AUTOMATICALLY DETERMINING DOMINANT MOTIONS IN CROWDED SCENES BY CLUSTERING PARTIAL FEATURE TRAJECTORIES

Anil M. Cheriyadat and Richard J. Radke

Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180 USA

ABSTRACT

We present a system for automatically identifying dominant motions in a crowded scene. Accurately tracking individual objects in such scenes is difficult due to inter- and intra-object occlusions that cannot be easily resolved. Our approach begins by independently tracking low-level features using optical flow. While many of the feature point tracks are unreliable, we show that they can be clustered into dominant motions using a distance measure for feature trajectories based on longest common subsequences. Results on real video sequences demonstrate that the approach can successfully identify both dominant and anomalous motions in crowded scenes. These fully-automatic algorithms could be easily incorporated into distributed camera networks for autonomous scene analysis.

Index Terms— Crowd Motion Trajectories, Longest Common Subsequence, Clustering

1. INTRODUCTION

Recent advances in visual sensor technology, digital communications, and networking have enabled the deployment of a growing number of camera networks for varied surveillance applications. One current research focus involves integrating intelligent vision systems with these visual sensors to develop smart camera networks that can automate processes such as surveillance and event recognition. While earlier vision systems (e.g., [1, 2]) were focused on developing efficient tracking techniques for relatively isolated objects, our goal here is to automatically identify patterns of motion in highly crowded scenes, where it may be very difficult to accurately track individual objects.

In many visual tracking applications, object motion is represented by trajectories of feature points, segmented object contours [3] or object model centroids [4]. It is often important to group similar trajectories into clusters for modeling trajectory distributions or for understanding motion patterns.

This work was supported in part by the US National Science Foundation, under the award IIS-0237516. Thanks to Gabriel J. Brostow for providing the test video sequences and to the reviewers for their helpful comments.

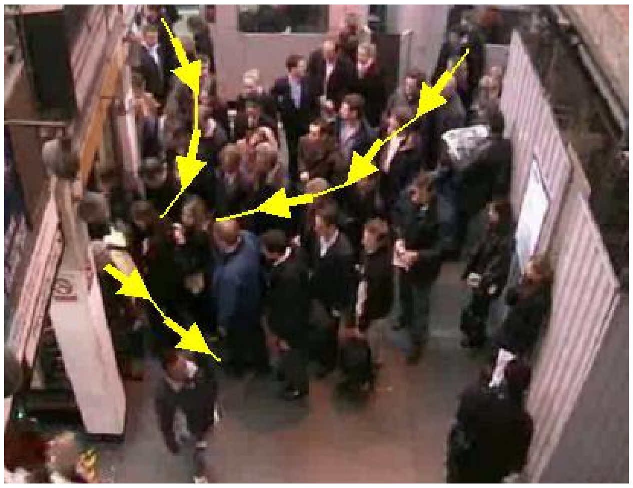

Fig. 1. An example frame from a video of a crowded scene. Dominant motions are indicated by the three arrowed lines.

In this paper, our objective is to cluster motion trajectories in scenes having high crowd density. Fig. 1 shows an example frame from a video of a high density crowd. The center of the image is crowded by tens of people getting on and off a train platform. High crowd density situations pose several challenges to automated tracking. For example, inter- and intra-object occlusions are highly common, and result in poor feature extraction and tracking. Hence, it is difficult to obtain a long, reliable track of a single feature point. Feature point tracks that represent motion of the same physical object are likely to disappear or diverge over the course of the video sequence. Clustering such motion trajectories is difficult, but is essential for several applications including event detection and content-based video retrieval.

Our approach begins by independently tracking low-level object features using an optical flow algorithm. Since our objective is to identify dominant, not individual, motions, we need not link, fix, or otherwise precondition these point tracks, unlike related work on counting individuals in a crowd (e.g.,[5, 6]). Instead, our clustering is based on the similarity of point track segments measured using an algorithm based on longest common subsequences (LCSS). While many of the individual feature point tracks are unreliable, we show that we can automatically cluster them using an appropriate ordering and metric, and identify dominant motions in a crowded scene.

The rest of the paper is organized as follows. In Section 2 we review some recent work on trajectory clustering. In Section 3 we outline our clustering framework. Results obtained from real video sequences are shown in Section 4. Section 5 concludes the paper with ideas for future work.

2. RELATED WORK

Formally, we define a trajectory as a set of points $\{(x_t, y_t), t = T_{init}, \ldots, T_{final}\}$ representing discrete spatial locations traversed by a single feature point over time. Generally, we expect feature points identified on the same physical object to have similar trajectories, as well as feature trajectories generated by other objects traversing the same spatial path.

Recently, Khalid and Naftel [7] showed that time-series modeling can be applied to object trajectory classification and pattern discovery. In their work, high-dimensional trajectory data is projected to a suitable lower-dimensional coefficient space. Classification and pattern discovery analysis is performed in the lower-dimensional space. They concluded that motion trajectories were well represented by frequency-domain coefficients. The coefficient vectors were used as input to a neural network algorithm that learned similarities between object trajectories in an unsupervised manner.

In their work related to counting pedestrians in a video sequence, Antonini and Thiran [8] showed that motion trajectories projected onto the independent component analysis (ICA) space yielded a better representation for clustering than the original time-series data. Recent work by Junejo et al. [9] showed that graph cuts can be used for clustering trajectories. Nodes of the graph represent trajectories; each node is connected to every other node, and the edge weights are the Hausdorff distances between the trajectories. Graph cuts are used to recursively partition the graph into binary clusters consisting of similar trajectories. Alon et al. [10] proposed a system for clustering similar object motions, based on a hidden Markov model (HMM). They assumed that motion trajectories are generated from a mixture of HMM models and estimated the mixing coefficients using a expectation-maximization framework.

Piciarelli and Foresti [11] proposed an online clustering method where clusters are dynamic and built in real time as trajectory data is collected. Here, the object paths are defined using a tree representation, and path segments represent the branches. Their algorithm assigns probabilistic values for each branch of the tree based on the trajectory data collected. Dynamic time warping (DTW) is yet another tool used in time-series analysis. Yi et al. [12] used a DTW method to efficiently group and retrieve similar time series data. Brostow and Cipolla [5] proposed a probabilistic Bayesian framework for clustering feature point trajectories. Their objective was to detect independent motions in crowds for applications such as counting individuals in crowded scenes.

Our proposed clustering scheme is most closely related to previous work reported by Buzan et al. [13] on clustering trajectories and Vlachos et al. [14] on time-series analysis. In the work proposed by Buzan et al. [13], moving foreground objects represented by blobs are segmented using a statistical background model. Segmented blobs are matched from one frame to the next and an extended Kalman filtering technique is used to improve the reliability of the extracted trajectories. Clustering is performed by measuring similarity between pairs of trajectories using a longest common subsequence (LCSS) algorithm. We expect that this trajectory extraction method might not yield good results for high density crowd video due to the difficulty of extracting reliable object-level trajectories. Vlachos et al. [14] proposed a novel framework for discovering similarity in multi-dimensional time-series data, which resulted in a significant increase in the execution speed of the LCSS algorithm. Our clustering scheme takes advantage of this efficient method.

3. CLUSTERING FRAMEWORK

The input to our system is a set of feature point tracks represented as

$$\{\{(x_t^i, y_t^i), t = T_{init}^i, \ldots, T_{final}^i\}, i = 1, \ldots, Z\}. \qquad (1)$$

Here, Z represents the total number of point tracks. The lengths of the tracks vary depending on the durations for which corresponding feature points are successfully tracked. Our goal is to cluster these point tracks into dominant patterns of motion- i.e., long trajectories through the scene along which a substantial number of feature point tracks exist. As mentioned above, this process is complicated by the fact that individual feature tracks in a crowd video are often short and unreliable. However, we make no attempt to link broken tracks or “fix” inaccurate ones, since these goals may involve scene-specific understanding. We overcome the problems by designing a distance metric that properly captures our intuition for when two trajectories are similar, and processing the trajectories in an order that encourages “good” clusters.

3.1. Extracting Feature Point Tracks

We first identify low-level features in the initial frame using the usual Tomasi-Kanade detector [15] as well as the Rosten-Drummond detector [16], a fast algorithm for finding corners. The low-level features are tracked over time using a hierarchical implementation [17] of the Kanade-Lucas-Tomasi optical flow algorithm [18]. To reduce computational load, new features are detected in every fifth frame. New features that are spatially too close to existing point tracks are discarded. The remaining new features are tracked along with the existing

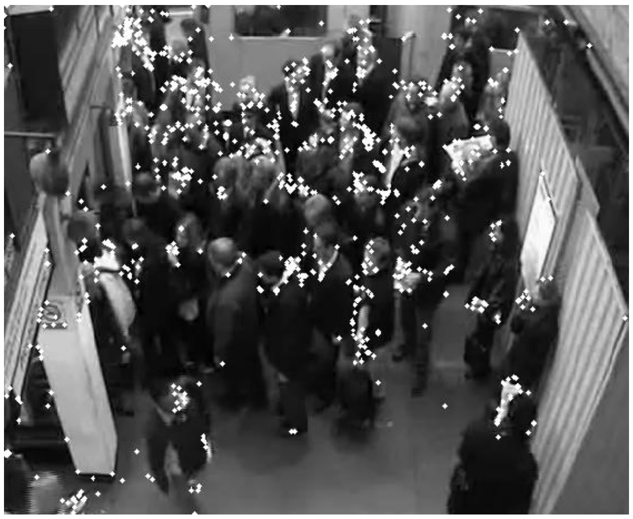

Fig. 2. Feature points identified for frame 300 of the platform sequence.

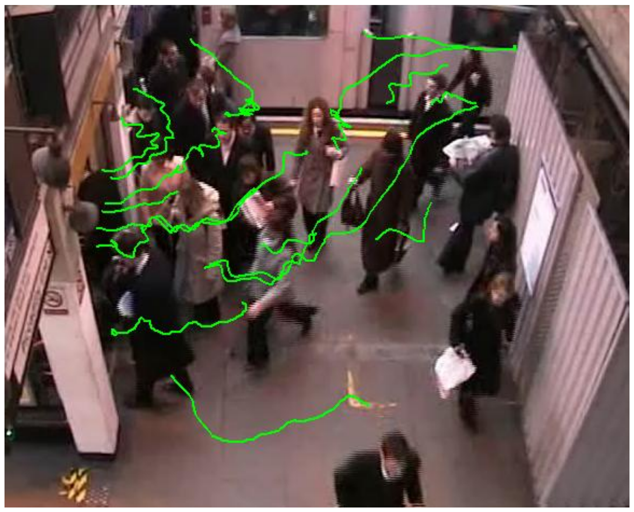

Fig. 3. Some of the longest point tracks extracted from a crowd scene video. The point tracks are overlaid on one of the frames from the video sequence.

point tracks to form a larger trajectory set. Fig. 2 shows the low-level feature points identified for an example frame.

High crowd density situations pose several challenges to feature point tracking. As a crowd gets denser, its movement gets slower, and due to inter- and intra-object occlusions, tracking feature points becomes difficult. Fig. 3 shows several of the longest feature point tracks extracted from a crowd sequence. The tracks are overlaid on one of the frames for spatial reference. We can observe from the figure that many of the feature point tracks cover only a small part of each object's motion. Feature point tracks exhibit large variations in their spatial extent and temporal duration, and it is not uncommon for tracks to be broken, or for one track to be "left off" by one object and "picked up" by a new object. One way to overcome these difficulties is to perform some type of spatial and temporal pre-conditioning of the trajectories. Such pre-conditioning is likely to succeed only if there are relatively few fragmented trajectories along with relatively many complete trajectories. In the work reported by Rabaud and Belongie [6] for counting moving objects in a crowd, trajectory conditioning is achieved by propagating a spatial window along the temporal direction of each trajectory. New spatial coordinates for fragmented trajectories are obtained by averaging other trajectory coordinates inside this spatial window. Since the objective is to count moving objects in a crowd, their interest is in extracting a set of equally sized point tracks representing a single object's motion even though the temporal duration of the tracks are small. In contrast, our goal is to identify dominant motions in crowd and hence our interest is in collecting reliable feature point tracks that represent an object's movement completely. Trajectory conditioning strategies applied over a longer temporal duration along the fragmented tracks might yield unreliable information. Instead, we use a clustering scheme based on longest common subsequences described below that requires no spatial or temporal pre-conditioning.

3.2. Longest Common Subsequences

Our goal is to cluster feature point tracks that are spatially close to each other and have a similar direction of motion. We therefore require a distance metric for comparing point tracks, which we base on the longest common subsequence for this pair.

Let A and B denote feature point tracks obtained by tracking two feature points. Similar to our earlier definition (1), we define A and B as follows:

$$A = \{(x_t, y_t), t = T_{init}, \ldots, T_{final}\} \tag{2}$$

$$B = \{(x'_t, y'_t), t = T'_{init}, \ldots, T'_{final}\}. \tag{3}$$

Let N and M represent the lengths (i.e., $T_{final} - T_{init}$) of tracks A and B respectively. The algorithm for identifying the longest common subsequence is implemented using a dynamic programming framework. To explain our implementation, we define the sequences $Head(A)$ and $Head(B)$ as follows:

$$Head(A) = \{(x_i, y_i), i = 1, 2, \ldots, N-1\} \tag{4}$$

$$Head(B) = \{(x'_i, y'_i), i = 1, 2, \ldots, M-1\}. \tag{5}$$

The longest common subsequence $LCSS(A, B)$ is defined recursively by Equation (6).

Here, the constant δ controls the flexibility of matching sequences in time and the constant ϵ controls the spatial matching threshold.

$$LCSS_{\delta,\epsilon}(A,B) = \begin{cases} 0 & \text{if } A \text{ or } B \text{ is empty} \\ 1 + LCSS_{\delta,\epsilon}(Head(A), Head(B)) & \text{if } \|A_N - B_M\|_2 < \epsilon \text{ and } |N-M| < \delta \\ \max(LCSS_{\delta,\epsilon}(Head(A), B), LCSS_{\delta,\epsilon}(A, Head(B))) & \text{otherwise} \end{cases} \tag{6}$$

A weighted version of the LCSS that is more suitable for computing trajectory similarity is given by:

$$S_{\delta,\epsilon}(A,B) = \frac{LCSS_{\delta,\epsilon}(A,B)}{\min(N,M)} \tag{7}$$

Let I denote the index array associating the longest common subsequence of matching points between each tracks. Fig. 4 shows the longest common subsequence of matching points identified for an example pair of feature point tracks with δ set to the length of longest track and ϵ set to 20. The index array I is defined as follows:

$$I = (I_i^A, I_i^B), i = 1, \dots, L \tag{8}$$

Here I^A and I^B represent the indices for tracks A and B respectively, and L denotes the total number of matching points.

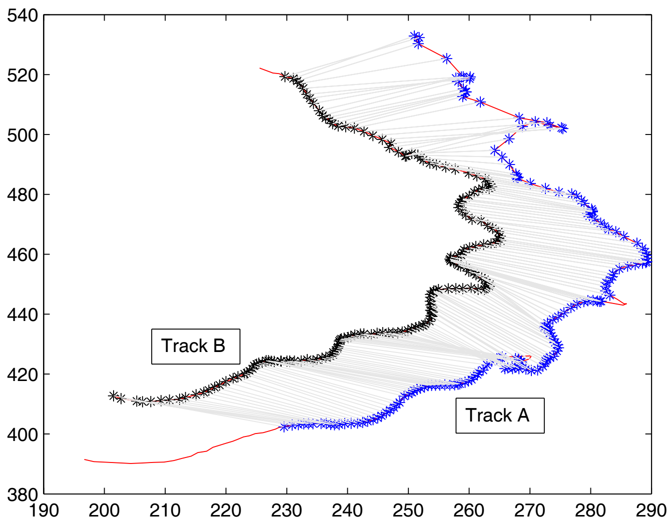

Fig. 4. The longest common subsequence extracted from two point tracks. Track A is longer than Track B. The dots on each track indicate matching points.

We define the *matching ratio* R for each track in a pair as the number of matching points L divided by the original size of the track. For example, the matching ratio for track A is computed as $R = L/N$.

We compute two further measures between the feature point tracks using the index sets I. First, the *spatial similarity* between feature point tracks A and B is defined as:

$$D_{spt}(A,B) = max\{\|A(I_i^A) - B(I_i^B)\|_2, i = 1, \dots, L)\}. \tag{9}$$

Second, the *directional similarity* between tracks A and B is defined as:

$$D_{ang}(A,B) = avg\{abs(\theta_i^A - \theta_i^B), i = 1, 2, \dots, L\}. \tag{10}$$

Here, θ_i^A is the angle of the vector $A(I_{i+1}^A) - A(I_i^A)$, measured in $[-\pi/2, \pi/2]$ (similarly for θ_i^B).

3.3. Clustering Trajectories

At this point we are ready to define our clustering algorithm as follows:

1. Point tracks are first sorted in descending order of the size of the tracks. Let $S = \{A^1, A^2, \dots, A^Z\}$ represent this sorted list, with lengths $\{N^1, N^2, \dots, N^Z\}$, respectively. Tracks with N below a threshold (usually 1) are discarded. Let C represent the set containing the cluster centers. Initially $n = 1$ and $C = \{A^1\}$; that is, the longest track is used as the initial cluster center.

2. Using the normalized longest common subsequence measure $S_{\delta,\epsilon}$, matching points I are found between each cluster center C^i and the next largest track A^{n+1}.

3. Using the matching points found for each cluster center and the next largest track, the spatial similarity D_{spt}^i, directional similarity D_{ang}^i, and matching ratio R^i are computed.

4. From the set C, find the cluster center i^* that minimizes the combination $D_{spt}^i + \alpha D_{ang}^i$, where α is a weighting factor. Typically, we use $\alpha = 0.5$, because the directional similarity measure can be susceptible to noise. The directional similarity measure D_{ang} can also be replaced with suitably scaled matching ratio value R.

5. If $D_{spt}^{i^*} < \tau_{spt}$, $R^{i^*} < \tau_R$, and $N^{n+1} > \tau_N$ for thresholds $\tau_{spt}, \tau_R, \tau_N$, then the track A^{n+1} is assigned as a new cluster center. That is, the next longest track is sufficiently long and sufficiently dissimilar from any existing cluster center.

6. Otherwise, track A^{n+1} is assigned to the cluster with center C^{i^*}.

7. Set $n = n + 1$ and iterate Steps 2-6 until all the tracks are processed.

The threshold τ_{spt} controls the spatial separation needed for the tracks to be part of separate clusters. The threshold

τ_R represents the percentage of matching points that needs to identified on a track for it to be part of a cluster. The threshold τ_N indicates the minimum track length required to be assigned as a cluster center. After all the tracks have been clustered, clusters having very few members (e.g. around 10) are discarded since they do not represent dominant motions.

We note that to obtain good performance, it is especially important to cluster the tracks by decreasing length. While a complete track for each dominant motion is not required, it is essential to have at least one single track that contain a major part of the dominant motion. Since the tracks are sorted based on their size, tracks that cover a major part of each dominant motion are automatically picked up by the clustering algorithm as cluster centers. The shorter tracks are later assigned to each cluster based on similarity. From our experiments, we observed that the clustering algorithm is able to identify clusters associated with the dominant motions after analyzing the first 30 to 50 point tracks. As a future extension of this work, we intend to modify our algorithm to iteratively improve the cluster center representation as the cluster size grows. This will allow the algorithm to offer similar clustering performance in case there are no longer tracks available to be used as cluster centers.

4. EXPERIMENTAL RESULTS

We tested the algorithm on two different video sequences having different crowd densities. The first video sequence, termed the *platform sequence*, shows tens of people entering and exiting a train platform. Fig. 5 shows a few example frames from this video sequence. One group of people heads towards the exit directly from the train, and another group of people heads towards the same exit from other parts of the platform. The scene quickly becomes congested near the exit. There are also a few people entering the platform in the opposite direction through an entry gate near the exit. Dominant motions identified manually for this video sequence were shown in Fig. 1 as yellow arrowed lines.

Features were tracked over 300 frames, which resulted in a total of around 1500 point tracks. The extracted feature points were fed into our clustering algorithm, which automatically identified three dominant motions, as illustrated in Fig. 6. We can see that the clusters semantically correspond to the same dominant motions manually identified in Fig. 1.

The second video sequence, termed the *campus sequence*, shows a busy campus walkway, as illustrated in Fig. 7. Around 1000 feature points were tracked and the sequence had around 700 frames. In this case, the algorithm correctly identified the two dominant upward and downward motions. However, the algorithm also identified a third non-trivial cluster indicating a substantial anomalous motion. This anomaly corresponds to a single person who begins by walking up the campus lane but suddenly takes a U-turn to join a group walking downward.

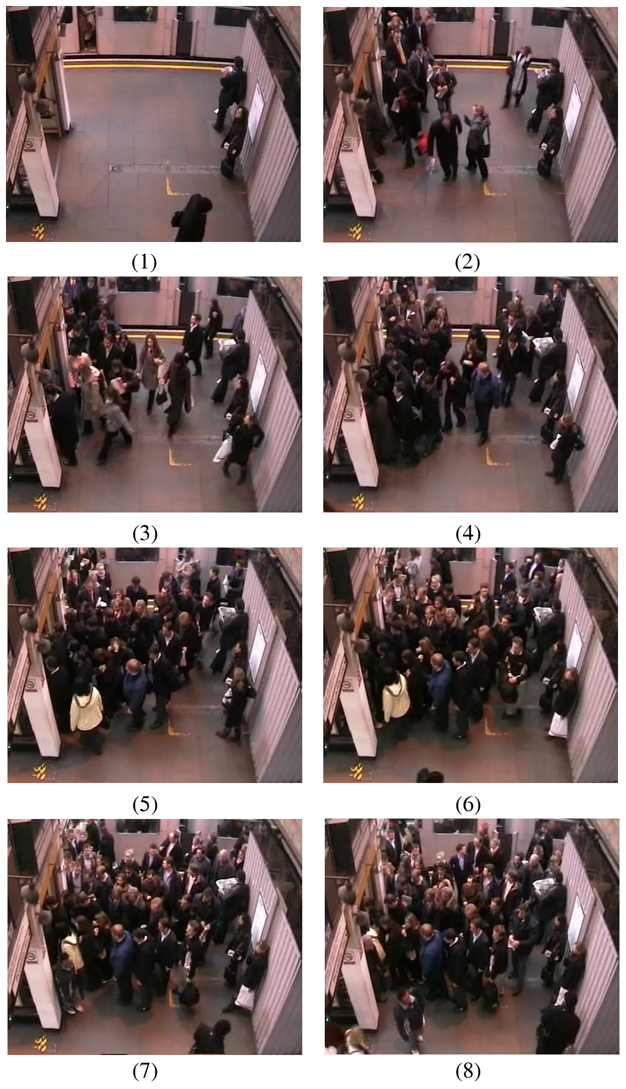

Fig. 5. Example frames from the *platform* sequence.

5. CONCLUSIONS

We presented a system for automatically identifying dominant motions in crowd by clustering low level feature point tracks. The feature point trajectories extracted from dense crowd scenes are often fragmented. However, results on real video sequences demonstrate that the proposed clustering algorithm can identify both dominant and anomalous motions in crowded scenes by clustering these partial feature trajectories. The proposed algorithm relies on longer tracks for representing cluster centers. To overcome this limitation, in the future we plan to iteratively update each cluster center based on a medial-axis-like line through the cluster members.

Future improvements will include better techniques for

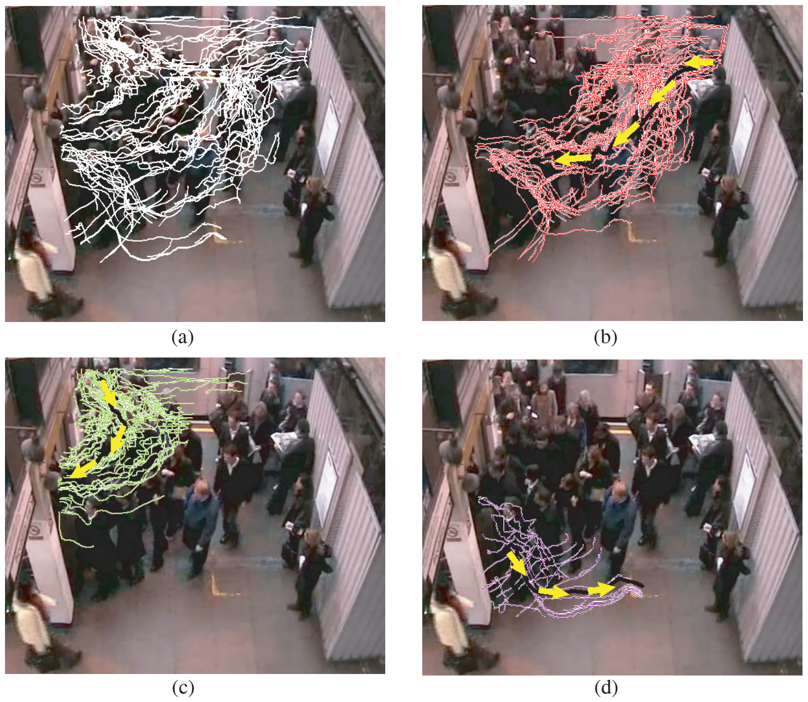

Fig. 6. (a) Some of the point tracks extracted from the *platform* video sequence. (b), (c) and (d) show three different clusters identified by the algorithm. The arrows overlaid on each dominant motion cluster show its direction.

feature point track extraction to reduce point track noise. Recently proposed feature point tracking techniques based on piecewise smoothness models [19] and on combining local and global motion models [20] may be suitable for point track noise reduction. We will also address a distributed camera implementation of the proposed algorithm.

6. REFERENCES

[1] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proc. of IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001.

[2] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[3] M. S. Allili and D. Ziou, "Object contour tracking in videos by matching finite mixture models," in *Proc. of IEEE Conference on Video and Signal Based Surveillance*, 2006.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[5] G. J. Brostow and R. Cipolla, "Unsupervised Bayesian detection of independent motion in crowds," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 594–601.

[6] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 705–711.

[7] S. Khalid and A. Naftel, "Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space," *Multimedia Systems*, vol. 12, no. 3, pp. 227–238, 2006.

[8] G. Antonini and J. P. Thiran, "Counting pedestrians in video sequences using trajectory clustering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 8, pp. 1008–1020, 2006.

[9] I. N. Junejo, O. Javed, and M. Shah, "Multi feature path modeling for video surveillance," in *Proceedings of the 17th International Conference on Pattern Recognition*, 2004, vol. 2, pp. 716–719.

[10] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data," in *Proc.*

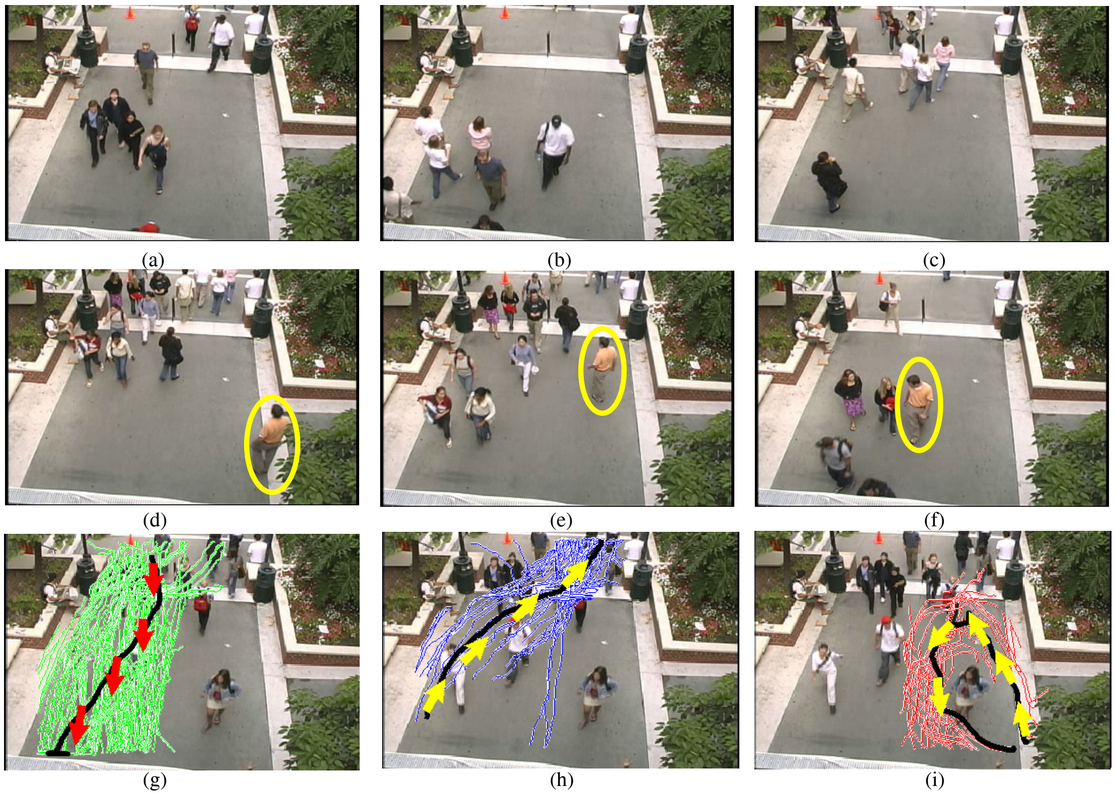

Fig. 7. (a)-(f) Several example frames from the *campus* sequence. Many people walk up and down the campus lane, while a single person (indicated by an ellipse) walks up the campus lane and makes a U-turn. (g), (h) and (i) show three different clusters identified by the algorithm for this video sequence.

of IEEE conference on Computer Vision and Pattern Recognition, 2003.

[11] C. Piciarelli and G. L. Foresti, "On-line trajectory clustering for anomalous events detection.," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1835–1842, 2006.

[12] B. Yi, H.V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proc. of International Conference on Data Engineering*, 1998, pp. 201–208.

[13] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories in video," in *Proc. of International Conference on Pattern Recognition*, 2004.

[14] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measure," in *Proc. of SIGKDD*, 2003.

[15] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, April 1991.

[16] E. Rosten and T. Drummond, "Machine learning for high speed corner detection," in *Proc. of European Conference on Computer Vision*, 2006.

[17] G. Bradski, "Opencv:examples of use and new applications in stereo, recognition and tracking," in *Proc. of International Conference on Vision Interface*, 2002.

[18] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of 7th International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[19] P. Sand and S. Teller, "Particle video: Long range motion estimation using point trajectories," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[20] A. Buchanan and A. Fitzgibbon, "Combining local and global motion models for feature point tracking," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.