Multiple Human Tracking in High-Density Crowds

Irshad Ali and Matthew N. Dailey

Computer Science and Information Management
Asian Institute of Technology,
Bangkok, Thailand
Irshad.Ali@ait.ac.th,
mdailey@ait.ac.th

Abstract. In this paper, we present a fully automatic approach to multiple human detection and tracking in high density crowds in the presence of extreme occlusion. Human detection and tracking in high density crowds is an unsolved problem. Standard preprocessing techniques such as background modeling fail when most of the scene is in motion. We integrate human detection and tracking into a single framework, and introduce a confirmation-by-classification method to estimate confidence in a tracked trajectory, track humans through occlusions, and eliminate false positive detections. We use a Viola and Jones AdaBoost cascade classifier for detection, a particle filter for tracking, and color histograms for appearance modeling. An experimental evaluation shows that our approach is capable of tracking humans in high density crowds despite occlusions.

Keywords: Human detection, head detection, pedestrian tracking, crowd tracking, AdaBoost detection cascade, particle filter.

1 Introduction

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

The pedestrian tracking problem is very difficult when the task is to monitor and manage large crowds in gathering areas such as airports and train stations. There has been a great deal of progress in recent years, but still most state-of-the-art systems are inapplicable to large crowd management situations because they rely on either background modeling [1,2,3], body part detection [3,4], or body shape models [5,6,1]. These techniques make it impossible to track large numbers of people in very crowded scenes in which the majority of the scene

J. Blanc-Talon et al. (Eds.): ACIVS 2009, LNCS 5807, pp. 540–549, 2009.

is in motion (rendering background modeling useless) and most of the humans' bodies are partially or fully occluded. Under these conditions, we believe that the *human head* is the only body part that can be robustly detected and tracked, so in this paper we present a method for tracking pedestrians by detecting and tracking their heads rather than their full bodies.

Our system assumes a single static camera placed a sufficient height that the heads of people traversing the scene can be observed. For initial detection we use a standard Viola and Jones Haar-like AdaBoost cascade [7], and for tracking we use a particle filter [8,9] for each head that incorporates a simple motion model and a color histogram-based appearance model.

Although this basic approach works very well in many cases, it suffers from two major issues: 1) shadows and other objects often cause false head detections, and 2) tracked heads are frequently lost due to partial or full occlusion. To address these issues, we introduce a *confirmation-by-classification* method that, on each frame, first uses the Viola and Jones classifier to confirm the tracking result for each live trajectory, then eliminates any live trajectory that has 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 a short period of time.

In an experimental evaluation with our current implementation, we find that on cheap hardware, the system requires approximately 2 seconds per frame to process a 640×480 video stream containing an average of 35.35 individuals per frame using 20 particles per head. We achieve a hit rate of 76.8% with an average of 2 false positives per frame. To our knowledge, this is the largest-scale human tracking experiment performed thus far, and the results are extremely encouraging. In future work, with further algorithmic improvements and run time optimization, we hope to achieve robust, real time pedestrian tracking for even larger crowds.

2 Human Head Detection and Tracking

In this section we first provide a summary of our tracking algorithm and then provide the details for each step. A block diagram is shown in Fig. 1.

2.1 Summary

1. Acquire input crowd video V.
2. In first frame v_0 of V, detect heads. Let N_0 be the number of detected heads.
3. Initialize trajectories T_j, $1 \leq j \leq N_0$ with initial positions $\boldsymbol{x}_{j,0}$.
4. Initialize occlusion count O_j for each trajectory to 0.
5. Initialize the appearance model (color histogram) $\boldsymbol{h}_j$ for each trajectory from the region around $\boldsymbol{x}_{j,0}$.
6. For each subsequent frame v_i of input video,
   (a) For each existing trajectory T_j,
      i. Use 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 $\boldsymbol{x}_{j,i}^{(k)}, 1 \leq k \leq K$.

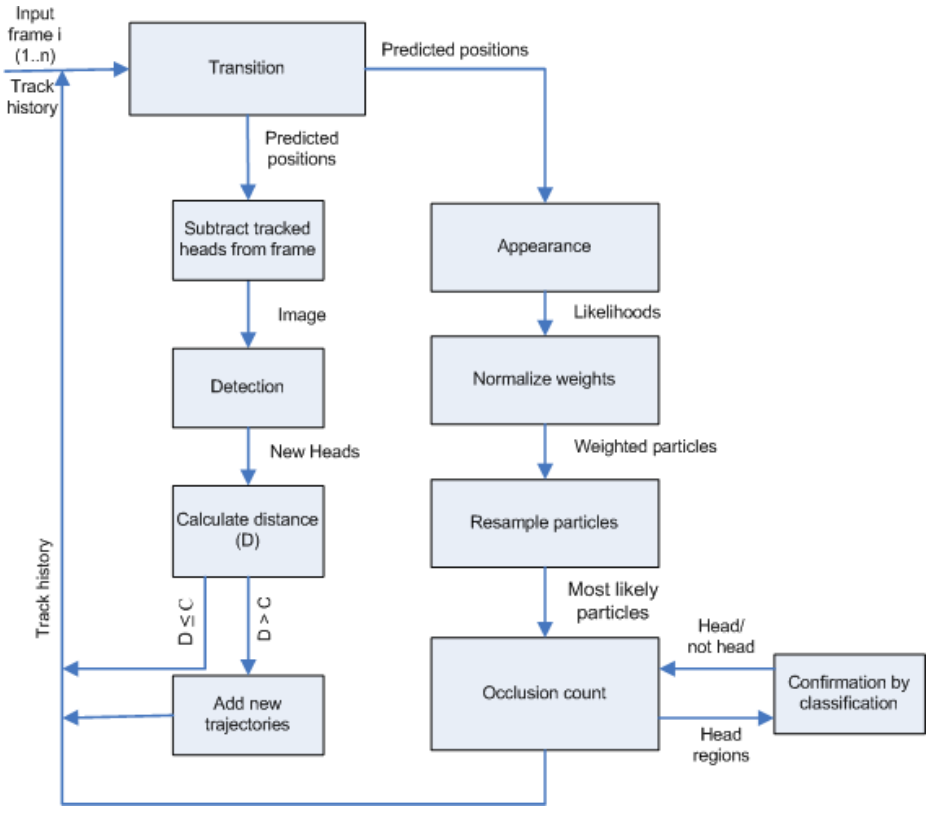

Fig. 1. Block diagram of the tracking algorithm. D is the distance between a newly detected head and a the nearest predicted location, and C is a threshold (in pixels) less than the width of the tracking window.

ii. Compute the color histogram $\boldsymbol{h}^{(k)}$ and likelihood $p(\boldsymbol{h}^{(k)} \mid \boldsymbol{x}_{j,i}^{(k)}, \boldsymbol{h}_j)$ for each particle k using the appearance model.
iii. Resample the particles according to their likelihood. Let k^* be the index of the most likely particle.
iv. Perform confirmation by classification: run the head detector on the location $\boldsymbol{x}_{j,i}^{(k^*)}$. If the location is classified as a head, reset $O_j \leftarrow 0$; else increase $O_j \leftarrow O_j + 1$.
v. If O_j is greater than threshold, remove trajectory j.

(b) Search for new heads in frame v_i and compute the Euclidean distance $D_{j,k}$ between each newly detected head k and each existing trajectory T_j. When $D_{j,k} > C$ for all j, where C is a threshold (in pixels) less than the width of the tracking window, initialize a new trajectory for detection k.

2.2 Detection

For object detection, there are many possibile algorithms; we use the Viola and Jones technique [7,10]. We train an AdaBoost cascade using Haar-like features off line. Then, at run time, we use the classifier in two ways, 1) as a detector, running a sliding window over the image at the specific range of scales expected for the scene, or 2) as a confirmer, to check whether the maximum likelihood head position predicted by the particle filter is sufficiently head-like to continue tracking.

2.3 Particle Filter

For tracking we use a particle filter [8,9]. The particle filter is well known to enable robust object tracking (see e.g. [11,12]). We use the standard approach in which the uncertainty about an object's state (position) is represented as a set of weighted particles, each particle representing one possible state. The filter propagates particles particles from frame $i-1$ to frame i using a motion model, computes a weight for each propagated particle using a sensor or appearance model, then resamples the particles according to their weights. The initial distribution for the filter is centered on the location of the object the first time it is detected. 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 $p(\boldsymbol{h}^{(k)} \mid \boldsymbol{x}_{j,i}^{(k)}, \boldsymbol{h}_j)$ 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. Without resampling, over time, the highest-weight particle would tend to a weight of one and the other weights would tend to zero. Resampling removes many of the low weight particles and replicates the higher-weight particles. We thus obtain a new set of equally-weighted particles. We use the resampling technique described in [13].

2.4 Motion Model

Our motion model is based on a second-order auto-regressive dynamical model. The autoregressive model assumes the next state y_t of a system is a function of some number of previous states and a noise random variable ϵ_t:

$$y_t = f(y_{t-1}, y_{t-2}, ..., y_{t-p}, \epsilon_t).$$

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.

2.5 Appearance Model

Our appearance model is based on color histograms. We compute a color histogram $\boldsymbol{h}_j$ in HSV space for each newly detected head and save it to compute particle likelihoods in future frames. To compute a particle's likelihood we use the Bhattacharyya similarity coefficient between model histogram $\boldsymbol{h}_j$ and observed histogram $\boldsymbol{h}^{(k)}$ as follows, assuming n bins in each histogram:

$$p(\boldsymbol{h}^{(k)} \mid \boldsymbol{x}_{j,i}^{(k)}, \boldsymbol{h}_j) \propto e^{-d(\boldsymbol{h}_j, \boldsymbol{h}^{(k)})},$$

where

$$d(\boldsymbol{h}_j, \boldsymbol{h}^{(k)}) = 1 - \sum_{b=1}^{n} \sqrt{h_{j,b} h_b^{(k)}}$$

and $h_{j,b}$ and $h_b^{(k)}$ denote bin b of $\boldsymbol{h}_j$ and $\boldsymbol{h}^{(k)}$, respectively. 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 we currently use a global histogram computed over the entire detection window because of its simplicity.

2.6 Confirmation-by-Classification

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

Recovery from Misses. Due to occlusion and appearance variation, we may not detect all heads in the first frame or when they initially appear. To solve this problem 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 50% of the width of the detection window.

Reduction of False Detections. Shadows and other non-head objects in the scene tend to produce transient false detections and tracking errors. In order to prevent transient false detections from being tracked through time, we use the head detector to confirm the estimated head position for each trajectory and eliminate any trajectory not confirmed for some number of frames. To implement this, we use a trajectory occlusion count. 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 estimated position through detection. Occlusion counts of trajectories not confirmed through classification are incremented, and occlusion counts of confirmed trajectories are reset to $O_j = 0$. Any trajectory that is not confirmed for some threshold number of frames is eliminated.

Tracking through Temporary Occlusion. The occlusion counting scheme just described also serves to help track a head through a partial or full occlusion. When a tracked head becomes partially or fully occluded, it will typically fail the confirmation by classification test, in which case we increase the trajectory's occlusion count. So long as an occlusion is brief and the motion model is not severely violated, the trjactory can be recovered in a subsequent frame.

3 Experimental Evaluation

3.1 Training Data

To train the Viola and Jones Haar-like AdaBoost cascade detector, we cropped 4325 heads from videos collected from various places and scaled them to 20×20 pixels. We also collected 2200 negative images. The detailed training parameters are given in Table 1. The training process required about 72 hours on an Intel Pentium 4 2.8GHz with 2GB RAM. We used the OpenCV `haartraining` utility to train the classifier.

Table 1. Head detector training parameters

Parameters	Values	Description
npos	4235	Number of positive samples
nneg	2200	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	20	Training image patch width and height
boosttypes	DAB	Discrete AdaBoost.

3.2 Test Data

To evaluate our algorithm, we captured a 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 2. We then hand labeled the locations of all heads at least 20×20 pixels in the first few frames and a sample of the subsequent frames with gaps to test long tracking. We labeled a total of 40 frames containing a total of 1414 heads, for an average of 35.35 heads/frame. To evaluate our algorithm we compare the tracked heads with the hand-labeled ground truth and determine number of correctly tracked heads, missed heads and false positives.

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 the simple crowd density measure

$$\text{Crowd density} = \frac{\text{Total number of pixels in all individuals' bounding boxes}}{\text{Total number of pixel in the frame}}.$$

Fig. 2. A sample frame from the Mochit station dataset

According to this measure, the highest crowd density in our test sequence is 0.63, which is higher than that of any publicly-available pedestrian tracking video database.

Since the publicly available pedestrian tracking databases only contain low density crowds, a direct comparison of our high-density tracking results with other researchers' low-density tracking results is unfortunately not possible. We have also attempted to run the publicly available pedestrian tracking implementations on our data, but they fail due to different assumptions about the data. In any case, to encourage other researchers to attempt tracking the pedestrians in our data set, we make the data available at `http://www.cs.ait.ac.th/mdailey/headtracking/acivs09-5950.zip`.

3.3 Implementation Details

We implemented the system in C++ with OpenCV without any special code optimization. Our approach relies mainly on object detection without a need for background subtraction. Our algorithm can thus track both moving and static humans in the scene. We detect heads and create initial trajectories from the first frame and then track heads from frame to frame. Further implementation details are given in the following sections.

Trajectory Initialization and Termination. We use the head detector to find heads in the first frame and create initial trajectories. To detect when new heads appear in subsequent frames, we also run the detector to search for heads in regions of each frame not predicted by the motion model for some existing trajectory. 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.

Identity Management. It is also important to assign and maintain object identities (IDs) automatically during tracking. We assign a unique ID to each trajectory during initialization. The tracking and confirmation-by-classification processes maintain the object ID during tracking. Trajectories that are temporarily lost due to occlusion are reassigned the same ID on recovery to avoid identity changes.

(a) Frame 2. (b) Frame 10.

(c) Frame 17. (d) Frame 22.

Fig. 3. Sample tracking results on the Mochit test video. Red rectangles indicate estimated head positions, and green rectangles indicate ground truth head positions.

3.4 Results

There were an average 35.35 individuals per frame over the 40 hand-labeled ground truth frames, for a total of 1414 heads. We used 20 particles per trajectory. The average correct tracking rate was 76.8%, with 2.05 false positives per frame and 8.2 missing heads per frame. The processing time was approximately 2 seconds/frame for a frame size of 640×480 on an Intel Pentium 4 2.8GHz with 2GB RAM. A few sample frames with tracking results are shown in Figure 3. Using a smaller frame size, a faster machine, parallelization or a GPU or multicore CPU, and/or code-level optimization, we expect that the processing time could be reduced significantly. Detailed results are shown in Table 2.

Table 2. Tracking Results

Total Heads	Tracked	False Positives	Missed
1414	1086 (76.8%)	82 (2.05/frame)	328 (8.2/frame)

Finally, to determine the difficulty of our data set in comparison to other existing pedestrian tracking datasets, we compared the density of our dataset with the CAVIAR dataset [14] and the Campus Plaza sequence [1], using the method described in Section 3.2. The results of the comparison are shown in Table 3.

Table 3. Crowd density comparison results

Mochit	CAVIAR [14]	Campus Plaza [1]
0.63	0.27	0.23

4 Discussion and Conclusion

Tracking people in high density crowds such as the one shown in Figure 2 is a real challenge and is still an open problem. In this paper, we propose a new algorithm based on a combination of head detection, appearance-based tracking with a particle filter, and confirmation-by-classification. Our experimental results demonstrate the promise of the method. It is particularly encouraging that the particle filter works well with a very small number of particles (20).

To further understand the performance of the algorithm, we examined the errors it makes on our test set more closely. We found that most of the missed heads were those which were partially or fully occluded. We also found that most of the false detections were shadows or human body parts other than heads. A few false detections arose from background features such as holes in the ground.

In future work we plan a more extensive evaluation of the method and improvements in the appearance model using other image features such as contours.

Acknowledgments

We thank Tao Zhao for providing the Campus Plaza sequence to us. Irshad Ali was supported by a graduate fellowship from the Higher Education Commission (HEC), Pakistan.

References

1. Zhao, T., Nevatia, R., Wu, B.: Segmentation and tracking of multiple humans in crowded environments. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(7) (2008)
2. Wu, B., Nevatia, R.: Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. International Journal of Computer Vision 75(2), 247–266 (2007)
3. Wu, B., Nevatia, R., Li, Y.: Segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2008)
4. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2008)
5. Ramanan, D., Forsyth, D.A., Zisserman, A.: Tracking people by learning their appearance. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(1), 65–81 (2007)
6. Zhao, T., Nevatia, R.: Tracking multiple humans in crowded environment. In: Proceedings IEEE Conference Computer Vision and Pattern Recognition, vol. 2 (2004)
7. Viola, P., Jones, M.: Robust real time object detection. International Journal of Computer Vision (IJCV) 57, 137–154 (2001)
8. Isard, M., Blake, A.: A mixed-state condensation tracker with automatic model-switching. In: IEEE International Conference on Computer Vision, pp. 107–112 (1998)
9. Doucet, A., de Freitas, N., Gordon, N.: Sequential Monte Carlo Methods in Practice. Springer, New York (2001)
10. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 511–518 (2001)
11. Kang, H.G., Kim, D.: Real-time multiple people tracking using competitive condensation. Pattern Recognition 38, 1045–1058 (2005)
12. Martnez, S.V., Knebel, J., Thiran, J.: Multi-object tracking using the particle filter algorithm on the top-view plan. In: European Signal Processing Conference, EUSIPCO (2004)
13. Rui, Y., Chen, Y.: Better proposal distributions: Object tracking using unscented particle filter. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. II786–II793 (2001)
14. CAVIAR: Data set, `http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1`