Realtime Anomaly Detection using Trajectory-level Crowd Behavior Learning

Aniket Bera
University of North Carolina
Chapel Hill, NC, USA
ab@cs.unc.edu

Sujeong Kim
SRI International
Princeton, NJ, USA
sujeong.kim@sri.com

Dinesh Manocha
University of North Carolina
Chapel Hill, NC, USA
dm@cs.unc.edu

Abstract

We present an algorithm for realtime anomaly detection in low to medium density crowd videos using trajectory-level behavior learning. Our formulation combines online tracking algorithms from computer vision, non-linear pedestrian motion models from crowd simulation, and Bayesian learning techniques to automatically compute the trajectory-level pedestrian behaviors for each agent in the video. These learned behaviors are used to segment the trajectories and motions of different pedestrians or agents and detect anomalies. We demonstrate the interactive performance on the PETS 2016 ARENA dataset as well as indoor and outdoor crowd video benchmarks consisting of tens of human agents.

Website: *http://gamma.cs.unc.edu/RCrowdT/Anomaly/*

1. Introduction

There has been a growing interest in developing computational methodologies for simulating and analyzing the movements and behaviors of crowds in real-world videos. This include simulation of large crowds composed of a large number of pedestrians or agents, moving in a shared space, and interacting with each other. Some of the driving applications include surveillance, training systems, robotics, navigation, computer games, and urban planning.

In this paper, we deal with the problem of interactive anomaly detection in crowd videos and develop approaches that perform no precomputation or offline learning. Our research is motivated by the widespread use of commodity cameras that are increasingly used for surveillance and monitoring, including sporting events, public places, religious and political gatherings, etc. One of the key challenges is to devise methods that can automatically analyze the behavior and movement patterns in crowd videos to detect anomalous or atypical behaviors [10]. Furthermore, many of these applications desire interactive or realtime performance, and do not rely on apriori learning or label-

Figure 1: Our method extracts trajectories and computes pedestrian movement features at interactive rates. We use the learned behavior and movement features to detect anomalies in the pedestrian trajectories. The lines indicate behavior features (explained in detail in Section 3.5). The yellow lines indicate anomalies detected by our approach.

ing. Many algorithms have been designed to track individual agents and/or to recognize their behavior and movements and detect abnormal behaviors) [6]. However, current methods are typically limited to sparse crowds or are designed for offline or non-realtime applications.

We present an algorithm for realtime anomaly detection in low to medium density crowd videos. Our approach uses online methods to track each pedestrian and learn the trajectory-level behaviors for each agent by combining nonlinear motion models and Bayesian learning. Given a video stream, we extract the trajectory of each agent using a realtime multi-person tracking algorithm that can model different interactions between the pedestrians and the obstacles. Next, we use a Bayesian inference technique to compute the trajectory behavior feature for each agent. These trajectory behavior features are used for anomaly detection in terms of pedestrian movement or behaviors. Our approach involves no offline learning and can be used for interactive surveillance and any crowd videos. We have implemented our system on a multi-core PC and have applied it to both indoor and outdoor crowd videos containing up to tens of pedestrians. We are able to compute crowd agents' trajectories and behavioral features in less than a tenth of a second.

2. Related Work

There is extensive research in computer vision and multimedia analyzing crowd behaviors and movements from videos [10]. Most of the work has focused on extracting useful information including behavior patterns and situations for surveillance analysis through activity recognition and abnormal behavior detection. Certain methods focus on classifying the most common, simple behavior patterns (linear, radial, etc.) in a given scene. However, most of these methods are designed for offline applications and tend to use a large number of training videos for offline learning of patterns for detecting common crowd behavior patterns [16], normal and abnormal interactions [11], human group activities [13]. Other methods are designed for crowd analysis using a large number of web videos [14]. However, these techniques employ either manual selection methods or offline learning techniques for behavior analysis and therefore, cannot be used for interactive applications. Other methods are based on low-density tracking data to learn agent intentions [12] or pre-processing techniques that decompose crowd scenes into main agents and background agents [17]. All of these methods perform offline computations, and it is not clear whether they can be directly used for interactive applications.

3. Trajectory Behavior Learning

In this section, we present our interactive trajectory-level behavior computation algorithm.

3.1. Terminology and Notation

We first introduce the notation used in the remainder of the paper.

Pedestrians: We use the term *pedestrian* to refer to independent individuals or human-like agents in the crowd. Their trajectories and movements are extracted by our algorithm using a realtime multi-person tracker.

State representation: A key aspect of our approach is to compute the state of each pedestrian and each pedestrian cluster in the crowd video. Intuitively, the state corresponds to the low-level motion features that are used to compute the trajectory-level behavior features. In the remainder of the paper, we assume that all the agents are moving on a 2D plane. Realtime tracking of pedestrians is performed in the 2D image space and provides an approximate position, i.e. (x, y) coordinates, of each pedestrian for each frame of the video. In addition, we infer the velocities and intermediate goal positions of each pedestrian from the sequence of its prior trajectory locations. We encode this information regarding a pedestrian's movement at a time instance using a state vector. In particular, we use the vector $\mathbf{x} = [\mathbf{p}\ \mathbf{v}\ \mathbf{g}]^{\mathbf{T}}$, $\mathbf{x} \in \mathbb{R}^6$ to refer to a pedestrian's state. The state vector consists of three 2-dimensional vectors: $\mathbf{p}$ is the pedestrian's position, $\mathbf{v}$ is its current velocity, and $\mathbf{g}$ is the intermediate goal position. The intermediate goal position is used to compute the optimal velocity that the pedestrian would have taken had there been no other pedestrians or obstacles in the scene. As a result, the goal position provides information about the pedestrian's immediate intent. In practice, this locally optimal velocity tends to be different from $\mathbf{v}$ for a given pedestrian. The state of the entire crowd, which consists of individual pedestrians, is the union of the set of each pedestrian's state $\mathbf{X} = \bigcup_i \mathbf{x_i}$.

Pedestrian behavior feature: The pedestrians in a crowd are typically in motion, and their individual trajectories change as a function of time. The behavior of the crowd can be defined using macroscopic or global flow features, or based on the gestures and actions of different pedestrians in the crowd. In this paper, we restrict ourselves to trajectory-level behaviors or movement features per agent and per cluster, including current position, average velocity (including speed and direction), cluster flow, and the intermediate goal position. These features change dynamically. Our goal is to interactively compute these features from tracked trajectories, and then use them for behavior analysis.

3.2. Overview

Our overall approach consists of multiple components: a real-time multi-person tracker, state estimation, and behavior feature learning. One of our approach's benefits and its difference from prior approaches is that our approach does not require offline training using large number of training

examples. As a result, it can be directly applied to any new or distinct crowd video. We extend our behavior learning and pedestrian tracking pipeline from [3, 7]. Fig. 3 highlights these components. The input into our algorithm is one frame of real-world crowd video at a time, and our goal is to compute these behavior features for each agent from these frames. An adaptive multi-person or pedestrian tracker is used to compute the observed position of each pedestrian on a 2D plane, denoted as $(\mathbf{z}_0 \cdots \mathbf{z}_t)$. Furthermore, we use new state estimation and behavior-learning algorithms that can also compensate for the tracking noise and perform robust behavior analysis.

We do not make any assumptions about the dynamics or the actual velocity of each agent in the crowd. Since we do not know the dynamics or true state of each agent, we estimate its state $\mathbf{x}$ from the recent observations for each pedestrian. We use a Bayesian inference technique to estimate the most likely state of each pedestrian in an online manner and thereby compute the state of the overall crowd, $\mathbf{X}$. Based on estimated real crowd states, we compute the trajectory behavior feature of each agent. These features are grouped together to analyze the behavior or movement patterns, and are also used for various training and surveillance applications.

Interactive State Computation: We use an online approach that is based on the current and recent states of each pedestrian. In other words, it does not require future knowledge or future state information for any agent. Because we estimate the state during each frame, our formulation can capture the local and global behavior or the intent of each agent.

3.3. Realtime Multi-person Tracking

Our approach uses a realtime multi-person tracking algorithm to extract the pedestrian trajectories from the video. There is considerable research in computer vision literature on online or realtime tracking. In our case, any online tracker that requires the knowledge of a specific pedestrian motion model can be used. In particular, we use particle filters as the underlying algorithm for multi-person tracking. The particle filter is a parametric method that solves non-Gaussian and non-linear state estimation problems. Particle filters are frequently used in object tracking because they can recover from lost tracks and occlusions. The particle tracker's tracking uncertainty is represented in a Markovian manner by only considering information from present and past frames.

To reliably estimate the motion trajectory in a dense crowd setting, we use RVO (reciprocal velocity obstacle) [18] – a local collision-avoidance and navigation algorithm – as the non-linear motion model. For more details we direct our readers to [3, 2, 4]

Each agent is represented as a 2D circle in the plane, and the parameters used for motion estimation for each agent consist of the representative circle's radius (which fully and completely encloses the vertical project of the pedestrian), maximum speed, neighbor distance, and local time horizon. The RVO algorithm assumes that each pedestrian is fully aware of the current position and velocity of other nearby agents. Given each agent's state at a particular time-step, the RVO algorithm computes a collision-free state for the next time-step.

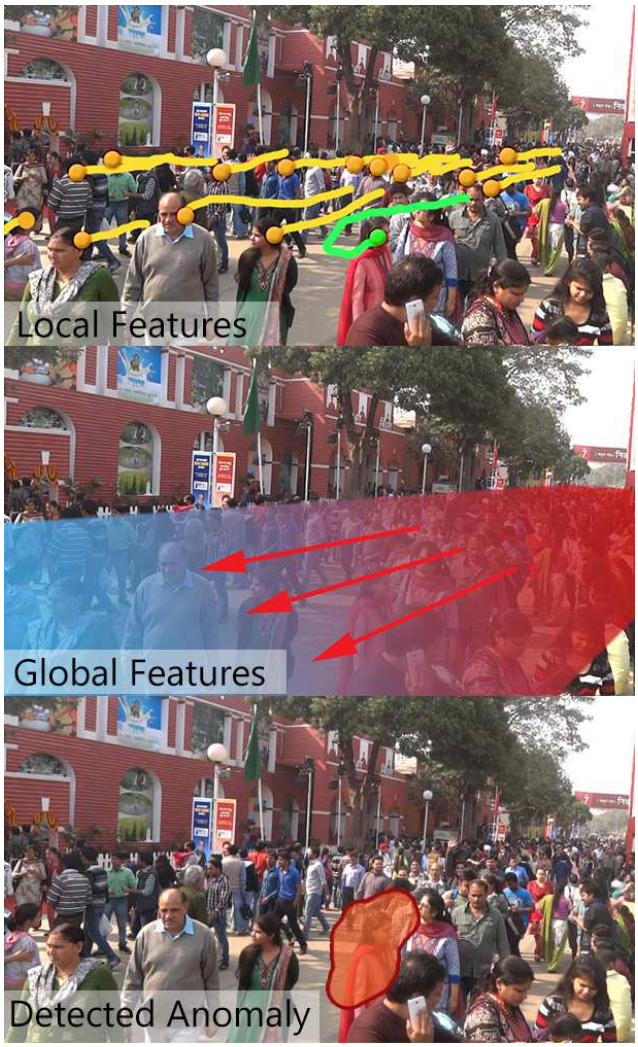


Figure 2: The **Anomaly Detection** In this example, we see that one pedestrian (marked with green) suddenly makes a U-turn (local feature) in a crowd where everyone is walking in a specific direction/field (global feature). Our system detects this as an anomaly (Refer to Section 3.8).

3.4. State estimation

A key component of our approach is to estimate the state of each agent based on the tracked data. We estimate the

state of each real agent and the cluster (Refer Section 3.6) it is part of during each frame. The state estimation is performed in the world-space coordinates by transforming the observations, $\mathbf{z}_t \in \mathbb{R}^2$, from the multi-person tracker output, which is in image-space coordinates. In this way, we are able to minimize the camera distortion error in the trajectory, which eventually improves the accuracy of our local navigation motion model. Moreover, the state information computed for each agent can then be used in for different applications.

We use the Ensemble Kalman Filter (EnKF), which is an extension of Kalman Filtering, to compute the most likely state of each agent, $\mathbf{x_t}$, based on previous observations, $(\mathbf{z}_1, \cdots \mathbf{z}_t)$. Each agent's state constitutes a part of the overall crowd state $\mathbf{X}_t$. Per-agent inferencing permits us to easily accommodate the entering and leaving agents in the environment, which is important in dynamic scenarios (moving obstacles, changing pedestrian behavior). The crowd state and interactions among pedestrians are still approximated by our state-transition model, m_t. In our case, we again use the RVO as the motion model m_t for local navigation. EnKF predicts the next state based on the transition motion model and Q_t. When a new observation becomes available, R_t is updated based on the difference between the observations and the prediction, which is then used to compute the state of the real pedestrian.

3.5. Local Pedestrian Behavior Feature Extraction

The state estimation provides the position, velocity, and intermediate goal position for each agent at a given time. Based on the series of states, we compute the trajectory behavior features, which describe the past and future trajectory characteristics at the current location.

The pedestrian trajectory behavior feature describes the characteristics of each agent during a certain time window corresponding to the last w seconds. The behavior feature vector consists of the current position, the average velocity during the time window, and the intermediate goal of an agent. We encode the connection between the recent velocity, v^{avg}, of a pedestrian and the intermediate goal position, g, at the current position. We denote the local behavior feature vector, $\mathbf{b^l}$, which is a six-dimensional vector, as follows:

$$\mathbf{b^l} = [\mathbf{p}\ \mathbf{v}^{avg}\ \mathbf{g}]^T, \tag{1}$$

where $\mathbf{p}$, $\mathbf{v}^{avg}$, and $\mathbf{g}$ are each two-dimensional vectors that represent the current position, average velocity during the time window $t - w$ through t, and the estimated intermediate goal position computed as part of state estimation, respectively.

The duration of the time window is typically set based on the characteristics of a scene and the underlying application. Small time windows are effective in terms of capturing details in dynamically changing scenes with many rapid velocity changes that are caused by some agents moving quickly. Larger time windows, which tend to smooth out abrupt changes in motion, are more suitable for scenes that have fewer changes in pedestrian movement. In our case, we maintain the time window between 0.5 and 1.0 seconds in our applications.

3.6. Global Pedestrian Behavior Feature Extraction

To capture the essence of a pedestrian behavior in a crowd, we need to capture both the individual level movement features and also the dynamics of the group or cluster of which it is a part. Our approach computes global movement flows of pedestrians in semi-dense to dense settings (the importance of global features increases when the density increases). It is not uncommon for some nearby pedestrians to have similar trajectories. As a result, we compute clusters of pedestrians in a crowd based on their positions, velocity, inter-pedestrian distance, orientations, etc. We initially assign each pedestrian to a separate cluster, one consisting of a single pedestrian. We then merge these clusters by analyzing their relative velocities and their geometric proximity, which is a function of the Euclidean distance between the clusters, the speed of each agent, and their motion. In our experiments, we found that a bottom-up approach is more efficient than a top-down approach for crowds composed of small clusters.

We compute a connectivity graph among the pedestrians. There is an edge between vertices of this graph if and only if the two pedestrians are together for some period of time and their velocities are close to each other. The density of this graph helps us define intra-cluster proximity. Eventually, all the behavior features vectors (as explained in Section 3.5) of every cluster are computed in the same way they are computed per agent. This corresponds to the global pedestrian features $\mathbf{b}^g$ to predict their global movement. This clustering divides the crowd into subsections with similar characteristics.

In the last section, we presented an algorithm to compute the trajectory level behavior features at each time step from a given video. These trajectory level features can be used for different applications related to crowd scene analysis. In this section, we highlight the use of these features and characteristics to detect anomalies based on motion segmentation. Our approach does not use any offline learning methods, and is based on unsupervised classification methods. We highlight their performance on many challenging scenarios.

3.7. Motion segmentation

The goal behind motion segmentation is to clearly classify variations of pedestrian behaviors in a crowd. Our trajectory-level behavior features can also be used for mo-

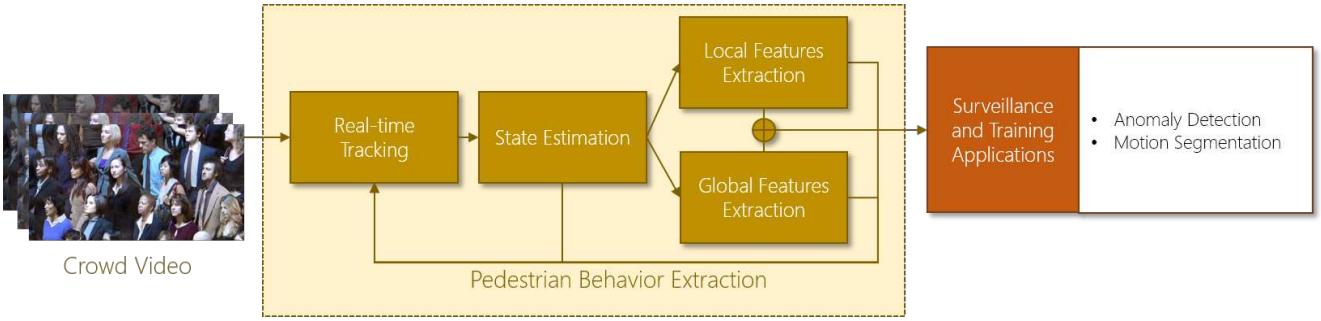


Figure 3: **Overview of our approach.** We highlight the different stages of our interactive algorithm: tracking, pedestrian state estimation, and behavior learning. The local and global features refer to individual vs. overall crowd motion features. These computations are performed at realtime rates for each input frame.

tion pattern segmentation. Typically, motion pattern segmentation techniques segment spatial regions on an image/video based on the similarity of the pedestrians' movement patterns.

Flow-based methods are often used to segment crowd movements in videos [1]. These techniques mostly work well for structured scenes. Coherent filtering [19] uses tracklets instead of trajectories; thus, it can accommodate unstructured scenarios. Meta-tracking [5] tracks sets of particles and is effective for unstructured scenarios with high density crowds. See, for example, [10]. In terms of segmentation results, our method yields similar results as meta-tracking in terms of handling both structured and unstructured scenarios with low or high densities.

We use the K-means data-clustering algorithm to group the trajectories' behavior features observed during a certain time window. Because we are focused on temporal local behavior analysis, we discard the data observed before a particular threshold time or earlier frames. We classify these features into K groups of flows, which we call behavior clusters. K and N are user-defined values that represent the total number of the clusters and the total number of collected behavior features, respectively, and $K \leq N$. A set of behavior clusters $B = \{B_1, B_2, ..., B_K\}$ is computed as follows:

$$\underset{B}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\mathbf{b}_i \in B_k} dist(\mathbf{b}_i, \mu_k), \tag{2}$$

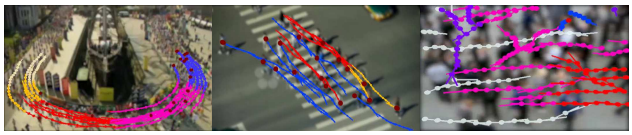
where $\mathbf{b}_i$ is a behavior feature vector, μ_k is a centroid of each cluster, and $dist(b_i, \mu_k)$ is a distance measured between the arguments. Further details about the behavior feature extraction and classification can be found in [8].

In our case, the distance between two pedestrian feature vectors is computed as

$$\begin{aligned} dist(\mathbf{b}_i, \mathbf{b}_j) = & \, c_1 \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ & + c_2 \left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} wdt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} wdt) \right\| \\ & + c_3 \left\| \mathbf{g}_i - \mathbf{g}_j \right\|, \end{aligned} \tag{3}$$

which corresponds to the weighted sum of the distance between three points: current positions, previous positions, and future positions. c_1, c_2, and c_3 are the weights.

Each behavior cluster is visualized with eight different colors based on the direction of the velocity components of its centroid. Fig. 4 shows the segmentation examples in structured, unstructured, and highly unstructured videos. For the Marathon video, we show that the segmentation from the sparse samples matches the behavior patterns of entire crowds. In terms of computation, our algorithm takes only tens of milliseconds for clustering computation during each frame.



(a) Marathon [1] (b) Crossing [15] (c) 879-38 [14]

Figure 4: **Motion segmentation of structured and unstructured scenarios:** Different colors indicate clusters grouped by similarity of behavior or movement features at interactive rates. We use eight discrete colors for visualization of the results in these benchmarks.

3.8. Anomaly detection

Anomaly detection is an important problem that has been the focus of research in diverse research areas and applications. It corresponds to the identification of pedestrians, events, or observations that do not conform to an expected pattern or to other pedestrians in a crowd dataset. Typically, the detection of anomalous items or agents can lead to improved automatic surveillance. Anomaly detection can be categorized into two classes based on the scale of the behavior that is being extracted [9]: global anomaly detection and local anomaly detection. A global anomaly typically

Video	Density	Total Frames	BLT
ARENA (01_01)	Low	1060	0.002
ARENA (01_02)	Low	890	0.004
ARENA (03_05)	Low	1440	0.002
ARENA (03_06)	Low	1174	0.002
ARENA (06_01)	Low	2941	0.001
ARENA (06_04)	Low	1582	0.002
ARENA (08_02)	Low	792	0.002
ARENA (08_03)	Low	746	0.006
ARENA (10_03)	Low	1173	0.002
ARENA (10_04)	Low	1188	0.005
ARENA (10_05)	Low	894	0.004
ARENA (11_03)	Low	329	0.001
ARENA (11_04)	Low	729	0.002
ARENA (11_05)	Low	666	0.002
ARENA (14_01)	Low	1081	0.001
ARENA (14_03)	Low	1242	0.004
ARENA (14_05)	Low	1509	0.001
ARENA (14_06)	Low	857	0.002
ARENA (14_07)	Low	1312	0.004
ARENA (15_02)	Low	917	0.004
ARENA (15_05)	Low	903	0.004
ARENA (15_06)	Low	660	0.001
ARENA (22_01)	Low	2079	0.002
ARENA (22_02)	Low	1006	0.001
ARENA (23_01)	Low	712	0.001
Crossing	Medium	238	0.03
Marathon	High	450	0.02
879-38	High	349	0.01
UCSD-Peds1-Cart	Low	200	0.004
UCSD-Peds1-Biker	Low	200	0.009
IITF-5	High	876	0.0512
NPLC-1	Low	775	0.012
NDLS-1	High	941	0.049

Table 1: Performance of trajectory level behavior learning on a single core for different benchmarks: We highlight the number of frames of extracted trajectories, the time spent in learning pedestrian behaviors (BLT - Behavior Learning Time (in sec)). Our learning and trajectory computation algorithms demonstrate interactive performance on these complex crowd scene analysis scenarios.

affects a large portion of, if not the entire, crowd and local anomaly is limited to an individual scale (for example, individuals moving against the crowd flow). We primarily use our trajectory-based behavior characteristics for local anomaly detection. In other words, we detect a few behaviors that are rare and are only observed in the video during certain periods. These periods can be as long as the length of the video or as short as a few hundred frames. In other words, we classify an anomaly as temporally uncommon behavior. For example, a person's behavior going against the flow of crowds may be detected as an anomaly at one point, but the same motion may not be detected as an anomaly later in the frame if many other pedestrians are moving in the same direction.

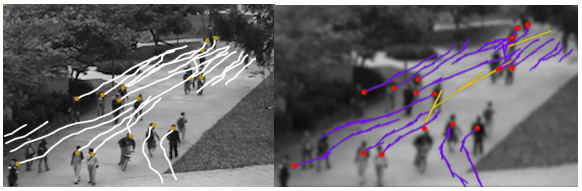
For anomaly detection we compare the distance between the local and global pedestrian features of every pedestrian (computed using equation 3). When an anomaly appears in a scene, the anomaly features typically tend to be isolated in the cluster of which it is a part. In other words, the pedestrian's motion will be different from that of the surrounding crowd. If the Euclidean distance between the *global* and *local* feature (refer Sections 3.5 and 3.6) is more than a threshold value, we classify it as an anomaly.

$$dist(\mathbf{b}^l, \mathbf{b}^g) > Threshold \tag{4}$$

This threshold is a user-tunable parameter. If this threshold is set low, the sensitivity of the anomaly detection will increase and vice-versa.

4. Quantitative Results

We compare the accuracy of our motion segmentation and anomaly detection methods using the quantitative metrics presented in Table 1 and Table V, as described in Li et al. [10]. Table 1 in [10] provides a true detection rate for motion pattern segmentation. It is based on the criterion that the approach successfully detected the regions containing the moving pedestrians. Although we cannot directly compare the numbers with pixel-based performance measures, MOTP values (Table 1) can be an indirect measure for the true detection rate motion segmentation. Compared to the values range of **0.4-1.0** in [15], the corresponding values computed by our approach are in the range of **0.7-0.8** in terms of detecting moving pedestrians, even for unstructured videos. These numbers indicate that the performance of our method is comparable to the state of the art.



(a) (b)

Figure 5: **Anomaly Detection**. We also evaluate other datasets like UCSD. Trajectories of 63 real pedestrians are extracted from a video. One person in the middle walks against the flow of crowd. Our method can capture the anomaly of this pedestrian's behavior or movement by comparing the behavior features with those of other pedestrians.

Fig. 5 shows the results of anomaly detection in different crowd videos. **879-38 video dataset [14]**: The trajectories of 63 pedestrians are extracted from the video. One

Reference	Dataset	Performance				
		Area under ROC Curve	*Accuracy*	*DR*	*Equal Error Rate*	*Online/Offline*
Our Method	UCSD	**0.873**	**85%**	**-**	**20%**	**Online**
Wang 2012		0.9	-	85%	-	Offline
Cong 2013		0.86	-	-	23.9	Offline
Cong 2012		0.98-0.47	46%	46%	20%	Offline
Thida 2013		0.977	-		17.8%	Offline
Our Method	879-44	**0.97**	**80%**	**-**	**13%**	**Online**
Our Method	ARENA	**0.91**	**76%**	**-**	-	**Online**

Table 2: Comparison of Anomaly Detection techniques. All the reference methods have been explained in detail in [10]. Our method has comparable results with the state of the art offline methods in anomaly detection.

Video Name	Camera ID	Threat Level
11_03	TRK_RGB_1	High
15_02	TRK_RGB_1	High
22_02	ENV_RGB_3	High
14_06	TRK_RGB_1	Medium
15_06	TRK_RGB_1	Medium
14_07	TRK_RGB_1	Low
10_04	TRK_RGB_1	Low
06_01	TRK_RGB_1	Low
10_05	TRK_RGB_1	Low

Table 3: Details of the anomalies detected in the ARENA Dataset.

person in the middle is walking against the flow of pedestrians through a dense crowd. Our method can distinguish the unique behavior of this pedestrian by comparing its behavior features with those found by methods. In **UCSD-Peds1-Biker** and **UCSD-Peds1-Cart** benchmarks, our method is able to distinguish parts of the trajectories of the biker and the cart because their speeds were noticeably different from those of other pedestrians.

Apart from ARENA, we evaluated the accuracy of the anomaly detection algorithm on the UCSD PEDS1 dataset [11] and compared it with Table V in Li et al. [10] in Table 2.

Our method successfully detected the following anomalies in the ARENA - *Person checking vehicle*, *different motion pattern*, *person on a bike*, *push and run*, *abnormal motion near vehicle*, *man touching vehicle*, *hit and run*, *suddenly people running* and *possible mugging*.

5. Conclusion and Future Work

We present an interactive approach for computing trajectory level behavior features from crowd videos and demonstrate its use in surveillance and training applications. Our approach is general, can handle moderately dense crowd videos, and can compute the trajectory and movement behavior for each agent during each time step. A key benefit of our approach is that it can capture dynamically changing movement behaviors of pedestrians and thereby be used for dynamic or local behavior analysis.

Limitations: The performance and accuracy of our algorithm is governed by the tracking algorithm, which can be noisy, sparse, or may lose tracks. Furthermore, current realtime methods may not work well in very dense crowd videos, e.g., those with thousands of agents in a single frame. Our online learning algorithm is useful only for capturing local pedestrian trajectory characteristics, whereas offline learning methods can compute many global characteristics. For example, our anomaly detection and motion segmentation algorithms will only capture unique/rare behaviors observed in temporally adjacent frames.

Future Work: There are many avenues for future work. In addition to overcoming the limitations of our work, we would like to combine the characteristics of pedestrian dynamics with other techniques that can model complex crowd behaviors. We would like to extend them for intelligent surveillance applications and also to predict future crowd states.

References

[1] S. Ali and M. Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1–6, June 2007.

[2] A. Bera, S. Kim, and D. Manocha. Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In *Graphics Interface*, 2015.

[3] A. Bera and D. Manocha. Realtime multilevel crowd tracking using reciprocal velocity obstacles. In *Proceedings of Conference on Pattern Recognition, Sweden*, 2014.

[4] A. Bera and D. Manocha. Reach-realtime crowd tracking using a hybrid motion model. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 740–747. IEEE, 2015.

[5] P.-M. Jodoin, Y. Benezeth, and Y. Wang. Meta-tracking for video scene understanding. In *Advanced Video and Signal Based Surveillance, IEEE International Conference on*, pages 1–6, Aug 2013.

[6] S. J. Junior et al. Crowd analysis using computer vision techniques. *IEEE Signal Processing Magazine*, 27(5):66–77, 2010.

[7] S. Kim, A. Bera, A. Best, R. Chabra, and D. Manocha. Interactive and adaptive data-driven crowd simulation. In *Virtual Reality*. IEEE, 2016.

[8] S. Kim, A. Bera, and D. Manocha. Interactive crowd content generation and analysis using trajectory-level behavior learning. In *International Symposium on Multimedia*. IEEE, 2015.

[9] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1446–1453. IEEE, 2009.

[10] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan. Crowded scene analysis: A survey. *Circuits and Systems*

for Video Technology, IEEE Transactions on, 25(3):367–386, March 2015.

[11] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1975–1981, 2010.

[12] S. R. Musse, C. R. Jung, J. C. S. Jacques, and A. Braun. Using computer vision to simulate the motion of virtual agents. *Computer Animation and Virtual Worlds*, 18(2):83–93, 2007.

[13] B. Ni, S. Yan, and A. Kassim. Recognizing human group activities with localized causalities. In *Computer Vision and Pattern Recognition, 2009. IEEE Conference on*, pages 1470–1477. IEEE, 2009.

[14] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert. Data-driven crowd analysis in videos. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1235–1242, Nov 2011.

[15] J. Shao, C. Loy, and X. Wang. Scene-independent group profiling in crowd. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2227–2234, June 2014.

[16] B. Solmaz, B. E. Moore, and M. Shah. Identifying behaviors in crowd scenes using stability analysis for dynamical systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10):2064–2070, 2012.

[17] L. Sun, X. Li, and W. Qin. Simulating realistic crowd based on agent trajectories. *Computer Animation and Virtual Worlds*, 24(3-4):165–172, 2013.

[18] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*. 2011.

[19] B. Zhou, X. Tang, and X. Wang. Coherent filtering: Detecting coherent motions from crowd clutters. In *Proc. of the European Conference on Computer Vision - Part II*, pages 857–871, 2012.