Collective Dynamics

Modeling Trajectory-level Behaviors using Time Varying Pedestrian Movement Dynamics

Aniket Bera · Sujeong Kim · Dinesh Manocha

University of North Carolina at Chapel Hill, North Carolina, USA,
E-mail: ab@cs.unc.edu, sujeong@cs.unc.edu, dm@cs.unc.edu

Received: 6 February 2018 / Accepted: 7 May 2018
DOI: 10.17815/CD.2018.15

Abstract We present a novel interactive multi-agent simulation algorithm to model pedestrian movement dynamics. We use statistical techniques to compute the movement patterns and motion dynamics from 2D trajectories extracted from crowd videos. Our formulation extracts the dynamic behavior features of real-world agents and uses them to learn movement characteristics on the fly. The learned behaviors are used to generate plausible trajectories of virtual agents as well as for long-term pedestrian trajectory prediction. Our approach can be integrated with any trajectory extraction method, including manual tracking, sensors, and online tracking methods. We highlight the benefits of our approach on many indoor and outdoor scenarios with noisy, sparsely sampled trajectory in terms of trajectory prediction and data-driven pedestrian simulation.

Keywords Pedestrians · crowds · behavior learning · pedestrian dynamics · multi-agent

1 Introduction

The modeling of pedestrian movement dynamics has received considerable attention in multiple fields, including computer-aided design, urban planning, robotics, and evacuation planning. In many of these applications, the goal is to generate trajectories and behaviors of virtual pedestrians that are similar to those observed of humans in real-world environments. The most common approaches used to model pedestrian and crowd movement are based on agent-based models that treat individuals as autonomous agents who can perceive the environment to make independent decisions about their behavior or movement. Agent-based methods have been well studied in different fields for decades and various formulations have been proposed for global and local navigation. However, current approaches are unable to simulate the dynamic nature, variety, and subtle aspects of real-world pedestrian motions.

Advances in sensor (e.g., camera) technologies have made it possible to easily capture videos of pedestrian and crowd motion. Such videos are widely available on the Internet (e.g., YouTube). It is possible to use computer vision methods to extract trajectories of pedestrians from these videos. These trajectories correspond to the location of each pedestrian on the walking plane as a time-dependent function. There is considerable interest in utilizing these real-world trajectories to learn pedestrian behaviors and use them for different applications. In particular, there is a new class of algorithms, called data-driven pedestrian [1–4] or crowd simulation, in which such real-world trajectories are used for simulating the pedestrians in a synthetic environment.

However, current techniques available to extract trajectories from videos have many limitations and the use of such data-driven methods for pedestrian simulation is therefore restricted. The accuracy of video tracking methods varies with the number of pedestrians and crowd density as well as the video resolution and the illumination conditions. As a result, use of these methods is currently limited to isolated pedestrians or sparse crowds. Many behavior learning algorithms require a high number of training videos to learn the movement patterns offline, and typically extract a fixed set of parameters that are used as a global characterization of pedestrian behavior or trajectories; thus, they may not be suited to capturing the dynamic nature or time-varying behaviors of pedestrians that are observed in the real-world scenes.

Main Results:

In this paper, we present statistical algorithms to learn the characteristics of pedestrian movement from trajectories extracted from real videos. These characteristics are used to compute collision-free trajectories of virtual pedestrians whose movement patterns resemble those of pedestrians in the original video. Our approach is automatic and interactive and captures the dynamically changing movement behaviors of real pedestrians. We demonstrate its applications for many data-driven crowd simulations, where we can easily add hundreds of virtual pedestrians, generate dense crowds, and change the environment or the situation.

Our goal is to develop robust techniques that can account for noise in trajectory datasets and extract high-level characteristics of time-varying pedestrian movement dynamics (TVPMD). TVPMD is the descriptor of the movements of pedestrians at each point of time, as opposed to the averaged over long sequence of inputs. Our techniques include automatic and interactive approaches to extract the movement patterns and motion dynamics of pedestrians. There is a large collection of crowd videos available on the Internet and our statistical algorithms make it easier to extract a large library of dynamic movement patterns for different real-world situations.

We present a fast method to learn the characteristics of pedestrian movement dynamics from 2D trajectories extracted from a single video on the fly. First, our formulation models the group of pedestrians or the crowd as a complex system with non-linear dynamics and computes the most likely state of each pedestrian from noisy trajectory data using Bayesian inference. We then compute the TVMPD, the high-level characteristics of the

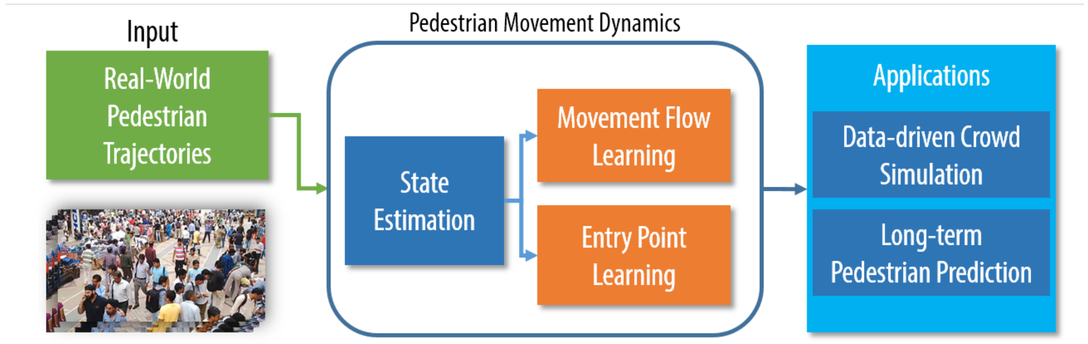

Input
Real-World Pedestrian Trajectories
Pedestrian Movement Dynamics
State Estimation
Movement Flow Learning
Entry Point Learning
Applications
Data-driven Crowd Simulation
Long-term Pedestrian Prediction

Figure 1 **Pedestrian Movement Dynamics Computation:** Our method takes extracted trajectories of real-world pedestrians as input. We use the Bayesian inference technique to estimate the most likely state of each pedestrian. Based on the estimated state, we learn time-varying behavior patterns. These behavior patterns are used as underlying rules for data-driven simulation, for pedestrian prediction, and can also be combined with other multi-agent simulation algorithms. Our approach can perform all these computations in tens of milliseconds.

pedestrians, based on the estimated states. These characteristics consist of three components that correspond to movement patterns and motion dynamics: the entry point or starting position of each individual human in the environment, the movement flow used to estimate its preferred velocity or intermediate goal position, and the local collision avoidance technique. We show that these three components can generate pedestrian movements that are similar to those observed in the original videos and that they can also be used in slightly varying environments.

Furthermore, we present algorithms to combine these characteristics with other agent-based models, which allows us to adapt their movement to a new environment or a new situation. We also use these movement dynamics to improve long-term pedestrian trajectory prediction (Fig. 1).

We have implemented our approach and evaluated the benefits of pedestrian movement dynamics on several indoor and outdoor scenarios. The original videos of these scenes have tens of real-world pedestrians, and we are able to reliably compute pedestrian movement dynamics at interactive rates on a desktop PC. We are able to demonstrate up to 12% improvement in long term pedestrian prediction and about 3.5e-01 seconds feature computation time improvement for data-driven pedestrian simulation algorithms using our approach.

For the rest of the paper we use the terms "pedestrian", "agent", "real-world agent" interchangeably indicating a real person in a video. We also use the terms "virtual agent", "virtual pedestrian", and "user-controlled agent" interchangeably indicating a simulated person.

The rest of the paper is organized as follows. Section 2 provides an overview of related work in pedestrian movement dynamics, data-driven crowd simulation, and behavior learning. We introduce the terminology and present our interactive pedestrian dynamics-learning algorithm in Section 3. In Section 4, we highlight the benefits of our approach for adaptive data-driven crowd simulation and pedestrian trajectory prediction. We describe our implementation and highlight the performance on different benchmarks in Section 5.

2 Related Work

In this section, we give a broad survey of related work, including multi-agent simulation, data-driven crowd movement dynamics, and behavior learning.

2.1 Pedestrian Trajectory Simulation

Some of the commonly used techniques for simulating crowd behaviors and trajectory computation are based on agent-based models, including rule-based methods, which use a set of behavioral rules to guide the behavior of each pedestrian [5, 6]. Another group of algorithms includes force-based methods that model interactions among pedestrians using attraction or repulsion forces [7], and velocity-based [8, 9] and vision-based approaches [10], which are useful for collision-free local navigation. Another set of algorithms is based on continuum techniques, which compute fields for pedestrians to follow based on continuum flows [11] or fluid models [12]. Many extensions and experiments have also been proposed to understand pedestrian flows [13–15] and density relationships [16, 17].

2.2 Data-driven Crowd Movement Dynamics

Data-driven methods use real-world motion specifications or captured data to generate the trajectory or behavior of each pedestrian. At a broad level, prior work in data-driven methods can be classified into offline methods (which involve preprocessing) and interactive algorithms.

Offline Methods: There is a large body of work in computer graphics and animation that captures human motion data and performs motion synthesis based on *Motion Patches* [18] and extensions that can be used to generate a dense crowd with multiple interacting human-like characters [19]. These methods can also model close interactions between pedestrians. Other data-driven methods emphasize generating trajectory-level behaviors using video data or recorded trajectories. For example, data extracted using semi-automatic trackers are used to generate group behaviors [20]. The virtual scenes can be populated by copying and pasting small pieces of real-world crowd data [21] or by using efficient data structures to represent sequences of motion in a large database for motion retrieval [22]. A different class of data-driven algorithms uses real-world crowd data to learn or optimize the motion-model parameters for agent-based simulation algorithms. This class includes a density-based measure [23], a similarity-based entropy metric [24] to learn or evaluate parameters from a given scenario; offline optimization methods that use real-world trajectory data to compute the best parameters for simulated motion models [25, 26]; and a data-driven framework that analyzes the quality of and anomalies in crowd simulations by comparing them to given training data [27].

Interactive Methods: There is large body of computer vision literature on realtime pedestrian tracking from videos, and these can be used to generate 2D agent trajectories for data-driven simulation [28, 29]. However, current methods are limited to generating 2D trajectories and cannot handle any changes in the environment or simulate different trajectory behaviors from those observed in the videos. There is a large body of work on interactive editing of crowd trajectories and animations [30, 31]. These methods can be directly used on extracted trajectories or on the full body motion of animated characters to generate plausible pedestrian movement simulations for different environments.

2.3 Pedestrian Prediction

Prior work in pedestrian prediction [32, 33] makes simple assumptions on pedestrian movement, such as the use of constant velocity or constant acceleration motion models. In order to improve the accuracy and deal with medium-to-high density crowds, more accurate motion models and interaction rules have been used. Bruce et al. [34] and Gong et al. [35] predict pedestrians' motions by estimating their destinations. Liao et al. [36] obtain a Voronoi graph from the environment and predict a pedestrian's motion along the edges. Luber et al. [37] track pedestrians using a Kalman filter-based tracker along with Helbing's social force model. Mehran et al. [38] apply the social force model to detect people's abnormal behaviors from videos. Pellegrini et al. [39] use an energy function to build up a goal-directed short-term collision-avoidance motion model. Bera et al. [40] improve pedestrian prediction and tracking accuracy by using reciprocal velocity obstacles and hybrid motion models. Yamaguchi et al. [41] use an agent-based behavioral model called ATTR and learn additional social and personal properties from the behavioral priors, such as grouping information and destination information, to perform pedestrian tracking and prediction. Fulgenzi et al. [42] use a probabilistic velocity-obstacle approach, combined with the dynamic occupancy grid. This method assumes obstacles have constant linear velocity.

2.4 Video-Based Crowd Analysis

There is extensive work in computer vision, multimedia, and robotics that analyzes the behavior and movement patterns in crowd videos, as surveyed in [43, 44], where the main objectives include human behavior understanding and recognition and crowd activity recognition for detecting abnormal behaviors [45, 46]. Many of these methods use a large number of training videos to learn the patterns offline [47, 48]. Other methods utilize motion models to learn crowd behaviors [49, 50] or machine learning methods [51, 52]. In contrast, our goal is to develop improved techniques for interactive data-driven crowd simulation.

3 Time-Varying Pedestrian Movement Dynamics

In this section, we present our interactive algorithm that learns time-varying pedestrian dynamics from real-world, 2D pedestrian trajectories. We assume that these trajectories are extracted from observations using standard tracking algorithms.

3.1 Pedestrian State

We first define specific terminology used in the paper. We use the term *pedestrian* to refer to independent individuals or agents in the crowd. We use the notion of *state* to specify the trajectory and behavior characteristics of each pedestrian. The components used to define a state govern the fidelity and realism of the resulting crowd simulation. Because the input to our algorithm consists of 2D position trajectories, our state vector consists of the information that describes the pedestrian's movements on a 2D plane. We use the symbol $\mathbf{x} \in \mathbb{R}^6$ to refer to a pedestrian's state:

$$\mathbf{x} = [\mathbf{p}\ \mathbf{v}^c\ \mathbf{v}^{pref}]^{\mathbf{T}}, \tag{1}$$

where $\mathbf{p}$ is the pedestrian's position, $\mathbf{v}^c$ is its current velocity, and $\mathbf{v}^{pref}$ is the preferred velocity on a 2D plane. The preferred velocity is the optimal velocity that a pedestrian would take to achieve its intermediate goal if there were no other pedestrians or obstacles in the scene. In practice, $\mathbf{v}^{pref}$ tends to be different from $\mathbf{v}^c$ for a given pedestrian. We use the symbol $\mathbf{S}$ to denote the current state of the environment, which corresponds to the states of all other pedestrians and the current positions of the obstacles in the scene. The state of the crowd, which consists of individual pedestrians, is a union of the set of each pedestrian's state $\mathbf{X} = \bigcup_i \mathbf{x}_i$, where subscript i denotes the i^{th} pedestrian. Our state formulation does not include any full body or gesture information. Moreover, we do not explicitly model or capture pairwise interactions between pedestrians. However, the difference between $\mathbf{v}^{pref}$ and $\mathbf{v}^c$ provides partial information about the local interactions between a pedestrian and the rest of the environment.

3.2 Pedestrian Movement Dynamics

Pedestrian dynamics consist of those factors that govern pedestrians' trajectory behaviors, i.e., the factors that change the state of the pedestrians. We model pedestrian dynamics using three components: starting position or entry point, movement flow, and the local collision-free navigation rule. Formally, we represent the characteristics of these dynamics for each pedestrian with a vector-valued function, $f()$, with an initial value determined by the function, $E()$:

$$\mathbf{x}^{t+1} = f(t, \mathbf{x}^t) = [P(\mathbf{x}^t)\ I(\mathbf{x}^t)\ G(t, \mathbf{x}^t)]^{\mathbf{T}}; \quad \mathbf{x}^0 = E(t^0). \tag{2}$$

For each pedestrian in the crowd, the function $G : \mathbb{R} \times \mathbb{R}^6 \times \mathbb{S} \to \mathbb{R}^2$ maps time t, current state of the pedestrian $\mathbf{x} \in \mathbf{X}$, and current state of the simulation environment $\mathbf{S} \in \mathbb{S}$ to a preferred velocity $\mathbf{v}^{pref}$. Function $I : \mathbb{R}^6 \times \mathbb{S} \to \mathbb{R}^2$ computes the interactions with other pedestrians and obstacles in the environment and is used to compute the collision-free current velocity $\mathbf{v}^c$ for local navigation. The function $P : \mathbb{R}^2 \to \mathbb{R}^2$ computes the position, given $\mathbf{v}^c$; $E : \mathbb{R} \to \mathbb{R}^2$ computes the initial position for time t_0, which is the time at which a particular pedestrian enters the environment. The three components of the pedestrian dynamics (entry point, movement flow, and local collision-free navigation) can be mapped to the functions $E()$, $G()$, and $I()$, respectively. We learn $E()$ and $G()$ from the 2D trajectory data. The local collision-free navigation rule $I()$ can be chosen by the data-driven algorithm.

We refer to our interactive method as learning time-varying pedestrian movement dynamics (TVPMD). Fig. 1 gives an overview of our approach, including computation of TVPMD and using that computation for crowd simulation. The input to our method consists of the trajectories extracted from a sensor. The trajectories are time-series observations of the positions of each pedestrian in a 2D plane. The output TVPMD consists of entry point distributions and movement flows learned from the trajectory data. Notably, our approach is interactive and operates based on current and recent states; in other words, it does not require future knowledge of an entire data sequence and does not have to re-perform offline training steps whenever new real-world pedestrian trajectory data is acquired or generated. As a result, our approach can effectively capture local and/or individual variations and the characteristics of time-varying trajectory behaviors. We use TVPMD for data-driven crowd simulation in Section 4.

3.3 State Estimation

The trajectories extracted from a real-world video tend to be noisy and may have incomplete tracks [53]; thus, we use the Bayesian-inference technique to compensate for any errors and to compute the state of each pedestrian.

At each time-step, the observation of a pedestrian computed by a tracking algorithm is the position of each pedestrian on a 2D plane, denoted as $\mathbf{z}^t \in \mathbb{R}^2$. The observation function $h()$ provides $\mathbf{z}^t$ of each pedestrian's true state $\hat{\mathbf{x}}^{\mathbf{t}}$ with sensor error $\mathbf{r} \in \mathbb{R}^2$, which is assumed to follow a zero-mean Gaussian distribution with covariance Σ_r:

$$\mathbf{z}^t = h(\hat{\mathbf{x}}^t) + \mathbf{r}, \mathbf{r} \sim N(0, \Sigma_r). \tag{3}$$

$h()$ can be replaced with any tracking algorithms or synthetic algorithms that provide the trajectory of each pedestrian.

The state-transition model $f()$ is an approximation of true real-world crowd dynamics with prediction error $\mathbf{q} \in \mathbb{R}^6$, which is represented as a zero-mean Gaussian distribution with covariance Σ_q:

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t) + \mathbf{q},\ \mathbf{q} \sim N(0, \Sigma_q). \tag{4}$$

We can use any local navigation algorithm or motion model for function $f()$, such as social forces, Boids, or velocity obstacles. The motion model computes the local collision-free paths for the pedestrians in the scene.

We use an Ensemble Kalman Filter (EnKF) and Expectation Maximization (EM) with the observation model $h()$ and the state transition model $f()$ to estimate the most likely state $\mathbf{x}$ of each pedestrian. EnKF uses an ensemble of discrete samples assumed to follow a Gaussian distribution to represent the distribution of the potential states. EnKF is able to provide state estimation for a non-linear state-transition model. During the prediction step, EnKF predicts the next state based on the transition model and Σ_q. When a new observation is available, Σ_q is updated based on the difference between the observation and the prediction, which is used to compute the state of the pedestrian. In addition, we

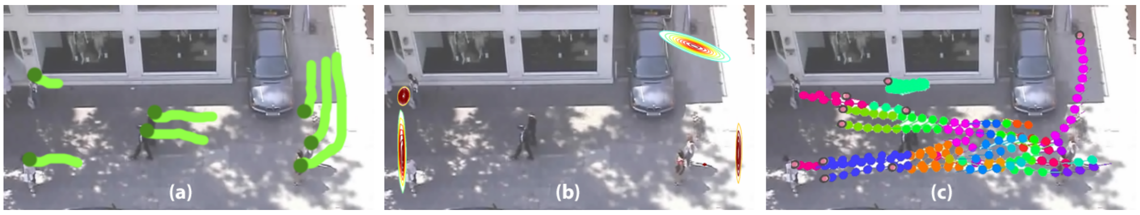


Figure 2 **Pedestrian Movement Dynamics Learning:** A one-frame example: (a) input consists of pedestrian trajectories (green) from the video; (b) probabilistic distributions of entry points at one frame computed using the Gaussian Mixture Model (shown as elliptical regions); and (c) movement flows grouped by the characteristics of pedestrian dynamics, in which each grouping is represented by the same color.

run the EM step to compute the covariance matrix Σ_q to maximize the likelihood of the state estimation.

EM for state estimation: Expectation Maximization (EM) is an iterative process that maximizes the likelihood of the latent variable [54]. The EM process repeats the ψ step, which computes the expected value for Σ_q (in our case by using EnKF) and the M step, which computes the distribution with the computed value Σ_q during the previous ψ step. By using EM, the likelihood of the state estimation being accurate to given observation data can be maximized. It is performed by maximizing the expected log-likelihood (*ll*) of covariance matrix Σ_q:

$$\psi(ll(\Sigma_q)) = -\sum_{t=0}^{t-1} \psi((\mathbf{x}^{t+1} - f(\mathbf{x}^t)^{\mathbf{T}} {\Sigma_q}^{-1} (\mathbf{x}^{t+1} - f(\mathbf{x}^t)))). \tag{5}$$

We can estimate this value by finding the average error for each sample in the ensemble at each timestep for each agent.

3.4 Dynamic Movement Flow Learning

We compute the movement features, which are used as descriptors for local pedestrian movement. These movement features are grouped together and form a cluster of a movement flow.

Movement Feature The movement features describe the characteristics of the trajectory behavior at a certain position at time frame t. The characteristics include the movement of the agent during the past w frames, which we call the *time window*, and the intended direction of the movement (preferred velocity) at this position.

The movement feature vector is represented as a six-dimensional vector:

$$\mathbf{b} = [\mathbf{p}\ \mathbf{v}^{avg}\ \mathbf{v}^{pref}]^T, \tag{6}$$

where $\mathbf{p}$, $\mathbf{v}^{avg}$, and $\mathbf{v}^{pref}$ are each two-dimensional vectors representing the current position, average velocity during past w frames, and estimated preferred velocity computed as part of state estimation, respectively. $\mathbf{v}^{avg}$ can be computed from $(\mathbf{p}^t - \mathbf{p}^{t-w.dt})/w.dt$, where dt is the time-step.

The duration of the time window, w, can be set based on the characteristics of a scene. Small time windows are good at capturing details in dynamically changing scenes with many rapid velocity changes, which are caused by some pedestrians moving quickly. Larger time windows, which tend to smooth out abrupt changes in motion, are more suitable for scenes that have little change in pedestrians' movement. For our results, we used 0.5 to 1 second of frames to set the value of w.

Movement Flow Clustering

At every w steps, we compute new behavior features for each agent in the scene using Eq. 6. We group similar features (average velocities and preferred velocities) and find K most common behavior patterns, which we call *movement flow clusters*. We use recently observed behavior features to learn the time-varying movement flow.

K-means clustering is an iterative algorithm that first assigns the cluster membership (i.e., which cluster the points belong to) for each data point, which is the dynamics feature $\mathbf{b}_i$:

$$\mathbf{B}_k = \{\mathbf{b}_i : dist(\mathbf{b}_i, \mu_k) \leq dist(\mathbf{b}_i, \mu_l) \forall l, 1 \leq l \leq K\}. \tag{7}$$

It updates the centroids of each cluster until there is no change in μ_k:

$$\mu_k = \frac{1}{|\mathbf{B}_k|} \sum_{\mathbf{b}_i \in B_k} \mathbf{b}_j. \tag{8}$$

We use the k-means data clustering algorithm to classify these features into K movement flow clusters. K and N^f are user-defined values that represent the total number of the clusters and the total number of collected behavior features, respectively, and $K \leq N^f$. A set of movement-flow clusters $B = \{B_1, B_2, ..., B_K\}$ is computed as follows:

$$\underset{B}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{b_i \in B_k} dist(b_i, \mu_k), \tag{9}$$

where b_i is a movement feature vector, μ_k is a centroid of each flow cluster, and $dist(b_i, \mu_k)$ is a distance measure between the arguments. In our case, the distance between two feature vectors is computed as

$$\begin{aligned} dist(b_i, b_j) = c_1 &\left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ + c_2 &\left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} w\, dt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} w\, dt) \right\| \\ + c_3 &\left\| (\mathbf{p}_i + \mathbf{v}_i^{pref} w\, dt) - (\mathbf{p}_j - \mathbf{v}_j^{pref} w\, dt) \right\|, \end{aligned} \tag{10}$$

which corresponds to the weighted sum of the distance among three points: current positions, previous positions, and estimated future positions (which are extrapolated using v^{pref}, c_1, c_2, and c_3 as the weight values). Comparing the distance between the positions rather than mixing the points and the vectors eliminates the need to normalize or standardize the data. Each movement-flow cluster contains adjacent features that have similar average velocities and preferred velocities (see Fig. 2 (c)).

3.5 Entry-Points Learning

Entry points are a component of pedestrian dynamics we want to learn to estimate when real pedestrians enter the scene. These starting positions and timings for each agent are very important and govern their overall trajectory. We use a multivariate Gaussian mixture model to learn the time-varying distribution of entry points, which will be used as the initial position $\mathbf{x}^0$ for a newly added pedestrian in a data-driven crowd simulation. We define $E()$ as the function that provides a position sampled from the learned distributions. For a non-spherical distribution, a Gaussian distribution is preferred; the distribution of entry points, which are scattered near the scene's boundary and often correspond to long elliptical regions, is frequently non-spherical (see Fig. 2 (b)).

We assume that the distribution of entry points, e, from which the function $E()$ samples, is a mixture of J components and that each of the components is a multivariate Gaussian distribution of a two-dimensional random variable, $\mathbf{p}$, with a set of parameters $\Theta = (\alpha_1, \cdots, \alpha_J, \theta_1, \cdots, \theta_J)$:

$$e(\mathbf{p}|\Theta) = \sum_{j=1}^{J} \alpha_j e_j(\mathbf{p}|\mu_j, \theta_j), \tag{11}$$

$$e_j(\mathbf{p}; \theta_j) = \frac{1}{2\eta|\Sigma_j|^{1/2}} exp(-\frac{1}{2}(\mathbf{p}-\mu_j)^T \Sigma_j^{-1}(\mathbf{p}-\mu_j)). \tag{12}$$

Each component e_j is a Gaussian distribution given by the parameters $\theta_j = (\mu_j, \Sigma_j)$, where μ_j is the mean of the component j and Σ_j is a 2×2 covariance matrix. α_j is a mixture weight, which is the probability of a point $\mathbf{p}$ that belongs to the component j. $\alpha_j \in [0, 1]$ for all i and the sum of α_js are constrained to 1 ($1 = \sum_{j=1}^{J} \alpha_j$). From an initial guess of the parameters θ_j, we perform EM to learn these parameters $\theta_j = (\mu_j, \Sigma_j)$ from the given entry points collected from the real pedestrian trajectories. The entry point distribution is updated whenever we have a new observation of a pedestrian entering near the boundary of the scene (i.e., the starting positions of a trajectory). We use only the recent N^e observations of entry positions from trajectories and discard old observations. A large value for N^e can capture the global distribution of entry points, whereas a smaller value for N^e can better capture the dynamic changes of the distribution. Although we update the model frequently, we can exploit the locality in distributions because the new distribution is evolving from the previous distribution. We use the previous parameters and choose cluster j, which satisfies $\text{argmin}_j ||\mathbf{p} - \mu_j||$, as our initial guess for the new distributions.

EM for Gaussian Mixture Model (GMM): The E step updates the membership weights α_j for all components, and the M step uses the updated membership weights and data to update these parameters. E and M steps are iteratively processed until the log-likelihood ($ll(\Theta)$) of the mixture model converges:

$$ll(\Theta) = \sum_{l=1}^{L} \log e(\mathbf{z}_l^0|\Theta) = \sum_{l=1}^{L} (\log \sum_{j=1}^{J}) \alpha_k e_k(\mathbf{z}_j^0|\theta_j), \tag{13}$$

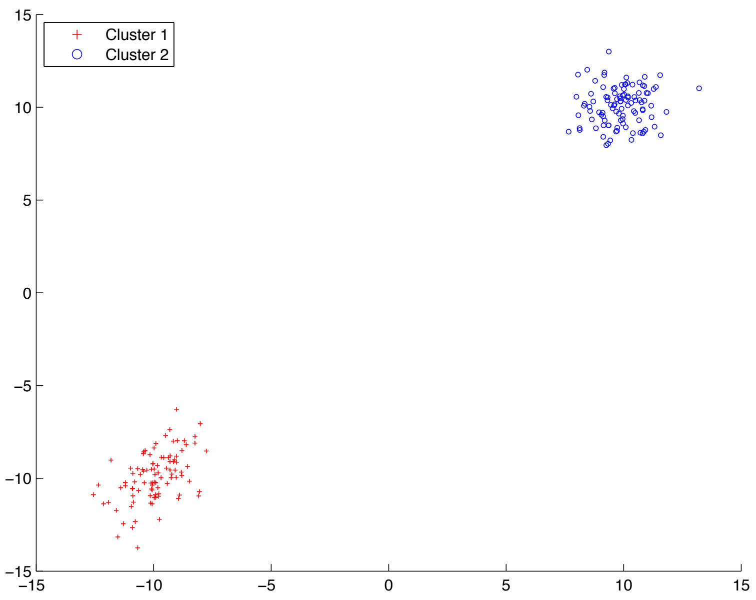


Figure 3 **Synthetic data points** are sampled from g_1 and g_2 with varying weights for each frame. Red dots are points sampled from g_1 and blue dots are points sampled from g_2. x and y axes refer to the respective x and y co-ordinates in the image space.

where L is the number of observation points, which corresponds to a set of observed entry points $Z^0 = \mathbf{z}_j^0 | j \in R$.

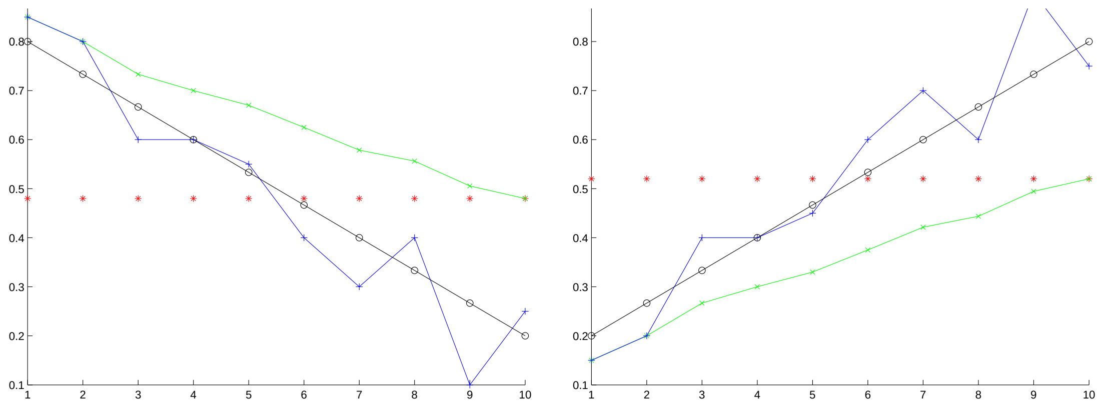
3.6 GMM for Entry Point Learning

In Sec. 3, we discussed the time window w used for TVPMD learning. We can leverage the range of local prediction based on the value of the time window w. As w gets larger, the estimation captures more global characteristics of the data. In this section, we present results from an experiment conducted with synthetic data to highlight the comparisons between local and global estimation.

In our experiment, synthetic data is generated from two multivariate Gaussian distributions, g_1 and g_2: $f(x) = (\eta)g_1(x) + (1-\eta)g_2(x)$, where $g_1 = N(\mu_1, \Sigma_1)$ and $g_2 = N(\mu_2, \Sigma_2)$.

We sample 20 points for each frame (10 total frames) while decreasing the value of η gradually from 0.8 to 0.2. The sampled points are shown in Fig. 3.

We use GMM+EM to learn the parameters μ_j, Σ_j, and η. We compare three cases. The first case learns a mixture model using all the points. This model may best describe the global distribution observed during all the frames. The second case learns a mixture model using all past data. For example, at frame t, the model takes into account all points generated from frame 1 to frame t. The final case uses time window $w = 3$. In other words, the mixture model is learned from the local data sampled from three recent frames. We refer to these models as global, accumulative, and local methods, respectively. Fig. 4 shows the comparisons between these methods. The figure shows the approximated η and $1-\eta$



(a) Cluster 1 (estimation of g_1) (b) Cluster 2 (estimation of g_2)

Figure 4 **Estimated weight of cluster1 (g_1) and cluster2 (g_2) during ten frames.** Ground truth (black), global (red), accumulative (green), and local (blue) the where x-axis refers to the time-step (sec) and y-axis refers to the weight. The global model does not perform as well as the accumulative or local model, when the data changes over time. Accumulative model gradually converge to global result. Local model generates noisy approximation due to smaller number of samples, but generally gives good approximation to the ground truth distribution.

for g_1 and g_2, respectively. As shown in the figure, the global model does not perform as well as the accumulative or local models when data changes over time. The accumulative model gradually converges to the global result. The local model generates noisy approximation due to a smaller number of samples, but generally gives good approximation of ground truth data.

Performance improvement:

In Sec. 3.5, we discussed how we use the previous parameters as an initial guess for updating the distributions. When we update the entry point estimation with new observations, we can use previously learned distributions (i.e., entry point distributions at $t-w$ frame) as priors. For example, we can assign each new data point x to the closest cluster j: $argmin_j||x-\mu_j||$. Since entry points tend to have similar distributions and previously observed entry points are stored during the time window, having such prior information provides very good estimation for the model and also improves the performance. From our experiment, the average number of iterations for EM has reduced more than 3 times when we use the prior compared to random initialization.

4 Applications

In this section, we use our TVPMD algorithm for data-driven crowd simulation and improved long-term pedestrian prediction.

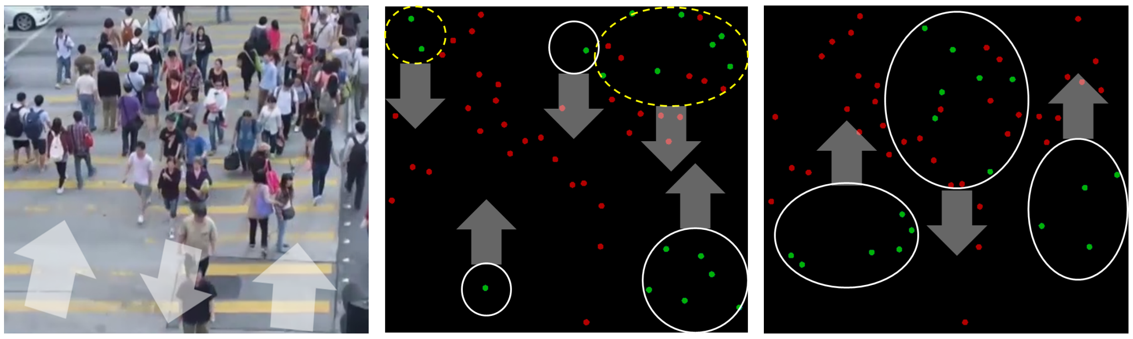

(a) Original Video (b) Without TVPMD (c) With TVPMD

Figure 5 **Manko Benchmark:** We highlight the benefits of entry point and movement flow learning. (a) A frame from a Manko video, which shows different flows corresponding to lane formation (shown with white arrows); (b) and (c) We compute collision-free trajectories of 18 virtual pedestrians (shown in green) along with the extracted trajectories of 42 real pedestrians (shown in red) from the Manko scenario. For (b), we use random entry points (dashed yellow circles) for virtual pedestrians and goal positions on the opposite side of the street. White circles highlight the virtual pedestrians who are following the same movement flow as neighboring real pedestrians. For (c), we use TVPMD (entry point distribution and movement flow learning) to generate virtual pedestrians' movements. The virtual agents follow the lane formation, as observed in the original video.

4.1 Data-driven Crowd Simulation

The first part of this section gives details of our algorithm used to compute TVPMD. In the second part, we use TVPMD to compute the state of the virtual crowd based on local collision avoidance and situational trajectory adaption methods. For a detailed discussion of data-driven crowd simulation, we refer our readers to [55].

4.1.1 Asymmetric behavior

The TVPMD computation allows user interaction with the agents in the scene by adding obstacles during the simulation. Furthermore, our method can be extended to allow users to directly control any pedestrian in the scene. In this case, we need to model the asymmetric behavior for the user-controlled pedestrian.

Asymmetric behavior modeling of agents has been studied and different modeling techniques have been proposed based on social forces and reciprocal velocity obstacles [56–58]. We use a similar approach for modeling collision avoidance between a user-controlled pedestrian and the rest of the virtual pedestrians in the scene. Currently, we use a velocity-based local navigation technique called ORCA [8] for local navigation. It computes the current velocity $\mathbf{v}^c$ from $\mathbf{v}^{pref}$ and the set of collision-free ORCA constraints. This algorithm assumes symmetric behavior, which means that all the pedestrians have equal responsibility for avoiding a collision. However, we impose 100% of the collision-avoidance responsibility on the user-controlled agent in the simulation when it has an impending collision with other pedestrians or obstacles.

In this case, the ORCA algorithm can be slightly modified to handle both symmetric and asymmetric behaviors. Each ORCA constraint is a linear constraint computed using velocity obstacles. Given two pedestrians, A and O, we compute the minimum vector $\mathbf{u}$ of the change in relative velocity needed to avoid collision. Formally, the ORCA constraint on A's velocity induced by O is given as

$$ORCA_{A|O} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_A + d\mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}, \tag{14}$$

where $\mathbf{v}_A$ is A's current velocity, $\hat{\mathbf{u}}$ is the normalized vector $\mathbf{u}$, and d is a constant value that determines the minimum change of the velocity in the direction of $\hat{\mathbf{u}}$. Normally when we deal with only virtual pedestrians in a scene, we set $d = 1/2$. This implies that each pedestrian shares the responsibility for avoiding collisions equally. When it comes to collision avoidance with a user-controlled pedestrian, we set $d = 1$, which means the user-controlled pedestrian A is solely responsible for collision-free navigation with respect to the rest of the environment. As result, the user-controlled pedestrian A moves further away to avoid collision. If A has multiple neighboring pedestrians, each neighboring pedestrian will result in a separate ORCA constraint while computing the new velocity for A. Local navigation is performed by computing the new current velocity for A ($\mathbf{v}^c$) that is closest to its preferred velocity ($\mathbf{v}^{pref}$), while satisfying all ORCA constraints:

$$\mathbf{v}_A^c = \underset{\mathbf{v} \in ORCA_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}_A^{pref}\|, \tag{15}$$

where $\mathbf{v}^c$ and $\mathbf{v}^{pref}$ are the new velocity and preferred velocity, respectively.

4.2 Long-term Pedestrian Prediction

A key aspect in any real-time prediction algorithm is estimating the motion of the pedestrian in a crowd. In this section, we give an overview of a real-time algorithm that learns movement flows from real-world 2D pedestrian trajectories that are extracted from video. Our approach involves no pre-computation or pre-learning, and can be combined with any real-time pedestrian trackers. For a detailed discussion of the pedestrian prediction approach, we refer our readers to [59].

Fig. 6 gives an overview of our approach, including computation of movement flows and their use in pedestrian prediction. The input to our method consists of a live or streaming crowd video. We extract the initial set of trajectories using an online particle-filter based pedestrian tracker. These trajectories are time-series observations of the positions of each pedestrian in the crowd. The various components used in our algorithm are shown in the figure and explained below. The output is the predicted state of each agent that is based on learning the local and global pedestrian motion patterns (Figure 7). For more details and results we point the readers to [59].

5 Results

In this section, we describe the implementation of our method and highlight its performance on different scenarios. Our system runs at interactive rates on a desktop machine with a 3.4 GHz Intel i7 processor and 8GB RAM. For state estimation, we use

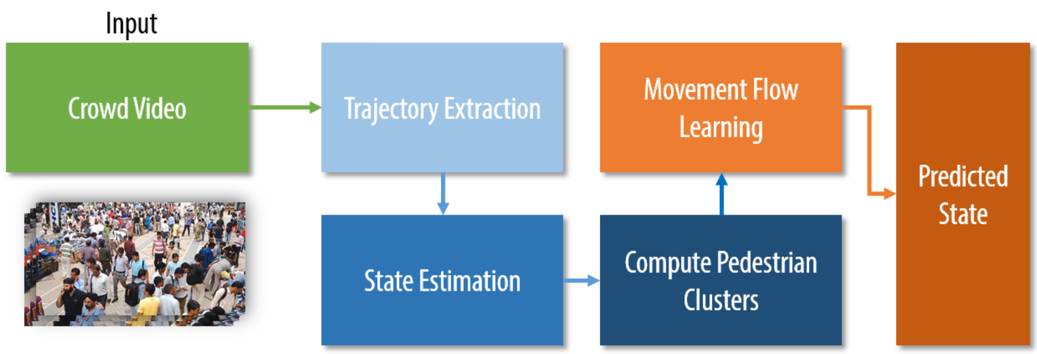


Figure 6 **Pedestrian Prediction:** We highlight various components of our real pedestrian path prediction algorithm. Our approach computes the movement flow from realtime 2D trajectory data and uses it to improve prediction accuracy.

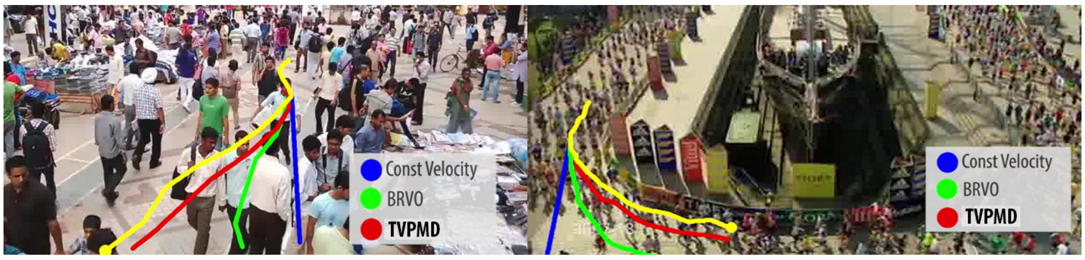


Figure 7 **Pedestrian Prediction Results:** We demonstrate the improved accuracy of our pedestrian path prediction algorithm using TVPMD over prior real-time prediction algorithms (BRVO, Const Vel) and compare them with the ground truth (in yellow). We observe upto 12% improvement in accuracy.

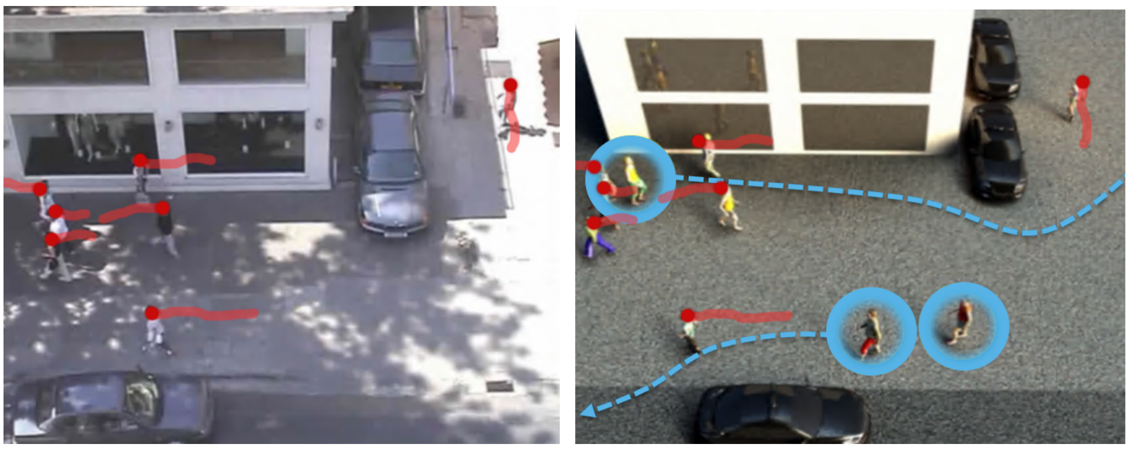
velocity-based reasoning as the state transition model, $f()$. For collision-avoidance computation, we use a publicly available library [8], and we use different real pedestrian tracking datasets corresponding to indoor and outdoor environments as the input for the TVPMD computation algorithm. These datasets are generated using manual tracking, an online multiple-person tracker, a KLT tracker, synthetic data, and 3D range sensor tracking [23, 51, 60]. Tab. 2 presents more details on these datasets along with the number of tracked pedestrians and the number of virtual pedestrians in the data-driven simulation. Our algorithms compute collision-free trajectories for the virtual pedestrians.

The TVPMD is able to capture the movement patterns and motion dynamics from the extracted trajectories. We have demonstrated the benefit of our pedestrian dynamics learning algorithm on several challenging benchmarks, including structured and unstructured benchmarks. Furthermore, we demonstrate its benefits on different scenarios: robust to noisy and varying sensor data (ATC Mall and Train Station scenarios), interactive computations (Black Friday Shopping Mall scenario), handling structured environments (Marathon scenario), adapting to a situation (Explosion scenario), and high-density simulation (Train Station scenario).

We have also applied it to the 2D trajectories generated from different crowd videos and compared the prediction accuracy with the ground truth data, that was also generated

Dataset	Challenges	Density	# Tracked	ConstVelocity		Kalman Filter		BRVO		TVPMD-based Approach	
				1 sec	5 secs	1 sec	5 secs	1 sec	5 secs	1 sec	5 secs
Students	BV, IC, PO	Medium	65	65.0%	58.2%	66.9%	61.0%	69.1%	63.6%	72.2%	66.8%
Campus	BV, IC, PO	Medium	78	62.4%	57.1%	63.5%	59.0%	66.4%	59.1%	69.6%	59.5%
seq_hotel	IC, PO	Low	390	74.7%	67.8%	76.7%	68.3%	76.9%	69.2%	79.5%	70.1%
Street	IC, PO	Low	34	78.1%	70.9%	78.9%	71.0%	81.4%	71.2%	83.8%	72.7%

Table 1 **Crowd Scene Benchmarks:** We highlight many attributes of these crowd videos, including density and the number of tracked pedestrians. We use the following abbreviations for some characteristics of the underlying scene: Background Variations (BV), Partial Occlusion (PO), and Illumination Changes (IC). We highlight the results for short-term prediction (1 sec) and long-term prediction (5 sec). We notice that our TVMPD algorithm results in higher accuracy for long-term prediction and dense scenarios. For more details and results we point the readers to [59].



(a) (b)

Figure 8 (a) A frame from a video of pedestrians (From Figure 2) in a street with extracted trajectories (shown in red); (b) Our simulation algorithm computes collision-free trajectories of virtual pedestrians (shown in blue) in the 3D virtual environment, which have the same movement flows as extracted trajectories (red).

using a pedestrian tracker. The underlying crowd videos have different pedestrian density corresponding to low (i.e. less than 1 pedestrian per squared meter) and medium (1-2 pedestrians per squared meter). We highlight the datasets, their crowd characteristics, and the prediction accuracy of different real-time algorithms for short-term and long-term prediction in Tab. 1. We include comparisons to constant velocity (ConstVelocity) and a Kalman filter. Finally, we also compare the accuracy with the Bayesian reciprocal velocity obstacle (BRVO) algorithm [61] that computes a more individualized motion model for estimating local movement patterns.

It is important to predict the trajectory over a longer time-horizon. Our approach is able to perform long-term prediction (5-6 seconds) and exhibits much higher accuracy than prior methods (see Tab. 1). We notice that our algorithm results in higher accuracy for long-term prediction and dense scenarios. We use a simple prediction metric to evaluate the accuracy of both long and short term prediction. The average human stride length is about 0.8 meters [62]. A prediction is counted as successful when the estimated mean error between the prediction result and the ground truth value at that time instant is less than this constant. We define prediction accuracy as the ratio of the number of "successful" predictions and total number of tracked pedestrians in a scene. We use our algorithm for long and short term prediction across a large number of datasets, highlighted in Tab. 1.

Scenario	Sensor	# Tracked Peds.	# Virtual Peds.	# Static Obst.	# Input Frames	Avg. time TVPMD	Avg. time DDS.
Manko	Online Tracking	42	70	0	373	0.075	0.037
Marathon	Online Tracking	18	500	33	450	0.040	0.098
Explosion	Online Tracking	19	110	0	238	0.030	0.031
Street	Manual Tracking	147	167	0	9014	0.012	0.004
Train Station	KLT (Tracklets)	200	200-943	37-47	999	0.053	0.005
ATC Mall	3D Range Sensors	50	207	38	7199	0.023	0.022
BlackFriday	Synthetic Data	6	271-1000	20-28	109	0.085	0.037
IITF-1	Online Tracking	18	500	33	450	0.041	0.003
IITF-3	Online Tracking	19	110	0	238	0.031	0.027
IITF-5	Online Tracking	18	500	33	450	0.045	0.029
NPLC-1	Online Tracking	19	110	0	238	0.032	0.013
NPLC-3	Online Tracking	18	500	33	450	0.048	0.041
NDLS-2	Online Tracking	19	110	0	238	0.034	0.030

Table 2 Performance of TVPMD on a single core for different scenarios. We highlight the number of real and virtual pedestrians, the number of static obstacles, the number of frames of extracted trajectories, and the time (in seconds) spent in different stages of our algorithm. DDS is the Data-Driven Scene computation time (in seconds), which includes the time for computing the additional collision avoidance constraints when virtual agents are introduced and other simulation overheads.

6 Conclusions, Limitations, and Future Work

We present statistical algorithms to learn the characteristics of pedestrian movement from trajectories extracted from real videos. These characteristics are used to compute collision-free trajectories of virtual pedestrians whose movement patterns resemble those of pedestrians in the original video. Our approach is automatic and interactive and captures the dynamically changing movement behaviors of real pedestrians. We demonstrate its applications for many data-driven crowd simulations, where we can easily add hundreds of virtual pedestrians, generate dense crowds, and change the environment or the situation.
Limitations: The performance of our learning algorithm is governed by the accuracy of the input trajectories. Current algorithms for automatic pedestrian tracking can only handle low-to-medium density crowds. Our learning algorithm makes some assumptions about sensor error and statistical distributions and is only useful for capturing the characteristics of local pedestrian dynamics for each pedestrian, whereas offline learning meth-

ods can compute many global characteristics. We consider only characteristics like movement flows, entry points etc. to compute the trajectories of virtual pedestrians and do not consider other aspects of pedestrian behaviors or states, full body actions or the interactions among pedestrians. Our approach only computes the trajectories, but we need to combine our method with techniques that can generate plausible animation and rendering. For example, we use out-of-box rendering toolkit which may result in some motion artifacts. Finally, even though in theory our algorithm can generate very dense simulations, it is limited by the underlying motion model, so only a collision-free movement without physical interaction is possible since we don't model other aspects of pedestrian behaviors or states, full body actions or the interactions among pedestrians.
Future Work: There are many avenues for future work. In addition to overcoming the limitations of our work, our interactive TVPMD can also be combined with other data-driven crowd simulation algorithms and offline behavior learning methods. We would like to combine the pedestrian dynamics characteristics with other techniques that can model complex crowd behaviors or multi-character motion synthesis techniques [18, 19].

7 Acknowledgements

This work was supported by National Science Foundation award 1305286, ARO contract W911NF-16-1-0085, and a grant from the Boeing company.

References

[1] Bera, A., Kim, S., Manocha, D.: Realtime anomaly detection using trajectory-level crowd behavior learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 50–57 (2016)

[2] Wolinski, D., Guy, S., Olivier, A.H., Lin, M., Manocha, D., Pettré, J.: Parameter estimation and comparative evaluation of crowd simulations. In: Computer Graphics Forum, vol. 33, pp. 303–312. The Eurographics Association and Blackwell Publishing Ltd. (2014)

[3] Kim, S., Bera, A., Manocha, D.: Interactive crowd content generation and analysis using trajectory-level behavior learning. In: Multimedia (ISM), 2015 IEEE International Symposium on, pp. 21–26. IEEE (2015)

[4] Bera, A., Kim, S., Manocha, D.: Online parameter learning for data-driven crowd simulation and content generation. Computers & Graphics **55**, 68–79 (2016)

[5] Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: SIGGRAPH '87, pp. 25–34. ACM, New York, NY, USA (1987). doi:10.1145/37401.37406

[6] Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: Symposium on Computer animation, pp. 99–108 (2007)

[7] Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**, 4282–4286 (1995)

[8] van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics Research: 14th ISRR (STAR), vol. 70, pp. 3–19 (2011)

[9] Karamouzas, I., Overmars, M.: Simulating and evaluating the local behavior of small pedestrian groups. IEEE Trans. on Visualization and Computer Graphics **18**(3), 394–406 (2012)

[10] Ondřej, J., Pettré, J., Olivier, A.H., Donikian, S.: A synthetic-vision based steering approach for crowd simulation. ACM Trans. Graph. **29**(4), 123:1–123:9 (2010)

[11] Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: ACM SIGGRAPH 2006, pp. 1160–1168. ACM (2006)

[12] Narain, R., Golas, A., Curtis, S., Lin, M.C.: Aggregate dynamics for dense crowd simulation. ACM Trans. Graph. **28**(5), 122:1–122:8 (2009)

[13] Zhang, J., Klingsch, W., Schadschneider, A., Seyfried, A.: Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. J. Stat. Mech. **2012**(02), P02002 (2012)

[14] Burghardt, S., Klingsch, W., Seyfried, A.: Analysis of flow-influencing factors in mouths of grandstands. In: Pedestrian and Evacuation Dynamics, vol. 4 (2012)

[15] Kretz, T., Grnebohm, A., Schreckenberg, M.: Experimental study of pedestrian flow through a bottleneck. Journal of Statistical Mechanics: Theory and Experiment p. P10014 (2006)

[16] Seyfried, A., Steffen, B., Klingsch, W., Boltes, M.: The fundamental diagram of pedestrian movement revisited. J. Stat. Mech. (10) (2005)

[17] Narang, S., Best, A., Curtis, S., Manocha, D.: Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. PLoS ONE **10**(4), e0117856 (2015). doi:10.1371/journal.pone.0117856

[18] Lee, K.H., Choi, M.G., Lee, J.: Motion patches: Building blocks for virtual environments annotated with motion data. ACM Trans. Graph. **25**(3), 898–906 (2006). doi:10.1145/1141911.1141972

[19] Yersin, B., Maïm, J., Pettré, J., Thalmann, D.: Crowd patches: populating large-scale virtual environments for real-time applications. In: Interactive 3D graphics and games, pp. 207–214 (2009)

[20] Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: Symposium on Computer Animation, pp. 109–118 (2007)

[21] Li, Y., Christie, M., Siret, O., Kulpa, R., Pettré, J.: Cloning crowd motions. In: Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '12, pp. 201–210 (2012). URL http://dl.acm.org/citation.cfm?id=2422356.2422385

[22] Kapadia, M., Chiang, I.k., Thomas, T., Badler, N.I., Kider Jr., J.T.: Efficient motion retrieval in large motion databases. In: Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 19–28 (2013). doi:10.1145/2448196.2448199

[23] Lerner, A., Fitusi, E., Chrysanthou, Y., Cohen-Or, D.: Fitting behaviors to pedestrian simulations. In: Symp. on Computer Animation, p. 199208 (2009)

[24] Guy, S.J., van den Berg, J., Liu, W., Lau, R., Lin, M.C., Manocha, D.: A statistical similarity measure for aggregate crowd dynamics. ACM Trans. Graph. **31**(6), 190:1–190:11 (2012). doi:10.1145/2366145.2366209

[25] Wolinski, D., Guy, S.J., Olivier, A.H., Lin, M.C., Manocha, D., Pettré, J.: Parameter estimation and comparative evaluation of crowd simulations. In: Eurographics (2014)

[26] Berseth, G., Kapadia, M., Haworth, B., Faloutsos, P.: Steerfit: Automated parameter fitting for steering algorithms. In: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014). doi:10.2312/sca.20141129

[27] Charalambous, P., Karamouzas, I., Guy, S.J., Chrysanthou, Y.: A data-driven framework for visual crowd analysis. Computer Graphics Forum **33**(7), 41–50 (2014). doi:10.1111/cgf.12472

[28] Zhang, K., Zhang, L., Yang, M.H.: Real-time compressive tracking. In: ECCV, pp. 864–877 (2012)

[29] Bera, A., Kim, S., Manocha, D.: Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In: Proceedings of Graphics Interface (2015)

[30] Jordao, K., Pettré, J., Christie, M., Cani, M.P.: Crowd Sculpting: A space-time sculpting method for populating virtual environments. Computer Graphics Forum **33**(2), 351–360 (2014). doi:10.1111/cgf.12316

[31] Kwon, T., Lee, K.H., Lee, J., Takahashi, S.: Group motion editing. ACM Trans. Graph. **27**(3), 80:1–80:8 (2008). doi:10.1145/1360612.1360679

[32] Cui, J., Zha, H., Zhao, H., Shibasaki, R.: Tracking multiple people using laser and vision. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2116–2121. IEEE (2005)

[33] Kratz, L., Nishino, K.: Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. Pattern Analysis and Machine Intelligence, IEEE Transactions on (99), 11 (2011)

[34] Bruce, A., Gordon, G.: Better motion prediction for people-tracking. In: Proc. of the International Conference on Robotics and Automation (ICRA), New Orleans, USA (2004)

[35] Gong, H., Sim, J., Likhachev, M., Shi, J.: Multi-hypothesis motion planning for visual object tracking (2011)

[36] Liao, L., Fox, D., Hightower, J., Kautz, H., Schulz, D.: Voronoi tracking: Location estimation using sparse and noisy sensor data. In: IROS (2003)

[37] Luber, M., Stork, J., Tipaldi, G., Arras, K.: People tracking with human motion predictions from social forces. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pp. 464–469 (2010)

[38] Mehran, R., Oyama, A., Shah, M.: Abnormal crowd behavior detection using social force model. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition,CVPR, pp. 935–942 (2009)

[39] Pellegrini, S., Ess, A., Schindler, K., Van Gool, L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In: ICCV, pp. 261–268 (2009)

[40] Bera, A., Manocha, D.: Reach: Realtime crowd tracking using a hybrid motion model. ICRA (2015)

[41] Yamaguchi, K., Berg, A., Ortiz, L., Berg, T.: Who are you with and where are you going? In: Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1345–1352 (2011). doi:10.1109/CVPR.2011.5995468

[42] Fulgenzi, C., Spalanzani, A., Laugier, C.: Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In: Robotics and Automation, 2007 IEEE International Conference on, pp. 1610–1616 (2007). doi:10.1109/ROBOT.2007.363554

[43] Li, T., Chang, H., Wang, M., Ni, B., Hong, R., Yan, S.: Crowded scene analysis: A survey. Circuits and Systems for Video Technology, IEEE Transactions on **25**(3), 367–386 (2015)

[44] Borges, P., Conci, N., Cavallaro, A.: Video-based human behavior understanding: A survey. Circuits and Systems for Video Technology, IEEE Transactions on **23**(11), 1993–2008 (2013). doi:10.1109/TCSVT.2013.2270402

[45] Hu, W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviors. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **34**(3), 334–352 (2004)

[46] Kim, S., Bera, A., Manocha, D.: Interactive crowd content generation and analysis using trajectory-level behavior learning. Tech. rep., University of North Carolina at Chapel Hill (2015)

[47] Zen, G., Ricci, E.: Earth mover's prototypes: A convex learning approach for discovering activity patterns in dynamic scenes. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 3225–3232 (2011). doi:10.1109/CVPR.2011.5995578

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

[49] Mehran, R., Oyama, A., Shah, M.: Abnormal crowd behavior detection using social force model. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 935–942 (2009). doi:10.1109/CVPR.2009.5206641

[50] Pellegrini, S., Gall, J., Sigal, L., Gool, L.: Destination flow for crowd simulation. In: Computer Vision ECCV 2012. Workshops and Demonstrations, vol. 7585, pp. 162–171 (2012). doi:10.1007/978-3-642-33885-4_17

[51] Zhou, B., Wang, X., Tang, X.: Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 2871–2878 (2012). doi:10.1109/CVPR.2012.6248013

[52] Sun, L., Li, X., Qin, W.: Simulating realistic crowd based on agent trajectories. Computer Animation and Virtual Worlds **24**(3-4), 165–172 (2013). doi:10.1002/cav.1507

[53] Enzweiler, M., Gavrila, D.M.: Monocular pedestrian detection: Survey and experiments. PAMI pp. 2179–2195 (2009)

[54] McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions (Wiley Series in Probability and Statistics), 2 edn. Wiley-Interscience (2008)

[55] Kim, S., Bera, A., Best, A., Chabra, R., Manocha, D.: Interactive and adaptive data-driven crowd simulation. In: IEEE Virtual Reality (VR), pp. 29–38. IEEE (2016)

[56] Curtis, S., Manocha, D.: Pedestrian simulation using geometric reasoning in velocity space (in PEDS, 2012)

[57] Kim, S., Guy, S.J., Liu, W., Lau, R.W., Lin, M.C., Manocha, D.: Predicting pedestrian trajectories using velocity-space reasoning. In: WAFR (2012)

[58] Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphics Forum **26**(3), 655–664 (2007). doi:10.1111/j.1467-8659.2007.01089.x

[59] Bera, A., Kim, S., Randhavane, T., Pratapa, S., Manocha, D.: Glmp-realtime pedestrian path prediction using global and local movement patterns. ICRA (2016)

[60] Brscic, D., Kanda, T., Ikeda, T., Miyashita, T.: Person position and body direction tracking in large public spaces using 3d range sensors. IEEE Transactions on Human-Machine Systems **43**(6), 522–534 (2013)

[61] Kim, S., Guy, S.J., Liu, W., Wilkie, D., Lau, R.W., Lin, M.C., Manocha, D.: Brvo: Predicting pedestrian trajectories using velocity-space reasoning. The International Journal of Robotics Research p. 0278364914555543 (2014)

[62] Reynolds, T.R.: Stride length and its determinants in humans, early hominids, primates, and mammals. American Journal of Physical Anthropology (1987)