Learning Motion Patterns in Crowded Scenes Using Motion Flow Field

Min Hu, Saad Ali and Mubarak Shah
Computer Vision Lab, University of Central Florida
{*mhu,sali,shah*}*@eecs.ucf.edu*

Abstract

Learning typical motion patterns or activities from videos of crowded scenes is an important visual surveillance problem. To detect typical motion patterns in crowded scenarios, we propose a new method which utilizes the instantaneous motions of a video, i.e, the motion flow field, instead of long-term motion tracks. The motion flow field is a union of independent flow vectors computed in different frames. Detecting motion patterns in this flow field can therefore be formulated as a clustering problem of the motion flow fields, where each motion pattern consists of a group of flow vectors participating in the same process or motion. We first construct a directed neighborhood graph to measure the closeness of flow vectors. A hierarchical agglomerative clustering algorithm is applied to group flow vectors into desired motion patterns.

1. Introduction

Learning typical motion patterns of moving objects in the scene from videos is an important visual surveillance task. A significant amount of effort has been been put in this area. In [8], the system first tracks moving objects, learns the motion patterns, and finally uses these motion patterns for abnormal event detection. Johnson and Hogg [3] use neural networks to model motion paths from trajectories. In [2], trajectories are iteratively merged into a path. Wang et al. [5] proposed a trajectory similarity measure to cluster trajectories and then learn the scene models from trajectory clusters. Porikli [1] represents trajectories in the HMM parameter space; the affinity matrices are then computed and eigenvalue decomposition is applied to find the clusters. Instead of using trajectories of individual objects, Vaswani et al. [6] model the motion of all the moving objects performing the same activity by analyzing the temporal deformation of the "shape" which is constructed by joining the locations of objects at any time t. These methods are based on long-term motion tracks of moving objects and are applicable to the scenes where moving objects are not densely crowded and motion tracks of objects are reliable and are readily available. All these methods require tracking algorithms to generate long-term locations of individual objects. Little attention has been paid to learning motion patterns in crowds where reliable tracks are harder to obtain.

To deal with this special scenario, instead of using long-term motion tracks of moving objects we propose a new method for learning typical motion patterns using the motion flow field. The motion flow field is a set of flow vectors representing the instantaneous motions in a video. Each flow vector is a four-dimensional vector including the location and instantaneous velocity of a point detected in a frame. We first use existing optical flow method to compute flow vectors in each frame and then combine them into a global motion field. Given the motion flow field, we detect the typical motion patterns by clustering flow vectors. Without loss of generality, we assume each flow vector is associated with only one motion pattern, i.e., no flow vector belongs to two different motion patterns. Thus, each motion pattern is a cluster of flow vectors. To make the proposed method applicable to general motions or activities, we make no assumption regarding the models of motion patterns. To cluster flow vectors, we first need to answer what is a motion pattern. According to the gestalt theory of humans' visual perception, the main factors used in grouping are proximity, similarity, closure, simplicity and common fate (elements with same moving direction are seen as a unit). According to these laws, a motion pattern should be smooth and the neighboring flow vectors should be similar and close. Note that even flow vectors in the same motion pattern may be very different. For example, the pattern of "U-turn" contains the flow vectors whose velocity are opposite. We use the neighborhood graph to measure the similarity and proximity of flow vectors. A summary of the proposed method is given in Fig. 1.

2. Motion Flow Field Generation

Given an input video, for each frame we compute sparse optical flows (instantaneous velocities) on inter-

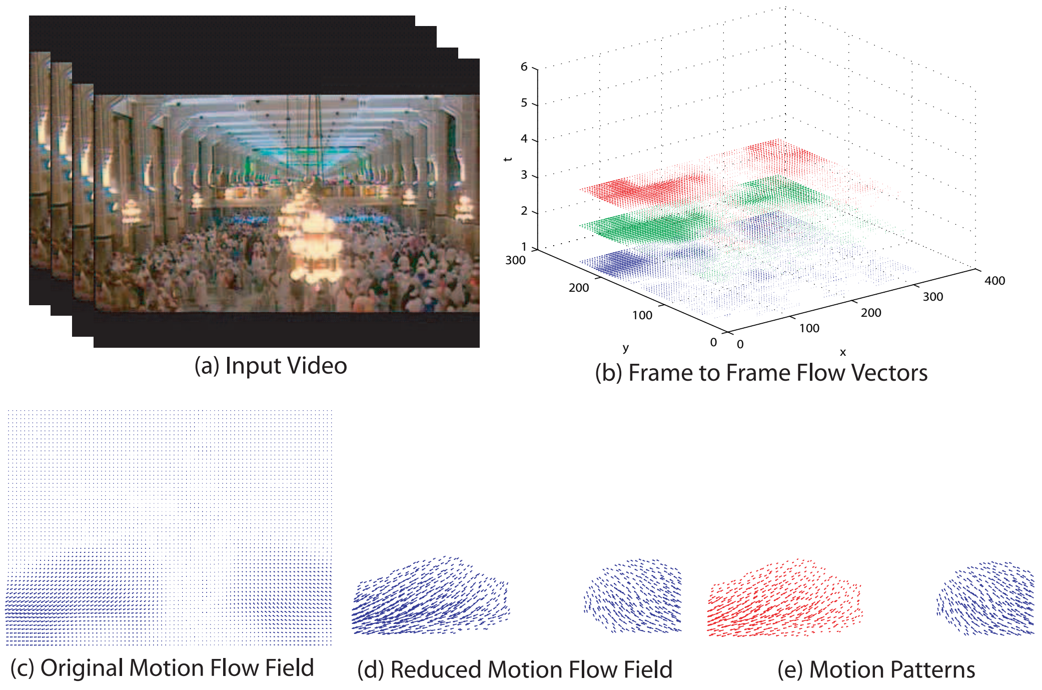

Figure 1. The main steps involved in proposed motion pattern detection method. Given an input video (a), frame-to-frame flow vectors (b), which are combined in a single global motion flow field (c). The original motion flow field (containing thousands of flow vectors) is reduced to the new motion flow field in (d) containing only hundreds of flow vectors. Finally typical motion patterns are detected (e)

est points ([4]) or dense optical flows for all pixels ([7]) in the image. The location $X_i = (x_i, y_i)$ and the velocity $V_i = (v_{x_i}, v_{y_i})$ of a point, i, in the image are then combined into a vector $p_i = (X_i, V_i)$. After flow vectors in all frames are computed, we gather them into a single global flow field. This flow field may contain thousands of flow vectors, and, therefore, it is computational expensive to obtain the shortest paths based on such a large number of data. These flow vectors always contain lots of redundant information and noise. We use Gaussian ART (the unsupervised version of Gaussion ARTMAP see [9]) to reduce the number of flow vectors from thousands to hundreds. The reduced number of flow vectors still maintain the geometric structure of the flow field without effecting detecting of motion patterns. Fig. 1-(d) shows the reduction results of the motion flow field in Fig. 1-(e). The number of flow vectors were reduced from 1417 to 744.

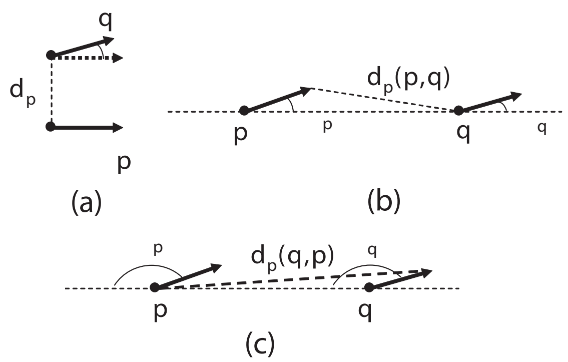

Figure 2. Distances in different cases: (a) flow vectors p and q are on parallel curves; (b,c) p and q are on the same curve, and q follows p. (b) shows the distance $D(p,q)$ where (c) shows the distance $D(q,p)$.

3. Neighborhood Graph of the Flow Field

To exploit the proximity of flow vectors and the geometric structure of high-dimensional data, we construct a neighborhood graph of flow vectors in the flow field. The general neighborhood graph is an undirected graph where edges connect neighboring points. The distance between a pair of points is then given by the "shortest path", i.e., geodesic distance, between them on the neighborhood graph. In the flow field, as described earlier each point is represented by a vector $p = (X, V)$ where $X = (x, y)$ is the location and $V = (v_x, v_y)$ is the velocity. To capture the spatial structure and direction of the flow field, we extend the original undirected neighborhood graph to the directed graph. The forward distance from flow vector p to q is defined as

$$D(p, q) = (d_p(p, q)d_s(p, q))^2 \tag{1}$$

The reason of using the square is given below. The spatial distance $d_p(p, q)$ and the directional difference $d_s(p, q)$ are defined by two hypotheses (see Fig. 2):

1. p and q are on two parallel curves. In this case,

$$d_p(p, q) = \|X_p - X_q\|, \tag{2}$$

$$d_s(p, q) = \left(\frac{2}{1 + \varepsilon + \overline{V_p} \cdot \overline{V_q}}\right)^2, \tag{3}$$

where $\varepsilon = 10^{-6}, \overline{V} = V/\|V\|$.

2. p and q are on the same curve, and q follows p. In this case,

$$d_p(p, q) = \|X_p + V_p - X_q\|, \tag{4}$$

$$d_s(p, q) = \frac{2}{1 + \varepsilon + \cos\theta_p} \cdot \frac{2}{1 + \varepsilon + \cos\theta_q} \tag{5}$$

$$\cos\theta_p = \overline{V_p} \cdot \overline{X_q - X_p}, \tag{6}$$

$$\cos\theta_q = \overline{V_q} \cdot \overline{X_q - X_p}. \tag{7}$$

The final distance is chosen as the minimum of the distances in the above two hypotheses. After the distance matrix is computed, we find the neighbors among the vectors to reduce the computational burden in future steps. B is a neighbor of A if and only if $D(A, B)$ equals the shortest path (obtained by Floyd's algorithm) from A to B. Because we used the square in computing $D(A, B)$, $D(A, B)$ is not necessarily less then $D(A, C) + D(C, B)$. If the directional difference factor were not included in $D(A, B)$, it is easy to see that when the angle $\angle ACB > \pi/2$, $D(A, B) > D(A, C) + D(C, B)$, and B is not a neighbor of A, because C is roughly in the middle. Note that if B is a neighbor of A, A is not necessarily a neighbor of B. If B is not neighbor of A, the edge from A to B is removed, i.e. $D(A, B)$ is set to infinity. This step removes many edges for future considerations.

4. Detecting Motion Patterns

In the flow field, each motion pattern consists of a group of flow vectors participating in the same process or motion. Detecting motion patterns is therefore the problem of clustering flow vectors. We use a hierarchical agglomerative clustering algorithm, i.e, single-linkage clustering algorithm, to cluster flow vectors. The complete detection algorithm is given below.

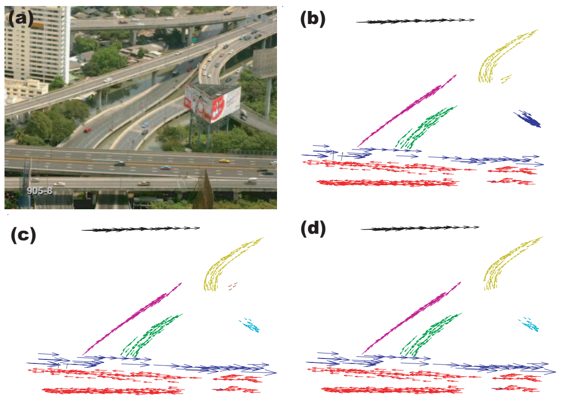

Figure 3. Hierarchical clustering based on the neighborhood graph: (a) one image of the input video, different clustering results: (b) 9 clusters, (c) 12 clusters, (d) 7 clusters with removed noise.

According to the law of the common fate, flow vectors with the similar directions should be considered as a unit. This leads to the idea of iteratively grouping the nearest neighbors into clusters using the agglomerative hierarchical clustering algorithms. One important reason we use the hierarchical clustering algorithm is that its computational complexity is polynomial in the number of edges in the neighborhood graph. Note that although the distance matrix computed in the previous step can identify the neighbors very efficiently and produce accurate results, it does not necessarily serve as a

Algorithm 1: Motion Pattern Detection Algorithm.

```
Input: a flow field $F$ of a set of flow vectors
       $\{p_1, p_2, ..., p_N\}$, and the number of clusters $n$
Output: a set of motion patterns $\{M_1, M_2, ...\}$.
1  Compute the distance matrix $D$ according to (1).
2  Compute the shortest path length $L$ using Floyd's
   method.
3  for $i \leftarrow 1$ to $N$ do
4      for $j \leftarrow 1$ to $N$ do
5          if $D(i, j) > L(i, j)$ then
6              $D(i, j) \leftarrow \infty$
7          end
8      end
9  end
10 Apply the hierarchical agglomerative clustering
   algorithm.
```

Algorithm 2: Hierarchical Agglomerative Clustering Algorithm

```
Input: Distance matrix (adjacent matrix of the
       neighborhood graph or geodesic distance matrix)
       with size $N \times N$ of a flow field, and $n$, the
       number of clusters.
Output: Output: $n$ clusters $\{G_1, G_2, ..., G_n\}$.
1 for $m \leftarrow N$ to $n$ do
2     Identify the two nearest points $p, q$ and combine $q$
      into $p$.
3     for $i \leftarrow 1$ to $N$ do
4         $D(p,i) \leftarrow min\{D(p,i), D(q,i)\}$,
5         $D(i,p) \leftarrow min\{D(i,p), D(i,q)\}$,
6         $D(q,i) \leftarrow \infty$,
7         $D(i,q) \leftarrow \infty$.
8     end
9 end
```

good measure for our clustering algorithm. The reason is that if both B and C are neighbors of A, we may tend to choose the one with similar direction to A regardless of the spatial distance. Therefore, we provide the option to modify the finite distances to the directional difference defined in (3).

The clustering algorithm works as follows. Initially every vector is treated as a cluster. We use a greedy approach to merge the clusters efficiently: in each step, we choose the shortest edge and merge the corresponding vectors (say p and q) by setting the distance $D(p,i)$ to $min\{D(p,i), D(q,i)\}$ and $D(i,p)$ to $min\{D(i,p), D(i,q)$ for each vector i and by removing the edges connecting q (i.e. setting $D(q,i) = D(i,q) = \infty$ for each i). When the number of clusters reaches a specified number n, we stop merging. Since the noisy flow vectors may result into independent clusters, we need to remove the clusters with very few vectors, and in our experiments we set n slightly higher than our desired number of clusters. To have a better control on the final number of clusters, we could start noise removal when the current number of clusters is close to the desired one. Fig. 3 shows the results with different numbers (9, 12) of clusters and after noise removal. From this figure, we can see that the clustering results are similar. The difference lies in some small groups (the size is less than 5) generated by increasing the number of clusters. These small groups will be deleted by noise removal step. After noise removal, seven clusters are detected (Fig. 3-(d)).

5. Experiments

We have tested the method on videos of some complex scenes. Fig. 4 shows the crossing on the Hong Kong street. In this video, people cross the road from two opposite sides and intersect in the middle part. Seen from the detected motion flow field, the intersection of their flow vectors make the clustering problem hard. By using the directed neighborhood graph, these intersecting flow vectors are correctly distinguished and the corresponding two major motion patterns are correctly detected. More results are shown in Fig. 5. The first row shows the results of a traffic scene. The vehicles on the two left lanes and those on the two right lanes move in opposite directions. Our method detects seven motion patterns. Four of them correspond to those in the ground truth. One of three is caused by a short side road on the right. Two extra clusters on the top are falsely classified due to the perspective distortions. The second row shows results of another traffic scene. Three motion patterns corresponding to three lanes in are detected. Note that when motions on two right lanes are merged together it is impossible and unnecessary to distinguish them. In the results shown in the third row, there are two typical motions. One is on the top of the image, people move in one direction in the building. In the other motion pattern, people move around a center in the image. This results in some false classification. The results for a more complex crowded scene are shown in the last row. People move in different directions.

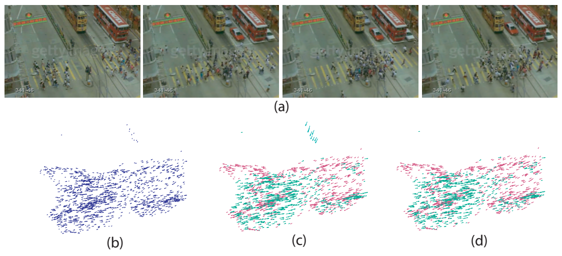

Figure 4. Motion pattern detection in a video of a street in Hong Kong. (a) Images of the input video (frame numbers are 1, 100, 180 and 248 respectively from left to right); (b) Motion flow field; (c) Detected motion patterns compared to (d) manually generated ground truth.

To measure the performance of our method, we compare the the results of our method with the ground truth on the test data. The ground truth is manually generated from the detected motion flow field. Table. 5 shows the comparison results on the number of motion patterns, miss-classification error.

6. Conclusions

In this paper, we propose a new method for learning typical motion patterns in challenging crowded scenes

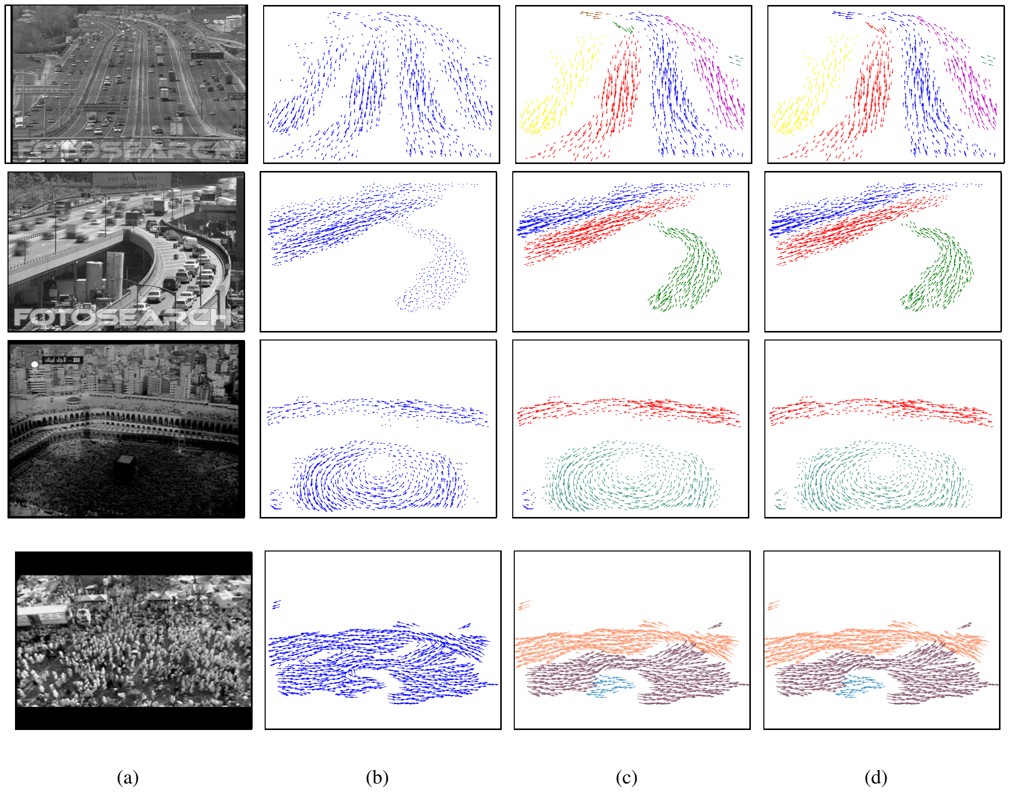

Figure 5. Motion pattern detection. (a) Input videos; (b) Motion flow field; (c) Detected motion patterns compared to (d) manually generated ground truth. Each row demonstrates the results for one video.

video	number of motion patterns		misclassification rate among flow vectors
	detected	ground truth	
high-way	7	5	2.4%
3-way traffic	3	3	0
Hajj005	3	2	2.4%
old street	5	7	5.3%
Hajj009	3	3	0
Hong Kong St	2	2	0.7%

Table 1. Comparison results between our method and the ground truth.

using the motion flow field representing the instantaneous motions in a video. The neighborhood graph is constructed to measure the similarity and proximity of flow vectors. The agglomerative clustering algorithm is applied to cluster flow vectors into motion patterns.

Acknowledgements: This research was funded by the US Government VACE program.

References

[1] F. M. Porikli, *Trajectory Pattern Detection by HMM Parameter Space Features and Eigenvector Clustering*, ECCV, 2004.

[2] D. Makris and T. Ellis, *Path Detection in Video Sequence*, IVC, Vol. 30, 2002.

[3] N. Johnson et al., *Learning the Distribution of Object Trajectories for Event Recognition*, IVC, 14, 1996.

[4] B. D. Lucas and T. Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*, IJCAI, 1981.

[5] X. Wang et al., *Learning Semantic Scene Models by Trajectory Analysis*, ECCV, 2006.

[6] N. Vaswani et al., *Activity Recognition Using the Dynamics of the Configuration of Interacting Objects*, CVPR, 2003.

[7] R. Gurka et al., *Computation of Pressure Distribution Using PIV Velocity Data*, Workshop on Particle Image Velocimetry, 1999.

[8] W. E. L. Grimson et al., *Using Adaptive Tracking to Classify and Monitor Activities in a Site*, CVPR, 1998.

[9] J. R. Williamson, *Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps*, Neural Netw., 1996.