Visual Analysis of Trajectory Clusters for Video Surveillance

Shantanu Singh\textsuperscript{*} \quad Raghu Machiraju\textsuperscript{*} \quad Richard Parent\textsuperscript{*}

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 behavior. 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 environments, 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 behavior 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.

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 analyzing 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 behavior. There has been previous work towards creating a visualization 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 interaction. Similarly, a visual framework for clustering and anomaly detection has been proposed[4].

3 OVERVIEW

Fig 1 gives an overview of our visual analysis 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 \( \Pi \) is represented by a sequence of points in two dimensions.

\[
\Pi = \langle p_1, p_2, \ldots, p_k \rangle
\]
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 \( \Omega \) is given by:

\[
d_1(\Pi, p) = \min_{1 \leq i \leq k} d_2(p, p_i)
\]

where \( d_2 \) is the point-wise Euclidean distance.

**Asymmetric trajectory-trajectory distance:** The asymmetric distance between two trajectories \( \Pi_1 = \{p_1, p_2, \ldots, p_k\} \) and \( \Pi_2 = \{q_1, q_2, \ldots, q_k\} \) is given by

\[
d_2(\Pi_1, \Pi_2) = \max_{1 \leq i \leq k} d_1(\Pi_1, q_j)
\]

**Symmetric trajectory-trajectory distance:** The symmetric distance between \( \Pi_1 \) and \( \Pi_2 \) is defined as:

\[
d_3(\Pi_1, \Pi_2) = \max \{d_2(\Pi_1, \Pi_2), d_2(\Pi_2, \Pi_1)\}
\]

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

\[
\Omega = \{c_1, c_2, \ldots, c_k\}
\]

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

\[
c_i = \{p_i, \hat{p}_i, f_i, r_i, l_i\}
\]

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

\[
\Omega = \{\Omega_r, \hat{\Omega}, F, \Omega_w, \Omega_t\}
\]

where \( \Omega_r = \{p_1, p_2, \ldots, p_n\} \), \( \hat{\Omega} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n\} \), \( F = \{f_1, f_2, \ldots, f_n\} \), \( \Omega_w = \{r_1, r_2, \ldots, r_n\} \) and \( \Omega_t = \{l_1, l_2, \ldots, l_n\} \).

**Trajectory-codebook distance**: The distance between a trajectory \( \Pi \) and a codebook \( \Omega \) is defined as

\[
d_4(\Pi, \Omega) = \min(d_3(\Pi, \Omega_r), d_3(\Pi, \Omega_t))
\]
Algorithm 1 Construct codebook set C given trajectory set T

Require: $T = \{\Pi_1, \Pi_2, \ldots, \Pi_k\}$

1: $C \leftarrow \emptyset$

2: for $i = 1$ to $N$ do
3: $d^* \leftarrow \min \{d_4(\Pi_i, \Omega) : \Omega \in C\}$
4: Let $\Omega^*$ be the codebook such that $d^* = d_4(\Pi_i, \Omega^*)$
5: if $d^* > d_{max}$ or $C = \emptyset$ then
6: /* Create a new codebook */
7: $\Omega_i \leftarrow \Pi_i$, $\Omega_j \leftarrow \Pi_j$, $\Omega_r \leftarrow \Pi_r$
8: Compute $\Omega$ from $\Omega_i$
9: $F \leftarrow \{1, 1, \ldots, 1\}$ where $n = |\Pi_i|$
10: $\Omega_{new} \leftarrow (\Omega_i, \Omega_r, \Omega_r)$
11: $C \leftarrow C \cup \Omega_{new}$
12: else
13: /* Update the codebook $\Omega^*$ */
14: Let $\Pi_i = \langle q_1, q_2, \ldots, q_m \rangle$
15: Let $\Omega^*_i = \langle p_1, p_2, \ldots, p_k \rangle$
16: for $j = 1$ to $M$ do
17: Let $k^*$ be the index such that $d_4(p_{k^*}, q_j) = d_4(\Omega^*_i, q_j)$
18: /* Project $q_j$ onto the normal at $p_{k^*}$ and update $p_{k^*}$ */
19: $q'_j \leftarrow (q_j - p_{k^*}) \cdot p_{k^*} + p_{k^*}$
20: $p_{k^*} \leftarrow \text{line}
21: /* Update $l_{k^*}$ and $r_k$ using $q'_j$ */
22: if $d_4(q'_j, l_{k^*}) < d_4(q'_j, l_{k^*})$ then
23: if $\|p_{k^*} - q'_j\|^2 + \|q'_j - l_{k^*}\| \geq \|p_{k^*} - r_{k^*}\|$ then
24: $r_{k^*} \leftarrow q'_j$
25: end if
26: else
27: if $\|p_{k^*} - q'_j\|^2 + \|q'_j - l_{k^*}\| \geq \|p_{k^*} - l_{k^*}\|$ then
28: $l_{k^*} \leftarrow q'_j$
29: end if
30: end if
31: $f_{k^*} \leftarrow f_{k^*} + 1$
32: end for
33: Recompute $\Omega^*$ from $\Omega^*_i$
34: end if
35: end for

Algorithm 2 Hierarchically cluster the codebook set C

Require: $C = \{\Omega_1, \Omega_2, \ldots, \Omega_k\}$

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 $\text{sum}(F_j) < \text{sum}(F_j)$ then
8: $\Omega_{new} \leftarrow \text{merge}(\Omega_i, \Omega_j)$
9: else
10: $\Omega_{new} \leftarrow \text{merge}(\Omega_i, \Omega_j)$
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 is the distance threshold used in Algorithm 1.

**ANOMALY DETECTION:**

1. Let $d^* = d_4(\Pi, \Omega^*)$
2. $\Omega^*$ defines the bounds for the predicted trajectory.

4.3 Transition Graph

Algorithm 3 describes the steps to create a transition graph $\Psi$ 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
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.

Algorith 3 Construct $\Psi$ from $C$

**Require:** $C = \{\Omega_1, \Omega_2, \ldots, \Omega_k\}$

1. /* $n_x$ 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*/
2. declare $W[1..n_y,1..n_x]$, /*$W$ is array of bins in the 2D space.*/
3. initialize $W[i][j] \leftarrow 0$ for $1 \leq i \leq n_y, 1 \leq j \leq n_x$
4. for $i = 1$ to $L$ do
5.  Let $\Omega_i = (c_1, c_2, \ldots, c_K)$
6.  for $j = 1$ to $K$ do
7.    $c_j = (p_j, p_j, f_j, r_j, l_j)$
8.    $row \leftarrow \frac{p_j}{w}; col \leftarrow \frac{f_j}{h}$
10. end for
11. end for
12. $W \leftarrow \frac{\sum_{i=1}^{L} W[i][j]}{L}$ /* Normalize $W*/
13. declare $G \leftarrow \phi$
14. /*$G$ is adjacency list of the transition graph*/
15. for $i = 1$ to $L$ do
16.  Let $\Omega_i = (c_1, c_2, \ldots, c_K)$
17.  for $j = 1$ to $K$ do
18.    $\Omega_j = (c_1, c_2, \ldots, c_K)$
19.    $row \leftarrow \frac{p_j}{w}; col \leftarrow \frac{f_j}{h}$
20.    if $j = 1$ then
21.      $node_{prev} = (row, col)$
22.    else
23.      if $W[row][col] > \tau$ then
24.        $node_{cur} \leftarrow (row, col)$
25.        $G \leftarrow G \cup (node_{prev}, node_{cur})$
26.        $node_{prev} \leftarrow node_{cur}$
27.      end if
28.    end if
29.  end for
30. 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.
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. Nevertheless, 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-0428224.

References

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

Figure 6: Transition graphs generated with bin size 20x20 and applying threshold= (a) 0 (b) 0.03
Figure 7: Anomaly detection module: (a) A normal trajectory. The closest codebook is displayed (b) Anomalous trajectory

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


