Visual Analysis of Trajectory Clusters for Video Surveillance

Shantanu Singh* Raghu Machiraju* Richard Parent*

Department of Computer Science and Engineering
The Ohio State University

ABSTRACT

In this paper, we present an interactive trajectory analysis system designed for a surveillance setup. A codebook construction technique is used to cluster pedestrian trajectories. The codebooks provide a model which can be used for prediction, anomaly detection. and analysis of pedestrian behaviour.A graphical user interface is presented that allows an operator to perform these tasks interactively.

1 INTRODUCTION

With the increasing demands on homeland security, there has been a pressing need for the development of technologies to assist in the prevention and detection of attacks. The most significant step towards prevention and detection has been the increase in the level of vigilance in all forms. Video surveillance is a key tool that provides ubiquitous vigilance, enabling security personnel to safely monitor the most complex and dangerous of environments. However, even in the simplest of enviroments, a video surveillance operator may face an enormous information overload. There are often too many cameras, much fewer display screens, and even fewer personnel to monitor them. It is nearly impossible to monitor individual objects scattered across multiple views of the environment. It thus becomes imperative to extract higher level information from the video streams in order to make the surveillance task more feasible for the operator. The trajectories of moving objects extracted from a scene provide a reasonable high level information about the scene dynamics. By analyzing these trajectories, an operator can gain significant insight into the dynamics of the environment.

In this paper we focus on the analysis of pedestrian trajectories in large public spaces.A trajectory potentially encodes information about the behaviour of a person or group of persons in an environment. For example, in a parking lot scenario, one may infer the intent of a person through his trajectory - the motion of a person winding in and out of a row of cars might indicate suspicious activity. Such inferences can be used to further aid the operator in the surveillance task.

Our approach is to extract trajectory data from archived video footage of a scene, provide the operator with a framework to visualize the data, and aid the process of making inferences about the dynamics of the scene. We realize our goals by creating a codebook representation of trajectory data which then facilitates a prediction and anomaly detection mechanism. The model itself is further processed to present a summarized view of the dynamics of the scene at different spatial scales. The operator-driven analysis is realized through the use of a graphical user interface, which not only allows the operator to browse and drill down but also enables the detection of anomalous behavior and the learning of trends.

*e-mail: {singhsh,raghu,parent}@cse.ohio-state.edu

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, we provide an overview of the system with a schematic explaining the interaction of its components. Section 4 describes the processing modules. In Section 5, the analysis-visualization loop is explained. Experiments conducted with the system on a surveillance video are detailed in Section 6. Section 7 summarizes the conclusions of this work.

2 RELATED WORK

The goal of automating the task of video surveillance has been the focus of a significant portion of research in the field of computer vision. Much of the work has been in the direction of object detection [13] and tracking [14]. There has been recent work [11] [10] in the direction of analysing pedestrian trajectories in order to understand the activity in the scene.

Systems such as W4[7] and VSAM[2] attempt to provide a complete automated surveillance system, dealing with all stages of processing from the low-level operations of detection and tracking to extracting high-level information, such as pedestrian behaviour.There has been previous work towards creating a vizualization framework to support security operations [9]

The more general problem of using visualization techniques for clustering and anomaly detection has received much of attention [12, 8, 4]. The hierarchical clustering explorer [12] provides a technique to improve clustering of high dimensional data through user interacation. Similarly, a visual framework for clustering and anomaly detection has been proposed[4].

3 OVERVIEW

Fig 1 gives an overview of our visual anlysis system. The interaction between two components - the processing module and the analysis-visualization loop - is the mainstay of our system. The extraction of trajectories, their clustering and transitional graph analysis is carried out in the processing module. Trajectory data extracted from a video is clustered using a codebook generation technique described in the next section. The clusters obtained are then used to create a transitional graph describing activity scene and to provide a model for prediction and anomaly detection. The analysis-visualization loop is used interactively with the clusters, transitional graph, and prediction modules to gain insight about the trajectory data. This component is often employed in an iterative drill-down fashion by the user based on what he or she observes in the raw video or the analyses.

4 PROCESSING MODULES

4.1 Clustering by Codebook Construction

Clustering is central to analysis of information in our system. The input to the clustering algorithm are 2D trajectories extracted from video sequences. A trajectory Π is represented by a sequence of points in two dimensions.

$$\Pi = \langle p_1, p_2, \ldots, p_k \rangle \tag{1}$$

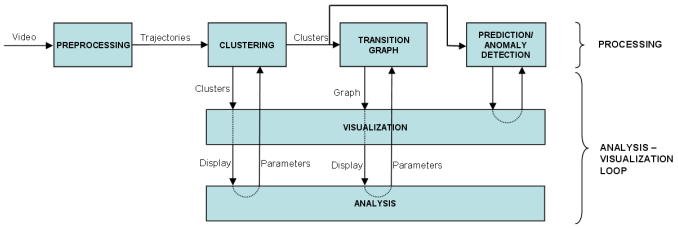


Figure 1: System Overview

The clustering algorithm constructs a codebook using a technique that is tantamount to performing vector quantization [6] in a vector space consisting of trajectories. It should be noted that the trajectories or vectors in this cluster space can be of unequal length. Labels or codes are accorded to clusters which in turn are composed of trajectories spatially proximate to each other. Our method is similar to what was reported in [10].

The key to any viable clustering algorithm are distance measures. Given, that the algorithms deal with points and trajectories (connected set of points), we define five specific measures.

Point-trajectory distance: The distance between a point p and a trajectory Π is given by:

$$d_1(\Pi, p) = \min_{1 \le i \le k} d_e(p, p_i) \tag{2}$$

where d_e is the point-wise Euclidean distance.

Asymmetric trajectory-trajectory distance: The asymmetric distance between two trajectories $\Pi_1 = \langle p_1, p_2, \ldots, p_{k_1} \rangle$ and $\Pi_2 = \langle q_1, q_2, \ldots, q_{k_2} \rangle$ is given by

$$d_2(\Pi_1, \Pi_2) = \max_{1 \le i \le k_2} d_1(\Pi_1, q_j) \tag{3}$$

Symmetric trajectory-trajectory distance: The symmetric distance between Π_1 and Π_2 is defined as:

$$d_3(\Pi_1, \Pi_2) = \max(d_2(\Pi_1, \Pi_2), d_2(\Pi_2, \Pi_1)) \tag{4}$$

It should be noted that the metric d_3 is also known as the Hausdorff distance.

Trajectories are clustered into codebooks. A codebook represents a group of trajectories that are close to each other by the d_3 distance measure. A codebook is represented by

$$\Omega = \langle c_1, c_2, \ldots, c_n \rangle \tag{5}$$

where each node, c_i is a 5-tuple defined by:

$$c_i = \langle p_i, \hat{p}_i, f_i, r_i, l_i \rangle \tag{6}$$

where, in turn, p_i is the position of a node, $\hat{p}_i$ is the normal unit vector at the node, f_i is the frequency with which the node has been updated, and r_i and l_i represent the the rightmost and leftmost points that have updated the node. A codebook can also be viewed as another 5-tuple, hence the notation

$$\Omega = \langle \Omega_c, \hat{\Omega}, F, \Omega_r, \Omega_l \rangle \tag{7}$$

where $\Omega_c = \langle p_1, p_2, \ldots, p_n \rangle$, $\hat{\Omega} = \langle \hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \rangle$, $F = \langle f_1, f_2, \ldots, f_n \rangle$, $\Omega_r = \langle r_1, r_2, \ldots, r_n \rangle$ and $\Omega_l = \langle l_1, l_2, \ldots, l_n \rangle$,

Trajectory-codebook distance: The distance between a trajectory Π and a codebook Ω is defined as

$$d_4(\Pi, \Omega) = \min(d_3(\Pi, \Omega_l), d_3(\Pi, \Omega_r)) \tag{8}$$

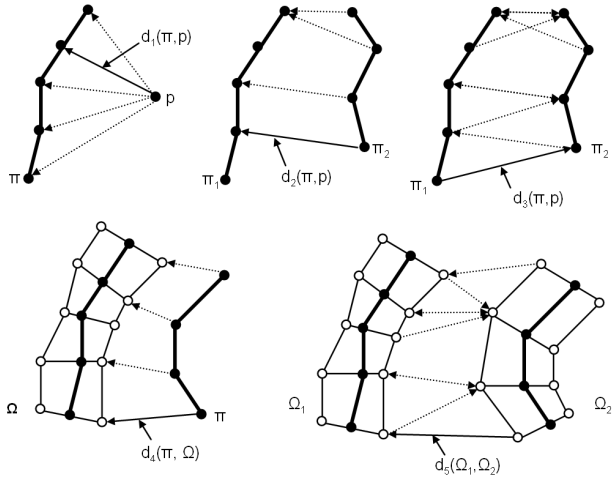


Figure 2: The five distance measures

This distance measure is used to check whether a trajectory falls within the cluster defined by a codebook.

Codebook-codebook distance: The distance between two codebooks Ω_1 and Ω_2 is defined as

$$d_5(\Omega_1, \Omega_2) = \min(d_4(\Omega_{1_l}, \Omega_2), d_4(\Omega_{1_r}, \Omega_2)) \tag{9}$$

The distance measure d_5 is used to decide whether two codebooks should be merged.The five distance metrics are illustrated in Fig. 2.

With these distances defined, we can now describe the algorithm for construction of codebooks. Algorithm 1 details the steps for the construction. In each iteration a new trajectory is used to either update the existing set of codebooks (C) or create a new codebook. C is initialized to the empty set (line 1). Thereafter the new trajectory is compared with all the codebooks in C and the closest matching codebook is found (lines 3-4).If the distance to the closest codebook is greater than a distance threshold d_{max}, or if the codebook set is empty, then a new codebook is added to C. The new codebook is initialized using the nodes of the trajectory (lines 6-11).

If the trajectory matches a codebook with a distance less that d_{max} then that codebook is updated using the nodes of the trajectory (lines 13-33). In the update process, each node of the trajectory is considered and the d_1 distance measure is used to find the closest matching node in the codebook. The matching codebook node is then moved towards the trajectory node, along the normal direction at the node, by a distance proportional to the weight of the codebook.

The clustering algorithm requires the computation of $\hat{\Omega}$ which is the ordered set of normals for each node of the codebook. To compute $\hat{\Omega}$, a cubic spline [5] is fit through the set of points in Ω_c and the normals at the nodes of the codebook are computed.

Once the codebooks are constructed, they undergo a second stage of clustering. A hierarchical clustering algorithm, explained in Algorithm 2, is used to merge similar codebooks in order to create a better clustering. The merge operation used in lines 8 and 10 is similar to the updating operation performed in Algorithm 1 lines 13-33. The codebook with the higher frequency is chosen as the reference. The lower frequency codebook is used to update it. Each node of the lower frequency codebook is iterated through and the node in the reference codebook that is closest to it is updated, just as in Algorithm 1 (lines 17-33).

Through this process of clustering the given set of trajectories, we have quantized the trajectory space and hence the physical do-

Algorithm 1 Construct codebook set C given trajectory set T

Require: $T = \{\Pi_1, \Pi_2, \ldots, \Pi_N\}$
1: $C \leftarrow \phi$
2: **for** $i = 1$ to N **do**
3: $d^* \leftarrow \min\{d_4(\Pi_i, \Omega) : \Omega \in C\}$
4: Let Ω^* be the codebook such that $d^* = d_4(\Pi_i, \Omega^*)$
5: **if** $d^* > d_{max}$ or $C = \phi$ **then**
6: /* *Create a new codebook* */
7: $\Omega_c \leftarrow \Pi_i$, $\Omega_l \leftarrow \Pi_i$, $\Omega_r \leftarrow \Pi_i$
8: Compute $\hat{\Omega}$ from Ω_c
9: $F \leftarrow \langle \underbrace{1, 1, \ldots, 1}_{n-times} \rangle$ where $n = |\Pi_i|$
10: $\Omega_{new} \leftarrow \langle \Omega_c, \hat{\Omega}, F, \Omega_r, \Omega_l \rangle$
11: $C \leftarrow C \cup \Omega_{new}$
12: **else**
13: /* *Update the codebook* Ω^* */
14: Let $\Pi_i = \langle q_1, q_2, \ldots, q_M \rangle$
15: Let $\Omega_c^* = \langle p_1, p_2, \ldots, p_K \rangle$
16: **for** $j = 1$ to M **do**
17: Let k^* be the index such that $d_e(p_{k^*}, q_j) = d_1(\Omega_c^*, q_j)$
18: /* *Project* q_j *onto the normal at* p_{k^*} *and update* p_{k^*} */
19: $q'_j \leftarrow (q_j - p_{k^*}).\hat{p}_{k^*} + p_{k^*}$
20: $p_{k^*} \leftarrow \frac{f_{j^*} p_{j^*} + q'_i}{f_{j^*}+1}$
21: /* *Update* l_{k^*} *and* r_{k^*} *using* q'_j */
22: **if** $d_e(q'_j, r_{j^*}) < d_e(q'_j, l_{j^*})$ **then**
23: **if** $\|p_{k^*} - q'_j\| + \|q'_j - r_{k^*}\| \geq \|p_{k^*} - r_{k^*}\|$ **then**
24: $r_{k^*} \leftarrow q'_j$
25: **end if**
26: **else**
27: **if** $\|p_{k^*} - q'_j\| + \|q'_j - l_{k^*}\| \geq \|p_{k^*} - l_{k^*}\|$ **then**
28: $l_{k^*} \leftarrow q'_j$
29: **end if**
30: **end if**
31: $f_{j^*} \leftarrow f_{j^*} + 1$
32: **end for**
33: Recompute $\hat{\Omega}^*$ from Ω_c^*
34: **end if**
35: **end for**

Algorithm 2 Hierarchically cluster the codebook set C

Require: $C = \{\Omega_1, \Omega_2, \ldots, \Omega_L\}$
1: $updated \leftarrow true$
2: **while** $updated$ **do**
3: $updated \leftarrow false$
4: $d^* \leftarrow \min\{d_5(\Omega_i, \Omega_j) : \Omega_i, \Omega_j \in C\}$
5: Let $\Omega_{i^*}, \Omega_{j^*}$ be the codebooks such that $d^* = d_5(\Omega_{i^*}, \Omega_{j^*})$
6: **if** $d^* < d_{max}$ **then**
7: **if** $sum(F_{i^*}) < sum(F_{j^*})$ **then**
8: $\Omega_{new} \leftarrow merge(\Omega_{i^*}, \Omega_{j^*})$
9: **else**
10: $\Omega_{new} \leftarrow merge(\Omega_{j^*}, \Omega_{i^*})$
11: **end if**
12: $C \leftarrow (C - \{\Omega_{i^*}, \Omega_{j^*}\}) \cup \Omega_{new}$
13: $updated \leftarrow true$
14: **end if**
15: **end while**

main under scrutiny into a set of sub-regions or codebooks. The codebooks provide a model of the trajectories which are used for prediction/anomaly detection and construction of a transition graph, as described in the subsequent sections.

4.2 Prediction and Anomaly Detection

If the frequencies of the codebooks are normalized, i.e., if we divide the frequency of each codebook by the sum of frequencies of all codebooks, then the codebook set approximates the density of the distribution of trajectories. The codebooks can then be used for anomaly detection as well as prediction. To detect an anomaly we simply check if the closest codebook is greater or lesser than d_{max}, which the distance threshold used in Algorithm 1

ANOMALY DETECTION:
1: $d^* \leftarrow \min\{d_4(\Pi, \Omega) : \Omega \in C\}$
2: **if** $d^* > d_{max}$ **then**
3: Π is anomalous
4: **else**
5: Π is normal
6: **end if**

where C is the codebook set.

Similarly, to predict the course of a trajectory as it evolves we check at each step the closest codebook that it matches.

PREDICTION:
1: Let Ω^* be the codebook such that $d^* = d_4(\Pi, \Omega^*)$
2: Ω^* defines the the bounds for the predicted trajectory.

4.3 Transition Graph

Algorithm 3 describes the steps to create a transition graph Ψ from the set of codebooks C. The algorithm works as follows. The codebook frequencies are accumulated into $2D$ spatial bins (lines 4-11). A threshold (specified by the user through the Analysis-Visualization loop) is used to prune bins with low frequency. A node in the graph corresponds to a $2D$ bin. An edge is added between two nodes when a codebook passes through the two corresponding bins (lines 15-30). A node is assigned the frequency of its corresponding bin.

The transition graph provides an understanding of the pedestrian traffic through a sequence of transitions. The nodes represent the points of transition. The edges represent the movement from one node to another. The edges incident on a node indicate directions of pedestrian movement from that location.

5 ANALYSIS-VISUALIZATION LOOP

The Analysis-Visualization loop, realised through a graphical user interface, provides the operator with mechanism to interact with the processing modules. The GUI is shown in Fig. 3. The interaction with each of the modules is described below.

1. **Codebook Generation** The quality of the clusters generated depend upon the parameter d_{max} which is the distance threshold in the codebook construction algorithm. A good clustering is characterized by the well-separatedness of codebooks, which can be checked by visual inspection. The GUI allows the operator to specify this parameter and visualize the clustering. The codebooks can then be filtered based on their frequency. By setting a threshold, the operator can specify what percentage, by frequency, of the codebooks are displayed.

2. **Prediction and Anomaly detection** The operator is allowed to load existing trajectory data or synthesize a trajectory by drawing it on the screen . When the operator draws the trajectory, the system displays the best matching codebook and

updates the display at every step of user input. By constructing a synthetic trajectory, the operator can gain an insight into the trajectories that are flagged as anomalies by the system.

3. **Transition Graph construction** The construction of the transition graph requires binning of the 2D space. The operator can specify the number of bins for the two dimensions. A threshold value can also be specified which filters the bins prior to the construction of the graph. Bins with a normalized frequency above the threshold are considered when constructing the graph. Both the bins sizes and the threshold control the resolution of the graph, and thereby the information that can be gained from it. By specifying small bin sizes and a low threshold, the algorithm produces a graph with edges of short length, describing pedestrian traffic with high specificity. To get a less granular view of the traffic, higher values for bin sizes and threshold can be set.

Algorithm 3 Construct Ψ from C

Require: $C = \{\Omega_1, \Omega_2, \ldots, \Omega_L\}$
/*n_x and n_y are the number of bins in the x and y dimensions. w and h are the width and height of the view of the scene*/
declare $W[1..n_x][1..n_y]$ /*W is array of bins in the 2D space.*/
initialize $W[i][j] \leftarrow 0$ for $1 \leq i \leq n_x, 1 \leq j \leq n_y$
for $i = 1$ to L **do**
  Let $\Omega_i = \langle c_1, c_2, \ldots, c_K \rangle$
  for $j = 1$ to K **do**
    Let $c_j = \langle p_j, \hat{p}_j, f_j, r_j, l_j \rangle$
    $row \leftarrow \left\lceil n_x \frac{p_j^x}{w} \right\rceil$; $col \leftarrow \left\lceil n_y \frac{p_j^y}{h} \right\rceil$
    $W[row][col] \leftarrow W[row][col] + f_j$
  end for
end for
$W \leftarrow \frac{W}{sum(W)}$ /* Normalize W */
declare $G \leftarrow \phi$
/*G is adjacency list of the transition graph*/
for $i = 1$ to L **do**
  Let $\Omega_i = \langle c_1, c_2, \ldots, c_K \rangle$
  for $j = 1$ to K **do**
    Let $c_j = \langle p_j, \hat{p}_j, f_j, r_j, l_j \rangle$
    $row \leftarrow \left\lceil n_x \frac{p_j^x}{w} \right\rceil$; $col \leftarrow \left\lceil n_y \frac{p_j^y}{h} \right\rceil$
    if $j = 1$ **then**
      $node_{prev} = (row, col)$
    else
      if $W[row][col] > t$ **then**
        $node_{cur} \leftarrow (row, col)$
        $G \leftarrow G \cup (node_{prev}, node_{cur})$
        $node_{prev} \leftarrow node_{cur}$
      end if
    end if
  end for
end for

6 Experiments

We describe a setup to show how the system can be operated in order to gain an understanding of pedestrian traffic by analysing trajectory data. The analysis-visualization loop is carried out by the operator, with the GUI providing the interface between the operator and the processing units.

The data used for this experiment is from a security camera video mounted on a university building. The video is 10 minutes in length and is captured at a 320x240 resolution. The preprocessing step involves the extraction of pedestrian trajectories, which was done using a mean shift tracker [3], followed by manual correction of the tracks. The extracted trajectories are shown in Fig. 4.

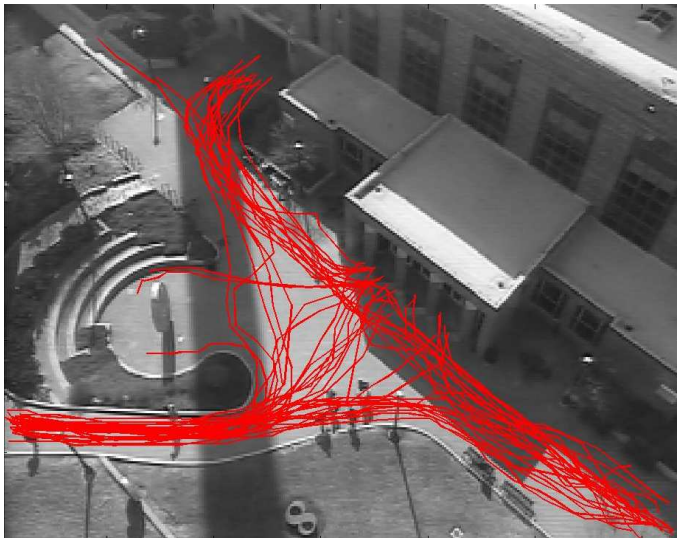


Figure 4: Pedestrian trajectories extracted from surveillance video

The operator first clusters the trajectories using the codebook construction algorithm. The algorithm requires the specification of the distance threshold d_{max}, which controls the quality of clustering. The operator can arrive at a good clustering (characterized by well separated clusters) by visualizing the codebooks generated for different values of d_{max}. In the experiment, codebooks were generated with values of d_{max} in the range 15 to 30 in steps of 1. Codebooks generated with values 15, 25 and 30 are shown in Fig. 5. We see that $d_{max} = 15$ generates clusters that are cluttered. By filtering through the codebooks generated with $d_{max} = 30$, we find that the codebooks begin to get distorted, as seen in Fig. 5(d). This is because with a high threshold, trajectories with different orientations update the same codebook, resulting in a codebook with distortions. Codebooks generated with $d_{max} = 25$ give a well separated set of trajectory clusters.

Once the codebooks are generated, the operator can construct a transition graph to visualize pedestrian traffic in the scene. The transition graph shows how pedestrians transit across the environment, as described previously. Construction of the graph also requires the specification of some parameters. The operator can adjust the bin size and the weight threshold until the graph provides an uncluttered view of pedestrian transitions. By setting a higher threshold on the frequency of the bins, the operator can view the entry -exit pairs, since most or all of the intermediate nodes fall below the threshold. The entry-exit pairs are the pairs of locations where the pedestrians make an entry into, or exit from, the scene.

In the experiment, we specify the number of bins along the rows and the columns to be 20, and the threshold for the bin weights to be 0. The graph generated is seen in Fig.6(a). The size of the nodes are proportional to the weight of the bins they correspond to. The graph in Fig.6(a) provides a high resolution view of the pedestrian traffic. In order to see the entry-exit pairs, we set a higher threshold. Fig.6(b) shows the entry-exit pairs in the scene.

Finally, the operator the prediction and anomaly detection mechanism to analyse the motion of a single pedestrian. A stored pedestrian trajectory can be loaded and checked for the existence of an anomaly. Alternatively, a trajectory can be synthesized by the operator by drawing on the screen, and the prediction mechanism can

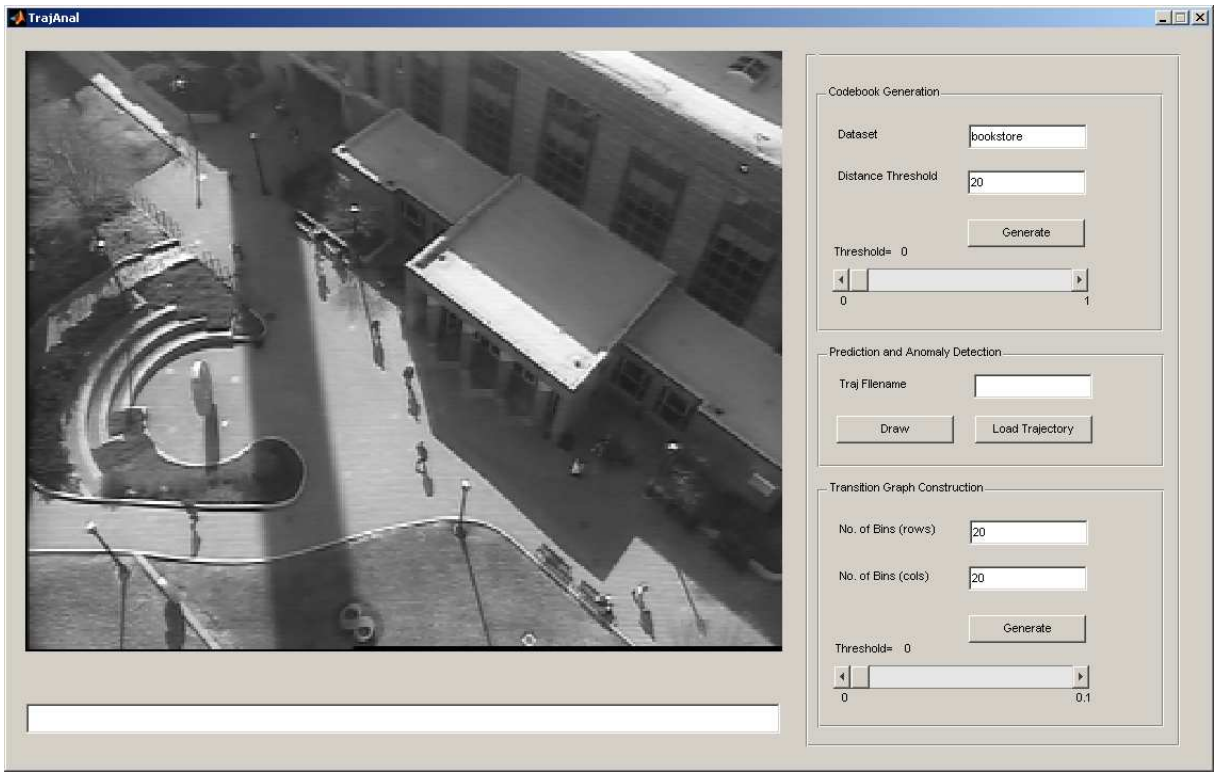


Figure 3: Graphical User Interface

be used to view the probable trajectories.

To experiment with this mode, we first load two sample trajectories from a database into the anomaly detection module. The trajectory in Fig.7(a) is a normal trajectory, which the detection mechanism recognizes. The closest matching codebook gets displayed. Fig.7(b) shows an anomalous trajectory which gets highlighted in red.

We now synthesize a trajectory by drawing it on the screen. Fig 8 shows the successive stages of the prediction. In Fig. 8(a)-(c) the trajectory matches existing codebooks, and the bounds of the codebook provide a prediction of the trajectory. In Fig. 8(d), the trajectory has gone ”out of course”, and does not match any of the codebooks, and hence gets tagged as anomalous.

Through these experiments we see that by providing an interactive GUI, the operator can work with the processing modules to visualize the behaviour of pedestrians in the scene, as evinced by their trajectories.

7 Conclusions and Future Work

We presented a system to analyse pedestrian behaviour suited for a surveillance environment. The system processes pedestrian trajectory data and generates a clustering by constructing codebooks. The codebooks are used to predict trajectories and generate a graph that depicts pedestrian movement. The codebooks themselves can be displayed to provide a summary of the pedestrian traffic. By simulating a trajectory through the scene, the operator can understand pedestrian behaviour in the scene through the predicted codebooks that are displayed at each stage of the input. Anomalous trajectories can also be identified in a similar way.

The system provides a method to analyze pedestrian trajectories, which is one of the many inputs to a surveillance system. In order to better serve surveillance needs, a system that integrates further information, such as time of day, pedestrian pose, etc. would facilitate a more robust analysis framework. Nevetheless, the visual analysis framework presented in this paper can provide a basis for the design of such systems.

8 Acknowledgements

We thank Dr. J. W. Davis, Mr. P. Maughan and Dr. D. Woods for their valuable comments on the initial ideas.This research was supported in part by the National Science Foundation under grant IIS-042824.

References

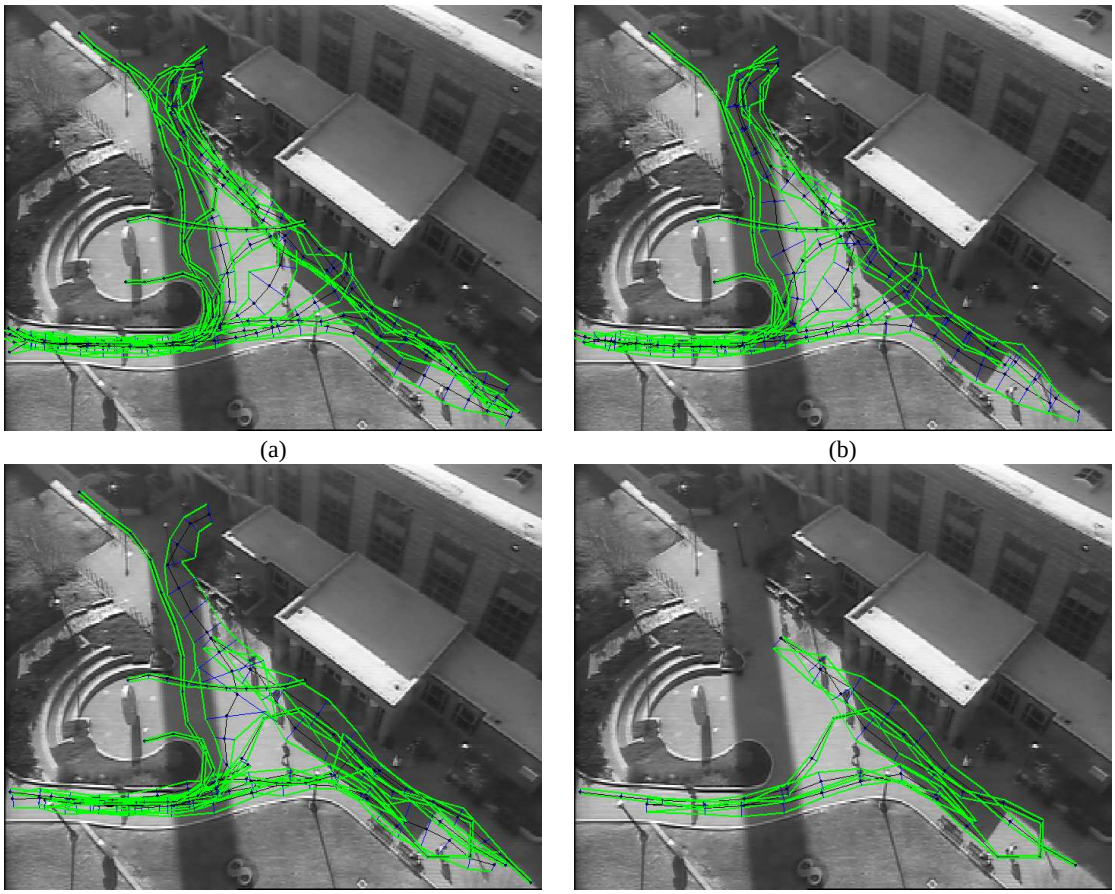
[1] J. Alon, S. Sclaro, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR)*, 2003.

[2] Robert Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, and Osamu Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000.

[3] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1197, Washington, DC, USA, 1999. IEEE Computer Society.

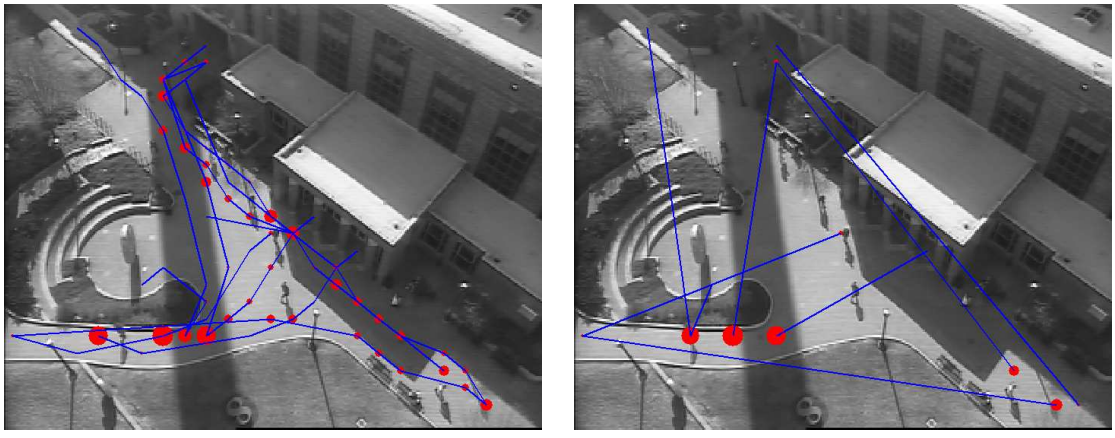
[4] I. Davidson and M. Ward. A particle visualization framework for clustering and anomaly detection. In *Proc. KDD Workshop on Visual Data Mining*, Sept. 2001.

[5] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics (2nd ed. in C): principles and practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.



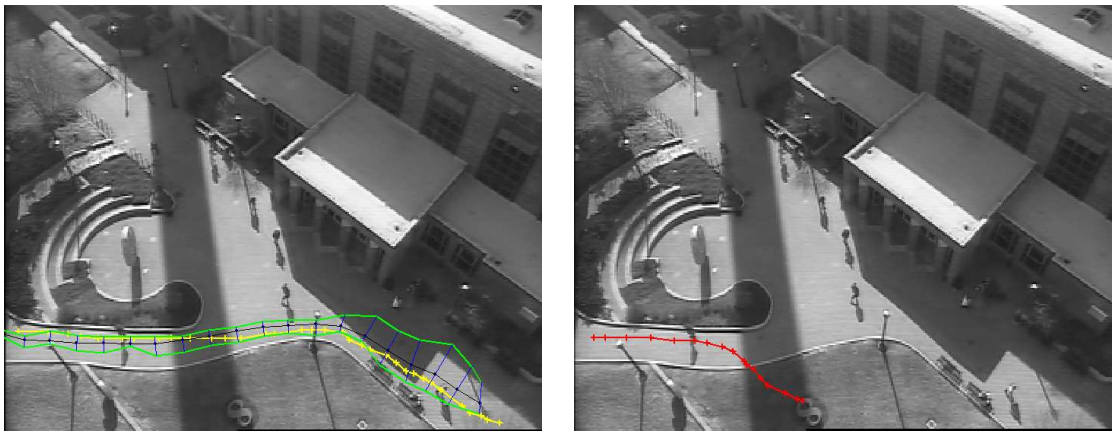
(a) (b) (c) (d)

Figure 5: Codebooks generated with (a) $d_{max} = 15$ (b) $d_{max} = 25$ (c) $d_{max} = 30$ (d)Distortions produced with $d_{max} = 30$



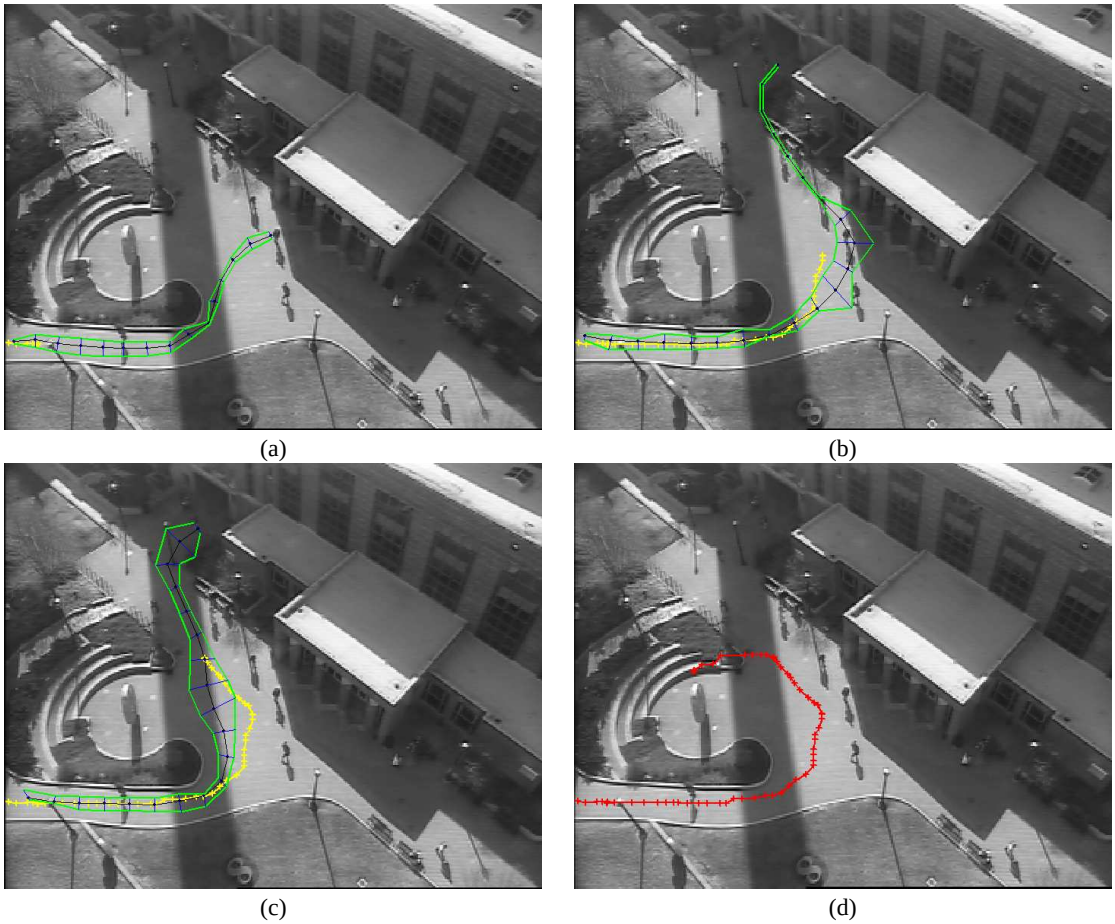
(a) (b)

Figure 6: Transition graphs generated with bin size 20x20 and applying threshold= (a)0 (b)0.03



(a) (b)

Figure 7: Anomaly detection module: (a) A normal trajectory. The closest codebook is displayed (b) Anomalous trajectory



(a) (b)

(c) (d)

Figure 8: (a) Start drawing (b),(c) Updated prediction (d)Anomaly identified

[6] R. Gray. Vector quantization. In *IEEE Signal Processing Magazine*, pages 4– 29. IEEE Computer Society, 1984.

[7] Ismail Haritaoglu, David Harwood, and Larry S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):809–830, 2000.

[8] Alfred Inselberg and Tova Avidan. Classification and visualization for high-dimensional data. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 370–374, New York, NY, USA, 2000. ACM Press.

[9] D.B. Koch. 3d visualization to support airport security operations. In *Aerospace and Electronic Systems Magazine, IEEE*, volume 19, pages 23–28, June 2004.

[10] D. Makris and T. Ellis. Finding paths in video sequences. In *British Machine Vision Conference 2001, Manchester, UK*, 2001.

[11] C. Piciarelli, G.L. Foresti, and L. Snidaro. Trajectory clustering and its applications for video surveillance. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 40–45, 2005.

[12] Jinwook Seo and Ben Shneiderman. Interactively exploring hierarchical clustering results. volume 35, pages 80–86, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[13] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Mitsubishi Electric Research Lab Technical Report. TR-2003-90 (2003)*, 2003.

[14] T. Zhao and R. Nevatia. 3d tracking of human locomotion: a tracking as recognition approach. In *International Conference on Pattern Recognition*, pages I: 546–551, 2002.