Detecting Dominant Motions in Dense Crowds

Anil M. Cheriyadat and Richard J. Radke, *Senior Member, IEEE*

Department of Electrical, Computer, and Systems Engineering

Rensselaer Polytechnic Institute, Troy, NY, 12180 USA

cheria2@rpi.edu, rjradke@ecse.rpi.edu

(Invited Paper)

Abstract

We discuss the problem of detecting dominant motions in dense crowds, a challenging and societally important problem. First, we survey the general literature of computer vision algorithms that deal with crowds of people, including model- and feature-based approaches to segmentation and tracking as well as algorithms that analyze general motion trends. Second, 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 smooth 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 analysis, object trajectories, longest common subsequences, clustering, anomaly detection

I. Introduction

The automated analysis of images and video of crowded scenes is becoming increasingly important for understanding patterns of activity in urban areas. Crowds can arise in benign situations such as busy streets, sporting events, public celebrations, transit hubs, and retail environments, as well as in more dangerous or confrontational scenarios including political rallies, mobs, brawls, natural disasters, or mass evacuations. In practice, analyzing images or video of densely crowded scenes is often a manual process. Typically, a grid pattern is superimposed on an aerial photo of a crowd, and visual counts of the population of a subset of grid squares are combined via a weighted average to arrive at crowd size estimates. Estimates derived in this way frequently disagree with those of "on-the-spot" event organizers in politically sensitive rally or protest situations [1], [2], which can lead to allegations

Please address correspondence to R.J. Radke.

of governmental or media bias [3]. We believe that computer vision algorithms can aid in the impartial analysis of crowded scenes. Furthermore, automated trend analysis of crowd video can answer questions that aerial still photos cannot, such as the directions of dominant flow, the sources of bottlenecks, the time that crowd size peaks, the rate at which a crowd disperses, and so on.

This paper has two main contributions. First, we present a survey of computer vision algorithms that deal with crowds of people. We begin by reviewing algorithms for background subtraction, a fundamental preprocessing step in many crowd analysis algorithms. Next, we review model-based crowd analysis algorithms, in which some type of human model is applied to segmentation or tracking. This is followed by a discussion of feature-based algorithms, in which the patterns of motion of interest points are used as the basis for analysis, with no explicit human model. Finally, we discuss research on estimating activity patterns in crowds, which may not require explicit person-level segmentation. These algorithms are suitable for highly dense crowds and form the background for the second half of the paper.

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

Our second contribution is an algorithm for automatically identifying dominant patterns of motion in highly crowded scenes, initially proposed in [4]. Figure 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. Such high-density crowd situations pose several challenges to automated tracking algorithms in which humans are represented by parametric models [5], outer contours [6] or blob centroids [7]. Inter- and intra-object occlusions are highly common, resulting in poor low-level feature extraction and high-level model fitting. Our approach is instead to analyze motion patterns based on tracked low-level feature points, although long, reliable feature tracks may be difficult to obtain in dense crowds. Clustering such “noisy” motion trajectories is difficult, but essential for several applications including motion pattern analysis, anomalous 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., [8], [9]). Instead, our clustering is based on the similarity of point track segments measured using longest common subsequences. 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 smooth dominant motions in a crowded scene by fitting polynomials to the cluster centers.

The remainder of the paper is organized as follows. Section II contains our survey of computer vision algorithms for crowded scenes. In Section III, we review recent work on trajectory clustering. In Section IV, we outline our clustering framework. Results obtained from several real video sequences are presented in Section V. Section VI concludes the paper with ideas for future work.

II. Computer Vision for Crowds

In this section, we survey computer vision algorithms proposed for detecting or analyzing the motion of people in crowds. We begin with a general discussion of background modeling, followed by specific applications to detecting and tracking people. Since a huge number of papers have addressed the detection and tracking of isolated people or a small number of people, we focus on situations germane to crowded scenes. For a good survey of general analysis and tracking of humans in video, see [10].

While he did not explicitly address computer vision problems, Still [11] extensively studied crowd dynamics from the perspective of simulated entities that choose paths through a geometric space based on the principle of least effort. The resulting simulations provide insight into the nature of crowd dynamics and can be used to validate hypotheses about the effects of architectural changes or the causes of dangerous incidents.

A. Background Modeling

Many algorithms for detecting people in video taken by a stationary camera rely on background subtraction, based on a model learned from "empty" frames or estimated on-the-fly. We briefly mention several techniques for the general foreground/background separation problem here. For more information on background modeling in general, please refer to [12], [13]. For a survey of the closely related problem of change detection, see [14].

Commonly, a mixture of Gaussians is used to model pixel intensities for background subtraction. That is, the probability of observing the intensity $I_t(x, y)$ at location (x, y) and time t is modeled as the weighted sum of K Gaussian distributions. At each point in time, the probability that a pixel's intensity is due to each of the mixtures is estimated, and the most likely mixture defines the pixel's class. The mean and covariance of each background pixel are usually initialized by observing several seconds of video of an empty scene.

Stauffer and Grimson [15] presented a widely used approach to dynamically update a background model composed of a multi-Gaussian mixture. At each time instant, every pixel value is compared against the existing set of models at that location to find a match. The parameters for the matched model are updated based on a learning factor. If there is no match, the least-likely model is discarded and replaced by a new Gaussian with statistics initialized by

the current pixel value. The weights for all models are updated and renormalized. Finally, the models that account for some predefined fraction of the recent data are deemed "background" and the rest "foreground". Additional steps were proposed to cluster and classify foreground pixels into semantic objects (including people) and track the objects over time.

Other notable approaches to background modeling are based on non-parametric kernel density estimates for the intensity of the background and each foreground object [16], historical values of the minimum intensity, maximum intensity, and maximum interframe difference [17], and an adaptive Wiener filter to predict each pixel's current value from a linear combination of its k previous values [12].

Densely crowded scenes pose several problems for background modeling algorithms, since an empty background image may be impossible to obtain, and even so, almost all of the pixels may be foreground in each frame. Consequently, many crowd analysis algorithms apply model-based or feature-based approaches for detecting and tracking humans, as discussed next.

B. Model-Based Approaches

In Wren et al.'s Pfinder algorithm [5], following background subtraction, dynamic "blob" models for a person's head, hands, torso, etc. are fit to the foreground pixels for each object using a minimum-description-length (MDL) approach combined with a Kalman filter. However, this algorithm was not designed for video containing multiple, overlapping people.

Isard et al. proposed a system called Bramble [18] that used Bayesian particle filtering to track multiple person-blobs through a video sequence. The number of people and their locations (i.e. support regions) are simultaneously estimated. They learned a 4-Gaussian mixture model for the background from video of an empty scene, and assumed that foreground objects were modeled by a 16-Gaussian mixture. While effective, the system was only demonstrated to track up to three people; Zhao and Nevatia [19] noted that particle-filtering approaches were unlikely to scale well in dense crowds. Khan et al. [20] described a Markov chain Monte Carlo (MCMC)-based particle filter for tracking many ants in a dish that avoids the computational limits of joint particle filters, although the scenario they address is qualitatively different than video of humans in crowds.

Haritaoglu et al.'s W^4 algorithm [17] used a second-order motion/appearance model to track the heads, hands, feet, and torsos of isolated people, and integrated a posture analysis component to determine body pose and estimate whether each person was carrying an object. This paper was notable for being one of the first algorithms to deal with multiple people walking in groups. They first used a statistical model to separate the silhouettes of individuals from those of multi-person groups. Next, they considered the one-dimensional vertical projection of a multi-person blob as the basis for counting and segmenting the people in the blob. Their key observation was that large peaks in the vertical projection often correspond to the heads of individual people. Once the number of heads in each multi-person blob was determined, the blob was split into individuals by associating pixels with the closest head. Tracking individuals in a moving multi-person blob was based on tracking and associating heads over time. While effective, the algorithm makes a key assumption that all of the heads in each multi-person group can be unambiguously

separated from the background, which is difficult in occluded or highly crowded situations.

Zhao and Nevatia [21] presented a key advance in finding people in crowds that addressed the difficulty of head segmentation in the W^4 algorithm. They observed that in crowds, heads frequently do not lie on the boundary of foreground blobs, so that vertical projection can be uninformative. Therefore, they augmented a W^4-type head detector with a head-and-shoulders model applied to the edge map within the interior of each foreground blob. These two detectors generate a number of head candidates that direct a Markov chain Monte Carlo (MCMC) algorithm to create new human hypotheses. They parameterized each human with a low-dimensional ellipsoidal model for the head, torso, and legs, and used MCMC to maximize the a posteriori probability of the number of people and their parameters. They used an exponential prior on the number of people in the scene and a mean-shift algorithm for updating the model parameters. While it was demonstrated on images containing 10-20 people moving in crowds, the method was somewhat slow and operated on one still frame at a time, not taking advantage of the temporal coherence of video. Zhao and Nevatia refined their approach and applied it to tracking multiple humans in video of a crowded environment in [19]. They used the same ellipsoidal model and overall approach, with some new MCMC dynamics to address the temporal aspect of video. They reported good performance at 3 frames per second, but only on a single test video. The overall framework is summarized in the journal paper by Wu and Nevatia [22], which includes discussion of a boosting-based method to learn body part detectors using edgelets and the Bayesian combination of these into human hypotheses.

Khan and Shah [23] proposed a multiview approach to tracking people in somewhat crowded scenes. While human features are not explicitly modeled, it is assumed that the feet of pedestrians are visible and that all motions occur on a ground plane. By mapping estimated foreground regions from different images onto the same plane using homographies, feet regions can be extracted and clustered into individuals using graph cuts.

C. Feature-Based Approaches

An alternate approach to human-specific models for crowd analysis is simply to track interest points such as Harris corners [24], and cluster these into individual people. For example, Tu and Rittscher [25] posed the clustering of interest points into individuals as finding a set of maximal cliques in a graph. The interest points form the graph vertices, and each edge weight corresponds to the probability that two vertices arise from the same individual. Since their video was taken from an overhead view, these edge weights were based on the assumption that the vertices lie on the circular contour of a human seen from above. They used a soft-assign approach to optimize the memberships of interest points into clusters that converge from the outside of the graph (i.e. interest points that unambiguously belong to a single individual) inwards. It is not clear whether this approach would work equally well from a non-overhead view.

Rittscher et al. [26] later proposed a second approach for segmenting people in crowds, using maximum-likelihood estimation of shape parameters from image observations via hidden assignment vectors of features to cliques. They used a variant of the expectation-maximization algorithm for the estimation. The features are extracted from the bounding contours of foreground blob silhouettes, and each clique (representing an individual) is parameterized as

a simple bounding box.

Leibe et al. [27] proposed an approach for detecting pedestrians in crowded scenes, assuming that they walk parallel to the image plane. This approach requires training on a set of hundreds of ground-truth segmentations of pedestrians. These are used to build a bag-of-features type codebook for objects (based on squares centered at detected interest points). These local detectors are combined with global shape cues learned from training silhouettes into an MDL-type method to assign foreground pixels to human support regions using mean shift estimation [7].

Rodriguez and Shah [28] recently proposed a related approach for detecting and segmenting humans in medium-to-dense crowd scenes with many inter-object occlusions. Their algorithm integrates both global and local cues, learning local descriptors consistent with different global postures using a training dataset. For a new image, detected local descriptors described using shape contexts [29] vote for human locations and postures.

D. Estimating Activity Patterns

In densely crowded scenes, it may be difficult or impossible to accurately segment individual people; instead, a rough person count or estimation of the dominant motions in the scene may be of interest. In addition to the new algorithm we describe in the remainder of the paper, here we review related work on learning activity patterns in crowded video.

Reisman et al. [30] described an algorithm for processing the optical flow from a vehicle-mounted camera that simply detects whether or not a crowd exists in front of the vehicle. Davies et al. [31] described a technique for estimating crowd density in which the number of foreground pixels remaining after background subtraction and the number of edge pixels in an image were combined using a linear Kalman filter to estimate the number of pedestrians. Marana et al. [32] described an approach in which feature vectors of crowd images obtained from statistical and spectral texture analysis were used to train a neural network to classify crowd density into five qualitative categories.

Andrade et al. [33] described an approach using principal component analysis (PCA) on optical flow fields and spectral clustering on Hidden Markov Models (HMMs) for identifying unusual events in crowds. However, their results were only tested on two cases of simulated data [34].

Yan and Forsyth [35] described a simple head-counting algorithm for analyzing the behavior of pedestrians in a public space for architectural design purposes, with heuristics for splitting multi-person blobs into individuals. Their goal was overall analysis of pedestrian patterns and not highly accurate counting; also, their camera was also sufficiently far from the scene that good background models could be built despite the presence of large numbers of people.

Brostow and Cipolla [8] discussed scenarios in which crowds were so dense that background subtraction or model-based detection approaches would fail, since the crowd takes up most of the frame and there are few meaningful boundaries between entities. They proposed an unsupervised Bayesian algorithm for clustering tracked low-level interest points based entirely on motion, not on appearance. Using a spatial prior and a likelihood model for coherent

motion, they obtained qualitatively encouraging results on crowded videos of people, bees, ants, and so on. The algorithm we describe below was inspired by this approach.

Zhan et al. [36] proposed an algorithm for discovering flow paths in crowded scenes without any explicit object or feature point tracking. They estimated probability density functions for the foreground occurrence and motion direction at each pixel, and used these to trace high-probability paths through the image starting from its boundary. These paths were fit with splines to give smooth estimates of motion "modes". This idea seems promising and should be studied more thoroughly.

Finally, Ali and Shah [37] studied the problem of segmenting video of extremely dense crowds (e.g. marathons, rallies, or pilgrimages) from a bird's eye view. While it is impossible to resolve individuals at this scale, they treated the moving crowd as an aperiodic dynamical system that could be studied with Lagrangian particle dynamics. They showed how the crowd flow could be divided into regions with qualitatively coherent behavior using normalized cuts, and how changes in flow could be interpreted as abnormalities or instabilities. However, since all particles with the same source or destination must be grouped together, it may be difficult to extract all of the semantically important dominant motions in a given video.

III. Related Work on Trajectory Clustering

Now we turn to the main technical contribution of this paper, a framework for automatically determining dominant motions in crowded scenes by clustering partial feature trajectories that extends the work proposed in [4]. 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 [38] 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 [39] 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. [40] 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. [41] 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 [42] 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. Yi et al. [43] proposed a method based on dynamic time warping (DTW) to efficiently group and retrieve similar time series data. As discussed above, Brostow and Cipolla [8] 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. [44] on clustering trajectories and Vlachos et al. [45] on time-series analysis. In the work proposed by Buzan et al. [44], 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. As discussed further below, this approach might not yield good results for high-density crowd video due to both the difficulty of extracting reliable object-level trajectories and the high computational cost of comparing all pairwise tracks. Vlachos et al. [45] 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.

IV. Clustering Framework

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

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

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.

A. Extracting Feature Point Tracks

We first identify low-level features in the initial frame using the standard Shi-Tomasi-Kanade detector [46] as well as the Rosten-Drummond detector [47], a fast algorithm for finding corners. The low-level features are tracked over time using a hierarchical implementation [48] of the Kanade-Lucas-Tomasi optical flow algorithm [49]. To reduce computational load, new features are detected in every fifth frame. New features that are spatially too close

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.

to existing point tracks are discarded. The remaining new features are tracked along with the existing point tracks to form a larger trajectory set. Figure 2 shows the low-level feature points identified for an example frame.

High-density crowd 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. Figure 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 [9] 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. However, 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.

B. 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 A' denote two feature point tracks defined as:

$$A = \{(x_t, y_t), t = 1, \ldots, N\} \tag{2}$$

$$A' = \{(x'_t, y'_t), t = 1, \ldots, N'\}. \tag{3}$$

$$M(A_n, A'_{n'}) = \begin{cases} 0 & \text{if } A_n \text{ or } A'_{n'} \text{ is empty} \\ 1 + M(A_{n-1}, A'_{n'-1}) & \text{if } \|A(n) - A'(n')\|_2 < \epsilon \text{ and } |n - n'| < \delta \\ \max(M(A_{n-1}, A'_{n'}), M(A_n, A'_{n'-1})) & \text{otherwise} \end{cases} \tag{4}$$

Here, both tracks have been shifted to start at $t = 1$ for notational convenience. We define the subsequence A_n for $1 \leq n \leq N$ as the first n elements of A, i.e. $A_n = \{(x_t, y_t), t = 1, \ldots, n\}$. $A'_{n'}$ is defined similarly for $1 \leq n' \leq N'$. A_0 and A'_0 are defined to be the empty set.

We now recursively define a matching cost $M(A, A')$ between the trajectories by (4). Here, the constant δ controls the flexibility of matching sequences in time and the constant ϵ controls the spatial matching threshold. The value $M(A_N, A'_{N'})$ determines the matching cost between the entire trajectory pair, and can be efficiently computed using dynamic programming as follows.

We define a two-dimensional array Q with N rows and N' columns, populated from the upper left element to the lower right element as described in Algorithm 1. The matching cost $M(A_N, A'_{N'})$ is given by the maximum array value $Q(N, N')$. The longest common subsequence between the two trajectories is implicitly defined by the corresponding points induced by the process of computing M. We denote this set of point pairs as I, namely

Algorithm 1 Determine matching cost

for n = 1 to N **do**
  for n' = 1 to N' **do**
    if n=1 or n'=1 **then**
      $Q(n, n')$ = 0
    else if $\|A(n) - A'(n')\|_2 < \epsilon$ and $|n - n'| < \delta$ **then**
      $Q(n, n') = 1 + Q(n-1, n'-1)$
    else
      $Q(n, n') = \max(Q(n-1, n'), Q(n, n'-1))$
    end if
  end for
end for

$$I = (I_i^A, I_i^{A'}), i = 1, \ldots, L, \tag{5}$$

where I^A and $I^{A'}$ represent the matching indices for tracks A and A' corresponding to the matching cost M, and L denotes the total number of matching points. The longest common subsequence can be obtained by the usual dynamic programming backtracking algorithm that begins at the lower right element of Q and ends at the upper left element, as described in Algorithm 2.

Algorithm 2 Determine longest common subsequence

$n \leftarrow N$
$n' \leftarrow N'$
$I \leftarrow \emptyset$
while $n > 1$ and $n' > 1$ **do**
  if $\|A(n) - A'(n')\|_2 < \epsilon$ and $|n - n'| < \delta$ **then**
    $I \leftarrow \{I, (n, n')\}$
    $n \leftarrow n - 1$
    $n' \leftarrow n' - 1$
  else if $Q(n, n'-1) \geq Q(n-1, n')$ **then**
    $n' \leftarrow n' - 1$
  else
    $n \leftarrow n - 1$
  end if
end while

After computing I we remove horizontal and vertical links along the boundaries of the array from I and the definition of L.

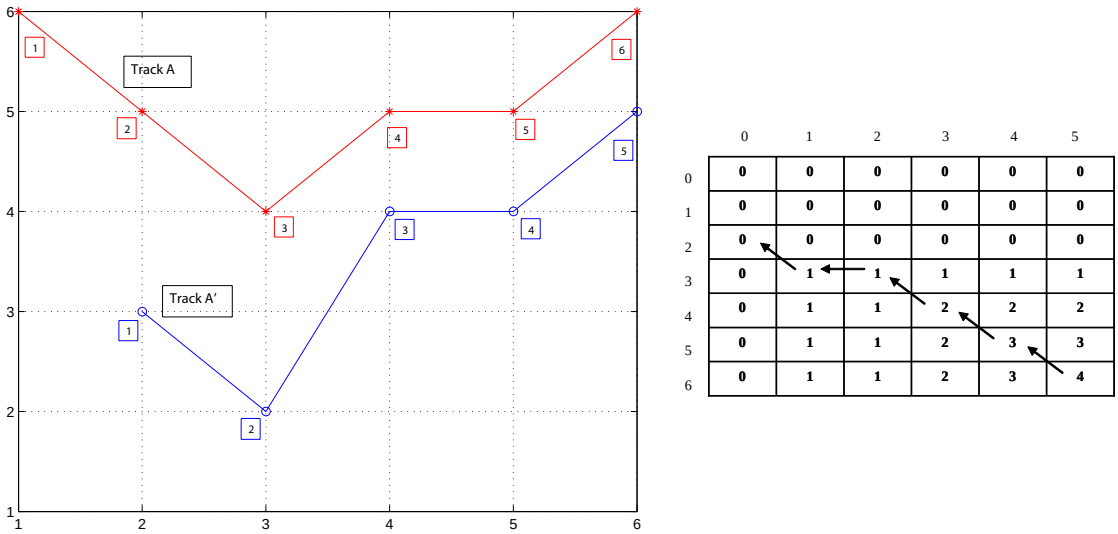

Fig. 4. (a) Trajectories A and A' having lengths 6 and 5 respectively. (b) The two-dimensional array populated for computing the matching cost. The first row and column indicate initial conditions. The arrows indicate the backtracking sequence order to find the matching point pairs.

Figure 4(a) illustrates two example trajectories A and A'. For this example $N = 6$, $N' = 5$, $\epsilon = 1.5$, and $\delta = 2$. Figure 4(b) illustrates the populated array Q constructed by Algorithm 1, and indicates the backtracking path constructed by Algorithm 2. In this case, the matching points are found to be $I = \{(6,5),(5,4),(4,3),(3,1)\}$, corresponding to the matching cost $M(A,A')$=4.

Figure 5 shows the longest common subsequence of matching points identified for several example pairs of feature point tracks extracted from a crowd scene video with δ set to the length of the longest track and ϵ set to 50.

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 also compute the *spatial similarity* between two feature point tracks A and A' as:

$$D_{spt}(A,A') = max\{\|A(I_i^A) - A'(I_i^{A'})\|_2, i = 1,\ldots,L)\}. \quad (6)$$

C. 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 length of the tracks. Let $S = \{A^1, A^2, \ldots, A^Z\}$ represent this sorted list, with lengths $\{N^1, N^2, \ldots, N^Z\}$, respectively. Tracks with N below a threshold (usually 1)

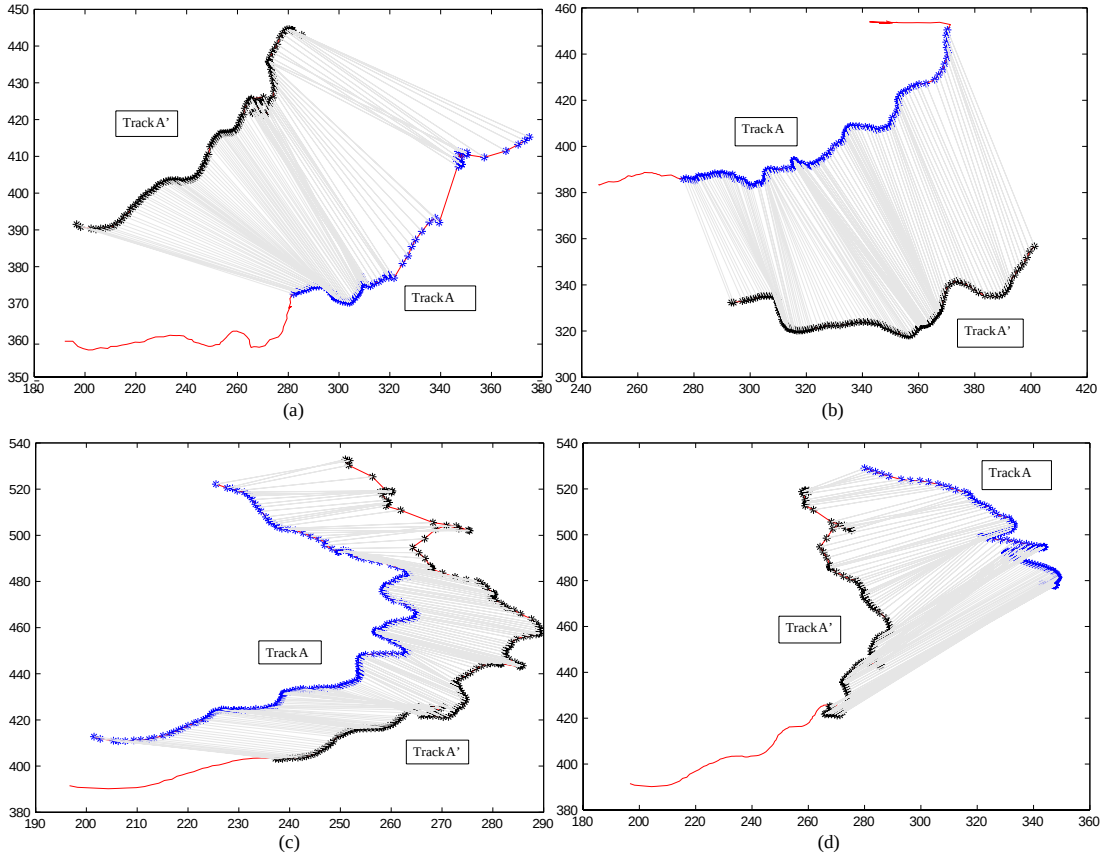

Fig. 5. The longest common subsequences extracted for four pairs of point tracks. The gray lines between each track pair indicate matching points.

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 longest common subsequence algorithm, matching points I are found between each cluster center C^i and the next longest track A^{n+1}.
3) Using the matching points found between each cluster center C^i and A^{n+1}, the spatial similarity D^i_{spt} and matching ratio R^i (with respect to the track) are computed.
4) The cluster center C^{i^*} that minimizes the combination $D^i_{spt} + \alpha R^i$ is computed, where α is a weighting factor that scales between R^i (always between 0 and 1) and D^i_{spt} (which depends on the pixel dimensions of the image and the spatial extent of the tracks). We used α=100 in all the experiments reported here.
5) If $D^{i^*}_{spt} > \tau_{spt}$, $R^{i^*} < \tau_R$, and $N^{n+1} > \tau_N$ for thresholds $\tau_{spt}, \tau_R, \tau_N$, then the track A^{n+1} is established as a new cluster center. That is, a new cluster is formed if 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) The first time a cluster's size exceeds a positive multiple of a value V, the cluster center is updated using a least-squares polynomial fit through its members' data as described below. For our experiments, we used a sixth-degree polynomial with V=30. That is, the cluster center is updated when it contains 30, 60, 90, etc. members.
8) Set $n = n + 1$ and iterate Steps 2-7 until all the tracks are processed.
9) To obtain the final dominant motion paths, clusters are merged if their centers' similarity cost is greater than 50% (see below). Polynomials are fit through the final members of each cluster as in Step 7.

In Step 5, the thresholds τ_{spt}, τ_R, and τ_N control the spatial proximity, percentage of matching points, and minimum track length required for a track to join a cluster. For the experiments described in the next section, τ_{spt} is set between 25 and 150, with higher values corresponding to video sequences with larger frame sizes. The matching ratio threshold τ_R is set between 0.25 and 0.5. The threshold τ_N is set between 150 and 250. After all the tracks have been clustered, clusters having very few members (e.g. around 20) are discarded since they do not represent dominant motions.

In Step 7, we periodically replace the cluster center by a least-squares polynomial fit through its member points once sufficient data has been collected. In our initial algorithm [4], long trajectories served as cluster centers without being updated. However, in dense crowds with unreliable feature tracks, no single trajectory may be long or accurate enough to well-represent the eventual cluster. Here, we improve the cluster representation by fitting a K^{th} degree polynomial $y = \sum_{k=0}^{K} a_k x^k$ through the collected (x, y) points of all trajectories associated with the cluster. In this way, the eventual cluster centers will smoothly interpolate the complete dominant motions represented by the cluster, as illustrated in Figure 6. The polynomial fit is only recomputed periodically to reduce the algorithm's computational cost. Khalid and Naftel [38] used a similar polynomial fitting strategy for trajectory representation. Another approach would be to fit a piecewise polynomial (e.g., a spline) through the trajectory points; however, to avoid overclustering around noisy trajectories, the cluster representation should be rough rather than fitting trajectories too closely.

Clustering with short, fragmented feature point tracks can result in separate clusters forming at different parts of a single "true" dominant motion path, especially during the first few iterations. While the cluster center update in Step 7 mitigates this issue, some over-clustering may remain at the end of Step 8. For this reason, in Step 9 cluster centers with significant overlap are merged if they meet the criterion $M(A, A') > 0.5 \cdot \min(N, N')$.

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, at least one single track should 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 initially chosen by the clustering algorithm as cluster centers. The shorter tracks are later assigned to each cluster based on similarity, and refine the cluster centers via the polynomial fit once a cluster has sufficient membership. 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.

(a) (b)

Fig. 6. (a) A few trajectories from one cluster is shown here. None of the trajectories covers the entire motion path. (b) The corresponding final cluster center obtained by periodically updating the cluster center with a polynomial fit covers the entire motion path.

V. Experiments

A. Results

Here, we report results of the algorithm on several video sequences with differing crowd densities. The original source videos, as well as results from each stage of the algorithm, can be viewed at http://www.ecse.rpi.edu/~rjradke/jstsp/.

The first video sequence, termed the *platform sequence*, shows tens of people entering and exiting a train platform. Figure 7 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 Figure 1 as yellow arrowed lines. Features were tracked over 300 frames, which resulted in a total of around 1200 point tracks. The extracted feature points were fed into our clustering algorithm, which automatically identified four dominant motions, as illustrated in Figure 8. We can see that the clusters semantically correspond to the same dominant motions manually identified in Figure 1.

The second video sequence, termed the *campus sequence*, shows a busy campus walkway, as illustrated in Figure 9. Around 1000 feature points were tracked and the sequence had 600 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.

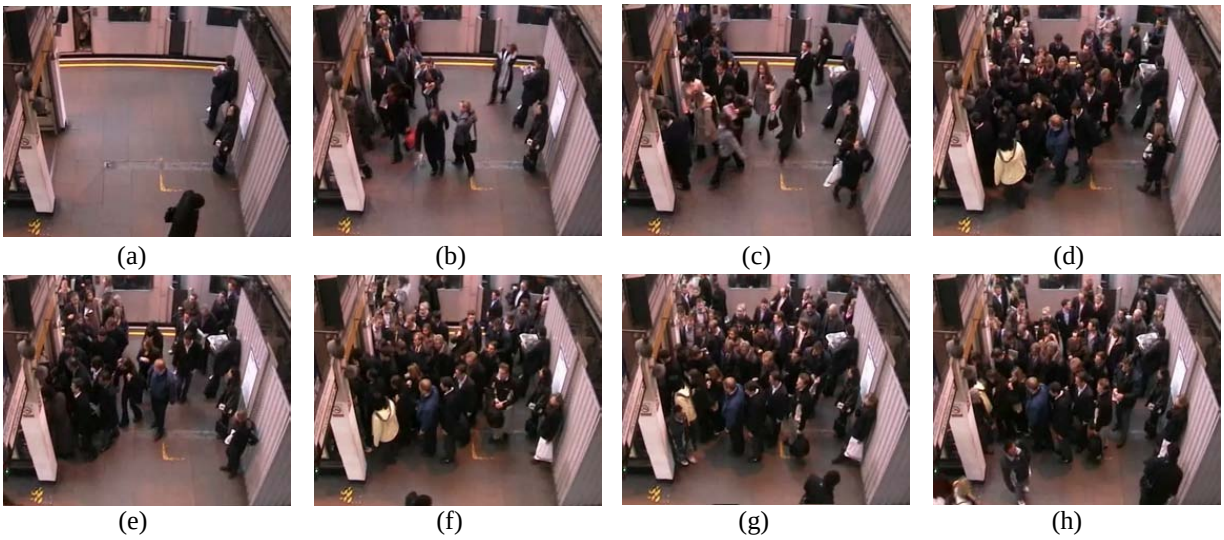

Fig. 7. Example frames from the *platform* sequence.

Fig. 8. (a)-(d) Four different clusters identified by the algorithm for the *platform* sequence. (e)-(h) Cluster centers generated by the algorithm representing the four dominant motions. The solid red rectangle denotes the starting location. (This figure is best viewed in color.)

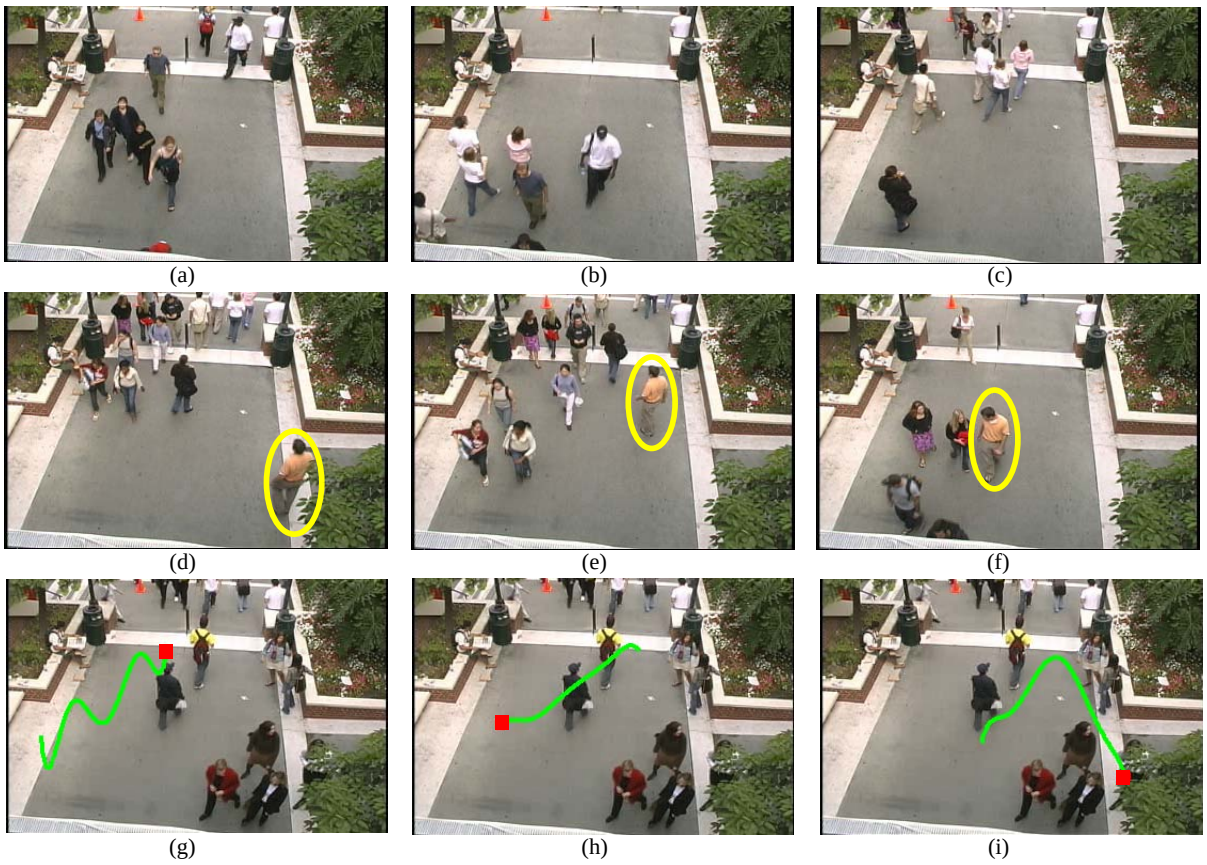

Fig. 9. (a)-(f) Several example frames from the *campus* sequence. Many people walk up and down the campus lane, while a single person (indicated by an ellipse) walks up the campus lane and makes a U-turn. (g), (h) and (i) show three different dominant motion paths identified by the algorithm for this video sequence. (This figure is best viewed in color.)

The third video sequence, termed the *escalator sequence*, shows people getting on two escalators, as illustrated in Figure 10. In this case, although all people move in the same direction, the algorithm correctly identifies two different clusters representing forward crowd motion at the two escalators. Around 700 feature points were tracked across 300 frames for this sequence.

The fourth video sequence, termed the *airport sequence*, was obtained from the PETS database for benchmarking visual tracking algorithms [50]. This sequence shows two individuals loitering in front of a ticketing office, while other individuals cross through the image or enter the ticketing office, as illustrated in Figure 11. For this sequence, our algorithm identified four dominant motions: one representing people loitering in front of the ticketing office, a second representing people entering the ticketing office, and two others indicating people crossing through the scene. Around 300 feature points were tracked over 1200 frames.

We note that it is difficult to quantitatively validate and analyze the results (e.g., in the form of ROC curves), since the number and nature of dominant motions in the scene require a subjective human interpretation. We believe our

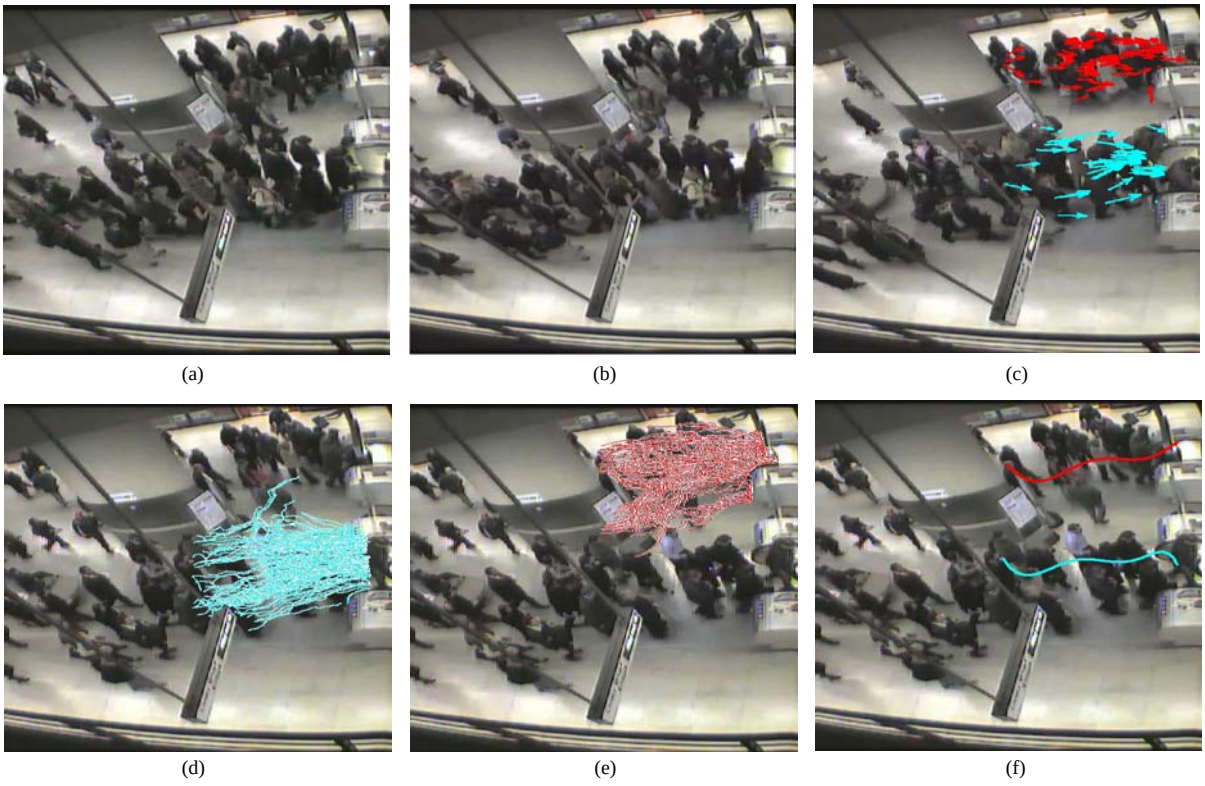

Fig. 10. (a)-(b) Example frames from the *escalator* sequence, illustrating people getting on two different escalators. (c) Two dominant motions identified by the algorithm are shown here by the red and blue arrows overlaid on a sample frame. (d)-(e) illustrate the clusters identified by the algorithm and (f) shows the two polynomially fit cluster centers. (This figure is best viewed in color.)

results agree with common sense and neither over- or under-segment the semantically relevant dominant motions in the scenes studied here.

B. Parameter Sensitivity

To study the sensitivity of the clustering results to different values of the parameters, we ran the algorithm on the escalator sequence with different values of τ_{spt} and τ_R, and τ_N fixed to 200. The number of clusters resulting from each (τ_{spt}, τ_R) combination is given in Table I. As would be expected, more stringent clustering criteria (i.e., higher τ_R and lower τ_{spt}) result in the generation of more clusters. However, the range of τ_{spt} and τ_R values that generate the "natural" two dominant motions is fairly broad. To generate the clustering result illustrated in Figure 10, we used $\tau_{spt} = 120$ and $\tau_R = 0.25$. Our experimental analysis showed that for a parameter setting of $\tau_{spt} = 90$ and $\tau_R = 0.25$, the clustering algorithm generated an additional cluster representing the motion of a few people moving from one queue to the other. Readers are recommended to watch the online video of this sequence to observe this secondary "cross-over" motion. As mentioned above, it is difficult to determine an objective "ground truth" in such

Fig. 11. (a)-(d) Example frames from the *airport sequence*. The rectangular box overlaid on the frames denotes the region where analysis was performed. (e)-(h) Trajectories belonging to the four different clusters identified by the algorithm. (i)-(l) The cluster centers generated by the algorithm correspond to the dominant motions in the video sequence: (i) crossing from right to left, (j) crossing from left to right, (k) entering the ticketing office from the left, (l) entering from the right and loitering. (This figure is best viewed in color.)

TABLE I

THE NUMBER OF CLUSTERS RESULTING FROM VARIOUS (τ_{spt}, τ_R) COMBINATIONS FOR THE ESCALATOR SEQUENCE.

τ_{spt} | τ_R	0.25	0.5	0.75
30	4	8	14
60	4	7	11
90	3	6	7
120	2	2	2
150	2	2	2

scenarios.

Since the parameters are related to the image coordinate system, they are also affected by the video frame size (e.g., the *platform sequence* is 576×720 while the *campus sequence* is 240×360). If the camera is calibrated and pixels can be related to physical distances in world coordinates, it may be possible to reduce the parameter variability.

Some tuning of the basic parameters is generally required to obtain semantically useful results for a given sequence, depending on the video frame size, the crowd density, and the speed of individuals. Before the algorithm is run on a long video sequence, (e.g. morning commuters entering and exiting a train platform during prime

commuting hours of a work week), an analyst can tune the parameters to appropriate values by observing the estimated dominant motions over a short (e.g. 1000 frame) characteristic training sequence. These parameters should suffice for categorizing subsequent dominant motions over a longer term.

C. Timings and computational complexity

The main computationally intensive steps of our algorithm are feature tracking and trajectory clustering. As mentioned above, we use a hierarchical implementation [48] of the Kanade-Lucas-Tomasi optical flow algorithm [49] for feature tracking, which is fairly fast. The trajectories are generated roughly at the frame rate of the original sequence (i.e. 15 frames per second).

To compare two trajectories of lengths N and N', Algorithm 1 for creating the table entries of Q requires $O(N \cdot N')$ computations, and Algorithm 2 for extracting the longest common subsequence requires $O(N + N')$ computations. Clustering algorithms such as [44] that compute a pairwise similarity measure between all the trajectories in a sequence as the basis for clustering would require $O(Z^2)$ longest common subsequence computations. Since Z is on the order of hundreds or thousands in our examples, such algorithms would be computationally intractable. On the other hand, since our algorithm only requires each trajectory to be compared against the current cluster centers, it requires $O(CZ)$ longest common subsequence computations, where C is the ultimate number of clusters. In our examples, C is less than 5, i.e., at least two orders of magnitude less than Z.

The experiments were conducted on an Intel dual-core 2.4GHz processor with 2GB RAM. In practice, our non-optimized Matlab algorithm for clustering took around 5 minutes for the platform sequence and 2-3 minutes for all the other sequences. We found that the computational time is proportional to the typical length of the trajectories. Since the dense crowd moves relatively slowly in the platform sequence, long feature point tracks are generated that span the entire frame and take longer to compare. In such cases, our algorithm has a large speed advantage over one that measures the similarity between all pairs of tracks.

VI. Conclusions

We presented a system for automatically identifying dominant motions in crowds 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.

We believe that the proposed algorithm can be directly used for long-term crowd scene analysis. The generated dominant motion paths are semantically-useful representations of long-term crowd activity. One next step might be to develop dominant-motion-based video query algorithms to extract video segments that contain or exclude desired behaviors.

In the future, we plan to incorporate improved techniques for feature point track extraction to reduce point track noise. Recently proposed feature point tracking techniques based on piecewise smoothness models [51] and combining local and global motion models [52] may be suitable for point track noise reduction. We also plan to

analyze longer videos, in which the notions of dominant vs. anomalous motion may change over time. Finally, we plan to investigate algorithms for automatically tuning the parameters of the algorithm, leveraging image-to-world coordinate transformations when they are available, and pursuing information-theoretic approaches [53] to determine the optimal clustering.

A further direction for future work is the implementation of the current algorithm for an embedded platform in a multi-camera network. Since the algorithm distills high data rate video into a small number of polynomial curves, it is a good choice for aggregating and sharing data in a bandwidth-limited distributed camera network. The next step is to design distributed algorithms for multiple camera nodes to fuse and make decisions based on their detected dominant motions as the basis for higher-level scene analysis. For example, the local dominant motions could be used to build a network-level picture of crowd volume or unrest and direct resources accordingly.

ACKNOWLEDGMENTS

Thanks to Gabriel Brostow for providing several of the test video sequences. This work was supported in part by the US National Science Foundation, under the award IIS-0237516.

REFERENCES

[1] W. Buchanan, “Photos show 65,000 at peak of S.F. rally; Aerial study casts doubt on estimates of 200,000,” The San Francisco Chronicle, February 21, 2003.

[2] M. Levenson, “Crowd size could be in the eye of beholder,” The Boston Globe, November 1, 2004.

[3] D. Howell, “Dissatisfaction on the marches,” The Washington Post, February 4, 2007.

[4] A. Cheriyadat and R. Radke, “Automatically determining dominant motions in crowded scenes by clustering partial feature trajectories,” in *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC-07)*, September 2007.

[5] 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.

[6] 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.

[7] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[8] 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*, vol. 1, 2006, pp. 594–601.

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

[10] D. M. Gavrila, “The visual analysis of human movement: A survey,” *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82–98, 1999.

[11] G. K. Still, “Crowd dynamics,” Ph.D. dissertation, University of Warwick, 2000.

[12] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, “Wallflower: Principles and practice of background maintenance,” in *Proc. ICCV ’99*, 1999, pp. 255–261.

[13] R. Collins, A. Lipton, and T. Kanade, “Introduction to the special section on video surveillance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 745–746, August 2000.

[14] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, “Image change detection algorithms: A systematic survey,” *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 294–307, March 2005.

[15] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, August 2000.

[16] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using non-parametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, July 2002.

[17] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, August 2000.

[18] M. Isard and J. MacCormick, "BraMBLe: A bayesian multiple-blob tracker," in *Proc. of International Conference on Computer Vision*, 2001, pp. 34–41.

[19] T. Zhao and R. Nevatia, "Tracking multiple humans in crowded environment," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 406–413.

[20] Z. Khan, T. Balch, and F. Dellaert, "An MCMC-based particle filter for tracking multiple interacting targets," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2004.

[21] T. Zhao and R. Nevatia, "Stochastic human segmentation from a static camera," in *Proc. of IEEE Workshop on Motion and Video Computing*, 2002.

[22] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, 2007.

[23] S. M. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Proceedings of the European Conference of Computer Vision*, 2006.

[24] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988.

[25] P. Tu and J. Rittscher, "Crowd segmentation through emergent labeling," in *Proc. ECCV Workshop on Statistical Methods in Video Processing*, 2004.

[26] J. Rittscher, P. Tu, and N. Krahnst, "Simultaneous estimation of segmentation and shape," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 486–493.

[27] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 878–885.

[28] M. D. Rodriguez and M. Shah, "Detecting and segmenting humans in crowded scenes," in *Proceedings of ACM Multimedia*, 2007.

[29] G. Mori, S. Belongie, and J. Malik, "Efficient shape matching using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1832–1837, 2005.

[30] P. Reisman, O. Mano, S. Avidan, and A. Shashua, "Crowd detection in video sequences," in *Proceedings of the 2004 IEEE Intelligent Vehicles Symposium*, 2004.

[31] A. Davies, J. H. Yin, and S. Velastin, "Crowd monitoring using image processing," *Electronics and Communication Engineering Journal*, vol. 7, no. 1, pp. 37–47, February 1995.

[32] A. Marana, S. Velastin, L. Costa, and R. Lotufo, "Automatic estimation of crowd density using texture," in *Proceedings of the International Workshop on Systems and Image Processing (IWSIP-97)*, 1997.

[33] E. Andrade, S. Blunsden, and R. Fisher, "Modelling crowd scenes for event detection," in *Proceedings of the 18th International Conference on Pattern Recognition*, vol. 1, 2006, pp. 175–178.

[34] E. Andrade and R. Fisher, "Simulation of crowd problems for computer vision," in *Proceedings of the First International Workshop on Crowd Simulation*, 2005.

[35] W. Yan and D. A. Forsyth, "Learning the behavior of users in a public space through video tracking," in *Proc. of IEEE Workshop on Applications of Computer Vision*, vol. 1, 2005, pp. 370–377.

[36] B. Zhan, P. Remagnino, and S. Velastin, "Mining paths of complex crowd scenes," in *Proceedings of the First International Symposium on Advances in Visual Computing (ISVC)*, December 2005.

[37] S. Ali and M. Shah, "A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[38] 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.

[39] 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.

[40] 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*, vol. 2, 2004, pp. 716–719.

[41] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[42] 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.

[43] B. Yi, H. 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.

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

[45] 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.

[46] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

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

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

[49] 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.

[50] "PETS benchmark data," 2007, UK EPSRC REASON Project consortium, Proc. of IEEE workshop on Performance Evaluation and Tracking and Surveillance, http://www.cvg.rdg.ac.uk/PETS2007/data.html.

[51] 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.

[52] 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.

[53] A. Cohen, C. Bjornsson, Y. Chen, G. Banker, E. Ladi, E. Robey, S. Temple, and B. Roysam, "Automatic summarization of changes in image sequences using algorithmic information theory," in *Proc. of IEEE International Symposium on Biomedical Imaging*, 2008.

Anil Cheriyadat Anil Cheriyadat received his B.Tech degree from the Cochin University of Science and Technology in 1997 and the M.S. degree in Electrical Engineering from Mississippi State University in 2003. He is currently pursuing his Ph.D. in the department of Electrical, Computer and Systems engineering at Rensselaer Polytechnic Institute, NY, USA.

Anil started his career (1997-2001) as a software engineer and was involved in developing and maintaining business applications on high-end IBM servers. From 2003-2005, he held research associate positions at Mississippi State University and Oak Ridge National Laboratory and was engaged in developing pattern recognition and machine learning algorithms for overhead image information extraction. Currently he is at Oak Ridge National Laboratory pursuing his Ph.D. research on object detection and tracking in crowded environments. His main research interests are in computer vision, pattern recognition and signal processing. Anil received the Barrier Fellowship scholarship (2002-03) for academic excellence from the department of Electrical and Computer Engineering, Mississippi State University.

Richard J. Radke Richard J. Radke received the B.A. degree in mathematics and the B.A. and M.A. degrees in computational and applied mathematics, all from Rice University, Houston, TX, in 1996, and the Ph.D. degree from the Electrical Engineering Department, Princeton University, Princeton, NJ, in 2001. For his Ph.D. research, he investigated several estimation problems in digital video, including the synthesis of photorealistic ”virtual video”, in collaboration with IBM’s Tokyo Research Laboratory. He has also worked at the Mathworks, Inc., Natick, MA, developing numerical linear algebra and signal processing routines.

Dr. Radke joined the faculty of the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, in August, 2001, where he is now an Associate Professor. He is also associated with the National Science Foundation Engineering Research Center for Subsurface Sensing and Imaging Systems (CenSSIS). His current research interests include distributed computer vision problems on large camera networks, modeling 3D environments with visual and range imagery, deformable registration and segmentation of three- and four-dimensional biomedical volumes, and machine learning for radiotherapy applications. Dr. Radke received a National Science Foundation CAREER Award in 2003 and is a member of the 2007 DARPA Computer Science Study Group.