CLUST : simulating realistic crowd behaviour by mining pattern from crowd videos

Zhao, Mingbi; Cai, Wentong; Turner, S. J.

2017

Zhao, M., Cai, W., & Turner, S. J. (2018). CLUST : simulating realistic crowd behaviour by mining pattern from crowd videos. *Computer Graphics Forum*, 37(1), 184-201.
doi:10.1111/cgf.13259

<https://hdl.handle.net/10356/83271>

<https://doi.org/10.1111/cgf.13259>

This is the peer reviewed version of the following article: Zhao, M., Cai, W., & Turner, S. J. (2018). CLUST : simulating realistic crowd behaviour by mining pattern from crowd videos. *Computer Graphics Forum*, 37(1), 184-201, which has been published in final form at <http://dx.doi.org/10.1111/cgf.13259>. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions.

Downloaded on 26 Apr 2023 21:29:53 SGT

CLUST: Simulating Realistic Crowd Behavior by Mining Pattern from Crowd Videos

M. Zhao¹ W. Cai¹ and S. J. Turner²

¹Nanyang Technological University, Singapore

²King Mongkut's University of Technology Thonburi, Thailand

Abstract

In this paper, we present a data-driven approach to simulate realistic locomotion of virtual pedestrians. We focus on simulating low level pedestrians' motion, where a pedestrian's motion is mainly affected by other pedestrians and static obstacles nearby, and the preferred velocities of agents (direction and speed) are obtained from higher level path planning models. Before the simulation, collision avoidance processes (i.e., examples) are extracted from videos to describe how pedestrians avoid collisions, which are then clustered using hierarchical clustering algorithm with a novel distance function to find similar patterns of pedestrians' collision avoidance behaviors. During the simulation, at each time step, the perceived state of each agent is classified into one cluster using a neural network trained before the simulation. A sequence of velocity vectors, representing the agent's future motion, is selected among the examples corresponding to the chosen cluster. The proposed CLUST model is trained and applied to different real world datasets to evaluate its generality and effectiveness both qualitatively and quantitatively. The simulation results demonstrate that the proposed model can generate realistic crowd behaviors with comparable computational cost.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multiagent systems

1. Introduction

Crowd simulation is an active research field that has increasing number of applications in various research areas such as virtual environment [PSAB08], object tracking [PESVG09, LSTA10], computer animation [ST05, SGV*12] and civil planning [KSMK05, PA*11]. A core issue in crowd simulation is to simulate the motions of pedestrians in virtual worlds in a human-like manner so that the simulation looks realistic. Various mathematical models have been proposed over the last few years to simulate realistic crowd motions based on the current knowledge, the hypothesis and assumptions observed from pedestrians' motion, e.g., least biomechanical energy spent [GCC*10], collision-free velocities which are closest to the preferred velocities [VDBGLM11] and a set of rules that employed by all agents in crowd [Rey99]. In order to reproduce complex crowd behaviors, some researchers include personal attributes such as personality, emotion, group interactions and cultural difference [MT97, ALL*11, SJCO06, LZC*08] in their models, or define more finely tuned situation-specific rules and strategies [ST05, LMM03, SGC04, HLZ13]. However, identifying the correct rules that influence pedestrian's motions requires a great amount of domain knowledge and besides, the wide variety of steering behaviors are difficult to generate by considering only a limited number of factors.

More recently, human crowd motion data has been introduced in predicting pedestrians' movements. Some researchers calibrate the parameters in their models using human motion data to improve the accuracy of motion prediction [POO*09, ABW06, PPD07, ST09, MHG*09, JHS07, PESVG09, YBOB11]. However, a pre-specified model is still needed before the calibration process.

In another branch of the data-driven approaches [LCHL07, LCL07], examples are extracted from crowd videos before the simulation as a description of the motion behaviors of pedestrians under given surroundings. An example contains two parts: state which refers to the surrounding environment of a specific pedestrian such as other pedestrians and static obstacles; action that stores the corresponding movement of the pedestrian in the given state, which can be either a two dimensional velocity vector indicating the position of the pedestrian at next time step [LCHL07] or a sequence of velocity vectors over a short period of time [LCL07]. During the simulation, for each agent a similar situation (i.e., state) is found in the examples and the corresponding action is copied.

Compared to other models, example-based models may produce realistic crowd motion, because examples contain a variety of different motion behaviors appeared in the real world. However, the performance depends on two aspects: how example database is constructed from crowd videos before the simulation and how a similar

situation is found in the examples accurately and efficiently during the simulation.

Most of the existing example-based models only record the current positions of nearby pedestrians to limit the dimensions of the state vector and thus reduce the computation time in searching for a similar situation from examples during the simulation. However, pedestrians usually make decisions based on a short period of time [CA07]. Thus using only the current situation may not properly describe the whole situation a pedestrian is facing. Second, action in the existing example-based models is simply the instantaneous speed and moving direction of a pedestrian or a trajectory segment with fixed length. However, pedestrians make their decisions deliberately as a response to their current specific situations (i.e., to avoid collisions), which may be an instant change of velocity at the next time step or a series of velocities to gradually avoid collisions. Therefore, actions may have different durations, which need to be deliberately determined to properly describe the reactions of pedestrians in different perceived surroundings. Finally, there are usually tens of thousands of examples extracted from videos and the number of examples increases as there are more agents in the video. Therefore searching examples in a brute force manner during the simulation is impractical especially in simulating crowded scenarios. Therefore in the existing example-based models, the dimension of the state space is usually reduced by applying Principle Component Analysis (PCA). Due to the possible loss of accuracy in dimension reduction process, unnatural motions may appear during the simulation.

We propose a clustering-based data driven model to focus on simulating the collision avoidance behavior of pedestrians in low density. Compared to other data-driven models, we make improvement from both quality and performance aspects. First, we include the previous motion conditions to construct states of pedestrians. Second, we identify examples of pedestrians according to their surroundings. In each example, the subject pedestrian executes the action of the example to avoid a pedestrian on his/her collision course (referred to as collidee), based on the observed surrounding situations (i.e., the state of the example). Each example is transformed to a collision avoidance process (CAP) to facilitate clustering process. Third, we group CAPs into clusters using hierarchical clustering approach, and the corresponding examples are also grouped accordingly. A neural network classifier is trained thereafter based on the clustering result to learn the mapping between examples and their cluster memberships. More detailed features are defined to distinguish between actions belonging to the same cluster, which are organized into a k -d tree. During the simulation, an agent first selects one cluster using the trained neural network classifier and then determines an action by searching the k -d tree of that cluster. By clustering examples before the simulation, we reduce the search space of agents during the simulation from all examples to a cluster of examples.

This idea of mining clusters from pedestrians' collision avoidance behaviors (i.e., CAPs) in the videos are previously introduced in [ZTC13]. Several improvements have been made in this paper. First, the definition of example is updated, in order to represent the state and action more accurately. Several metrics are defined in this paper to measure the importance of a pedestrian's neigh-

bors. The length of each pedestrian's action is determined by the surrounding condition that is influencing the pedestrian, instead of the instantaneous velocity of the pedestrian used in [ZTC13]. Second, a new linkage criterion, the modified Hausdorff distance, is proposed in this paper to calculate the distance between two clusters, because the commonly used linkage criteria (e.g., complete linkage used in [ZTC13]) tend to be sensitive to the extreme entities in two clusters (e.g., smallest distance and largest distance), especially in the case of human movement data which involve motion uncertainty and tracking noise. Additionally, in [ZTC13], after obtaining a set of actions from one cluster after classification, the first action that does not cause collisions is selected in a brute force manner. In this paper, we select multiple features to model the pedestrian's action and store actions of one cluster in a k -d tree. Therefore the best action can be found more quickly and accurately. Finally, in [ZTC13], the proposed model is evaluated by replicating the RVO2 model [VDBGML11] behavior, whereas in this paper, real-world motion data is applied.

In [ZCT15], we compared CLUST model with other crowd modeling approaches such as RVO2 [VDBGML11], social force [HFV00], approximate nearest neighbor search [LCHL07], and perception-action graph [CC14] against the real world motion data quantitatively on various metrics. However, the CLUST model is only briefly described. In this paper, more technical details of the CLUST model, such as example extraction, clustering and classification, are provided. Besides, the clustering results of different input datasets are also illustrated in this paper to demonstrate the effectiveness of the clustering approach. Additionally, cross validation is applied instead of hold-out method in the model evaluation. In addition to quantitative measurements, qualitative evaluation is also included. Finally, the generality of the proposed model is evaluated by applying the CLUST model trained in one dataset to another dataset.

The rest of this paper is organized as follows: We first report related work in Section 2 together with an overview of the proposed approach in Section 3. Section 4 describes how an example database is constructed, including extracting examples from video data and clustering them into clusters. This is followed by training a neural network classifier and constructing k -d trees in Section 5. Section 6 describes the velocity prediction procedure for simulating agents during the simulation. Section 7 presents the experimental results and finally Section 8 concludes the paper.

2. Related Work

Several crowd simulation models have been proposed over the past few years. One of the early works, rule-based models proposed by Reynolds [Rey87, Rey99], define a set of steering rules that agents will follow based on the observation of the surrounding environment. Many works followed thereafter, focusing on introducing more rules in the aspect of group interaction [MT97], higher level navigational behaviors [ST05] and real-time simulation in complex and structured environment [LD04].

Cellular Automaton (CA) models partition the space into squares, triangle, hexagon and so on. Some models define various rules to determine which neighboring cell to move to at next time

step. Other models update the transition probability of the neighboring cells and move agents accordingly [BA98, BA01, BKSZ01, KS02].

In social force models [HM95, HFV00, MPG*10, PB06], the social forces that influence the motion of an agent is modeled by the combination of the attraction of desired destination, the repulsive forces of other agents, and the repulsive forces of obstacles. The instantaneous velocity of agent is then defined according to Newton's Law of Motion.

In recent works, Van den Berg et al. [VdBLM08, vdBPS*08, VDBGLM11, GCK*09, GLM10, SvdBGM11, GvdBLM10] proposed reciprocal velocity obstacles model which is formulated based on the concept of velocity obstacles introduced in [FS98]. The velocity obstacles of other agents imposed on agent A is the set consisting of all those velocities $\mathbf{v}_A$ of A that will cause a collision with other agents within next τ time. The predicted velocity of A is the average of the current velocity and a velocity that lies outside the velocity obstacles, assuming that other agents perform similar decision logic. In this model, τ is the only parameter to be set before simulation. Increasing the value of τ , agents will be more sensitive to other agents, but there will be less choices of velocities for each agents.

With the development of human motion capture and video analysis technologies, more and more real world pedestrians motion data are available for crowd analysis and simulation. These real motion data can be used to calibrate the numerical parameters of crowd models as in [ABW06, JHS07, MHG*09, PPD07, POO*09, ST09]. Other researchers focus on automatic detecting human activities [PW12], motion patterns [WO16], and stationary crowd group [?] from the human motion data.

Recently, example-based approaches [LCHL07, LCL07, JCP*10, CC14] directly extract examples from crowd videos to form an example database. In simulation, an agent searches the database to find an example, whose state is the most similar to their observed state, and copies the action accordingly. The approximate nearest neighbor search model (ANN) is such an approach proposed by Lee et al. [LCHL07] in which the observed states of agents are matched to states extracted from crowd videos directly. A database consisting of state-action samples extracted from crowd videos are constructed before the simulation. Here, the state $\mathbf{s}_i^t \in \mathbb{R}^N$ of agent i at time t is a vector consisting of motion features of itself and other nearby agents and the action $\mathbf{a}_i^t \in \mathbb{R}^2$ of agent i at time t is simply the instantaneous velocity of i . To improve the searching efficiency in simulation, the Principle Component Analysis (PCA) is applied to reduce the dimensionality of the state vectors. Given a new state perceived in simulation, an action is predicted by selecting K nearby samples and combine them to produce an output.

The perception-action graph model (PAG) [CC14] is also a data-driven approach inspired by the previous work on motion graphs [KGP02, PP10]. Compared to the ANN model, besides storing state-action samples in a k -d tree, a perception-action graph (PAG) is also constructed where possible transitions between similar temporal perception patterns (TPPs), which are similar to the states in the ANN model, are connected. The edges of the PAG graph store trajectory segments (actions) that cause transitions of TPPs. During the simulation, agents first search the k -d tree to select a starting

node on the PAG graph and then traverse on the graph and apply actions stored on the currently visiting edges. The PAG model avoids continuous database searches so that computation time is reduced. As agents traverse on the graph, it is possible that the observed TPP and the one on the node the agent is currently visiting are not similar, especially when the PAG graph is not well connected. To handle this error, a jump policy is proposed so that if the difference between the observed TPP and the TPP on the current node is large, agents can jump to another better matching node by performing a k -d tree search. This error threshold is important to the PAG model's performance. A smaller error threshold will cause more frequent k -d tree searches and therefore will result in a better quality crowd but cost more computation time. If this jump policy is used at every time step for every agent, the PAG model is basically equal to the ANN model.

Qualitative evaluation methods are commonly applied in the evaluation of a large number of crowd models. In their evaluation process, the models are applied to different test scenarios and the quality of the simulation is judged by model developers through look-and-feel approaches [DH03a, DH03b, HD05, HDB03]. Qualitative evaluation is simple and straight-forward. This subjective approach can give a direct impression of the believability of the simulated trajectories. However, it may not be sufficient to provide comprehensive and systematic assessment of the model. Recently, quantitative evaluation metrics are proposed by researchers to statistically compare the quality of crowd models. In [WOO16], latent path patterns are discovered from real world videos and/or simulations using a new Stochastic Variational Dual Hierarchical Dirichlet Process (SV-DHDP) model. A quantitative metric is proposed thereafter to compute the similarity between both real and simulated datasets. Charalambous et.al [CKGC14] utilized outliers detection algorithms to quickly identify errors in simulations when compared to real-world data in terms of user-defined metrics. In [GVDBL*12], a crowd simulator's ability to reproduce aggregate crowd motions of real pedestrians are measured using an information-theoretic method.

3. Overview

The two stage crowd simulation framework is illustrated in Figure 1 and is described briefly in this section.

Each pedestrian can sense the surrounding situations within an vision area. It is assumed that Pedestrians and obstacles which are not in this vision area has no influence on the subject pedestrian's motion. Pedestrians located in the subject pedestrian's vision area are referred to as the neighbors of the subject pedestrian.

Before the simulation, for each pedestrian at every time step in the video, the neighbors of the subject pedestrian are checked for target collidEE. Once a target collidEE is found, a collision is detected. The relative positions of the target collidEE during a pre-defined time steps prior to the current time step (denoted as N_p) are recorded as the state of an example. The subject pedestrian then takes an action during the following time steps to avoid the collision with the target collidEE until the current collision is resolved or a new collision is detected. The global positions of the subject pedestrian during the time steps that the target collidEE of the subject pedestrian remains the same are recorded as an action taken

Figure 1: Overview of CLUST: (a) Procedures to be performed before the simulation; (b) Procedures to be performed during the simulation.

by the subject pedestrian to avoid the target collidee. The state and action form an example of how the subject pedestrian avoids the potential collision under the specific situation. A collision avoidance process (denoted as CAP) is generated from an example by changing the action to the relative positions of the target collidee in the subject pedestrian's local coordinate system, in order to facilitate the clustering process.

The CAPs are clustered using the agglomerative hierarchical clustering approach and then the corresponding examples are grouped accordingly. A neural network classifier is trained to map between the state of an example and its group membership. In order to find the best action from actions belonging to the same cluster during the simulation, we define multiple features to represent each action and store the actions of the same cluster in a k -d tree using these features as index. Therefore, after preprocessing, CLUST model contains two components: a neural network classifier and one k -d tree for each cluster.

During the simulation, at every time step, each agent observes the surrounding situations (i.e., other pedestrians and static obstacles) to detect target collidee. Once a target collidee is detected, a state is formed using the relative positions of the collidee and then classified into one cluster using the trained neural network to obtain a set of possible actions, from which one best action is selected by searching the k -d tree of the selected cluster using features.

In summary, there are four steps that need to be done before the simulation (see Figure 1a):

1. For each pedestrian in the video, examples containing the states of current situations and the actions taken by the subject pedestrians to avoid collisions are detected and extracted from real world video data. Examples are then transformed to CAPs.
2. CAPs are clustered into clusters using hierarchical clustering algorithm. After clustering, every example is labeled with its corresponding CAP's cluster membership.
3. A neural network classifier is trained using states as input and

their cluster memberships as output. The trained neural network is used to classify the perceived state of an agent into one cluster during the simulation.

4. All actions in one cluster are organized in a k -d tree using features in order to find the best action during the simulation.

Likewise, three steps are performed during the simulation in order to steer an agent in the scenarios (see Figure 1b):

1. The simulated agent senses the surrounding environment and form the query state.
2. The query state is classified into one cluster using the neural network trained before the simulation.
3. The best action is selected by querying the k -d tree of the classified cluster using more specified features.

4. Constructing Example Database

In this section, we discuss how to construct an example database from crowd motion videos. Table 1 lists some of the basic notations used in this paper. The trajectories of pedestrians in videos are tracked and recorded as a sequence of position vectors. We aim at constructing an example database based on these extracted trajectories. An example $E = (\mathbf{s}, \mathbf{a})$ contains two parts: state $\mathbf{s}$ and action $\mathbf{a}$.

4.1. Extracting Examples and Collision Avoidance Processes

For each pedestrian in the video data, it contains three steps to generate examples. First, potential collidees are checked at every time step. If a target collidee is identified, an example is generated and a CAP is extracted from the generated example.

Each pedestrian is able to sense and respond to the surroundings within a vision area, which is a circle sector defined by the radius R and the central angle θ . The sensing radius R varies based on the density of the crowd and θ is usually set to 270° [Kah73].

Table 1: The basic notations.

Name	Summary
E	an example
s	the state of an example.
a	the action of an example.
CAP	a collision avoidance process.
$Neighbor(i, t)$	a set of pedestrians located in pedestrian i 's vision area at current time step t .
$MMD(i, j)$	the Minimum Mutual Distance between a neighbor j and the subject pedestrian i .
$t_{MMD}(i, j)$	the time step when the subject pedestrian i and neighbor j reaches their smallest mutual distance.
$collidee(i, t)$	the collidee of the subject pedestrian i at time step t .
$(x'_{j,i}, y'_{j,i})$	the relative position of collidee j in the local coordinate system of subject pedestrian i at time step t .
$DTW(CAP_i, CAP_j)$	the optimal Euclidean distance between CAP_i and CAP_j using Dynamic Time Warping.
$d_{CAP}(CAP_i, CAP_j)$	the distance between CAP_i and CAP_j .
$d_C(C_i, C_j)$	the distance between two clusters C_i and C_j .

This definition of vision area is based on the observation of Goffman [Gof09]. He stated that pedestrians scan the surrounding area looking for other pedestrians and obstacles. Thus the vision area can be considered as a raster of visionary radials which form a circle sector. Such radial-based shapes are widely adopted by many other crowd models [HLZVT11, Feu00, CC14]. A set of pedestrians located in pedestrian i 's vision area at current time step t_c is defined as pedestrian i 's neighbors, denoted as $Neighbor(i, t_c)$. These neighbors are checked to see if there is any collision that occurs and resolves during the following time steps (see Algorithm 1).

A collision is detected by first calculating the Minimum Mutual Distance between a neighbor j and the subject pedestrian i (i.e., $MMD(i, j)$), which measures the smallest mutual distance between i and j during the time steps that j is a neighbor of i :

$$MMD(i, j) = \min_{t'} \left\| (x'_i, y'_i) - (x'_j, y'_j) \right\| \quad (1)$$

$$t' \in \{t | j \in Neighbor(i, t), t > t_c\}$$

where (x'_i, y'_i) and (x'_j, y'_j) are the positions of i and j at time step t' respectively and $Neighbor(i, t)$ is a set of pedestrians that located in i 's vision area at t . The time step $t_{MMD}(i, j)$ when the subject pedestrian i and neighbor j reaches their smallest mutual distance is:

$$t_{MMD}(i, j) = \arg \min_{t'} \left\| (x'_i, y'_i) - (x'_j, y'_j) \right\| \quad (2)$$

$$t' \in \{t | j \in Neighbor(i, t), t > t_c\}$$

If $MMD(i, j)$ is smaller than $2d$ where d is the diameter of an agent, it means that the mutual distance between i and j will reach minimum point in the future. Neighbor j is identified as a potential collidee. The intuition behind this definition is that if two pedestrians become very close to each other at some future time step, it is highly possible that a collision will happen and they will try to avoid each other in the following time steps until reaching the minimum mutual distance to resolve the collision. In this case, the subject pedestrian i stores the value of $t_{MMD}(i, j) - t_c$ as the Time To Collision with potential collidee j .

ALGORITHM 1: Identify collidee of pedestrian i at time step t_c

Input: neighbors of pedestrian i at t_c , $Neighbor(i, t_c)$
Output: target collidee of pedestrian i at t_c , $collidee(i, t_c)$

```

for every  $j \in Neighbor(i, t_c)$  do
    calculate  $MMD(i, j)$ ;
    calculate  $t_{MMD}(i, j)$ ;
    if  $MMD(i, j) \leq \epsilon$  then
        calculate  $t_{MMD}(i, j) - t_c$  as the Time To Collision;
        if distance between  $i$  and  $j$  is smaller than  $\epsilon$  from  $t_c$  to  $t_{MMD}(i, j)$  then
             $j$  is group-walking potential collidee;
        else
             $j$  is non-group-walking potential collidee;
        end
    else
         $j$  is not a potential collidee;
    end
end
if no potential collidee is identified then
     $collidee(i, t_c) \leftarrow \emptyset$ 
else
    if potential collidees include both group-walking potential collidees and non-group-walking potential collidees then
         $collidee(i, t_c) \leftarrow$  non-group-walking potential collidee with the smallest Time To Collision;
    else
         $collidee(i, t_c) \leftarrow$  potential collidee with the smallest Time To Collision;
    end
end

```

After all neighbors of i are checked for potential collidees, the next step is to identify the target collidee that pedestrian i will be dealing with at following time steps. It is the potential collidee who has the smallest Time To Collision value. This may not be the po-

tential colldee who is the closest to the subject pedestrian i , as a potential colldee who is slightly away may be walking faster than a nearby potential colldee and thus the subject pedestrian is more likely to deal with him/her first. A potential colldee is classified as either a group walking potential colldee (walking in a group with the subject pedestrian) or non-group-walking potential colldee (walking alone). A potential colldee is identified as a group-walking potential colldee if the mutual distance between the potential colldee and the subject pedestrian remains smaller than $2d$ from t_c to $t_{MMD}(i; j)$. It is shown that pedestrians pay more attention to other pedestrians that approaching them face-to-face than those walking with them as groups [EY97]. Moreover, it is more likely that non-group walking potential colldees cause collisions with the subject pedestrian [Wol73]. Therefore, non-group walking potential colldees are granted with higher priority in collision avoidance through the following rules: If potential colldees include both group-walking and non-group-walking potential colldees, the *colldee*(i, t_c) is selected among non-group-walking potential colldees who has the smallest Time to Collision. Otherwise, the potential colldee with the smallest Time To Collision is identified as the colldee *colldee*(i, t_c) of the subject pedestrian i at time step t_c .

After identifying the target colldee at every time step when pedestrian i is in the video, a set of examples are generated and stored in the database. An example contains two parts: state and action. The state $\mathbf{s}$ of each example contains the relative positions of the target colldee in the subject pedestrian's local coordinate system in a predefined number of consecutive time steps (N_p):

$$\mathbf{s} = \{(x_{j,i}^{t_c-N_p}, y_{j,i}^{t_c-N_p}), \dots, (x_{j,i}^{t_c-1}, y_{j,i}^{t_c-1}), (x_{j,i}^{t_c}, y_{j,i}^{t_c})\} \quad (3)$$

where $(x_{j,i}^t, y_{j,i}^t)$ is the relative position of the colldee j in the local coordinate system of the subject pedestrian i at time step t , and t_c is the time step when a new target colldee is identified. A pedestrian's local coordinate system is centered at his/her current position and is pointed to his/her destination (see Figure 2a).

The action $\mathbf{a}$ of an example is a sequence of two-dimensional position vectors of the subject pedestrian in the global coordinate system until a new target colldee is identified:

$$\mathbf{a} = \{(x_i^{t_c}, y_i^{t_c}), (x_i^{t_c+1}, y_i^{t_c+1}), \dots, (x_i^{t_c+l_a}, y_i^{t_c+l_a})\} \quad (4)$$

where (x_i^t, y_i^t) is the position of the subject pedestrian i at time step t in the global coordinate system and l_a is the length of the action which is the consecutive time steps that the subject pedestrian has the same target colldee.

In order to identify similar patterns among examples, an example, which contains state $\mathbf{s}$ and action $\mathbf{a}$, is transformed to a two-dimensional time series sequence-collision avoidance process (CAP)-to facilitate clustering process. The state of an example is the relative position of the target colldee in the local coordinate system of the subject pedestrians, and the action of an example is recorded as the position of the subject pedestrian in the global coordinate. To generate a CAP, the action of an example is also repre-

sented using the relative position of the target colldee in the local coordinate system of the subject pedestrian:

$$\hat{\mathbf{a}} = \{(x_{j,i}^{t_c}, y_{j,i}^{t_c}), (x_{j,i}^{t_c+1}, y_{j,i}^{t_c+1}), \dots, (x_{j,i}^{t_c+l_a}, y_{j,i}^{t_c+l_a})\} \quad (5)$$

Therefore, a CAP is generated by concatenating $\mathbf{s}$ and $\hat{\mathbf{a}}$:

$$\begin{aligned} CAP &= \{\mathbf{s}, \hat{\mathbf{a}}\} \\ &= \{(x_{j,i}^{t_c-N_p}, y_{j,i}^{t_c-N_p}), \dots, (x_{j,i}^{t_c}, y_{j,i}^{t_c}), (x_{j,i}^{t_c}, y_{j,i}^{t_c}), \dots, (x_{j,i}^{t_c+l_a}, y_{j,i}^{t_c+l_a})\} \\ &= \{(rx^1, ry^1), \dots, (rx^{l_{cap}}, ry^{l_{cap}})\} \end{aligned} \quad (6)$$

where (rx^k, ry^k) represents the k th point in the CAP for simplicity and l_{cap} is the length of the CAP, which equals to the sum of the length of the state $N_p + 1$ and the length of the action l_a .

Let's take pedestrians motion in Figure 2 as an example, which shows the situation of the subject pedestrian and his/her neighbors at time steps t_1 and t_2 ($t_1 < t_2$). The mutual distance between pedestrian i and his/her neighbors during the time steps from t_1 to t_5 (when the subject pedestrian leaves the scene) is illustrated in Figure 3. At t_1 , there are seven pedestrians in the vision area of i (see Figure 2a). The walking directions of all pedestrians at t_1 are shown in Figure 2b. The Minimum Mutual Distance for all neighbors at t_1 are calculated and shown with colors in Figure 2c. The pedestrians with smaller Minimum Mutual Distance are shown with darker red colors.

At time step t_1 , neighbor j is the only one whose $MMD(i, j)$ is smaller than ϵ (see Figure 3). Therefore, j is identified as the target colldee of the subject pedestrian i and a new example is generated at t_1 . Neighbor j remains as the target colldee from t_1 to t_2 and the position of the subject pedestrian at these time steps are added to the action. At t_2 , a new potential colldee k appears and j and k are both categorized as non-group-walking potential colldees. The Time To Collision of k is $t_4 - t_2$ which is larger than j 's Time To Collision (i.e., $t_3 - t_2$) and thus j is still the target colldee of the subject pedestrian until the subject pedestrian i and colldee j reach their minimum mutual distance at t_3 . From t_3 to t_4 , k is the only potential colldee and a new example between the subject pedestrian and neighbor k is generated at t_3 . After t_4 , no target colldee is found. For pedestrian i , two examples are generated and thus two CAPs are then generated.

4.2. Distance between Collision Avoidance Processes

Before clustering CAPs, the distance between them has to be quantified. Distance between CAP $i = \{(rx_i^1, ry_i^1), \dots, (rx_i^{l_{cap_i}}, ry_i^{l_{cap_i}})\}$ and CAP $j = \{(rx_j^1, ry_j^1), \dots, (rx_j^{l_{cap_j}}, ry_j^{l_{cap_j}})\}$ is essentially the distance between two time series sequences with various lengths (l_{cap_i} and l_{cap_j}). Therefore, Dynamic Time Warping (DTW) with a window [BC94] is used to find the optimal alignment between them that minimizes the distance. Suppose the u th point alignment between CAP i and CAP j aligns the k th point in CAP i with the m th point in CAP j . Its Euclidean distance can then be defined as:

$$d(u) = \|(rx_i^k, ry_i^k) - (rx_j^m, ry_j^m)\| \quad (7)$$

Figure 2: Example generation.

Figure 3: The mutual distance between pedestrian i and his/her neighbors at each time step when i is in the video.

The DTW distance between CAPs i and j is calculated by summing up the Euclidean distance between all point alignments:

$$DTW(i, j) = \sum_{u \in \Gamma} d(u) \quad (8)$$

where Γ is the total number of point alignments between CAPs i and j and can be defined using the longer length of the two CAPs: $\Gamma = \max(lcap_i, lcap_j)$.

From all the point alignments between CAPs i and j , the number of times that the Euclidean distance between a point alignment is below ϵ is counted:

$$\delta(u) = \begin{cases} 1 & \text{if } d(u) \leq \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$\rho(i, j) = \sum_{u \in \Gamma} \delta(u), \quad (10)$$

The distance between CAPs i and j is the average DTW distance between them, scaled by $\rho(i, j)$:

$$d_{CAP}(i, j) = \frac{DTW(i, j)}{\rho(i, j) |\Gamma|} \quad (11)$$

By doing this, two CAPs that have similar points over a longer period of time would have a smaller distance. Imposing aggregated

measurement over time helps to get better and meaningful clustering result against tracking noise.

4.3. Clustering Collision Avoidance Processes

We apply a bottom-up hierarchical clustering algorithm [DE84] which starts with treating CAPs as separate clusters and merge those clusters with smallest distances to gradually build larger clusters. This clustering algorithm returns a hierarchy of clusters, a structure that may contain more valuable information and does not require us to specify in advance the number of clusters. It also allows us to determine the best number of clusters based on the output hierarchy. One of the goal of data mining is to infer knowledge from data and choosing the number of clusters automatically is one important task. Besides, this clustering algorithm operates on distance matrix defined using a distance measure such as the one defined above and does not require elements to be in Euclidean space.

In order to merge two clusters C_i and C_j , three linkage criteria are commonly used to calculate the distance $d_C(C_i, C_j)$ between the two clusters C_i and C_j :

- Single linkage:
 $d_C(C_i, C_j) = \min\{d_{CAP}(i, j) : i \in C_i, j \in C_j\}$
- Complete linkage:
 $d_C(C_i, C_j) = \max\{d_{CAP}(i, j) : i \in C_i, j \in C_j\}$
- Average linkage:
 $d_C(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{i \in C_i} \sum_{j \in C_j} d_{CAP}(i, j)$

where i and j are two CAPs in clusters C_i and C_j respectively.

These commonly used linkage criteria tend to be sensitive to the extreme entities in two clusters (e.g., smallest distance and largest distance), especially in the case of human movement data which involve motion uncertainty and tracking noise. We propose another linkage approach, the modified Hausdorff distance to overcome such problem. Similar to [GCR12], for two clusters C_i and C_j , the directed distance from C_i to C_j is:

$$h(C_i, C_j) = \frac{1}{|C_i| \times |\Psi_i|} \sum_{i \in C_i} \sum_{j \in \Psi_i} d_{CAP}(i, j) \quad (12)$$

where Ψ_i is a set consisting of CAPs in cluster C_j whose distances

to CAP_i are in the larger half among all the distances $d_{CAP}(i, l)$, $l \in C_j$. Hence $|\Psi_i| = \lceil |C_j|/2 \rceil$.

Then the symmetric distance between C_i and C_j is

$$d_C(C_i, C_j) = \frac{1}{2}(h(C_i, C_j) + h(C_j, C_i)) \quad (13)$$

Determining the number of clusters of the data is an open question. By using hierarchical clustering, we can test various number of clusters after building the hierarchy of clusters by simply cutting the hierarchy at different levels. Numerically, a good clustering partition means the distances between CAPs in the same cluster are small and the distances between CAPs in the different clusters are large. Therefore, silhouette coefficient [Rou87] is adopted to evaluate the clustering result with different number of clusters and to select the most precise one among them.

The silhouette coefficient works as follows. For CAP i in one cluster, we first calculate its average distance to all other CAPs within the same cluster, denoted as d_i^{in} ; then calculate its average distance to all CAPs located in other clusters, denoted as d_i^{out} . The silhouette coefficient for the CAP i is:

$$\lambda_i = \frac{d_i^{out} - d_i^{in}}{\max(d_i^{out}, d_i^{in})} \quad (14)$$

The average silhouette coefficient of all CAPs is calculated to measure the goodness of one clustering:

$$\lambda = \frac{\sum_{i=1}^n \lambda_i}{n} \quad (15)$$

where n is the total number of CAPs in the database. The value of λ_i varies between -1 and 1 and the larger the better.

For each number of clusters, we calculate its average silhouette coefficient and choose the one with the highest score.

5. Training the neural network classifier and k -d Trees

After clustering, each CAP belongs to one cluster, and the corresponding example is assigned with the same cluster membership. We use this information to train a classifier, which will then be used to classify a query state to one cluster in the simulation. We apply the neural network classifier [Tu96]. The advantage of neural network, comparing to other supervised learning algorithm is its ability to implicitly detect complex nonlinear relationships between dependent and independent variables and the availability of multiple training algorithms [Tu96]. Artificial neural network is able to approximate any function, regardless of its linearity and thus suitable for complex problems. In our classification problem, it is difficult to formulate a possible non-linear relationship of input variables and thus artificial neural network is employed.

The input of the neural network is the state of an example and the output is the example's cluster membership. The cluster membership of an example is described by a column vector

Figure 4: Neighborhood formation used in action selection.

$C = (C_1, C_2, \dots, C_k, \dots)^T$. If an example belongs to cluster i , $C_i = 1$. Otherwise $C_i = 0$.

The neural network has three layers. The input layer has $(N_p + 1) \times 2$ neurons (that is, $(N_p + 1)$ two dimensional coordinates). The number of neurons in the output layer equals to the number of clusters (M). The number of neurons in the hidden layer is mostly determined dynamically during training. The initial number of neurons is set based on a commonly used empirically-derived rule: the optimal size of the hidden layer is usually between the size of the input and the size of the output layers [Hea08]. After testing all possible values, the configuration that give the highest average prediction accuracy is selected. Amongst all generated examples, 50 percent are used to train the neural network, 15 percent are used for validation to prevent overfitting and the remaining 35 percent are used to test the performance of the neural network. The network is trained using Levenberg-Marquardt optimization [Mor78].

During the simulation, after classifying a query state into a cluster, a set of possible actions are obtained. In order to search for the optimal action from these possible actions, we select the following features to differentiate between actions:

- Self speed: The average speed of a pedestrian.
- Intended walking direction: angle between the current moving direction and the vector pointing from pedestrians's current position to the goal position.
- Neighborhood formation: The general information of the motion of nearby pedestrians. The vision area around an agent is divided into strips as shown in Figure 4 where d denotes the diameter of an agent and the number of strips depends on the vision radius R . Each strip is assigned with one of the three values: 1 if the nearest neighbor in this strips is walking in the same direction; -1 if the nearest neighbor in this strip is walking in the opposite direction and 0 if no neighbors is found in this strip.

Actions belonging to the same cluster are organized into one k -d tree using the features defined above. During the simulation, these features are used as input to query the k -d tree to find the best action.

Therefore, after pre-processing, a neural network classifier and M k -d trees, one for each cluster, are generated for utilization in simulation.

6. Simulation

During the simulation, for each agent at every time step, the target collidree is first identified using the method described in Section 4.1

ALGORITHM 2: Velocity Prediction

Input: Surrounding situations of pedestrian i
Output: $\mathbf{v}_i(t+1)$: velocity of pedestrian i at next time step $t+1$

```

collidee( $i, t$ )  $\leftarrow$  find_collidee();
if collidee( $i, t$ ) ==  $\emptyset$  then
    |  $\mathbf{v}_i(t+1) \leftarrow$  preferred_velocity;
else
    | if me.saved_action  $\neq \emptyset$  and
      | collidee( $i, t$ ) == me.saved_collidee then
      | |  $\mathbf{v}_i(t+1) \leftarrow$  execute(me.saved_action);
    | else
      | me.saved_collidee  $\leftarrow$  collidee( $i, t$ );
      |  $\mathbf{s}_i^t \leftarrow$  generate_state(me, collidee( $i, t$ ));
      | cluster  $\leftarrow$  classify( $\mathbf{s}_i^t$ );
      | me.saved_action  $\leftarrow$  search_action(cluster);
      |  $\mathbf{v}_i(t+1) \leftarrow$  execute(me.saved_action);
    | end
  end
end

```

with some minor modification. The future position of other pedestrians are unknown during simulations and thus are extrapolated linearly using their current velocities. If no target collidee is found, the preferred velocity is selected (see Algorithm 2). If the target collidee remains the same, no velocity prediction is required and the next position in the current action is selected to calculate the velocity. If the current action sequence runs out or the target collidee changes, a new action is selected in two phases (see Figure 1b). First, the state of the subject pedestrian is classified into one cluster using the neural network trained before the simulation. Second, an action is selected through *search_action()* function. The k -d tree of the identified cluster is searched using additional features and K -nearest actions (sorted in ascending order according to the distance to the input action) are returned. These actions (i.e., position sequences) are converted to velocity sequences respectively, which are then tested through the *collision_detection()* function. The first action that does not cause collisions with all other observed pedestrians and static obstacles is selected for the subject agent. If no collision free action is found, the action that has the largest number of collision free time steps (which must be larger than 1) is selected. If all K actions will collide at next time step, the subject pedestrian performs an emergency stop at next time step. The subject agent will have to re-assess the situation and perform the classification and k -d tree searching at next time step.

The *collision_detection()* function mentioned above takes a testing velocity sequence, the current velocities of other pedestrians and locations of static obstacles as input, and returns a boolean value indicating whether a collision will occur if the subject agent follows the testing velocity sequence. The velocities of other agents are either assigned or extrapolated. Following the concept of Velocity Obstacle [FS98], if the distance between the subject pedestrian and any pedestrians or obstacles nearby is smaller than $r_i + r_j$ at any testing time step, where r_i and r_j are radii of the subject pedestrian and j th nearby pedestrian (if agents have the same radius,

$r_i + r_j = d$), the testing velocity sequence is identified as having a collision.

7. Experimental Results

We compare the simulation results of CLUST model with other four state-of-the-art models against the real world motion data in this section. To prove the generality of the approach, in addition to train and test on the same dataset, the experiments are also conducted by applying the CLUST model trained using one dataset to a different dataset. If the same dataset is used for both training and testing, the simplest kind of model validation method is the hold-out method. The dataset is separated into two sets, namely training set and testing set. The model is trained using the training set only and are then used to predict the values in the testing set. This kind of method may depend significantly on how the dataset is divided. N -fold cross validation is a commonly used method to improve over the hold-out method. Suppose that the motion data in one dataset are evenly partitioned into N subsets. The holdout method is repeated N times. Each time, one of the N subsets is used as testing set and the other $N - 1$ subsets are used together to train the model. In this section, if the same dataset is used for both training and testing the CLUST model, 10-fold cross validation is applied to the *jp* dataset and 2-fold cross validation is used in the *cross* dataset because of its limited input size. Otherwise, if one dataset is used for training and another dataset is used for testing, cross validation is not applied and the entire dataset is used for either training or testing. The CLUST model trained using *jp* or *cross* dataset is referred to as CLUST(*jp*) or CLUST(*cross*) model respectively.

The crowd models are validated with real world data both qualitatively and quantitatively. On one hand various metrics are introduced to compare between the motion of real pedestrians and simulated agents statistically and then identify the model that replicates crowd motion most accurately. These metrics capture a wide range of motion features. On the other hand, the simulation trajectories of different models are visually compared with the trajectories in the video. Moreover, the simulation frame rates of different models are calculated to compare their timing performance.

7.1. Datasets

We adopt two real-world datasets *jp* and *cross* as the ground truth and evaluation scenarios (Figure 5).

jp. The sequence in *jp* dataset was recorded at Jurong Point shopping mall in Singapore in 2013. The scenario is a corridor area with shops at each side. The trajectories of pedestrians were manually labeled at every 0.1 s together with the boundary of the walking area. Pedestrians were mainly walking in two directions.

cross Dataset *cross* contains pedestrians' motion at a diagonal crossing at Oxford Circus in central London. The trajectories of pedestrians were annotated manually every 10 frames per second. Although there were lines on the ground guiding pedestrians' motion, it did not post a strict constraint on pedestrians and they frequently walked out of the designed area in the video. Therefore no obstacles were annotated in this scenario. Unlike previous datasets, pedestrians walked towards from four directions at the same time.

Figure 5: Real-world datasets used in the experiments.

Trajectories in `jp` and `cross` datasets are transformed from image coordinate to world coordinated using the method described in [ZCLY15].

To statistically quantify and compare the features of these adopted datasets, the following statistics are calculated:

- ρ : average density of crowd (m^{-2}).
- v : average speed of crowd (ms^{-1}).
- n_{MMA} : the number of main motion axis.

The average crowd density and speed are calculated based on Voronoi partition as suggested in [Zha12]. The agents' motion directions are categorized into four MMAs as suggested in Figure 5c, i.e., north, south, west and east. Each main motion axis occupies a section of $\frac{\pi}{2}$. The number of MMAs of pedestrians in the video is recorded as n_{MMA} .

The statistics of the datasets used in the evaluation are shown in Table 2. The `cross` dataset has higher density and contains pedestrians walking from four MMAs. The `jp` dataset mainly contain two MMAs.

	Trajectories	Annotation	ρ	v	n_{MMA}
<code>jp</code>	332	10	0.2114	1.6760	2
<code>cross</code>	93	10	0.3145	1.6373	4

Table 2: The information of datasets.

7.2. State-Of-the-Art Models

Four state-of-the-art models, social force (SF), RVO2, ANN, and PAG, are selected as a comparison to the proposed model. They are discussed in the related work section. We implement our model and the above state-of-the-art models in Java using MASON multi-agent simulation library [LCRP*05] and apply the same settings for all five models when possible. The values of some of model-specific parameters are obtained from the respective model's paper. Others are empirically selected such that they produce better performance.

7.3. Simulation Settings

The start positions, goal positions and preferred speed of agents are generated from the real-world motion data with the time step of

entering the scenario and they are applied to all models to drive the simulations. The goal position of each pedestrian is identified as his last position in the scenario and the preferred speed of a pedestrian is calculated as his average speed in the scenario approximately.

7.4. Pre-processing of the Proposed Model

The discovered clusters of CAPs (i.e., the relative positions of the target collidees) in `jp` and `cross` datasets are shown in Figures 6 and 7. Different clusters are shown in different colors.

The final number of clusters of CAPs is determined as the one which has the largest silhouette coefficient. The final number of clusters clustered using different linkage criteria and the corresponding silhouette coefficient values are listed in Tables 3 and 4 for `jp` and `cross` datasets respectively. It is shown that the modified Hausdorff distance generates the highest average silhouette coefficient, i.e., the best clustering result, among all linkage criteria.

	Average	Complete	Single	Hausdorff
Number of clusters	6	7	2	7
Coefficient	0.5095	0.464	0.4052	0.6055

Table 3: The number of clusters and its corresponding silhouette coefficient using different linkage criteria in the `jp` scenario.

	Average	Complete	Single	Hausdorff
Number of clusters	3	12	2	7
Coefficient	0.6782	0.6151	0.0474	0.7009

Table 4: The number of clusters and its corresponding silhouette coefficient using different linkage criteria in the `cross` scenario.

It is clear from Figure 6, that CAPs representing the subject pedestrian walking together with the collidees exist in the `jp` scenario, while in the `cross` scenario such phenomenon is not identified (see Figure 7). This is because pedestrians in the `cross` scenario are constantly facing potential collisions with the oncoming pedestrians and thus collidees are mostly non-group-walking pedestrians. In the `cross` scenario, non-group-walking collidees approximately come from three main directions: front, left and right. While in the `jp` scenario, non-group-walking collidees are

Figure 6: Clusters clustered using different linkage criteria in the `jp` scenario. Figure 7: Clusters clustered using different linkage criteria in the `cross` scenario.

mainly from the front direction. As describe in [RD06], there are three types of collisions between two pedestrians: *Towards* collision which may occur when two pedestrians are walking straight into each other, or with some smaller overlap of their bodies; *Glancing* collision which is a side-on collision between two pedestrians whose paths will cross in the future; and *Away* collision which is the situation where the subject is behind the collidee, but plans to overtake it. Accordingly, the CAPs show the similar trend. The collidees coming from the front direction in the `jp` and `cross` scenarios generate *Towards* collision on the subject pedestrian and the collidees who walk from the left and right direction in the `cross` scenario cause *Glancing* collision with the subject pedestrian. In both scenarios, *Away* collision is not identified because in the `jp` scenario, pedestrians walk in a very ease mode in a shopping mall and in the `cross` scenario, there is not much space available for overtaking. Also, in the `jp` scenario, pedestrians maintain larger personal space during interaction than those in the `cross` scenario.

Different linkage criteria influence the clustering results. Hierarchical clustering using single linkage results in two clusters in both scenarios, one of which has very large size, similar to the discussion in [MRS*08]. The clustering results also indicate that complete linkage generates compact clusters. It has the largest number of clusters among all linkage criteria in both scenarios. It is

noted that the number of clusters generated by different linkage criteria in the `jp` scenario are similar to each other except for the single linkage. The silhouette coefficient value reflects how well the mined clusters are separated. Among four linkage criteria, the modified Hausdorff approach scores the highest, followed by the average linkage and the complete linkage. Single linkage approach has the smallest coefficient especially in the `cross` scenario which is approximate zero. Figure 6 shows that in the `jp` scenario, except for the single linkage, all other linkage criteria produce visually reasonable clustering results, where the examples are mainly clustered as four groups: two clusters of *Towards* collision respectively, and two clusters of pedestrians walking together with the pedestrians at the right side or left side. The advantage of modified Hausdorff approach is better illustrated in the `cross` scenario where the six larger clusters clustered using the modified Hausdorff approach contain two clusters of *Towards* collision and four clusters of *glancing* collision.

After clustering, a neural network classifier is trained to map between examples and their cluster memberships. Figure 8 shows the confusion matrix of the trained classifiers of the `jp` scenario and the `cross` scenario respectively. Both classifiers have high prediction accuracy and thus are able to classify sensed state of agents in simulations accurately. The difference between clusters in `jp`

Figure 8: Classification accuracy of the trained neural network in the JP and the Cross scenario..

dataset (Figure 6) is more distinctive than those in `cross` dataset (Figure 7) as discussed in the clustering results. For `jp` dataset, *Towards* collisions, where two pedestrians are walking straight into each other, are identified. On the other hand, for `cross` dataset, since pedestrians are walking in more directions, besides *Towards* collisions, *Glancing* collisions, which is a side-on collision between two pedestrians, are also identified. Therefore, the classification accuracy depends on the level of distinctiveness of the clusters.

7.5. Quality Comparisons

In order to numerically compare different models' performance in predicting agents' motion, a list of metrics which have been introduced in previous studies [Fru71, GCC*10, KHvBO09, SKFR09, Hu14] are adopted to quantify agents' motion in various aspects:

- ϵ : average collision rate at each time step.
- L_i : travel distance of agent i (m).
- D_i : perpendicular deviation distance of agent i (m).
- ΔV_i : speed change of agent i (m s^{-1}).
- ΔA_i : angle change of agent i ($^\circ$).
- E_i : biomechanical energy spent by agent i (J).
- $steerE_i$: biomechanical energy spent by agent i in collision avoidance (J).

Collisions may occur between pedestrians in the real world. Pedestrians in daily life are less likely and willing to collide with others especially in normal emotion condition. The number of collisions can be used as a metrics to quantify the quality of crowd models. We calculate the collision rate using the number of agents who have collision divided by the total number of agents in the scenario instead of the absolute value of collisions in order to make the result comparable across different scenarios. Agents are represented as a circle with a center and a radius in both crowd videos and simulations. As defined in Section 6, if the distance between the center of two agents is smaller than the sum of their radius, two agents are considered as collision agents (i.e., collided with each other). If the circle of an agent intersects with a static obstacle, the agent is identified as collision agent as well.

L_i is defined as the sum of agent i 's displacement in one simulation run:

$$L_i = \sum_{t=1}^{T_i} \|\mathbf{p}_i(t) - \mathbf{p}_i(t-1)\| \quad (16)$$

where $\mathbf{p}_i(t)$ denotes the position vector of i at time t and T_i is the total number of time steps during which agent i is in the scenario.

The perpendicular deviation distance of an agent D_i has a direct influence on the appearance of the simulation result and is calculated as:

$$D_i = \sum_{t=0}^{T_i} PD(t) \quad (17)$$

where $PD(t)$ denotes the perpendicular distance from point $\mathbf{p}_i(t)$ to the line that connects agent i 's start position and end position.

ΔV_i and ΔA_i reflect the locomotion of agents and are calculated as:

$$\Delta V_i = \sum_{t=1}^{T_i} \|\mathbf{v}_i(t) - \mathbf{v}_i(t-1)\| \quad (18)$$

$$\Delta A_i = \sum_{t=1}^{T_i} \min(|\theta_i(t) - \theta_i(t-1)|, 2\pi - |\theta_i(t) - \theta_i(t-1)|) \quad (19)$$

where $\mathbf{v}_i(t)$ is the velocity vector of i at time step t and $\theta_i(t)$ means the walking angle of agent i at time step t ($\theta_i(t) \in [0^\circ, 360^\circ]$).

The metrics above are similar to those introduced in [Hu14] but are measured based on every single agent instead of all the agents in the simulation.

As introduced in [GCC*10], the biomechanical energy spent by agent i along its path is calculated as:

$$E_i = \sum_{t=0}^{T_i} m_i (e_s + e_w \|\mathbf{v}_i(t)\|^2) \Delta t \quad (20)$$

where e_s and e_w are constants with estimated values of $2.23 \text{ J kg}^{-1} \text{ s}^{-1}$ and $1.26 \text{ J s kg}^{-1} \text{ m}^{-2}$ for an average human as suggested in [Whi14] and the mass m_i is set as 70 kg in this paper. In order to measure the biomechanical energy agent i spends to interact with other agents and obstacles, we use the metric $steerE_i$ proposed in [Hu14], which is calculated as:

$$steerE_i = \sum_{t=0}^{T_i} m_i (e_s + e_w \|\mathbf{v}_i(t) - \mathbf{v}_i(t-1)\|^2) \Delta t \quad (21)$$

These metrics are calculated both for the real-world pedestrians using video data and the simulated agents. The validity of crowd models are determined by comparing the metrics values between crowd models and the real-world data. Specifically, for each metric, we first calculate the error between the metric value of an agent in crowd models and its corresponding pedestrian in the real-world dataset and then average across all agents to obtain the average error of the metric. For example, the average travel distance error is defined as $\frac{1}{N} \sum_{i=1}^N |L_i - \tilde{L}_i|$, where $\tilde{L}_i$ denotes the travel distance of real pedestrian i (i.e., the counterparts of agent i) in the video and N is the total number of agents in the simulation.

Table 5 summaries the average collision rate of crowd models and real pedestrians on real world scenarios. For each dataset, the CLUST model that trained using the same dataset is considered and the result is averaged using cross validation. It is noted that almost no collision occurs if agents are simulated using the RVO2 and SF models on all scenarios. While the ANN, PAG and CLUST

models are more likely to generate collisions between agents. The CLUST model generates less collisions compared to the ANN and PAG models, due to the precise extraction of collision avoidance behaviors from the real world data and the two-staged action selection method.

ε	GT	RVO2	SF	ANN	PAG	CLUST
jp	0.00	0.00	0.00	0.09	0.05	0.04
cross	0.02	0.00	0.00	0.11	0.04	0.02

Table 5: Average collision rate.

7.5.1. jp

Figure 9: Simulation results for the `jp` scenario.

Figure 9 illustrates the trajectories of pedestrians (agents) up to a similar time step. *agt1* is walking towards the left side of the scenario and *agt2* is moving in the opposite direction (see Figure 9a). Agents in the RVO2 and SF models generate perfect symmetric motion with almost the same values of deviations for the two agents (Figures 9b and 9c). These deviations are smaller than those of the real pedestrians. The trajectories of the SF model has a sudden motion change when two agents are near (Figure 9c). This indicates that agents in the SF model start to avoid each other at a later stage. Agents in the ANN and PAG models are less consistent in selecting collision avoidance behaviors (Figures 9d and 9e) due to the inaccuracy in searching for the nearest example for the current situation. The CLUST(jp) and CLUST(cross) models produces the trajectories of *agt1* and *agt2* that more consistent with *agt1* and *agt2* in the video and maintains similar deviations to that of real pedestrians (Figures 9f and 9g). This indicates a more accurate action selection method.

The average errors of metrics for various models are shown in Table 6. It is noted that the CLUST model outperforms other models in terms of travel distance, deviation, angle change, energy spent and steering energy consumed. This demonstrates that

Metrics	L_i	D_i	ΔV_i	ΔA_i	E_i	$steerE_i$
RVO2	0.829	16.33	6.275	323.8	31.37	18.70
SF	0.789	15.24	5.879	292.6	29.76	19.40
ANN	0.751	13.96	3.374	185.2	28.41	17.88
PAG	6.074	23.1	5.829	354.3	34.07	37.78
CLUST(jp)	0.693	10.23	3.554	179.4	25.52	15.42
CLUST(cross)	0.729	10.34	3.825	179.6	25.81	15.89

Table 6: Errors in the `jp` scenario.

the crowd simulated by the CLUST model behaves more like real world crowd in terms of these criteria. As for the speed change, the ANN model scores the best, while the CLUST model is a close second best. The CLUST models trained by different datasets generate similar amount of errors, because the pedestrians' situations in the `jp` dataset (i.e., interacting with oncoming pedestrians in the opposite direction) also appear in the `cross` dataset.

7.5.2. cross

Figure 10: Simulation results for the `cross` scenario.

Figure 10 compares the trajectories of pedestrians (agents) up to an early time step in the `cross` scenario. With an increased density and number of pedestrians flows (pedestrians walk from four MMAs instead of two), pedestrians show intention to form groups with nearby pedestrians who walk in the same MMA when

avoiding pedestrians from other MMAs. It is observed that pedestrians in the same group are likely to share similar motion behaviors and an individual prefers to detour around a small group rather than intruding into them (Figure 10a). Therefore, five pedestrians who walk from bottom to top turn left to avoid pedestrians from other MMAs (see Figure 10a). The corresponding agents in the CLUST(cross) model walk in a similar manner (deviate to the left as shown in Figure 10g) and generate similar amount of average deviations. However, agents trained by CLUST(jp) model, do not present such group behaviors obviously (Figure 10f), because the jp dataset does not cover all the situations an agent may encounter in the cross dataset. Similarly, agents in the RVO2, SF, ANN, and PAG models also show little sign of grouping behavior (Figures 10b, 10c, 10d, and 10e). Consequently, agents in the ANN and PAG models result in collisions more frequently than other models (see Table 5). Although no serious congestions are observed in the RVO2 and SF models, they may lead to serious congestions and much longer time to finish the simulation, if crowd density keep increasing.

Metrics	L_i	D_i	ΔV_i	ΔA_i	E_i	$steerE_i$
RVO2	0.391	4.930	17.24	147.9	117.1	90.42
SF	1.555	3.989	11.62	186.69	256.6	145.3
ANN	0.744	6.074	8.852	100.3	151.0	104.0
PAG	3.614	15.01	9.376	134.7	274.5	141.3
CLUST(jp)	1.165	4.346	12.97	93.25	106.5	81.52
CLUST(cross)	0.382	3.447	10.06	82.80	96.17	66.07

Table 7: Errors in the cross scenario.

The summary of average errors for the cross scenario is shown in Table 7. Similar to those observed in the jp scenario, the ANN model produces the smallest error in the speed change ΔV_i , while the CLUST model achieves the smallest error in all other metrics. The average errors of the ANN and PAG models increase largely due to the limited input size of the real world data which limits the coverage of extracted examples in the database in the ANN model and the connectivity of the motion graph in the PAG model. In contrast, the trained neural network classifier in the CLUST model learns the relationship between input state and output cluster membership, and hence it has a better prediction of previously unseen input states. Agents simulated by the CLUST(jp) model show larger errors than that by the CLUST(cross) model, because the jp dataset does not cover all the situations a pedestrian may encounter in the cross dataset. Pedestrians in the jp dataset mainly walk from the opposite directions; whereas pedestrians in the cross dataset need to interact with pedestrians from three other directions.

7.6. Missing Rate of the CLUST Model

When all K nearest actions will collide at next time step, it means that an agent fails to find an example to follow. This may happen if agents encounter new situations that are largely different from the training video. It is unlikely to happen if the scenario is similar (i.e., trained and tested on the same or similar datasets). We record the total number of times agents cannot find an example to follow during the entire simulation as the missing rate in the following

four situations: (1) the CLUST model trained by and applied to jp dataset (denoted as CLUST(jp) $\rightarrow$ jp); (2) the CLUST model trained by jp dataset and applied to cross dataset (denoted as CLUST(jp) $\rightarrow$ cross); (3) the CLUST model trained by cross dataset and applied to jp dataset (denoted as CLUST(cross) $\rightarrow$ jp); (4) the CLUST model trained by and applied to cross dataset (denoted as CLUST(cross) $\rightarrow$ cross). If the same dataset is used for both training and testing the CLUST model, cross validation is applied. The results are shown in Table 8. It is observed that the missing rate is the lowest if the CLUST model is trained and tested using the same dataset. The missing rate for CLUST(cross) model tested on jp dataset is lower than CLUST(jp) model applied to cross dataset. This is because the jp dataset does not cover all the situations an agent may encounter in the cross dataset, but the pedestrians' situations in the jp dataset (i.e., interacting with oncoming pedestrians in the opposite direction) also appear in the cross dataset.

Models $\rightarrow$ Scenario	Missing Rate
CLUST(jp) $\rightarrow$ jp	11
CLUST(jp) $\rightarrow$ cross	25
CLUST(cross) $\rightarrow$ jp	13
CLUST(cross) $\rightarrow$ cross	7

Table 8: The missing rate for CLUST models tested on jp and cross datasets respectively.

7.7. Timing Comparisons

Figure 11: Average computation time per time step of each procedures in five datasets.

An agent in the CLUST model typically goes through three procedures in predicting future trajectories: scanning nearby agents to identify any collisions (t_{identify}), classifying the observed state into one cluster (t_{classify}) and selecting one optimal collision free action from the cluster (t_{action}). To further investigate the computation costs of the three procedures of motion prediction in the CLUST model, we calculate the average computation time per time step per agent of each procedure in various real-world scenarios. For each dataset, the CLUST model that trained using the same

	RVO2	SF	ANN	PAG	CLUST
jp	126.53	121.42	86.14	139.44	142.20
cross	95.32	96.22	50.61	65.60	65.93

Table 9: Simulation frame rates of five models on all datasets.

dataset is considered and the result is averaged using cross validation. The results are shown in Figure 11. It is observed that for the jp dataset, the procedure of searching for the optimal action within one cluster dominates the entire computation cost, while identification and classification procedures generally take much shorter time. Agents in the cross dataset require longer time to identify collisions as a result of increased density. The shorter time of agents' action selection procedure is attributed to a smaller input database size, that results in a faster k -d tree query, and the shorter length of actions. Actions extracted from the cross dataset have shorter length because pedestrians modify their motion behaviors more dynamically due to the faster changing surrounding situations in an increased density.

The simulation frame rates (not including visualization) of five simulation models tested on the datasets are shown in Table 9. For each dataset, the CLUST model that trained using the same dataset is considered and the result is averaged using cross validation. If the simulation time step of a model is δt , a frame rate larger than $1/\delta t$ indicates that the model is possible to execute in real time. We emphasize more on the relative comparisons of models instead of absolute values because a more optimized code and more powerful hardware would surely improve the simulation frame rates of the same model. From the results shown in Table 9, generally the RVO2 and SF models run faster than the data-driven models because real-time database searching in these data-driven models requires longer computation time. Among data-driven models, the ANN model runs the slowest because both the PAG and CLUST models implement optimization strategies to reduce database searching time.

8. Conclusion and Future Work

In this paper, we propose a data-driven model to simulate crowd behaviors by mining examples from crowd videos. We apply a clustering approach to automatically discover clusters from pedestrian's trajectories and apply them in the real-time simulation. We test our approach with four state-of-the-art simulation models on different types of scenarios. The simulation results demonstrate that our approach is effective in the tested scenarios in terms of comparing measured metrics from the simulation with those of the video data. Furthermore, the result shows that if the CLUST model is trained using a dataset with more complicated scenario, it can be applied to another dataset with less complicated scenario (as shown in jp dataset). In addition the simulation frame rate is comparable to that of the PAG model. This demonstrates that two level example matching helps to improve both accuracy and efficiency. The simulation results also show insights of the various simulation models. The RVO2 model produces smaller motion changes compared to the real pedestrians. The motion changes of agents simulated by

the SF model increase significantly as crowd density increases. The ANN model captures the speed of agents most accurately among all the models. The ANN and PAG models generate less smooth and structural trajectories for agents. As for the computation time, The RVO2 and SF models run faster than data-driven models. The ANN model is the slowest among the data driven models while the performance of the CLUST model is comparable to that of the PAG model.

One of the limitation of the proposed model is that if the crowd become very dense, considering only one nearby agent will not be enough to accurately mine clusters. One possible solution and future work is to consider the k most important neighbors and modify the distance function accordingly. In addition, more motion features between the subject pedestrian and the colidee such as relative velocities can also be included to better represent the collision avoidance behaviors of real pedestrians. Finally, during the simulation, the future positions of other agents are currently predicted by linear extrapolation based on the current velocities if no velocity sequences are assigned to them. Such an approach can be further improved by taking into account the past velocities of agents. A regression model can be trained to learn the motion of agents and thus predict future motion.

References

- [ABW06] ANTONINI G., BIERLAIRE M., WEBER M.: Discrete choice models of pedestrian walking behavior. *Transportation Research Part B: Methodological* 40, 8 (2006), 667–687. 1, 3
- [ALL*11] AYDT H., LEES M., LUO L., CAI W., LOW M. Y.-H., KADIRVELEN S. K.: A computational model of emotions for agent-based crowds in serious games. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on* (2011), vol. 2, IEEE, pp. 72–80. 1
- [BA98] BLUE V. J., ADLER J. L.: Emergent fundamental pedestrian flows from cellular automata microsimulation. *Transportation Research Record: Journal of the Transportation Research Board* 1644, 1 (1998), 29–36. 3
- [BA01] BLUE V. J., ADLER J. L.: Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B: Methodological* 35, 3 (2001), 293–312. 3
- [BC94] BERNDT D. J., CLIFFORD J.: Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (1994), AAAI Press, pp. 359–370. 6
- [BKSZ01] BURSTEDDE C., KLAUCK K., SCHADSCHNEIDER A., ZIT-TARTZ J.: Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications* 295, 3 (2001), 507–525. 3
- [CA07] CHERNOVA S., ARKIN R. C.: From deliberative to routine behaviors: a cognitively inspired action-selection mechanism for routine behavior capture. *Adaptive Behavior* 15, 2 (2007), 199–216. 2
- [CC14] CHARALAMBOUS P., CHRYSANTHOU Y.: The pag crowd: A graph based approach for efficient data-driven crowd simulation. *Computer Graphics Forum* 33, 8 (2014), 95–108. 2, 3, 5
- [CKGC14] CHARALAMBOUS P., KARAMOUZAS I., GUY S. J., CHRYSANTHOU Y.: A data-driven framework for visual crowd analysis. *Computer Graphics Forum* 33, 7 (2014), 41–50. 3
- [DE84] DAY W. H., EDELSBRUNNER H.: Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification* 1, 1 (1984), 7–24. 7

- [DH03a] DAAMEN W., HOOGENDOORN S.: Controlled experiments to derive walking behaviour. *European journal of transport and infrastructure research EJTI*, 3 (1) (2003). 3
- [DH03b] DAAMEN W., HOOGENDOORN S.: Experimental research of pedestrian walking behavior. *Transportation Research Record: Journal of the Transportation Research Board*, 1828 (2003), 20–30. 3
- [EY97] EGETH H. E., YANTIS S.: Visual attention: Control, representation, and time course. *Annual review of psychology* 48, 1 (1997), 269–297. 6
- [Feu00] FEURTEY F.: Simulating the collision avoidance behavior of pedestrians. *Master's Thesis* (2000). 5
- [Fru71] FRUIN J. J.: *Pedestrian planning and design*. Tech. rep., 1971. 12
- [FS98] FIORINI P., SHILLER Z.: Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research* 17, 7 (1998), 760–772. 3, 9
- [GCC*10] GUY S. J., CHHUGANI J., CURTIS S., DUBEY P., LIN M., MANOCHA D.: Pedestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation* (2010), Eurographics Association, pp. 119–128. 1, 12
- [GCK*09] GUY S. J., CHHUGANI J., KIM C., SATISH N., LIN M., MANOCHA D., DUBEY P.: Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 177–187. 3
- [GCR12] GE W., COLLINS R. T., RUBACK R. B.: Vision-based analysis of small groups in pedestrian crowds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34, 5 (2012), 1003–1016. 7
- [GLM10] GUY S. J., LIN M. C., MANOCHA D.: Modeling collision avoidance behavior for virtual humans. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 2-Volume 2* (2010), International Foundation for Autonomous Agents and Multiagent Systems, pp. 575–582. 3
- [Gof09] GOFFMAN E.: *Relations in public*. Transaction Publishers, 2009. 5
- [GVDBL*12] GUY S. J., VAN DEN BERG J., LIU W., LAU R., LIN M. C., MANOCHA D.: A statistical similarity measure for aggregate crowd dynamics. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 190. 3
- [GvdBLM10] GUY S. J., VAN DEN BERG J., LIN M. C., MANOCHA D.: Geometric methods for multi-agent collision avoidance. In *Proceedings of the twenty-sixth annual symposium on Computational geometry* (2010), ACM, pp. 115–116. 3
- [HD05] HOOGENDOORN S. P., DAAMEN W.: Pedestrian behavior at bottlenecks. *Transportation science* 39, 2 (2005), 147–159. 3
- [HDB03] HOOGENDOORN S. P., DAAMEN W., BOVY P. H.: Extracting microscopic pedestrian characteristics from video data. In *Transportation Research Board Annual Meeting* (2003), pp. 1–15. 3
- [Hea08] HEATON J.: *Introduction to neural networks with Java*. Heaton Research, Inc., 2008. 8
- [HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature* 407, 6803 (2000), 487–490. 2, 3
- [HLZ13] HU N., LEES M. H., ZHOU S.: A pattern-based modeling framework for simulating human-like pedestrian steering behaviors. In *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology* (2013), ACM, pp. 179–188. 1
- [HLZVT11] HU N., LEES M., ZHOU S., VISWANATHAN T. V.: Pattern based motion for crowd simulation. *Transactions on edutainment VI* (2011), 99–110. 5
- [HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Physical review E* 51, 5 (1995), 4282. 3
- [Hu14] HU N.: *Spatio-temporal patterns and pedestrian simulation*. PhD thesis, Nanyang Technological University, Singapore, 2014. 12
- [JCP*10] JU E., CHOI M. G., PARK M., LEE J., LEE K. H., TAKAHASHI S.: Morphable crowds. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 140. 3
- [JHS07] JOHANSSON A., HELBING D., SHUKLA P. K.: Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems* 10, supp02 (2007), 271–288. 1, 3
- [Kah73] KAHNEMAN D.: *Attention and effort*. Citeseer, 1973. 4
- [KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM transactions on graphics (TOG)* 21, 3 (2002), 473–482. 3
- [KHvBO09] KARAMOUZAS I., HEIL P., VAN BEEK P., OVERMARS M. H.: A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*. Springer, 2009, pp. 41–52. 12
- [KS02] KIRCHNER A., SCHADSCHNEIDER A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications* 312, 1 (2002), 260–276. 3
- [KSMK05] KLÜPFEL H., SCHRECKENBERG M., MEYER-KÖNIG T.: Models for crowd movement and egress simulation. In *Traffic and Granular Flow* 3. Springer, 2005, pp. 357–372. 1
- [LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association, pp. 109–118. 1, 2, 3
- [LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Computer Graphics Forum* 26, 3 (2007), 655–664. 1, 3
- [LCRP*05] LUKE S., CIOFFI-REVILLA C., PANAIT L., SULLIVAN K., BALAN G.: Mason: A multiagent simulation environment. *Simulation* 81, 7 (2005), 517–527. 10
- [LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum* 23, 3 (2004), 509–518. 2
- [LMM03] LOSCOS C., MARCHAL D., MEYER A.: Intuitive crowd behavior in dense urban environments using local laws. In *Theory and Practice of Computer Graphics, 2003. Proceedings* (2003), IEEE, pp. 122–129. 1
- [LSTA10] LUBER M., STORK J. A., TIPALDI G. D., ARRAS K. O.: People tracking with human motion predictions from social forces. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (2010), IEEE, pp. 464–469. 1
- [LZC*08] LUO L., ZHOU S., CAI W., LOW M. Y. H., TIAN F., WANG Y., XIAO X., CHEN D.: Agent-based human behavior modeling for crowd simulation. *Computer Animation and Virtual Worlds* 19, 3–4 (2008), 271–281. 1
- [MHG*09] MOUSSAÏD M., HELBING D., GARNIER S., JOHANSSON A., COMBE M., THERAULAZ G.: Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society B: Biological Sciences* 276, 1668 (2009), 2755–2762. 1, 3
- [Mor78] MORÉ J. J.: The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*. Springer, 1978, pp. 105–116. 8
- [MPG*10] MOUSSAÏD M., PEROZO N., GARNIER S., HELBING D., THERAULAZ G.: The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one* 5, 4 (2010), e10047. 3
- [MRS*08] MANNING C. D., RAGHAVAN P., SCHÜTZE H., ET AL.: *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008. 11
- [MT97] MUSSE S. R., THALMANN D.: *A model of human crowd behavior: Group inter-relationship and collision detection analysis*. Springer, 1997. 1, 2

- [PA*11] PEACOCK R. D., AVERILL J. D., ET AL.: *Pedestrian and evacuation dynamics*. Springer Science & Business Media, 2011. 1
- [PB06] PELECHANO N., BADLER N. I.: Modeling crowd and trained leader behavior during building evacuation. *Computer Graphics and Applications, IEEE* 26, 6 (2006), 80–86. 3
- [PESVG09] PELLEGRINI S., ESS A., SCHINDLER K., VAN GOOL L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 261–268. 1
- [POO*09] PETTRÉ J., ONDŘEJ J., OLIVIER A.-H., CRETUAL A., DONIKIAN S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 189–198. 1, 3
- [PP10] PEJSA T., PANDZIC I. S.: State of the art in example-based motion synthesis for virtual characters in interactive applications. *Computer Graphics Forum* 29, 1 (2010), 202–226. 3
- [PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum* 26, 3 (2007), 665–674. 1, 3
- [PSAB08] PELECHANO N., STOCKER C., ALLBECK J., BADLER N.: Being a part of the crowd: towards validating vr crowds using presence. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 136–142. 1
- [PW12] POPOOLA O. P., WANG K.: Video-based abnormal human behavior recognition—a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 865–878. 3
- [RD06] RYMILL S. J., DODGSON N. A.: *Psychologically-based simulation of human behaviour*. PhD thesis, University of Cambridge, 2006. 11
- [Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics* 21, 4 (1987), 25–34. 2
- [Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Game Developers Conference* (1999), vol. 1999, pp. 763–782. 1, 2
- [Rou87] ROUSSEUW P. J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65. 8
- [SGC04] SUNG M., GLEICHER M., CHENNEY S.: Scalable behaviors for crowd simulation. *Computer Graphics Forum* 23, 3 (2004), 519–528. 1
- [SGV*12] SNAPE J., GUY S. J., VEMBAR D., LAKE A., LIN M. C., MANOCHA D.: Reciprocal collision avoidance and navigation for video games. In *Game Developers Conf., San Francisco* (2012). 1
- [SJCO06] SILVERMAN B. G., JOHNS M., CORNWELL J., O'BRIEN K.: Human behavior models for agents in simulators and games: part i: enabling science with pmfserv. *Presence: Teleoperators and Virtual Environments* 15, 2 (2006), 139–162. 1
- [SKFR09] SINGH S., KAPADIA M., FALOUTSOS P., REINMAN G.: Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds* 20, 5-6 (2009), 533–548. 12
- [ST05] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 19–28. 1, 2
- [ST09] SCOVANNER P., TAPPEN M. F.: Learning pedestrian dynamics from the real world. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 381–388. 1, 3
- [SvdBGM11] SNAPE J., VAN DEN BERG J., GUY S. J., MANOCHA D.: The hybrid reciprocal velocity obstacle. *Robotics, IEEE Transactions on* 27, 4 (2011), 696–706. 3
- [Tu96] TU J. V.: Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology* 49, 11 (1996), 1225–1231. 8
- [VDBGML11] VAN DEN BERG J., GUY S. J., LIN M., MANOCHA D.: Reciprocal n-body collision avoidance. In *Robotics research*. Springer, 2011, pp. 3–19. 1, 2, 3
- [VdBLM08] VAN DEN BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (2008), IEEE, pp. 1928–1935. 3
- [vDBPS*08] VAN DEN BERG J., PATIL S., SEWALL J., MANOCHA D., LIN M.: Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games* (2008), ACM, pp. 139–147. 3
- [Whi14] WHITTLE M. W.: *Gait analysis: an introduction*. Butterworth-Heinemann, 2014. 12
- [WO16] WANG H., O'ÁŽSULLIVAN C.: Globally continuous and non-markovian crowd activity analysis from videos. In *European Conference on Computer Vision* (2016), Springer, pp. 527–544. 3
- [Wol73] WOLFF M.: Notes on the behaviour of pedestrians. *People in places: The sociology of the familiar* (1973), 35–48. 6
- [WOO16] WANG H., ONDŘEJ J., O'SULLIVAN C.: Path patterns: Analyzing and comparing real and simulated crowds. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2016), I3D '16, ACM, pp. 49–57. URL: <http://doi.acm.org.ezlibproxy1.ntu.edu.sg/10.1145/2856400.2856410>, doi:10.1145/2856400.2856410. 3
- [YBO11] YAMAGUCHI K., BERG A. C., ORTIZ L. E., BERG T. L.: Who are you with and where are you going? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 1345–1352. 1
- [YWL*17] YI S., WANG X., LU C., JIA J., LI H.: l_0 regularized stationary-time estimation for crowd analysis. *IEEE transactions on pattern analysis and machine intelligence* 39, 5 (2017), 981–994. 3
- [ZCLY15] ZHONG J., CAI W., LUO L., YIN H.: Learning behavior patterns from video: a data-driven framework for agent-based crowd modeling. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), International Foundation for Autonomous Agents and Multiagent Systems, pp. 801–809. 10
- [ZCT15] ZHAO M., CAI W., TURNER S. J.: Evaluation of crowd models in low density scenarios using real-world crowd data. In *Proceedings of the 2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications* (2015), IEEE Computer Society. 2
- [Zha12] ZHANG J.: *Pedestrian fundamental diagrams: Comparative analysis of experiments in different geometries*. PhD thesis, Forschungszentrum Jülich, 2012. 10
- [ZTC13] ZHAO M., TURNER S. J., CAI W.: A data-driven crowd simulation model based on clustering and classification. In *Proceedings of the 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications* (2013), IEEE Computer Society, pp. 125–134. 2