Learning Behavior Patterns from Video: A Data-driven Framework for Agent-based Crowd Modeling

Jinghui Zhong[1], Wentong Cai[1], Linbo Luo[2], and Haiyan Yin[1]

[1]School of Computer Engineering, Nanyang Technological University, Singapore
{jinghuizhong, aswtcai, haiyanyin}@ntu.edu.sg
[2]School of Cyber Engineering, Xidian University, China
lbluo@xidian.edu.cn

ABSTRACT

This paper proposes a generic data-driven crowd modeling framework to generate crowd behaviors that can match the video data. The proposed framework uses a dual-layer mechanism to model the crowd behaviors. The bottom layer models the microscopic collision avoidance behaviors, while the top layer models the macroscopic crowd behaviors such as the goal selection patterns and the path navigation patterns. Based on the dual-layer mechanism, an automatic learning method is proposed to learn the model components from video data. To validate its effectiveness, the proposed framework is applied to generate the crowd behaviors in New York Grand Central Terminal. The simulation results demonstrate that the proposed method is able to construct effective model that can generate the desired emergent crowd behaviors and can offer promising prediction performance.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence, Intelligent agents; J.4 [**Computer Applications**]: Social and Behavioral Sciences, Sociology; J.7 [**Computer Applications**]: Computers in Other Systems, Process control

General Terms

Algorithms

Keywords

Agent-Based Modeling; Crowd Simulation; Collision Avoidance Behavior; Data-Driven Modeling; Macroscopic Behavior

1. INTRODUCTION

Crowd modeling and simulation is an active research field that has drawn increasing attention from industry, academia and government recently [1, 5, 18, 21, 24, 32]. The most common approach to simulating crowd dynamics is the agent-based modeling (ABM) paradigm, which treats individuals as autonomous agents that can perceive information and make decisions independently based on their own behavior rules. Due to the significant increase in computing power, the agent-based crowd modeling has been developing rapidly over the past decade, with a range of successful applications in computer game, film design, evacuation planning and military training [3, 7, 15, 22, 30, 34].

One fundamental issue in agent-based crowd modeling is to find the correct parameter settings and behavior rules of agents, so that the simulation can match the behavior observed in the physical system. However, it is difficult to derive the agent behavioral rules in agent-based simulation because an agent's behavior is affected by its environment as well as the surrounding agents. Moreover, calibrating the parameter settings of agent behavior model is also a time-consuming and tedious process.

To address the above issues, various data-driven crowd modeling approaches have been proposed in the past few years. These approaches generally can be classified into two groups. The first group aims to learn examples (mostly in terms of state-action pairs) from video data. The examples learned are then used to update the movements of agents in particular situations so that the simulated behaviors look similar to the video data [9, 10, 11, 12, 17, 29]. Meanwhile, the second group aims to calibrate the model parameters through automatic methods such as evolutionary algorithms (EAs), so that the simulated behaviors can match the video data [8, 20, 27, 31]. Existing works have shown the potential of data-driven approaches in automatic crowd modeling, but they mainly focus on matching microscopic spatial crowd behaviors. In many practical scenarios (e.g., in a train station), the crowd behaviors are complicated and contain a number of macroscopic motion patterns, such as the goal selection pattern and the path navigation pattern. Some of the patterns are spatial while others are temporal. In these scenarios, considering only the microscopic spatial features is not sufficient to generate the desired emergent crowd dynamics.

To solve the above problem, a novel data-driven crowd modeling framework is proposed in this paper. The proposed framework makes use of a dual-layer agent-based modeling mechanism to model the behaviours of agents. The bottom layer focuses on microscopic collision avoidance behaviours. Meanwhile, the top layer models multiple spatial and temporal macroscopic patterns such as the goal selection patterns, the path navigation patterns, the distribution of preferred speed, and the frequency of pedestrians entering the area. By considering multiple spatial and temporal behavior patterns simultaneously, the proposed framework provides a general and flexible way to model complicated crowd behaviors in practice. Moreover, based on the dual-layer modeling mechanism, an automatic learning method is proposed to construct model components based on off-line video data. A refinement method to incrementally refine model components using on-line video is also discussed. To validate its effectiveness, the proposed data-

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

driven framework is applied to simulate the pedestrian behaviors in New York Grand Central Terminal. The simulation results have demonstrated the effectiveness of the proposed framework.

2. RELATED WORK

In recent years, as the automated technology of collecting and processing real-world crowd data become more and more prevalent, the research on data-driven crowd modeling has gained increasing attention from researchers, and a number of data-driven modeling approaches have been proposed up to now.

The most common class of existing data-driven modeling approaches focus on deriving microscopic trajectory patterns and using the derived patterns to drive agents' movements. Typically, patterns are defined by examples, but the way how an example is defined differs. The way to determine which example is to be used during the simulation to drive agent's movement is also different. For example, Lee et al. [10] proposed to use a hierarchical model to generate behaviors of agents. They used a finite state machine to control the transitions between high-level activities such as joining or leaving a group, while using state-action pairs to control the low-level motions of agents. Each state-action pair determines the future velocity for agents under particular situations. Their work focused on learning state-action pairs from video to control the low-level motion of agents. Eunjung et.al. [9] proposed a data-driven crowd modeling approach which can construct a large scale crowd that looks similar to the input video, in terms of crowd formation and moving styles. In their model, a database of formations and trajectory segments are firstly extracted from video. Then each agent iteratively selects a trajectory segment from the database to extend its moving trajectory so that it dose not collide with other agents and the formation constraint is also satisfied. Musse et al. [17] proposed a data-driven crowd modeling approach for simulating human behaviors in normal life. They extracted trajectories from videos to learn the desired velocities of agents in different regions. Lerner et al. [11] proposed a data-driven crowd modeling method which requires extracting a database of examples from video in advance. Each example is a short trajectory segment. During the simulation, when the trajectory segment of an agent is exhausted or there is a significant change in the current situation, the agent will query a similar example to extend its trajectory. Zhao et al. [29] proposed a data-driven crowd modeling approach also using examples extracted from video. In their method, each example consists of the surrounding state information and the corresponding moving velocity. An Artificial Neural Network (ANN) is trained to classify the examples into several groups. In each time step of the simulation, the trained ANN classifies each agent into one of the groups according to the input state perceived by the agent, and a suitable example is selected from the group to update the velocity of the agent.

Some of the existing works focus on designing automatic methods to calibrate model parameters using video data so that the simulated behaviors can match with those in the video. For example, Johansson et al. [8] used an EA to tune parameters of the commonly used crowd model– the social force model (SFM) [7]. Similarly, in [19, 20] and [27], optimization algorithms such as the Gradient-based Newton method and the Genetic Algorithm (GA) are used to calibrate parameters of SFM and the reciprocal velocity obstacles (RVO) model [25] respectively. Yamaguchi et al. [28] formulated a behavior model by considering multiple personal and social factors. Machine learning techniques were designed to automatically estimate the setting of each factor so as to fit the video data. Zhong et al. [31] proposed an EA to calibrate parameters of the SFM so that the crowd density distribution in the simulation can match with those in the video.

The existing data-driven crowd modeling approaches mainly focus on matching microscopic spatial behavior features. Different from the existing approaches, our work focuses on matching multiple macroscopic features, both spatial (e.g., the path navigation patterns) and temporal (e.g., the frequency of pedestrians entering an area). In this way, our proposed framework provides a flexible way to model the complicated crowd behaviors in practice.

In the computer vision community, there has been an increasing amount of work on learning global motion patterns of crowds from video data [2, 13, 14, 16, 4, 33]. For example, Ali and Shah [2] proposed a lagrangian particle dynamics approach to detect and segment crowd flows in dense scenarios. Lin et al. [13, 14] proposed a lie algebraic approach to model crowd flows from video. Mehran et al. [16] proposed a streakline representation of crowd flow which helps to more accurately recognize spatial and temporal changes in the video scene. Weina et al. [4] proposed a method to detect groups of individuals which are travelling together in video scenes. Zhou et al. [33] proposed to use Linear Dynamic System (LDS) to model the moving pattern of pedestrians in the video data. They utilized an Expectation Maximization (EM) algorithm to learn parameter settings of the LDS. However, these works focus on detecting collective motion patterns such as flow field from video data. They have not been integrated with the agent-based crowd models such as SFM and RVO to generate realistic crowd behaviors. The research on integrating multi-agent crowd modeling techniques with computer vision techniques for automatic crowd modeling is still in its initial stage. This paper goes a step further in this direction.

3. DATA-DRIVEN CROWD MODELING

3.1 Model Definition

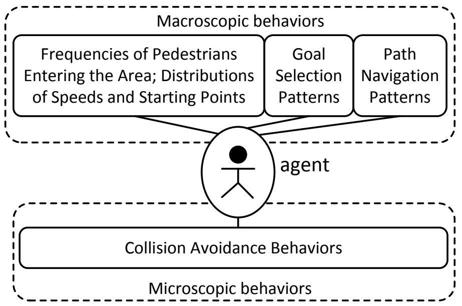

Macroscopic behaviors
Frequencies of Pedestrians Entering the Area; Distributions of Speeds and Starting Points
Goal Selection Patterns
Path Navigation Patterns
agent
Collision Avoidance Behaviors
Microscopic behaviors

Figure 1: The proposed dual-layer agent-based crowd model.

This paper focuses on simulating crowd movements in a well defined area with a number of entry regions (SRs) and exit regions (ERs). A pedestrian enters the area via an IR and leaves the area via an OR. Such a scenario is very common in public places such as a train station. In such a scenario, the behaviors of pedestrians are mainly dependent on four factors: 1) how pedestrians enter the area, 2) how pedestrians select their goals, 3) how pedestrians navigate to their goals, and 4) how pedestrians avoid collisions with other pedestrians and obstacles. In this paper, a dual-layer agent-based crowd modeling framework is proposed to model the behaviors of pedestrians in such a scenario. As shown in Fig. 1, the bottom layer models the last factor (i.e., the collision avoidance behaviors), while the top layer adopts three components to model the other factors respectively.

Specifically, in the bottom layer, there are various collision avoidance models that can be used, such as the SFM and the RVO2.

In this paper, the RVO2 [25], which was proposed by Guy et al. in 2008, is adopted, owning to its highly effectiveness in generating realistic collision avoidance behaviors [27]. In the RVO2, each agent takes into account the observed velocities of other agents to select an optimal velocity. For an agent, each neighboring obstacle is assigned with an optimal reciprocal collision avoidance line. This line is used to determine the feasible velocity space that can ensure no collision occurs between agent and obstacle in a given time period. Then the optimal collision-avoidance velocity is obtained using linear programming. The readers are refereed to [25] for the details of RVO2.

The top layer models the macroscopic behaviors of the crowd by using three components. The first component describes how new pedestrians enter the simulated area. Each SR (or ER) is approximately represented by a rectangle region. Inspired by [33], the behaviors of new pedestrians appearing in each SR is modeled as a Poisson process:

$$f(k_i;\lambda) = \frac{\lambda^{k_i} exp(-\lambda)}{k_i!} \tag{1}$$

where k_i is the number of new pedestrians entering the i-th SR during a time period of λ. The starting positions of pedestrians entering each IR are different from one another, so do the preferred speeds of pedestrians. For simplicity, this paper uses the Gaussian distribution to model the starting positions of pedestrians in each SR and the preferred speeds of pedestrians respectively.

The second components models the goal selection patterns of pedestrians by using a probability graph. The probability of pedestrians moving from the i-th SR to the j-th ER is labelled as $p_{i,j}$. In this way, the goal selection patterns of pedestrians can be represented by the following matrix:

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & ... & p_{1,M} \\ ... & ... & ... & ... \\ p_{N,1} & p_{N,2} & ... & p_{N,M} \end{bmatrix}. \tag{2}$$

where N is the number of SRs and M is the number of ERs.

The last component models the path navigation patterns of pedestrians. In this paper, we propose to use velocity field to control the moving directions of pedestrians. The velocity field consists of a series of position-direction pairs. Each position-direction pair determines the future moving directions of pedestrians near the position. Note that pedestrians at a same position can have different moving directions, if their goals are different. Hence, in the proposed method, each ER is associated with one velocity field. Pedestrians choosing an ER as their goals will be navigated by the corresponding velocity field. To construct the velocity fields of all ERs, the entire region is divided into discrete grids, with each grid having a size of $\Delta \times \Delta$. Suppose the entire region is divided into $H \times W$ grids, then the velocity field of the i-th ER can be represented by the following matrix:

$$\mathbf{V}_i = \begin{bmatrix} v_{i,1,1} & v_{i,1,2} & ... & v_{i,1,W} \\ ... & ... & ... & ... \\ v_{i,H,1} & v_{i,H,2} & ... & v_{i,H,W} \end{bmatrix}. \tag{3}$$

where $v_{i,j,k}$ is a two-dimensional vector which represents the future moving direction of the pedestrians in the grid cell (j,k). The value of $v_{i,j,k}$ is scenario specific and will be learned from the input video data. Since the moving pattern of pedestrians from one SR to an ER is determined by a series of position-direction pairs, using velocity field is very flexible to generate complicated moving patterns of pedestrians such as the “S" shape moving patterns.

In the next subsection, we will describe how the above model components such as k, **P** and $\mathbf{V}_i$ can be obtained from video data.

Once the model components are constructed, the crowd simulation can be carried out as described in Algorithm 1. In Algorithm 1, the global goal of an agent is the ER selected based on **P**. The local goals of an agent are viewpoints to guide the agent to its global goal. Suppose an agent selects the i-th ER as the global goal. The current position of the agent is x, and the grid cell (j,k) contains the agent. Then the local goal of the agent at x is set equal to the center point grid cell (j,k) plus $v_{i,j,k}$. At each time step of the simulation, a number of new agents are first generated according to the Poisson process of each SR. The starting positions and velocities of new agents are initialized accordingly based on the statistical information extracted from video data. The global goal of each new agent is set according to **P**. Then the local goal of each agent is obtained based on the corresponding velocity field. Finally, the RVO2 is used to update the position and speed of each agent so that all agents can reach their local goals.

Algorithm 1: CROWD SIMULATION PROCEDURE.

```
1 for $step = 1$ to $max\_steps$ do
2     for $i = 1$ to $M$ do
3         Sample a number of new agents according to $k_i$;
4         Set the starting position and velocity of each new
          agent;
5         Set the global goal of each new agent according to $\mathbf{P}$
6     for $i = 1$ to total number of agents in the scene do
7         Suppose the global goal of the $i$-th agent ($A_i$) is the
          $j$-th ER;
8         Update the local goal of $A_i$ according to $\mathbf{V}_j$
9     Use RVO2 to update the velocities and positions of all
      agents.
```

3.2 Construction of Model Components

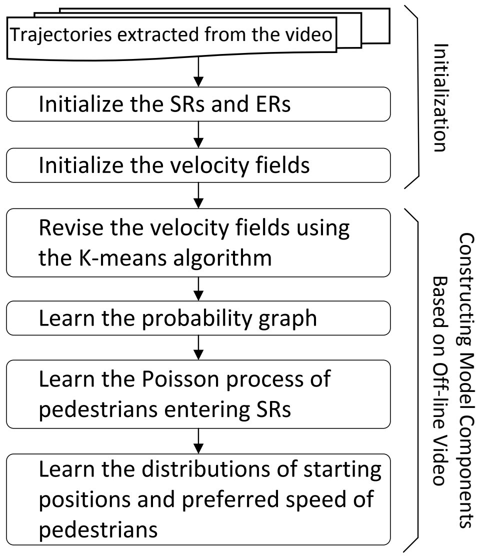
This section proposes an automatic method to construct model components, so that the simulation can fit the video data. We focus on constructing the three components in the top layer, while the bottom layer is configured using the default values of RVO2. This is because that the top layer has a much stronger influence on the macroscopic behaviors while the default settings of RVO2 have been shown effective to generate realistic microscopic collision avoidance behaviors. Specifically, we consider that the data extracted from videos are trajectories, each of which is represented by a series of vectors:

$$\begin{aligned} T &= < e_1, e_2, ..., e_n > \\ &= < (x_1, y_1), ..., (x_n, y_n) > \end{aligned} \tag{4}$$

where $e_i = (x_i, y_i)$ represents the position of the i-th point of the trajectory, and n represents the total number of points of the trajectory. In this study, the trajectories are extracted by the KL keypoint tracker [23]. Due to occlusion and scene clutters, most trajectories extracted from the video in dense crowd scenarios are short and highly fragmented. Our objective is to construct the model components based on these extracted trajectories. As shown in Fig. 2, procedures to construct the model components are as follows.

3.2.1 Initialization

Firstly, to reduce the training time, each trajectory is divided into a small number of segments (denoted as $< g_1, g_2, ..., g_k >$), which are used to represent the entire trajectory. The segmentation points are chosen from the original extracted points appropriately,



Trajectories extracted from the video
Initialize the SRs and ERs
Initialize the velocity fields
Initialization
Revise the velocity fields using the K-means algorithm
Learn the probability graph
Learn the Poisson process of pedestrians entering SRs
Learn the distributions of starting positions and preferred speed of pedestrians
Constructing Model Components Based on Off-line Video

Figure 2: The model construction procedure.

so that the length of each segment is equal to or slightly larger than $\sqrt{2}\Delta$. In this way, each grid cell contains at most one segment point. This can facilitate the construction of velocity fields, as will be described later. Algorithm 2 illustrates the procedure to obtain the segmentation points.

Algorithm 2: TRAJECTORY SEGMENTATION PROCEDURE.

```
1  $k = 1$;
2  for $i = 1$ to $n$ do
3      if $i = 1$ or $i = n$ then
4          $g_k = e_i$;
5          $k = k + 1$;
6      else
7          $d = ||g_{k-1} - e_i||$;
8          if $d > \sqrt{2}\Delta$ then
9              $g_k = e_i$;
10             $k = k + 1$;
11 return $< g_1, g_2, ..., g_k >$
```

After that, the velocity field of each ER (i.e., $\mathbf{V}_i$) is then initialized by assigning each grid with a vector that directly points to the ER. That is , the value of $v_{i,j,k}$ is set as:

$$v_{i,j,k} = \begin{cases} \Gamma(\overrightarrow{AB}), \text{if grid cell } (j,k) \text{ contains no obstacle.} \\ (0,0), \text{if grid cell } (j,k) \text{ contains obstacles.} \end{cases} \tag{5}$$

where A is the centre point of grid cell (j,k), B is the center point of the rectangle that represents the i-th ER, and function $\Gamma(\overrightarrow{a})$ normalizes the input vector $\overrightarrow{a}$ so that the length of $\overrightarrow{a}$ is equal to $\sqrt{2}\Delta$.

3.2.2 *Construction of Model Components Based on Off-line Video*

In this step, the K-means clustering algorithm is firstly used to classify all trajectories into M groups, where M is the total number of ERs. The trajectories in the same group are expected to terminate at the same ER. To apply the K-means algorithm, the center of the i-th cluster is defined as

$$\mathbf{C}_i = \begin{bmatrix} c_{i,1,1} & c_{i,1,2} & ... & c_{i,1,W} \\ ... & ... & ... & ... \\ c_{i,H,1} & c_{i,H,2} & ... & c_{i,H,W} \end{bmatrix} \tag{6}$$

where $c_{i,j,k}$ is initially set equal to $v_{i,j,k}$. Accordingly, each trajectory is represented by:

$$\mathbf{F}_i = \begin{bmatrix} \xi_{i,1,1} & \xi_{,1,2} & ... & \xi_{i,1,W} \\ ... & ... & ... & ... \\ \xi_{i,H,1} & \xi_{i,H,2} & ... & \xi_{i,H,W} \end{bmatrix} \tag{7}$$

where the value of $\xi_{i,j,k}$ is set by:

$$\xi_{i,j,k} = \begin{cases} \Gamma(\overrightarrow{g_h g_{h+1}}), \text{if } \exists g_h \in \text{grid cell}(j,k) \\ (0,0), \text{ otherwise} \end{cases} \tag{8}$$

where g_h is the h-th segmentation point of the trajectory.

With the above representation, the distance between the i-th trajectory and the j-th cluster is calculated by:

$$\begin{aligned} D(i,j) =& ||\mathbf{F}_i \ominus \mathbf{C}_j|| \\ =& \sum_{u<H, v<W, \xi_{i,u,v} \neq (0,0)} (||\xi_{i,u,v} - c_{j,u,v}||) \end{aligned} \tag{9}$$

where $||\overrightarrow{a}||$ returns the length of vector $\overrightarrow{a}$.

Then in each iteration of the K-means clustering process, each trajectory is classified into the nearest cluster. At the end of each iteration, the center of each cluster is updated by:

$$\mathbf{C}_i = \mathbf{E} \oslash \mathbf{I} \tag{10}$$

$$\mathbf{E} = \mathbf{F}_{k_1} + \mathbf{F}_{k_2} +, ..., +\mathbf{F}_{k_m} + \mathbf{C}_i \tag{11}$$

$$\mathbf{I} = \aleph(\mathbf{F}_{k_1}) + \aleph(\mathbf{F}_{k_2}) +, ..., +\aleph(\mathbf{F}_{k_m}) + \aleph(\mathbf{C}_i) \tag{12}$$

where $k_1, k_2, ..., k_m$ are indices of trajectories classified into the i-th cluster; $\mathbf{E}$ records the sum of vectors in each grid cell; $\mathbf{I}$ records the number of non-zero vectors in each grid cell; function $\aleph(\mathbf{F}_i)$ converts $\mathbf{F}_i$ to a binary matrix:

$$\aleph(\mathbf{F}_i) = \begin{bmatrix} I_{1,1} & I_{1,2} & ... & I_{1,W} \\ ... & ... & ... & ... \\ I_{H,1} & I_{H,2} & ... & I_{H,W} \end{bmatrix} \tag{13}$$

where

$$I_{j,k} = \begin{cases} 1, \text{if } \xi_{i,j,k} \neq (0,0) \\ 0, \text{otherwise} \end{cases} \tag{14}$$

$\oslash$ is a special operation which returns a new matrix by:

$$\mathbf{E} \oslash \mathbf{I} = \begin{bmatrix} \frac{E_{1,1}}{I_{1,1}} & \frac{E_{1,2}}{I_{1,2}} & ... & \frac{E1,W}{I_{1,W}} \\ ... & ... & ... & ... \\ \frac{E_{H,1}}{I_{H,1}} & \frac{E_{H,2}}{I_{H,2}} & ... & \frac{E_{H,W}}{I_{H,W}} \end{bmatrix}. \tag{15}$$

The general procedure of the proposed training process is illustrated in Algorithm 3.

After classifying all trajectories, each $\mathbf{C}_i$ is then used as the temporary velocity field of the corresponding ER. In order to make the moving curves of pedestrians smoother, the temporary velocity field is further revised by considering the neighboring influences:

$$v_{i,j,k} = \Gamma(\sum_{u<H, v<W} exp(\frac{-d_{(j,k),(u,v)}}{\omega}) c_{i,u,v}) \tag{16}$$

Algorithm 3: K-MEANS TRAINING PROCEDURE.

```
1 initialize $\mathbf{C}_i$ and $\mathbf{F}_i$ by Eq.(6) to Eq.(8).
2 for $i = 1$ to $max\_iterations$ do
      /* $N$ is the total number of
         trajectories used for training.   */
3     for $j = 1$ to $N$ do
4         for $k = 1$ to $M$ do
5             Calculate $D(j,k)$ by Eq.(9)
6         $u = min_k\{D(j,k)|k = 1, ..., M\}$
7         Classify trajectory $j$ to cluster $u$
8     Update the centers of clusters by Eq.(10).
9 return $\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_M$
```

where ω (e.g., $\omega = 2$) is a scaling factor and $d_{(j,k),(u,v)}$ is the distance between grid cell (j,k) and grid cell (u,v), which can be calculated by:

$$d_{(j,k),(u,v)} = \sqrt{((j-u)*\Delta)^2 + ((k-v)*\Delta)^2} \qquad (17)$$

In this way, the vector that is closer to grid cell (j,k) will have a larger influence on $v_{i,j,k}$.

Once the velocity fields of all ERs are obtained, they can then be utilized to learn the state transition rates between SRs and ERs (i.e., the probability graph). To achieve this, the trajectories that start from the i-th SR and end at other regions (i.e., the last point of the trajectory is not in the i-th SR) are selected for analysis. We omit the other trajectories so as to avoid re-calculating multiple trajectories of the same pedestrian. The goal of each selected trajectory is set to be the ER whose velocity field has the smallest distance to the trajectory. Denote $S_{i,j}$ as the total number of trajectories that have the i-th SR as the origin and the j-th ER as the goal, then the transition rate from the i-th SR to the j-th ER is set to be:

$$p_{i,j} = \frac{S_{i,j}}{\sum_{k=1}^{M} S_{i,k}} \qquad (18)$$

The frequency of new pedestrians appearing in the i-th SR (i.e., k_i) is also learned by analyzing the above selected trajectories. The first time step of each selected trajectory that has the i-th SR as the origin indicates a new pedestrian appearing in the i-th SR, and the average number of pedestrians appearing in the i-th SR during a time period of λ is calculated as k_i. Moreover, the mean and the covariance of the first point of the selected trajectories are calculated. This information is used to set the starting positions of new pedestrians in the simulation. In addition, the mean and the standard deviation of speeds are also calculated to generate the preferred speeds of agents in the simulation.

3.3 Discussion of Refining Model Components Based on On-line Video

In some practical applications, new video data can be continuously obtained on line during the simulation. In these cases, it is desirable to make use of the new data to continuously revise the model, so that the model can automatically adapt to the new situations. The proposed framework has a good potentiality to meet this requirement. This subsection briefly discusses some techniques that can be used to deal with the on-line video data.

Suppose there is a new trajectory ($\mathbf{F}'$) extracted from on-line video. We can first classify it to the nearest cluster (denoted as the i-th cluster), and then update the center of the i-th cluster by:

$$E = E_0 + \mathbf{F}' \qquad (19)$$

$$I = I_0 + \aleph(\mathbf{F}') \qquad (20)$$

$$C = E \oslash I \qquad (21)$$

where E_0 and I_0 are calculated by Eq.(10) in the previous training process. Then the velocity field of the i-th ER can be updated using Eq. (16). In this way, the velocity fields can be incrementally revised without processing all extracted trajectories.

To update k_i, two statistical values sp and si should be recorded during the off-line learning process, where sp is the total number of pedestrians appearing in the i-th SR and si is the total number of time intervals (i.e., a time interval = λ). Then k_i can be updated by:

$$k_i = \frac{sp + \Delta_p}{si + \Delta_i} \qquad (22)$$

where Δ_p and Δ_i are the number of pedestrians and the number of time intervals calculated based on the new data respectively.

Similarly, the transition probability graph can be incrementally updated by:

$$p_{i,j} = \frac{sa_{i,j} + \Delta n_{i,j}}{\sum_{j=1}^{M}(sa_{i,j} + \Delta n_{i,j})} \qquad (23)$$

where $sa_{i,j}$ is the total number of old trajectories from the i-th SR to the j-th ER, $\Delta n_{i,j}$ is the total number of new trajectories from the i-th SR to the j-th ER. The mean and variance of speed can also be incrementally updated using the methods proposed in [26].

4. SIMULATION STUDIES

4.1 Simulation Settings

To test the effectiveness of the proposed data-driven modeling framework, this section applies the framework to simulate the crowd behaviors in New York Grand Central Terminal. A 33-minutes video is used for model training and validation[1]. The video has a resolution of 480 x 720 and the frame rate is 24fps. Fig. 3 shows a frame of the video. There are 40000 trajectories extracted from the video. In this paper, the first 20000 trajectories are used for training, and the next 20000 trajectories are used for validation.

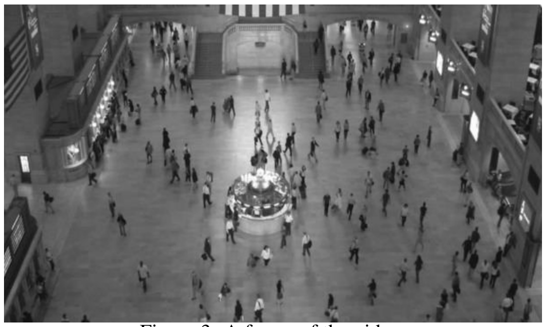


Figure 3: A frame of the video.

To generate more realistic crowd behaviors, the 40000 trajectories are transformed from "Pixel coordinate system" to "World coordinate system" in this paper. Fig. 4 plots the transformed trajectories used in the simulation studies[2]. Before constructing

[1]The video and extracted trajectories are downloaded from `http://www.ee.cuhk.edu.hk/~xgwang/grandcentral.html`

[2]The transformed trajectories, the simulation videos and the related source code of the paper can be downloaded from: `http://crowds.sce.ntu.edu.sg/index.php/research/ddabcm`

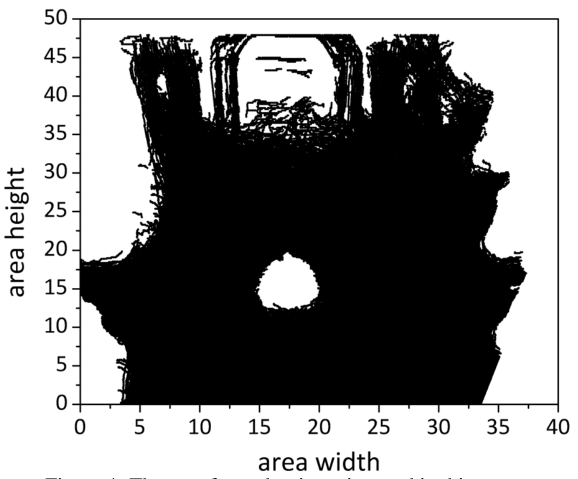


Figure 4: The transformed trajectories used in this paper.

the model components, eight SRs adn eight ERs are marked using the following eight rectangles : SR1 = ER1 ={0, 32, 18, 47}, SR2 = ER2 ={18, 32, 37, 47}, SR3 = ER3 ={34, 21, 37, 30}, SR4 = ER4 = {34, 8, 37, 19}, SR5 = ER5 = {34, 0, 37, 8}, SR6 = ER6 = {18, 0, 34, 1}, SR7 = ER7 = {0, 0, 18, 1}, and SR8 = ER8 = {0, 8, 4, 19}, where the position of each rectangle is denoted in the form of {left, bottom, right, top}.

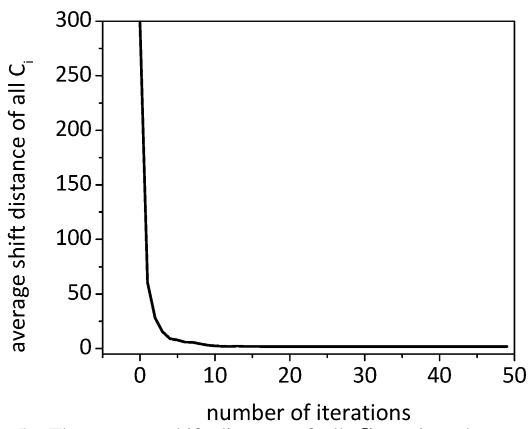


Figure 5: The average shift distance of all C_i against the number of iterations.

4.2 Model Training Results

In the model training process, the maximum iterations of the K-means clustering algorithm is set to be 50. Fig. 5 shows the average shift distance of all $\mathbf{C}_i$ at each iteration. It can be observed that the average shift distance drops quickly during the first 10 iterations, which indicates that the classification result quickly converges within a few number of iterations. Fig. 6 shows an example of the final learned velocity fields. It can be observed that the moving curves look realistic and can help agents to avoid obstacles. Table 1 lists the transition probabilities between SRs and ERs, where the first column describes the SRs and the first row describes the ERs. The last row describes the sum of the probability values in each column. It can be observed that ER4 is more likely to be chosen as the goal, which is consistent with what was observed from the video. The mean and covariance of moving speeds is 1.09m/s and 0.282 respectively. The average number of new pedestrians appearing in the 8 SRs during 5 seconds are 1.105, 1.993, 1.140, 0.140, 1.049, 12.189, 6.965 and 1 respectively.

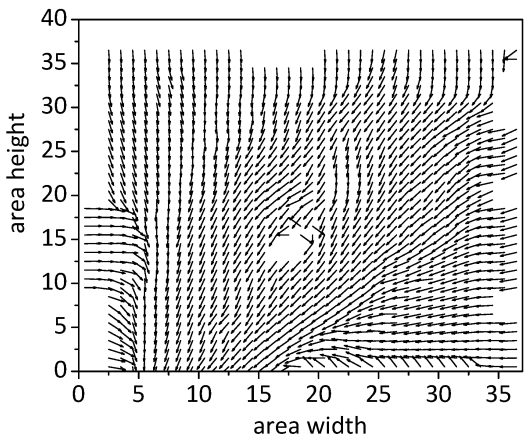


Figure 6: An example of the trained velocity fields.

4.3 Model Validation

Firstly, we validate the constructed model by comparing the densities distributions of the crowds in the simulation and the video data. We omit the first 20000 trajectories which are used for training purpose, and use the next 20000 trajectories to calculate the densities of the crowds in video data. The simulations are run using the same number of time steps as in the video data. The entire region is evenly divided into 37×48 grids, with each grid having a size of $1m \times 1m$. The center points of all grids are used as representative points to estimate the density distribution of the crowd. As suggested in Helbing's work [6] , the local density of the representative point (i, j) is computed by:

$$\rho_{i,j} = \frac{1}{2\pi R^2} \sum_{k=1}^{n} exp(-\frac{d(i,j,k)^2}{R^2}) \tag{24}$$

where n is the number of points in all trajectories, $d(i, j, k)$ is the distance from the k-th point to point (i, j), and R is a scaling parameter. In this way, a point in a trajectory that is closer to the representative point (i, j) will have a larger contribution to its local density value.

We compare the proposed model (labelled as Dual-layer Agent-Based Crowd model, D-ABC) with four other models. The first model (labelled as D-ABC/V) is the same as D-ABC model except that it uses the initial velocity fields rather than the trained velocity fields for navigation. The second version (labelled as D-ABC/T) is the same as D-ABC except that agents in D-ABC/T randomly choose ERs as their goals rather than using the transition probabilities listed in Table 1. The third model (labelled as D-ABC/VT) is the same as D-ABC/V except that agents in D-ABC/VT randomly choose ERs as their goals. The fourth model (labelled as RVO2-random) is a simple RVO2 model without using velocity fields and the transition probabilities. In the RVO2-random model, the number of new pedestrians entering each SR during a time period of λ is fixed as $\sum_{i=1}^{8}(k_i)/8$.

Fig. 7 shows the density distributions of the crowds of the video and those generated by the five models. Compared with the results of the other four models, the result of D-ABC looks more similar to that of the video data. There are two significant features in the density distribution of the video data. The first feature is that the density values in the middle circle are very small, because there is a counter located in that area. The second feature is that the density values of a narrow area from (20, 0) to (37, 15) are the highest among all regions. Without using the probability graph, the result of D-ABC/T fails to contain the second feature. Meanwhile,

Table 1: The transition probabilities between SRs and ERs.

	ER1	ER2	ER3	ER4	ER5	ER6	ER7	ER8
SR1	0.0000	0.0514	0.0882	0.1102	0.2720	0.1470	0.1985	0.1327
SR2	0.0825	0.0000	0.0596	0.1284	0.1743	0.1651	0.2155	0.1746
SR3	0.1081	0.0608	0.0000	0.1013	0.0540	0.2635	0.2635	0.1488
SR4	0.0000	0.1666	0.7500	0.0000	0.0000	0.0833	0.0000	0.0001
SR5	0.1890	0.0731	0.0365	0.6402	0.0000	0.0304	0.0121	0.0187
SR6	0.1471	0.1334	0.1363	0.4198	0.0559	0.0000	0.0214	0.0861
SR7	0.2555	0.2161	0.1544	0.1927	0.0202	0.0149	0.0000	0.1462
SR8	0.1363	0.1363	0.0757	0.0909	0.1212	0.0303	0.4090	0.0003
sum	0.9185	0.8377	1.3007	**1.6835**	0.6976	0.7345	1.12	0.7075

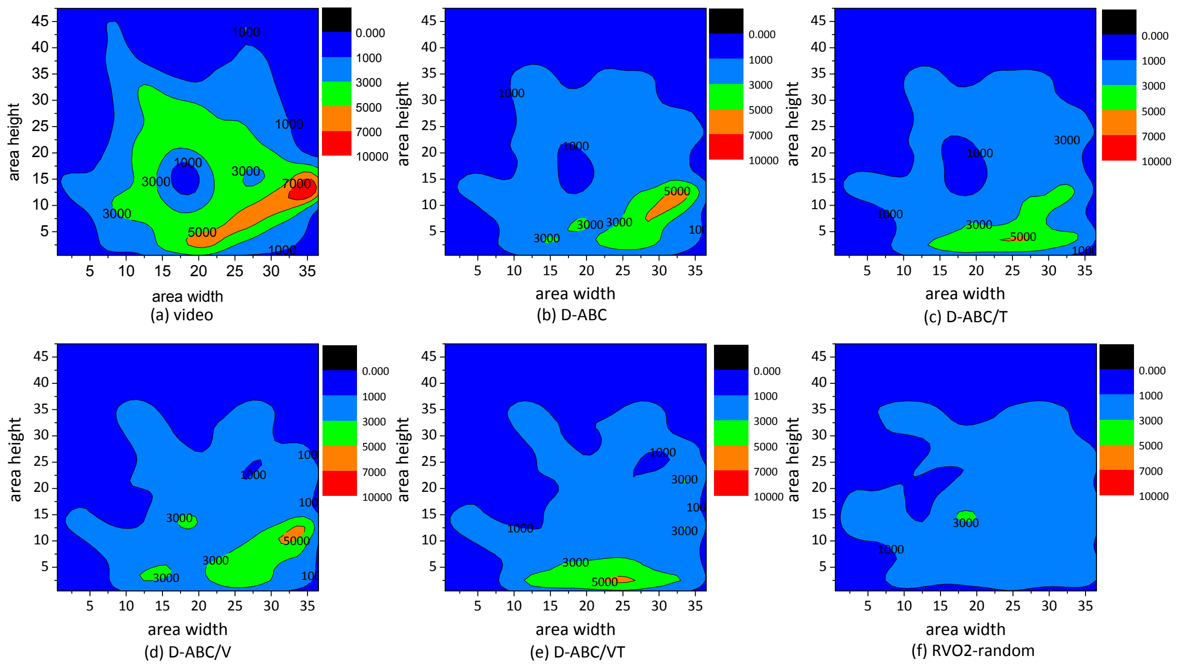


Figure 7: The crowd density distributions of the video and the five models.

Table 2: The $error$ of the five density distributions.

Model	D-ABC	D-ABC/T	D-ABC/V	D-ABC/VT	RVO2-random
$error$	1113.35	1312.23	1215.66	1380.24	1590.75

the result of D-ABC/V fails to contain the first feature, due to the lack of good velocity fields. The results of D-ABC/VT and RVO2-random fail to contain both features. This is because that the agents in these two models choose goals randomly and can not intelligently avoid obstacles in the middle region. By using the trained velocity fields and the probability graph, the result of D-ABC contains both features.

To quantitatively measure the similarity between the simulated behaviors and those observed in the video data, we calculate the distance between the density distributions of the simulated crowd and those of the video data:

$$error = \sum_{t=1}^{T} \sqrt{\frac{\sum_{i=1}^{H}\sum_{j=1}^{W}(\rho_{i,j}(t) - \rho_{i,j}^{0}(t))^2}{H * W}} \quad (25)$$

where $\rho_{i,j}(t)$ is the simulated density value of point (i,j) at t time step, and $\rho_{i,j}^{0}(t)$ is that of the video data. Table 2 lists the $error$ values of the five models. It is clear that D-ABC obtains the smallest $error$ value. The above results demonstrate that the proposed training method is effective to improve the performance of the model in generating desired emergent crowd dynamics.

Secondly, we investigate the prediction performance of the

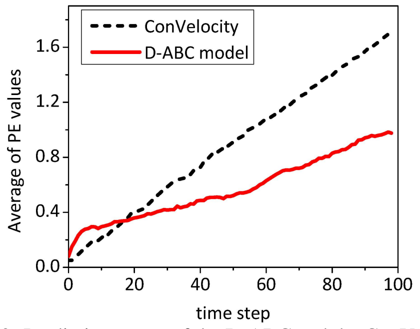


Figure 9: Prediction errors of the D-ABC and the ConVelocity.

proposed D-ABC model. We randomly choose 20 relatively long trajectories for prediction analysis. For each trajectory, its first 1/3 points are assumed to be known in advance. Based on the 1/3 points, the D-ABC model firstly estimates the goal of each trajectory by choosing an ER whose velocity field is the closest to the trajectory. Then the D-ABC model uses the corresponding velocity field to determine the future moving directions. During the simulation, each trajectory is associated to one agent. The trajectories of other background pedestrians are set the same as in the video data, while that of the chosen one is determined by the D-ABC model. The first 1/3 points of the chosen agent

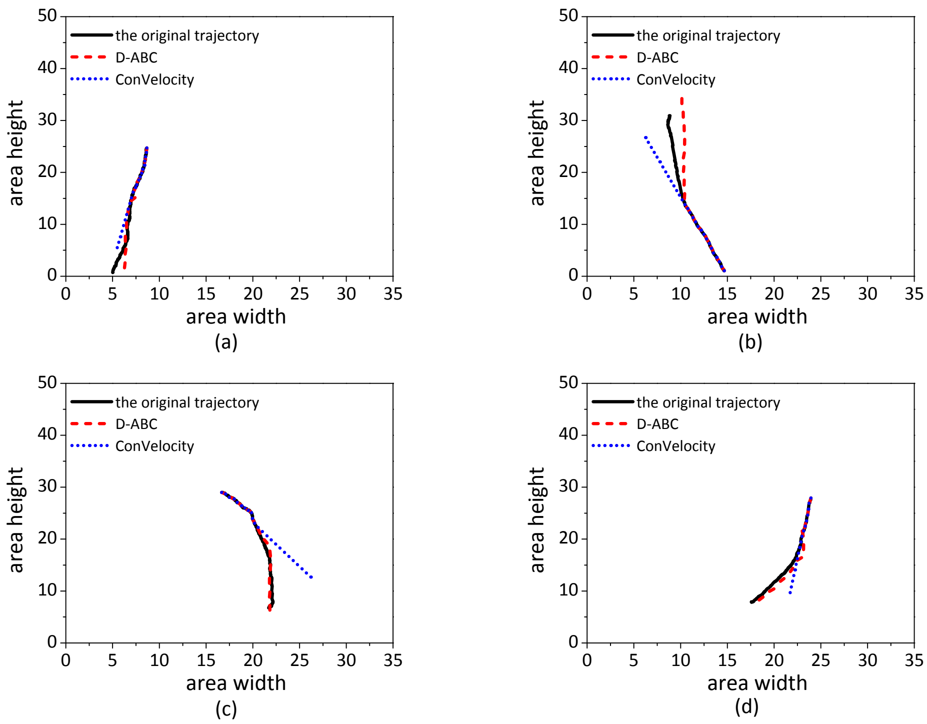


Figure 8: Examples of the predicted trajectories found by the D-ABC and the ConVelocity.

Table 3: Average of the final prediction errors.

Method	ConVelocity	D-ABC
Average of PE	2.372	1.66
p-value	0.01647	N/A

is also set as the same as in the video data. We compare the performance of the D-ABC model with a constant velocity method (referred as ConVelocity) which uses the average velocity of the past observation to predict future positions. Fig. 8 shows four examples of the prediction results. It can be observed that the predicted trajectories offered by the D-ABC model can better match with the true trajectories.

To quantify the difference of the prediction performance, the mean prediction error is used for comparison. For a specific trajectory, the prediction error is calculated by:

$$PE = \sum_{k=a}^{b}(||r_k - r'_k||) \tag{26}$$

where a and b are the starting time step and the end time step of the prediction period, r_k and r'_k are the positions of the k-th point in video data and in the simulation respectively. Fig. 9 shows the average predicted errors of the 20 trajectories going with the time steps. Clearly, as the number of time steps increases, the predicted error of the D-ABC model becomes much better than that of the ConVelocity.

Table 3 lists the average of the final prediction errors of the D-ABC and the ConVelocity. It can be observed that the average prediction error of the D-ABC is smaller than that of the Con-Velocity. In addition, we perform a one tail t-test to check the significant difference of the 20 prediction results. The p-value (PE of ConVelocity is smaller than or equal to that of the D-ABC) is 0.01647 which indicates that the performance of the proposed D-ABC is significantly better than that of the ConVelocity at a 0.05 confidence level.

5. CONCLUSION

This paper has proposed a generic data-driven crowd modeling framework to generate realistic crowd behaviours that can match the video data. Our method models crowd behaviours in a hierarchical manner, by considering both the microscopic collision avoidance behaviours and multiple macroscopic behaviours. An automatic method is proposed to construct the model components from the video data. The proposed framework was used to simulate the crowd behaviours in New York Grand Central Terminal. The simulation results have shown that our proposed framework is effective to generate the desired crowd dynamics and can provide promising prediction performance.

The proposed methodology can construct components of crowd model by learning behavior patterns from video. Once a well-fit crowd model is constructed, it then can be used to predict the crowd behaviors under different initial conditions. As the framework considers several components for specifying agent behaviours, it would be flexible. For instance, the algorithm for collision avoidance could be changed, or other components for considering other issues could be integrated. In the current work, the knowledge learned from the video is only suitable for the specific scenario. Therefore, using machine learning techniques such as genetic programming to distill generic behavior rules that are applicable to different scenarios is a promising future research work. In addition, applying the proposed method to practical applications such as crowd tracking is another interesting research direction.

Acknowledgments

The research reported in this paper is financially supported by the Tier 1 Academic Research Fund (AcRF) under project Number RG23/14. Linbo Luo is partially supported by “Fundamental Research Funds for the Central Universities" (Project XJS14013).

REFERENCES

[1] S. Ali, K. Nishino, D. Manocha, and M. Shah. *Modeling, Simulation and Visual Analysis of Crowds*. Springer New York, 2013.

[2] S. Ali and M. Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *CVPR'07*, pages 1–6. IEEE, 2007.

[3] H. Finnsson and Y. Björnsson. Simulation-based approach to general game playing. In *AAAI*, volume 8, pages 259–264, 2008.

[4] W. Ge, R. Collins, and R. Ruback. Vision-based analysis of small groups in pedestrian crowds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):1003–1016, May 2012.

[5] S. J. Guy, M. C. Lin, and D. Manocha. Modeling collision avoidance behavior for virtual humans. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 575–582, 2010.

[6] D. Helbing, A. Johansson, and H. Z. Al-Abideen. Dynamics of crowd disasters: An empirical study. *Physical review E*, 75(4):046109, 2007.

[7] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[8] A. Johansson, D. Helbing, and P. K. Shukla. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in complex systems*, 10(supp02):271–288, 2007.

[9] E. Ju, M. G. Choi, M. Park, J. Lee, K. H. Lee, and S. Takahashi. Morphable crowds. In *ACM Transactions on Graphics*, volume 29, page 140. ACM, 2010.

[10] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–118. Eurographics Association, 2007.

[11] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer Graphics Forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[12] A. Lerner, E. Fitusi, Y. Chrysanthou, and D. Cohen-Or. Fitting behaviors to pedestrian simulations. In *SIGGRAPH'09*, pages 199–208. ACM, 2009.

[13] D. Lin, E. Grimson, and J. Fisher. Learning visual flows: A lie algebraic approach. In *CVPR'09*, pages 747–754. IEEE, 2009.

[14] D. Lin, E. Grimson, and J. Fisher. Modeling and estimating persistent motion with geometric flows. In *CVPR'10*, pages 1–8. IEEE, 2010.

[15] L. Luo, S. Zhou, W. Cai, M. Y. H. Low, F. Tian, Y. Wang, X. Xiao, and D. Chen. Agent-based human behavior modeling for crowd simulation. *Computer Animation and Virtual Worlds*, 19(3-4):271–281, 2008.

[16] R. Mehran, B. E. Moore, and M. Shah. A streakline representation of flow in crowded scenes. In *ECCV'10*, pages 439–452. Springer, 2010.

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

[18] S. R. Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):152–164, 2001.

[19] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV'09*, pages 261–268. IEEE, 2009.

[20] P. Scovanner and M. F. Tappen. Learning pedestrian dynamics from the real world. In *ICCV'09*, volume 9, pages 381–388, 2009.

[21] W. Shao and D. Terzopoulos. Autonomous pedestrians. In *SIGGRAPH'05*, pages 19–28, 2005.

[22] D. Thalmann. *Crowd simulation*. Wiley Online Library, 2007.

[23] C. Tomasi and T. Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[24] J. Tsai, N. Fridman, E. Bowring, M. Brown, S. Epstein, G. Kaminka, S. Marsella, A. Ogden, I. Rika, A. Sheel, et al. Escapes: evacuation simulation with children, authorities, parents, emotions, and social comparison. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 457–464, 2011.

[25] J. Van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA'08*, pages 1928–1935. IEEE, 2008.

[26] B. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[27] D. Wolinski, S J Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré. Parameter estimation and comparative evaluation of crowd simulations. In *Computer Graphics Forum*, volume 33, pages 303–312. Wiley Online Library, 2014.

[28] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *CVPR'11*, pages 1345–1352. IEEE, 2011.

[29] M. Zhao, S. J. Turner, and W. Cai. A data-driven crowd simulation model based on clustering and classification. In *2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 125–134. IEEE, 2013.

[30] J. Zhong, W. Cai, L. Luo, and M. Lees. Ea-based evacuation planning using agent-based crowd simulation. In *Proceedings of the 2014 Winter Simulation Conference*, pages 395–406. IEEE Press, 2014.

[31] J. Zhong, N. Hu, W. Cai, M. Lees, and L. Luo. Density-based evolutionary framework for crowd model calibration. *Journal of Computational Science*, 6:11–22, 2015.

[32] J. Zhong, L. Luo, W. Cai, and M. Lees. Automatic rule identification for agent-based crowd models through gene expression programming. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1125–1132. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[33] B. Zhou, X. Wang, and X. Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *CVPR'12*, pages 2871–2878. IEEE, 2012.

[34] S. Zhou, D. Chen, W. Cai, L. Luo, M. Y. H. Low, F. Tian, V. S.-H. Tay, D. W. S. Ong, and B. D. Hamilton. Crowd modeling and simulation technologies. *ACM Trans. Model. Comput. Simul*, 20(4):20, 2010.