International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **1**, February **2018**

ICIC International ©2018 ISSN 1349-4198
pp. **127–145**

COLLISION AVOIDANCE APPROACH FOR EXAMPLE-BASED CROWD SIMULATION

Weiwei Xing, Lili Zhu, Xiang Wei and Peng Bao

School of Software
Beijing Jiaotong University
No. 3, Shangyuancun, Haidian District, Beijing 100044, P. R. China
{ wwxing; 14121686; 14112081; baopeng }@bjtu.edu.cn

Received June 2017; revised October 2017

Abstract. *Collision avoidance plays an essential role in crowd simulation and determines the simulation quality to some extent. We propose a novel method to improve the collision avoidance performance based on the existing crowd simulation methods. The example-based method is chosen as the basic method by reasons of the realistic simulation result. Considering collision avoidance in the example-based method relies heavily on data captured from the real world, velocity obstacle (VO) is introduced to calculate an action of an agent when collision-free example cannot be found. We finally optimize the performance of collision avoidance by performing the novel collision fixing (CF) algorithm. The CF takes advantage of obtained actions to detect potential collisions and employs repulsive force to avoid the collisions. Experiments are conducted to evaluate the performance of the proposed method, and the results show that the method greatly reduces collision times, especially in complex simulation scenarios, while keeping the scenarios otherwise realistic.*

Keywords: Agent-based modeling, Crowd simulation, Collision avoidance, Example-based simulation

1. **Introduction.** Crowd simulation serves many fields, such as social phenomenon study, simulated training, and multimedia-related applications, including virtual reality, video games, the movie industry, etc. Agent-based simulation models are widely applied since they model people as agents that can perform a number of different behaviors. Collision avoidance is the essential behavior of people in most simulation scenarios. Rendering plausible collision-avoidance behavior has recently been the focus of many researchers' efforts.

In the field of agent-based simulation, simulation methods are generally divided into rule-based methods and data-driven methods. For rule-based approaches, researchers simulate collision avoidance by defining various rules according to the joint effect of physical, psychological, and sociological factors [1-7]. As subtle factors have impacts on humans, it is extremely difficult to describe all the aspects of human behaviors. Therefore, although efficient collision avoidance can be achieved, realistic and natural motions are difficult to simulate. To improve upon the realism of simulations, real-life motion data captured have been increasingly applied in [8-12]. Consequently, data-driven techniques appear in extensive application and the example-based method belongs to one type of the data-driven methods. In the example-based methods, agents can directly learn actions encoded in the real-life data without the need to define rules.

Although the example-based methods can automatically capture the realism of human motion, these methods are currently subject to the challenge of effectively avoiding collisions when the motion data are not sufficient. Due to the rigorous conditions and the

demand of accurate trajectory-tracking techniques, capturing enough motion trajectories that cover diverse simulation scenarios is extremely difficult. Agents might collide when the motion data do not cover enough of the various states of the agents. Thus, it is important to rigorously solve the collision avoidance problem in the example-based simulation to maintain realistic crowd simulation results.

In this paper, we propose an effective collision avoidance approach to decrease the possibility of collision for the example-based crowd simulation and maintain its natural behaviors among crowd simultaneously. In the simulation system, we firstly try to calculate actions for agents with the example-based method. To deal with the problem of potential collision that arises when no available example can be found, the velocity obstacle (VO) based rule [13] is implemented to calculate the safe action for the current agent. Theoretically, VO model can take care of a collision very efficiently; however, we find that when introduced in the example-based model, VO could not eliminate all of the collisions if its parameters are inappropriate. Finally, to further avert the collision, a novel collision fixing (CF) algorithm is proposed to improve upon the performance of collision avoidance. The CF algorithm makes good use of actions which have been calculated and uses the repulsive force to simulate collision-free actions.

The main contributions of our work in this paper are as follows.

(1) In the field of example-based simulation, we introduce the VO rule to calculate actions to be taken when no collision-free example can be found. The introduced rule could remedy the shortage of examples and eliminate the collisions to some extent.

(2) Different from the strategy of predicting collisions, our work proposes CF algorithm to detect potential collisions and employs the repulsive force to avoid the collisions effectively.

(3) Drastically decrease collision times without compromising the realism of the original example-based simulation methods and make real data be applied to more simulation scenarios.

The remainder of this paper is organized as follows. Firstly, related works are reviewed and compared with our approach in Section 2. Section 3 introduces the general mechanism of our method. In Section 4, the example extraction and the example database construction are then presented. Section 5 expounds the complete simulation process, especially the collision avoidance strategy. Next, we conduct experiments to evaluate the collision avoidance and the realistic performance of the proposed method in Section 6. Finally, Section 7 concludes our research and looks into future works.

2. **Related Works.**

Vast works of research have been dedicated to crowd simulation and recent surveys [14,15] provide outstanding overviews. Among existing models, agent-based simulation, which considers each simulated person as an agent, becomes more and more popular considering that it can simulate crowds with various characteristics and diverse behaviors. The recently developed data-driven methods belong to the agent-based model. The data-driven methods make use of collision predictions and avoidance maneuvers implicitly encoded in data, which represent real-world human trajectories.

One group of works aims to design and train energy functions or artificial models whose parameters are calibrated by data in order to simulate crowd behavior that match the behavior observed in video sequences [8,16]. Therefore, the parameters are determined automatically. Although automatically calibrating parameters by means of data could outperform manual calibration, the performance of collision avoidance is tied to the design of the models. Several works focus on learning crowd dynamics in the provided video sequences. Collective behaviors in crowded scenes were learned from video sequences through a hybrid model of dynamic pedestrian-agents in [9]. Some newly proposed works

[17-22] learned macro-motion patterns of specific scenes in videos and used non-data-driven models, such as the social force model and reciprocal velocity obstacles (RVO) to achieve collision avoidance. Macro-dynamics of a crowd in these works can be consistent with real trajectories, but the reality of localized behavior might not be guaranteed.

Another widely investigated branch is example-based simulation. In example-based approaches, examples are defined as state-action pairs and extracted from human trajectories. Some research, similar to the first group of work mentioned earlier, trains learning algorithms through examples. For instance, locally weighted linear regression or other suitable algorithms were introduced so that the actions of agents could be predicted with algorithms [10,23,24]. Although these works avoid manually obtaining parameters, synthesizing realistic simulations might be restricted to the formulation of models or algorithms. Specifically, M. Zhao et al. used the action in the example as the preferred velocity of a collision avoidance model [25], whereas the actual collision avoidance relies on the realistic performance of the applied model.

To reduce artificial intervention, some popular methods, which are closely relevant to our work, make agents act directly as suitable examples so as to make actions more closely mirror the real world. In the general process, firstly, an example database is constructed in advance. Then during the simulation, a set of similar examples are obtained and collision-free actions are selected from these examples. In [11], A. Lerner et al. predefined a function to match similar examples with agents. However, when no collision-free examples were found in the matched examples, one example that could steer the agent away from a collision was searched, though not matching the surrounding configuration. To improve the matching efficiency, M. Zhao et al. clustered examples and selected examples by means of an artificial neural-network classifier [26]. As for collision avoidance under the case of no collision-free examples, an emergency stop was performed. Another time-reduction strategy took advantage of the data structure of the graph in [27], wherein the perception-action graph (PAG) was presented to store similar patterns on nodes and actions on edges. Moreover, the work conducted a greedy mechanism: if neighbors' actions had been updated, it would use the updated actions to predict collisions, and it otherwise used their current velocities. However, when no nodes could avoid collisions, agents would compulsorily select other nodes or ignore the collision. In summary, the efforts for collision avoidance in the existing works were less arduous and collisions could not be eliminated efficiently.

We notice that when the real data are not sufficient to describe the simulation scenario, collision-free examples are difficult to find. Under this circumstance, collisions are likely to occur and lead to less realistic simulation results. As far as we know, existing works based on the method of learning behaviors directly from examples mainly focused on speeding up the search process. However, the situation where suitable examples could not be found was rarely given importance. Compared with previous works, we elaborate on the advantage of example-based methods and devote more effort to addressing the problem of collision avoidance.

3. **The Framework of Collision Avoidance.**

The framework of the proposed collision avoidance approach is shown in Figure 1. Firstly, we design the example formulas and construct example databases before the simulation. Then, we perform the example selection, calculate actions, and perform the collision avoidance process during the simulation. The detailed explanation is as follows.

(1) Before simulation, examples are classified into two categories according to the existence of other individuals or obstacles within the field of view of the current individual.

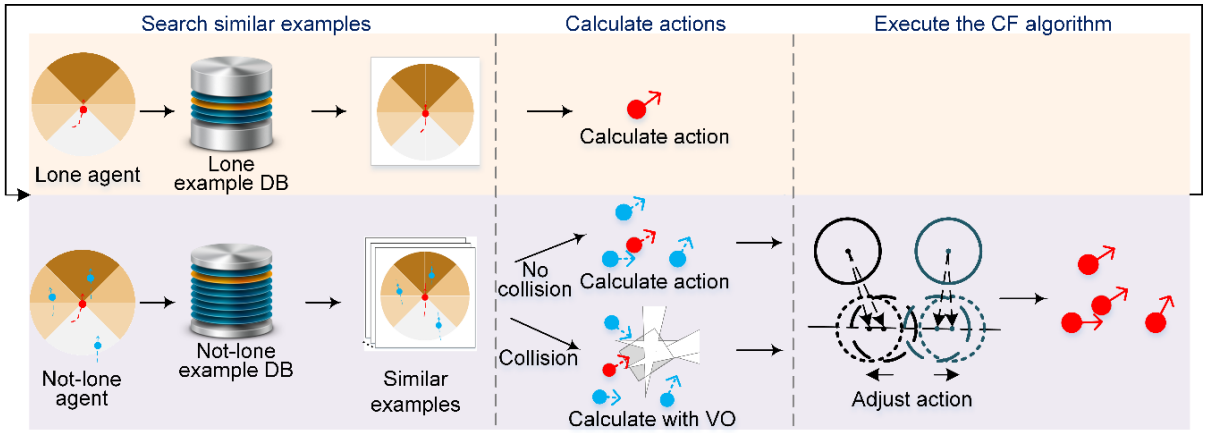

FIGURE 1. Framework overview of the proposed collision avoidance approach

We design example formulas by considering the psychological and behavioral characteristics of human. Thus, examples are extracted from real trajectories of human beings according to the formulas, and the lone example database (DB) and the not-lone example database are constructed.

(2) During simulation, the agent is classified as a lone agent or not-lone agent depending on the state. Then, similarity matching across the states in the corresponding DB is performed to find similar examples. The classification strategy could save data space and improve searching efficiency.

(3) For the lone agent, a random-selection strategy is employed among the top similar examples to obtain its action, which will make behaviors look natural. For the not-lone agent, the collision-free action is selected from the most similar examples. For the case where no safe examples exist, the VO-based rule is introduced to calculate one action.

(4) After all of the not-lone agents obtain the actions, a novel CF algorithm motivated by the repulsive force is executed to detect collisions among not-lone agents and to adjust the actions to avoid potential collisions effectively.

4. **Example Definition and Database Construction.** Before the simulation, we extract examples from real crowd trajectories, which are represented by the 2D position coordinates of individuals and the corresponding time. Considering the necessity of executing collision avoidance, we classify examples into two categories, i.e., the *lone example* for individuals without other individuals or obstacles within the field of view (FOV) during a short period of time, and the *not-lone example*. FOV is a circle centered on each tracked individual or agent with radius r. For lone individuals, it is unnecessary to consider the possibility of collision while it is necessary for not-lone ones.

We take the psychological and behavioral characteristics of human into consideration and define each example as $\langle \boldsymbol{state}, \boldsymbol{action} \rangle$, in which $\boldsymbol{state}$ contains environmental and individual states and $\boldsymbol{action}$ is the next motion direction. For the *lone example*, the state is represented as $\boldsymbol{state}_{\boldsymbol{ex},\mathbf{0}} \in \mathbb{R}^2$:

$$\boldsymbol{state}_{\boldsymbol{ex},\mathbf{0}} = \langle \bar{v}, \alpha \rangle \tag{1}$$

where $\bar{v}$ is the specific individual (subject)'s average speed during a short period of time, and α is the angle between the direction of current motion and the preferred direction of the subject.

For the *not-lone example*, the state is denoted as $\boldsymbol{state}_{\boldsymbol{ex},\mathbf{1}} \in \mathbb{R}^{M\times N+2}$:

$$\boldsymbol{state}_{\boldsymbol{ex},\mathbf{1}} = \langle s_{1,1}, s_{1,2}, \ldots, s_{1,N}, s_{2,1}, \ldots, s_{2,N}, \ldots, s_{M,1}, \ldots, s_{M,N}, \bar{v}, \alpha \rangle \tag{2}$$

where M represents the current frame of human motion sequence, and N is the number of sectors divided in the FOV. $s_{m,n}$ describes the relative position relationship between the current subject and surrounding individuals or obstacles (neighbors) within the FOV, which is defined as:

$$s_{m,n} = \begin{cases} \left(1 - \frac{d_{n,i}^2}{r^2}\right) * e^{-\frac{|\alpha_{n,i}|}{180}}, & 0 < d_{n,i} < r \\ 0, & d_{n,i} \geq r \end{cases} \tag{3}$$

where $m = 1, \ldots, M$, $n = 1, \ldots, N$, and i identifies the neighbors in the sector n. $d_{n,i}$ is the distance between the subject and neighbor i. $\alpha_{n,i}$ is the angle between the subject's current direction of movement and the direction pointing to the neighbor i from the subject.

In Equation (3), $d_{n,i}$ and $\alpha_{n,i}$ are determined by:

$$\langle d_{n,i}, \alpha_{n,i} \rangle = \arg\min_{\alpha_{n,j}} \left(\min_{d_{n,j}} \{ \langle d_{n,j}, |\alpha_{n,j}| \rangle \} \right) \tag{4}$$

where $j = 1, \ldots, J$ and J is the number of neighbors in the current sector n. Figure 2 illustrates the variables in Equations (3) and (4).

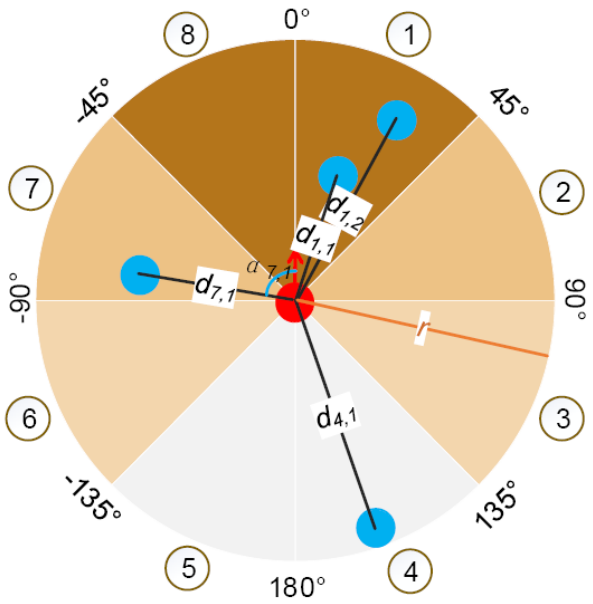

FIGURE 2. The situation for the not-lone example, in which the circle represents the FOV, which is centered on the subject

With the absolute value of the angle $\alpha_{n,i}$ increasing, the value of $e^{-\frac{|\alpha_{n,i}|}{180}}$ decreases and thus leads to $s_{m,n}$ decreasing. Equation (3) is formulated based on the fact that human beings tend to pay more attention to the areas ahead and nearby neighbors in the real world.

action $\in \mathbb{R}^2$ is defined as a vector $\langle x, y \rangle$, which indicates the direction of motion in the next step.

Based on the real trajectories and equations above, we can calculate and extract examples from real trajectories. Next, we classify the examples into two databases, i.e., lone example DB and not-lone example DB for searching efficiently. A KD-tree is built as a DB correlated with the lone example. As for the not-lone examples, the principal components analysis (PCA) is employed to map the dimension of $\boldsymbol{state_{ex,1}}$ into a lower-dimension space for eliminating dispensable information and searching similar states more

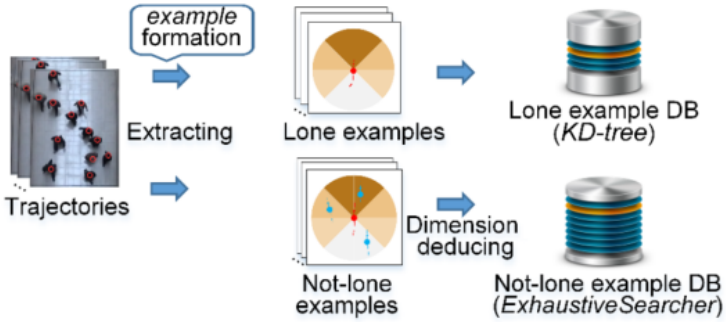

FIGURE 3. Example DB construction

efficiently. As the KD-tree tends to search more slowly with higher dimensions (generally higher than 10), we build the ExhaustiveSearcher object using Matlab R2014b as the DB correlated with the not-lone example in this paper. The process for the example DB construction is briefly illustrated in Figure 3. Classifying examples will save data space and improve the searching performance compared with unifying the example formula and stored in one database.

5. **Collision Avoidance Mechanism.** Constructing example databases is a precondition for example-based simulation. In this section, the approach for selecting examples, calculating actions, and collision avoidance for not-lone agents during a simulation will be introduced.

5.1. **Example selection strategy.** Agents are classified as two categories: lone agent and not-lone agent. For the lone agents, we calculate the Euclidean distance as the similarity between the lone agent and *lone example* according to their states. Then, we pick one out of the top ten similar examples randomly for each agent. The random strategy enables lone agents to avoid going straight and show slight variation in action, so that they can step to the goals naturally.

For the not-lone agents, the cosine distance between states of the agent and the not-lone example is computed as the similarity. As cosine distance values are naturally normalized, further inter-feature normalization is not necessary, although retrieval results based on Euclidean distance and cosine distance are similar in high dimensional data spaces [28]. Then, we search top k similar examples by the K-nearest neighbors (KNN) algorithm. Thus, each not-lone agent can attain k similar examples.

5.2. **Action calculation by VO.** With selected examples, actions are calculated for agents. For the lone agents, we can obtain the action by multiplying the ***action*** in the selected example by $\bar{v}$ in Formula (1) without considering the problem of collision. We will focus on the action calculation for the not-lone agents in the following.

Each agent is abstracted as a circle with radius R, and collision is deemed to happen when the distance between any two centers is smaller than $2R$ as shown in Figure 4. In order to predict a collision, we make use of the actions of neighbors who are sorted before the subject, and assume neighbors who are sorted after the subject select the actions of the most similar examples.

In the ideal condition, agents could always get collision-free actions using examples. However, because trajectory data cannot always fully describe the simulation scenarios, the problem of not having any existing safe action in similar examples remains. To deal with this problem, we employ the velocity obstacle (VO) based rule to calculate at least one safe velocity. Briefly speaking, VO is a set of colliding velocities with neighbors. In the

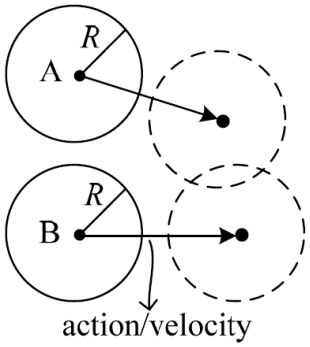

FIGURE 4. The situation of collision between two agents

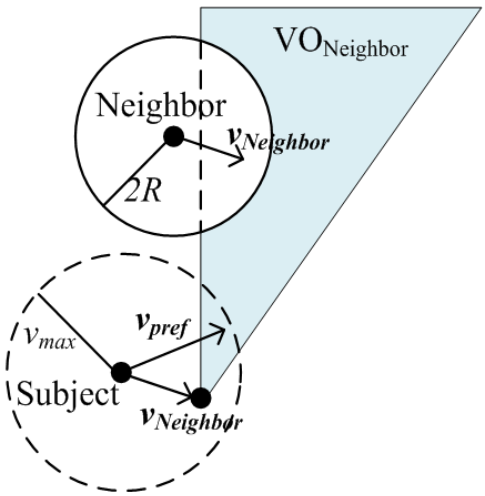

FIGURE 5. Method of calculating feasible velocity by VO

multi-agent simulation, each agent should calculate velocities outside the combined VO on the basis of the velocities of its neighbors. Moreover, velocities should be restricted to lie within an admissible range. In our work as illustrated in Figure 5, we restrict velocities to lie within the admissible set $V(\boldsymbol{v})$, which is defined as formation (5):

$$V(\boldsymbol{v}) = \{\boldsymbol{v} | \|\boldsymbol{v}\| < v_{\max}\} \tag{5}$$

where $v_{\max}$ is the maximum agent velocity.

Ideally, velocities should be chosen outside the combined VO and inside $V(\boldsymbol{v})$. However, in our work, velocities inside the VO are allowed, considering the possibility that the combined VO occupies the entire set of $V(\boldsymbol{v})$. Motivated by J. V. D. Berg et al. [29], the penalty function (6) is introduced to deal with this issue:

$$penalty(\boldsymbol{v}) = \frac{1}{tc(\boldsymbol{v})} + w\|\boldsymbol{v_{pref}} - \boldsymbol{v}\| \tag{6}$$

where $tc(\boldsymbol{v})$ is the time before collision with a given neighbor under the velocity $\boldsymbol{v}$, $\boldsymbol{v_{pref}}$ is the preferred velocity whose magnitude is the current speed of the subject, and the direction is pointing to the goal. w is a factor controlling the contribution of the difference between $\boldsymbol{v_{pref}}$ and $\boldsymbol{v}$ to the penalty. The penalty function is designed to measure the fitness of a velocity. Specifically, when the time to collision is longer and velocity is closer to the preferred velocity of the agent, the penalty is smaller, i.e., the velocity is more suitable.

For efficient calculation, we sample a number N of velocities (denoted as $\{\boldsymbol{v}'\}$) inside $V(\boldsymbol{v})$, and select the velocity $\boldsymbol{v_{next}}$ with the minimal penalty as the action for the agent:

$$\boldsymbol{v_{next}} = \arg \min_{\boldsymbol{v}' \in \{\boldsymbol{v}'\}} penalty(\boldsymbol{v}') \tag{7}$$

5.3. **Collision-fixing algorithm for avoiding potential collisions.** According to the VO-based rule, we calculate actions for the agents who cannot find available examples. The VO model can efficiently avoid collisions in theory. Unfortunately, in our framework, VO's efficiency is somewhat impacted if the parameter N in Section 5.2 is not suitably assigned. Besides this, it is also possible that the time to the collision in taking the attained velocity $\boldsymbol{v_{next}}$ is less than one frame. Under this circumstance, the velocity (namely action) may lead to a collision. In order to avoid a possible collision, a novel CF algorithm is proposed in this paper.

With the current position and calculated action, the expected position of the agent in the next frame can be calculated, and thereby we can locate potential collisions. Based on these conditions, CF checks the collision that is definitely going to occur and recalculates actions. The algorithm principle is motivated by the social force model in which repulsive effects are utilized to avoid an object or individual [30]. In CF algorithm, the expected positions of agents that are going to collide in the next frame will be moved along the direction of the largest repulsive force. In this way, the colliding agents are pushed away to the safe positions which are taken as the new expected positions.

Supposing three agents $\langle A, B, C\rangle$ will collide in the next step as shown in Figure 6(a). Using the CF algorithm, we adjust expected positions pairwise, i.e., $\langle A, B\rangle$ and $\langle A, C\rangle$. In

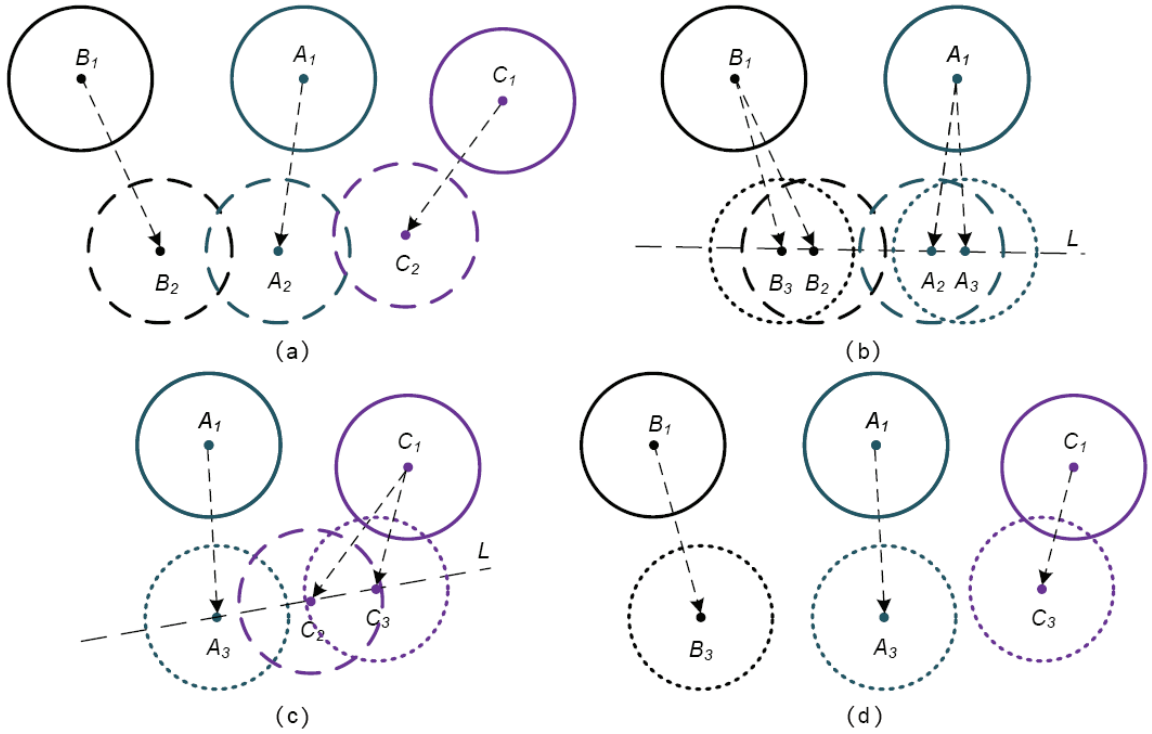

FIGURE 6. Demonstration of the CF algorithm for fixing collisions. Circles represent agents A, B and C; A_n, B_n and C_n represent centers of agents, where $n = 1$ indicates the current position, $n = 2$ for the expected position, and $n = 3$ for the new expected position. (a) shows the current and expected positions of agents. (b) and (c) illustrate position adjustment of A, B and C. (d) presents new expected positions with the repulsion effect.

Figure 6(b), the direction of the largest repulsive force between A and B on the positions A_2 and B_2 lies on the line (namely L) determined by A_2 and B_2. Detailed calculation should follow the three rules below.

(1) The new expected positions A_3 and B_3 for agent A and B should lie on the line L according to the largest repulsion effect.

(2) The center distance $|A_3B_3|$ as shown in Figure 6(b) should not be smaller than the sum of $A.R$ and $B.R$, which are the radii of agents A and B respectively.

$$|A_3B_3| \geq A.R + B.R \tag{8}$$

In Formula (8), $|A_3B_3|$ represents the final distance between the two agents after the repulsive force works. The distance is defined by experiments on the underlying simulation scenario.

(3) The agent mass is roughly regarded as equal, so the agents are repulsed by the same force magnitude. Thus, the adjustment distance $|A_2A_3|$ for agent A and $|B_2B_3|$ for agent B should be equal as shown in Equation (9).

$$|A_2A_3| = |B_2B_3| \tag{9}$$

Based on the above rules and the positions of A_1, A_2, B_1, B_2, the new expected positions A_3, B_3 can be calculated. Taking agent A for example, the calculating process is presented below. Key points and vectors are shown in Figure 7.

Prerequisites:

(1) The positions of $A_1, A_2, B_2 \in \mathbb{R}^2$

(2) The distance between A_3 and B_3, $|A_3B_3|$

(3) $|A_2A_3| = |B_2B_3|$

Derivation:

(1) $|A_2A_3| = (|A_3B_3| - |A_2B_2|)/2$

(2) $\overrightarrow{A_2A_3} = \frac{\overrightarrow{B_2A_2}}{|B_2A_2|} \times |A_2A_3|$

(3) $A_3 = A_2 + \overrightarrow{A_2A_3}$, $\overrightarrow{A_1A_3} = A_3 - A_1$

Now we focus on the agent pair of $\langle A, C\rangle$ as shown in Figure 6(c). As agent A has been adjusted, we only need to adjust the expected position of agent C to avoid a collision between A and C. In this case, repulsive force merely has an effect on agent C. Therefore, the first two rules remain the same but the adjustment distance in the rule (3) is replaced with Equation (10):

$$|C_2C_3| = |A_3C_3| - |A_3C_2| \tag{10}$$

The result after employing CF is shown in Figure 6(d). Obviously, A, B and C will not collide with each other in the next step after adjustment; however, collisions with

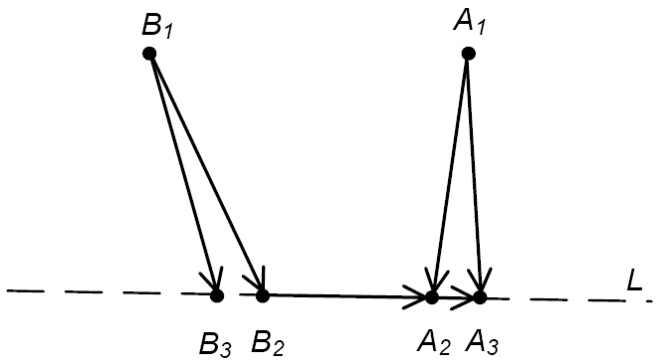

FIGURE 7. Positions and actions of agents A and B

other agents may happen, since the other agents have obtained their next actions during the time when the three agents have changed their own. To address this problem, the center distance (the magnitude of repulsive force) in the second rule will be dynamically reduced until collisions decrease. If the center distance is reduced to the sum of the two agents' R and the number of collisions has not decreased, the current agent pair will not be adjusted.

The pseudocode of CF is presented as Algorithm 1. In the algorithm, p ($p \geq 1$) is a parameter to determine the center distance (in line 7). In the real world, people prefer to preserve their personal space and maintain a distance from others, so we first try large repulsive force, then the smaller one (in line 16).

Algorithm 1 CF

```
Input: agents – set of not alone agents; p – parameter for position adjustment
1   collisionAgents, collisionPairs ← calculateCollision(agents.expectedPostn);
2   collisionAgents.label ← 0;
3   for each ⟨A, B⟩ in collisionPairs
4     if A.label + B.label < 2
5       leftAgents ← getLeftAgents(agents, A, B);
6       do
7           distance = p*(A.R + B.R);
8           postnA, postnB ← adjustPosition(A, B, distance);
9           newCollisionNum = collisionNum(postnA, postnB, leftAgents);
10          oldCollisionNum = collisionNum(A.expectedPostn, B.expectedPostn,
                                           leftAgents);
11          if newCollisionNum < oldCollisionNum
12            updateNewPosition(A, B, postnA, postnB);
13            updateLabel(A, B);
14            break;
15          end if
16          shrink(p);
17      while p ≥ 1
18    end if
19  end for
```

Different from methods of predicting collision, the CF algorithm is proposed to fix knowable collisions in the simulation system. Due to working only when there exist potential collisions, CF is less invasive to the example-based method. If CF changes the actions, it will introduce two advantages. Firstly, collisions will reduce in the next frame. As elucidated above, the new action is calculated based on the obtained action and the slight change of the obtained action is guaranteed by the parameter p, so the trajectories of agents will be continuous. Secondly, the following actions are easier to be obtained from the example DB when there is no collision.

6. Experiments and Results.

In this section, we first introduce the experimental setup and demonstrate the impact of data on the state similarity. Then we examine our method and compare it with state-of-the-art methods from the aspects of collision avoidance and real-world performance.

6.1. Dataset and experimental setup.

Our experiments are conducted on a PC with Intel(R) Core(TM) i7 CPU of 3.40 GHz and the dataset is from the Crowd Data of SNU

Movement Research Lab[1]. We selected the trajectories of behavior *oneway* and *aggressive* to build our example DB. People walk in normal speeds from the same starting side to the opposite in a rectangle region in the *oneway* model and walk faster in that of the *aggressive*. Table 1 shows the number of trajectories and extracted examples. We constructed the *not-lone example DB* on the basis of the 2833 not-lone examples and constructed the *lone example DB* based on the 1280 lone examples.

TABLE 1. Trajectories dataset and extracted examples

Behaviors	**Trajectories**	**Examples**	**Not-lone examples**	**Lone examples**
oneway	67	2648	1687	961
aggressive	46	1465	1146	319
total	113	4113	2833	1280

According to the real-world trajectories in Table 1, we constructed a rectangle with 310×200 unit length (l) as the range of the simulation environment, and agents are represented as circles with radius 7.5 l in the 2D plane. We define the unit of velocity as l/f, in which f represents frame. The initial speeds $\{v\}$ of agents are generated by Gaussian distribution, $v \sim \mathcal{N}(5.4, 1.8^2)$ where a mean of 5.4 is estimated from the real-world trajectory data and the standard deviation is set to 1.8.

In the real-world crowd data of Table 1, there are nearly 6 individuals per frame walking in the scenario. In order to study the proposed collision-avoidance approach in this paper, we designed six different simulation scenarios for our experiments by setting different entrance positions, frequency, and motion destinations. Each simulation lasts for 100 frames. Thus, we acquired six scenarios with different densities and walking behavior as introduced in Table 2 and illustrated in Figure 8.

TABLE 2. Simulation scenarios

Scenario	*7 a/f, Oneway*	*19 a/f, Oneway*	*27 a/f, Oneway*	*35 a/f, Oneway*	*17 a/f, Twoway*	*15 a/f, Crossway*
Density (agent/frame)	7	19	27	35	17	15
Behavior	one-way	one-way	one-way	one-way	two-way	cross-way

6.2. **Dataset impact on state similarity.** In our work, we evaluated the impact of a dataset on the example-based crowd simulation, especially the impact of state similarity between not-lone agents and the corresponding examples.

To obtain different databases for evaluation, we constructed five *not-lone example DBs* out of the 2833 not-lone examples in Table 1. We extracted about 500, 1000, 1500, 2000, and 2500 examples respectively from the 2833 not-lone examples. In the experiments, the 2833 not-lone examples were classified into 100 clusters by K-Means, based on the cosine distance, and the corresponding extracted proportions in the clusters were 17.65% ($\approx 500/2833$), 35.30%, 52.95%, 70.60%, and 88.25%. According to the five proportions, we extracted the corresponding examples from 100 clusters and constructed five *not-lone example DBs* named *DB 500*, *DB 1000*, *DB 1500*, *DB 2000*, and *DB 2500*, where the number denotes the extracted example amount. Additionally, the original *lone example DB* was utilized for lone-agents in all of the experiments. The simulation method for this experiment is such that when collision-free examples cannot be found, the agent will

[1]http://mrl.snu.ac.kr/

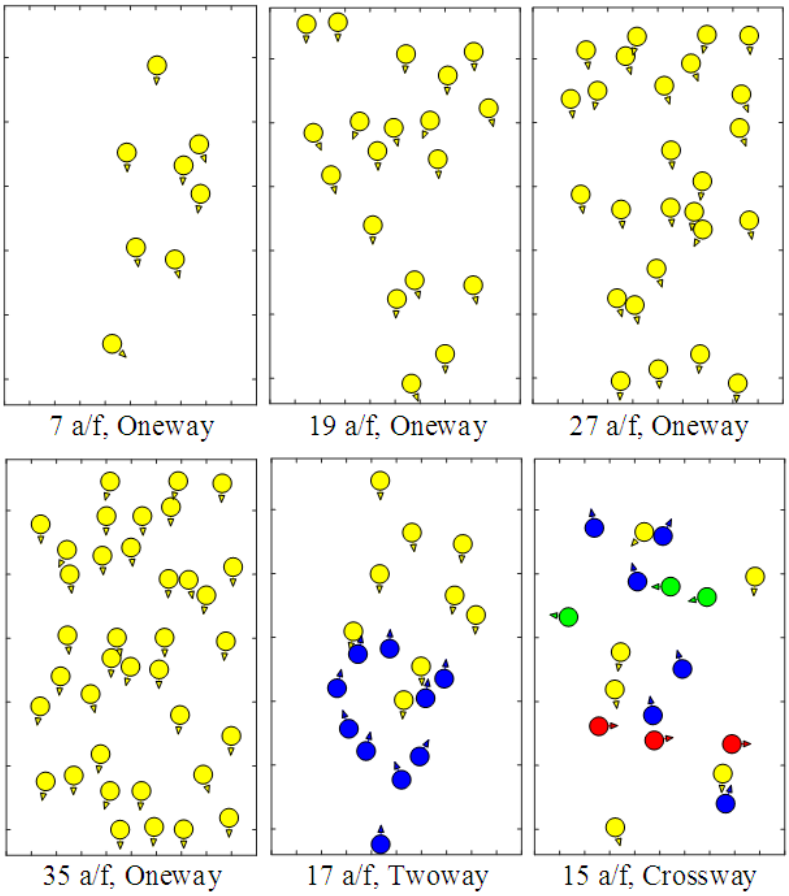

FIGURE 8. Screenshots of the six simulation scenarios

act with the most similar example's action without introducing either the VO or CF algorithms.

During simulation, the top N similarities for each not-lone agent in every frame, as one record row, were stored in one table. After simulation, a matrix D_{MN} could be attained, where $D = (d_{ij})$, and M represents the total number of records. Then we calculated the average similarity sim based on the column D: $sim_j = \frac{\sum_{i=1}^{M} d_{ij}}{M}$, where $j = 1, \ldots, N$.

We firstly tested the six scenarios on *DB 2500* and next tested the scenario *7 a/f, Oneway* on the five DBs. In Figure 9, the average similarities for the top 100 most similar examples are shown. In Figures 9(a) and 9(b), the horizontal axis represents the rank of the most similar examples as j, and the vertical axis represents the corresponding average similarity sim_j. As Figure 9(a) shows, similarity decreases as the scenario increasingly differs from the scenario of the real-world data. On the other side, in Figure 9(b), where the amount of data decreases, the similarity also decreases. The phenomenon can be explained as both the complexity of the scenario and the example amount in the real-world data will impact the state similarity between the agent and the example. In other words, the real-world data might impact the quality of simulation results. With more sufficient real-world data, the result of example-based simulation tends to be of higher quality. We further demonstrated this assertion in the latter experiments in the case of collision avoidance.

6.3. **Collision avoidance and simulation results.** To demonstrate the effectiveness of the proposed method, experiments were conducted to test the performance of collision avoidance and the level of realism.

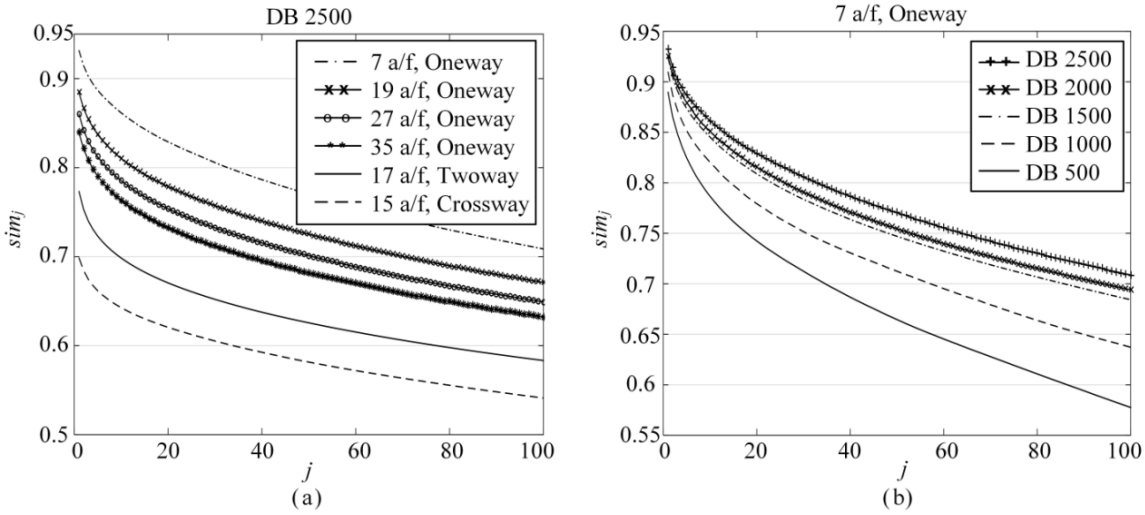

FIGURE 9. Similarity curves: (a) the similarity curves for each simulation scenario performed on DB 2500; (b) the similarity curves for scenario 7 a/f, Oneway performed on the five DB

6.3.1. *Collision avoidance.* Firstly, we tested the effectiveness of the proposed CF algorithm. The time of collision was recorded both before and after applying the proposed CF algorithm during the simulation in each frame. After performing several rounds of simulation, average times of collision per frame, denoted as t_f, were calculated. Since there were no collisions before or after executing the CF algorithm in the scenario *7 a/f, Oneway* and *19 a/f, Oneway*, results of the other four scenarios are shown in Figure 10. In scenario *27 a/f, Oneway* as shown in Figure 10(a), the CF algorithm eliminates potential collisions in the frames from 20 to 30 and 82; in scenario *35 a/f, Oneway* as shown in Figure 10(b), the CF algorithm eliminates most of the potential collisions in the frames from 26 to 65; in scenario *17 a/f, Twoway* and *15 a/f, Crossway* as shown in Figure 10(c) and Figure 10(d) respectively, the CF algorithm could obviously reduce collisions. Because CF algorithm checks potential collisions and fixes the dangerous actions, collisions will be avoided. When the parameter p decreases to the minimum and collision times still do not reduce, CF algorithm will not recalculate the actions. Hence, there exist cases where collision remains the same, e.g., frames from 60 to 70 in Figure 10(a). We can conclude that the proposed CF algorithm can reduce collisions and produce a remarkable effect especially in complex scenarios.

We then performed a series of experiments to compare the collision avoidance performance of our simulation method named *OURS* with *PAG* [27]. In order to prevent interference, we implemented PAG with our example definition. In the experiment, we use the times of the examples that could not be found (t_n) and the times of collision (t_c) during simulation to evaluate the simulation result. The averages of t_n and t_c are calculated and shown in Table 3. Besides collision times, times of example not found t_n also indicate the realistic results of different simulation methods to some extent. The smaller t_n indicates the simulation is more realistic, because more safe actions are found in the examples. From Table 3, it can be seen that collisions seldom occur even in the complex scenarios *17 a/f, Twoway* and *15 a/f, Crossway* with our method. Since the pure example-based simulation method relies on the real-world data completely, collisions are easy to occur when the data could not fully describe the simulation scenario. However, with the introduced rule and further collision avoidance algorithm, our method could avoid most collisions so as to simulate more simulation scenarios. The simulation

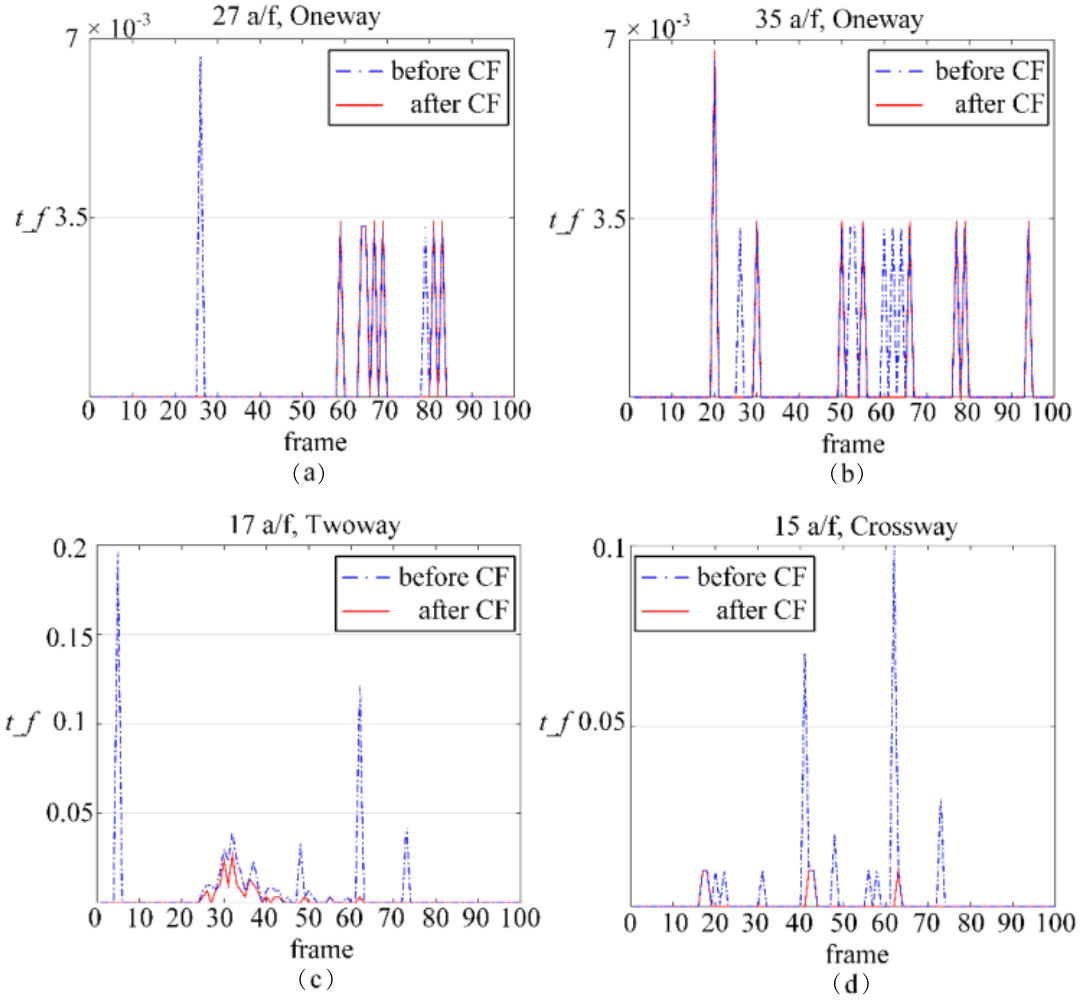

FIGURE 10. Comparison of average collision times before and after executing the CF algorithm

TABLE 3. Times of example not found and collision comparison of PAG and OURS

method / scenario	PAG		OURS	
	t_n	t_c	t_n	t_c
7 a/f, Oneway	0.10	0.00	0.10	**0.00**
19 a/f, Oneway	0.47	0.05	0.29	**0.00**
27 a/f, Oneway	1.56	0.18	1.08	**0.02**
35 a/f, Oneway	3.74	0.47	2.44	**0.03**
17 a/f, Twoway	121.93	56.53	78.36	**0.16**
15 a/f, Crossway	112.37	55.08	58.38	**0.05**

result demonstrates that our method is extremely effective when there are no collision-free examples during a simulation.

6.3.2. *Realistic performance.* In addition to collision avoidance, we conducted experiments to evaluate realistic performance with respect to trajectory, energy cost and velocity by comparing OURS with PAG and the velocity-based method optimal reciprocal collision avoidance (ORCA) [31]. The results are as follows.

Trajectory. We randomly chose 20 trajectories from the real-world data in Table 1 and the simulated trajectories from scenario *7 a/f, Oneway*, respectively, which are shown in Figure 11. In ORCA, if one agent would not collide with others with the preferred action, it does not change the current action. Thereby, straight trajectories were usually preferred by ORCA as shown in Figure 11(d). Because people tend not to perambulate in perfectly straight lines in the real world, ORCA performs less realistically. On the other hand, in Figures 11(a), 11(b) and 11(c), trajectories simulated by OURS are similar to PAG and real-world trajectories.

To quantitatively demonstrate that the simulated trajectories with OURS can guarantee the natural simulation, we calculate discrete Fréchet distance [32] to measure the trajectory similarity. According to the start points and the time entering the scenario in the real-world trajectories as shown in Figure 11(a), we ran the simulation with PAG, OURS and ORCA respectively. The average discrete Fréchet distance was calculated and shown in Table 4.

Since a shorter Fréchet distance indicates a higher curve similarity, the trajectories simulated by OURS and PAG are more similar to each other and more similar to the real-world trajectories compared with those of ORCA. Because the actions of agents in the example-based method are mainly obtained from the real-world data, the simulated trajectories tend to be similar to the real-world ones. The conclusion can be drawn that our method maintains a level of realism on par with the example-based simulation method.

Energy Cost. It is well known that in the real world, an obstructed path usually consumes more bodily energy than does an unobstructed path. In a simulation, agents

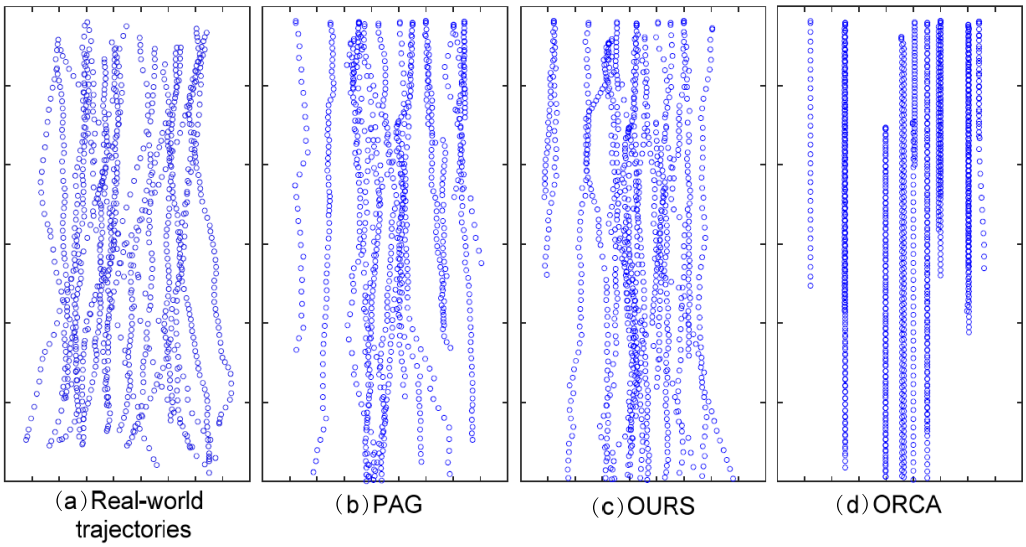

FIGURE 11. Trajectories sampled from the real-world data and the simulated trajectories of the scenario 7 a/f, Oneway using the methods of the PAG, OURS, and ORCA

TABLE 4. The average discrete Fréchet distance between the real-world trajectory and simulated trajectory

Distance	Real-world trajectories	PAG	OURS
PAG	**59.91**	0	36.89
OURS	**60.00**	**36.89**	0
ORCA	**75.78**	**68.04**	**68.23**

need to change their velocities to avoid collision more frequently in the complex scenarios, which results in an extra energy cost. Therefore, the energy cost can be taken as the measurement of the real-world results. To measure energy cost characteristic of the simulation, we calculated the energy cost per unit time with Formula (11) [33]:

$$E_w = 2.23 + 1.26v^2 \tag{11}$$

where E_w is in watts per kilogram body mass and v is the speed in m/s. In our experiment, we customized v in l/f, set body mass m as 50 kg and thereby produced the following energy cost formula for each piece of motion trajectory:

$$E_p = m \sum_{i=1}^{F} (2.23 + 1.26|v_i|^2) \tag{12}$$

where F is the number of frames and $i = 1, \ldots, F$.

The straight line from the starting point to the goal of trajectories is viewed as the desired motion path. The straight-line distance denoted as l_p represents the path length. Thus, we could get $\langle l_p, E_p \rangle$ pairs for each track segment. Then, the average E_p of the same l_p was calculated and it represented the general energy cost for a given path. To assess the realism of a simulation, we compared the energy cost of PAG, OURS, and ORCA in scenario *7 a/f, Oneway* with the real-world data, and the results are shown in Figure 12. As in nature, the longer the path is, the more energy costs. When the paths are shorter than 40, the energy cost is similar among the real-world data and simulated trajectories. As the length increases, the energy cost of PAG and OURS stays close to that of the real-world data, while ORCA takes less energy cost and diverges from the others. This result can be explained by the example-based method presenting flexible motion in real life so that more energy is consumed while the rule-based method remains rigid and thereby costs less energy. On the other hand, it can be demonstrated that our method introduces rules, yet maintains the kinetic characteristics of real people.

Velocity. Finally, we compared the mean of the velocity and the standard deviation of the simulated trajectories among PAG, OURS, and ORCA as shown in Table 5. It is obvious that both the mean of the velocity and the standard deviation of OURS are close to those of PAG in all of the scenarios. Notably, the standard deviation of the velocity

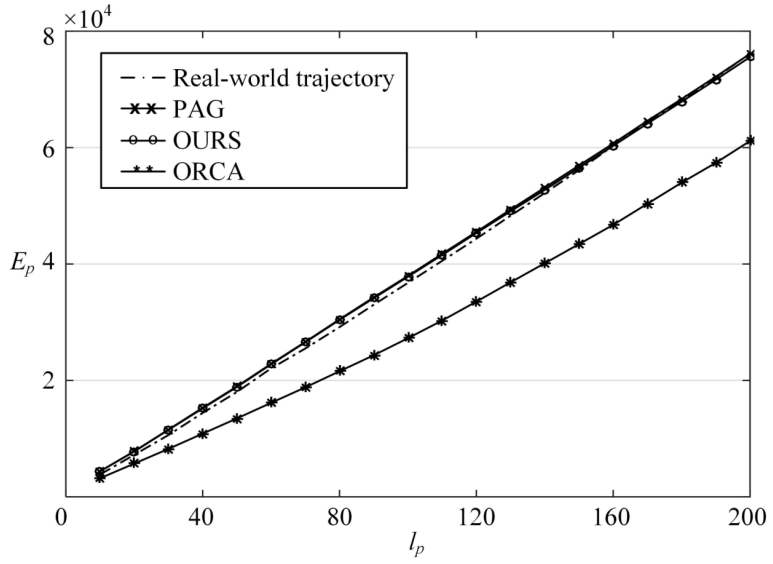

FIGURE 12. Energy cost for the path of real-world data and trajectories for scenario 7 a/f, Oneway using the PAG, OURS, and ORCA methods

TABLE 5. Comparison of the velocity mean and standard deviation of simulated trajectories with PAG, OURS, and ORCA in the six scenarios

method / scenario	mean			standard deviation		
	PAG	OURS	ORCA	PAG	OURS	ORCA
7 a/f, Oneway	5.32	**5.27**	5.40	0.92	**0.91**	0.10
19 a/f, Oneway	5.39	**5.40**	5.22	1.13	**1.12**	0.36
27 a/f, Oneway	5.51	**5.50**	5.18	1.23	**1.23**	0.45
35 a/f, Oneway	5.49	**5.48**	5.17	1.21	**1.21**	0.50
17 a/f, Twoway	5.65	**5.62**	5.26	1.39	**1.43**	0.53
15 a/f, Crossway	5.89	**5.89**	5.34	1.38	**1.46**	0.45

for the real trajectories in Table 1 is 1.16. Compared with ORCA, the standard deviation of the velocity for OURS and PAG is larger than ORCA and is closer to that of the real-world trajectories. It is again proven that our method can maintain similar velocity characteristics with the real-world trajectories. Due to less invasion to the example-based method, our method can maintain similar velocity characteristics with the real-world trajectories.

7. **Conclusions and Future Works.** In this paper, we proposed a novel and effective method to avoid collisions in the example-based crowd simulation method. First, we constructed a *lone example DB* and *not-lone example DB*. Actions of agents are to be mainly calculated from the selected examples. For lone agents, examples are selected randomly out of a collection of similar examples. For not-lone agents, similar examples whose actions are collision-free with their neighbors will be chosen. If collision-free examples could not be found, the VO-based rule is then implemented to calculate actions for the agents. Moreover, to further improve the effectiveness of collision avoidance, we proposed the CF algorithm to check and avoid potential collisions. A series of experiments were conducted to verify the collision avoidance and realistic performance for the method. From the experimental results, we can conclude that the proposed method performs well in avoiding collisions especially in complex scenarios, and meanwhile maintains the realistic results of example-based crowd simulation.

Since crowd simulation is subtle in various respects, there are many avenues for further research. In future work, we will focus on the following three points: examining our method with more complex scenes and performance in structural simulation scenarios, applying our method to improving collision avoidance where a crowd exhibits group behavior, and improving the calculated performance of the method.

Acknowledgment. This work is supported in part by National Natural Science Foundation of China (Nos. 61100143, 61272353, 61370128), Program for New Century Excellent Talents in University (NCET-13-0659), Beijing Higher Education Young Elite Teacher Project (YETP0583), and Fundamental Research Funds for the Central Universities (2014JBZ004, 2015RC031).

REFERENCES

[1] C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, *Proc. of the ACM SIGGRAPH Computer Graphics*, vol.21, no.4, pp.25-34, 1987.

[2] C. W. Reynolds, Steering behaviors for autonomous characters, *Proc. of the Game Developers Conference*, pp.763-782, 1999.

[3] S. J. Rymill and N. A. Dodgson, *Psychologically-Based Simulation of Human Behavior*, University of Cambridge, 2006.

[4] I. Karamouzas and M. H. Overmars, Simulating human collision avoidance using a velocity-based approach, *The Workshop on Virtual Reality Interactions & Physical Simulations*, pp.125-134, 2010.
[5] S. B. Liu, S. M. Lo, K. L. Tsui and W. L. Wang, Modeling movement direction choice and collision avoidance in agent-based model for pedestrian flow, *Journal of Transportation Engineering*, vol.141, no.6, 2015.
[6] S. Bandini, S. Manzoni and G. Vizzari, Situated cellular agents: A model to simulate crowding dynamics, *IEICE Trans. Information and Systems*, vol.87, no.3, pp.669-676, 2004.
[7] T. Kobayashi, S. Takahashi, M. Kunigami, A. Yoshikawa and T. Terano, Harnessing organizational deviation and Kaizen activities through agent-based modeling, *Proc. of the TRAFST Conference, Transdisciplinary Federation of Science and Technology*, p.72, 2011.
[8] P. Scovanner and M. F. Tappen, Learning pedestrian dynamics from the real world, *Proc. of the 12th International Conference on Computer Vision (ICCV)*, vol.9, pp.381-388, 2009.
[9] B. Zhou, X. Tang and X. Wang, Learning collective crowd behaviors with dynamic pedestrian-agents, *International Journal of Computer Vision*, vol.111, no.1, pp.50-68, 2015.
[10] K. H. Lee, M. G. Choi, Q. Hong and J. Lee, Group behavior from video: A data-driven approach to crowd simulation, *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA, USA, pp.109-118, 2007.
[11] A. Lerner, Y. Chrysanthou and D. Lischinski, Crowds by example, *Proc. of Computer Graphics Forum*, vol.26, no.3, pp.655-664, 2007.
[12] M. Zhao, S. J. Turner and W. Cai, A data-driven crowd simulation model based on clustering and classification, *Proc. of the 2013 IEEE/ACM the 17th International Symposium on Distributed Simulation and Real Time Applications*, pp.125-134, 2013.
[13] P. Fiorini and Z. Shiller, Motion planning in dynamic environments using velocity obstacles, *The International Journal of Robotics Research*, vol.17, no.7, pp.760-772, 1998.
[14] 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. Modeling and Computer Simulation (TOMACS)*, vol.20, no.4, p.20, 2010.
[15] D. C. Duives, W. Daamen and S. P. Hoogendoorn, State-of-the-art crowd motion simulation models, *Transportation Research Part C: Emerging Technologies*, vol.37, pp.193-209, 2013.
[16] J. Zhong, N. Hu, W. Cai, M. Lees and L. Luo, Density-based evolutionary framework for crowd model calibration, *Journal of Computational Science*, vol.6, pp.11-22, 2015.
[17] S. Patil, J. V. D. Berg, S. Curtis, M. C. Lin and D. Manocha, Directing crowd simulations using navigation fields, *IEEE Trans. Visualization and Computer Graphics*, vol.17, no.2, pp.244-254, 2011.
[18] S. Kim, A. Bera, A. Best, R. Chabra and D. Manocha, Interactive and adaptive data-driven crowd simulation, *Virtual Reality (VR)*, pp.29-38, 2016.
[19] A. Bera, S. Kim and D. Manocha, Online parameter learning for data-driven crowd simulation and content generation, *Computers & Graphics*, vol.55, pp.68-79, 2016.
[20] J. Zhong, W. Cai, L. Luo and M. Zhao, Learning behavior patterns from video for agent-based crowd modeling and simulation, *Autonomous Agents and Multi-Agent Systems*, vol.30, no.5, pp.1-30, 2016.
[21] J. Zhong, W. Cai, M. Lees and L. Luo, Automatic model construction for the behaviour of human crowds, *Applied Soft Computing*, 2017.
[22] W. Lu, X. Wei, W. Xing and W. Liu, Trajectory-based motion pattern analysis of crowds, *Neurocomputing*, vol.247, pp.213-223, 2017.
[23] P. Torrens, X. Li and W. A. Griffin, Building agent-based walking models by machine-learning on diverse databases of space-time trajectory samples, *Transactions in GIS*, vol.15, no.supplement s1, pp.67-94, 2011.
[24] C. D. Boatright, M. Kapadia, J. M. Shapira and N. I. Badler, Context-sensitive data-driven crowd simulation, *Proc. of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pp.51-56, 2013.
[25] M. Zhao, J. Zhong and W. Cai, A role-dependent data-driven approach for high density crowd behavior modeling, *ACM Conference*, pp.89-97, 2016.
[26] M. Zhao, W. Cai and S. J. Turner, CLUST: Simulating realistic crowd behaviour by mining pattern from crowd videos: CLUST, *Computer Graphics Forum*, vol.8, 2017.
[27] P. Charalambous and Y. Chrysanthou, The PAG crowd: A graph based approach for efficient data-driven crowd simulation, *Proc. of the Computer Graphics Forum*, vol.33, no.8, pp.95-108, 2014.
[28] G. Qian, S. Sural, Y. Gu and S. Pramanik, Similarity between Euclidean and cosine angle distance for nearest neighbor queries, *ACM Symposium on Applied Computing*, pp.1232-1237, 2004.

[29] J. V. D. Berg, M. Lin and D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, *Proc. of the IEEE International Conference on Robotics and Automation*, pp.1928-1935, 2008.

[30] D. Helbing, A mathematical model for the behavior of pedestrians, *Behavioral Science*, vol.36, no.36, pp.298-310, 1998.

[31] J. V. D. Berg, S. J. Guy, M. Lin and D. Manocha, Reciprocal n-body collision avoidance, *Robotics Research*, pp.3-19, 2011.

[32] T. Eiter and H. Mannila, Computing discrete Fréchet distance, *See Also*, vol.64, no.3, pp.636-637, 1994.

[33] M. W. Whittle, *Gait Analysis: An Introduction*, 4th Edition, Heidi Harrison, 2007.