Interactive and Adaptive Data-Driven Crowd Simulation

Sujeong Kim* Aniket Bera Andrew Best Rohan Chabra Dinesh Manocha†

University of North Carolina at Chapel Hill
http://gamma.cs.unc.edu/DDPD

ABSTRACT

We present an adaptive data-driven algorithm for interactive crowd simulation. Our approach combines realistic trajectory behaviors extracted from videos with synthetic multi-agent algorithms to generate plausible simulations. We use statistical techniques to compute the movement patterns and motion dynamics from noisy 2D trajectories extracted from crowd videos. These learned pedestrian dynamic characteristics are used to generate collision-free trajectories of virtual pedestrians in slightly different environments or situations. The overall approach is robust and can generate perceptually realistic crowd movements at interactive rates in dynamic environments. We also present results from preliminary user studies that evaluate the trajectory behaviors generated by our algorithm.

1 INTRODUCTION

Crowd simulation has received considerable attention in virtual reality (VR), games, computer-aided design, and robotics. Some of the driving applications include training of law enforcement officials or military personnel [40], virtual reality therapy for crowd phobias [38], investigation of pathological processes in mental disorders [7], pedestrian flow analysis of architectural models and urban layouts, etc. In these applications, one of the goals is to generate realistic crowd movements or emerging behaviors in the background, while the user is immersed in the scene and performing certain tasks.

Prior studies and evaluation in psychology and virtual environments have concluded that many aspects of pedestrian movement, including positions and orientations, are important for realistic human perception of crowds [9, 30]. Moreover, the accuracy of collision avoidance is important for generating plausible human motion for navigation [5]. Some user studies [42] suggested that the realism of character behavior in a virtual environment increases the sense of presence.

Crowd simulation techniques have been widely studied in the literature. Some of the commonly used techniques are based on synthetic multi-agent simulation algorithms [34, 29, 13, 41, 16, 11, 27, 28], continuum techniques [39, 25], cognitive or personality models [45, 8, 12], etc. In most of these approaches, the behaviors of human-like agents are governed by pre-defined rules or criteria. As a result, it can be hard to simulate the dynamic nature, variety, and subtle aspects of real-world crowd motions without considerable tuning of the parameters.

Many researchers have advocated use of data-driven or example-based crowd simulation algorithms to generate realistic crowd behaviors [19, 44, 22]. Their emergence grows out of the increasing availability of real-world examples of individual humans and crowds movements, driven in part by improvements in high-resolution cameras and motion-capture systems that can be used to generate large databases on real world crowd data [24, 15]. Moreover, advances in computer vision and tracking algorithms have made it possible to extract pedestrian trajectories at interactive rates and use them for data-driven crowd simulation [46, 1]. 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. Other techniques include offline methods for scalable multi-character motion synthesis [44, 37], which can model close interactions between pedestrians for non-interactive applications. Most behavior learning algorithms require a large number of training videos to learn the patterns offline and typically extract a fixed set of parameters that are used as a global characterization of pedestrian behavior or trajectories [47, 14]; thus, they may not be suited to capturing the dynamic nature or time-varying behaviors of pedestrians that is required for virtual environments.

Figure 1: **Interactive Crowd Simulation:** Our adaptive data-driven algorithm can generate realistic crowd trajectory behaviors of a large number of agents at interactive rates. The pedestrian dynamics and movement characteristics are captured from real-world trajectory data. Our approach can adapt the simulation to a dynamic environment.

Main Results: In this paper, we present a new crowd simulation approach that combines the realism of data-driven methods with the adaptation and interactivity of synthetic multi-agent techniques. We capture the motion characteristics of pedestrians by learning the movement patterns and motion dynamics from extracted 2D trajectories. These characteristics are represented as time-varying pedestrian dynamics and used to generate crowd movements in virtual environments that are similar to those observed in real-world videos. As the individual humans in the original video change their speed or movement, each virtual pedestrian in the simulation can dynamically adapt its trajectory to interact with other pedestrians and obstacles in the scene.

As compared to prior crowd simulation techniques, our approach offers many advantages. First, we use statistical algorithms to estimate the dynamics characteristics from noisy or sparse trajectory data. As a result, our approach can be easily integrated with current real-time pedestrian tracking algorithms and used with any crowd videos. Given the large collection of crowd videos available on the internet (e.g. YouTube), our approach makes it easier to instantly generate dynamic movement patterns for different situations. Sec-

*Now at SRI International
†e-mail:{sujeong, ab, best, rohanc, dm}@cs.unc.edu

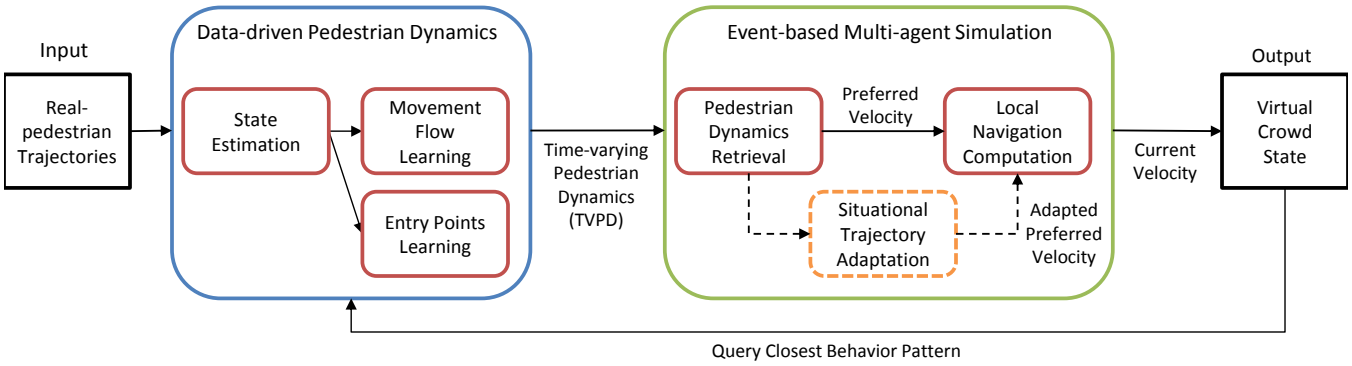


Figure 2: **Interactive data-driven simulation pipeline:** Our method takes extracted trajectories of real-world pedestrians as input. We first learn pedestrian dynamics from the input trajecotires, using state estimation followed by movement flow learning and entry point learning. The learned pedestrian dynamics can be combined with event-based multi-agent simulation algorithms and local navigation algorithms to create new crowd trajectories. All these computations can be performed in tens of milliseconds.

ond, our approach is interactive and we can easily update crowd movements as the environment changes. Third, the pedestrian dynamics characteristics can be combined with synthetic agent-based models that can change each individuals behavior in response to an external event, e.g. an external stressor resulting in a panic situation. Finally, these characteristics can be used to add an arbitrary number of virtual pedestrians to a given environment that have a similar movement pattern as the original video, and maintain the speed/density relationship.

The overall approach is automatic, involves no user editing, and can be used for interactive crowd simulations in dynamic environments. We have implemented our approach and tested it on several indoor and outdoor scenarios with varying pedestrian movement patterns, using trajectory data from a variety of crowd videos. The original videos of these scenes have tens of real-world pedestrians, and we are able to generate adaptive data-driven simulations with tens or hundreds of pedestrians in slightly different environments or situations. We also evaluated the benefits of our approach for virtual environments by performing two user-studies. Our preliminary results indicate that use of data-driven pedestrian dynamics results in realistic crowd trajectory behaviors.

The rest of the paper is organized as follows. We introduce the terminology and present our interactive pedestrian dynamics–learning algorithm in Section 2. In Section 3, we use these characteristics for data-driven crowd simulation. We describe our implementation and highlight its performance in various scenarios in Section 4. We discuss perceptual evaluation in Section 5. Finally, we analyze its performance and compare it with prior approaches in Section 6.

2 Data-Driven Pedestrian Dynamics (DDPD)

In this section, we present our interactive algorithm that learns time-varying pedestrian dynamics from real-world pedestrian 2D trajectories. We assume that these trajectories are generated using standard tracking algorithms. These are used as the underlying pedestrian characteristics for data-driven simulation. We direct the reader to the technical report [17] for mathematical and low-level details.

2.1 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 consist of 2D position trajectories, our state vector consists of the information that describes the pedestrian's movements on a 2D plane. We use the vector $\mathbf{x} = [\mathbf{p}\ \mathbf{v}^c\ \mathbf{v}^{pref}]^\mathbf{T}, \mathbf{x} \in \mathbb{R}^6$ to refer to a pedestrian's state, 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 state of all other pedestrians and the current position 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.

2.2 Pedestrian 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, $E()$:

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

For each pedestrian in the crowd, the function $G : \mathbb{R}\times\mathbb{R}^6\times\mathbb{S} \rightarrow \mathbb{R}^2$ 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 $I : \mathbb{R}^6 \times \mathbb{S} \rightarrow \mathbb{R}^2$ 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 $P : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ computes the position, given $\mathbf{v}^c$; $E : \mathbb{R} \rightarrow \mathbb{R}^2$ 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 data-driven pedestrian dynamics (DDPD). Fig. 2 gives an overview of our approach,

Algorithm 1: Adaptive Data-driven Crowd Simulation

Input: Observed position of each real agent $\mathbf{z}_i^t, i \in R$, Simulated state of each virtual agent $\mathbf{x}_i^t, i \in V$, Pedestrian local navigation model I, Composed situational trajectory adaption function J of n number of separate situational trajectory adaption function modules $J = J_1 \circ J_2 \circ \cdots \circ J_{n-1} \circ J_n$, $J(x) = x$ if $n = 0$, current time step t, last frame of the real-world data t_{end}, size of the time window w for DDPD

Output: State of virtual crowds $\mathbf{X}$

```
if t ≤ t_end then
    // State Estimation
    // 1.  DDPD computation
    foreach i ∈ R do
        Compute x_i^t from z_i^t;
    // Update DDPD
    if t%w == 0 then
        foreach i ∈ R do
            // compute pedestrian dynamics
               feature b_i and add to the
               closest behavior cluster B_k
            Compute b_i from x_i^t and x_i^{t-w};
            B_k = {b_i : argmin_k dist(b_i, μ_k) ;
        // Update Clusters
        while not converged do
            B_k = {b_i : dist(b_i, μ_k) ≤ dist(b_i, μ_l)∀l, 1 ≤
            l ≤ K} ;
            μ_k = 1/|B_k| Σ_{b_i ∈ B_k} b_j ;
// 2.  DDPD + Synthetic Algorithm
foreach i ∈ V do
    // Pedestrian Dynamics
    Compute b_i from x_i^t and x_i^{t-w};
    // Query PrefVelocity From DDPD
    c = argmin_k dist(b_i, μ_k);
    x_i^t.v^pref = μ_c.v^pref;
    // Situational Trajectory Behavior
       Adaption
    x_i^t.v^pref=J(x_i^t);
// Local Collision Avoidance
X_V = x_i^t | i ∈ V;
X_V^{t+1} = I(X_V);
```

including computation of DDPD and using them 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 DDPD 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 DDPD for data-driven crowd simulation in Section 3, which corresponds to the event-based multi-agent simulation part in the overview diagram (Figure 2).

2.3 State Estimation

The trajectories extracted from a real-world video tend to be noisy and may have incomplete tracks [10]; thus, we use 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:

$$z^t = h(\hat{\mathbf{x}}^t) + \mathbf{r}, \mathbf{r} \sim N(0, \Sigma_r). \tag{2}$$

$h()$ can be replaced with any tracking algorithms or synthetic algorithms that provides 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:

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t) + \mathbf{q},\ \mathbf{q} \sim N(0, \Sigma_q). \tag{3}$$

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. 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 run the EM step to compute the covariance matrix Σ_q to maximize the likelihood of the state estimation. For more details about EM-based state computation, please see the technical report [17].

2.4 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 *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 $\mathbf{b} = [\mathbf{p}\ \mathbf{v}^{avg}\ \mathbf{v}^{pref}]^T$, 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 $(\mathbf{p}^t - \mathbf{p}^{t-w})/w * dt$, 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. We group similar features 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.

We use the k-means data clustering algorithm to classify these features into K movement flow clusters. A set of movement-flow clusters $B = \{B_1, B_2, ..., B_K\}$ is computed as follows:

$$\underset{B}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{b_i \in B_k} dist(b_i, \mu_k), \tag{4}$$

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

$$\begin{aligned} dist(b_i, b_j) = c_1 \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ + c_2 \left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} w\, dt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} w\, dt) \right\| \\ + c_3 \left\| (\mathbf{p}_i + \mathbf{v}_i^{pref} w\, dt) - (\mathbf{p}_j - \mathbf{v}_j^{pref} w\, dt) \right\|, \end{aligned} \tag{5}$$

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.

The pseudo-code of the overall pedestrian movement flow learning algorithm is given in the Algorithm 1.

2.5 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. For example, unidimensional Poisson distribution was used to instantiate vehicles according to density distribution for traffic simulations [36].

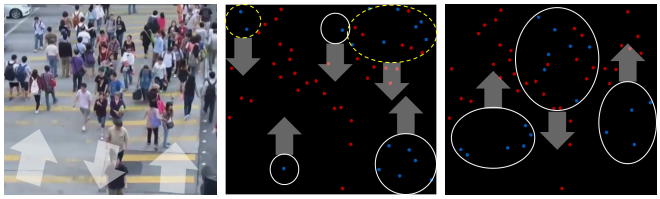
We assume that the distribution of entry points, e, that the function $E()$ samples from, is a mixture of J components and that each of the components is a multivariate Gaussian distribution of a two-dimensional random variable, $\mathbf{p}$, with a set of parameters $\Theta = (\alpha_1, \cdots, \alpha_J,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

$$e(\mathbf{p}|\Theta) = \sum_{j=1}^{J} \alpha_j e_j(\mathbf{p}|\mu_j, \theta_j), \tag{6}$$

$$e_j(\mathbf{p}; \theta_j) = \frac{1}{2\pi |\Sigma_j|^{1/2}} exp(-\frac{1}{2}(\mathbf{p} - \mu_j)^T \Sigma_j^{-1} (\mathbf{p} - \mu_j)). \tag{7}$$

Each component e_j is a Gaussian distribution given by the parameters $\theta_j = (\mu_j, \Sigma_j)$, where μ_j is 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 sum of α_j's 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. More details are provided in the supplemental material.

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 $\operatorname{argmin}_j ||\mathbf{p} - \mu_j||$, as our initial guess for the new distributions.



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

Figure 3: **Manko scenario:** 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 virtual pedestrians. For (b), we use random entry points 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 DDPD (entry point distribution and movement flow learning) to generate virtual pedestrians' movement. The virtual agent's follow the lane formation, as observed in the original video.

3 Adaptive Data-driven Crowd Simulation

In this section, we use DDPD to generate data-driven crowd simulations. In particular, DDPD are used to compute the starting position and the steering or preferred velocity of each pedestrian in the scene, which correspond to $E()$ and $G()$, respectively. During the simulation, the local collision avoidance rule, $I()$, is employed to perform collision-free navigation and compute the actual velocity of each pedestrian. Any of the local navigation agent-based models that compute the interaction between a pedestrian and the rest of the environment, i.e., other pedestrians and obstacles, can be used. Some widely used choices include social forces [13, 29], vision-based choices [27], hybrid method choices [25], velocity-obstacle-based choices [41, 16], and rule-based choices [34]. This decoupled usage of different components of DDPD enables us to use the original movement patterns in different or varying environments compared with those that were captured in the original video.

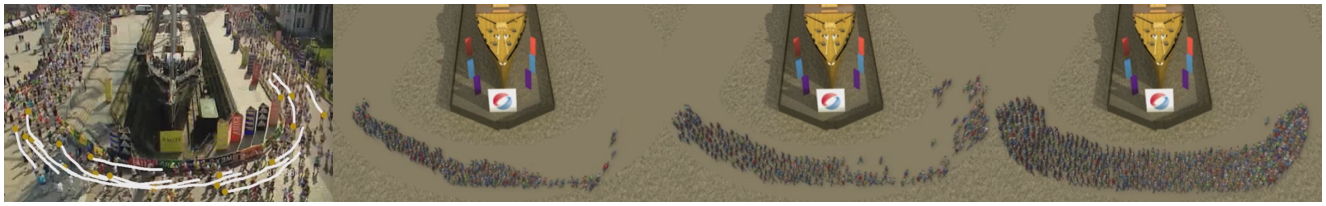
3.1 Pedestrian Dynamics Retrieval

Given a scene description, new virtual agents can be added to the scene at any time during the simulation. The initial position of each newly added virtual agent is sampled from the entry point distributions, which are modeled as a mixture of Gaussian distributions (see Equation 6). First, we select one of the Gaussian distributions, e_j, based on weight, α_j, and sample a point from the chosen distribution (Equation 7). Because we compute DDPD adaptively to the new observations, the parameters of the mixture model e that is used for entry point distribution also vary over time.

After a virtual agent is added to the scenario, we must compute its trajectory beginning from the entry point location. We use the notion of *preferred velocity* that is employed to specify the intermediate goal position for a given pedestrian in agent-based models (see Section 3.1). We use DDPD to compute the preferred velocity of each agent. At runtime, we compute this velocity by querying to which movement cluster that pedestrian belong to (set B in Equation 4). More specifically, we query the closest cluster using the behavior feature $\mathbf{b}_i$ of an agent i at time t, computed from its state estimation $\mathbf{x}^t$ and $\mathbf{x}^{t-w}$:

$$\underset{k}{\operatorname{argmin}}\, dist(\mathbf{b}_i, \mu_k), \tag{8}$$

where k is the label of the closest behavior cluster of the agent and μ_k is the centroid feature of the cluster k. We use the preferred velocity of the centroid feature to update the preferred velocity of



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

Figure 4: **Marathon Scenario:** We compare the performance of different algorithms used to generate the trajectories of 500 pedestrians in the Marathon scenario: (a) The original video frame with 18 extracted trajectories (white); (b) A multi-agent simulation using five intermediate goal positions along the track; (c) We run the same simulation with optimized parameters using an offline optimization method; and (d) Instead of the intermediate goals and/or optimized parameters, we use only DDPD. Notably, DDPD captures the pedestrian flow in the original video.

the agent. Because DDPD captures time-varying characteristics, the generated virtual pedestrian trajectories exhibit similar spatio-temporal characteristics in terms of the resulting trajectories.

3.2 Adapting to Different Environments and Situations

DDPD can be combined with other agent-based models that can change the trajectory or behavior of an agent depending on the environment or the situation. We refer to these as *situational trajectory adaption modules*; these modules are used to generate variations in the crowd simulation. We define the situational trajectory adaption function A as a composition of N^a separate situational trajectory adaption function modules:

$$A = A_1 \circ A_2 \circ \cdots \circ A_{N^a-1} \circ A_{N^a}, \tag{9}$$

where $A(x) = x$ (identity function), if $N^a = 0$. The situational trajectory adaption modules A_i act as a filtering method on the preferred velocity of a pedestrian based on its state and on the state of the environment. The output of a situational trajectory adaption A_i is an updated, adapted preferred velocity of the agent. As a result, our approach can be combined with any agent-based model (A_i) that incorporates the change in behavior or the trajectory by updating the preferred velocity. In our current system, we have integrated DDPD with a Density-dependent Filtering (DDF) module and General Adaptation Syndrome (GAS) behavior modules.

3.2.1 Density-dependent Filters

Density-dependent filters (DDF) are used to ensure that the trajectories of pedestrians in a dense crowd satisfy the speed/density relationships that are typically expressed using the Fundamental Diagram [3]. The density is often defined in terms of the number of pedestrians per square meter. In many data-driven simulations, we populate the scene with a large number of virtual pedestrians. It is important that their movement or density/velocity flow resembles that of real-world crowds, which is captured by the Fundamental Diagram. The DDF computes a new steering angle and the preferred speed of a pedestrian based on the crowd density in its neighborhood.

We can easily combine DDPD with DDF for dense crowd simulation. A pedestrian with radius r chooses θ to minimize the distance to its intermediate goal $\mathbf{g}$ during the time period τ

$$\arg\min_{\theta} \| \mathbf{g} - (\mathbf{p} + \mathbf{v}^{FD_\theta})\tau \|, \tag{10}$$

where $\mathbf{v}^{FD_\theta}$ represents the Fundamental Diagram adherent velocity [3], which can be computed as:

$$\mathbf{v}^{FD_\theta} = \frac{\mathbf{v}^\theta}{s^{pref}}(\rho^\theta / 2r), \tag{11}$$

where $\mathbf{v}^\theta$ is the input preferred velocity, s^{pref} is natural walking speed of the pedestrian, and ρ^θ is the density around the pedestrian. Finally, $\mathbf{v}_i^{FD_\theta}$ is the new preferred velocity of that pedestrian that is used as an input to the local navigation module.

3.2.2 Situational Trajectory Adaptation

We use a variation of a stressor-based behavior model proposed in [18] to generate heterogeneous behaviors and dynamic behavior changes in response to an environment or a situation. In the original work, the behavior changes are modeled by updating various simulation parameters (e.g., personal radius, speed, number of neighbors, etc.) of an agent who is affected by one or more stressors of many kinds, such as time pressure, area stressors, positional stressors, interpersonal stressors, etc. [18]. We approximate this model to use with various multi-agent methods in general.

We assume that a pedestrian is experiencing a perceived stress with a value of ψ. The perceived intensity of the stressor, ψ, can be different for each pedestrian, depending on the type of the stressor and/or the distance between the pedestrian and the stressor. Our goal is to compute a stress response for an agent, denoted as S, that follows the General Adaptation Syndrome (GAS) theory. We use an approximated GAS model that is formulated as follows:

$$\frac{dS}{dt} = \begin{cases} \alpha & \text{if } \psi > S \\ \{-\alpha \leq \frac{d\psi}{dt} \leq \alpha\} & \text{if } \psi = S \\ -\alpha & \text{if } \psi < S \end{cases} \tag{12}$$

where S is capped at a maximum rate α and at a maximum amount β. We also assume that pedestrians tend to move away from the stressor. Let us denote the vector with the direction away from the stressor as v^{st}. The new adjusted preferred velocity v^{adj} is represented as a weighted sum of v^{st} and the agent's preferred velocity v^{pref}, with preferred speed s^{pref} scaled and based on the magnitude of S:

$$v^{adj} = S(1 + s^{pref})((1 - \frac{S}{\beta})\|v^{pref}\| + \frac{S}{\beta}\|v^{st}\|). \tag{13}$$

Finally, $\mathbf{v}^{adj}$ is the new preferred velocity of the pedestrian who is affected by the stressor.

3.3 User Interactions

In addition to interactions with the agents, by adding stressors or increasing densities during the simulation, our method can be extended to allow users to directly control any pedestrian in the scene or be part of the crowd. In particular, we modify the collision avoidance behavior of the virtual agents towards the user. Our algorithm is based on the asymmetric behavior modeling that is based on social forces and reciprocal velocity obstacles [6]. Between the virtual agents, the local navigation algorithm assumes symmetric behavior, which implies that all the pedestrians have equal responsibility of avoiding a collision. However, we impose 100% of collision-avoidance responsibility to the user or the user-controlled character in the simulation, when it has an impending collision with other pedestrians or obstacles. More details are given in [17].

4 RESULTS

In this section, we describe our implementation 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 velocity-based reasoning as the state transition model, $f()$. For collision-avoidance computation, we use a publicly available library [41], and we use different real pedestrian tracking datasets corresponding to indoor and outdoor environments as the input for the DDPD computation algorithm. These datasets are generated using manual tracking, online multiple-person tracker, KLT tracker, synthetic data and 3D range sensor tracking [23, 47, 4]. Our approach makes no assumptions in connection with the underlying tracking algorithm that is used to generate these datasets. Table 1 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. We modeled the objects in these scenes using Maya and rendered the results using GOLAEM, a commercially available crowd rendering platform. For real-time renderings for the user study, we used Unreal game engine.

4.1 Scenarios

We used different indoor and outdoor scenarios to highlight the performance of our crowd simulation algorithm.

Street: This scenario illustrates the basic notion of our data-driven algorithm. We use smooth, manually annotated trajectories of 147 pedestrians [21]. In the resulting simulation, we compute the trajectories of pedestrians whose entry points and movement patterns are similar to those of the tracked pedestrians. In this case, DDPD captures both time-varying entry-point distributions and movement patterns and show that the virtual pedestrians have the same characteristics (See Video) .

Manko: We highlight the similar behaviors of the virtual pedestrians compared with those of tracked pedestrians. Furthermore, we generate crowd trajectories and behaviors using different options: (a) crowd simulation with a similar scene-setup, but the virtual pedestrian trajectories are generated without DDPD, and we instead use random positions for the entry points; (b) computing the entry points and movement flow for virtual pedestrians using DDPD. The benefits of our learning algorithm are shown in the video (see Figures 3) as the virtual pedestrian follow the lanes in the original video. In contrast, virtual agents simulated using method (a) often result in collisions when going against the flow.

Marathon: We show a *structured scenario* in which pedestrian movement is somewhat uniform. In the marathon scenario, the runners follow an arc-shaped route (see Figure 4). Although the motion pattern looks simple, it is hard to model this scenario using one fixed set of entry points and goal positions, which is commonly used in agent-based models. Our DDPD algorithm eliminates manually adding and adjusting entry/goal points and automatically computes the preferred velocity over different time periods. We use only 18 trajectories from the original video and employ them to compute DDPD. These characteristics are used to compute the trajectories of 500 virtual pedestrians as part of the marathon. We further compare our approach with other agent-based simulation algorithms that use many set of intermediate goals and offline parameter learning instead of DDPD (Figures 4 (b) and (c)). The different simulation results are shown in the video.

Black Friday Shopping Mall: In this scenario, we show that our DDPD approach can be combined with other agent-based methods that simulate various aspects of human behavior, as described in Section 4.1.1. We take a shopping mall scene generated from given trajectories and vary the pedestrian density. We combine DDPD with DDFs [3] and highlight the results in the video (See Figure 5). We also highlight the benefits of *interactive computation* of DDPD.

(a) (b) (c)

Figure 5: **Black Friday Shopping Mall scenario:** (a) We use six simple trajectories generated in an empty space. (b) and (c) We create a new layout by adding obstacles interactively, and also increase the number of pedestrians in the scenario. (b) has 271 pedestrians, (c) has the twice higher number of pedestrians generated at the same rate.

In particular, we add new obstacles during the simulation, and the data-driven crowd simulation algorithm can vary the trajectories to avoid collisions with the obstacles.

Explosion: We combine our method with GAS model *dynamic behavior changes and situational trajectory adaptation*. This scenario highlights how the trajectories and behaviors of the pedestrians change in response to an explosion. In this scenario, the pedestrians initially follow the DDPD computed from the extracted trajectories. We represent an explosion as an external event and model that using a stressor. We use the stressor-based behavior model [18] to compute the new preferred velocity for each pedestrian in the scene. We model a stressor from an explosion as a positional stressor and change the behavior of the virtual pedestrians based on the amount of stress that the explosion generates (See Figure 6).

(a) (b)

Figure 6: **Crossing Explosion Scenario:** (a) Initially, virtual pedestrians follow the DDPD computed from the extracted trajectories. (b) The user places an explosion during the simulation. The pedestrians who perceive the explosion begin running away from it in different directions.

Train Station: We demonstrate the performance of DDPD on noisy, tracked data. For this scenario, we use tracklets generated by a KLT tracking algorithm. Instead of tracking each individual agent, the KLT tracker computes particles that belong to one or more tracked pedestrians in the video. The length of the trajectories range from a few to tens of frames. The tracklets are noisy, and we compute the DDPD from these tracklets, using them to generate the trajectories of virtual pedestrians (see Figure 7 and the supplementary video). Our approach can produce the same movement flow as the original video. Moreover, we also highlight gradual density changes in the same environment and/or when we change the layout.

ATC Mall: This scenario demonstrates the performance of our algorithm with different sensor data. In this case, we use the 50 trajectories extracted in a shopping mall scenario using 3D range sensors, which comprises a sparse dataset with noisy trajectories. We compute the DDPD to generate the trajectories of 207 virtual pedestrians that exhibit similar time-varying movement patterns (see Video).

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

Figure 7: **Train Station scenario:** (a) Original Video. We use the tracklet data computed by the KLT algorithm as an input to our algorithm. We compute the pedestrian dynamics from the noise trajectories. (b) A frame from our data-driven crowd simulation algorithm in the same layout. (c) Crowd simulation in a modified layout as compared to the original video. (d) We increase the number of pedestrians in the scene. They have similar movement patterns as the original video and are combined with density-dependent filters for plausible simulation.

Scenario	Sensor	# Tracked Peds.	# Virtual Peds.	# Static Obst.	# Input Frames	Avg. time DDPDL	Avg. time Traj. Comp.
Manko	Online Tracking	42	70	0	373	0.075	5.63e-05
Marathon	Online Tracking	18	500	33	450	0.04	1.56E-03
Explosion	Online Tracking	19	110	0	238	0.03	2.56E-04
Street	Manual Tracking	147	167	0	9014	0.012	3.27E-06
Train Station	KLT (Tracklets)	200	200-943	37-47	999	0.05	4.93E-04
ATC Mall	3D Range Sensors	50	207	38	7199	0.023	2.97E-03
BlackFriday	Synthetic Data	6	271-1000	20-28	109	8.58E-03	2.75E-05

Table 1: Performance 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. Our learning and trajectory computation algorithms can be used for interactive crowd simulations.

5 User Studies and Evaluation

In this section, we evaluate the perceptual improvements achieved by our algorithm by performing user studies in two different VR setups based on 2D monitors and 3D HMDs. In particular, we asked the users to compare the trajectory behaviors generated by different interactive crowd simulation algorithms with those observed in real-world videos. The two algorithms are:

- RVO-based synthetic multi-agent simulation [41] that is widely used in games and virtual environments. We also used an enhanced version that uses offline parameter learning to improve the trajectory behaviors [43, 2].
- Adaptive data-driven algorithm that combines RVO with DDPD (see Section 4).

Objectives: Our main goal was to measure how close are the trajectory behaviors generated using our pedestrian dynamics learning algorithm to those observed in real-world videos. This includes evaluating the potential benefits of our pedestrian dynamics learning algorithm in two commonly used VR setups. Secondly, we want to measure the benefits of adaptive data-driven simulation algorithm over a synthetic crowd-simulation algorithm.

Hypothesis: The crowd trajectory behaviors generated by *our* adaptive data-driven algorithm would have a higher perceptual similarity to those observed in the original video, as compared to the ones generated using prior synthetic crowd simulation algorithm.

5.1 Experiment Scenarios

In order to evaluate different crowd simulation algorithms, we considered three different scenarios.

Scenario 1: We compare the simulation results generated using synthetic algorithm with and without DDPD, including entry point distributions and movement flow learning. We use an outdoor walking scene with pedestrian cross flows. There are around 40 pedestrians in the original video and around $60-70$ virtual pedestrians in the synthetically generated and rendered crowd simulation videos. (i) We use random entry points for virtual pedestrians and simulate their movements towards the other side of the street. (ii) We use entry point distributions to generate initial positions of the pedestrians, along with learned movement flows to compute pedestrian trajectories.

Scenario 2: In this scenario, we compare an offline parameter learning method and our DDPD algorithm that uses time-varying pedestrian dynamics. The scenarios corresponds to an outdoor running scene with 500 pedestrians in both the original video and the ones generated using crowd simulation algorithms. (i) We use the local navigation algorithm with optimized motion parameters and intermediate sub-goals; (ii) We use the learned movement flows from DDPD to simulate pedestrians. (i) and (ii) use the same entry point distributions learned from the extracted trajectories and same number of virtual pedestrians.

Scenario 3: In this scenario, we mainly focus on the movement flow learning component of our DDPD algorithm. We compare with the synthetic multi-agent algorithm with partial DDPD information based on entry point learning. We used an outdoor scene with sudden changes in the velocity of pedestrian. There were 50 pedestrians in the original video and around 100 virtual pedestrians in the videos generated using different crowd simulation algorithms. (i) We compute the movement flows corresponding to randomly chosen goal positions. (ii) We use the learned movement flows from the videos to simulate virtual pedestrian.

5.2 User Studies with 2D Monitors

For each scenario, we asked the users to compare the trajectories in the original crowd video with two synthetically generated crowd simulations. All the videos were about $15-20$ seconds long. Every user is asked to compare the movement patterns in the original video with those in the synthetically generated crowd simulations, by playing them simultaneously. We asked the users to first watch

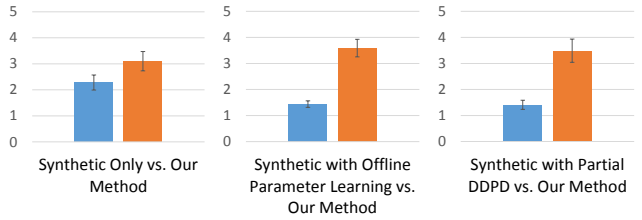


Figure 8: **Comparison of similarity scores for 2D screen** (higher is more similar) We compared the trajectory behaviors generated using the RVO-based synthetic algorithm with different configurations (blue) and our algorithm (synthetic algorithm + DDPD, shown in orange). These results indicate that the use of our pedestrian dynamics learning algorithm to compute entry points and movement flows considerably improves the perceptual similarity of our simulation to the pedestrian movements in the original video.

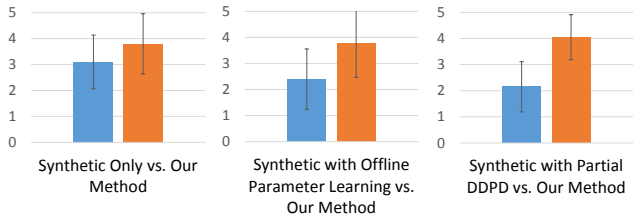


Figure 9: **Comparison of similarity scores for HMD** (higher is more similar) We compare crowd simulation result generated using RVO-based synthetic algorithm with different configurations (blue) and our method (synthetic algorithm + DDPD, orange bars in the graphs). Similar to the 2D screen user study, our method got higher scores.

the original video and then rate each synthetic video on a scale of $1-5$ in terms of comparing the similarity of movement patterns between the original video and a synthetic video. A score of 1 indicated most dissimilar and a score of 5 indicated most similar movement pattern. Since this was a perceptual study related to movement patterns and trajectory behaviors, we intentionally did not explicitly define or asked the user to classify the crowd behavior. Furthermore, we asked the users not to rate the video based on the rendering quality, foot-sliding, orientation issues, etc. We also encouraged the users to watch the original video and the synthetic video as many times as he/she wants and finally provide some (optional) feedback. There were 39 participants (51.61% female, 64.52% in the age group of $20-30$) for this study on a 2D computer monitor. The study had no time constraints and the participants were free to take breaks in-between the benchmarks as long as the web-session did not expire.

We measured the mean, variance of their scores, and computed the p-values using a two-tailed t-test (See Figure 8). The p-values for scenario1, scenario2, and scenario3 comparisons were $2.53e^{-05}$, $5.06e^{-21}$, and $1.71e^{-16}$, respectively. We observed that the crowd simulations generated by our DDPD learning algorithm scored much higher than the other two approaches for all scenarios, at a statistically significant rate (p-value<0.05). This also demonstrates that both components of DDPD, entry point learning and movement flow learning are important to generate realistic crowd trajectories.

5.3 User Studies with 3D HMDs

We used the Oculus Rift Development Kit 2 HMD, with 1080p resolution and positional tracking to render the crowd scenes generated using different algorithms.

Every user was randomly presented with these scenarios and synthetically generated crowd visualizations on an HMD for each scenario. The participants were able to freely navigate through the environment, but there was no interactions (e.g. collision avoidance) between the virtual pedestrians in the scene and the participants. In other words, they were passively observing the virtual crowds. Most of the users were asked to observe the agent trajectory behaviors from a distance. Each viewing experience was about $15-20$ seconds long. Similar to the 2D setup, the participants were asked to rate the quality of trajectories and movement patterns on a similarity score. The questionnaire used were the same as the 2D screen user study. There were 21 participants (57% female, 85.7% in the age group of $20-30$) for the study on an HMD. The participants for 3D HMD-based user study do not overlap with the 2D screen study. As a result, there was no learning effect for our user study.

We measured the mean, variance of their scores, and compute p-values using a two-tailed t-test (See Figure 9). The p-values for scenario1, scenario2, and scenario3 comparisons were 0.06, $1.0e^{-3}$, and $1.0e^{-5}$, respectively. We observed that the renderings of crowd simulations generated using our DDPD learning algorithm scored much higher than the other two approaches for scenario 2 and scenario 3, at a statistically significant rate (p-value<0.05). However, in Scenario 1, the p-value was 0.06 and therefore we were unable to derive statistically meaningful conclusions. We also observed that the variances of scores for the HMD-based user study were higher than those of 2D screen setup.

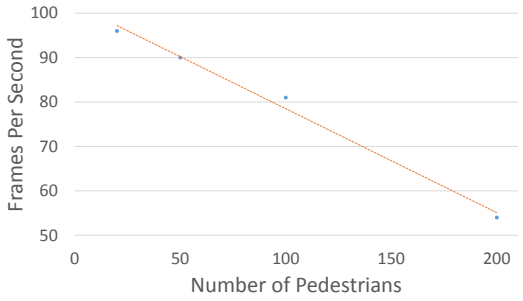


Figure 10: **Runtime performance of our user study scenarios.** Blue dots indicate the actual measurement and the orange line shows the linear approximation of the performance as the number of pedestrian increases.

We conjecture a possible explanation for these results based on the user studies performed in [9]. The viewing angle (first-person, eye level view, see Figure 1) gives less information about the environments and overall crowd behaviors than a top-down view. In general, the participants also preferred a bird-eye view as opposed to an eye-level view.

6 Analysis and Comparisons

In this section, we analyze the performance of our approach and compare it with previous methods. The DDPD were 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 many 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), high-density simulation (Train Station scenario).

6.1 Comparisons

Many prior agent-based techniques in the literature are known to compute collision-free trajectories of virtual pedestrians. The simplest methods use random points to compute entry points and goal destinations and compute collision-free paths for virtual pedestrians using any collision avoidance algorithm (e.g., social forces or reciprocal velocity obstacles). However, the resulting trajectories may not exhibit the same movement patterns or trajectories as real pedestrians, as observed in some benchmarks (see Figure 4(c)).

Parameter Learning: Recently, many offline optimization techniques have been proposed to learn the optimal motion parameters for parameterized multi-agent simulation algorithms using real-world trajectories [32, 43, 2]. These techniques improve the trajectory behaviors of virtual pedestrians with respect to resembling the tracked pedestrians. However, we still need good techniques to estimate the entry point, movement flow and/or the goal position for each agent (see Figure 4(d)). In many cases, entry points and destinations are specified or selected from manually pre-annotated regions in the scene [31, 47]. These parameter learning methods can be used to improve the interaction model $I()$ of our pedestrian dynamics model.

Offline Behavior Learning : Most prior work in computer vision and robot-human interactions uses offline learning methods from the training data to compute a fixed set of parameters that correspond to a global characterization [47, 14]. Many prior techniques based on manual tracking or manually annotated behavior labels [23] have also used for data-driven simulations, but they can be time consuming and limited to offline applications. On the other hand, our approach is interactive and automatic and uses one video source for trajectory data. Moreover, offline methods may not work well when there are large variations in individual behaviors or trajectories. Unlike prior offline methods, our approach does not learn fixed sets of destination points or sub-goals and can capture time-varying movement characteristics (see Figure 4(e)). Many offline methods based on multi-character motion synthesis [20, 44] can generate more realistic crowd simulations with multiple interacting characters. By contrast, our approach only uses extracted 2D trajectories and only generates the trajectories of virtual pedestrians. Our approach cannot perform simulations of full body motions or gestures.

Interactive applications: Compared to prior interactive data-driven methods [1], our approach captures the motion dynamics of the tracked pedestrians. This makes it possible to generate denser crowd simulations as well as use them for varying environments and situations. Furthermore, our approach is automatic and involves no user editing. There has been a pure simulation method to generate large and dense crowd trajectories at near interactive rates [25]. However, this method is a pure simulation method and it uses a macroscopic formulation. Our method, on the other hand, uses a multi-agent formulation and uses the pedestrian trajectory characteristics extracted from real videos.

Virtual Crowds with Real Videos: Many techniques have been proposed to insert virtual crowds as an overlay on a real-world video. This method uses an efficient algorithm for coupling camera-tracked humans with virtual agents [35]. Grid-based density field have been used to overlay several simulated virtual agents in real-world video [33]. Pellegrini et al. [31] simulate virtual agents automatically by adjusting the virtual agents' behavior to follow pre-learned behaviors. In contrast with these methods, our approach computes the trajectories for the real-world and virtual pedestrians. As a result, our approach is more flexible and can generate many complex scenarios highlighted in Table 1.

7 Conclusions, Limitations and Future Work

We present an interactive approach to learning the characteristics of pedestrian dynamics 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 is only useful for capturing the characteristics of local pedestrian dynamics for each pedestrian, whereas offline learning methods can compute many global characteristics. We consider only a few movement characteristics to compute the trajectories of virtual pedestrians and do not take into account other aspects of pedestrian behaviors or state, full body actions or the interactions among pedestrians. Furthermore, our approach may not work well if the layout of the obstacles in the virtual environment is quite different from that captured in the original video, e.g., some objects have been removed and that can result in different pedestrian behaviors. Since our approach only computes the trajectories, we also need to combine with techniques that can generate plausible animation and rendering. For example, we used GOLAEM and Unreal Engine for offline and real-time rendering, respectively. While the offline rendering software can result in higher quality rendering, we observe some artifacts especially for dynamic scenes. More specifically, we observe some discrepancies when the pedestrians change directions rapidly, such as in the explosion scenario, and the resulting pedestrians appear to be colliding even though the trajectories are collision-free. This is possible due to the underlying interpolation scheme and smoothness constraints that are based on a fixed number of pre-recorded walk cycles. Note that the adaptive data-driven algorithms described in Section 4 are conservative. In such cases, the agents can just touch the other agents that appears as collisions.

Future Work: There are many avenues for future work. In addition to overcoming the limitations of our work, our interactive DDPD 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 [20, 44].

In this paper, we focused on the use of our algorithm to generate realistic crowd trajectories and behaviors for 2D screens as well as 3D immersive environments (e.g. HMD). As a result, we performed user studies with both setups to get some preliminary evaluation results. We would like to use our approach for more VR-applications and perform more evaluations.

Acknowledgements

This work was supported by NSF awards 1117127, 1305286, ARO contract W911NF-14-1-0437, and a grant from the Boeing company.

References

[1] A. Bera, S. Kim, and D. Manocha. Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In *Proceedings of Graphics Interface*, pages 65–72, 2015.

[2] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos. Steerfit: Automated parameter fitting for steering algorithms. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, pages 113–122, 2014.

[3] A. Best, S. Narang, S. Curtis, and D. Manocha. Densesense: Interactive crowd simulation using density-dependent filters. *Symposium on Computer Animation*, pages 97–102, 2014.

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

[5] J.-R. Chen. *Planning plausible human motions for navigation and collision avoidance*. PhD thesis, UCL (University College London), 2014.

[6] S. Curtis, B. Zafar, A. Gutub, and D. Manocha. Right of way. *The Visual Computer*, pages 1–16, 2012.

[7] J. Diemer, G. W. Alpers, H. M. Peperkorn, Y. Shiban, and A. Mühlberger. The impact of perception and presence on emotional reactions: a review of research in virtual reality. *Frontiers in psychology*, 6, 2015.

[8] F. Durupinar, N. Pelechano, J. Allbeck, U. Gü anddü andkbay, and N. Badler. How the ocean personality model affects the perception of crowds. *Computer Graphics and Applications, IEEE*, 31(3):22 –31, may-june 2011.

[9] C. Ennis, C. Peters, and C. O'Sullivan. Perceptual effects of scene context and viewpoint for virtual pedestrian crowds. *ACM Trans. Appl. Percept.*, 8(2):10:1–10:22, Feb. 2011.

[10] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *PAMI*, pages 2179–2195, 2009.

[11] S. J. Guy, S. Curtis, M. C. Lin, and D. Manocha. Least-effort trajectories lead to emergent crowd behaviors. *Phys. Rev. E*, 85:016110, Jan 2012.

[12] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Symposium on Computer Animation*, pages 43–52. ACM, 2011.

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

[14] T. Ikeda, Y. Chigodo, D. Rea, F. Zanlungo, M. Shiomi, and T. Kanda. Modeling and prediction of pedestrian behavior based on the sub-goal concept. In *Robotics: Science and Systems'12*, page 504, 2012.

[15] M. Kapadia, I.-k. Chiang, T. Thomas, N. I. Badler, and J. T. Kider, Jr. Efficient motion retrieval in large motion databases. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 19–28, 2013.

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

[17] S. Kim, A. Bera, and D. Manocha. Interactive pedestrian dynamics learning from trajectory data. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 2015.

[18] S. Kim, S. J. Guy, D. Manocha, and M. C. Lin. Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Symposium on Interactive 3D Graphics*, I3D '12, pages 55–62. ACM, 2012.

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

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

[21] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[22] A. Lerner, Y. Chrysanthou, A. Shamir, and D. Cohen-Or. Data driven evaluation of crowds. In *Motion in Games*, pages 75–83. Springer, 2009.

[23] A. Lerner, E. Fitusi, Y. Chrysanthou, and D. Cohen-Or. Fitting behaviors to pedestrian simulations. In *Symp. on Computer Animation*, pages 199–208, 2009.

[24] Y. Li, M. Christie, O. Siret, R. Kulpa, and J. Pettré. Cloning crowd motions. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 201–210, 2012.

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

[26] A.-H. Olivier, J. Bruneau, G. Cirio, and J. Pettr. A virtual reality platform to study crowd behaviors. *Transportation Research Procedia*, 2:114 – 122, 2014. The Conference on Pedestrian and Evacuation Dynamics.

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

[28] S. Patil, J. van den Berg, S. Curtis, M. Lin, and D. Manocha. Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):244 –254, feb. 2011.

[29] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *Symposium on Computer animation*, pages 99–108, 2007.

[30] N. Pelechano, C. Stocker, J. Allbeck, and N. Badler. Being a part of the crowd: Towards validating vr crowds using presence. In *Proc. of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 136–142. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[31] S. Pellegrini, J. Gall, L. Sigal, and L. Gool. Destination flow for crowd simulation. In *ECCV Workshops and Demonstrations*, volume 7585, pages 162–171. 2012.

[32] J. Pettré, J. Ondřej, A.-H. Olivier, A. Cretual, and S. Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Symposium on Computer Animation*, SCA '09, pages 189–198. ACM, 2009.

[33] Z. Ren, W. Gai, F. Zhong, J. Pettré, and Q. Peng. Inserting virtual pedestrians into pedestrian groups video with behavior consistency. *The Visual Computer*, 29(9):927–936, 2013.

[34] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87*, pages 25–34. ACM, 1987.

[35] J. Rivalcoba, O. Gyves, I. Rudomin, and N. Pelechano. Coupling pedestrians with a simulated virtual crowd. In *Proc. of the International Conference on Computer Graphics and Applications (GRAPP'2014)*, January 2014.

[36] J. Sewall, D. Wilkie, and M. C. Lin. Interactive hybrid simulation of large-scale traffic. *ACM Trans. Graph.*, 30(6):135:1–135:12, Dec. 2011.

[37] H. P. H. Shum, T. Komura, and S. Yamazaki. Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):741–752, May 2012.

[38] M. Taffou. *Inducing feelings of fear with virtual reality: the influence of multisensory stimulation on negative emotional experience*. PhD thesis, Paris 6, 2014.

[39] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. In *ACM SIGGRAPH 2006*, pages 1160–1168. ACM, 2006.

[40] B. Ulicny and D. Thalmann. *Crowd simulation for interactive virtual environments and VR training systems*. Springer, 2001.

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

[42] V. Vinayagamoorthy, A. Brogni, M. Gillies, M. Slater, and A. Steed. An investigation of presence response across variations in visual realism. In *The 7th Annual International Presence Workshop*, pages 148–155, 2004.

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

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

[45] Q. Yu and D. Terzopoulos. A decision network framework for the behavioral animation of virtual humans. In *Symposium on Computer animation*, pages 119–128, 2007.

[46] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, pages 864–877. 2012.

[47] B. Zhou, X. Wang, and X. Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2871–2878, June 2012.