DETECTION AND TRACKING OF MULTIPLE HUMANS IN HIGH-DENSITY CROWDS

by

Irshad Ali

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

Examination Committee: Dr. Matthew N. Dailey (Chairperson)
Dr. Nitin V. Afzulpurkar (Member)
Dr. Raphael Duboz (Member)

Nationality: Pakistani
Previous Degree: Master of Engineering in Computer Science
Asian Institute of Technology (AIT), Thailand

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

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2012

Acknowledgment

I would like to thank ALLAH THE ALMIGHTY, who gave me courage and determination and helped me in all the steps of my life.

I would like to acknowledge the contributions, encouragements, help and support of the people around me. It would not have been possible to write this doctoral thesis without the help and support of my advisor, committee members, colleagues, friends, and family members.

First of all I thank to my advisor Matthew N. Dailey who gave me the chance to work in this interesting field of research. I feel motivated and encouraged every time I attend his meeting. His concrete feedback and practical guidance always helped me to stay focused in my research, which ultimately resulted in the completion of this dissertation. Besides my advisor I would also like to thank to Nitin V. Afzulpurkar and Raphael Duboz for their valuable suggestions, encouragement and constructive criticism.

This research was supported by the Higher Education of Pakistan (HEC). I am thankful Higher Education of Pakistan (HEC) for providing me financial support throughout my studies (M.S. and Ph.D.) at AIT and for funding me to present my work at an international conference.

It was an honor for me to be the part of the Vision and Graphics Lab (VGL) at AIT headed by Matthew. A special thanks goes to my colleagues in the VGL lab, especially Faisal Bukhari, Waheed Iqbal, Waheed Noor, Kifayat Ullah, Supawadee Chaivivatrakul, Kan Ouivirach, Shashi Gharti, Abdul Basit, Waqar Shahid, Jednipat Moonrinta, Zia Uddin, Sher Doudpota, and Sergio Goncalves, for their valuable feedback and support throughout my research at AIT.

I would like to thank Mr. Olivier Christian Nicole, laboratory and research manager for Computer Science and Information Management (CSIM) for providing excellent services to all students.

Last but not least, I would like to thank my parents, brother, and sisters, who have given me their unequivocal support throughout. I thank my wife Batool for her personal support, motivation, and great patience at all times; this dissertation would not have been possible without her support. I also owe my sincere love and gratitude to my children Haseeb, Hassam, and Haziqa for their love and prayers.

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. Visual surveillance system has become a popular research area in computer vision. Many algorithms exist to detect and track people in video streams. 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.

The emphasis in this work is on the development of a fully automatic algorithm to detect and track multiple humans in high-density crowds in the presence of extreme occlusion. Typical approaches such as background modeling and body part-based pedestrian detection fail when most of the scene is in motion and most body parts of most of the pedestrians are occluded. Under these conditions, we believe that the head is the only body part that can be robustly detected and tracked. In this dissertation, we therefore present a method for tracking pedestrians that detects and tracks heads rather than full bodies.

First, we use a Viola and Jones AdaBoost detection cascade to detect human heads in images. However, we do not rely particularly on the Viola and Jones method. Our method will improve upon the raw results of any head detector using a blend of detection, appearance-based tracking, and incremental head plane estimation. Any effective head detector could be plugged into our algorithm, capable of detecting heads in crowds.

Second, we integrate human detection and tracking into a single framework and introduce a *confirmation-by-classification* method for tracking that associates detections with tracks, tracks humans through occlusions, and eliminates false positive tracks. To track heads in video frames, we use particle filters, and for appearance modeling, we use color histograms.

Third, to further reduce false detections due to dense features and shadows, we introduce a method for estimation and utilization of a 3D head plane that reduces false positives while preserving high detection rates. The algorithm learns the head plane from observations of human heads incrementally, without any a priori extrinsic camera calibration information, and only begins to utilize the head plane once confidence in the parameter estimates is sufficiently high.

In an experimental evaluation, we show that confirmation-by-classification and head plane estimation together enable the construction of an excellent pedestrian tracker for dense crowds.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of the research presented in this dissertation is to develop techniques for automatic pedestrian detection and tracking in high-density crowds in the presence of extreme occlusion. In this chapter, I provide the motivation for the work, discuss the main challenges. discuss the contributions of the dissertation, provide a list of publications published as a part of the dissertation, and finally provide an outline of the dissertation.

1.1 Motivation

As public concern about crime and terrorist activity increases, the importance of 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 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 work, we focus on pedestrian tracking.

There are many commercial and open source visual surveillance systems available on the market, but these are manual and need additional resources such as human operators to observe the video. This is not feasible in high density crowds — observing the motion and behavior of every human in the crowd would be very difficult or impossible and error prone.

Automatic multiple human detection and tracking is a well studied problem, but the solutions proposed thus far are 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. The pedestrian tracking problem is especially difficult when the task is to monitor and manage a large crowd in gathering areas such as airports and train stations. See the example shown in Figure 1.1. 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 (Zhao et al., 2008; Wu & Nevatia, 2007; Wu, Nevatia, & Li, 2008; S. M. Khan & Shah, 2006; Berclaz, Fleuret, & Fua, 2006) , body part detection (Wu et al., 2008; Andriluka, Roth, & Schiele, 2008), or body shape models (Ramanan, Forsyth, & Zisserman, 2007; Zhao & Nevatia, 2004b; Zhao et al., 2008). These techniques are not applicable to heavily crowded scenes in which the majority of the scene is in motion (rendering background modeling useless) and most human bodies are partially or fully occluded. Under these conditions, we believe that the *head* is the only body part that can be robustly detected and tracked. In this dissertation I therefore present a method for tracking pedestrians that detects and tracks heads rather than full bodies.

Our system assumes a single static uncalibrated camera placed at a sufficient height so that the heads of people traversing the scene can be observed. For detection we use a standard Viola and Jones Haar-like AdaBoost cascade (Viola & Jones, 2001b), but the detector could be replaced generically with any real time detector capable of detecting heads in crowds. For

Figure 1.1: Examples of crowded scenes. People are standing or walking in both directions. There are other moving objects (motorcycles and cars) also present and moving faster than the rest of the crowd.

tracking we use a particle filter (Isard & Blake, 1998b; Doucet, Freitas, & Gordon, 2001) for each head that incorporates a simple motion model and a color histogram-based appearance model.

The main difficulty in using a generic object detector for human tracking is that the detector's output is unreliable; all detectors make errors. We have a tradeoff between detection rates and false positive rates: when we try to increase the detection rate, in most cases we also increase the false positive rate. However, we can alleviate this dilemma when scene constraints are available; detections inconsistent with scene constraints can be rejected without affecting the true detection rate. One such constraint is 3D scene information. We propose a technique that neither assumes known scene geometry nor computes interdependencies between objects. We merely assume the existence of a *head plane* that is parallel to the ground plane at the average human height. Nearly all human heads in a crowded scene will appear within a meter or two of this head plane. If the relationship between the camera and the head plane is known, and the camera's intrinsic parameters are known, we can predict the approximate size of a head's projection into the image plane, and we can use this information to reject inconsistent candidate trajectories or only search for heads at appropriate scales for each position in the image. To find the head plane, we run our head detector over one or more images of a scene at multiple scales, compute the 3D position of each head based on an assumed real-world head size and the camera's intrinsics, and then we find the head plane using maximum likelihood estimation.

When occlusion is not a problem, constrained head detection works fairly well, and we can use the detector to guide the frame-to-frame tracker using simple rules for data association and elimination of false tracks due to false alarms in the detector. However, when partial or full occlusions are frequent, data association becomes critical, and simple matching algorithms no longer work. False detections often misguide tracks, and tracked heads are frequently lost due to occlusion. To address these issues, we introduce a *confirmation-by-classification* method that performs data association and occlusion handling in single step. On each frame, we first use the detector to confirm the tracking prediction result for each live trajectory, then we eliminate live trajectories that have not been confirmed for some number of frames. This process allows us to minimize the number of false positive trajectories without losing track of heads that are occluded for short periods of time.

In an experimental evaluation, we shows that the proposed method provides for effective tracking of large numbers of people in a crowd. Using the automatically-extracted 3D head plane information improves accuracy, reducing false positive rates while preserving high detection rates. To my knowledge, the experiments in this dissertation represent the largest-scale individual human tracking study performed thus far, and the results are extremely encouraging. In future work, with further algorithmic improvements and runtime optimization, we hope to achieve robust, real time pedestrian tracking for even larger crowds.

1.2 Main challenges

Pedestrian detection and tracking in large crowds such as those shown in Figure 1.1 is still an open problem.

In high-density crowds, popular techniques such as background subtraction fail. We cannot

deal with this problem in ways similar to simple object tracking. In simple object tracking, we can assume that the number of concurrent objects to be tracked is not large and most parts of each object are visible most of the time. Inter-object occlusion and self occlusion in crowd situations makes detection and tracking much more challenging. Under these circumstances, we cannot perform segmentation, detection, and tracking separately. We should consider the entire problem holistically.

There are four main challenges in detecting and tracking pedestrians in crowds:

- **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 usually short-term.
- **Size of the visible region of the object**: as the density of a crowd increases, the visibility of each individual human decreases. Detecting and tracking partly visible people under dense conditions is very difficult.
- **Appearance ambiguity**: when target objects are small, their appearance tends to be less distinguished, making it more difficult to reliably track targets from frame to frame.

1.3 Contributions

In this dissertation, I propose, implement, and evaluate an automatic pedestrian tracker to track multiple people in high density crowds using a single static camera. The main contributions towards the solution in the presence of the problems mentioned before are as follows:

1. We combine a head detector and particle filter to track multiple people in high-density crowds.
2. We introduce a method for estimation and utilization of a *head plane* parallel to the ground plane at the expected human height that is extracted automatically from observations from a single, uncalibrated camera. The head plane is estimated incrementally, and when the confidence in the estimate is sufficiently high, we use it to reject likely false detections produced by the head detector.
3. We introduce a *confirmation by classification* method for tracking that associates detections, tracks humans through occlusions, and eliminates false positive tracks.
4. We have created, to the best of our knowledge, the most challenging existing dataset specifically for tracking people in high density crowds, and we have made the dataset with ground truth information publicly available for download[1].
5. We evaluate our algorithm on challenging datasets.

6. We publish[1] our source code, trained head detectors, and training data online for other researchers to share and improve upon.

1.4 Publications

I have published following papers as part of this dissertation:

1. Multiple Human Tracking in High-Density Crowds
   Irshad Ali and Matthew N. Dailey
   Image and Vision Computing, 2012,
   DOI: http://dx.doi.org/10.1016/j.imavis.2012.08.013

2. Multiple Human Tracking in High-Density Crowds
   Irshad Ali and Matthew N. Dailey
   In Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS), volume LNCS 5807, pp. 540–549.

3. Head Plane Estimation Improves the Accuracy of Pedestrian Tracking in Dense Crowds
   Irshad Ali and Matthew N. Dailey
   In Proceedings of Automation, Robotics and Vision (ICARCV), pp. 2054–2059

4. Multiple Human Tracking in High-Density Crowds
   Irshad Ali
   Master Thesis, 2009
   Computer Science & Information Management Department
   Asian Institute of Technology (AIT), Thailand.

1.5 Dissertation Outline

In this section, I provide an outline of the rest of the dissertation.

In Chapter 2, I provide a review of research related to head detection, tracking pedestrians in crowds, tracking applications, and commercial video surveillance products.

In Chapter 3, I describe our head plane estimation algorithm, consisting of the head detection method, how to estimate 3D head positions from 2D detections, linear head plane estimation, and nonlinear (maximum likelihood) head plane estimation.

In Chapter 4, I describe our tracking algorithm, consisting of a pseudo code description of the tracking algorithm, details on human head tracking using particle filters, and my "confirmation by classification" method.

[1] http://www.cs.ait.ac.th/vgl/irshad/

In Chapter 5, I provide details of our experimental evaluation of the method. First, I describe the training and testing data, then I give an overview of implementation details, explain the evaluation metrics I use, and finally, I provide the detection and testing results.

Finally, in Chapter 6, I conclude the dissertation, summarizing our contributions and providing some concluding remarks.

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. Human Detection in Crowds
2. Tracking Pedestrians in Crowds
3. Pedestrian Counting
4. Crowd Behavior
5. Commercial Products

2.1 Human Detection in Crowds

Detecting individuals in a crowded scene is challenging task for computer vision. 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.

2.1.1 Background Subtraction

Background modeling is one the most common techniques in human tracking, and almost all previous work uses background modeling to extract the foreground blobs from the scene.

(a) (b)

Figure 2.1: Pedestrian detection using background subtraction. (a) A sample frame. (b) Results of background subtraction. Reprinted from Zhao et al. (2008).

The idea is to analyze the extracted foreground blobs to detect individuals present in the image. See Figure 2.1 for an example. In application to crowd tracking, this kind of work 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.

2.1.2 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. We need a technique based on a robust feature set that can extract the objects from cluttered backgrounds under difficult illumination. I will provide a brief review of head detection in the next section; in this section, I review approaches related to general object detection.

The Viola and Jones (2001b, 2001a) approach is a very popular real time featuare-based object detection technique. The authors tested their method on face detection. To train their classifier, they used 4,916 faces (positive images) and 10,000 non-face images (negative images). They labeled all images by hand and scaled to 24×24 pixels. To test the algorithm, they used 130 images containing 507 faces. They achieved a 93.7% true positive rate with 422 false detections.

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

1. The method computes image features 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 2.2. The value of the integral image at discrete location (x, y) is computed using the equation:

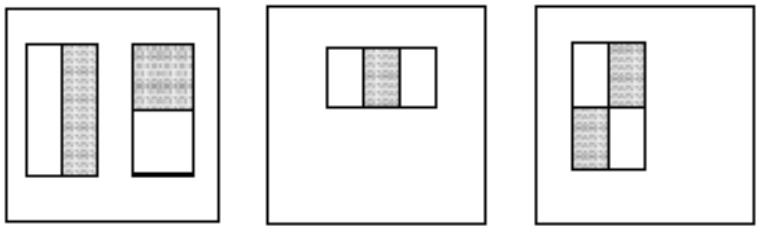
$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

   where $i(x', y')$ is the intensity of the pixel at location (x', y') in the original image. See Figure 2.3, which which shows the procedure for computing the integral image value at location (x, y).

2. The method learns classifiers using AdaBoost. AdaBoost selects weak classifiers from image features and produces a strong classifier.

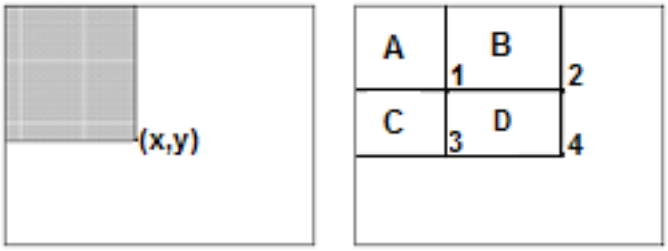
3. The weak learning algorithm, at each iteration, selects the feature that performs best in classifying the training set in terms of weighted error rate. The algorithm calculates an optimal threshold for each feature.

4. Finally, the method combines multiple strong classifiers (weak classifier ensembles) into a cascade that discards 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 2.4.



(a) (b) (c)

Figure 2.2: Example rectangle features in a detection window. A feature's value is computed as the difference between the sums of pixel intensities inside the white and black rectangles. (a) Two-rectangle features. (b) Three-rectangle feature. (c) Four-rectangle feature. Reprinted from Viola and Jones (2001b).



(a) (b)

Figure 2.3: Viola and Jones integral image. (a) The integral image value at location (x, y) is the sum of the intensities of the pixels in the original image from the top left 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 intensities in rectangular region A. The value at position 2 is sum of the pixel intensities in rectangular regions A and B. The value at position 3 is sum of pixel intensities s in rectangular regions A and C. The value at position 4 is the sum of pixel intensities in rectangular regions A, B, C and D. The sum of the pixel intensities within the rectangular region D can be computed as $4 + 1 - (2 + 3)$. Reprinted from Viola and Jones (2001b).

The Viola and Jones technique is one example of a feature-based technique. Another major category of object detection techniques is the category of shape-based techniques. As an example, Schindler and Suter (2008) perform object detection using a shape-based method. They achieve 83–91% detection rates at 0.2 false positives per image with an average processing time of 6.5 seconds for a 480×320 image. They first extract contour points on an object to represent its shape, then the compare the object shape against a template using a distance measure. Image contrast, reflection, and shadows are the main problems with this type of algorithm, as shown in Figure 2.6.

Dalal and colleagues (Dalal & Triggs, 2005; Dalal, 2006) use a linear SVM-based classifier with histograms of oriented gradient (HOG) descriptors for pedestrian detection. To detect the object, they sweep a 64×128 detection window over the image at different scales. A short description of the HOG approach (as shown in Figure 2.5) is given below:

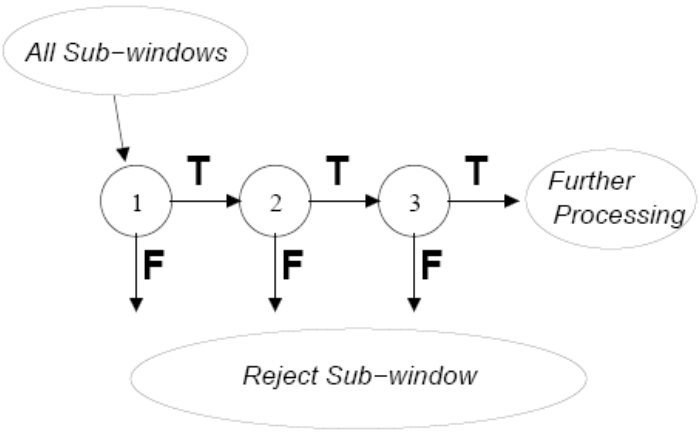


Figure 2.4: Object detection cascade structure. Reprinted from Viola and Jones (2001b).

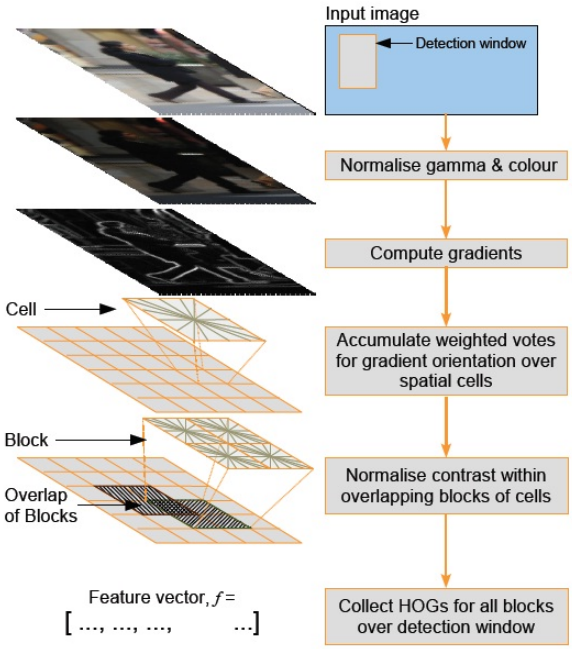


Figure 2.5: An overview of HOG feature extraction. Reprinted from Dalal (2006).

Table 2.1: Detection time comparison for different HOG-based approaches. Reprinted from Zhu et al. (2006).

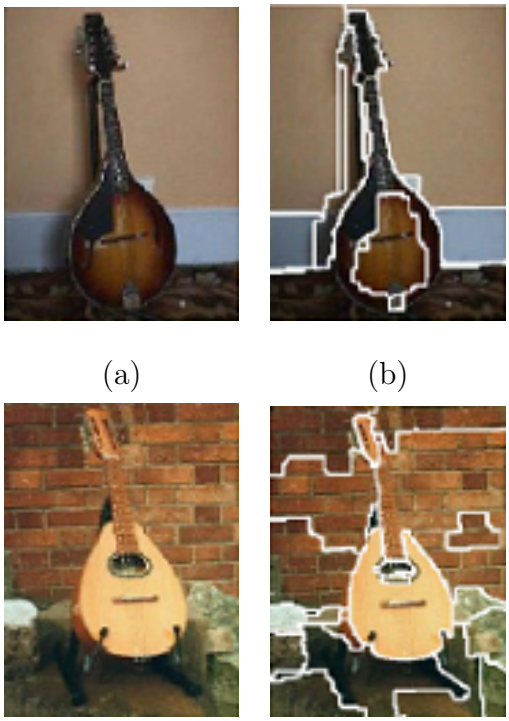
	Sparse scan (800 windows per image)	Dense scan (12,800 windows per image)
Dalal & Triggs	500ms	7sec
Cascade of Rect. features	11ms	55ms
Our approach (L1-norm)	26ms	106ms
Our approach (L2-norm)	30ms	250ms

1. The method normalizes the input image to reduce the influence of illumination effects.
2. The method computes the first order image gradients.
3. The method divides the image into small regions called "cells" (typically 8×8 pixels), then for each cell accumulates a single-dimensional histogram of the gradient vectors. The histogram divides the gradient vector's angle into 9 bins.
4. The method groups cells into blocks (2×2 cells) and normalizes each cell in the block. Next is a normalization process based on the sliding window (a cell appears in several blocks with different normalizations). This seems redundant, but the authors show that this normalization process improves performance. The normalized blocks are then called histograms of oriented gradient (HOG) descriptors.
5. Finally, the method collects the HOG descriptors for all overlapping blocks within the detection window and combines them into a big feature vector. The feature vector is then classified as a member of the object catagory or not a member of the object category.

Zhu et al. (2006) integrate the Viola and Jones classifier cascade with Dalal and Triggs' histograms of oriented gradient (HOG) technique, to reject most false detections early on. They also develop an optimized method for computing HOG features. They first compute gradient orientations and magnitudes and place them into 9-bin histograms. They then compute an integral image separately for each bin and store the results as different images. They use the resulting integral images to compute HOG features for any rectangular region. The algorithm requires only 4×9 integral image accesses per detection window. They show that their algorithm is 70 times faster than Dalal and Triggs' method, without reduction in detection performance (a comparison is shown in Table 2.1).

2.1.3 Head Detection

In high-density crowds, the head is the most reliably visible part of the human body. Detecting pedestrians through head detection is not new; a great deal of research in this direction



(a) (b)

(c) (d)

Figure 2.6: Challanges in contour-based object detection. (a) Image with good contrast. (b) Contours detected in image (a). (c) Image with poor contrast. (d) Contours detected in image (b). Reprinted from Schindler and Suter (2008).

(a) (b)

Figure 2.7: Head and shoulder detection. (a) Heads detected from foreground boundaries. (b) Ω-shape head and shoulder model. Reprinted from Zhao et al. (2008).

exists. Head detection in cluttered scenes is a very difficult task in which full visibility of the target cannot be guaranteed, and inter-object occlusion produces ambiguity of object appearance and shape. I review approaches to the problem here.

Zhao and colleagues (Zhao, Nevatia, & Wu, 2008; Zhao & Nevatia, 2004a) detect heads from object boundaries, edges, and foreground regions with previously-detected object regions removed. See Figure 2.7 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 body part detectors on examples of heads with shoulders, as well as other body parts. They use boosting with "edgelet" features to train their classifier. They detect the human body by combining the responses of different body part detectors, using a joint likelihood model. Their target is to track partially-occluded people. In high-density crowds, the visibility of the head and shoulder outline cannot be guaranteed. Thus, the method works best in low-density crowds in which the humans are isolated in such a way that at least their heads and shoulders are clearly visible.

Sim, Rajmadhan, and Ranganath (2008) use Viola and Jones-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 are created by normalizing histograms over windows positively classified by the first classifier. The main objective is to reduce the false positives of a standard classifier cascade using color bin images. They reduce the false alarm rate on their dataset from 35.9% to 23.9%, but the detection rate also decreases 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 it is not possible to recover misses during tracking.

Tang, Hung, and Chen (1997) and Nanda and Fujimura (2004) use a stereo camera to detect heads in the scene. They use an elliptical model for the human head. Using template matching, they extract head contours in a stereo image pair, and then they use a maximum likelihood detector to detect the human head. The main disadvantages of multiple camera systems are their higher computational requirements, cost, and setup/calibration effort. Most surveillance systems are based on monocular cameras, so it is difficult to use stereo cameras in applications with pre-existing hardware infrastructure.

Zou, Li, Yuan, and Xu (2009) propose an algorithm to detect moving human heads in a scene by extracting head contours from a background-subtracted image. After extracting all arcs, they classify the arcs into four categories using gradient information. They fit the arcs in different categories to an elliptical head contour model using least squares. The method works well in low-density crowds in which the humans are isolated, but in high-density crowds, algorithms based on background modeling fail.

Zhang, Huang, and Tan (2009) propose an algorithm to detect omega-shaped human head and shoulder profiles in images. Their head detector is a Viola and Jones cascade with Dalal and Triggs' histograms of oriented gradient (HOG) features. This approach works well in low-density crowds but is not scalable to high-density crowds, since we cannot guarantee the visibility of the heads and shoulders.

Our system assumes a single static uncalibrated camera placed at a sufficient height so that the heads of people traversing the scene can be observed. For detection, we use a standard Viola and Jones Haar-like AdaBoost cascade, but the detector could be replaced generically with any real time detector capable of detecting heads in high-density crowds.

2.1.4 Human Body Detection

Pedestrian detection, in which the entire body of a target humans is 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 and density of the crowd. If the crowd density is high, then correct detection of pedestrians is not possible.

Andriluka et al. (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 pedestrians in partially-occluded situations, such as when two pedestrians cross 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, for example, in Figure 2.8. The method learns a body part detector based on edgelet features using a boosting method. The response from all detectors is combined using a joint likelihood model. The system requires an average of 2.5 to 3.6 seconds per image. The algorithm is slow, and we cannot apply the body part detector concept in high-density crowds because the body parts are usually not visible.

Zhao et al. (2008); Zhao and Nevatia (2004a) use the 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. Their method handles inter-object occlusion in 3D space. The 3D human shape model is a good approach to handle the occlusion problem, but as already mentioned, we cannot use this approach in high-density crowds, because in high-density crowds, the human body is often not visible.

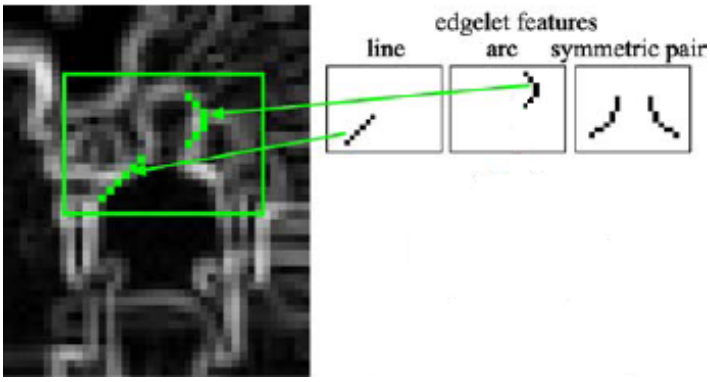
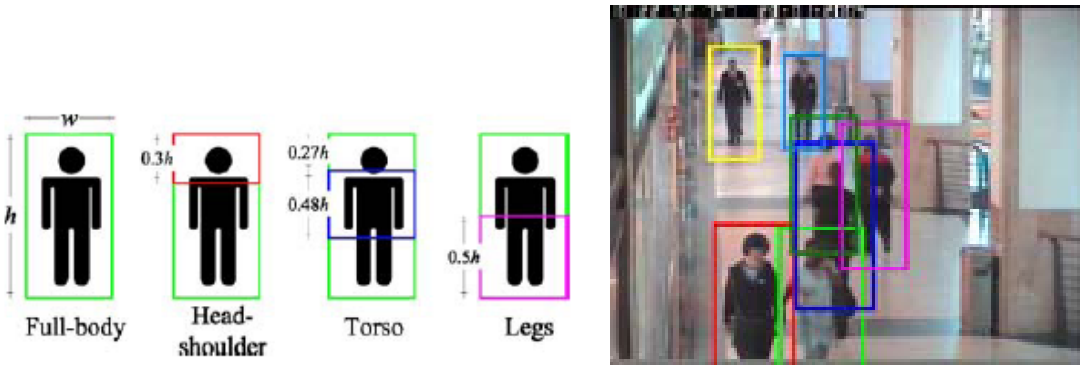


Figure 2.8: Edgelet features. Reprinted from Wu and Nevatia (2007).



(a) (b)

Figure 2.9: 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).

Figure 2.10: Pedestrian detection based on local and global cues. Reprinted from Leibe et al. (2005).

Leibe et al. (2005) integrate evidence over multiple detection 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. The system detects partially-occluded pedestrians in low-density crowds. See Figure 2.10 for an example.

2.2 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 at most a few people.

As previously discussed, when there are many humans moving in the scene, we face a high occlusion rate in the scene. Long, permanent, and nearly complete occlusions are the basic problems of pedestrian tracking in crowds. In this section I review various existing approaches.

2.2.1 Likelihood

While tracking objects from one frame to another frame, we search for the objects in the next frame. In the Bayesian approach, to perform this search, we attempt to find a model maximizing a likelihood based on appearance, shape, and so on for the object in the next frame.

Zhao and Nevatia (2004a); Zhao et al. (2008) propose a 3D human shape model and use color histograms for tracking people in crowds, as shown in Figure 2.11. As previously discussed,

(a) (b)

Figure 2.11: 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).

they first locate head positions and then estimate ellipsoid-based human shape. They combine detection and tracking in a 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 the best algorithms for tracking people, and their experiments include the largest crowd ever used, but the algorithm is still limited to 33 people. The idea of considering 3D space is good for crowd situations. Using 3D models, we can reduce false positives and reduce the effects of inter-object occlusion. The main limitation for my work is again that we cannot use human shape models 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 2.9. The method detects static body parts in each frame. The detection results are then input to a tracking module, which tracks parts from one frame to another. The method computes the likelihood from an appearance model based on color histograms and a dynamic model based on the detection response. This approach is good for robust tracking in case of partial occlusions, but as already discussed, it cannot be applied in high-density crowds.

Ramanan et al. (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 experiment with 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. The second method looks for entire people in each frame. They method assumes some key poses and builds models from those poses. 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. However, I only consider 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 of interest using a point distribution model. The model is somewhere between active shape models (Cootes, Taylor, Cooper, & Graham, 1995) and active appearance models (Cootes & Taylor, 2004). These models are statistical models based on shape and appearance. They describe objects in images and can be used to

detect the same object under deformation and pose change in another image. The algorithm requires a predefined region of interest. In the case of large crowds, the method might be extremely useful, but it may be a better choice to select features to track rather than tracking entire objects.

2.2.2 Multiple Object Tracking

Tracking people in crowds is a specific case of the more general problem of multiple object tracking. With multiple objects, the distribution of objects in the scene and their dynamics will be nonlinear and non-Gaussian. One of the most popular techniques for nonlinear, non-Gaussian Bayesian state estimation is the particle filter (Doucet, Freitas, & Gordon, 2001) method, also known as the sequential Monte Carlo method. Particle filters were first used by Isard and Blake (Isard & Blake, 1998b, 1998a) in computer vision. The particle filter method is well known to enable robust object tracking (see e.g.(M. D. Breitenstein, Reichlin, Leibe, Koller-Meier, & Gool, 2011; Kang & Kim, 2005; Martnez, Knebel, & Thiran, 2004; Vermaak, Doucet, & Perez, 2003; Z. Khan, Balch, & Dellaert, 2005; Okuma, Taleghani, Freitas, Little, & Lowe, 2004)). In my work, I use an object detector to guide a particle filter-based tracker. To track heads from frame to frame, we use the standard approach in which the uncertainty about an object's state (position) is represented as a set of particles with weights, with every particle in the set representing one state. The filter propagates particles from one frame to another frame 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 centered on the location of the object the first time it is detected. Here I review related approaches to multiple object tracking.

Ma, Yu, and Cohen (2009) consider the multiple object tracking problem as maximum a posteriori problem. A graph representation is used for all observations over time. They find the shortest path in a graph using motion and appearance of the object. When we apply segmentation on a cluttered scene, the object region is mixed with other object regions, and many objects share one image region. The authors apply merge, split, and mean shift operations on the graph to produce new hypotheses for objects.

Yang, Duraiswami, and Davis (2005) use color and edge orientation histogram features to track multiple objects using a particle filter. The original CONDENSATION algorithm shows poor performance when objects are occluded or the distance between objects is small. To overcome these problems, Kang and Kim (2005) set up a competition. This allows them to separate objects that are close to each other. The authors use a self-organizing map to build a human shape from body outlines. To track people from frame to frame, they use a hidden Markov model.

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

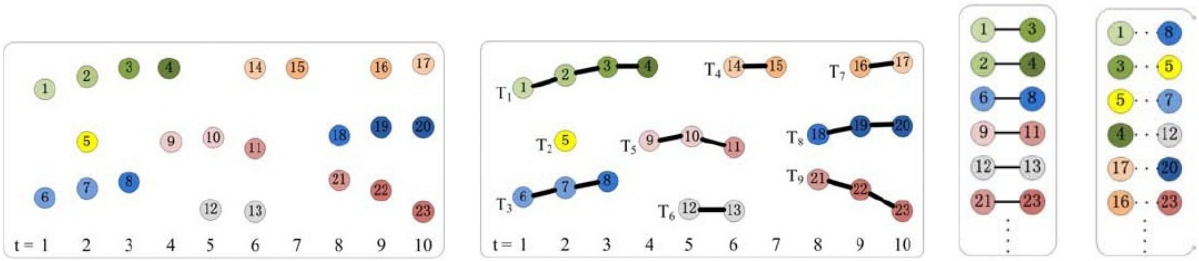


Figure 2.12: Tracking by detection. This method uses online training to associate detections with trajectories. (a) Initial detection. (b) Resulting small tracks. (c) Training examples (positive). (d) Training examples (Negative). Reprinted from Kuo et al. (2010).

They detect objects in 3D space and join those objects with hypothetical trajectories. They select the best trajectories using the generated hypotheses. Their method integrates detection and trajectory estimation problems, forming a single optimization problem.

Smith, Gatica-Perez, and Odobez (2005) present a Bayesian framework for tracking a variable number of interacting targets in the scene using a fixed camera. They use a particle filter to track objects across frames and a Bayesian framework to handle the multiple object tracking problem.

2.2.3 Tracking Through Detection

Building on recent advances in object detection, many researchers have proposed tracking methods that utilize object detection. These algorithms use appearance, size, and motion information to measure similarities between detections and trajectories. Many solutions exist for the problem of data association between detections and trajectories.

In the joint probabilistic data association filter approach (Rasmussen & Hager, 2001), joint posterior association probabilities are computed for multiple targets in Poisson clutter, and the best possible assignment is made on each time step.

Andriluka et al. (2008) combine detection and tracking in a single framework. To detect pedestrians, they use a part-based object detection model. As previously mentioned, the main objective of part based detection algorithms is to track objects under partial occlusion situations, such as when pedestrians cross each other on a walkway. The method extracts people-tracklets by analyzing consective frames and using these people-tracklets build models of pedestrians to tracking pedestrians over longer periods of time.

Reid (1979) generates a set of data-association hypotheses to account for all possible origins of every measurement over several time steps. The well-known Hungarian algorithm (Kuhn, 1955) can also be used for optimal matching between all possible pairs of detections and live tracker trajectories.

Recent work by van Gool and colleagues (M. D. Breitenstein et al., 2011; M. Breitenstein,

Reichlin, Leibe, Koller-Meier, & Gool, 2009; Kuo et al., 2010) uses an appearance-based classifier at each time step to solve the data association problem. See the example in Figure 2.12. In dense crowds, where appearance ambiguity is high, it is difficult to use global appearance-based data association between detections and trajectories; spatial locality constraints need to be exploited.

Huang, Wu, and Nevatia (2008) use a hierarchical association method to associate detections with trajectories. First, they generate reliable short trajectories by linking detection responses using conservative affinity constraints. Second, to form long trajectories, they associate short trajectories using more complex affinity measures. They formulate data association as a MAP problem and find optimal solutions using the Hungarian algorithm.

Mitzel, Horbert, Ess, and Leibe (2010) use the tracking-by-detection technique already described and perfomr segmentation in each frame. This approach is helpful when the detector misses detections in many frames. The authors use this approach to track multiple people from a mobile robot.

In my work, I combine head detection with particle filters to perform spatially-constrained data association. I use the particle filter to constrain the search for a detection for each trajectory.

2.2.4 Tracking Pedestrians in 3D

In object detection, we normally have a tradeoff between detection rates and false positive rates: if we try to increase the detection rate, in most cases we will also increase the false positive rate. However, we can avoid this dilemma when scene constraints are available; detections inconsistent with the constraints can be rejected without affecting the true detection rate. One such constraint is 3D scene information. In the literature, 3D information has been used for segmentation, occlusion reasoning, and to recover 3D trajectories. In this section I review both single- and multiple-view approaches.

2.2.4.1 Tracking from a Single View

The single-view approach, also known as the monocular approach, is the most frequently-used method in video surveillance, because it is low-cost, simple, and easy to deploy.

Rodriguez, Laptev, Sivic, and Audibert (2011) use head detection and 3D geometry estimation. They first compute the density of people in a scene, and, by combining the density information with the detected people, minimize an energy function to jointly optimize the estimates of the density and the locations of detected individuals in the scene. They use the scene geometry method proposed by Hoiem, Efros, and Hebert (2006, 2008). They select a few detected heads and compute vanishing lines to estimate the camera's height. After getting the camera height, the authors estimate the 3D locations of detected heads and compare each 3D location with the average human height to reject head detections inconsistent with the scene geometry. Rodriguez et al. only estimate the head plane in order to estimate the camera height, whereas we estimate the head plane incrementally from a series of head detec-

tions then use the head plane to reject detections once sufficient confidence in the head plane estimate is achieved. The Rodriguez et al. method compares detections to a fixed average human height to reject inconsistent heads. Our method is more adaptive and is only used if sufficient confidence in the head plane estimate is achieved.

Fengjun Lv, Zhao, and Nevatia (2006) propose a method for auto calibration from a video of a walking human. First, they detect the human's head and legs at leg-crossing phases using background subtraction and temporal analysis of the object shape. Then they locate the head and feet positions from the principal axis of the human blob. Finally, they find vanishing points and calibrate the camera. The method is based on background modeling and shape analysis and is effective for isolated pedestrians or small groups of people, but these techniques fail in high density crowds.

Zhao et al. (2008) use a known ground plane and a 3D part-based body model to segment the image and to track individuals in 3D. The basic idea of 3D tracking is to reduce inter-object occlusion in crowds. In their earlier work, (Zhao & Nevatia, 2004a) use a 3D ellipsoid to model the human body. They generate human hypotheses by detecting heads from foreground peaks. They assume a known ground plane in their work. They locate peoples' feet and transform them to 3D scene positions using the ground plane homography.

Rosales and Sclaroff (1999) track multiple people using moving blob segmentation and connected components. They assume that the objects are planar and that the depth of two opposite corners of the blob's bounding box are the same. They track these two feature points (bounding box corners) in consecutive frames and recover 3D points. Their aim in is to recover 3D trajectories to predict and handle occlusion problems in object tracking. The authors use an extended Kalman filter (EKF) to track blobs.

Hoiem et al. (2006, 2008) first estimate rough surface geometry (Hoiem, Efros, & Hebert, 2005, 2007) in the scene and then use this information to detect people at given locations in the scene. They estimate possible object locations before applying an object detector to the image. Their algorithm is based on recovery of surface geometry and camera height. They use a publicly available application, which divides the image into three classes: "ground," "vertical," and "sky," and five subclasses of "vertical:" planar surfaces facing "left," "center," and "right," and non-planar "solid" and "porous" surfaces. To recover the camera height, the authors use manually labeled training images; they use height distributions of objects (humans, vehicles) in the scene to learn the height of the camera. Finally, they put the object into perspective, modeling the location and scale in the image. They model interdependencies between objects in the scene, surface geometry, and viewpoint of the camera. Our algorithm works directly from the object detector's results, without any assumptions about scene geometry or interdependencies between objects. We first apply the head detector to the image and estimate a head plane parallel to the ground plane at the expected human height directly from detected heads without any other information. The head plane estimate is updated incrementally, and when the confidence in the estimate is sufficiently high, we use it to reject false detections produced by the head detector.

2.2.4.2 Tracking from Multiple Views

Most researchers use the previously-discussed monocular approach for tracking, since it is cheap, 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 benefit of the multiple view approach is that we potentially obtain better knowledge of 3D locations. The main disadvantage of multiple camera tracking systems is their higher computational requirements and the overhead for calibration.

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

Mittal and Davis (2003) use many synchronized cameras placed far from each other. The objective is to segment, detect, and track humans from multiple cameras. The authors introduce a region-based stereo algorithm. The algorithm searches for and finds points on the target. At the end, the method merges evidence from all cameras. The method thus solves the occlusion problem with moroe optimal detection and tracking. In a test based on 200 frames containing six humans, the authors experimented with four, eight, and sixteen cameras, finding that four cameras was sufficient for successful tracking.

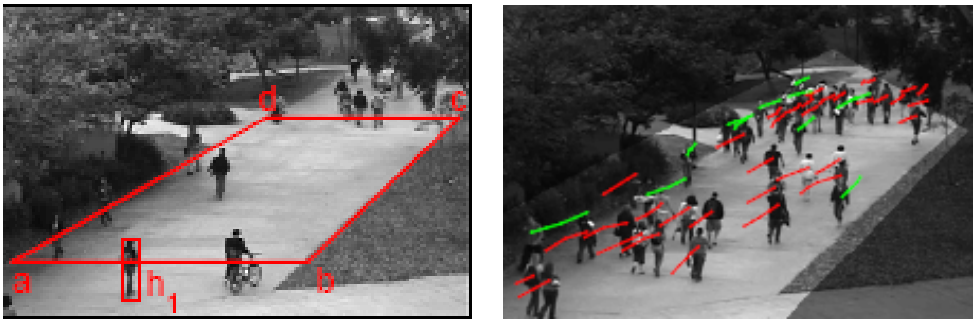
2.3 Pedestrian Counting

Pedestrian counting is a very important application of crowds. Different size of crowds have different problems and they should be solved differently. Lee, Goh, and Lam (2005) define the level of service boundaries for different sized crowds moving in both directions regarding area occupancy, pedestrian flow, and walking speed. They provide a clear idea of problems for different levels of crowds. They categorize the crowd as: free flow, restricted flow, dense flow, and jammed flow, based on crowd density (number of pedestrians per unit area).

Zhan, Monekosso, Remagnino, Velastin, and Xu (2008) provide an excellent survey on research going on in computer vision about crowd analysis and its applications. They also discuss and provide survey of research on crowds in other disciplines and how those research can be used in computer vision.

Chan et al. (2008) present a system to estimate dynamic crowd size, as shown in Figure 2.13. 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 the correspondences.

Rabaud and Belongie (2006) use a KLT-based tracker. KLT is a feature tracking algorithm. They use KLT to extract feature trajectories from videos. The feature trajectories are then clusterd. Finally, the method estimates the number of moving objects in a scene from the trajectory clusters.



(a) (b)

Figure 2.13: 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).

2.4 Pedestrian Behavior

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

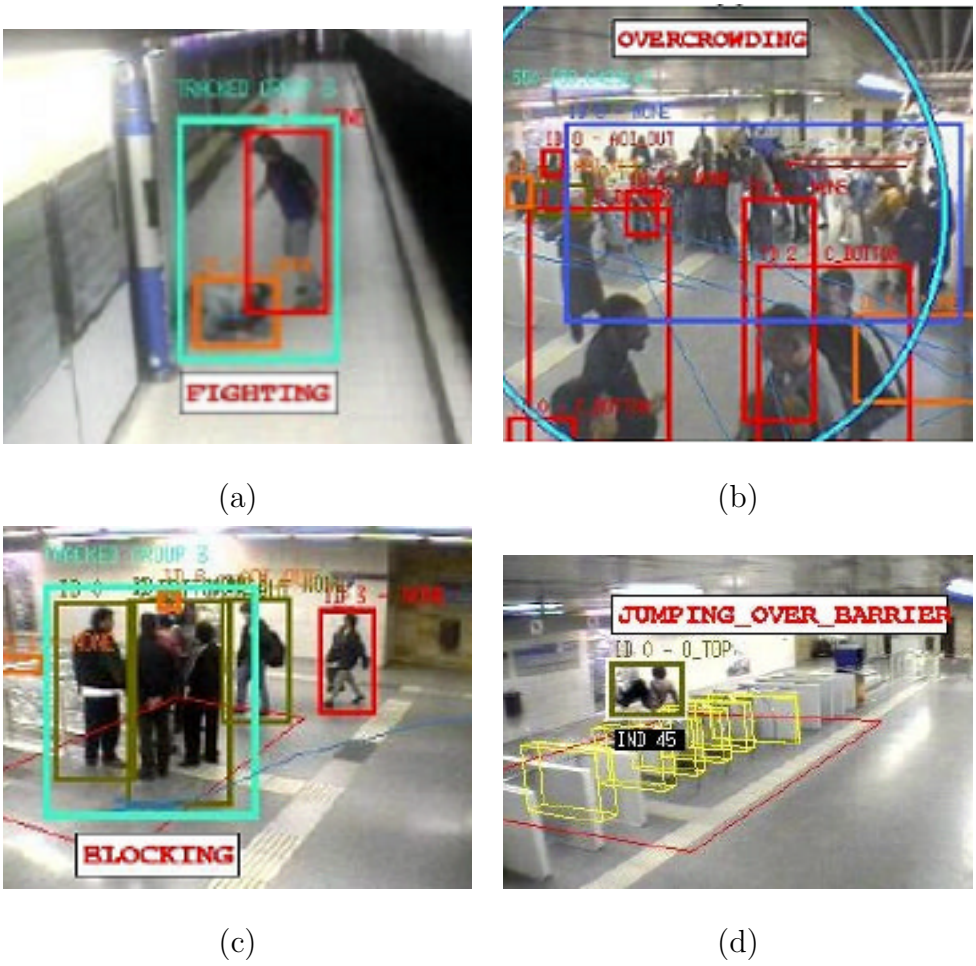
Cupillard et al. (2004) recognize isolated individuals, groups of people, or crowd behavior in scenes using multiple cameras. They detect motion and track objects and groups of objects from frame to frame. To estimate the positions and dimensions of people accurately, they combine the results from multiple cameras. For each tracked individual, they divide action (some example of which are shown in Figure 2.14) 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. **Scenarios** combine 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.

Z. Khan et al. (2005) present a particle filter and Markov chain Monte Carlo based algorithm to model 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.

Ge et al. (2011) use a hierarchical clustering algorithm to divide low- and medium-density crowds into small groups of individuals traveling together. See the example in Figure 2.15.



(a) (b)

(c) (d)

Figure 2.14: 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).

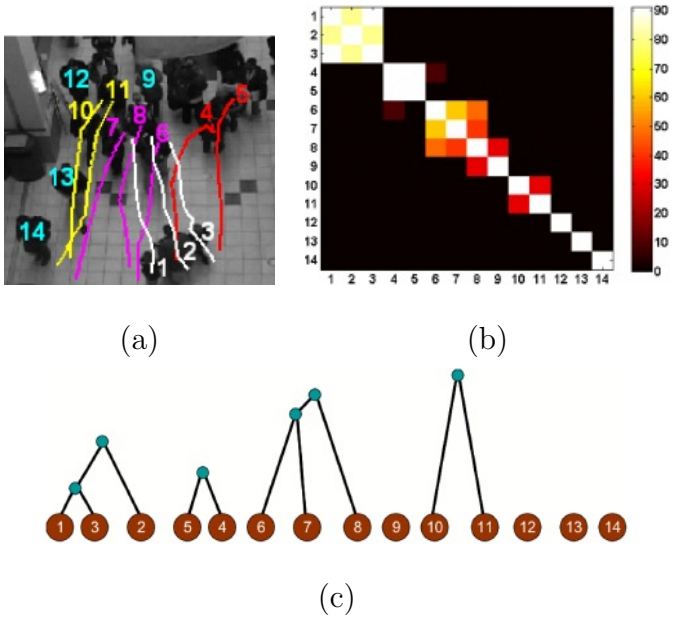


Figure 2.15: Detection of small groups in a crowd. (a) Detection of four groups. (b) Pairwise grouping metric. (c) Hierarchical clustering results. Reprinted from Ge et al. (2011).

To discover the group structure, the authors detect and track the moving individuals in the scene. The method has very interesting applications such as abnormal event detection in crowds and discovering pathways in crowds.

2.5 Commercial Products

There are a variety of commercial products (hardware and software) available for video surveillance. Table 2.2 shows a list of features and provides short descriptions of the available video surveillance systems. I review in detail only those intelligent automatic surveillance systems that can be used for people detection and tracking.

2.5.1 Brave Crowd Monitor, from Bravevideo

Brave Crowd Monitor (Bravevideo, 2009) is an intelligent video system designed to monitor crowds. The scene is divided into multiple regions, and the system computes and shows the density of the crowd in each region, as shown in Figure 2.16. The system triggers and alarm if the density of the crowd in any region exceeds a limit.

Table 2.2: Some important features of available commercial intelligent video surveillance systems.

Feature	Description
Crowd monitoring	Estimates crowd size, crowd density, and queue length.
Exit lane	Detects people going the wrong way.
Human tailgating or piggybacking	Detects more then one person tailgating or piggybacking on a single access card.
Tailgating or piggybacking vehicles	Detects more than one vehicle tailgating or piggybacking on a single access way.
Loitering	Detects humans or vehicles loitering in restricted areas.
Perimeter intrusion	Detects and tracks people, vehicles, or other object intruding into secure areas.
Removed stationary objects	Detects the removal or damage of fixed objects.
Long stay of vehicles	Detects vehicles stopped near prohibited areas for a long period of time.
Turnstile violation	Detects people hopping over or crawling under access gates.
Unaccompanied objects	Detects objects unaccompanied for a long time.
Face detection and tracking	Detects human faces in the scene and tracks them in video.
Object tracking	Detects and tracks any moving objects in video and shows their trajectories.
Intrusion detection	Tracks objects once they enter a ROI or cross a trip-wire.
Camera diagnostics	Detects of camera blurring, camera obstruction, or camera movement.
People counting	Statistical information on people counts.
Smoke detector	Detects fire or smoke in a ROI.
Queue waiting time	Records waiting time for each person in a queue.
License plate control	Detects license plates and records car numbers.
Abnormal behavior	Detects abnormal behavior such as fighting, running, and jumping.
ATM security	Provides ATM user safety, preventing ATM skimming and hidden camera use.

Figure 2.16: An example crowd density application. The scene is divided into 9 regions. The red numbers show the density of the crowds in each region. Reprinted from Bravevideo (2009).

Figure 2.17: An example people counting application. The system counts and shows the number of people in the scene. Reprinted from Bravevideo (2009).

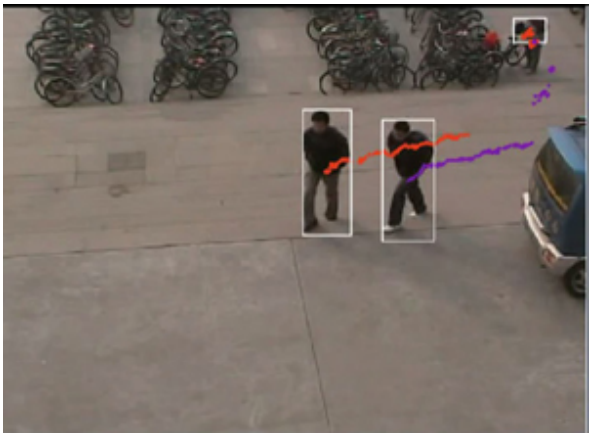


Figure 2.18: An example people tracking application. The system tracks moving objects in video sequences. Reprinted from Bravevideo (2009).

2.5.2 Brave Counter, from Bravevideo

Brave Counter (Bravevideo, 2009) is an intelligent video system designed to count the number of people going in and out of a door, as shown in Figure 2.17. The system analyzes the crowd's direction and counts the flow in each direction. The system is useful for controlling and gathering statistical information on entrances and exits of important places such as airports, railway stations, supermarkets, banks, hospitals, and schools.

2.5.3 Brave Tracker, from Bravevideo

Brave Tracker (Bravevideo, 2009) is an intelligent video system designed to track moving objects in a scene. The system has the ability to track objects using a pan tilt zoom (PTZ) camera and a normal stationary camera. The system draws trajectories in different colors, as shown in Figure 2.18. When used with a PTZ camera, the system tracks the object by adjusting the camera position and always shows the object in center of the image.

2.5.4 Face Detector and Tracker, from Analytic Video

Face Detector and Tracker (Analyticvideo, 2009) is designed to detect and track human faces in real time. The application is used in both video surveillance and biometric systems. The system can take input from multiple cameras and automatically adjust camera zooming and positioning for face capture with optimal view and resolution. The minimum detectable size of the face is 60×80 pixels. The detection rate is quoted as 0.95, and the miss rate is quoted at 0.05.

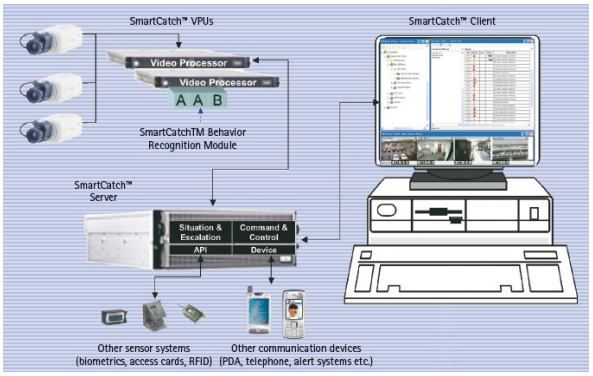


Figure 2.19: SmartCatch behavior recognition platform. Reprinted from NEC (2009).

2.5.5 Object Tracker, from Analytic Video

Object tracker (Analyticvideo, 2009) uses a fixed camera setup to detect and track moving objects in video sequences. The system is capable of recording the trajectory, speed, and size of moving objects in 2D and 3D metric space. The system also detects unusual camera movement and generates alarm messages.

2.5.6 SmartCatch, from NEC Corporation

NEC corporation is multinational company with offices in many countries. NEC in Australia, Singapore, and Hong Kong produce a security product called SmartCatch. SmartCatch (NEC, 2009) is na intelligent surveillance system comprising software and hardware as shown in Figure 2.19. Some important features of SmartCatch are as follows:

1. **Crowd Management**: SmartCatch detects changes in crowd size in a specific area. It computes the occupancy rate for that area and triggers an alarm if the rate is higher than a threshold.
2. **Exit Lane**: SmartCatch detects any people going the wrong way and generates an alarm.
3. **Tailgating/Piggybacking**: SmartCatch detects whether two or more people try to an area using the same access card.
4. **Loitering**: If one or more people remain in a sensitive area longer than a defined time, the system detects and tracks them individually.

2.5.7 Video Analytics, from IOImage

Video Analytics (IoImage, 2009) is an intelligent surveillance system that can be used with stationary cameras and pan tilt zoom (PTZ) cameras, as shown in Figure 2.20. The important features of the system are as follows:

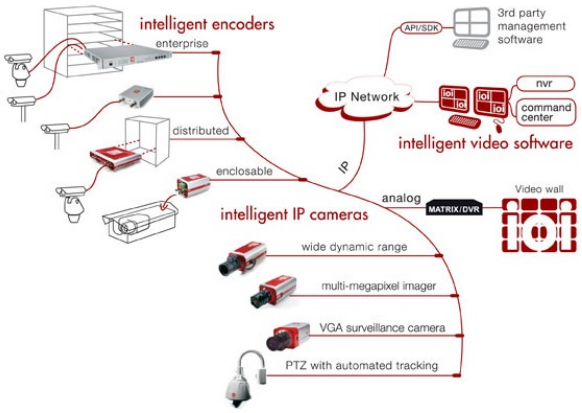


Figure 2.20: Video Analytics product layout. Reprinted from IoImage (2009).

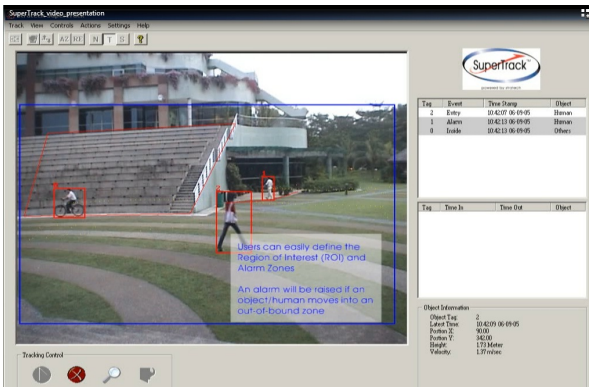


Figure 2.21: A screen shot of the Super Track application. Reprinted from StratechSystems (2009).

1. **Regional Entrance**: The system detects objects moving into a given prohibited area. The user can also set directional criteria for moving objects.

2. **Loitering**: If one or more people remain longer than a predefined amount of time in a sensitive area, the system detects and tracks them individually.

3. **Autonomous PTZ Tracking**: autonomous PTZ tracking is a self-directed vision-guided tracking function to target a moving object in a scene. The system automatically adjusts the camera pan, tilt, and zoom to keep the target in sight and in focus. When a moving object is detected from a static camera, the system automatically transfers responsibility for tracking that object to a PTZ camera.

4. **Automatic Optimization** : the system is capable of learning the background of the scene in view to enable automatic adjustment for dynamic lighting changes, to perform automatic image quality adjustment, and to perform automatic adjustment of video stream resolution and frames per second.

2.5.8 Super Track, from Stratech Systems Limited

Super Track (StratechSystems, 2009) is a software system for intelligent video surveillance. A screen shot is shown in Figure 2.21. The system is designed for object detection, classification and tracking, real-time crowd detection (people counting), intrusion detection, static object removal, unattended object detection, and intelligent alarm and event management.

2.5.9 Summary

Although there are many commercial products on the market doing human detection and tracking, they are limited to a few features, with the focus on helping human operators. I could not find any product for real time fully automatic crowd detection and tracking. After doing this market review, we can understand the importance and market demand for human detection and tracking in high density crowds. It needs more research and focus from both academia and the industry to solve the problem, which has a wide variety of market-oriented applications.

Chapter 3

Head Detection

For object detection, although more promising algorithms have recently appeared (Leibe, Leonardis, & Schiele, 2008; Dalal & Triggs, 2005), we currently use a standard Viola and Jones AdaBoost cascade (Viola & Jones, 2001b, 2001a) trained on Haar-like features offline with a few thousand example heads and negative images. At runtime, we use the classifier as a detector, running a sliding window over the image at the specific range of scales expected for the scene.

3.1 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 Haar-like features is computed over each training set image (as shown in Figure 2.3). A strong classifier that obtains a weighted vote of an ensemble of weak classifiers using individual features to build decision tree stumps is induced using AdaBoost (as shown in Figure 3.1). There are two good tutorials (Bajracharya, 2007; Seo, 2007) on how to train and detect pedestrians and faces using the OpenCV AdaBoost classifier cascade algorithm.

3.1.1 Positive Examples

To train an AdaBoost classifier, we need to prepare a set of positive images containing the object. 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 image. I select positive examples from several videos taken from different places. To select the positive examples from images, I wrote a program using OpenCV and C++ (examples are shown in Figure 3.2) allowing a human to select the locations of heads in the frame. The location and width/height of each head are used later in the training process.

The Viola and Jones method requires a large number of training examples to achieve a good detection rate. In my experiment I found that 5396 examples of heads were needed to achieve reasonable results.

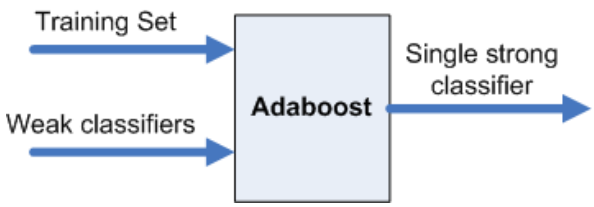


Figure 3.1: The main concept of Adaboost.

Figure 3.2: Example positive images used to train the head classifier.

3.1.2 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 a 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 3.3.

3.1.3 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.

3.1.4 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 classier. 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.

Figure 3.3: Example negative images used to train the classifier.

3.2 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 OpenCV `cvHaarDetectObjects()` function searches for an object in an input image. It performs many iterations. In each iteration, it uses a different scale and finds the rectangular regions in the image classified as positives by the classifier cascade. The function returns the locations of the objects in the image 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 to limit the size of the detection window at runtime.

Chapter 4

Tracking and Confirmation by Classification

In this chapter, first, I provide an introduction to our tracking algorithm. Second, I provide a summary of our head detection and tracking algorithm in pseudocode. Third, I give the details of each of the main components of the system. A block diagram is shown in 4.1

4.1 Introduction

The pedestrian tracking problem is especially difficult when the task is to monitor and manage a large crowd in gathering areas such as airports and train stations. We believe that the head is the only body part that can be robustly detected and tracked in these situations. We use a static camera placed at a sufficient height so that the heads of people traversing the scene can be observed.

To detect heads I use the Viola and Jones (2001b) technique. Details are given in Chapter 3. To reduce false positives produced by the head detector, we propose a technique for head plane estimation. Details are given in Chapter 5.

The tracker module uses a probabilistic model consisting of a motion model and an appearance model. The probabilistic model is based on a particle filter that is used to track heads from frame to frame. The motion model is used to predict the positions of heads in the next frame. The appearance model model gives a likelihood based on a color histogram, which is used to update the object state.

When occlusion is not a problem, constrained head detection works fairly well, and we can use the detector to guide the frame-to-frame tracker using simple rules for data association and elimination of false tracks due to false alarms in the detector. However, when partial or full occlusions are frequent, data association becomes critical, and simple matching algorithms no longer work. False detections often misguide tracks, and tracked heads are frequently lost due to occlusion. To address these issues, we introduce a *confirmation-by-classification* method that performs data association and occlusion handling in single step. On each frame, we first use the detector to confirm the tracking prediction result for each live trajectory, then we eliminate live trajectories that have not been confirmed for some number of frames. This process allows us to minimize the number of false positive trajectories without losing track of heads occluded for a short period.

The details of these modules are given in following sections.

4.2 Summary

1. Acquire input crowd video V.
2. In first frame v_0 of V, detect heads. Let $\boldsymbol{x}_{i,0} = (x_i, y_i)$, $i \in 1 \dots N$ be the 2D positions

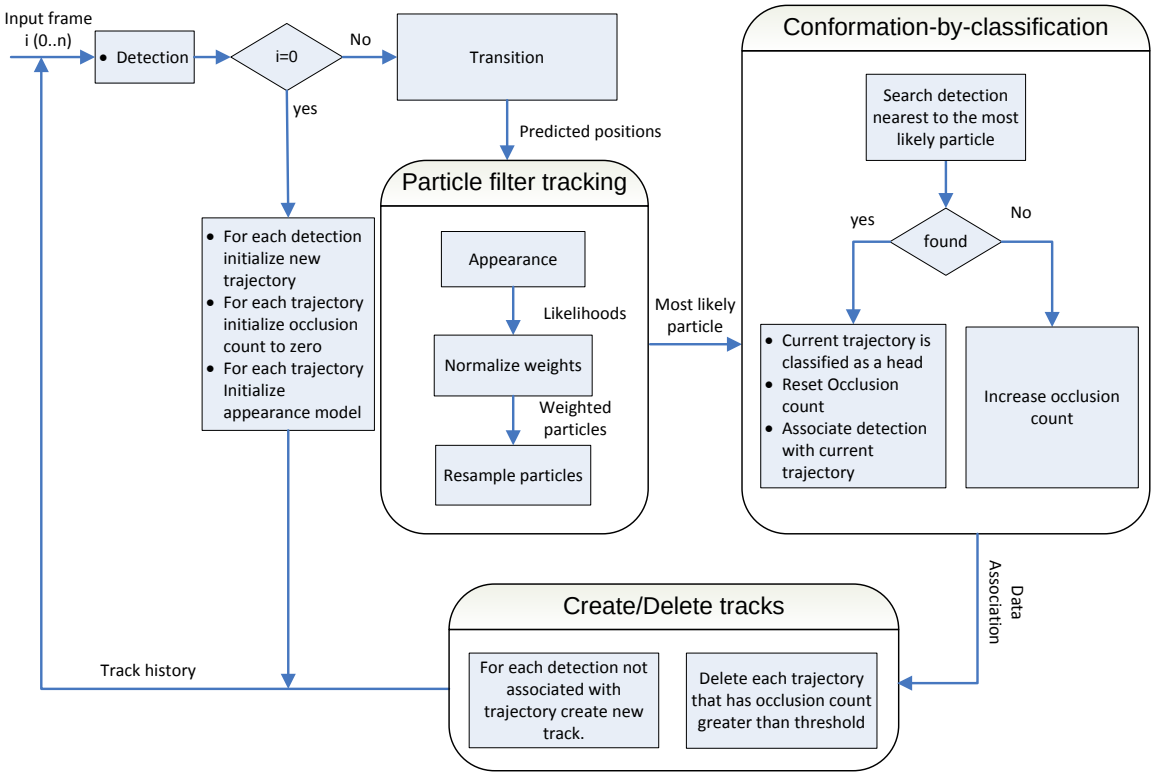


Figure 4.1: Block diagram of the tracking algorithm.

of the centers and let $h_i, i \in 1 \ldots N$ be the heights of the detection windows for the detected heads.

3. For each detected head i, compute the approximate 3D location $\boldsymbol{X}_i = (X_i, Y_i, Z_i)$ corresponding to $\boldsymbol{x}_{i,0}$ and h_i.

4. Find the 3D plane $\boldsymbol{\pi} = (a, b, c, d)$ best fitting the 3D locations $\boldsymbol{X}_i$ using RANSAC then refine the estimate of $\boldsymbol{\pi}$ using Levenberg-Marquardt to minimize the sum squared difference between observed and predicted head heights.

5. From the error covariance matrix for the parameters of $\boldsymbol{\pi}$, find the volume V of the error ellipsoid as an indicator of the uncertainty in the head plane.

6. Initialize trajectories T_j, $j \in 1 \ldots N$ with initial positions $\boldsymbol{x}_{j,0}$.

7. Initialize occlusion count O_j for each trajectory j to 0.

8. Initialize the appearance model (color histogram) $\boldsymbol{h}_{j,0}$ for each trajectory from the region around $\boldsymbol{x}_{j,0}$.

9. For each subsequent frame v_i of input video,

   (a) For each existing trajectory T_j,

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

      ii. Compute the color histogram $\boldsymbol{h}_{j,i}^{(k)}$ and likelihood $p(\boldsymbol{h}_{j,i}^{(k)} \mid \boldsymbol{x}_{j,i}^{(k)}, \boldsymbol{h}_{j,i-1})$ for each particle k using the appearance model.

iii. Resample the particles according to their likelihood. Let k_j^* be the index of the most likely particle for trajectory j.

(b) Perform confirmation by classification:

i. Run the head detector on frame v_i and get 3D locations for each detection.

ii. If head plane uncertainty V is greater than threshold, add the new observations and 3D locations, reestimate $\boldsymbol{\pi}$, and recalculate V (Steps 4–5).

iii. If head plane uncertainty V is less than threshold, use the 3D positions and current estimate of $\boldsymbol{\pi}$ to filter out detections too far from the head plane. Let $\boldsymbol{x}_l, l \in 1 \ldots M$ be the 2D positions of the centers of new detections after filtering.

iv. For each trajectory T_j, find the detection $\boldsymbol{x}_l$ nearest to $\boldsymbol{x}_{j,i}^{(k_j^*)}$ within some distance C. If found, consider the location classified as a head and reset O_j to 0; otherwise, increment O_j. In our experiments, we set C to 75% of the width (in pixels) of head j.

v. Initialize a new trajectory for each detection not associated with a trajectory in the previous step.

vi. Delete each trajectory T_j that has occlusion count O_j greater than a threshold and history length $|T_j|$ less than track survival threshold.

vii. Deactivate each trajectory T_j with occlusion count O_j greater than threshold and history length $|T_j|$ greater than or equal to track survival threshold.

4.3 Particle Filter

We use particle filters (Isard & Blake, 1998b; Doucet et al., 2001) to track heads. The particle filter is well known to enable robust object tracking (see e.g.(M. D. Breitenstein et al., 2011; Kang & Kim, 2005; Martnez et al., 2004; Vermaak et al., 2003; Z. Khan et al., 2005; Okuma et al., 2004)). We use the standard approach in which the uncertainty about an object's state (position) is represented as a set of particles with weights. Every particle in the set represents one state. Our method automatically initializes separate filters for each new trajectory. The initial distribution of the particles is centered on the location of the object the first time it is detected. The filters propagate particles from frame $i-1$ to frame i using a motion model then compute weights for each propagated particle using a sensor or appearance model. Here are the steps in more detail:

1. **Predict**: we predict $p(\boldsymbol{x}_{j,i} \mid \boldsymbol{x}_{j,i-1})$, 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, we measure the likelihood $p(\boldsymbol{h}_{j,i}^{(k)} \mid \boldsymbol{x}_{j,i}^{(k)}, \boldsymbol{h}_{j,i-1})$ using a color histogram-based appearance model. After computing the likelihood of each particle, we treat the likelihoods as weights, normalizing them to sum to 1.

3. **Resample**: we resample the particles to avoid degenerate weights, obtaining a new set of equally-weighted particles. We use sequential importance resampling (SIR) (Doucet et al., 2001).

4.3.1 Motion Model

We use a second-order auto-regressive dynamical model to predict the position of the object based on the previous history (last two consecutive frames). In particular, we assume the simple second-order linear autoregressive model

$$\boldsymbol{x}_{j,i} = 2\boldsymbol{x}_{j,i-1} - \boldsymbol{x}_{j,i-2} + \boldsymbol{\epsilon}_i$$

in which $\boldsymbol{\epsilon}_i$ is distributed as a circular Gaussian.

4.3.2 Appearance Model

Our appearance model uses color histograms to compute particle likelihoods. We use the simple method of quantizing to fixed-width bins. We use 30 bins for hue and 32 bins for saturation. Learning optimized bins or using a more sophisticated appearance model based on local histograms along with other information such as spatial or structural information would most likely improve our tracking performance, but the simple method works well in our experiments.

Whenever we create a new track, we compute a color histogram $\boldsymbol{h}_j$ for the detection window in HSV space and save it for comparison with histograms extracted from future frames. To extract the histogram from a detection window, we use a circular mask to remove the corners of the window.

We use the Bhattacharyya similarity coefficient between model histogram $\boldsymbol{h}_j$ and observed histogram $\boldsymbol{h}^{(k)}$ to compute a particle's likelihood as follows, assuming n bins in each histogram:

$$p(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{h}') \propto e^{-d(\boldsymbol{h},\boldsymbol{h}')} \qquad \text{(Equation 4.1)}$$

where

$$d(\boldsymbol{h}, \boldsymbol{h}') = 1 - \sum_{b=1}^{n} \sqrt{h_b h'_b}$$

and h_b and h'_b denote bin b of $\boldsymbol{h}$ and $\boldsymbol{h}'$, respectively.

When we track an object for a long time, its appearance will change, so we update the track histogram for every frame in which the track is confirmed.

4.3.3 Normalizing Particle Weights

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

4.3.4 Resampling Particles

To avoid weight degeneracy I resample the particles in each frame. Without resampling, over time, the highest weighted particle tends to a weight of one and others tend 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 make multiple copies of particles with high weights. This step produces a new set of unweighted particles.

4.4 Confirmation by Classification

To reduce tracking errors, we introduce a simple confirmation-by-classification method, described in detail in this section.

4.4.1 Recovery from Misses

Many researchers, for example Breitenstein et al. (M. D. Breitenstein et al., 2011), initialize trackers for only those detections appearing in a zone along the image border. In high density crowds, this assumption is invalid, so we may miss many heads. 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, in each image, we 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 C 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 C, we create a new trajectory for that detection. We currently set C to be 75% of the width of the detection window.

4.4.2 Data Association

With detection-based tracking, it is difficult to decide which detection should guide which track. Most researchers compute a similarity matrix between new detections in the frame and existing trajectories using color, size, position, and motion features then find an optimal assignment. These solutions work well in many cases, but in high density crowds in which the majority of the scene is in motion and most of the humans' bodies are partially or fully occluded, it tends to introduce tracking error such as ID switches. In this work, we use the particle filter to guide the search for a detection for each trajectory. For each trajectory T_j, we search for a detection at location $\boldsymbol{x}_{j,i}^{(k^*)}$ within some distance C, where $\boldsymbol{x}_{j,i}^{(k^*)}$ is the position of the most likely particle for trajectory j. We currently set C to be 75% of the width of the detection window. If found detection we consider that the location is classified as a head, the trajectory is confirmed in this frame, and associate the detection with the current track; if not found we consider the trajectory is occluded in this frame. We use confirmation and occlusion information to reduce tracking errors. Details are given in the next section.

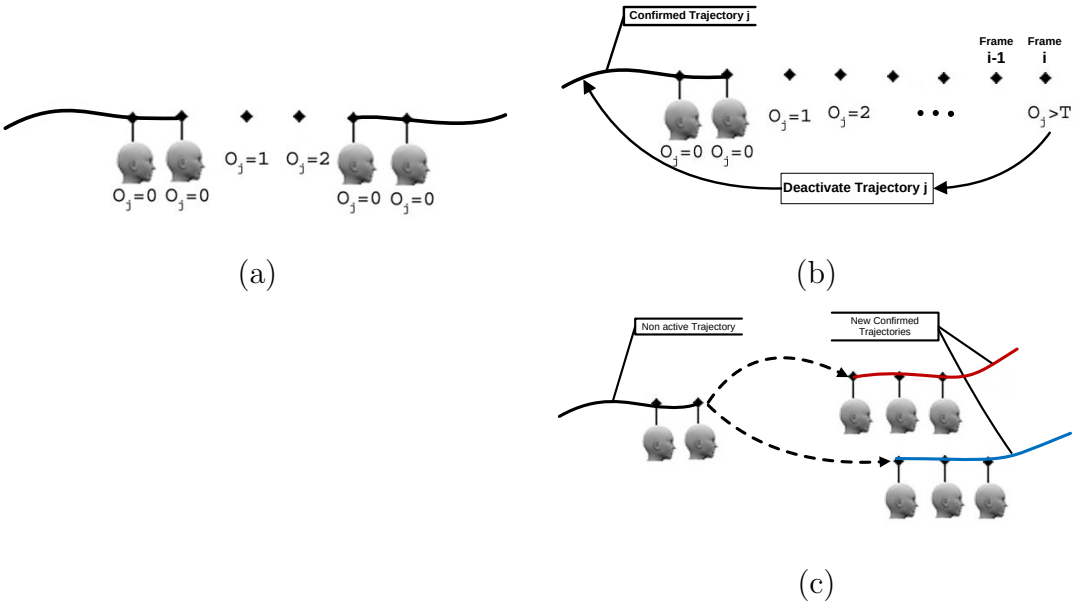


Figure 4.2: Occlusion count scheme. (a) Short occlusion. The occlusion count does not reach the deactivation threshold, so the head is successfully tracked through the occlusion. (b) Long occlusion. The occlusion count reaches the deactivation threshold, so the track is deactivated. (c) Possible reactivation of a deactivated trajectory. Newly confirmed trajectories are compared with deactivated trajectories, and if the appearance match is sufficiently strong, the deactivated track is reactivated from the position of the matching new trajectory.

4.4.3 Occlusion Count

Occlusion handling and inconsistent false track rejections are the main challenges for any tracking algorithm. To handle these problems, we introduce a simple occlusion count scheme. When head j is first detected and its trajectory is initialized, we set the occlusion count $O_j = 0$. After updating the head's position in frame i, we confirm the estimated position through detection as described in the previous section. On each frame, the occlusion count of each trajectories not confirmed through classification is incremented, and the occlusion count of each confirmed trajectory is reset to 0. An example of the increment and reset process is shown in Figure 4.2(a). The details of the algorithm are as follows:

1. **Eliminate false tracks**: shadows and other non-head objects in the scene tend to produce transient false detections that could lead to tracking errors. In order to prevent these false detections from being tracked by the appearance-based tracker through time, we use the head detector to confirm the estimated head position for each trajectory and eliminate any new trajectory not confirmed for some number of frames. A trajectory is considered transient until it is confirmed in several frames.

2. **Short Occlusions**: to handle short occlusions during tracking, we keep track of the occlusion count for each trajectory. If the head is confirmed before the occlusion count reaches the deactivation threshold, we consider the head successfully tracked through the occlusion. An example is shown in Figure 4.2(a).

3. **Long Occlusions**: when an occlusion in a crowded scene is long, it is often impossible

to recover, due to appearance changes and uncertainty in tracking. However, if the object's appearance when it reappears is sufficiently similar to its appearance before the occlusion, it can be restored. We use the occlusion count and number of confirmations to handle long occlusions through deactivation and reactivation. When an occlusion count reaches the deactivation threshold, we deactivate the trajectory. An example is shown in Figure 4.2(b). Subsequently, when a new trajectory is confirmed by detections in several consecutive frames, we consider it a candidate continuation of existing deactivated trajectories. An example is shown in Figure 4.2(c). Whenever a newly confirmed trajectory matches a deactivated trajectory sufficiently strongly, the deactivated trajectory is reactivated from the position of the new trajectory.

Chapter 5

Head Plane Estimation

In this chapter I introduce a method for estimation and utilization of a *head plane* parallel to the ground plane at the expected human height that is extracted automatically from observations from a single, uncalibrated camera. The head plane is estimated incrementally, and when the confidence in the estimate is sufficiently high, we use it to reject likely false detections produced by the head detector.

5.1 Introduction

In object detection, we normally have a tradeoff between detection rates and false positive rates: if we try to increase the detection rate, in most cases we will also increase the false positive rate. However, we can avoid this dilemma when scene constraints are available; detections inconsistent with the constraints can be rejected without affecting the true detection rate. One such constraint is 3D scene information. Several research groups have proposed the use of 3D information for segmentation, for occlusion reasoning, and to recover 3D trajectories. A detailed review is given in Chapter 2.

In this work, the main focus is to reduce the false positives produced by the object detector in order to improve the accuracy of the tracking algorithm. This algorithm does not rely on any object detection and tracking algorithm, and it can be used with any algorithm which is suitable for high density crowds. For pedestrian detection and tracking there are several possible options but we use the method proposed in Chapter 3 and Chapter 4.

In an experimental evaluation, we find that at detection time, using the 3D head plane information improves the accuracy of pedestrian tracking in dense crowds and reduces false positive rates while preserving high detection rates.

5.2 Summary

Our approach to head plane estimation is based on a few assumptions. We assume a pinhole camera with known focal length and that all human heads are approximately the same size in the real world. We further assume that the heads visible in a crowded scene will lie close to a plane that is parallel to the ground at the average height of the humans in the scene. Based on these assumptions, we can compute the approximate 3D position of a detected head using the size of the detection window then estimate the plane best fitting the data. Once sufficient confidence in the plane is obtained, we can then reject detections corresponding to 3D positions too far from the head plane. See Figure 5.1 for a schematic.

We use a robust incremental estimation method. First, we detect heads in the first frame and obtain a robust estimate of the best head plane using RANSAC. Second, we refine the estimate by minimizing the squared difference between measured and predicted head heights

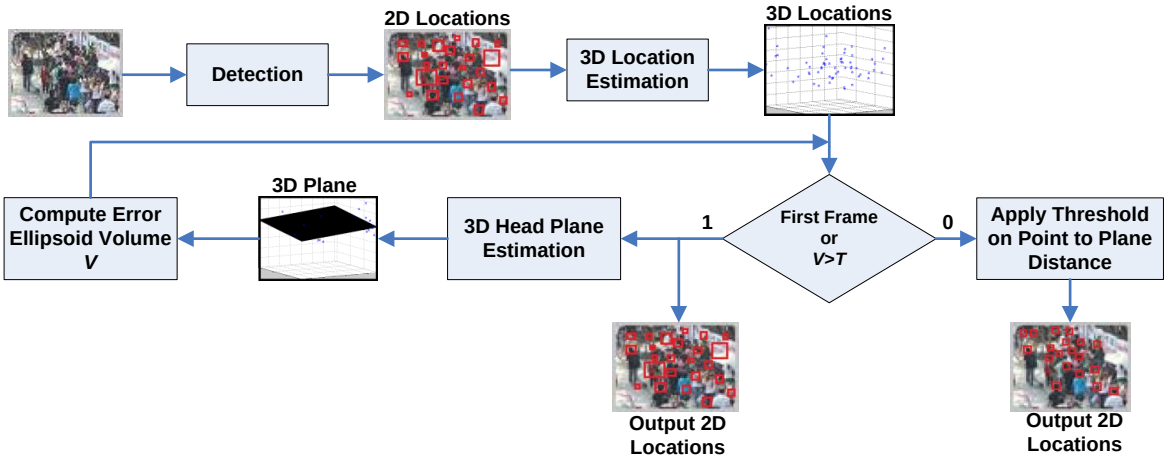


Figure 5.1: Flow of the incremental head plane estimation algorithm.

using the Levenberg-Marquardt nonlinear least squares algorithm. Using the normalized error covariance matrix, we compute the volume of the error ellipsoid, which indicates the uncertainty in the estimated plane's parameters. On subsequent frames, we add any new detections and reestimate the plane until the volume of the error ellipsoid is below a threshold. We determine the threshold experimentally. Details of the method are given below.

5.3 3D Head Position Estimation from a 2D Detection

Given the approximate actual height h_o of human heads, a 2D head detection at image position $\boldsymbol{x}_i = (x_i, y_i)$ with height h_i, and the camera focal length f, we can compute the approximate 3D location $\boldsymbol{X}_i = (X_i, Y_i, Z_i)$ of the candidate head in the camera coordinate system as follows.

$$Z_i = \frac{h_o}{h_i} f \qquad \text{(Equation 5.1)}$$

$$X_i = \frac{Z_i}{f} x_i = \frac{h_0}{h_i} x_i \qquad \text{(Equation 5.2)}$$

$$Y_i = \frac{Z_i}{f} y_i = \frac{h_0}{h_i} y_i \qquad \text{(Equation 5.3)}$$

5.4 Linear Head Plane Estimation

After getting a set of 3D locations $X = \{\boldsymbol{X}_i\}$, $i \in 1 \ldots n$ of possible heads, we estimate the plane $\pi = (a, b, c, d)$ where

$$aX + bY + cZ + d = 0$$

using linear least squares as follows. If $a^2 + b^2 + c^2 = 1$, then r_i is the signed distance of point $\boldsymbol{X}_i = (x_i, y_i, z_i)$ to the plane:

$$r_i = ax_i + by_i + cz_i + d.$$

We wish to minimize the objective function

$$q(X;\pi) = \sum_{i=1}^{n}(ax_i + by_i + cz_i + d)^2. \qquad \text{(Equation 5.4)}$$

q is minimized when its gradient is zero:

$$\frac{\partial q}{\partial a} = \sum_{i=1}^{n}[2x_i(ax_i + by_i + cz_i + d)] = 0 \qquad \text{(Equation 5.5)}$$

$$\frac{\partial q}{\partial b} = \sum_{i=1}^{n}[2y_i(ax_i + by_i + cz_i + d)] = 0 \qquad \text{(Equation 5.6)}$$

$$\frac{\partial q}{\partial c} = \sum_{i=1}^{n}[2z_i(ax_i + by_i + cz_i + d)] = 0 \qquad \text{(Equation 5.7)}$$

$$\frac{\partial q}{\partial d} = \sum_{i=1}^{n}[2(ax_i + by_i + cz_i + d)] = 0 \qquad \text{(Equation 5.8)}$$

we can write Equation 5.8 as $-d = ax_0 + by_0 + cz_0$ where

$$x_0 = \frac{\sum_{i=1}^{n} x_i}{n},\ y_0 = \frac{\sum_{i=1}^{n} y_i}{n},\ \text{and}\ z_0 = \frac{\sum_{i=1}^{n} z_i}{n}.$$

This shows that the best plane passes through the center of mass. Subtracting the center of mass from each point and substituting into Equations Equation 5.5–Equation 5.8, we get

$$\mathtt{W}\boldsymbol{x} = 0 \qquad \text{(Equation 5.9)}$$

where

$$\mathtt{W} = \begin{bmatrix} \sum_{i=1}^{n} x_i'^2 & \sum_{i=1}^{n} x_i'y_i' & \sum_{i=1}^{n} x_i'z_i' \\ \sum_{i=1}^{n} x_i'y_i' & \sum_{i=1}^{n} y_i'^2 & \sum_{i=1}^{n} y_i'z_i' \\ \sum_{i=1}^{n} x_i'z_i' & \sum_{i=1}^{n} y_i'z_i' & \sum_{i=1}^{n} z_i'^2 \end{bmatrix},$$

$x_i' = x_i - x_0$, $y_i' = y_i - y_0$, $z_i' = z_i - z_0$, and

$$\boldsymbol{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$

To avoid the trivial solution $\boldsymbol{x} = \mathbf{0}$, we impose the constraint $\|\boldsymbol{x}\| = 1$. The least squares solution is then the eigenvector $\boldsymbol{x}$ of $\mathtt{W}$ corresponding to the smallest eigenvalue.

Since the set of 3D locations X will in general contain outliers due to errors in head height estimates and false detections from the detector, before performing the above minimization, we eliminate outliers using RANSAC (Fischler & Bolles, 1981). On each iteration of RANSAC, we sample three points from X, compute the corresponding plane, find the consensus set for that plane, and retain the largest consensus set. The number of iterations is calculated adaptively based on the size of the largest consensus set. We set the number of iterations k to the minimum number of iterations required to guarantee, with a small probability of failure, that the model with the largest consensus set has been found. The probability of

selecting three inliers from X at least once is $p = 1 - (1 - w^3)^k$, where w is the probability of selecting an inlier in a single sample. We initialize k to infinity, then on each iteration, we recalculate w using the size of the largest consensus set found so far and then find the number of iterations k needed to achieve a success rate of p.

5.5 Nonlinear Head Plane Refinement

The linear estimate of the head plane computed in the previous section minimizes an algebraic objective function (Equation 5.4) that does not take into account the fact that head detections close to the camera are more accurately localized in 3D than head detections far away from the camera.

In this step, we refine the linear estimate of the head plane to minimize the objective function

$$q^*(\boldsymbol{\pi}) = \sum_{i=1}^{n} (h_i - \hat{h}_i)^2, \qquad \text{(Equation 5.10)}$$

where h_i is the height of the detection window for head i and $\hat{h}_i$ is the predicted height of head i based on the 2D location (x_i, y_i) of its detection and the plane $\boldsymbol{\pi}$. To calculate $\hat{h}_i$, we find the ray through the camera center $C = (0, 0, 0)$ passing through (x_i, y_i), find the intersection $\hat{\boldsymbol{X}}_i = \begin{bmatrix} \hat{X}_i & \hat{Y}_i & \hat{Z}_i \end{bmatrix}^T$ of that ray with the head plane $\boldsymbol{\pi}$, then calculate the expected height of an object with height h_0 at $\hat{\boldsymbol{X}}_i$ when projected into the image.

To find the intersection of the ray with the plane, given the camera matrix

$$\mathtt{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

containing the focal length f and principal point (c_x, c_y), we find an arbitrary point

$$\boldsymbol{X}'_i = \mathtt{K}^{-1} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

on the ray then find the scalar α such that

$$\begin{bmatrix} \alpha \boldsymbol{X}'_i \\ 1 \end{bmatrix}^T \boldsymbol{\pi} = 0.$$

Finally, we calculate $\hat{\boldsymbol{X}}_i = \alpha \boldsymbol{X}'_i$ and $\hat{h}_i = \frac{h_0}{\hat{Z}_i} f$.

We use the Lourakis implementation of the Levenberg-Marquardt nonlinear least squares algorithm (Lourakis, Jul. 2004) to find the plane $\boldsymbol{\pi}$ minimizing the objective function of Equation Equation 5.10 and obtain an error covariance matrix $\mathtt{Q}$ for the elements of $\boldsymbol{\pi}$.

5.5.1 Incremental Head Plane Estimation

Based on the parameter vector $\boldsymbol{\pi}$ and covariance matrix $\mathtt{Q}$ obtained as detailed in the previous section, we compute the volume of the error ellipsoid to indicate uncertainty in the estimated plane's parameters. Since the uncertainty in the plane's orientation only depends weakly on the distance of the camera to the head plane, whereas the uncertainty in the plane's distance to the camera depends strongly on that distance, we only consider the uncertainty in the plane's (normalized) orientation, ignoring the uncertainty in the distance to the plane. Let λ_1, λ_2, and λ_3 be the eigenvalues of the upper 3×3 submatrix of $\mathtt{Q}$ representing the uncertainty in the plane orientation parameters a, b, and c. The radii of the normalized plane orientation error ellipsoid are $r_i = \sqrt{\lambda_i/(a^2 + b^2 + c^2)}$ for $i \in 1, 2, 3$. The volume of the plane orientation error ellipsoid is then

$$V = \frac{4}{3}\pi r_1 r_2 r_3. \qquad \text{(Equation 5.11)}$$

V quantifies the uncertainty of the estimated head plane's orientation. During tracking, for each frame, we add any newly detected heads, reestimate the head plane, and recalculate V. If it is less than threshold, we stop the process and use the head plane to filter subsequent detections. We determine the threshold empirically.

Chapter 6

Experimental Evaluation

In this chapter, I provide details of a series of experiments to evaluate our algorithm. First I describe the training and test data sets. Second, I describe some of important implementation details. Third, I describe the evaluation metrics we use. Finally, I provide results and discussion for the detection and tracking evaluations.

6.1 Training Data

To train the Viola and Jones Haar-like AdaBoost cascade detector, we cropped 5396 heads (examples are shown in Figure 6.1) from videos collected at various locations and scaled each to 16×16 pixels. We also collected 4187 negative images not containing human heads. The detailed training parameters are listed in Table 6.1.

For the CAVIAR experiments described in the next section, since many of the heads appearing in the CAVIAR test set are very small, we created a second training set by scaling the same 5396 positive examples to a size of 10×10.

We use the OpenCV `haartraining` utility to train the classifier.

6.2 Test Data

There is no generally accepted dataset available for crowd tracking. Most researchers use their own datasets to evaluate their algorithms. In this work, for the experimental evaluation, we have created, to the best of our knowledge, the most challenging existing dataset specifically for tracking people in high density crowds; the dataset with ground truth information is publicly available for download at `http://www.cs.ait.ac.th/vgl/irshad/`.

We captured the video at 640×480 pixels and 30 frames per second at the Mochit light rail station in Bangkok, Thailand. A sample frame is shown in Figure 6.2. We then hand labeled the locations of all heads present in every frame of the video. For the ground truth data format and annotation, we followed the guidelines given for the VACE-II (Video Analysis and Content Extraction) workshop (Raju & Prasad, 2006). This means that for each head present in each frame, we record the bounding box, a unique ID that is consistent across frames, and a flag indicating whether the head is occluded or not. We labeled a total of 700 frames containing a total of 28,430 heads, for an average of 40.6 heads/frame.

To compare with existing state of the art research, we test our algorithm on the well-known indoor dataset Context Aware Vision using Image based Active Recognition (CAVIAR) (*The CAVIAR Data Set*, 2011). Since our evaluation requires ground truth positions of each pedestrian's head in each frame, we selected the four sequences from the "shopping center corridor view" for which the needed information is available. The sequences contain a total

(a) (b)

Figure 6.1: Positive head samples used to train the head detector offline. (a) Example images collected for training. (b) Example scaled images.

Table 6.1: Head detector training parameters.

Parameters	Values	Description
npos	5396	Number of positive samples
nneg	4187	Number of negative samples
nstages	20	Number of training stages
minhitrate	0.995	Minimum hit rate per stage (99.5%)
maxfalsealarm	0.5	Maximum false alarm rate per stage (50%)
mode	All	Use the full set of both upright and 45 degree rotated features
width, height	16	Training image patch width and height
boosttypes	DAB	Discrete AdaBoost.

Figure 6.2: Sample frame acquired at the Mochit light rail station in Bangkok, Thailand.

Table 6.2: Crowd density comparison results.

Mochit	CAVIAR (*The CAVIAR Data Set*, 2011)	Campus Plaza (Zhao et al., 2008)
0.63	0.27	0.23

of 6,315 frames, the frame size is 384 × 288, and the sequences were captured at 25 frames per second. There are a total of 26,950 heads over all four sequences, with an average of 4.27 heads per frame.

Our algorithm is designed to track heads in high density crowds. The performance of any tracking algorithm will depend upon the density of the crowd. In order to characterize this relationship we introduce a simple crowd density measure

$$D = \frac{\sum_i P_i}{N},$$

where P_i is the number of pixels in pedestrian i's bounding box and N is the total number of pixels in all of the images.

According to this measure, the highest per-frame crowd density in our Mochit test sequence is 0.63, whereas the highest per-frame crowd density in CAVIAR is 0.27. The crowd density in the Mochit sequence is higher than that in any publicly-available pedestrian tracking video database. The results of the comparison are shown in Table 6.2.

To directly evaluate the accuracy of our head plane estimation method, we use the People Tracking sequence (S2.L1) of Performance Evaluation of Tracking and Surveillance (PETS) (*PETS Benchmark Data*, 2009) dataset. In PETS, camera calibration information is given for every camera. In this sequence there are a total of eight views. In views 1, 2, 3, and 4,

the head sizes are very small; we excluded these views because our method has a limit on the minimum head size that can be detected. We trained a new head detector specifically on the PETS training data (which is separate from the people tracking test sequence), ran the tracking and head plane estimation algorithm on views 5, 6, 7, and 8, then compared our estimated head plane with the actual ground plane information that comes with the dataset.

6.3 Implementation Details

We implemented the system in C++ with OpenCV without any special code optimization. The system attempts to track anything head-like in the scene, whether moving or not, since it does not rely on any background modeling. We detect heads and create initial trajectories based on the first frame, and then we track heads from frame to frame. Further implementation details are given in the following sections.

6.3.1 Trajectory Initialization and Termination

We use the head detector to find heads in the first frame and create initial trajectories. As previously mentioned, rather than detect heads only in the border region of the image, we detect all heads in every frame. We first try to associate new heads with existing trajectories; when this fails for a new head detection, a new trajectory is initialized from the current frame. Any head trajectory in the "exit zone" (close to the image border) for which the motion model predicts a location outside the frame is eliminated.

6.3.2 Identity Management

It is also important to assign and maintain object IDs automatically during tracking. We assign a unique ID to each trajectory during initialization then maintain the ID during tracking. Trajectories that are temporarily lost due to occlusion are reassigned the same ID on recovery to avoid identity changes. During long occlusions, when a track is not confirmed for several frames, we deactivate that track and search for new matching detections in subsequent frames. If a match is found, the track is reactivated from the position of the new detection and reassigned the same ID.

6.3.3 Incremental Head Plane Estimation

As previously discussed, we incrementally estimate the head plane. During tracking, we collect detected heads cumulatively over each frame and perform head plane estimation. As discussed in Section 5.5.1, to determine when to start using the estimated plane to filter detections, we use the volume of the normalized plane orientation error ellipsoid (see Equation Equation 5.11) as a measure of the uncertainty in the estimate. Figure 6.3 shows how the error ellipsoid volume evolves over time on the Mochit data set as heads detected in subsequent

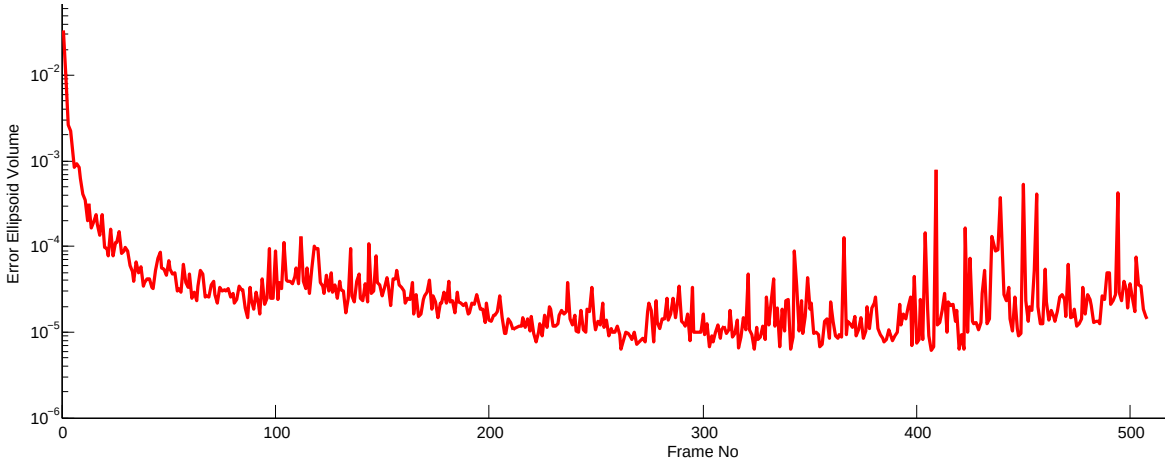


Figure 6.3: Error in head plane estimation. We plot the volume of the normalized plane orientation error ellipsoid, computed from the plane parameter estimate's covariance matrix after nonlinear optimization. The graph shows the volume of the ellipsoid after adding newly detected heads in each frame.

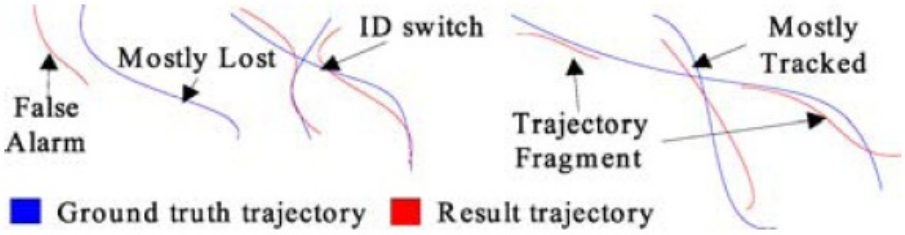


Figure 6.4: Evaluation criteria. Reprinted from Wu and Nevatia (2007).

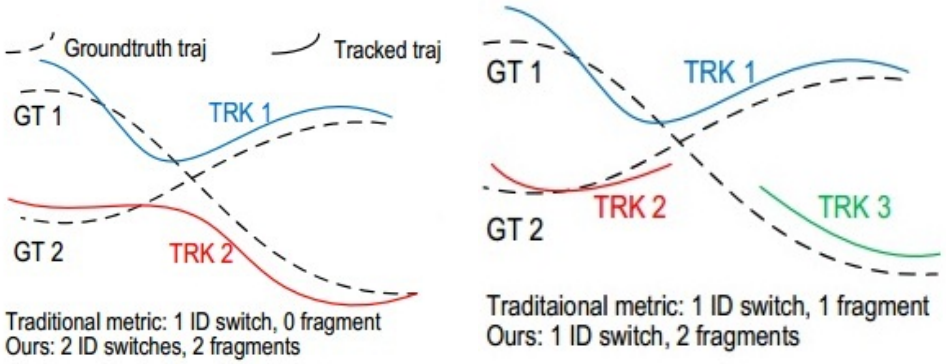
frames are added to the data set. We stop head plane estimation when the error ellipsoid volume is less than 0.0003.

As a quantitative evaluation of the head plane estimation method, we trained a new head classifier using the training set of the PETS dataset, then we ran our system on views 5–8 from the PETS People Tracking sequence (S2.L1). The estimation error was 305mm, 433mm, 355mm, and 280mm for the orthogonal distance between the plane and the camera center and 25°, 20°, 17°, and 16° for the orientation. This indicates that the method is quite effective at unsupervised estimation of the head plane.

6.4 Evaluation Metrics

In this section, we describe the methods we use to evaluate the tracking algorithm. Unfortunately there are no commonly-used metrics for human detection and tracking in crowded scenes.

We adopt measures similar to those proposed by Nevatia and colleagues (Wu & Nevatia, 2007; Kuo et al., 2010; Li et al., 2009) for tracking pedestrians in sparse scenes (see Figure 6.4). In their work, there are different definitions for ID switch and trajectory fragmentation



(a) (b)

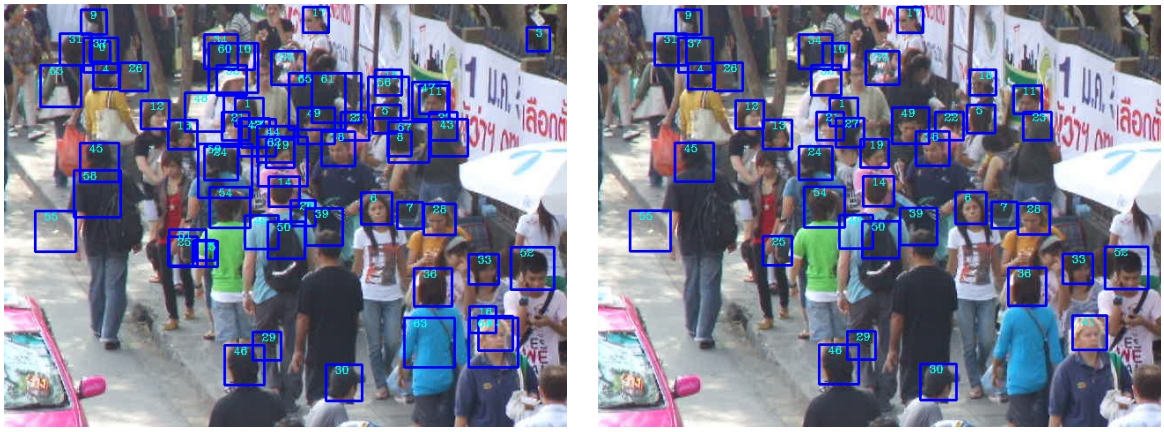
Figure 6.5: ID switch and trajectory fragmentation. (a) Scenario 1. (b)Scenario 2. Reprinted from Li et al. (2009).

errors, as shown in Figure 6.5. Wu and Nevatia (Wu & Nevatia, 2007) define ID switches as "identity exchanges between a pair of result trajectories," while Li, Huang, and Nevatia (Li et al., 2009) define an ID switch as "a tracked trajectory changing its matched GT ID." We adopt the definitions of ID switch and fragment errors proposed by Li, Huang, and Nevatia (Li et al., 2009). If a trajectory ID is changed but not exchanged, we count it as one ID switch, similarly for fragments. This definition is more strict and leads to higher numbers of ID switch and fragment errors, but it is well defined.

Bernardin and Stiefelhagen have proposed an alternative set of metrics, the CLEAR MOT metrics (Bernardin & Stiefelhagen, 2008), for multiple object tracking performance. Their MOTP (multiple-object tracking precision) and MOTA (multiple-object tracking accuracy) methods are not suitable for crowd tracking because they combine many factors into one scalar value.

Kasturi and colleagues (Kasturi et al., 2009) have also proposed a framework to evaluate object detection and tracking in video. They mainly focused on detecting and tracking faces, text, and vehicles in video. Their method are not suitable for crowd tracking because they combine many factors into one scalar value. We specifically use the following evaluation criteria:

1. **Ground truth (GT)**: number of ground truth trajectories.
2. **Mostly tracked (MT)**: number of trajectories that are successfully tracked for more than 80% of their length (tracked length divided by the ground truth track length).
3. **Partially tracked (PT)**: number of trajectories that are successfully tracked in 20%–80% of the ground truth frames.
4. **Mostly lost (ML)**: number of trajectories that are successfully tracked for less than 20% of the ground truth frames.
5. **Fragments (Frag)**: number of times that a ground truth trajectory is interrupted in the tracking results.



(a) (b)

Figure 6.6: Detection results for one frame of the Mochit test video. Rectangles indicate candidate head positions. (a) Without 3D head plane estimation. (b) With 3D head plane estimation.

Table 6.3: Detection results for single image with and without head plane estimation.

	GT	Hits	Misses	FP
Without head plane estimation	34	31	3	35
With head plane estimation	34	30	4	12

6. **ID switches (IDS)**: number of times the system-assigned ID changes over all ground truth trajectories.

7. **False trajectories (FAT)**: number of system trajectories that do not correspond to ground truth trajectories.

6.5 Detection Results

We trained our head detection cascade using the OpenCV `haartraining` utility. We set the number of training stages to 20, the minimum hit rate per stage to 0.99, and the maximum false-alarm rate per stage to 0.5. The training process required about 16 hours, for this experiment we used an Intel Pentium-4 (2.8GHz), 4GB RAM desktop PC.

Table 6.4: Overall detection results with and without head plane estimation.

	GT	Hits	Misses	FP
Without head plane estimation	24605	19277	5328	7053
With head plane estimation	24605	17950	6655	3328

Table 6.5: Tracking results with and without head plane estimation for the Mochit station dataset. Total number of trajectories is 74.

	MT%	PT%	ML%	Frag	IDS	FAT
Without head plane	67.6	28.4	4.0	43	20	41
With head plane	70.3	25.7	4.0	46	17	27

To test the system's raw head detection performance with and without head plane estimation on a single image, we ran our head detector on an arbitrary single frame extracted from the Mochit test data sequence. The results are summarized in Table 6.3 and visualized in Figure 6.6.

There are a total of 34 visible ground truth (GT) heads in the frame. Using the head plane to reject detections inconsistent with the scene geometry reduces the number of false positives (FP) from 35 to 12 and only reduces the number of detections (hits) from 31 to 30. The results show that the head plane estimation method is very useful for filtering false detections.

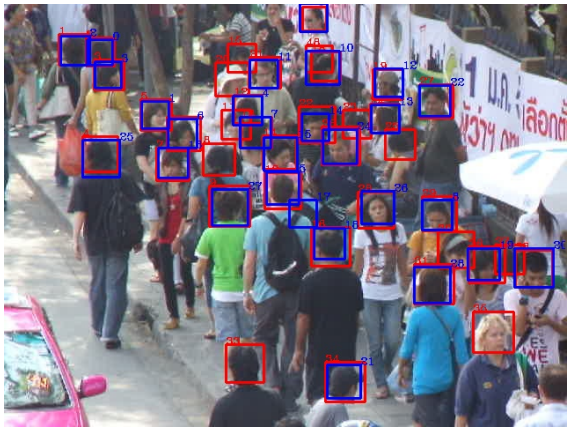
To test the system's head detection performance with and without head plane estimation on whole sequence, we ran our head detector on each frame extracted from the Mochit test data sequence and compared with the head location reported by our tracking algorithm. The results are summarized in Table 6.4.

There are a total of 24605 visible ground truth (GT) heads in the sequence of 700 frames. Our algorithm reduces the number of false positives (FP) from 7053 to 3328 and only reduces the number of detections (hits) from 19277 to 17950.

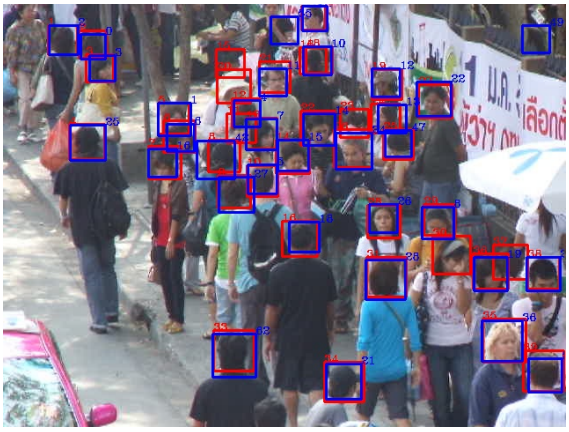
6.6 Tracking Results

In the Mochit station dataset (*Mochit station dataset*, 2009), there are an average of 40.6 individuals per frame over the 700 hand-labeled ground truth frames, for a total of 28,430 heads, and a total of 74 individual ground truth trajectories. We used 20 particles per head. Tracking results with and without head plane estimation are shown in Table 6.5. The head plane estimation method improves accuracy slightly, but more importantly, it reduces the false positive rate while preserving a high rate of successful tracking. For a frame size of 640 $\times$ 480, the processing time was approximately 1.4 seconds per frame, with or without head plane estimation, on a 3.2 GHz Intel Core i5 with 4 GB RAM. Figure 6.7 shows tracking results for several frames of the Mochit test video.

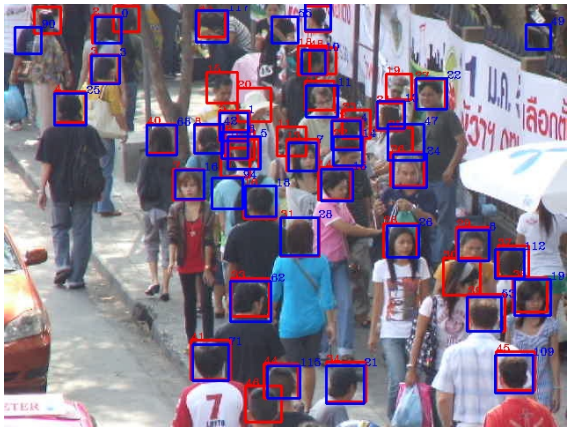
In the four selected sequences from the CAVIAR dataset for which the ground truth pedestrian head information is available, there are a total of 43 ground truth trajectories, with an average of 4.27 individuals per frame or 26,950 heads total over the 6,315 hand-labeled ground truth frames. Since our detector cannot detect heads smaller than 15×15 reliably, in the main evaluation, we exclude ground truth heads smaller than 15×15. However, to enable direct comparison to existing full-body pedestrian detection and tracking methods, we also provide results including the untracked small heads as errors. We again used 20 particles per head. Tracking results with and without small heads and with and without head plane



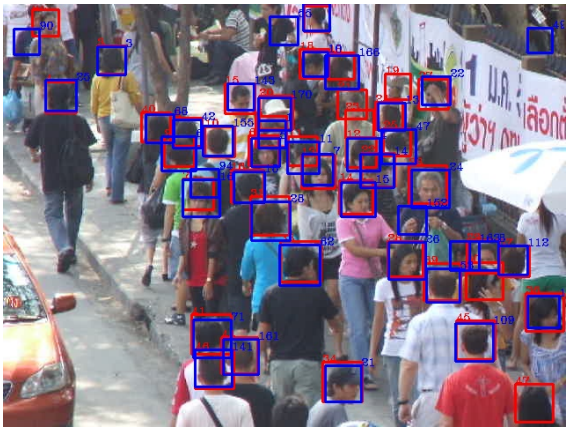
Frame 1



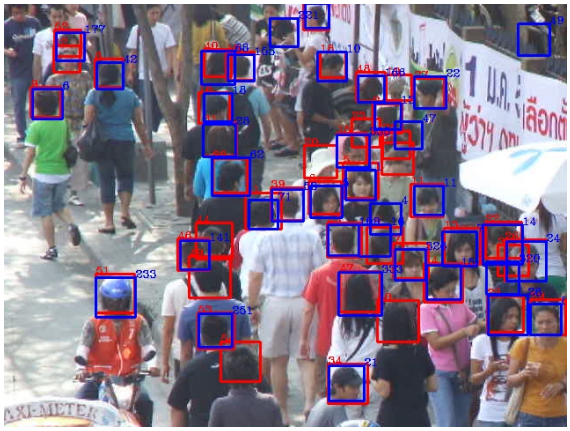
Frame 15



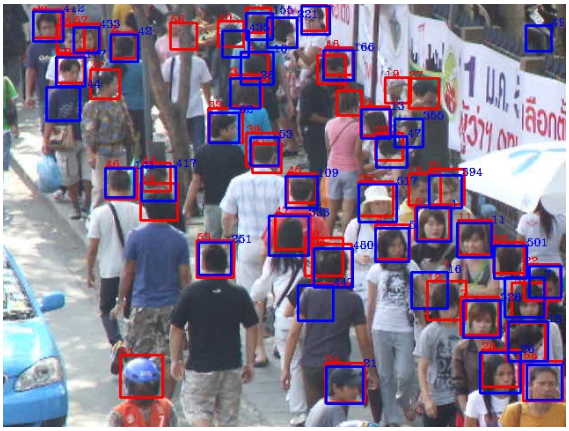
Frame 65



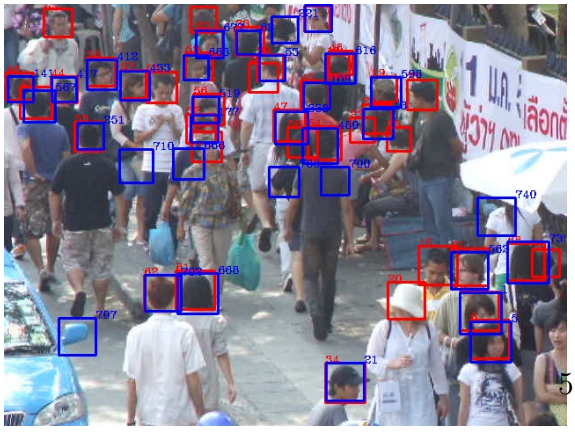
Frame 92



Frame 220



Frame 304

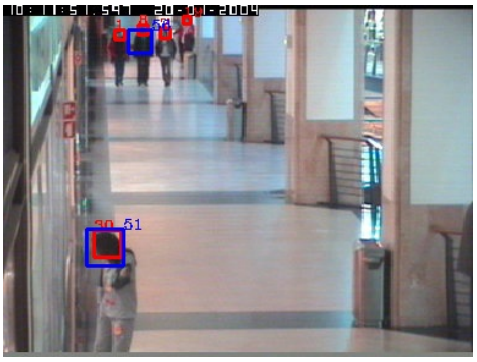


Frame 470

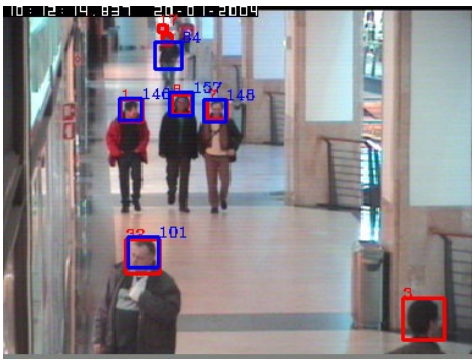
Frame 605

Figure 6.7: Sample tracking results for the Mochit test video. Blue rectangles indicate

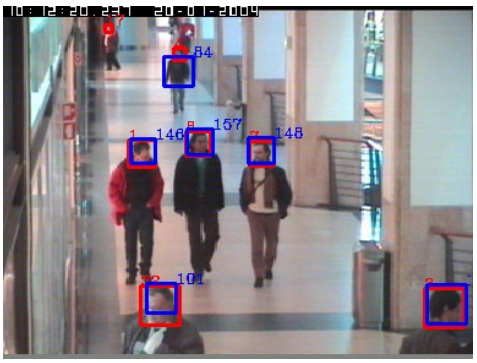
Frame 206



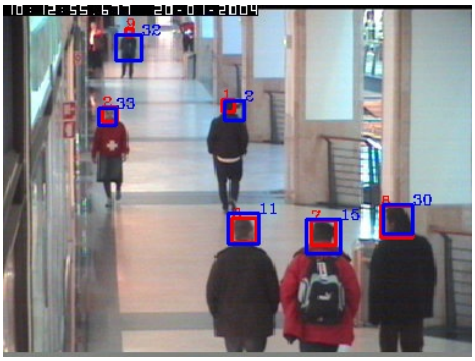
Frame 390



Frame 971

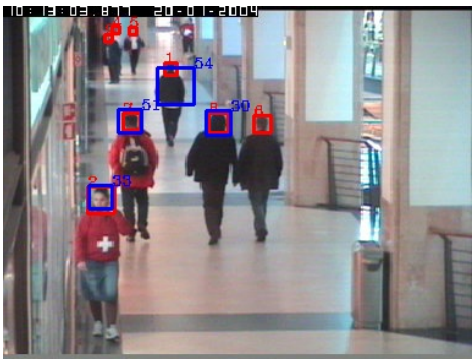


Frame 1106



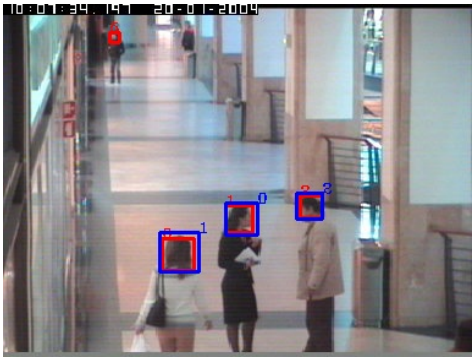
Frame 338

Frame 476

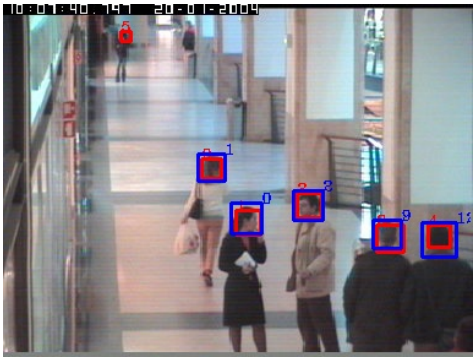


Frame 543

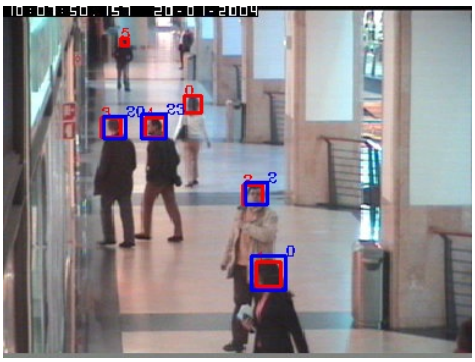
Frame 1471



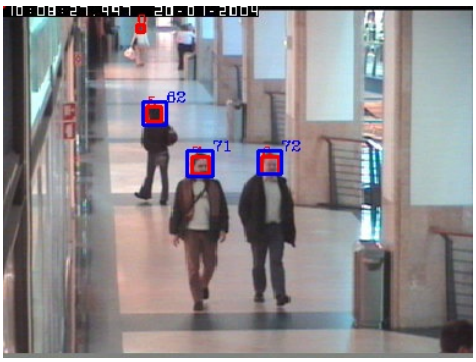
Frame 5



Frame 170



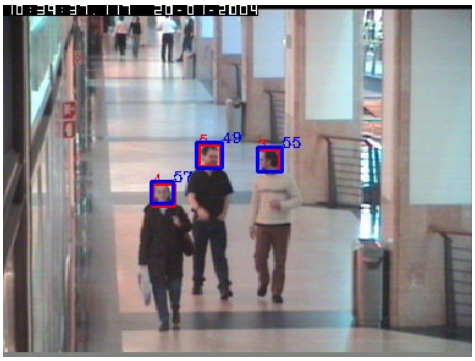
Frame 404



Frame 640

Frame 100

Frame 310



Frame 1475

Frame 1580

Figure 6.7: Few tracking example frames from CAVIAR dataset. Blue rectangles indicate estimated head positions; red rectangles indicate ground truth head positions.

Table 6.6: Tracking results for CAVIAR dataset.

	GT	MT%	PT%	ML%	Frag	IDS	FAT
Zhao, Nevatia, and Wu (Zhao et al., 2008)	227	62.1	-	5.3	89[1]	22[1]	27
Wu and Nevatia (Wu & Nevatia, 2007)	189	74.1	-	4.2	40[1]	19[1]	4
Xing, Ai and Lao (Xing, Ai, & Lao, 2009)[2]	140	84.3	12.1	3.6	24	14	-
Li, Huang and Nevatia (Li et al., 2009)	143	84.6	14.0	1.4	17	11	-
Ali and Dailey[3]	33	75.8	24.2	0.0	10	1	21
Ali and Dailey (without head plane)	33	75.8	24.2	0.0	14	2	34
Ali and Dailey (all heads)	43	16.3	58.1	25.6	11	1	21

[1] The Frag and IDS definitions are less strict than ours, giving lower numbers of fragments and ID switches.

[2] Does not count people less than 24 pixels wide.

[3] Does not count heads less than 15 pixels wide.

estimation are shown in Table 4. For a frame size of 384 × 288, the processing time was approximately 0.350 seconds per frame on the same 3.2 GHz Intel Core i5 with 4 GB RAM. Figure 6.7 shows tracking results for several frames of the CAVIAR test set.

Our head tracking algorithm is designed especially for high density crowds. We do not expect it to work as well as full-body tracking algorithms on sparse data where the full body is in most cases visible. Although it is difficult to draw any strong conclusion from the data in Table 4, since none of the reported work is using precisely the same subset of the CAVIAR data, we tentatively conclude that the method proposed in this dissertation gives comparable performance to the state of the art, even though it is using less information.

Although the researchers whose work is summarized in Table 4 have not made their code public to enable direct comparison on the Mochit test set, we would expect that any method relying on full body or body part based tracking would perform much more poorly than our method on that data set.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

Tracking people in high-density crowds such as the one shown in Figure 1.1 is a real challenge and is still an open problem. In this paper, we introduce a fully automatic algorithm to detect and track multiple humans 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 tracks. We find that confirmation by classification dramatically reduces tracking errors such as ID switches and fragments.

The main difficulty in using a generic object detector for human tracking is that the detector's output is unreliable; all detectors make errors. To further reduce false detections due to dense features and shadows, we present an algorithm using an estimate of the 3D head plane to reduce false positive head detections and improve pedestrian tracking accuracy in crowds. The method is straightforward, makes reasonable assumptions, and does not require any knowledge of camera extrinsics. Based on the projective geometry of the pinhole camera and an assumed approximate head size, we compute 3D locations of candidate head detections. We then fit a plane to the set of detections and reject detections inconsistent with the estimated scene geometry. The algorithm learns the head plane from observations of human heads incrementally, and only begins to utilize the head plane once confidence in the parameter estimates is sufficiently high.

We find that together, the confirmation-by-classification and head plane estimation methods enable the construction of an excellent pedestrian tracker for dense crowds. In future work, with further algorithmic improvements and runtime optimization, we hope to achieve robust, real time pedestrian tracking for even larger crowds.

References

Adam, A., Rivlin, E., Shimshoni, I., & Reinitz, D. (2008). Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *30*, 555–560.

Analyticvideo. (2009). *Analytic video systems.* (available at `http://www.analyticvideo.com/index.php?area=Technology`)

Andriluka, M., Roth, S., & Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–8).

Bajracharya, P. (2007). *Haartraining for pedestrian detection.* (Web tutorial available at `https://webeng.cs.ait.ac.th/cvwiki/opencv:tutorial:haartraining`)

Berclaz, J., Fleuret, F., & Fua, P. (2006). Robust people tracking with global trajectory optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Bernardin, K., & Stiefelhagen, R. (2008). Evaluatingmultiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, *2008*, 1-10.

Bravevideo. (2009). *Intelligent video surveillance system.* (available at `http://www.bravevideo.com/en/productsmoreTwo.asp?D_cataid=A0022`)

Breitenstein, M., Reichlin, F., Leibe, B., Koller-Meier, E., & Gool, L. V. (2009). Robust tracking-by-detection using a detector confidence particle filter. In *IEEE International Conference on Computer Vision (ICCV)* (p. 1514-1522).

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Gool, L. V. (2011). Online multi-person tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *33*(9), 1820-1833.

Cai, Y., Freitas, N. de, & Little, J. J. (2006). Robust visual tracking for multiple targets. In *European Conference on Computer Vision (ECCV)* (Vol. 3954, p.

107-118).

The CAVIAR data set. (2011). (available at `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`)

Chan, A. B., Liang, Z.-S. J., & Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–7).

Cootes, T. F., & Taylor, C. J. (2004). *Statistical models of appearance for computer vision.* Technical report, University of Manchester.

Cootes, T. F., Taylor, C. J., Cooper, D. H., & Graham, J. (1995). Active shape models their training and application. *Computer Vision and Image Understanding*, *61*, 38-59.

Cupillard, F., Avanzi, A., Bremond, F., & Thonnat, M. (2004). Video understanding for metro surveillance. In *IEEE International Conference on Networking, Sensing and Control* (Vol. 1, pp. 186–191).

Dalal, N. (2006). *Finding people in images and videos.* PhD thesis, Institut National Polytechnique de Grenoble.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Doucet, A., Freitas, N. de, & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. New York: Springer.

Fengjun Lv, M., Zhao, T., & Nevatia, R. (2006). Camera calibration from video of a walking human. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *28*(9), 1513-1518.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fleuret, F., Berclaz, J., Lengagne, R., & Fua, P. (2008). Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *30*, 267–282.

Ge, W., Collins, R. T., & Ruback, R. B. (2011). Vision-based analysis of small groups in pedestrian crowds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *34*(5), 1003–1016.

Hoiem, D., Efros, A., & Hebert, M. (2005). Geometric context from a single image. In *IEEE International Conference on Computer Vision (ICCV).*

Hoiem, D., Efros, A., & Hebert, M. (2006). Putting objects into perspective. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2137–2144).

Hoiem, D., Efros, A., & Hebert, M. (2007). Recovering surface layout from an image. *International Journal of Computer Vision (IJCV)*, *75*(1).

Hoiem, D., Efros, A., & Hebert, M. (2008). Putting objects into perspective. *International Journal of Computer Vision (IJCV)*, *80*(1), 3-15.

Huang, C., Wu, B., & Nevatia, R. (2008). Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision (ECCV).*

IoImage. (2009). *Video surveillance software.* (available at `http://www.ioimage.com/?p=Demos&ClusterID=489&MainCategory=444&TreeParentID=862&ParentID=642`)

Isard, M., & Blake, A. (1998a). CONDENSATION-conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, *29*, 5-28.

Isard, M., & Blake, A. (1998b). A mixed-state condensation tracker with automatic model-switching. In *IEEE International Conference on Computer Vision (ICCV)* (pp. 107–112).

Kang, H.-G., & Kim, D. (2005). Real-time multiple people tracking using competitive condensation. *Pattern Recognition*, *38*, 1045–1058.

Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., et al. (2009). Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *31*(2), 319–336.

Khan, S. M., & Shah, M.(2006). A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision (ECCV).*

Khan, Z., Balch, T., & Dellaert, F.(2005). MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *27*, 1805–1918.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*, 83-87.

Kuo, C.-H., Huang, C., & Nevatia, R.(2010). Multi-target tracking by on-line learned discriminative appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 685–692).

Lee, J. Y. S., Goh, P. K., & Lam, W. H. K.(2005). New level-of-service standard for signalized crosswalks with bi-directional pedestrian flows. *Journal of Transportation Engineering*, *131*, 957–960.

Leibe, B., Leonardis, A., & Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *In ECCV Workshop on Statistical Learning in Computer Vision.* (p. 17-32).

Leibe, B., Leonardis, A., & Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision (IJCV)*, *77*, 259-289.

Leibe, B., Schindler, K., & Gool, L. V.(2007). Coupled detection and trajectory estimation for multi-object tracking. In *IEEE International Conference on Computer Vision (ICCV)* (pp. 1–8).

Leibe, B., Seemann, E., & Schiele, B.(2005). Pedestrian detection in crowded scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 1, pp. 878–885).

Li, Y., Huang, C., & Nevatia, R.(2009). Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 2953-2960).

Lourakis, M. (Jul. 2004). *levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++.* available at `http://www.ics.forth.gr/~lourakis/levmar/`.

Ma, Y., Yu, Q., & Cohen, I. (2009). Target tracking with incomplete detection. *Computer Vision and Image Understanding*, *113*, 580–587.

Martnez, S. V., Knebel, J., & Thiran, J. (2004). Multi-object tracking using the particle filter algorithm on the top-view plan. In *European Signal Processing Conference (EUSIPCO).*

Mathes, T., & Piater, J. H. (2005). Robust non-rigid object tracking using point distribution models. In *British Machine Vision Conference, Oxford* (Vol. 2).

Mittal, A., & Davis, L. S. (2003). M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision (IJCV)*, *51*, 189-203.

Mitzel, D., Horbert, E., Ess, A., & Leibe, B. (2010). Multi-person tracking with sparse detection and continuous segmentation. In *European Conference on Computer Vision (ECCV).*

Mochit station dataset. (2009). (available at `http://www.cs.ait.ac.th/vgl/irshad/`)

Nanda, H., & Fujimura, K. (2004). A robust elliptical head tracker. In *IEEE International Conference on Automatic Face and Gesture Recognition (FGR)* (pp. 469–474).

NEC. (2009). *Intelligent video surveillance.* (available at `http://www.nec.com.sg/?q=solutions-and-services/intelligent-video-surveillance`)

Okuma, K., Taleghani, A., Freitas, N. D., Little, J. J., & Lowe, D. G. (2004). A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision (ECCV).*

PETS benchmark data. (2009). (available at `http://www.cvg.rdg.ac.uk/PETS2009/a.html`)

Rabaud, V., & Belongie, S. (2006). Counting crowded moving objects. In *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 1, pp. 705–711).

Raju, H., & Prasad, S. (2006). *Annotation guidelines for video analysis and content extraction (VACE-II).* (available at `http://isl.ira.uka.de/clear07/downloads/`)

Ramanan, D., Forsyth, D. A., & Zisserman, A.(2007). Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *29*(1), 65–81.

Rasmussen, C., & Hager, G. D. (2001). Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *23*(6), 560-576.

Reid, D. B.(1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, *24*(6), 843-854.

Rodriguez, M., Laptev, I., Sivic, J., & Audibert, J.-Y. (2011). Density-aware person detection and tracking in crowds. In *Ieee international conference on computer vision (iccv).*

Rosales, R., & Sclaroff, S.(1999). 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Rui, Y., & Chen, Y. (2001). Better proposal distributions: object tracking using unscented particle filter. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 2, pp. II786–II793).

Schindler, K., & Suter, D.(2008). Object detection by global contour shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *41*(12), 3736–3748.

Seo, N.(2007). *OpenCV haartraining (rapid object detection with a cascade of boosted classifiers based on Haar-like features).* (Web tutorial available at `http://note.sonots.com/SciSoftware/haartraining.html`)

Sim, C.-H., Rajmadhan, E., & Ranganath, S.(2008). Using color bin images for crowd

detection. In *IEEE International Conference on Image Processing (ICIP).*

Smith, K., Gatica-Perez, D., & Odobez, J.-M. (2005). Using particles to track varying numbers of interacting people. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 1, pp. 962–969).

StratechSystems. (2009). *Intelligent video surveillance and analysis system.* (available at `http://www.stratechsystems.com/iv_supertrack.asp`)

Tang, C.-Y., Hung, Y.-P., & Chen, Z. (1997). Automatic detection and tracking of human heads using an active stereo vision system. In *Asian Conference on Computer Vision (ACCV)* (Vol. LNCS 1351, pp. 632–639).

Vermaak, J., Doucet, A., & Perez, P. (2003). Maintaining multi-modality through mixture tracking. In *IEEE International Conference on Computer Vision (ICCV).*

Viola, P., & Jones, M. (2001a). Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 511–518).

Viola, P., & Jones, M. (2001b). Robust real time object detection. *International Journal of Computer Vision (IJCV)*, *57*, 137–154.

Wu, B., & Nevatia, R. (2006). Tracking of multiple, partially occluded humans based on static body part detection. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR)* (Vol. 1, pp. 951–958).

Wu, B., & Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision (IJCV)*, *75*(2), 247-266.

Wu, B., Nevatia, R., & Li, Y. (2008). Segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR).*

Xing, J., Ai, H., & Lao, S. (2009). Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1200–1207).

Yang, C., Duraiswami, R., & Davis, L. (2005). Fast multiple object tracking via a hierarchical particle filter. In *IEEE International Conference on Computer Vision (ICCV)* (Vol. 1, pp. 212–219).

Zhan, B., Monekosso, D. N., Remagnino, P., Velastin, S. A., & Xu, L.-Q.(2008). Crowd analysis: A survey. *Machine Vision and Applications*, *19*(5–6), 345–357.

Zhang, Z., Huang, K., & Tan, T. (2009). Rapid and robust human detection and tracking based on omega-shape features. In *IEEE International Conference on Image Processing (ICIP)* (pp. 2545–2548).

Zhao, T., & Nevatia, R. (2004a). Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *26*(9), 1208–1221.

Zhao, T., & Nevatia, R. (2004b). Tracking multiple humans in crowded environment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 2).

Zhao, T., Nevatia, R., & Wu, B.(2008). Segmentation and tracking of multiple humans in crowded environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *30*(7), 1198–1211.

Zhu, Q., Avidan, S., Yeh, M.-C., & Cheng, K.-T. (2006). Fast human detection by boosting histograms of oriented gradients. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Zou, W., Li, Y., Yuan, K., & Xu, D.(2009). Real-time elliptical head contour detection under arbitrary pose and wide distance range. *Journal of Visual Communication and Image Representation (JVCIR)*, *20*(3), 217–228.