Modeling Gap Seeking Behaviors for Agent-based Crowd Simulation

Linbo Luo
School of Cyber Engineering
Xidian University
Xi'an, China
lbluo@xidian.edu.cn

Cheng Chai
School of Cyber Engineering
Xidian University
Xi'an, China
cchaixd@yahoo.com

Suiping Zhou
School of Science and Technology
Middlesex University
London, United Kingdom
s.zhou@mdx.ac.uk

Jianfeng Ma
School of Cyber Engineering
Xidian University
Xi'an, China
jfma@mail.xidian.edu.cn

ABSTRACT

Research on agent-based crowd simulation has gained tremendous momentum in recent years due to the increase of computing power. One key issue in this research area is to develop various behavioral models to capture the microscopic behaviors of individuals (i.e., agents) in a crowd. In this paper, we propose a novel behavior model for modeling the gap seeking behavior which can be frequently observed in real world scenarios where an individual in a crowd proactively seek for gaps in the crowd flow so as to minimize potential collision with other people. We propose a two-level modeling framework and introduce a gap seeking behavior model as a proactive conflict minimization maneuver at global navigation level. The model is integrated with the reactive collision avoidance model at local steering level. We evaluate our model by simulating a real world scenario. The results show that our model can generate more realistic crowd behaviors compared to the classical social-force model in the given scenario.

Keywords

agent-based modeling, collision avoidance, crowd simulation, gap seeking behavior

1. INTRODUCTION

Human crowd is an intriguing social phenomena in nature. To simulate crowd dynamics, different modeling paradigms have been proposed including flow-based approach, entity-based approach and agent-based approach [23]. Among different paradigms, agent-based crowd modeling has emerged as the most popular and powerful one, due to its capability to model heterogeneous individual behaviors. It is also flexible to incorporate various behavioral factors in agent-based model, which is helpful to improve the realism of simulation. In recent years, many agent-based crowd models have been proposed for a wide range of applications in digital game design, military training, crowd evacuation planning, etc., [17, 19, 15, 20, 22].

The agent-based crowd modeling is a bottom-up approach that models microscopic-level behaviors of individuals. By simulating the individual's interactions with other agents and environment at such level, it is expected that global crowd patterns are emerged which can match the crowd patterns in real-life situations. Therefore, it is essential for agent-based crowd model to realistically and comprehensively capture various microscopic behaviors of individuals in a crowd. While the existing work has focused on the modeling of many microscopic behaviors such as path planning, collision avoidance, group movement, etc. [17, 10, 21, 12, 18], little work has been done to the modeling of the behavior as shown in Fig. 1, which we refer to as *gap seeking* behavior. This behavior occurs when a gap (i.e., empty space in crowd flow) is formed due to asynchronized movement of individuals in a crowd flow. When an individual observes such gap close to her/his walking path, she/he may make a detour to the gap in order to minimize the effort to avoid potential conflict with other people in the crowd. Such gap seeking behavior is especially prominent in medium to high-density crowds, such as the ones shown in Fig. 1.

Although the purpose of gap seeking behavior (i.e., minimize potential conflicts) is similar to collision avoidance behavior, we consider them as two different types of behaviors. The collision avoidance behavior is to *reactively* respond to the oncoming collisions with nearby agents and obstacles, whereas the gap seeking behavior is to *proactively* redirect an agent's path so as to minimize potential collisions. The gap seeking behavior may cause an individual to change her/his moving direction and/or speed in a more apparent manner, which often leads to the deviation of individual's intended path. Moreover, the individual performing gap seeking behavior may create additional gaps for other people to seek and this may cause new lanes to be formed in crowd flow. Thus, the occurrence of gap seeking behaviors can greatly

(a) Bidirectional crowd flow

(b) Perpendicular crowd flow

Figure 1: Gap seeking behaviors observed in two real world scenarios. Snapshots for scenario (a) and (b) are captured from the crowd experiment videos for the Hermes project [1] and SMDPC project [2] respectively.

affect the global movement patterns of a crowd.

In this paper, we aim to provide a computational model to capture the gap seeking behaviors observed in human crowds and investigate how the incorporation of such behaviors in agent-based crowd simulation can improve the realism of simulation results. To this end, we first propose a modeling framework which depicts an individual's motion planning at two levels - i.e., global navigation and local steering. The gap seeking behavior is incorporated as a high-level navigation behavior in the proposed framework. We further decompose the gap seeking behavior into three steps: gap detection, gap selection and gap seeking. To evaluate the effectiveness of the proposed model, we simulate a real world scenario using our model and the social-force model separately. The simulation results have demonstrated that the proposed modeling framework can generate crowd behaviors that are closer to the real world data.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes our modeling framework and provides details of the modeling steps for gap seeking behavior. In Section 4, we present a case study. Section 5 discusses the conclusion and future work.

2. RELATED WORK

To build an effective agent-based crowd simulation, one key issue is how to develop various behavioral models to capture the microscopic behaviors of individuals in a crowd [9]. The work on behavior modeling for agent-based crowd simulation can be generally classified into two categories: global navigation and local steering.

Global navigation models are concerned with how to generate high-level navigational decisions for agents to select next target location to move to in a given environment. These models are usually in the form of path planning where some classical planning algorithms (e.g., A* algorithm [4]) can be adopted. To support navigation in a complex environment, Shao and Terzopoulos [17] proposed a hierarchal environment model to achieve long-range and short-range path planning with A*. Ozcan and Haciomeroglu [11] proposed a modified A* algorithm that considers the flow directions of other agents. Patil et al. [13] proposed to use guidance fields that can be specified by a user or imported from video data, which are then converted into navigation fields to direct agent's movement.

Local steering models deal with the locomotion of agent with the consideration of agent's interactions with nearby agents and objects. Physically-inspired models proposed by Helbing et al. [5] use the physical and socio-psychological forces to model the interactions between agents. Pelechano et al. [14] combined a rule-based system with the force-based model to model heterogenous agents. Karamouzas et al. [8] presented a local avoidance method based on collision prediction. Kapadia et al. [7] proposed a novel egocentric pedestrian steering framework. They introduced affordance fields in an egocentric manner for local planning and steering. More recently, Reciprocal Velocity Obstacles (RVO) model [21] was introduced to generate collision-free locomotion of agent based on velocity space sampling.

Mid-term planning is a new kind of steering method proposed by Bruneau and Pettré [3]. The method models the navigation process between global navigation and local steering. The n next interactions occurring in time are explored and several navigation strategies within such period are identified. An agent will choose the strategy with the least cost which is defined by agent's energy consumption.

The modeling framework proposed in this paper is a layered architecture that contains both global navigation and local steering. The main difference between our work and previous work is that we introduce a new type of navigation behavior(i.e., gap seeking) which leads agent to temporally switch to a new target location (i.e., gap) in response to a dynamic crowd. The most similar approach to ours is probably the one by Bruneau and Pettré [3]. However, they considered empty tunnel existed in a crowd flow and used mid-term planning to navigate agent through such tunnel. However, in reality, such tunnel may not often be observed because of the arbitrariness and domino effect of human movements. Besides, they did not validate the model with real cases. Our method (i.e., gap seeking) is to proactively navigate towards the small holes in a crowd (i.e., gap) for minimizing potential conflicts. We believe such gap seeking behavior is more commonly observed in real scenarios and our model can generate such behaviors which may lead to simulation closer to real crowd behaviors.

3. MODELING FRAMEWORK

Fig. 2 shows our proposed modeling framework for incorporating gap seeking behavior in agent-based crowd simulation. The framework is composed of two major modules: *global navigation* and *local steering*.

The global navigation module is responsible for producing

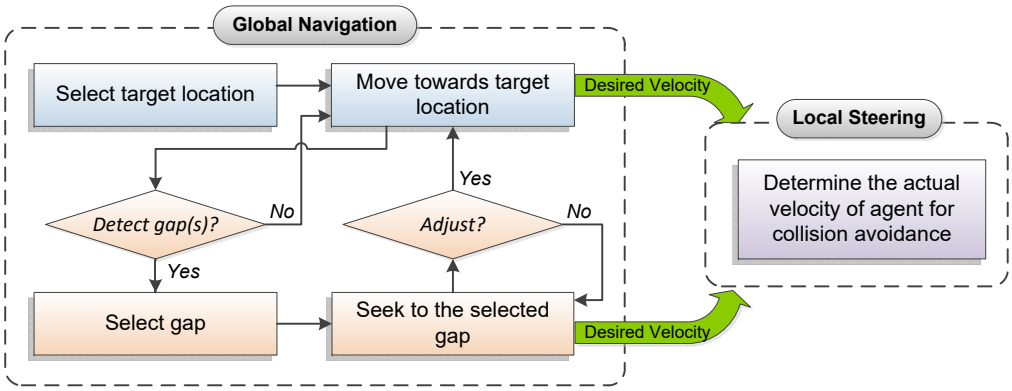

Figure 2: The framework of our model.

the desired velocity of an agent in order to navigate in the environment. In normal situations, the desired velocity is set based on the target location that the agent aims to reach. The target location can be the final destination of the agent or the intermediate way-points of a path that the agent follows. Here, we assume some existing path planning algorithms (e.g., A* algorithm) can be used to produce a series of way-points as target locations. Once the target location is selected, the agent will move towards it (i.e., setting the desired direction towards the target location).

In our model, besides moving towards the target location, an agent will opportunistically look for possible gaps (i.e., empty space in crowd flow) where the agent can detour so as to minimize its effort to avoid potential collisions. Therefore, our model first provides a gap detection mechanism to detect the nearby gap(s) within an agent's detection area. Given that multiple gaps are detected, a gap selection mechanism is then proposed to select the most desirable gap for agent to seek. The agent will then move to the selected gap by setting its desired velocity towards the center of the selected gap. Depending on the crowd dynamics, the agent will decide when to stop the gap seeking behavior and adjust its behavior back to moving towards the target location.

The *global navigation* module will set the desired velocity based on either the target location that the agent aims to reach or the location of selected gap while the agent performing gap seeking behavior. In either case, the desired velocity will be sent to the *local steering* module where the actual velocity will be determined for agents to steer according to the desired velocity while avoiding collisions with other agents and obstacles. In our model, we do not restrain the choice of the collision avoidance algorithms to be used for local steering. In our current implementation, we use the force-based algorithm based on the social-force model. Note that other collision avoidance algorithms (e.g., RVO model) can be used to further enhance the performance of simulation.

The detailed steps of gap seeking behavior modeling are described in the following sub-sections.

3.1 Gap Detection

In the first step, an agent needs to detect available gaps in its surrounding area. In our framework, agents move in a continuous space. However, it is inefficient to represent a gap in such environment representation. Therefore, we introduce a two-layer virtual world representation for gap detection as shown in Fig 3. The bottom layer is a continues space where agents perform local steering behavior. The top layer is a grid map which is used for gap detection. In our current implementation, the grid cell size is set to $0.1m \times 0.1m$ and each agent can occupy multiple grid cells.

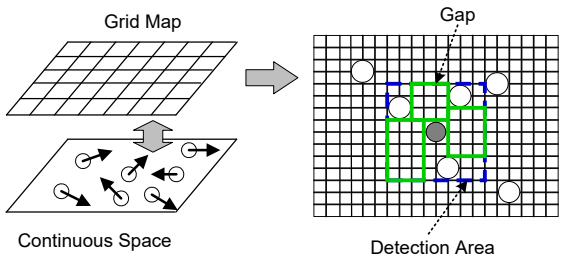

Figure 3: Layered virtual world representation.

In the grid map, we use a rectangle shape to represent a gap (see green rectangles in Fig. 3). To detect the gaps near an agent, we first define an detection area D (dashed blue rectangle in Fig. 3) within which the agent performs the gap detection. The detecting agent (i.e., the grey circle in Fig. 3) is located in the center of the detection area. Here, for ease of computation, we detect all the gaps surrounding the detecting agent without considering the agent's moving direction. In the *gap selection* step described later on, we will filter out the gaps that are out of agent's vision according to its moving direction. Given the detection area D, we specify the occupancy area O as the area within D which is occupied by other agents. The gap detection operation is thus performed within an exploration area E which is defined as:

$$E = D - O \tag{1}$$

In the grid map, the exploration area E contains the grid cells in the detection area D which are not occupied by other agents.

To detect all available gaps within the exploration area E, a *randomized* algorithm is proposed as shown in Algorithm 1. The algorithm works similar to crystal nucleation and a gap is detected by expanding a random seed (like a crystal seed) in a greedy manner by consuming nearby uncrystallized cells. A set of seed cells are first uniformly picked

Algorithm 1 Gap Detection Algorithm

1: $G \leftarrow \emptyset$ ▷ G is a set of detected gaps
2: $P \leftarrow$ UniformlyPick(E) ▷ find a set of growth seeds uniformly in E
3: **while** P is not empty **do**
4: $p \leftarrow$ PopRandom(P) ▷ randomly select a seed from P
5: $x \leftarrow$ NewRectangle(p.x, p.y) ▷ nucleate at p
6: $F \leftarrow$ {UP, DOWN, LEFT, RIGHT} ▷ allowed growth directions
7: **while** F is not empty **do**
8: $x \leftarrow$Expand(x, F, E) ▷ expand the rectangle as much as possible
9: **end while**
10: Insert(G, x) ▷ Add the detected gap to G
11: **end while**
12: RemoveDuplicate(G) ▷ Remove the duplicate gaps
13: **return** G

from the exploration area E (line 2 in Algorithm 1). Then, a seed cell is randomly selected and expands from one of four directions by one cell every time until it reaches the boundaries of area E or the cell occupied by other agent (lines 4-8 in Algorithm 1). The gap is detected through such rectangle expansion and it is added to the set of all detected gaps G in every iteration (line 10 in Algorithm 1). Due to the randomness of our algorithm, the algorithm may yield duplicate gaps from different seed cells. Therefore, the algorithm removes duplicate gaps in the end (line 12 in Algorithm 1).

3.2 Gap Selection

In the second step, an agent needs to select the most desirable gap from all the detected gaps in G. In case that G is empty (i.e., no detected gap), the agent will not proceed to the gap selection step. To select the most desirable gap, we propose three constraints with given priorities. The constraint with a higher priority is examined first and if the detected gap does not satisfy the constraint, the gap is discarded without checking the lower priority constraints. This can help to save the computational cost of our model. The proposed three constraints with the priorities from high to low are described as follows.

Constraint 1: *A gap should be within the vision of the detecting agent.* As mentioned previously, all surrounding gaps are detected in the gap detection step without considering agent's moving direction. In reality, a person's vision is usually limited to a fan-shape region centered along person's moving direction as shown in Fig 4. Thus, we need to filter out the gaps that are out of the vision of the detecting agent.

We check whether a gap is within the agent's vision using the following rules:

$$\|\vec{p_i^k})\| \leq R \tag{2}$$

where $\vec{p_i^k}$ is the vector pointing from agent i's current location to the center of kth detected gap, $\|\vec{p_i^k}\|$ computes the magnitude of $\vec{p_i^k}$, R is the vision radius of an agent.

$$angle(\vec{d_i}, \vec{p_i^k}) \leq \theta \tag{3}$$

where $\vec{d_i}$ is the vector denoting agent i's moving direction, *angle()* computes the angle between two vectors in 2D space and θ is half of the vision angle of an agent. In our model,

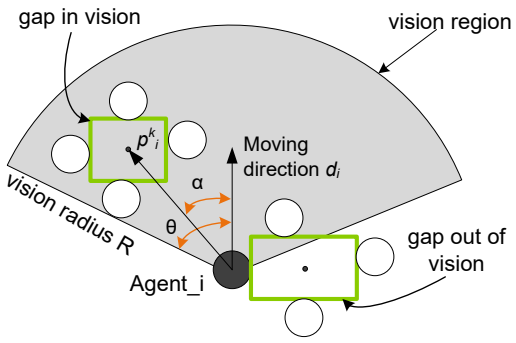

Figure 4: Selecting gap(s) based on agent's vision.

we select gap(s) as within agent's vision only when both equations 2 and 3 are satisfied.

Constraint 2: *A gap should be big enough for the detecting agent to move in.* In our model, if the detected gap is too small, the detecting agent will not consider it as a desirable gap. Since we use rectangle to represent a gap, the following rule is used to determine whether the size of gap is appropriate:

$$\min\{w_i^k, l_i^k\} \geq 2r_i \tag{4}$$

where w_i^k is the width of kth gap detected by agent i, l_i^k is the length of kth gap detected by agent i and r_i is the radius of agent i.

Constraint 3: *A gap should not deviate too much from the detecting agent's intended moving direction towards final destination.* In our model, we assume that an agent will not choose a gap whose direction from agent's current position deviates too much from agent's direction towards its destination (i.e., the final goal agent aims to reach). This is because it may take greater effort for an agent to reach its destination if seeking to such gaps. To avoid such gaps, the following rule is used:

$$angle(\vec{u_i}, \vec{p_i^k}) \leq \varphi \tag{5}$$

where $\vec{u_i}$ is the vector pointing from agent i's current location to its destination, $\vec{p_i^k}$ is the vector pointing from the center of agent i to the center of kth detected gap and φ is a user-defined threshold angle.

Given that all the above three constraints are satisfied, it is still possible that multiple detected gaps are selected. In such cases, the final selection of the most desirable gap is performed according to the following rule:

$$k^* = \arg\min(angle(\vec{u_i}, \vec{p_i^k})) \tag{6}$$

This means we select the k^*th gap whose direction $\vec{u_i}$ has the minimum angle with agent's direction towards its destination. Here, we assume that an agent selects the gap whose direction is most consistent with the direction towards the agent's destination.

3.3 Gap Seeking

In the last step, an agent needs to update the desired velocity $\vec{v_i^d}$ so as to move to the selected gap. In our current implementation, the direction of $\vec{v_i^d}$ is set to be from an agent's current location towards the center of the selected gap. The magnitude of $\vec{v_i^d}$ (i.e., the speed of agent) is set

according to the following rule:

$$\|\vec{v_i^{\mathrm{d}}}\| = \frac{V_{\max}}{1 + \exp[-\beta(s - \alpha s_{\min})]} \quad (7)$$

where $V_{\max}$ is the maximum speed of an agent, s is the area of the selected gap, $s_{\min}$ is the area of the smallest gap we can select (i.e., $s_{\min} = 4 * r_i^2$ in our model), and $\alpha \in (0, 1]$ and $\beta > 0$ are bounding coefficients.

According to equation 7, the speed of the agent increases when the area of the gap increases. Here, we consider a larger gap attracts agent to move at a higher speed. The highest speed an agent can take is $V_{\max}$ and the lowest speed is taken when the area of the gap is smallest (i.e. $s_{\min}$). For example, $\|\vec{v_i^{\mathrm{d}}}\| = V_{\max}/2$ when $s = s_{\min}$ and both α and β are set to 1.

Once $\vec{v_i^{\mathrm{d}}}$ is determined, the agent will move according to the new desired velocity for a period of T^{s} time. T^{s} is calculated as follows:

$$T^{\mathrm{s}} = \frac{\|\vec{p_i^{k*}}\|}{\|\vec{v_i^{\mathrm{d}}}\|} \quad (8)$$

where $\|\vec{p_i^{k*}}\|$ is the magnitude of the vector pointing from agent i's current location to the center of the selected gap (i.e., the distance between the detecting agent and the selected gap) and $\|\vec{v_i^{\mathrm{d}}}\|$ is the desired speed of the agent determined based on equation 7. After T^{s} has elapsed, the agent will adjust its desired velocity according to the target location it aims to reach. In our model, an agent always performs the collision avoidance behavior at local steering level. Thus, it is not necessary that the agent will reach to the center of the selected gap after T^{s}.

4. CASE STUDY

To evaluate the effectiveness of the proposed model, we conduct a case study by applying our proposed model to simulate a real world scenario. For comparison purpose, the classical social-force model [5] is also applied to simulate the same scenario. The simulation outputs generated by both models are compared against the real world data extracted from the scenario video. In this section, we first give the scenario description. Then, the simulation settings are described. Finally, we present our simulation results.

4.1 Scenario Description

The scenario we simulate is a real world scenario where two crowd flows move with an intersecting angle of 90 degree (see Fig. 5). The scenario was conducted by Plaue et al. at Technische Universität Berlin during the Long Night of the Sciences 2010 in Berlin [16].

In this scenario, one group of 54 individuals moves from left to right on a plain ground. Another group of 46 individuals moves from bottom to top starting at 1.81m wide staircase. The region where two crowd flows intersect is about 3m×3m. From the recorded videos of the scenario, we can observe that the gap seeking behaviors occur among the crowd. To initialize and validate our simulation of the scenario, we obtain the individuals' trajectories data from [2]. The trajectory data are captured at 15 fps, that is with a time step of 0.067 seconds.

4.2 Simulation Settings

Figure 5: A sample frame of perpendicular crowd scenario. Snapshot is obtained from [2].

Our agent-based crowd simulation is implemented in Repast Simphony[1] and 3D visualization is realized in Unity3D[2]. As the environment of the simulated scenario is relatively simple, we do not use any path planning algorithm to generate a path for global navigation. The target location of an agent is directly set to its corresponding individual's location at the last frame in the trajectories data. The initial location of an agent is set based on the corresponding individual's first position in the trajectories data and the initial speed of an agent is set to be the average speed of the corresponding individual in the first three recorded frames.

In our current framework, the local steering behavior for avoiding oncoming collisions is implemented using a force-based model (i.e., the repulsive interaction force defined in the social-force model). At the global navigation level, the direction of desired velocity is usually set towards the target location. In the case that an agent detects a gap, the gap seeking model is triggered and the desired velocity is updated accordingly. The parameters related to the gap seeking model are set as follows. The detection area D is 3m×3m. The vision radius and angle of an agent is set to 2.5m and 120 degree respectively. α and β values in equation 7 are set to be 0.5 and 0.75 respectively. $V_{\max}$ of an agent is $1.34m/s$ and the radius of agent is 0.25m. The simulation time step is set to be the same as the frame rate in the trajectories data (i.e., 0.067s).

4.3 Results

To visually observe the individual behaviors in the scenario, we have visualized the real world data and simulation results produced by our model and social-force model in Unity3D. Fig. 6 shows the snapshots of the 3D visualizations at the 250th and the 270th time steps respectively. It can be observed that the red-circled agent in the real world data is performing a gap seeking behavior during this period of time. Our model is able to replicate such gap seeking behavior similar to the real world data. However, the same agent in the social force model ignores the gap and moves directly towards its target location (i.e., a location close to the top right pillar).

To quantitatively evaluate the performance of our proposed model, we adopt the relative distance error metric

[1]http://repast.sourceforge.net/repast_simphony.php
[2]http://unity3d.com/

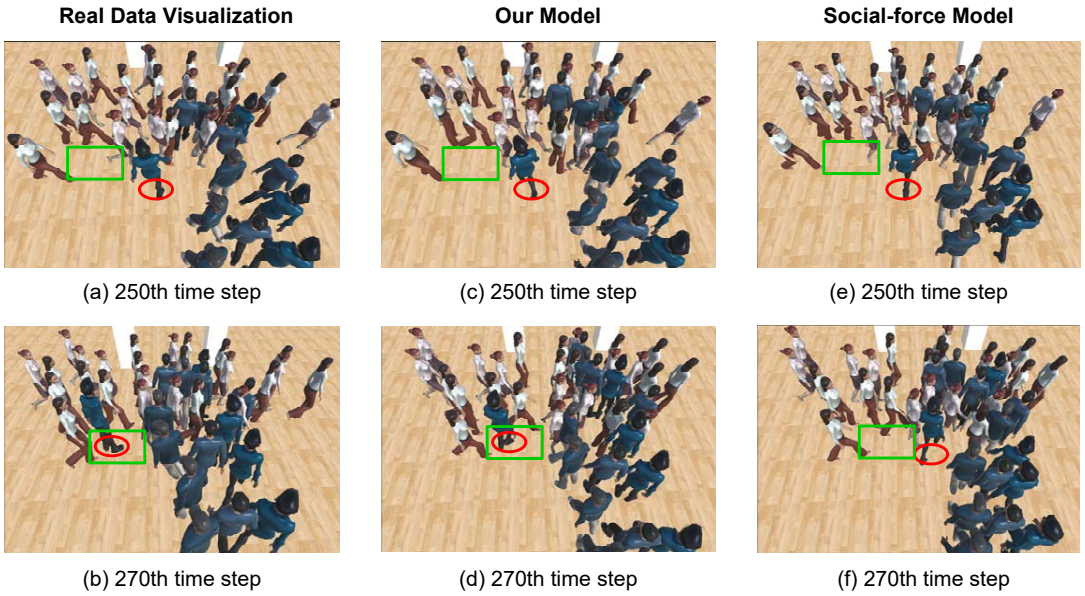

Figure 6: Snapshots of 3D visualizations of real world data and simulation results from our model and social-force model.

proposed in [6], which is calculated as follows:

$$\delta_{\text{err}} = \frac{\|x_i^{\text{sim}}(t+T) - x_i^{\text{real}}(t+T)\|}{\|x_i^{\text{real}}(t+T) - x_i^{\text{real}}(t)\|}) \quad (9)$$

where $x_i^{\text{real}}(t)$ is the position of individual i in real world data at time t, T is a user-defined time period and $x_i^{\text{sim}}(t+T)$ is the position of corresponding agent i after it moves according to the simulation model for a period of T with the simulation starts at time t. δ_{err} is a progressive error metric that measures the absolute difference between the simulated agent and the real world data when the simulation is re-initialized at each starting time t. To obtain the model performance over entire simulation, evaluations according to equation 9 are conducted at several different starting times t (for every 15 time steps in our implementation) and the average of δ_{err} (i.e., $\overline{\delta_{\text{err}}}$) over all the agents and starting times t are obtained.

Table 1: The average of $\overline{\delta_{\text{err}}}$ and its standard deviation (in parentheses) over 50 independent runs

	Social-force	Our model	p-value
$T = 1.675$	$0.43_{(0.018)}$	$0.29_{(0.021)}$	<0.0001
$T = 2.68$	$0.49_{(0.031)}$	$0.25_{(0.034)}$	<0.0005

For model evaluation based on $\overline{\delta_{\text{err}}}$, we first perform the simulations using our model and social-force model[3] for 50 independent runs respectively and each run is set with different random seed. We then calculate the average of $\overline{\delta_{\text{err}}}$ over 50 runs in two cases: T=1.675s (i.e., 25 simulation time steps) and T=2.68s (i.e., 40 simulation time steps). Table 1 shows the results of our evaluation. It can been seen that our proposed model has yielded smaller $\overline{\delta_{\text{err}}}$ value against real world data for both cases, compared to the social-force model. This shows that our model can generate crowd behaviors closer to the real world data.

To test the statistical significance of our results, we also conduct one-tailed two-sample t-test at 0.05 significance level. Our research hypothesis (i.e., the alternative hypothesis in t-test) is that the simulation using our proposed model can generate crowd behaviors which have smaller $\overline{\delta_{\text{err}}}$ value compared to the one using the social-force model. The small p-values from t-tests as shown in Table 1 strongly suggest that our research hypothesis is supported.

5. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a novel behavior model for gap seeking behavior, which is a microscopic behavior that we often observe in real world crowd scenarios. Unlike collision avoidance behavior, we consider gap seeking behavior as a proactive conflict minimization behavior when an individual navigates in a crowd. Thus, a two-level modeling framework is proposed which integrates a gap seeking behavior model at the global navigation level and uses collision avoidance algorithm at the local steering level. Specifically, the gap seeking behavior is modeled with three steps as gap detection, gap selection and gap seeking. Our simulation results have shown that our proposed model can replicate the gap seeking behavior as observed in real world scenario. The model is also able to generate crowd behaviors that are closer to real world data, compared to the classical social-force model.

We will continue to work on the development of our scenario generation framework along several directions. First, the design of the behavior model can be enhanced by considering the moving dynamics of the selected gap in the gap seeking step of the proposed model. The gap detection and selection step of the model may also be improved so as to

[3]Note that before the evaluation, we have calibrated the parameters of social force model using the evolutionary algorithm as described in [6]

generate more reasonable gaps. Second, we will apply the model to other crowd scenarios to evaluate the generality and scalability of our model.

Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant No. 61502370), China 111 Project (No. B16037) and Fundamental Research Funds for the Central Universities (Grant No. JB150305).

6. REFERENCES

[1] Hermes project:. http://www.asim.uni-wuppertal.de/en/research/hermes.html.

[2] Simulation of Multi Destination Pedestrain Crowds (SMDPC) project:. http://www.math.tu-berlin.de/projekte/smdpc/.

[3] J. Bruneau and J. Pettré. Energy-efficient mid-term strategies for collision avoidance in crowd simulation. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 119–127. ACM, 2015.

[4] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[5] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

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

[7] M. Kapadia, S. Singh, W. Hewlett, and P. Faloutsos. Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 215–223. ACM, 2009.

[8] I. Karamouzas, P. Heil, P. van Beek, and M. H. Overmars. A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, pages 41–52. Springer, 2009.

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

[10] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, 5(4):e10047, 2010.

[11] C. Y. Ozcan and M. Haciomeroglu. A path-based multi-agent navigation model. *The Visual Computer*, pages 1–10, 2015.

[12] S. Park, F. Quek, and Y. Cao. Simulating and animating social dynamics: embedding small pedestrian groups in crowds. *Computer Animation and Virtual Worlds*, 24(3-4):155–164, 2013.

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

[14] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108, San Diego, USA, 2007.

[15] N. Pelechano, C. Stocker, J. Allbeck, and N. Badler. 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*, pages 136–142, Estoril, Portugal, 2008.

[16] M. Plaue, M. Chen, G. Bärwolff, and H. Schwandt. Trajectory extraction and density analysis of intersecting pedestrian flows from video recordings. In *Photogrammetric Image Analysis*, pages 285–296. Springer, 2011.

[17] W. Shao and D. Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28, Los Angeles, USA, 2005.

[18] S. Stüvel, N. Magnenat-Thalmann, D. Thalmann, A. Egges, and A. F. Stappen. Hierarchical structures for collision checking between virtual characters. *Computer Animation and Virtual Worlds*, 25(3-4):331–340, 2014.

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

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

[21] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

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

[23] S. Zhou, D. Chen, W. Cai, L. Luo, M. Low, F. Tian, V.-H. Tay, D. Ong, and B. Hamilton. Crowd modeling and simulation technologies. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(4):20, 2010.