DETECTION AND TRACKING OF MULTIPLE HUMANS IN HIGH-DENSITY CROWDS

by

Irshad Ali

A research study submitted in partial fulfillment of the requirements for the degree of Master of Engineering in Computer Science

Examination Committee: Dr. Matthew N. Dailey (Chairperson)
Dr. Manukid Parnichkun (Member)
Dr. Nitin V. Afzulpurkar (Member)

Nationality: Pakistani
Previous Degree: Bachelor of Science in Computer Engineering
Samara State Technical University, Russia

Scholarship Donor: Higher Education Commission (HEC), Pakistan - AIT Fellowship

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2009
Acknowledgment

First of all I thank to my advisor Matthew Dailey for his continuous support in my master research study. He was always there to give advice and listen to my problems. I learnt from him how to do research and he showed me how to be persistent to accomplish any goal.

Besides my advisor I would like to thank to my committee members Nitin Afzulpurkar and Manukid Parnichkun for their valuable suggestions, encouragement and constructive criticism.

I would like to thank Higher Education of Pakistan (HEC) for providing me financial support throughout my studies at AIT.

I would like to thank to all my friends for their support and assistance during my studies. Last, but not least I thank my family for their unconditional support and encouragement.
Abstract

As public concern about crime and terrorist activity increases, the importance of public security is growing, and video surveillance systems are increasingly widespread tools for monitoring, management, and law enforcement in public areas. The visual surveillance system has become popular research area in computer vision. There are many algorithms exists to detect and track people in video stream. Human detection and tracking in high density crowds where object occlusion is very high is still an unsolved problem. Many preprocessing techniques such as background subtraction are fail in such situations.

I present a fully automatic approach to multiple human detection and tracking in high density crowds in the presence of extreme occlusion. We integrate human detection and tracking into a single framework, and introduce a confirmation-by-classification method to estimate confidence in a tracked trajectory, track humans through occlusions, and eliminate false positive detections. I search for new humans in those parts of each frame where humans have not been detected previously. I increase and decrease the weight of tracks through confirmation-by-classification process. This is helpful to remove those tracks (false positives) which not confirmed for long time. I keep the tracks which have not been confirmed only for a short period of time, to give chance to those tracks where human are occluded fully or partially for a short period of time to rejoin their tracks. We use a Viola and Jones AdaBoost cascade classifier for detection, a particle filer for tracking, and color histograms for appearance modeling.

An experimental evaluation shows that our approach is capable of tracking humans in high density crowds despite occlusions. On a test set with 35.35 humans per image achieves 76.8% hit rate with 2.05 false positives per image and 8.2 missed humans per image. The results form a strong basis for farthar research.
Table of Contents

Chapter | Title | Page
--- | --- | ---
Title Page | i
Abstract | iii
Table of Contents | iv
List of Figures | v
List of Tables | vii
1 | Introduction | 1
1.1 | Overview | 1
1.2 | Problem Statement | 1
1.3 | Objectives | 3
1.4 | Limitations and Scope | 3
1.5 | Research Outline | 3
2 | Literature Review | 4
2.1 | Object Detection | 4
2.2 | Human Detection in Crowds | 6
2.3 | Tracking Pedestrians in Crowds | 10
2.4 | Pedestrian Counting | 13
2.5 | Pedestrian Behavior | 14
3 | Methodology | 16
3.1 | System Overview | 16
3.2 | Classifier Training | 17
3.3 | Head Detection | 20
3.4 | Human Tracking | 20
4 | Experiments and Results | 25
4.1 | Overview | 25
5 | Conclusion and Recommendation | 29
5.1 | Conclusion | 29
5.2 | Contribution | 29
5.3 | Recommendations | 29

References | 31
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Example of a large crowd. People are walking in both directions along the road near the Mochit light rail station in Bangkok, Thailand. This picture shows the extreme occlusion of the pedestrians bodies.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Another example of a large crowd. People are standing and walking in both direction along the road near the Mochit light rail station in Bangkok, Thailand. There are some other moving objects (cars) also in the scene.</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Another example of a large crowd. In this scene people are walking in both directions along the road near the Mochit light rail station. One person on a motorcycle is moving faster than the rest of the crowd.</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Example rectangle features in a detection window. The feature value is equal to the difference of the sum of the pixel intensities inside the white rectangles and black rectangles. (a) Two-rectangle features. (b) Three-rectangle feature. (c) Four-rectangle feature. Reprinted from Viola and Jones (2001b).</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Viola and Jones integral image. (a) The integral image value is the sum of pixel values from the top-left corner of the image to the location ((x, y)). (b) An example of using integral image values to compute the sum of the intensities inside region D. The value at position 1 is the sum of pixel values in rectangular region A. The value at position 2 is sum of pixel values in rectangular regions A and B. The value at position 3 is sum of pixel values in rectangular regions A and C. The value at position 4 is sum of pixel values in rectangular regions A, B, C and D. The sum within rectangle D can be computed as (4+1-(2+3)). Reprinted from Viola and Jones (2001b).</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Structure of a cascade. Reprinted from Viola and Jones (2001b).</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Challenges in contour-based object detection. (a) Good contrasted image. (b) Detected contours of image shown in (a). (c) Poor contrasted image. (d) Detected contours of image shown in (b). Reprinted from Schindler and Suterb (2008).</td>
<td>7</td>
</tr>
<tr>
<td>2.5</td>
<td>Pedestrian detection using background subtraction. (a) A sample frame. (b) Results of background subtraction. Reprinted from Zhao, Nevatia, and Wu (2008).</td>
<td>7</td>
</tr>
<tr>
<td>2.6</td>
<td>Head and shoulder detection. (a) Heads detected from foreground boundaries. (b) (\Omega)-shape head and shoulder model. Reprinted from Zhao et al. (2008).</td>
<td>8</td>
</tr>
<tr>
<td>2.7</td>
<td>Edgelet features. Reprinted from Wu and Nevatia (2007).</td>
<td>9</td>
</tr>
</tbody>
</table>
2.8 Body part based detection system. (a) Spatial relations between body parts. (b) Tracking example using body part-based detection. Reprinted from Wu and Nevatia (2007).

2.9 Pedestrian detection based on local and global cues. Reprinted from Leibe, Seemann, and Schiele (2005).

2.10 Human tracking using a 3D shape model. (a) Ellipsoid-based 3D human shape model. (b) A tracking example using the human shape model. Reprinted from Zhao et al. (2008).

2.11 Pedestrian counting. (a) Perspective map. (b) An example of people counting. Red and green tracks show the flow of people walking towards or away from the camera. Reprinted from Chan, Liang, and Vasconcelos (2008).

2.12 Sample pedestrian behaviors. (a) Two people fighting. (b) Overcrowding. (c) Blocking an area. (d) A person jumping over a barrier. Reprinted from Cupillard, Avanzi, Bremond, and Thonnat (2004).

3.1 System overview.

3.2 The main concept of Adaboost.

3.3 Some example positive images used to train the head classifier.

3.4 Some example negative images used to train the classifier.

3.5 Block diagram of the tracking algorithm. $D$ is the distance between a newly detected head and a the nearest predicted location, $C$ is threshold (in pixels) less than the width of the tracking window, and $a$ is a constant, $a > 0 \leq 1$.

3.6 The figure shows the tracking error and detection response in consecutive frames. Red rectangle represent tracking window and green rectangle shows the detection.

4.1 Tracking results on real video frames. Red rectangles indicate detections, and green rectangles indicate actual head positions in the frame.
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Training Parameters.</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Head Detection Parameters.</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>Tracking Results.</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

As public concern about crime and terrorist activity increases, the importance of public security is growing, and video surveillance systems are increasingly widespread tools for monitoring, management, and law enforcement in public areas. Since it is difficult for human operators to monitor surveillance cameras continuously, there is a strong interest in automated analysis of video surveillance data. Some of the important problems include pedestrian tracking, behavior understanding, anomaly detection, and unattended baggage detection. In this research study, I focus on pedestrian tracking.

There are many commercial and open source visual surveillance systems available in the market but these are manual and need additional resources such as human operators to observe the video. This not feasible in high density crowds, where observing the motion and behavior of humans is a very difficult and boring job, making it error prone.

Automatic pedestrian detection and tracking is a well-studied problem in computer vision research, but the solutions thus far only able to track a few people. Inter-object occlusion, self-occlusion, reflections, and shadows are some of the factors making automatic crowd detection and tracking difficult. I will discuss these issues in chapter 2 in more detail.

The pedestrian tracking problem is especially difficult when the task is to monitor and manage large crowds in gathering areas such as airports and train stations. There has been a great deal of progress in recent years, but still, most state-of-the-art systems are inapplicable to large crowd management situations because they rely on either background modeling, body part detection, or body shape models. These techniques make it impossible to track large numbers of people in heavily crowded scenes in which the majority of the scene is in motion (rendering background modeling useless) and most of the human’s bodies are partially or fully occluded. Under these conditions, we believe that the human head is the only body part that can be robustly detected and tracked, so in this research study I present a method for tracking pedestrians by detecting and tracking their heads rather than their full bodies.

In this research study I propose, implement, and evaluate a large scale human tracking system, which can detect and track humans in large crowds.

1.2 Problem Statement

Pedestrian detection and tracking in large crowds such as those shown in Figures 1.1, 1.2, and 1.3 is still an open problem.

In high-density crowds some popular techniques such as background subtraction fail. We cannot deal with this problem in ways similar to simple object tracking, where the number
**Figure 1.1:** Example of a large crowd. People are walking in both directions along the road near the Mochit light rail station in Bangkok, Thailand. This picture shows the extreme occlusion of the pedestrians bodies.

**Figure 1.2:** Another example of a large crowd. People are standing and walking in both direction along the road near the Mochit light rail station in Bangkok, Thailand. There are some other moving objects (cars) also in the scene.

**Figure 1.3:** Another example of a large crowd. In this scene people are walking in both directions along the road near the Mochit light rail station. One person on a motorcycle is moving faster than the rest of the crowd.
of objects is not large and most part of the object are visible. Inter-object occlusion and self
occlusion in crowd situations makes detection and tracking challenging. In such situations we
also cannot perform segmentation, detection, and tracking separately. We should consider it
as single problem.

We have four main challenges during crowd detection and tracking:

- **Inter-object occlusion**: a situation in which part of the target object is hidden
  behind another. Inter-object occlusion becomes critical as the crowd density increases.
  In very high-density crowds, most of an object’s parts may not be visible.

- **Self occlusion**: sometimes an object occludes itself, for example when a person talks on
  a mobile phone, the phone and hand may hide part of the head. This type of occlusion
  is more temporary and short term.

- **Size of the visible region of the object**: as the density of a crowd increases, the
  size of the visible region of the object decreases. Detecting and tracking this object in
  a dense situation is very difficult.

- **Appearance ambiguity**: when target objects are small, then appearance tends to be
  less distinguished.

1.3 Objectives

The main objectives of this research study are as follows:

1. To develop a method to detect and track humans in a crowded video in the presence
   of extreme occlusion.
2. Evaluate the performance of the system on real videos.

1.4 Limitations and Scope

For successful detection and tracking, I require that at least the head must be visible. I assume
a stationary camera so that motion can be detected by comparing subsequent frames.

1.5 Research Outline

The organization of this research study is as follows. Chapter 2 describes the literature
review, chapter 3 describes the methodology, chapter 4 describes the experiments and results
and chapter 5 conclusion and recommendation.
Chapter 2

Literature Review

In this chapter I review existing methods for handling crowd scenes. The chapter is divided into five parts, according to the targets and applications:

1. Object Detection
2. Human Detection in Crowds
3. Tracking Pedestrians in Crowds
4. Pedestrian Counting
5. Crowd Behavior

2.1 Object Detection

Object detection in images is a challenging problem for computer vision; many approaches based on features and shape exist. The problem becomes more difficult in the case of high-density crowds where full visibility of the target cannot be guaranteed, and inter-object occlusion produces ambiguity of object appearance and shape.

The Viola and Jones (2001b, 2001a) approach is a very popular real time object detection technique. They use feature-based object detection. They tested their method on face detection. They used 4916 hand-labeled faces scaled and aligned to a base resolution of 24 × 24 pixels and 10,000 non-face (negative images) of size 24 × 24 pixels to train their classifier. On a test set containing 130 images and 507 faces, they achieved a 93.7% detection rate with 422 false detections.

A short description of the Viola and Jones approach is given below.

1. They compute an image feature using an integral image, which is very simple and quick to compute. There are three types of features: two-rectangle, three-rectangle, and four-rectangle. A feature’s value is equal to the difference of the sum of white rectangles and black rectangles as shown in Figure 2.1. The integral image value at location \((x, y)\) can be computed from the following equation.

\[
ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')
\]

where \(i(x', y')\) is the pixel intensity value at location \((x', y')\). The procedure for computing the integral image value at any location \((x, y)\) in an image is shown in Figure 2.2.

2. They learn classifiers based on AdaBoost. AdaBoost selects a small number of weak classifiers based on image features and produces a strong classifier.
Figure 2.1: Example rectangle features in a detection window. The feature value is equal to the difference of the sum of the pixel intensities inside the white rectangles and black rectangles. (a) Two-rectangle features. (b) Three-rectangle feature. (c) Four-rectangle feature. Reprinted from Viola and Jones (2001b).

Figure 2.2: Viola and Jones integral image. (a) The integral image value is the sum of pixel values from the top-left corner of the image to the location \((x, y)\). (b) An example of using integral image values to compute the sum of the intensities inside region D. The value at position 1 is the sum of pixel values in rectangular region A. The value at position 2 is sum of pixel values in rectangular regions A and B. The value at 3 is sum of pixel values in rectangular regions A and C. The value at position 4 is sum of pixel values in rectangular regions A, B, C and D. The sum within rectangle D can be computed as \(4+1-(2+3)\). Reprinted from Viola and Jones (2001b).

3. The weak learning algorithm, at each iteration select the feature that performs best in classifying the training set in terms of weighted error rate. The algorithm calculates a threshold for each feature.

4. Finally they combine multiple strong classifiers into a cascade to discard background regions (negative images) quickly. Each element of the cascade contains a strong classifier combining multiple weak classifiers by weighted voting. A positive result from the first classifier will be passed to second classifier, and positive results from the second will be passed to third classifier, and so on, as shown in Figure 2.3.

The Viola and Jones technique is one example of a feature-based technique. Another major category of object detection technique is the category of shape-based techniques. As an example Schindlera and Suterb (2008) perform object detection using shape-based method. They achieve 83-91% detection rate at 0.2 false positives per image with average processing
time of 6.5 seconds for 480 × 320 pixels size image. They first extract contour points of an object to represent the object shape. They propose a distance measure to compare a candidate shape to an object class template. Image contrast, reflection, and shadows are the main problems for using these kind of algorithms, as shown in Figure 2.4.

I select feature-based technique because it performs well in crowded scene where object boundaries are not clear, and it is very fast compare to the shape-based technique.

2.2 Human Detection in Crowds

The objective of human detection in crowds is to detect the individuals in a crowded scene. Many computer vision algorithms have been developed for this task. Success depends on the density of crowd; for example in high-density crowds, where occlusion is high and bodies are not visible, many algorithms based on the human body fail. If the crowd density is low and occlusion is partial or temporary then these algorithms work well. In the following sections I discuss these algorithms in more detail.

2.2.1 Background Subtraction

Background modeling is a very common technique in tracking, and almost all previous work uses background modeling to extract the foreground blobs from the scene. See Figure 2.5 for an example. As mentioned before, the previous work in crowd tracking is limited to a few people. When the crowd is large, there may be so many individuals that background subtraction fails to find meaningful boundaries between objects.
Figure 2.4: Challenges in contour-based object detection. (a) Good contrasted image. (b) Detected contours of image shown in (a). (c) Poor contrasted image. (d) Detected contours of image shown in (b). Reprinted from Schindler and Suter (2008).

Figure 2.5: Pedestrian detection using background subtraction. (a) A sample frame. (b) Results of background subtraction. Reprinted from Zhao et al. (2008).
Figure 2.6: Head and shoulder detection. (a) Heads detected from foreground boundaries. (b) Ω–shape head and shoulder model. Reprinted from Zhao et al. (2008).

2.2.2 Head Detection

In high-density crowds, the head is the most reliably visible part of the human body. Many researchers have attempted to detect pedestrians through head detection. I review this approach here.

Zhao and colleagues (Zhao, Nevatia, & Wu, 2008; Zhao, Nevatia, & Lv, 2004) detect heads from foreground boundaries, intensity edges, and foreground residues (foreground region with previously detected object regions removed). See Figure 2.6 for an example. These algorithms work well in low-density crowds but are not scalable to high-density crowds.

Wu and Nevatia (2007) detect humans using body part detection. They train their detector on examples of heads and shoulders, and other body parts. Their target is to track partially-occluded people. In high-density crowds the visibility of the head and shoulder cannot be guaranteed. The method works well in low density crowds in which the humans are isolated in such a way that at least their heads and shoulders are visible.

Sim, Rajmudhan, and Ranganath (2008) use the Viola-type classifier cascades for head detection. The results from the first classifier are further classified by the second classifier, which is based on color bin images. The color bin images created from normalized RG histogram of resulting windows from first classifier. Their main objective is to reduce the false positives of the first Viola-type head detector using color bin images. They reduced the false alarm rate from 35.9% to 23.9%, but the detection rate also decreased from 87.3% to 82.5%.

This approach is only good for head detection; for tracking the detection rate is more important than the false positive rate. We can reduce/eliminate false positives during tracking, but if we miss detection than there is no way to recover those missing heads during tracking.

2.2.3 Pedestrian Detection

Pedestrian detection in which the entire pedestrian bodies of target humans are detected, is also an interesting topic. There are several existing solutions using different approaches.
The main problem in pedestrian detection is occlusion, which again depends upon the size of crowd. If the crowd density is high then correct detection of pedestrians is still not possible. Andriluka, Roth, and Schiele (2008) combine detection and tracking in a single framework. To detect pedestrians they use a part-based object detection model. The algorithm detects all pedestrians in every frame. In their approach, the human body is represented by the configuration of the body parts. The articulation of each pedestrian is approximated in every frame from body parts or limbs. The main objective of part based detection algorithms is to track in partially occluded situations, such as pedestrians crossing each other on a walkway.

Wu and colleagues (Wu & Nevatia, 2007; Wu, Nevatia, & Li, 2008) use a body part detector approach. They introduce edgelet features in their work. An edgelet is a short segment of a line or a curve as shown in Figure 2.7. They learn the body part detector based on edgelet features using a boosting method. The response from all detectors is combined using a joint likelihood model. Their system took average of 2.5 to 3.6 seconds per image. These algorithms are slow. We cannot apply the body part detector concept in high-density crowds because the body parts are usually not visible.

Zhao et al. (2008, 2004) use previously discussed head detection technique to generate initial
human hypotheses. They use a Bayesian framework in which each person is localized by maximizing a posterior probability over location and shapes, to match 3D human shape models with foreground blobs. They handle the inter-object occlusion in 3D space. The 3D human shape model is a good approach to handle the occlusion problem but we cannot use this approach in high-density crowds because in high-density crowds the human body is often not visible.

Leibe et al. (2005) integrate evidence over multiple iterations and from different sources. They generate pedestrian hypotheses and create local cues from an implicit shape model (Leibe, Leonardis, & Schiele, 2004), which is a scale-invariant feature used to recognize rigid objects. To enforce global consistency, they add information from global shape cues. Finally, they combine both local and global cues. Their system detects partially occluded pedestrians in low density crowds. See Figure 2.9 for an example.

2.3 Tracking Pedestrians in Crowds

Tracking is one of the most researched areas of computer vision. In the case of pedestrian tracking, the focus of the all the previous research has been to estimate velocity, identify the object in consecutive frames, and so on. The previous work does not consider dense environments. This is why all previous tracking algorithms are limited to few people.

As previously discussed, occlusion occurs very frequently when there are many objects or humans moving in the scene. Long, permanent, and heavy occlusions are the basic problems of pedestrian tracking in crowds.

2.3.1 Likelihood

While tracking objects from one frame to another frame we search for the objects in the next frame. In Bayesian approach, to perform this search, we compute the likelihood (based on
appearance, shape and so on) of the object in the next frame. I review this approach here.

Zhao et al. (2004, 2008) propose a 3D human shape model and color histograms for tracking people in crowds, as shown in Figure 2.10. As previously discussed, they first locate head positions and then estimate ellipsoid-based human shape. They combine both detection and tracking in single step. They compute joint likelihood of objects based on appearance, and use Bayesian inference to propose human hypotheses in next frame. Their algorithm is one of best algorithms for tracking people and their experiments include the largest crowd ever used, but the algorithm is still limited to 33 people. This idea to consider 3D space is good for crowd situations. Using 3D models, we can reduce false positives and solve inter-object occlusion problems easily. However, we cannot use the human shape model in high-density crowds.

Wu and Nevatia (2006) track humans using their previously discussed body part detector. In their approach, the human body is the combination of four parts (full-body, head-shoulder, torso, and legs) as shown in Figure 2.8. In their approach they detect static body parts in each frame. The detection results are then input to a tracking module, which tracks from one frame to another. They compute the likelihood from appearance model based on color histogram and dynamic model based on detection response. This approach is good for robust tracking in case of partial occlusion but cannot apply in high density crowds.

Ramanan, Forsyth, and Zisserman (2007) first build a puppet model of each person’s appearance and then track by detecting those models in each frame. To build the model they use two different methods. In the first method, they look for candidate body parts in each frame, then cluster the candidates to find assemblies of parts that might be people, while the second method looks for entire people in each frame. They assume some key poses and build models from those poses. In their approach they do not detect the pedestrian body, but they build the body model based on appearance and calculate the likelihood based on this model in next frame. This approach is good for situations in which we have to deal with different poses of humans such as dancing, playing, and so on. I consider only standing and walking people.

Mathes and Piater (2005) obtain interest points using the popular color Harris corner detector. They build a model describing the object by a point distribution model. Their model
is somewhere between active shape models (Cootes, Taylor, Cooper, & Graham, 1995) and active appearance models (Cootes & Taylor, 2004). These are statistical models based on shape and appearance. They are used to describe the object in images, and later we can use them to match in another image. This algorithm requires a predefined region of interest. In case of large crowd it may be a better choice to select such features to track, rather than the entire object.

2.3.2 Multiple Object Tracking

Tracking people in crowds is a specific case of the more general problem of multiple object tracking. Multiple object means the distribution of objects in the scene and their dynamics will be nonlinear and none-Gaussian. One of the most popular technique for nonlinear, none-Gaussian Bayesian state estimation is the particle filter or sequential Monte Carlo (Doucet, Freitas, & Gordon, 2001) method. The particle filter technique is also known as CONDENSATION (Isard & Blake, 1998b, 1998a). To use a particle filter we need to define objects based on shape (i.e. contours), appearance models or features.

Ma, Yu, and Cohen (2009) consider the multiple target tracking problem as maximum a posteriori problem. A graph representation is used for all observations over time. They use both motion and appearance likelihoods and formulate the multiple target tracking problem as the problem of finding multiple optimal paths in the graph. When we apply segmentation on a cultured scene the object region is mixed with other foreground regions and one foreground region may correspond to multiple objects. To solve this problem, they generate new hypotheses to measure the graph from merge, split and mean shift operations.

Yang, Duraiswami, and Davis (2005) use color and edge orientation histogram features to track multiple objects using a particle filter.

The actual CONDENSATION algorithm shows poor performance when the object are occluded or the objects are too close to each other. To overcome these problems Kang and Kim (2005) use the competition rule. In this way they separate the the objects which are close to each other. In their approach, they use self-organizing map algorithm to build the human shape from the body outlines. To track the people from frame to frame they use Hidden Markov Model.

Since the original particle filter was designed to track single object, Cai, Freitas, and Little (2006) modify the particle filter algorithm for multiple target tracking. They define the trajectories in first frame, find nearest neighbor in next frame and associate them with existing trajectories. To solve the occlusion problem they stabilize the objects trajectories of the targets by combining mean shift and particle filter algorithm in a single framework.

Leibe, Schindler, and Gool (2007) first detect humans using their previous approach (Leibe, Seemann, & Schiele, 2005). After that, they estimate object trajectories in the ground plane. They detect objects in 3D space and join those objects with the hypothetical trajectories. They select the best trajectories using the generated hypothesis. At the end they integrate detection and trajectory estimation problems and form a single optimization problem.

Smith, Gatica-Perez, and Odobez (2005) present a Bayesian framework for tracking a variable number of interacting targets using a fixed camera. They use particle filters to track objects.
from one frame to another and use a Bayesian framework to handle the multiple object tracking problem.

### 2.3.3 Tracking from Multiple Views

Most researchers use the previously discussed monocular approach for tracking because it is simple and easy to deploy. The multiple view approach on the other hand, takes input from two or more cameras, with or without overlapping fields of view. The main advantage of using a multiple view approach is that we obtain more accurate 3D locations. The main disadvantage of multiple camera tracking systems is its higher computational requirements.

Fleuret, Berclaz, Lengagne, and Fua (2008) handle the occlusion problem using synchronized videos from multiple views taken at head level from very different angles. In their experiment, they use four cameras in an indoor environment. Their main objective is to track the people for a long period of time. They track up to six humans without any false positives or negatives. They do not mention the exact time, but they mentioned several minutes.

Mittal and Davis (2003) use many synchronized cameras placed far from each other. Their objective is to segment, detect and track humans from multiple cameras. They introduce a region-based stereo algorithm. The algorithm is designed to find 3D points inside the target object. At the end they merge evidence from all cameras. They thus solve the occlusion problem and produce a more optimal detection and tracking algorithm. Their test is based on 200 frames contains six humans. They use four, eight, and sixteen cameras and found that four cameras were enough for successful tracking.

### 2.4 Pedestrian Counting

An important crowd feature is crowd density. It is natural to think that crowds of different density should receive a different level of attention (Zhan, Monekosso, Remagnino, Velastin, & Xu, 2008). Lee, Goh, and Lam (2005) define the level-of-service for different size of crowds, to improve pedestrian travel. These services used to design and planning of the pedestrian facilities. They use area occupancy, crowd flow, and walking speed of the pedestrian to define these services.

Chan et al. (2008) present a system to estimate dynamic crowd size, as shown in Figure 2.11. They segment the crowd into different parts based on motion. After segmentation they extract features from every segmented part. They find correspondences between crowd size and segmented area, and use Gaussian process regression to learn this correspondences.

Rabaud and Belongie (2006) use a KLT-based tracker. KLT is a feature tracking algorithm. They extract feature trajectories from videos using the KLT algorithm. The feature trajectories are then clustered. Finally, they estimate the number of moving objects in a scene from those trajectory clusters.
2.5 Pedestrian Behavior

Behavior analysis is another important topic in computer vision. These methods attempt to capture the interaction between people using different approaches. I review this work here.

Cupillard et al. (2004) recognize isolated individuals, groups of people, or crowd behavior in scenes using multiple cameras. They detect motion and track objects and groups frame to frame. To estimate the positions and dimensions of people accurately, they combine the results from multiple cameras. For each tracked individual, they divide the action (some example of which shown in Figure 2.12) along the following three dimensions:

1. **States**: describe situations in the scene; for example, a state might be that an individual is close to object of interest (ticket vending machine) or that a group of people is agitated.

2. **Events**: describe changes in state; for example, a group or individual might enter the zone of interest.

3. **Scenario**: a combination of states and events; for example, a scenario might involve an individual moving fast or jumping over a barrier.

Adam, Rivlin, Shimshoni, and Reinitz (2008) present an algorithm to detect special events in crowd situations. Example include a person running in a crowd where everyone else is walking, or one person walking in a direction different from all others. They use multiple local monitors to produce alerts of such events. At the end, they integrate all local alerts and decide whether it is an unusual event or not.

Khan, Balch, and Dellaert (2005) present a particle filter and Markov chain Monte Carlo based algorithm to solve the problem of object interaction. In the framework, objects are influenced by the behavior of other objects. They use a Markov random field for identity management during tracking. They successfully track a crowd of ants.

---

**Figure 2.11**: Pedestrian counting. (a) Perspective map. (b) An example of people counting. Red and green tracks show the flow of people walking towards or away from the camera. Reprinted from Chan et al. (2008).
Figure 2.12: Sample pedestrian behaviors. (a) Two people fighting. (b) Overcrowding. (c) Blocking an area. (d) A person jumping over a barrier. Reprinted from Cupillard et al. (2004).
Chapter 3
Methodology

In this chapter I describe the approach. I use for detecting and tracking people in large crowds. The chapter is divided into following sections:

1. System Overview
2. Classifier Training
3. Head Detection
4. Human Tracking

3.1 System Overview

I use a static camera placed at a sufficient height such that the heads of the pedestrians are visible.

To detect heads I use the Viola and Jones (2001b) technique. The training process is offline and build a classifier using AdaBoost. The acquired video is input to detection module, which detects heads in the first frame and create initial trajectories.

The tracker module uses a probabilistic framework consisting of a motion model and an appearance model. The probabilistic model is based on a particle filter that is used to track heads into the next frame. The motion model is used to predict the position of heads in the next frame. The appearance model model gives a probability based on a color histogram, which is used to update the object state. The details of these modules are given in following sections.

3.1.1 Summary

1. Acquire input crowd video \( V \).
2. In first frame \( v_0 \) of \( V \), detect heads. Let \( N_0 \) be the number of detected heads.
3. Initialize trajectories \( T_j \), \( 1 \leq j \leq N_0 \) with initial positions \( \vec{x}_{j,0} \).
4. Initialize trajectory weights \( w_j = 1 \).
5. Initialize the appearance model (color histogram) \( \vec{h}_j \) of each head region.
6. For each subsequent frame \( v_i \) of input video,
   (a) For each existing trajectory \( T_j \),

16
i. Use motion model to predict the distribution \( p(\vec{x}_{j,i} | \vec{x}_{j,i-1}) \), over locations for head \( j \) in frame \( i \), creating a set of candidate particles \( \vec{x}_{j,i}^{(k)}, 1 \leq k \leq K \).

ii. Compute the local color histogram \( \vec{h}^{(k)} \) and the likelihood \( p(\vec{h}^{(k)} | \vec{x}_{j,i}^{(k)}, \vec{h}_j) \) for each particle \( k \) using the appearance model.

iii. Resample the particles according to their likelihood. Let \( k^* \) be the index of the most likely particle.

iv. Run the head detector on the location \( \vec{x}_{j,i}^{(k^*)} \). If the location is classified as a head, reset \( w_j \leftarrow 1 \); else decrease \( w_j \leftarrow w_j - a \).

v. If \( w_j \) is below threshold, remove trajectory \( j \).

(b) Search for new heads in frame \( v_i \) not within some distance \( D \) of the prediction from an existing trajectory. Initialize a new trajectory for each new detection.

\[ \text{Figure 3.1: System overview.} \]

3.2 Classifier Training

The classifier training module works offline. It is based on the Viola and Jones (2001b) approach, which is a supervised learning method. At each stage a set of simple Haar-like feature (as shown in Figure 2.2) is computed over the training set image. A strong classifier is
3.2.1 Positive Examples

We need to collect positive images that contain only objects of interest. In my case this is the head, from every view. I use a square window identify the location and size of heads in each training images. I select positive examples from several videos taken from different places as shown in Figures 1.1, 1.2 and 1.3. To select positive example I wrote a program using OpenCV and C++ (example are shown in Figure 3.3) allowing a human to select the locations of heads in the frame. The location and width/height are used later in training process.

There should be sufficient samples to achieve a good detection rate. In my experiment I found that 4,000 examples were needed to achieve reasonable results.
3.2.2 Negative Images

Images that do not contain heads are called negative or background images. The variability of negative images is much higher than that of positive images. To achieve an acceptable false positive rate, we should add a large number of negative images.

I found some background images on Seo’s website (Seo, 2007). These images are used for face detection. I also cropped some negative images from my own set of input videos. Some examples are shown in Figure 3.4.

3.2.3 Creating Training Samples

I use the OpenCV `cvCreateTrainingSamples()` function to create positive and negative training examples. There are two ways of creating samples: first, to create training samples from one image, applying distortions, and second, to create training samples without applying distortions. I create samples without applying distortion. I use outdoor videos where a large number of people are moving. In such situations, I need to handle all possible orientations of the head and varying light condition; therefore, I need a large number of positive samples.

3.2.4 Classifier Cascade Training

In this step I train the classifier using the `cvCreateTreeCascadeClassifier()` function of OpenCV. I use positive training samples and the negative images to train the classifier. The output of this step is a classifier cascade in the form of an XML file that is used in the next
step to detect heads.

3.3 Head Detection

After we get a classifier from the training step, we use this classifier to classify regions as head or non-head in input images. I use the OpenCV function `cvHaarDetectObjects()` with some modifications.

The function scans the input image several times at different scales and finds the rectangular regions in the image classified as positives by the classifier cascade and returns those regions as a sequence of rectangles. We can control the scale parameter; for example, a scale of 1.1 means a 10% increase in the detection window size in each pass over the image. We can also control the minimum window size to detect. I added a maximum detection window size parameter.

3.4 Human Tracking

In this section I discuss my strategy for tracking humans from one frame to next frame. A block diagram of my algorithm is shown in Figure 3.5.

3.4.1 Particle Filter Tracking

For tracking I use a particle filter (Isard & Blake, 1998b; Doucet, Freitas, & Gordon, 2001). The particle filter is well known to enable robust object tracking (see e.g. Kang & Kim, 2005; Martinez, Knebel, & Thiran, 2004). I use the standard approach in which the uncertainty about an object’s state (position) is represented as a set of weighted particles, each particle representing one possible state. The filter propagates particles particles from time $i - 1$ to time $i$ using a motion model, computes a weight for each propagated particle using a sensor or appearance model, then resamples the particles according to their weights. The initial distribution for the filter is based on the location of the object the first time it is detected. Here are the steps in more detail:

1. **Predict:** I predict $p(\vec{x}_{j,i} | \vec{x}_{j,i-1})$, a distribution over head $j$’s position in frame $i$ given our belief in its position in frame $i - 1$. The motion model is described in the next section.

2. **Measure:** for each propagated particle $k$, I measure the weight $p(\vec{h}(k) | x_{j,k}^{(k)})$ using a color histogram-based appearance model. After computing the weight of each particle I normalize the weights to sum to 1.

3. **Resample:** I resample the particles to avoid degenerate weights. Without resampling, over time, the highest-weight particle would tend to a weight of one and other weights would tend to zero. Resampling removes many of the low weight particle and multiplies
Figure 3.5: Block diagram of the tracking algorithm. $D$ is the distance between a newly detected head and the nearest predicted location, $C$ is threshold (in pixels) less than the width of the tracking window, and $a$ is a constant, $a > 0 \leq 1$. 
the higher-weight particles. I thus obtain a new set of equally-weighted particles. I use
the resampling technique described in (Rui & Chen, 2001).

3.4.1.1 Motion Model

My motion model is based on a second-order auto-regressive dynamical model. The au-
toregressive model predicts the next state of the system based on some number of previous
states.

\[ y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \epsilon_t) \]

The linear auto-regressive model follows the form:

\[ y_t = b_0 + \sum_{i=1}^{p} b_i y_{t-i} + \epsilon_t. \]

where \( y_t \) is the variable value at time \( t \), \( b_i \) regression coefficient, \( p \) auto-regression rank and \( \epsilon_t \) is noise. In my model, I use \( p = 2 \), \( b_0 = 0 \), \( b_1 = 2 \) and \( b_2 = -1 \)

In particular, I predict each particle’s positions in frame \( i \) based on its location \( (x, y) \) in
previous two frames.

3.4.1.2 Appearance Model

Our appearance model is based on color histograms. I compute a color histogram \( \vec{h}_j \) in
HSV for each newly detected head and save it to compute the likelihood of particles in
future frames. To compute the likelihood I use the Bahattacharyya similarity coefficient.
The likelihood between the model histogram \( \vec{h}_j \) and the observed histogram \( \vec{h}^{(k)} \) as follows,
assuming \( n \) bins in each histogram:

\[ p(\vec{h}^{(k)} \mid \vec{x}^{(k)}_{j,i}, \vec{h}_j) = e^{-\lambda d(\vec{h}_j, \vec{h}^{(k)})} \]

where

\[ d(\vec{h}_j, \vec{h}^{(k)}) = 1 - \sum_{i=1}^{n} \sqrt{\vec{h}_j(i) \cdot \vec{h}^{(k)}(i)} \]

3.4.1.3 Normalizing Particles Weights

After getting the likelihood for each particle. I normalize the weights to sum to 1.
3.4.1.4 Resampling Particles

To avoid weight degeneracy I resample the particles in each frame. Without resampling, over time, the highest weighted particle tends to weight of one and other tends to zero. To resample the particles I use the technique described by Rui and Chen (2001). The result of resampling is to remove most low-weight particles and multiply particles with high weights. This step produces a new set of unweighted particles.

3.4.2 Trajectory Optimization

To reduce tracking errors, I introduce a simple trajectory optimization method, described in detail in this section.

3.4.2.1 Recovery from misses

Due to occlusion and appearance variation, we may not detect all heads in the first frame or when they initially appear. To solve this problem, I search for new heads in all regions of the image not predicted by the motion model for a previously tracked head. Any newly detected head within some distance $D$ of the predicted position of a previously tracked head is assumed to be associated with the existing trajectory and ignored. If the distance is greater than $D$, I create a new trajectory for that detection. I set $D$ to be 50% of the width of the detection window.

3.4.2.2 Reduction of false detections

Shadows and other non-head objects in the scene tend to produce false detections and tracking errors. See example in Figure 3.6. To remove these false detections, I use the head classifier to confirm tracked positions and eliminate trajectories not confirmed for some number of frames. To implement this, I use a trajectory weight for trajectory $j$, $0 \leq w_j \leq 1$. When head $j$ is first detected and its trajectory is initialized, I set $w_j = 1$. While tracking I confirm the tracked window through detection. The weights of trajectories not confirmed through classification are decreased by a constant $a$, and the weights of confirmed trajectories are reset to 1. Any trajectory whose weight is 0 is eliminated.

3.4.2.3 Tracking through temporary occlusion

The trajectory weighting scheme just described also serves to help track a head through a partial or full occlusion. As long as the occlusion is brief, the trajectory will not be deleted and can be recovered in a subsequent frame.
Figure 3.6: The figure shows the tracking error and detection response in consecutive frames. Red rectangle represent tracking window and green rectangle shows the detection.
Chapter 4
Experiments and Results

4.1 Overview

This chapter describes the experiments I have performed to evaluate the detection and tracking modules and the results on the evaluation.

The Viola and Jones detector achieved a 79.97% detection rate with 0.8 false positives per image in head detection and 7.07 missing heads per image. In tracking I achieved 76.8% true positive, 2.05 false positives per image and 8.2 missing heads per frame. There were average 35.35 heads per image. The average tracking time was 2 seconds per frame for frame size 640×480 on Intel Pentium 4 2.8GHz with 2GB RAM.

4.1.1 Training data

For training classifier I cropped the 20 × 20 pixels head region from videos frames collected from different places. I used 4325 positive samples and 2200 negative images to train the classifier. The detail parameters are given in Table 4.1. The training process took around 72 hours to train the classifier on Intel Pentium 4 2.8GHz with 2GB RAM. I used OpenCV haartraining utility to train the classifier.

4.1.2 Testing data

To evaluate the algorithm I created a hand labeled heads in each frame and compare with the tracking windows. For testing purpose I used real video taken from Mochit BTS station in Bangkok, Thailand (the sample frame is shown in Figure 1.1. I labeled the heads 20 × 20 pixels or greater. In my test video there are average 35.5 people per frame and I use total 1414 labeled heads. I labeled first few frames consecutively and then with gaps to test long tracking. The frame size is 640×480 pixels.

4.1.3 Implementation details

I use C++ with OpenCV without any special code optimization to develop the system. My approach is based on object detection (detection parameters are given in Table 4.2) and I am not using background subtraction. My algorithm can track both moving and static human in the scene, which also main limitation of previous approaches. I detect the heads and initialize the trajectories in first frame and track the heads from frame to frame. The detail implementation details are given in the following sections.
### Table 4.1: Training Parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>npos</td>
<td>4235</td>
<td>Number of positive samples</td>
</tr>
<tr>
<td>nneg</td>
<td>2200</td>
<td>Number of negative samples</td>
</tr>
<tr>
<td>nstages</td>
<td>14</td>
<td>Number of training stages</td>
</tr>
<tr>
<td>minhitrate</td>
<td>0.995</td>
<td>Minimum hit rate 99.5%</td>
</tr>
<tr>
<td>maxfalsealarm</td>
<td>0.5</td>
<td>False alarm rate 50%</td>
</tr>
<tr>
<td>mode</td>
<td>All</td>
<td>Use full set of upright and 45 degree rotated feature set</td>
</tr>
<tr>
<td>width, height</td>
<td>20</td>
<td>width and height</td>
</tr>
<tr>
<td>boosttypes</td>
<td>DAB</td>
<td>Discrete Ada Boost.</td>
</tr>
</tbody>
</table>

#### 4.1.3.1 Trajectory initialization and termination

In the first frame the detection process detects the heads and trajectories are initialized. The detection process detects when new heads appear in any frame. The trajectory optimization process will try to associate the heads with existing trajectories, if it fail to associate then a new trajectory will be initialized in the current frame.

If an object is in the exit zone (end of frame) and the motion model is predicts the location outside the frame then that trajectory will be removed.

#### 4.1.3.2 Identity management

It is also important to assign and maintain the objects identities (ID) automatically throughout the tracking. In my approach I assign unique ID to the trajectories during initialization. My tracking and trajectory optimization process ensure to maintain the object ID during tracking. The temporary lost tracks due to occlusion will get back the same ID to avoid identity switch.

#### 4.1.4 Results

There were average 35.5 people per frame and total 1414 heads. Some example frames are shown in Figure 4.1. The average correct tracking rate was 76.8%, with 2.05 false positives per frame and 8.2 missing heads per frame. The Tracking time per frame was 2 seconds/frame for frame size 640×480 on Intel Pentium 4 2.8GHz with 2GB RAM. The detail results are shown in table 4.3
Figure 4.1: Tracking results on real video frames. Red rectangles indicate detections, and green rectangles indicate actual head positions in the frame.
**Table 4.2:** Head Detection Parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalefactor</td>
<td>1.1</td>
<td>Scale factor used to scale the search window during scanning. Scale factor 1.1 means increasing window by 10%.</td>
</tr>
<tr>
<td>minneighbors</td>
<td>2</td>
<td>Number of neighbor rectangles ( (\text{minneighbors} - 1) ) that makes an object. All the groups which is less than ( \text{minneighbors} - 1 ) are rejected.</td>
</tr>
<tr>
<td>flags</td>
<td>0</td>
<td>Mode of operation. Currently only CV_HAAR_DO_CANNY_PRUNING can be set. It is used to do Canny edge detection to reject few image regions. The object with too few or too much edges will be rejected.</td>
</tr>
<tr>
<td>minsize</td>
<td>20 ( \times 20 )</td>
<td>Minimum size of object to detect.</td>
</tr>
</tbody>
</table>

**Table 4.3:** Tracking Results.

<table>
<thead>
<tr>
<th>Total Heads</th>
<th>Tracked</th>
<th>False Positives</th>
<th>Missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1414</td>
<td>1086</td>
<td>82</td>
<td>328</td>
</tr>
</tbody>
</table>
Chapter 5

Conclusion and Recommendation

5.1 Conclusion

Tracking people in high density crowds such as the one shown in Figures 1.1, 1.2, and 1.3 is a real challenge and is still an open problem. In this research study, I propose a new algorithm based on a combination of head detection, appearance-based tracking with a particle filter, and confirmation-by-classification. My experimental results demonstrate the promise of the method. It is particularly encouraging that the particle filter works well with a very small number of particles (20).

Followings are some important facts we found during the experiments I performed in my research:

- Mostly of the missed heads were those which were partially or fully occluded (self occlusion or inter-object occlusion). It might be possible to further improve detection under partial occlusion using a large training database.
- Most of the false detections were shadows or human body parts other than heads. A few false detections arose from background features such as holes in the ground.
- Particle filters work well for tracking people in crowds. Most previous researchers used 100 or more particles per object. Computation time increases linearly with the number of particles. I achieve good results using only 20 particles per object because of the confirmation-by-classification step.

5.2 Contribution

The first contribution of this research study is demonstrate the feasibility of detecting and tracking successfully in high density crowds. The most recent similar work to this is Zhao et al. (2008). In their experiments they tracked up to 33 people in a frame. A sample frame is also show in Figure 2.10. The second contribution of this research study is the confirmation-by-classification method of integrating detection and tracking. This algorithm improved the tracking results. Using confirmation-by-classification method I achieved good results with only 20 particles per object, saving a lot of computation time and improving the tracking rate. The tracking time is only 2 seconds per frame.

5.3 Recommendations

In future work I plan a more extensive evaluation of the method and improvements in the algorithm. I plan the following additional work to improve the system:
1. Currently pedestrian detection and tracking is in 2D. I plan to apply similar technique in 3D, by estimating the ground plane or using a known ground plane. Tracking in 3D simplifies the occlusion problem and reduces the false positives.

2. In this research study because of time limitations I use only one method for object detection. I plan to experiment with other object detection methods such as finding head contour in the image. This should improve both the detection and tracking rates.
References


