Crowd Simulation with Feedback Based on Locomotion State

Masaki Oshita
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
Fukuoka, Japan
oshita@ai.kyutech.ac.jp

Jumpei Harazono
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
Fukuoka, Japan
harazono.jumpei403@mail.kyutech.jp

Kunio Yamamoto
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
Fukuoka, Japan
kunio@ai.kyutech.ac.jp

***Abstract*—Most crowd simulation methods treat each agent as a point and simulate the movements of these points. Although animations of virtual characters can be generated based on the trajectories of points, the generated motions often become unnatural. For example, an agent may change its walking direction suddenly in the middle of a step, even though it is impossible for humans to do so. We propose a crowd simulation method that considers the locomotion state of an agent. When an agent is in the middle of the step, a feedback force is applied to the agent to prevent the agent from suddenly changing the walking direction. However, when an agent is standing or at the beginning of the step, the agent is allowed to change its walking direction freely. We present experimental results and evaluate the effectiveness of the proposed method.**

***Index Terms*—crowd simulation, crowd animation, locomotion**

I. Introduction

Crowd simulation has been used to create computer animations of virtual characters. However, most crowd simulation methods treat each agent as a point and simulate the movements of these points. Although animations of virtual characters can be generated based on the trajectories of points, the generated motions often become unnatural. For example, an agent may change its walking direction suddenly in the middle of a step, even though it is impossible for humans to do so. Consequently, unnatural walking motion can be generated. Although such artifacts can be reduced by post-processing, such as inverse kinematics, the gap between the simulated point and the generated motion becomes an issue. Such gaps may cause collisions between the generated walking motions of agents. Bipeds, including humans, change their walking direction by swinging their legs toward the desired direction. The direction of motion of the swinging leg is determined at the beginning of the step and cannot be changed during this step. The agent should be moved according to this type of locomotion state.

In this study, we propose a crowd simulation method that considers the locomotion state of an agent. Our key idea is to adjust the velocity of the agent so that it is maintained during the step while it is allowed to be changed at the beginning of the step. Although our approach can be applied to any type of crowd simulation method, in this study, we apply it to the social force model [1], [2] which is a common crowd simulation method. The social force model also treats each agent as a point and simulates the movements of the points by considering several forces that work on the agent. We introduce a feedback force based on the locomotion state of the agent. When an agent is in the middle of the step, a feedback force is applied to the agent to prevent the agent from suddenly changing the walking direction. However, when an agent is standing or at the beginning of the step, the agent is allowed to change its walking direction freely. We also introduce a motion generation method based on a feed-forward model. Instead of simply applying a walking motion to the positions of agents computed by crowd simulation, a walking motion is generated in such a way that the position of the walking motion follows the position in crowd simulation. We present experimental results and evaluate the effectiveness of the proposed method.

The remainder of this paper is organized as follows. Section II reviews related work. Section III provides an overview of our proposed method. Sections IV and V describe the baseline methods for crowd simulation and locomotion generation, respectively. Section VI describes the feedback-force model. Section VII presents the experimental results and a discussion. Finally, Section VIII concludes the paper.

II. Related Work

Crowd simulation is an important research topic in the field of computer animation. Many crowd simulation methods have been proposed [1]–[7]. Most previous methods treat each agent as a point and simulate the movements of the points. Although many studies have demonstrated the resulting animations of virtual characters, they simply applied walking motions to the simulated trajectories of agents, and the generated animations were not natural. Some research works have employed more-sophisticated motion-generation methods. Sakuma et. al. [8] used a state machine that contained four states and a dozen motions. Hughes et. al. [6] applied side-stepping motions when they were appropriate, based on the trajectory of the agent. Although these approaches improve the naturalness of motions, the information of motion generation is not used in crowd simulation, and the gap between crowd simulation and motion generation can become an issue.

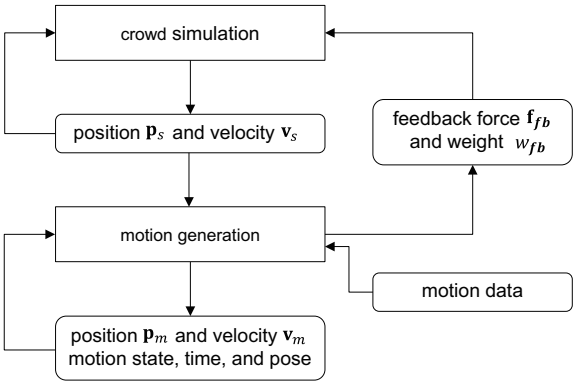


Fig. 1. The system structure and data flow.

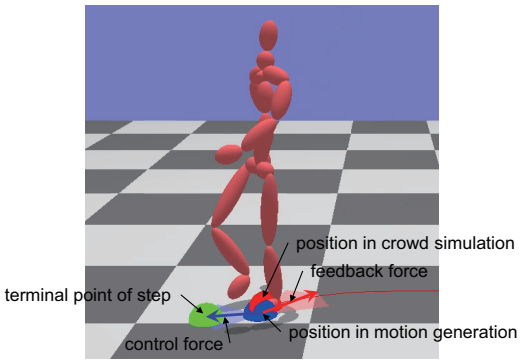


Fig. 2. Visualization of positions and forces in our system. The blue sphere represent the position in crowd simulation $\mathbf{p}_s$. The red sphere represent the position in motion generation $\mathbf{p}_m$. The green sphere represent the terminal position of the current walking cycle $\mathbf{p}_t$. The control force (blue arrow) $\mathbf{f}$ is computed as a sum of driving and collision avoidance forces in crowd simulation. The feedback force (red arrow) $\mathbf{f}_{fb}$ is computed based on the terminal position of current walking cycle so that the moving direction is maintained during the middle of walking cycle.

There are many methods for generating walking motions of virtual characters [9]–[11]. They can be combined with a crowd simulation method to generate animations of the virtual characters. However, as explained in Section I, because the crowd simulation method does not consider the locomotion state, the unnatural movements of agents can be generated.

In this research, we introduce feedback from motion generation to crowd simulation to solve the problem of combining crowd simulation and motion generation.

III. System Overview

Our system consists of crowd simulation and motion generation, as shown in Figure 1. Because our system generates animations of agents on the fly, it can be used for interactive applications, such as computer games and virtual reality.

Crowd simulation and motion generation have their own states for each agent, as shown in Figures 1 and 2. In crowd simulation, the position $\mathbf{p}_s$ and velocity $\mathbf{v}_s$ are updated based on a social force model (see Section IV). In motion generation, position $\mathbf{p}_m$ and velocity $\mathbf{v}_m$ are updated to follow position $\mathbf{p}_s$ and velocity $\mathbf{v}_s$ of the crowd simulation (see Section V). An important feature of our system is the feedback from motion generation to the crowd simulation (see Section VI). The feedback force $\mathbf{f}_{fb}$ and its weight w_{fb} are computed based on the locomotion state and terminal position of the current walking cycle $\mathbf{p}_t$.

Motion generation uses a set of pre-created motion data, including the body model. Although different motion data and body models can be used for individual agents, in our experiments, we applied the same body model and motion data (walking and standing motions) to all the agents.

IV. Crowd Simulation

For crowd simulation, we applied a simple force-based model that extends the social force model [1]. The position $\mathbf{p}_s$ and velocity $\mathbf{v}_s$ of each agent were updated based on the sum of the driving and collision avoidance forces. Each agent has goal position $\mathbf{p}_g$ and maximum speed parameter s_{max}. The velocities of the agents were controlled within the speed limit.

A driving force $\mathbf{f}_d$ that moves the agent towards the goal position $\mathbf{p}_g$ is applied to the agent. First, the desired velocity $\mathbf{v}_d$ for moving towards the goal position $\mathbf{p}_g$ at the maximum speed parameter s_{max} is computed as follows:

$$\mathbf{v}_d = s_{max}\frac{\mathbf{p}_g - \mathbf{p}_s}{|\mathbf{p}_g - \mathbf{p}_s|}. \tag{1}$$

The driving force (acceleration) is then computed to obtain the desired velocity at a certain time T_d according to

$$\mathbf{f}_d = k_d\frac{\mathbf{v}_d - \mathbf{v}_s}{T_d}. \tag{2}$$

In our implementation, $T_d = 1$ s. For the scale parameter k_d, $k_d = 1$ is used.

The collision avoidance force $\mathbf{f}_c$ which is the sum of the repelling forces against nearby agents, is computed as:

$$\mathbf{f}_c = \sum_{i<G} -k_c R\left(\frac{|\mathbf{p}_i - \mathbf{p}_s|}{d_c}\right)\frac{\mathbf{p}_i - \mathbf{p}_s}{|\mathbf{p}_i - \mathbf{p}_s|},\ R(x) = e^{-2x^2}, \tag{3}$$

where d_c and k_c are distance and scale parameters, respectively. $R(x)$ denotes a radial basis function. G is the group of nearby agents within distances d_c and $\mathbf{p}_i$ denotes the positions of nearby agents. We used $d_c = 2.0$ and $k_c = 2.0$. The force to avoid collisions with obstacles and walls $\mathbf{f}_o$ is computed in a similar manner.

Finally, the control force $\mathbf{f}$ applied to the agent is computed as

$$\mathbf{f} = \mathbf{f}_d + \mathbf{f}_c + \mathbf{f}_o. \tag{4}$$

The velocity $\mathbf{v}_s$ and position $\mathbf{p}_s$ of the agent are updated according to the control force $\mathbf{f}$ and time step Δt of the simulation.

$$\mathbf{v}'_s = \mathbf{v}_s + \Delta t\mathbf{f},\quad \mathbf{p}'_s = \mathbf{p}_s + \Delta t\mathbf{v}'_s. \tag{5}$$

With our feedback force model, in addition to the control force $\mathbf{f}$, the feedback force $\mathbf{f}_{fb}$ is used. The details are presented in Section VI.

V. Locomotion Generation

In this section, we introduce a locomotion generation method that uses a cycle of straightforward walking and standing motions. This method uses two states, walking and standing. During walking, the current pose of the walking motion is applied to the position $\mathbf{p}_m$ and orientation o_m of the agent. The orientation of agent o_m is computed based on the changes in position $\mathbf{p}_m$. The current pose is obtained from the walking motion based on the current generic motion time t $(0 < t < 1)$, which is incremented cyclically based on velocity $\mathbf{v}_m$. When each foot is in contact with the ground, inverse kinematics is applied to the leg pose to maintain the foot in the contact position. During standing, the current pose of the agent was obtained from standing motion data.

We introduce a feed-forward model in which the position $\mathbf{p}_m$ and velocity $\mathbf{v}_m$ in motion generation follow the position $\mathbf{p}_s$ and velocity $\mathbf{v}_s$ in crowd simulation. A virtual force for locomotion $\mathbf{f}_m$ is computed such that position $\mathbf{p}_m$ and velocity $\mathbf{v}_m$ satisfy position $\mathbf{p}_s$ and velocity $\mathbf{v}_s$ in a certain time window T_m. The acceleration $\mathbf{a}_v$ that satisfies the velocity $\mathbf{v}_s$ at T_m is computed by:

$$\mathbf{a}_v = \frac{\mathbf{v}_s - \mathbf{v}_m}{T_m}. \tag{6}$$

The acceleration $\mathbf{a}_p$ that satisfies position $\mathbf{p}_s$ at T_m is computed by

$$\mathbf{a}_p = \frac{2}{T_m}(\mathbf{p}_s - \mathbf{p}_m - T_m\mathbf{v}_m). \tag{7}$$

The locomotion force (acceleration) $\mathbf{f}_m$ is determined by $\mathbf{a}_v$ and $\mathbf{a}_p$ as follows:

$$\mathbf{f}_m = w_p\mathbf{a}_p + (1 - w_p)\mathbf{a}_v, \tag{8}$$

where blending weight w_p is used to adjust the balance between the two constraints. In our experiments, $T_m = 0.5$ and $w_p = 0.5$ are used. The position $\mathbf{p}_m$ and velocity $\mathbf{v}_m$ are updated based on the force $\mathbf{f}_m$ using Equation (5).

The state of the agent is changed between walking and standing, as follows: During walking, when the velocity is lower than the threshold at the end of a walking cycle, the state is changed to standing. While standing, when the position is moved over the threshold, the state is changed to walking. During the transition between walking and standing, the poses of the two motions are interpolated.

Although our method can generate walking motion along a curved trajectory, the generated motion may become unnatural when the agent makes sharp turns or small steps, even at the beginning of the step. Such motions can be generated by introducing other states and motion data. Because our primary objective was to introduce the feedback force, such extensions of locomotion generation were not tested.

VI. Feedback Force Based on Locomotion State

This section describes the feedback force from motion generation to the crowd simulation.

The feedback force $\mathbf{f}_{fb}$ maintains the velocity of the agents. It is computed in a manner similar to the locomotion force $\mathbf{f}_m$ using equations (6)–(8). Instead of the position $\mathbf{p}_m$, velocity $\mathbf{v}_m$ and time T_m, the position $\mathbf{p}_t$, velocity $\mathbf{v}_t$, and time T_t of the terminal point of the current walking cycle are used. If the feedback force $\mathbf{f}_{fb}$ becomes excessively large at the end of the walking cycle, it is limited to the maximum speed s_{max}.

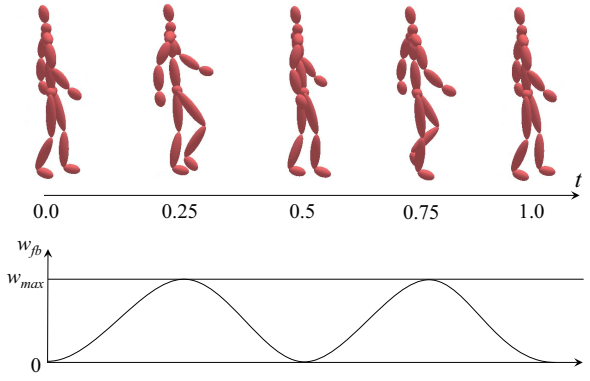


Fig. 3. Walking cycle and feedback force weight function. In the middle of step at around t=0.25 and 0.75, a stronger force is applied. At the beginning and end of step around t=0.5 and 1.0, a weaker force is applied.

The strength of the feedback force was adjusted based on the locomotion state, as shown in Figure 3. In the middle of the step, a stronger force was applied to prevent changes in the direction of movement. At the beginning and end of the step, a weaker force was applied so that the agent could freely change its moving direction. The weight of the feedback force w_{fb} is computed using a sinus function as follows:

$$w_{fb} = 0.5 \times w_{max}(1 - \cos(2\pi t)), \tag{9}$$

where t is the current generic motion time $(0 < t < 1)$ and w_{max} is the maximum weight parameter. In our experiments, $w_{max} = 0.5$ is used.

Based on the feedback force $\mathbf{f}_{fb}$ and its weight w_{fb}, the combined force $\mathbf{f}'$ is computed as:

$$\mathbf{f}' = (1 - w_{fb})\mathbf{f} + w_{fb}\mathbf{f}_{fb}. \tag{10}$$

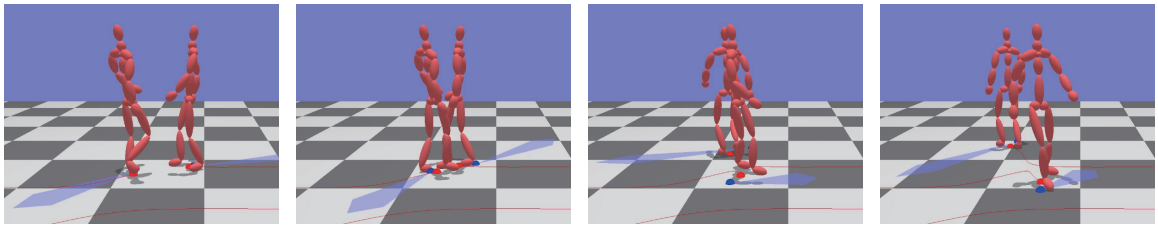
The combined force $\mathbf{f}'$ is used with equation (5) to update the position $\mathbf{p}_s$ and velocity $\mathbf{v}_s$ of the agent in the crowd simulation.

By introducing this method, the average driving and collision avoidance forces decreased. To compensate for this, the parameters for these forces, k_d, k_c, k_o are multiplied by two.

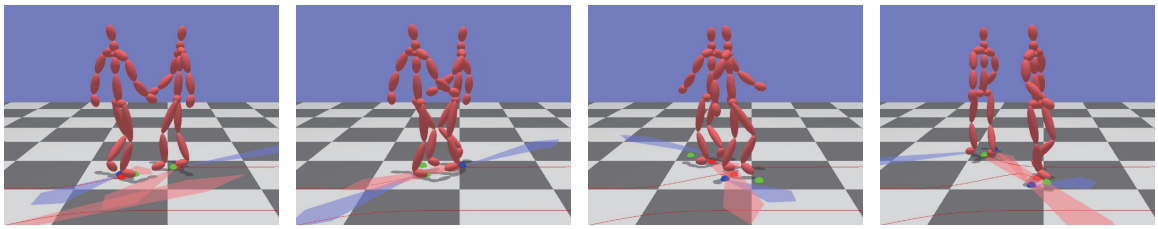
VII. Experimental Results

In this section, we present experimental results. As it is difficult to quantitatively evaluate the naturalness of the generated motions, we present the results of crowd animation with and without our method on several scenes and compare them to identify the effectiveness of our method. The resulting animations were included in the accompanying video.

First, we present the results for a scene with a small number of agents in Figure 4. One agent moves from left to right through two other agents that move in the opposite direction.



(a) generated animation without feedback force



(b) generated animation with feedback force

Fig. 4. Results of simulation obtained without (top) and with (bottom) our method. The series of images show the progress of two simulations (left to right). One agent that travels from left to right passes between two agents that travel from right to left.

After the agent avoids the first opponent, the direction of movement must be changed to avoid the second opponent. Without the feedback force, when the agent changes direction, the generated motion becomes unnatural. With the feedback force, although the agent still changed its moving direction, the naturalness of the generated motion was improved.

Second, we presented the results for a scene with a relatively large number (approximately 30) of agents in the accompanying video. Agents randomly appeared on the left or right end of the scene and moved toward the other end of the scene. In the resulting animations, motions similar to those in the first scene are observed.

Using feedback force, responsiveness may be reduced in exchange for the naturalness of motion. This can be solved by introducing a method for predicting the movements of nearby agents and responding to them early.

Regarding computation speed, because our method is simple, the additional computational costs for computing and applying the feedback force are small. It is possible to animate a large number of agents in real-time.

VIII. Conclusion

In this study, we proposed a crowd simulation method that considers the locomotion state of an agent. Our key idea is to adjust the velocity of the agent so that it is maintained during the step while it is allowed to be changed at the beginning of the step. Although we applied our approach to the social force model in this study, it can also be applied to other crowd simulation methods.

Our future work will extend our method to consider more details of the locomotion state. Our current method uses only the walking motion time and does not use the pose of the agent for crowd simulation. By considering this, more precise control is possible. The application of our approach to other crowd simulation and motion generation methods will be investigated in future work. In particular, a combination of data-driven motion generation methods, such as [10], [11], can generate more natural motions.

Acknowledgements

This work was supported in part by a Grant-in-Aid for Scientific Research (No. 21K12192) from the Japan Society for the Promotion of Science (JSPS).

References

[1] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, 1995.

[2] Y. Sun and H. Liu, "Crowd evacuation simulation method combining the density field and social force model," *Physica A: Statistical Mechanics and its Applications*, vol. 566, p. 125652, 2021.

[3] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions on Graphics (ACM SIGGRAPH 2006)*, vol. 25, no. 3, pp. 1160–1168, 2006.

[4] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Transactions on Graphics (ACM SIGGRAPH Asia 2009)*, vol. 28, no. 5, pp. 122:1–8, 2009.

[5] J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*, 2011, pp. 3–19.

[6] R. Hughes, J. Ondrej, and J. Dingliana, "Holonomic collision avoidance for virtual crowds," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA) 2014*, 2014, pp. 203–111.

[7] J. Lee, J. Won, and J. Lee, "Crowd simulation by deep reinforcement learning," in *Motion, Interaction and Games (MIG) 2018*, 2018, pp. 2:1–7.

[8] S. K. Takeshi Sakuma, Tomohiko Mukai, "Psychological model for animating crowded pedestrians," *Computer Animation and Virtual Worlds*, vol. 16, no. 3-4, pp. 343–351, 2005.

[9] N. Lockwood and K. Singh, "Biomechanically-inspired motion path editing," in *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2011*, 2011, pp. 281–287.

[10] K. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 473–482, 2002.

[11] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. Article No. 42, 1–13, 2017.