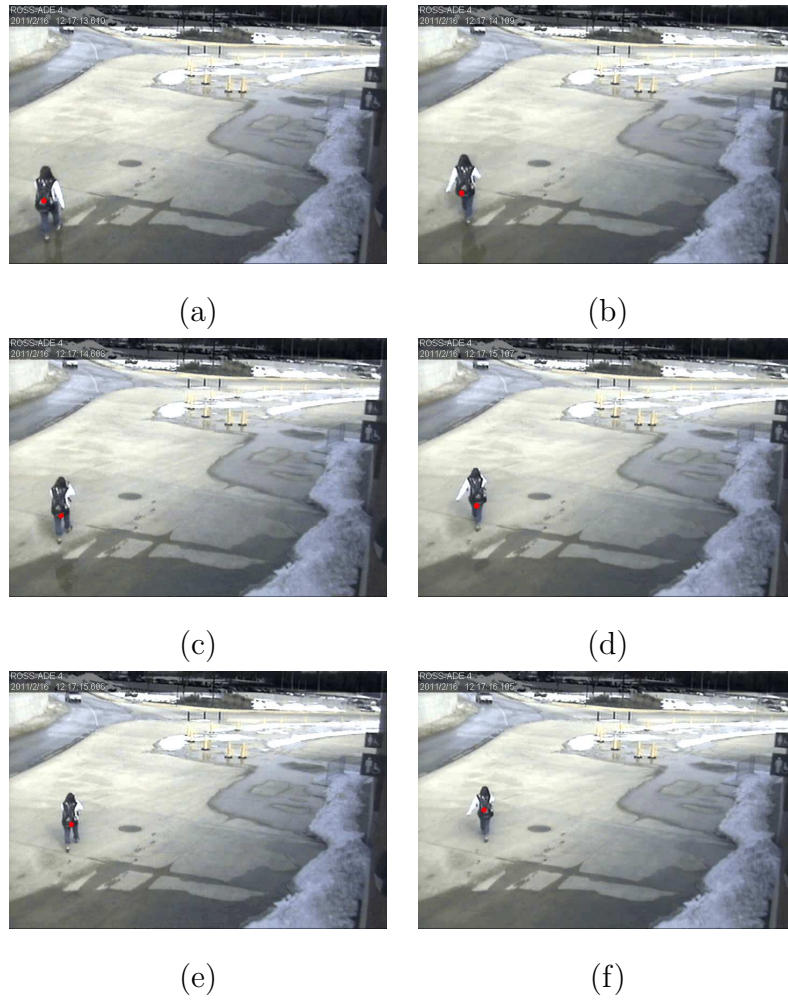


Fig. 4.10. Object tracking of an outdoor sequence with a person walking; Frame number (a)5 (b)10 (c)15 (d)20 (e)25 (f)30.

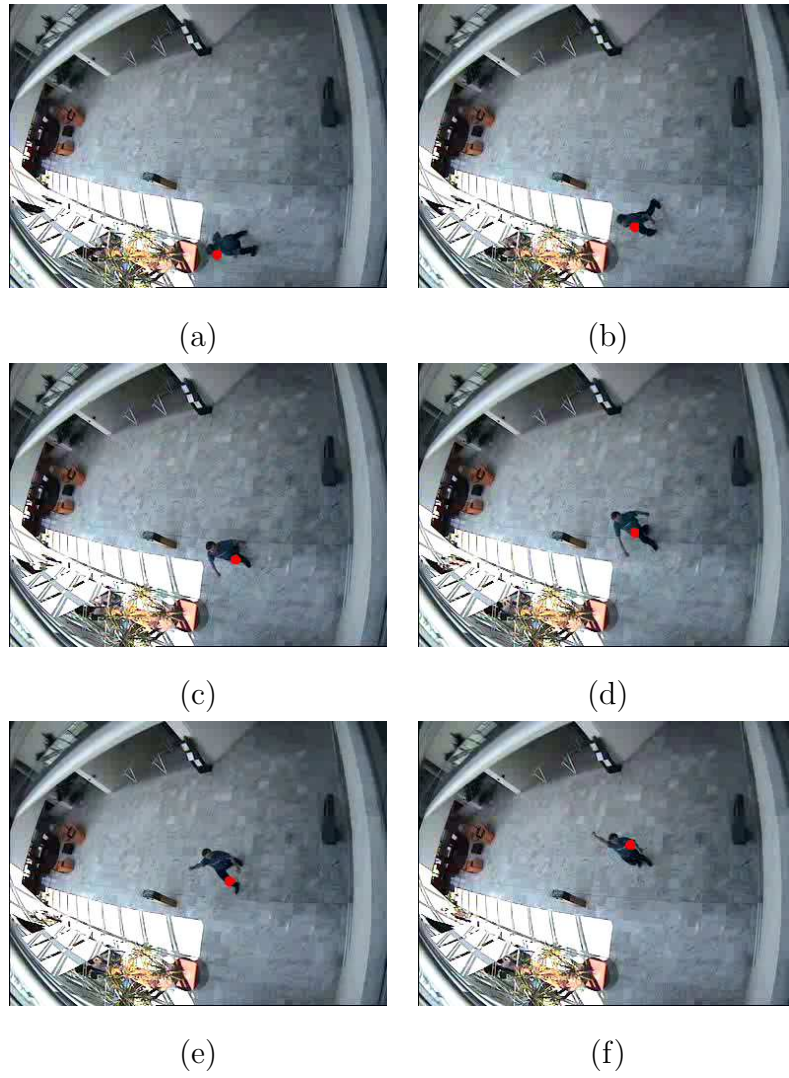


Fig. 4.11. CAVIAR dataset with a person walking and raising his/her hand.

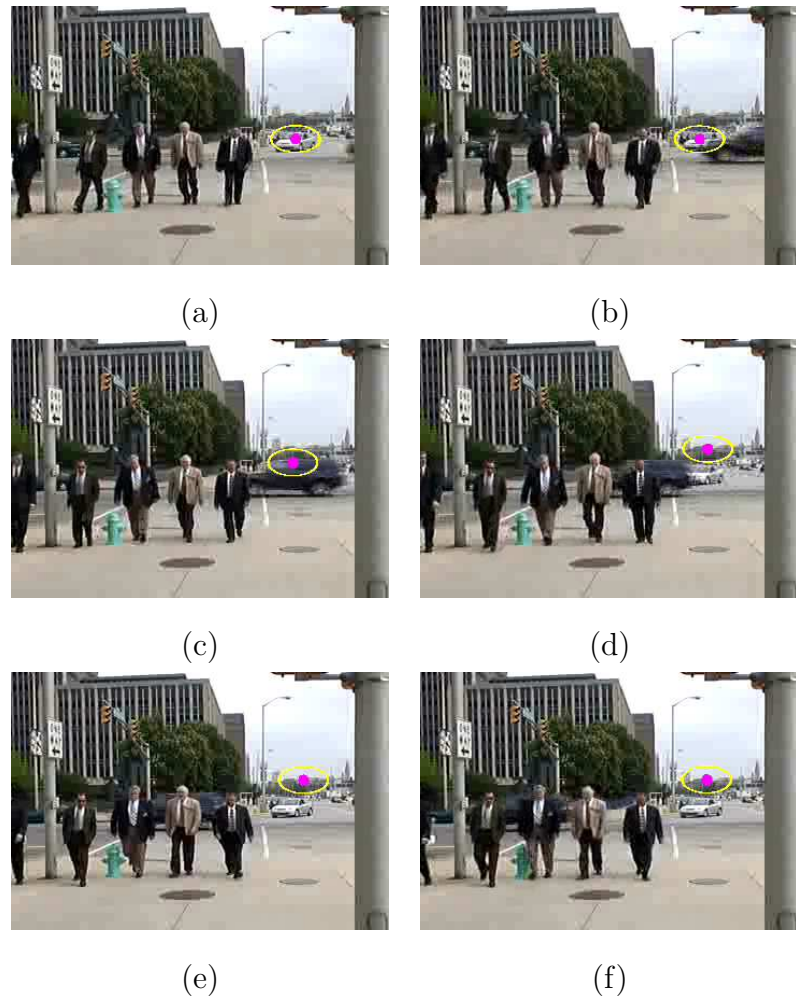


Fig. 4.12. An example where the tracking method fails when there is a complete occlusion.

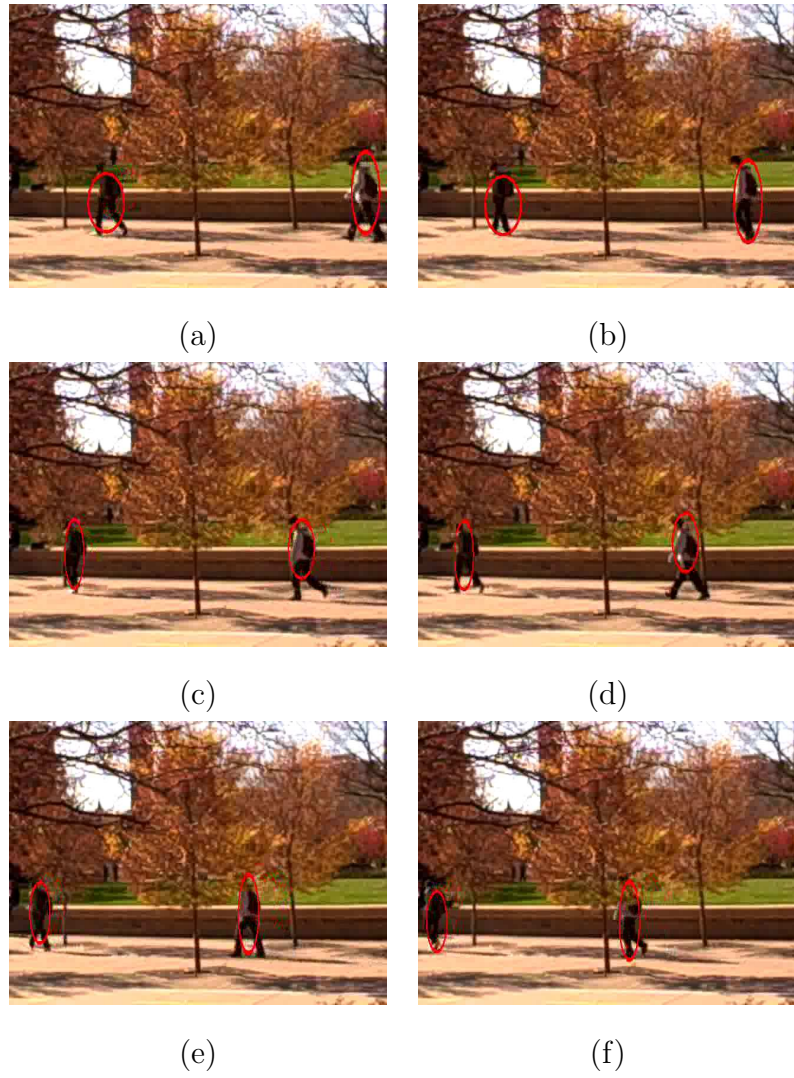


Fig. 4.13. A sequence with multiple objects being tracked.

weights and another significant portion of particles are assigned with lo weights. The likelihood function is more discriminative with a dynamic σ .

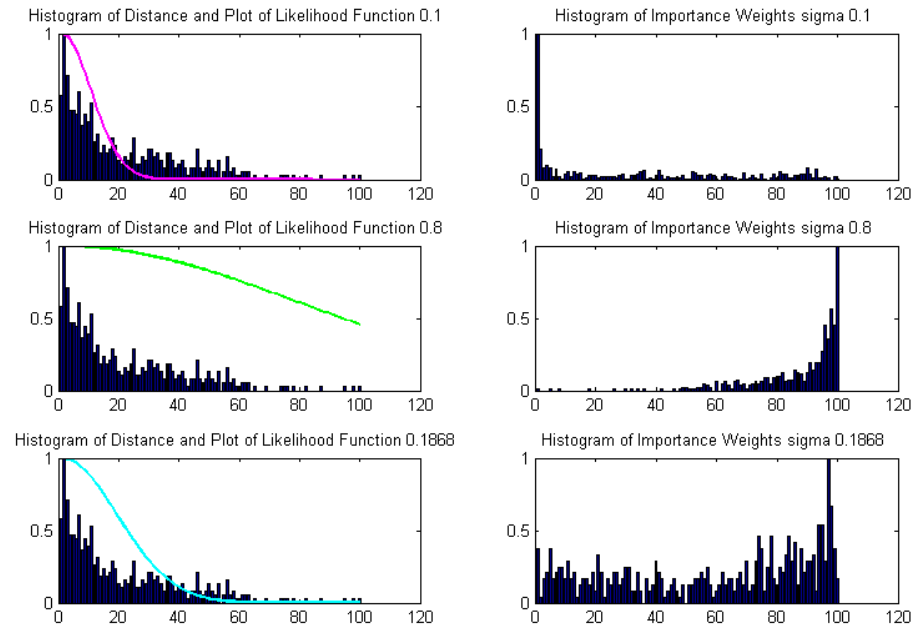


Fig. 4.14. Left: Histograms of distances and likelihood function with 3 different σ s Right: Histograms of importance weights resulted from different σ s

4.2.3 Implementation on the Nokia N810 Internet Tablet

We implemented our object tracking methods on a hand held device, the Nokia N810 Internet Tablet. The Nokia N810 is a portable tablet computer with a 400 MHz ARM processor. It runs a Linux-base operating system known as Maemo. The specifications of the device are:

- Processor: 400 MHz TI OMAP 2420
- Memory: 128 MB Random access memory
- Display: 800×480

- Camera: 640×480 VGA

A picture of the device is shown in Figure 4.15.



Fig. 4.15. The Nokia N810 Internet Tablet Computer.

The detail of the implementation is described as follows:

- Particle filtering described in Section 3.2.1 is used
- Only color feature (no edge) described in Section 3.2.1 is used
- Dynamic parameter setting described in Section 3.2.2 is used
- Pixel sub-sampling for color density estimation described in Section 3.2.3 is not used
- Number of particles used is 100

With the above implementation the device executes our object tracking method at approximately 5 frame/second.

4.3 Crowd Analysis

4.3.1 Training and Testing for Crowd Density Estimation

For each test sequence, a trip-wire \mathfrak{R} and a region of interest G are specified with graphical user input. Training with user input is required for two purposes – 1) to associate the texture features extracted from the GLCM of the ROI with different levels of crowd density and 2) to associate the number of foreground pixels on the trip-wire with the number of persons for every crowd density level. The ROI is chosen such that the trip-wire is bounded by it. An example of the trip-wire and the ROI is shown in Figure 4.16.



Fig. 4.16. An example of the trip-wire and ROI.

Training of Texture Features

To associate the texture features extracted from the GLCM of the ROI with different levels of crowd density, a training sequence is used and T frames are randomly selected from it. The ROI from each of the T frames is displayed to the user who

then assigns a label l to the frame which serves as the ground truth. The ROI of each labeled frame is used to determine the GLCM and the 16-dimensional texture feature vectors t_{n_i} (where n_i is the index of the i^{th} training frame and $i = 1, 2, \dots, T$) are extracted from each GLCM. After all T feature vectors are determined, the average of the set of feature vectors associated with the j^{th} texture class is used as the “model” of the j^{th} density level:

$$\tau_j = \frac{\sum_{i=1}^T t_{n_i} \times \delta[l(n_i) - j]}{\sum_{i=1}^T \delta[l(n_i) - j]}. \quad (4.3)$$

Here $\delta(x)$ represents an indicator function (or a Kronecker delta function) which is zero everywhere except at $x = 0$ where it takes unit value.

Training of Foreground Pixel Counts

After constructing the texture models for different crowd density levels, the crowd density level of each frame in the entire testing sequence is determined using the models according to Equation 3.59. A plot of the result of this step is shown in Figure 4.17. This plot also serves as an illustration for the training of foreground pixel counts. A “stable” period of the plot is chosen for foreground pixel counts training of each texture class. A set of consecutive frames is considered “stable” if they all have the same texture classification.

Background subtraction is used to determine the foreground regions and this process is done in the trip-wire region only. Hence the method has a very low computational cost. The accumulative foreground pixels \tilde{S}_T are counted for the entire “stable” period. Simultaneously, the frames are displayed to the user who counts the number of persons ν_T crossing the trip wire (which can be fractional). These values are used to estimate the scaling constant C_l for each texture class (or crowd density level) l .

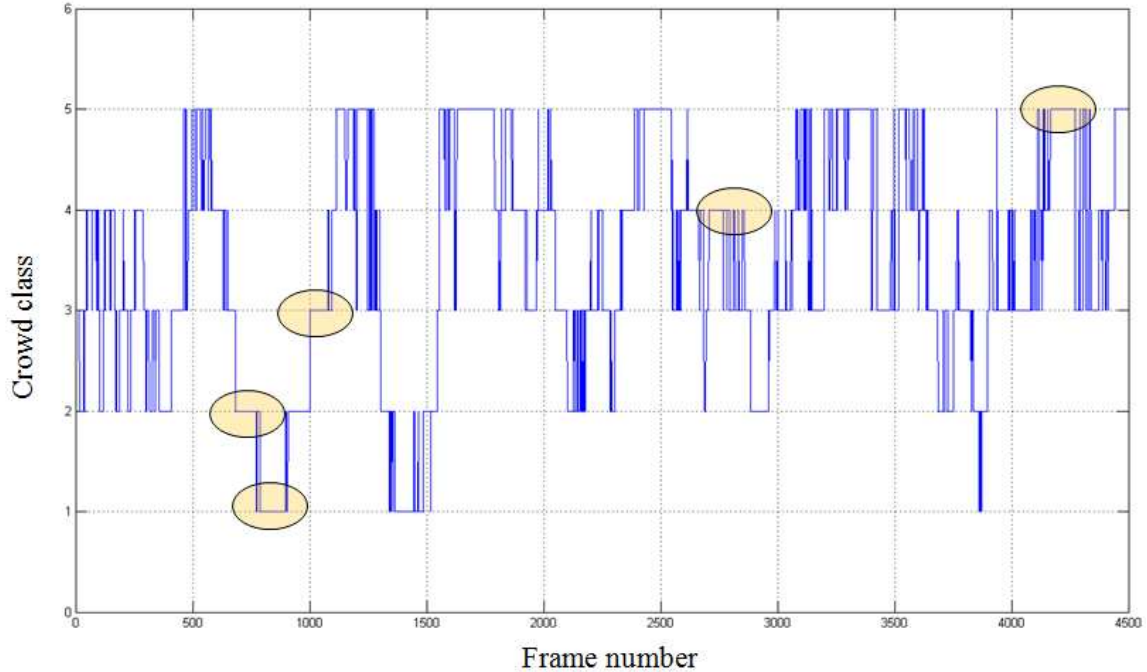


Fig. 4.17. A plot of a five-class texture classification on a test sequence. The “stable” periods are circled.

Testing: Finding the Number of Persons

Given a frame of a testing sequence the GLCM of the ROI is first determined and the texture features are extracted from it. The crowd density level is then determined using the models of different texture classes from the training process. Given a crowd density level, the number of foreground pixels on the trip-wire is determined using background subtraction. Finally, the number of persons is determined according to Equation 3.60. This process is illustrated in Figure 4.18.

4.3.2 Crowd Flow Results

The methods described above were tested on surveillance videos most of which are obtained from publicly available datasets [108, 109]. The UCF data was used by

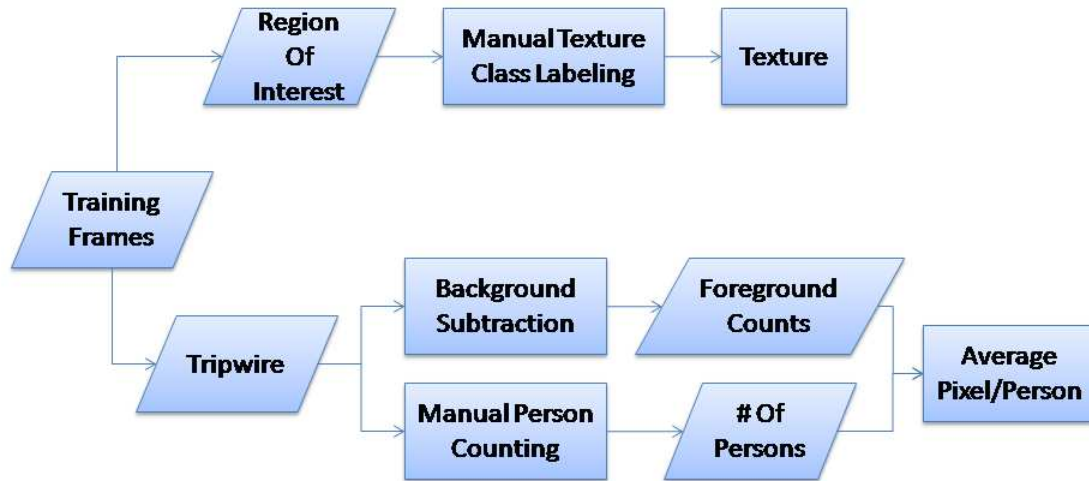


Fig. 4.18. A block diagram illustrating the training process.

Ali and Shah to test their methods for tracking subjects in crowded scenes [110] while the UCSD dataset is designed for detecting anomalies in pedestrian crowds [111]. We present the results in two parts. First, we show the effectiveness of texture based classification for crowd density level estimation. We then present the final output of our flow estimation system for several sequences.

Figure 4.19 shows examples of video frames of different crowd density levels from two test sequences. The density levels (estimated and ground truth) are provided in the captions. Note that bigger labels are used to represent higher density crowds. Also note that the crowdedness levels are decided relative to each sequence. Thus, the class 2 frames of the two sequences have very different density of persons.

The results of final flow estimation are presented in Table 4.5. We provide the length of the test sequence, the number of persons crossing the trip wire (estimated and ground truth), and the number of crowd density levels seen in the duration of the



Fig. 4.19. Examples of crowd density estimation on three video sequences. The estimated and true (in parentheses) density levels are (left to right) – Top: 3(3), 1(1), 2(2), Second: 1(1), 3(3), 4(4), Third: 2(2), 3(3), 1(1), Fourth: 1(1), 1(1), 2(2), Bottom: 1(1), 1(1), 1(1).

test. In most cases the estimate is close to the true flow irrespective of the change in crowd density levels. Furthermore, there is no systematic over- or under-estimation in our method.

Table 4.5
Testing results for crowd flow estimation.

Seq	Frames	Estim Flow	True Flow	Density Levels
1	200	16	19	1,2,3,4,5
2	100	1.0	1	1,2
3	100	7.9	6.5	1,2,3
4	200	25.5	24	1,2,3,4
5	110	9.1	8.5	2,3,4
6	110	4.7	5	1,2

The results above are tested on sequences between 4 and 20 seconds in length. While our method would work equally well on longer sequences, it has been designed to be particularly useful in real-time surveillance situations where instantaneous flow might be more important than an average estimated over a long time. Our test sequences also capture the difficult cases where crowd density fluctuates heavily (as shown in column 5 of Table 4.5).

It should be noted that the accuracy of the methods is sensitive to the training of texture models. More specifically, the random selection of frames affects how well the texture models represent each class. We observe that choosing more number of crowd density levels can result in higher accuracy in flow estimation but the gain in accuracy becomes smaller with very large number of levels. Also the ROI is selected such that it can represent the texture of the local density near the trip-wire. It should be large enough to contain multiple persons around the trip wire. However, a very large ROI would incur unwanted influence from persons far from the trip wire and would increase computational cost.

Our method obtains the scaling constant for each texture level from training. This constant can be interpreted as the average pixel per person pass through the trip wire

for each texture level. Consider the case in a training video, when a person walk pass a trip wire at an average speed, at a certain frame rate, say 25 fps, for this trip wire of a fixed size, it requires 1 second (25 frames) for the person to walk pass the trip wire completely. We would obtain the number of accumulated foreground pixels in these 25 frames, say 500 pixels. If we use only these 25 frames for training, then the scaling constant would simply be 500. But if a person runs pass the trip wire, it would take less time (and hence less number of frames) to pass the trip wire completely. Less number of accumulated foreground pixels (say 200) would be estimated as a result of this. With our proposed method, the estimated number of persons in this case would be $200 \times \frac{1}{500}$ which is 0.4 and is underestimated.

5. CONCLUSIONS AND FUTURE WORK

In this thesis we developed several methods for moving object detection (specifically background model initialization, background maintenance, and foreground detection), particle filtering based object tracking (the appearance model and methods for parameter settings for the likelihood model), a fast implementation for appearance model construction, and crowd analysis (crowd flow estimation).

We consider robustness and computational cost as the major design goals of our work. Our methods are designed to be proactive and capable of processing the video stream for event analysis in real-time. Ideally they can assist human operators with identification of important events in videos and responding to them in a timely manner.

5.1 Thesis Contributions

This thesis describes our proposed methods for background subtraction, object tracking, and crowd flow estimation for applications in visual surveillance. We consider robustness and computational cost as the major design goals of our work. and responding to them in a timely manner. The main contributions of the thesis are as follows.

- **Moving Object Detection** - In the area of moving object detection a technique robust to background dynamics using background subtraction with adaptive pixel-wise background model update is described. A foreground-background pixel classification method using adaptive thresholding is presented. Another technique that is robust to sudden illumination changes using an illumination model and a statistical test is presented. We also propose a fast implementation of the method using an extension of integral images.

- **Target Object Tracking** - Once a moving object is detected, the foreground object mask generated is used to initiate object tracking in a particle filtering framework. We propose a method for robust tracking with dynamic parameter setting for the likelihood model of particle filtering. We investigate the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation. In addition, we propose a fast method to construct appearance model of particle filtering for object tracking application.
- **Crowd Flow Estimation** - We present a novel method for pedestrian “flow” estimation that counts the number of persons passing a detection line (trip wire) or a designated region over a period of time. The method is designed to be robust to varying pedestrian flow rate (crowdedness). We assume different level of crowdedness will result in different level of occlusion. We utilize texture feature to classify different levels of occlusion. Number of foreground pixels estimated from background subtraction is used as a complementary feature to estimate the pedestrian flow.

5.2 Future Work

Our methods can be improved and extended in the following ways:

- An important problem that has been over looked in the literature of object tracking is the integration of contextual/semantic information. For example, in a vehicle tracking application, the location of vehicles should be constrained to paths on the ground as opposed to somewhere in the air. Such a problem can be improved by integrating object recognition into the object tracking process.
- The block size for Gaussianity test is an empirically determined threshold, the problem of optimal block size should be investigated.
- Further investigation is need concerning visual features and how they can be used for object tracking. We use low level visual features such as color and

edge distributions. Other more complex visual features such as SIFT and region covariance have been used for object tracking. Multiple features can be combined as a complementary to each other, some features could be “cascaded” in a hierarchical framework for more robust tracking.

- The crowd flow estimation technique we propose does not account for pedestrian velocity. For example, if one person runs pass a trip wire very quickly, the count will be lower than one because of his/her velocity. Motion vectors determined using optical flow can be introduced into the method to account for varying velocity. Our crowd flow estimation technique also does not account for direction of the movement. For example it can not differentiate a person walking left to right or right to left.
- We also make assumptions on the perspective of the field of view for the proposed crowd flow estimation technique. Perspective distortion should be taken into account for scenes with greater angle of view.

5.3 Publications

Ka Ki’s publications include:

Journal Articles:

1. **Ka Ki Ng** and Edward J. Delp, “Fast and Robust Object Detection and Tracking for Lightweight Visual Surveillance Systems,” *IEEE Transactions on Circuits and Systems for Video Technology*, in preparation.
2. **Ka Ki Ng** and Edward J. Delp, “Robust Crowd Flow Estimation Using Multiple Visual Features for Scenes with Changing Crowd Densities,” *Journal of Electronic Imaging*, in preparation.

Conference Papers

1. **Ka Ki Ng**, Satyam Srivastava, and Edward J. Delp, “Segmentation With Sudden Illumination Changes Using A Shading Model And A Gaussianity Test,”

Proceedings of the International Symposium on Image and Signal Processing and Analysis, Dubrovnik, Croatia, September 2011.

2. Satyam Srivastava, **Ka Ki Ng**, and Edward J. Delp, "Crowd Flow Estimation Using Multiple Visual Features for Scenes with Changing Crowd Densities," *Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance*, Klagenfurt, Austria, September 2011.
3. Satyam Srivastava, **Ka Ki Ng**, and Edward J. Delp, "Co-Ordinate Mapping and Analysis of Vehicle Trajectory for Anomaly Detection," *Proceedings of the IEEE International Conference on Multimedia and Expo*, Barcelona, Spain, July 2011.
4. Satyam Srivastava, **Ka Ki Ng**, and Edward J. Delp, "Color Correction for Object Tracking Across Multiple Cameras," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011.
5. **Ka Ki Ng**, Priyanka Agarwal, Nathan Mullen, Dzung Du, and Ilya Pollak, "Comparison of Several Covariance Matrix Estimators for Portfolio Optimization," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011.
6. **Ka Ki Ng**, and Edward J. Delp, "Background Subtraction Using A Pixel-Wise Adaptive Learning Rate For Object Tracking Initialization," *Proceedings of the SPIE Conference on Visual Information Processing and Communication*, vol. 7882, San Francisco, CA, January 2011.
7. **Ka Ki Ng**, and Edward J. Delp, "Object Tracking Initialization Using Automatic Moving Object Detection," *Proceedings of the SPIE Conference on Visual Information Processing and Communication*, vol. 7543, San Jose, CA, January 2010.

8. **Ka Ki Ng**, and Edward J. Delp, “New Models For Real-Time Tracking Using Particle Filtering,” *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, vol. 7257, San Jose, CA, January 2009.
9. Fengqing Zhu, **Ka Ki Ng**, Golnaz Abdollahian, and Edward J. Delp, “Spatial and Temporal Models For Texture-Based Video Coding,” *Proceedings of the SPIE Conference on Video Communications and Image Processing*, vol. 6508, San Jose, CA, January 2007.

LIST OF REFERENCES

LIST OF REFERENCES

- [1] M. Valera and S. Velastin, "Intelligent distributed surveillance systems: a review," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192–204, April 2005.
- [2] H. Kruegle, *CCTV Surveillance, Second Edition: Video Practices and Technology*. Oxford, UK: Elsevier Butterworth-Heinemann, 2006.
- [3] T. Reeve, "How many cameras in the uk? only 1.85 million, claims acpo lead on cctv," Web archive: <http://www.securitynewsdesk.com/2011/03/01/how-many-cctv-cameras-in-the-uk/>, Accessed: November 2011.
- [4] R. Shah, "Effectiveness of red light cameras in chicago," Web archive: <http://www.eyeingchicago.com>, Accessed: November 2011.
- [5] M. Shah, O. Javed, and K. Shafique, "Automated visual surveillance in realistic scenarios," *IEEE Transactions on Multimedia*, vol. 14, no. 1, pp. 30–39, January 2007.
- [6] Y. Tian *et al.*, "IBM smart surveillance system (s3): event based video surveillance system with an open and extensible framework," *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 315–327, September 2008.
- [7] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 657–662, April 2006.
- [8] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: a systematic survey," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 294–307, March 2005.
- [9] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, July 1997.
- [10] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, August 2000.
- [11] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision*, Princeton, NJ, October 1998, pp. 8–14.
- [12] A. Talukder and L. Matthies, "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, September 2004, pp. 3718–3725.

- [13] Li and Chellappa, “A generic approach to simultaneous tracking and verification in video,” *IEEE Transactions on Image Processing*, vol. 11, pp. 530–544, May 2002.
- [14] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *ACM Computing Surveys*, vol. 38, no. 4, December 2006.
- [15] K. Hariharakrishnan, D. Schonfeld, P. Raffy, and F. Yassa, “Video tracking using block matching,” *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, September 2003.
- [16] N. Haering, P. L. Venetianer, and A. Lipton, “The evolution of video surveillance: an overview,” *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 279–290, September 2008.
- [17] H. M. Dee and S. A. Velastin, “How close are we to solving the problem of automated visual surveillance?” *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 329–343, September 2008.
- [18] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 34, no. 3, pp. 334–352, August 2004.
- [19] M. F. Abdelkader, R. Chellappa, Q. Zheng, and A. L. Chan, “Integrated motion detection and tracking for visual surveillance,” *Proceedings of the IEEE International Conference on Computer Vision Systems*, Washington, DC, USA, January 2006, p. 28.
- [20] A. M. McIvor, “Background subtraction techniques,” *Proceedings of Image and Vision Computing*, Hamilton, New Zealand, November 2000, pp. 147–153.
- [21] A. Dore, M. Musso, and C. Regazzoni, “Map particle selection in shape-based object tracking,” *Proceedings of the IEEE International Conference on Image Processing*, San Antonio, Texas, September 2007, pp. 341–344.
- [22] M. K. Pitt and N. Shephard, “Filtering via simulation: Auxiliary particle filters,” *Journal of the American Statistical Association*, vol. 94, no. 446, p. 590, June 1999.
- [23] C. Yang, R. Duraiswami, and L. Davis, “Fast multiple object tracking via a hierarchical particle filter,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, Beijing, China, October 2005, pp. 212–219.
- [24] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.
- [25] S. K. Zhou, R. Chellappa, and B. Moghaddam, “Visual tracking and recognition using appearance-adaptive models in particle filters,” *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1491–1506, November 2004.
- [26] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, Hilton Head Island, SC, USA, 2000, pp. 142–149.

- [27] Y. Dedeoglu, *Algorithms for Smart Video Surveillance: Moving Object Detection, Tracking and Classification*. German: LAP LAMBERT Academic Publishing, 2010.
- [28] M. Isard and A. Blake, “Condensation – conditional density propagation for visual tracking,” *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.
- [29] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, “Color-based probabilistic tracking,” *Proceedings of the European Conference on Computer Vision*. Copenhagen, Denmark: Springer-Verlag, May 2002, pp. 661–675.
- [30] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, “Detection and location of people in video images using adaptive fusion of color and edge information,” *Proceedings of the International Conference on Pattern Recognition*, Barcelona, Spain, September 2000, pp. 627–630.
- [31] R. T. Collins *et al.*, “A system for visual surveillance and monitoring,” Final Report CMU-RI-TR-00-12, Carnegie Mellon University, 2000.
- [32] Q. Hu, S. Li, K. He, and H. Lin, “A robust fusion method for vehicle detection in road traffic surveillance,” *Proceedings of the International Conference on Intelligent Computing*, Changsha, China, August 2010, pp. 180–187.
- [33] O. Javed, K. Shafique, and M. Shah, “A hierarchical approach to robust background subtraction using color and gradient information,” *Proceedings of the IEEE Workshop on Motion and Video Computing*, Orlando, Florida, December 2002, pp. 22–27.
- [34] T. Tanaka, S. Yoshinaga, A. Shimada, R.-I. Taniguchi, T. Yamashita, and D. Arita, “Object detection based on combining multiple background models,” *IPSN Transactions on Computer Vision and Applications*, vol. 2, pp. 156–168, November 2010.
- [35] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, “Wallflower: Principles and practice of background maintenance,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, Kerkyra, Greece, September 1999, pp. 255–261.
- [36] Y. Chen, C. Chen, C. Huang, and Y. Hung, “Efficient hierarchical method for background subtraction,” *Pattern Recognition*, vol. 40, no. 10, pp. 2706–2715, October 2007.
- [37] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting moving objects, ghosts, and shadows in video streams,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1337–1342, October 2003.
- [38] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, “Improving shadow suppression in moving object detection with hsv color information,” *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, Oakland, CA, August 2001, pp. 334–339.
- [39] A. Prati, R. Cucchiara, I. Mikic, and M. Trivedi, “Analysis and detection of shadows in video streams: a comparative evaluation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, December 2001, pp. 571–576.

- [40] L. Vosters, C. Shan, and T. Gritti, "Background subtraction under sudden illumination changes," *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, Boston, Massachusetts, August 2010, pp. 384–391.
- [41] B. Xie, V. Ramesh, and T. Boult, "Sudden illumination change detection using order consistency," *Image and Vision Computing*, vol. 22, no. 2, pp. 117–125, February 2004.
- [42] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, June 1975.
- [43] M. Piccardi, "Background subtraction techniques: a review," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, The Hague, Netherlands, October 2004, pp. 3099–3104.
- [44] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," *Proceedings of Image and Vision Computing New Zealand*, Auckland, New Zealand, November 2002, pp. 267–271.
- [45] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME – Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, March 1960.
- [46] S. Haykin, *Kalman Filtering and Neural Networks*. Wiley, 2001.
- [47] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, Lake Louise, Alberta, Canada, October 2000, pp. 153–158.
- [48] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.
- [49] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, May 2002.
- [50] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "Color features for tracking non-rigid objects," *Chinese Journal of Automation - Special Issue on Visual Surveillance*, vol. 29, no. 3, pp. 345–355, May 2003.
- [51] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001.
- [52] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.
- [53] R. Tan and J. W. Davis, "Differential video coding of face and gesture events in presentation videos," *Computer Vision and Image Understanding*, vol. 96, pp. 200–215, November 2004.

- [54] A. Elgammal, R. Duraiswami, and L. S. Davis, "Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1499–1504, November 2003.
- [55] Q. Wang and J. Liu, "Visual tracking using the kernel based particle filter and color distribution," *Proceedings of the International Conference on Neural Networks and Brain*, vol. 3, Beijing, China, October 2005, pp. 1730–1733.
- [56] A. Tyagi and J. Davis, "A context-based tracker switching framework," *Proceedings of the IEEE Workshop on Motion and video Computing*, Copper Mountain, CO, January 2008, pp. 1–8.
- [57] J. Lichtenauer, M. Reinders, and E. Hendriks, "Influence of the observation likelihood function on particle filtering performance in tracking applications," *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, Seoul, Korea, May 2005, pp. 767–772.
- [58] H. Ying, X. Qiu, J. Song, and X. Ren, "Particle filtering object tracking based on texture and color," *Proceedings of the IEEE International Symposium on Intelligence Information Processing and Trusted Computing*, Huanggang, China, October 2010, pp. 626–630.
- [59] N. Bouaynaya, W. Qu, and D. Schonfeld, "An online motion-based particle filter for head tracking applications," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, Philadelphia, PA, USA, March 2005, pp. 225–228.
- [60] N. Bouaynaya and D. Schonfeld, "On the optimality of motion-based particle filtering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 1068–1072, July 2009.
- [61] T. Xiong and C. Debrunner, "Stochastic car tracking with line- and color-based features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 324–328, December 2004.
- [62] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," *Proceedings of the European Conference on Computer Vision*, Graz, Austria, May 2006, pp. 589–600.
- [63] A. Tyagi, J. Davis, and G. Potamianos, "Steepest descent for efficient covariance tracking," *Proceedings of the IEEE Workshop on Motion and video Computing*, Copper Mountain, CO, January 2008, pp. 1–6.
- [64] H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive object tracking based on an effective appearance filter," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1661–1667, September 2007.
- [65] Y. Rui and Y. Chen, "Better proposal distributions: object tracking using unscented particle filter," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Kauai, HI, 2001, pp. 786–793.
- [66] G. Strovik, "Particle filters for state-space models with the presence of unknown static parameters," *IEEE Transactions on Image Processing*, vol. 50, no. 2, pp. 281–289, February 2002.

- [67] F. C. Crow, "Summed-area tables for texture mapping," *Proceedings of the Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, July 1984, pp. 207–212.
- [68] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, Kauai, HI, December 2001, pp. 511 – 518.
- [69] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, San Diego, CA, June 2005, pp. 829–836.
- [70] J. L. Ritendra Datta, Dhiraj Joshi and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys*, vol. 40, no. 2, pp. 1–60, 2008.
- [71] C. Huang, K. Lin, and F. Long, "A fast eye localization algorithm using integral image," *Proceedings of the International Symposium on Computational Intelligence and Design*, vol. 1, no. Changsha, Hunan, China, December 2009, pp. 231–234.
- [72] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, New York, NY, June 2006, pp. 1491–1498.
- [73] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Proceedings of the European Conference on Computer Vision*, Graz, Austria, May 2006, pp. 404–417.
- [74] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, Rochester, New York, September 2002, pp. 900–903.
- [75] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, Beijing, China, October 2005, pp. 166–173.
- [76] B. Zhan, D. Monekosso, P. Remagnino, S. Velastin, and L. Xu, "Crowd analysis: A survey," *Machine Vision and Applications*, vol. 19, no. 5, pp. 345–357, 2008.
- [77] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, California, June 2005, pp. 886–893.
- [78] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo, "Automatic estimation of crowd density using texture," *Proceedings of the International Workshop on Systems and Image Processing*, Poland, May 1997.
- [79] R. Ma, L. Li, W. Huang, and Q. Tian, "On pixel count based crowd density estimation for visual surveillance," *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, Singapore, December 2004, pp. 170–173.

- [80] D. Kong, D. Gray, and H. Tao, "Counting pedestrians in crowds using viewpoint invariant training," *Proceedings of the British Machine Vision Conference*, Oxford, UK, September 2005.
- [81] G. G. Lee, B. S. Kim, and W. Y. Kim, "Automatic estimation of pedestrian flow," *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*, Vienna, Austria, September 2007, pp. 291–296.
- [82] B. S. Kim, G. G. Lee, J. Y. Yoon, J. J. Kim, and W. Y. Kim, "A method for counting pedestrians in crowded scenes," *Proceedings of the International Conference on Intelligent Computing*, Shanghai, China, September 2008, pp. 1117–1126.
- [83] A. B. Chan, Z. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, June 2008, pp. 1–7.
- [84] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding*, vol. 73, pp. 428–440, March 1999.
- [85] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Journal of Computing Surveys*, vol. 43, pp. 1–43, April 2011.
- [86] P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1473–1488, November 2008.
- [87] T. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 8, pp. 630–632, August 1978.
- [88] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," *Proceedings of the European Workshop on Advanced Video Based Surveillance Systems*, London, United Kingdom, September 2001, pp. 1–5.
- [89] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, July 2002.
- [90] M. Gurcan, Y. Yardimci, A. Cetin, and R. Ansari, "2-d adaptive filtering based gaussianity tests in microcalcification detection," *Proceedings of the SPIE IS&T Electronic Imaging: Visual Communications and Image Processing*, vol. 3309, San Jose, California, January 1998, pp. 625–633.
- [91] M. Nafi Gurcan, Y. Yardimci, and A. Enis Cetin, "Influence function based gaussianity tests for detection of microcalcifications in mammogram images," *Proceedings of the IEEE International Conference on Image Processing*, Kobe, Japan, October 1999, pp. 407–411.
- [92] R. Ojeda, J.-F. Cardoso, and E. Moulines, "Asymptotically invariant gaussianity test for causal invertible time series," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, Munich, Germany, April 1997, pp. 3713–3716.

- [93] S.-Y. Chien, Y.-W. Huang, B.-Y. Hsieh, S.-Y. Ma, and L.-G. Chen, "Fast video segmentation algorithm with shadow cancellation, global motion compensation, and adaptive threshold techniques," *IEEE Transactions on Multimedia*, vol. 6, no. 5, pp. 732–748, October 2004.
- [94] I. Sato, Y. Sato, and K. Ikeuchi, "Illumination from shadows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 3, pp. 290–300, March 2003.
- [95] A. Troccoli and P. Allen, "Recovering illumination and texture using ratio images," *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, Chapel Hill, NC, USA, June 2006, pp. 655–662.
- [96] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*. McGraw-Hill, 1995.
- [97] F. S. H. Jr., *Computer Graphics*. London: Collier Macmillan, 1990.
- [98] L. Shapiro and G. Stockman, *Computer Vision*. Upper Saddle River, New Jersey: Prentice Hall, 2001.
- [99] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–575, May 2003.
- [100] K. Deguchi, O. Kawanaka, and T. Okatani, "Object tracking by the mean-shift of regional color distribution combined with the particle-filter algorithm," *Proceedings of the International Conference on Pattern Recognition*, vol. 3, Cambridge, UK, August 2004, pp. 506–509.
- [101] H. Sugano and R. Miyamoto, "A real-time object recognition system on cell broadband engine," *Proceedings of Pacific-Rim Symposium on Image and Video Technology*, Santiago, Chile, December 2007, pp. 932–943.
- [102] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982.
- [103] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, March 1960.
- [104] M. Piccardi, "Background subtraction techniques: A review," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, The Hague, Netherlands, October 2004, pp. 3099–3104.
- [105] K. K. Ng and E. J. Delp, "Object tracking initialization using automatic moving object detection," *Proceedings of the SPIE/IS&T Electronic Imaging: Visual Information Processing and Communication*, vol. 7543, San Jose, California, January 2010.
- [106] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, May 1979.
- [107] CAVIAR: Context Aware Vision using Image-based Active Recognition, "CAVIAR Test Case Scenarios," Online: <http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/>, Accessed: November 2011.

- [108] University of Central Florida, Computer Vision Lab, “Data Sets and Test Data,” Online: <http://server.cs.ucf.edu/~vision/data.html>, Accessed: February 2011.
- [109] University of California, San Diego, Statistical Visual Computing Lab, “UCSD Anomaly Detection Dataset,” Online: <http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm>, Accessed: February 2011.
- [110] S. Ali and M. Shah, “Floor fields for tracking in high density crowd scenes,” *Proceedings of the European Conference on Computer Vision*, Marseille, France, October 2008, pp. 1–14.
- [111] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, “Anomaly detection in crowded scenes,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, California, June 2010, pp. 1975–1981.

VITA

VITA

Ka Ki Ng was born in Hong Kong. She received her B.S. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, Indiana in 2005. Her research interests include video compression, image processing, and video analytics.

Ka Ki joined the Direct Ph.D. program at Purdue University, West Lafayette, Indiana in 2005. In Spring 2006, she joined the Video and Image Processing Laboratory (VIPER) as a Research Assistant. Her major advisor Professor Edward J. Delp is the Charles William Harrison Distinguished Professor of Electrical and Computer Engineering. While in the graduate program, Ka Ki was supported by the United State Department of Homeland Security. During the summer of 2007, she was a Student Intern at the Thomson Inc., Indianapolis, IN, and the summer of 2011, she was a Student Intern at Qualcomm Inc., San Diego, CA.

Ka Ki is a student member of the IEEE and the IEEE Signal Processing Society.