Crowd Simulation Using Velocity Field Map and LSTM Neural Network

Yuanyuan Peng
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Iizuka, Fukuoka, Japan
peng.yuanyuan549@mail.kyutech.jp

Masaki Oshita
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Iizuka, Fukuoka, Japan
oshita@ai.kyutech.ac.jp

***Abstract*—In this paper, we propose a novel method that uses a velocity field map and long short-term memory (LSTM) neural network to control agents to avoid collisions with nearby agents by considering their previous trajectories. Our method uses a series of continuous velocity field maps that represent the surrounding conditions of the controlling agent. A velocity field map comprises 2-channel images containing the positions and velocities of the controlling and nearby agents. We employed an LSTM neural network to predict the current velocity field map from the velocity field maps of the previous frames to determine the desired velocity of the controlling agent. The agent is controlled using a social force method based on the desired velocity. The novelty of our method lies in the use of the velocity field map and LSTM neural network. By using the velocity field map, the information of the controlling agent and its surrounding salutation is efficiently represented. In addition, by using a series of continuous velocity field maps and LSTM, the agent is efficiently controlled considering the previous frames. We present experimental results and discuss the advantages of the proposed approach.**

***Index Terms*—Crowd simulation, velocity field, deep learning, LSTM**

I. Introduction

Crowd simulation is the process of simulating the movements of a large number of agents, and has several applications such as in computer animation and video games. Controlling agents toward their target positions while avoiding collisions with other agents and obstacles is an important problem. Although several algorithms have been developed, it is still difficult to realize human-like behaviors. Humans determine their heading directions based on surrounding information; however, the decision-making process is complex and difficult to replicate using algorithms or models.

In this paper, we propose a novel method that uses velocity field map and long short-term memory (LSTM) neural network for controlling agents to avoid collision with nearby agents by considering their previous trajectories. Our method uses a series of continuous velocity field maps that represent the surrounding conditions of the controlling agent. A velocity field map comprises 2-channel images that contain the positions and velocities of the controlling and nearby agents. We employed an LSTM neural network to predict the current velocity field map from the velocity field maps of the previous frames to determine the desired velocity of the controlling agent. The agent is controlled using a social force method based on the desired velocity.

The novelty of our method lies in the use of the velocity field map and LSTM neural network. By using the velocity field map, the information of the controlling agent and its surrounding salutation is efficiently represented. In addition, by using a series of continuous velocity field maps and LSTM, the agent is efficiently controlled considering the previous frames.

We present the experimental results and discuss the advantages of the proposed approach.

The remainder of this paper is organized as follows: Section II reviews the related studies. In Section III, we describe the proposed method. Section IV presents the experimental results and a discussions. Finally, Section V concludes the paper.

II. Related Work

Several methods have been developed for crowd simulation, such as model-based methods that control agents via algorithms that mimic human behavior. These include the social force model [1], flocking model [2], path-planning algorithms [3], [4], and a combination of macroscale and microscale models [5], [6]. Moreover, algorithms that focus on collision avoidance [7]–[9] have been developed. Reinforcement learning has also been used to determine the parameters for the controlling agents [10], [11].

Data-driven methods [12]–[15] that use sample data from crowd animations have also been developed. These approaches aim to synthesize plausible crowd animations rather than controlling individual agents.

Recently, some methods have employed machine-learning techniques to control agents. Boatright et al. [16] used multiple decision trees classified according to the patterns of nearby agents in nine directions. Shamsul et al. [17] used a support vector machine to select a class with an appropriate heading direction. These methods utilize the positions and velocities of a fixed number of nearby agents as feature vectors. They cannot consider the previous trajectories of the surrounding agents.

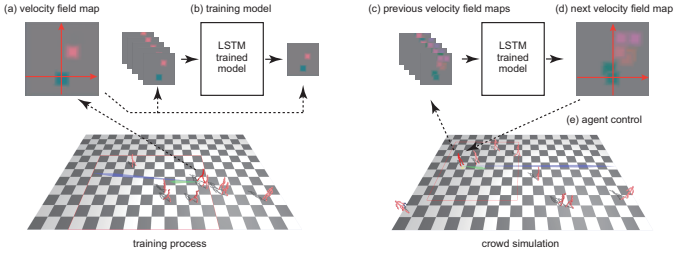

Fig. 1. Overview of the proposed method. (a) Velocity field maps (2-channel images) are generated from example crowd animations. They contain the positions and velocities of the controlling and nearby agents. (b) An LSTM neural network is trained to predict the next velocity field map from five previous velocity field maps. (c) When controlling an agent, five previous velocity field maps are generated. (d) Using the trained model, the next velocity field map is estimated. The desired velocity is obtained from the estimated image. (e) The agent is controlled by using a force-based model based on the desired velocity.

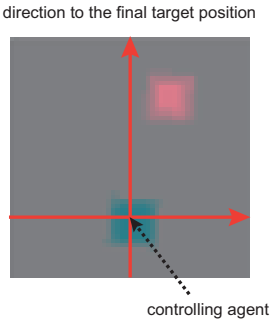

Fig. 2. Velocity field map in agent space. The map is a 2-channel image that represents the situation of the $8\mathrm{m} \times 8\mathrm{m}$ space around an agent as viewed from above. The y-axis faces the final target position of the agent. The blobs represent the positions and velocities of the controlling and nearby agents. The velocities in the x- and y-axes are depicted by the red and green channels of pixels, respectively. In this velocity field map, the controlling agent is moving upward, while one nearby agent is moving downward.

Oshita proposed a method that employs a heat map and neural convolutional network for controlling the agents [18], and our method adopts a similar approach. The previous method used a grayscale heat map containing only the relative speeds of surrounding agents. It also uses a convolutional neural network and current heat map to determine the heading direction of the controlling agents. Therefore, the previous trajectories of the surrounding agents could not be considered.

Our method solves the problems of previous methods [18] by employing a velocity field map and an LSTM neural network. We present a comparison between our method and the previous method in Section IV.

III. Proposed Method

An overview of the proposed method is presented in Figure 1. Our method uses velocity field maps and an LSTM neural network. During the training process, sets of consecutive velocity field maps were generated from sample crowd animations, and the LSTM neural network was trained on the data. In the simulation process, the trained model was used to predict the next velocity field map from the velocity field maps of the previous frames to determine the desired velocity of the controlling agent. The agent was controlled using a force-based method at the desired velocity.

In this section, we describe our velocity field map, neural network design, training process, and simulation process.

A. Velocity Field Map

The velocity field map is a 2-channel image containing information on the horizontal velocities of the controlling and surrounding agents from the top view, as shown in Figure 2. In our implementation, we use a velocity field map of 32×32 pixels to represent the $8\mathrm{m} \times 8\mathrm{m}$ space around the agent. The coordinates of the velocity field map are defined such that the y-axis faces the target position of the agent. Since the side containing the target position is more important than the other side, the agent is located at a lower center position in the velocity field map. Each blob represents the position

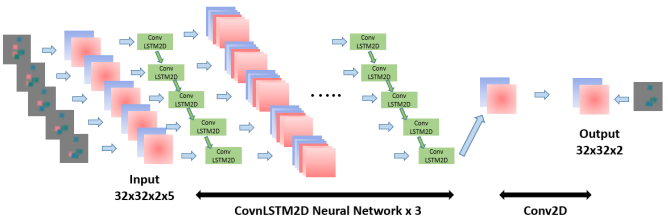

Fig. 3. Network design.

and velocity of a controlling or nearby agent. Each agent was drawn as a 4×4-pixel blob on a velocity field map.

The transformation matrix $\mathbf{M}$ from the scene coordinates to the velocity field map coordinates is computed based on position $\mathbf{p}$ and target position $\mathbf{t}$ of the controlling agent. The velocity field map was generated by drawing the blobs of the controlling agent and nearby agents within the surrounding area on the 2-channel image. The inverse transformation $\mathbf{M}^{-1}$ is used to compute the desired velocity $\mathbf{v}_d$ in scene coordinates from the velocity field map coordinates.

To determine each channel value for an agent, its velocity was scaled from $(-v_{max}, v_{max})$ to $(0.0, 1.0)$. If the velocity exceeded this range, it was clamped within the range. In our implementation, we set $v_{max} = 2\mathrm{m/s}$. The background pixel values of the velocity field map are $(0.5, 0.5)$.

The desired velocity of the controlling agent was determined by estimating the velocity field map of the next frame from the velocity field maps of the previous frames using an LSTM trained model.

B. Neural Network Design

The design of the proposed neural-network model is illustrated in Figure 3. The input was a series of 2-channel images representing the velocity field map of the previous frames. The output is a 2-channel image representing the velocity field map of the next frame. For our implementation, we used five previous images. The interval between frames was 1.5

seconds. The image size was 32×32 pixels, as explained in Section III-A.

The TensorFlow framework [19] was used to train the network model. The network comprised three ConvLSTM layers and one Conv2D layer. The RMSprop optimization method was used to train the model using the Tanh function for activation. The average squared mean was used as the error function.

C. Training Process

Existing crowd animations were used to generate sample velocity field maps to train our model. Examples of crowd animations include the trajectories of people in a scene. Each trajectory contains the initial and terminal times, positions, and orientations of the agents at the keypoints. The terminal position of the trajectory is used as the target position for the agent. Each set of training examples contained six velocity field maps from five consecutive previous frames and the next frame. A set of velocity field maps was generated for each agent in each animation frame.

D. Simulation Process

In a crowd simulation, each agent has its position $\mathbf{p}$, velocity $\mathbf{v}$, and target position $\mathbf{t}$. The desired velocity $\mathbf{v}_d$ of each agent in each frame is determined using the trained model. Five velocity field maps of the previous frames were generated for each agent. The trained model estimated the velocity field map of the next frame.

The channel values at the position of the controlling agent (origin in the coordinates shown in Figure 2) in the estimated velocity field map are used to determine the desired velocity. We calculated the average of the 16 pixels around the agent's position. The velocity in the velocity field map coordinates was transformed into the desired velocity $\mathbf{v}_d$ in scene coordinates.

The position and velocity of agents $\mathbf{p}, \mathbf{v}$ were updated based on the desired velocity $\mathbf{v}_d$ using the social force method [1], [18] as follows:

First, the driving force (acceleration) $\mathbf{f}_d$ was computed to obtain the desired velocity $\mathbf{v}_d$ as

$$\mathbf{f}_d = \frac{\mathbf{v}_d - \mathbf{v}}{s} \tag{1}$$

where s denotes the time required for the agent to reach the desired velocity. In our implementation, $s = 2$ seconds was used.

The collision avoidance force, which is the sum of the repelling forces against nearby agents, was computed as:

$$\mathbf{f}_c = \sum_{i<G} -k_c \left(1 - \frac{|\mathbf{p}_i - \mathbf{p}|}{d_c}\right) \frac{\mathbf{p}_i - \mathbf{p}}{|\mathbf{p}_i - \mathbf{p}|}, \tag{2}$$

where d_c and k_c are the distance and scale parameters, G is the group of nearby agents within distance d_c and $\mathbf{p}_i$ denotes the positions of the nearby agents. We used $d_c = 1.5$ and $k_c = 0.5$.

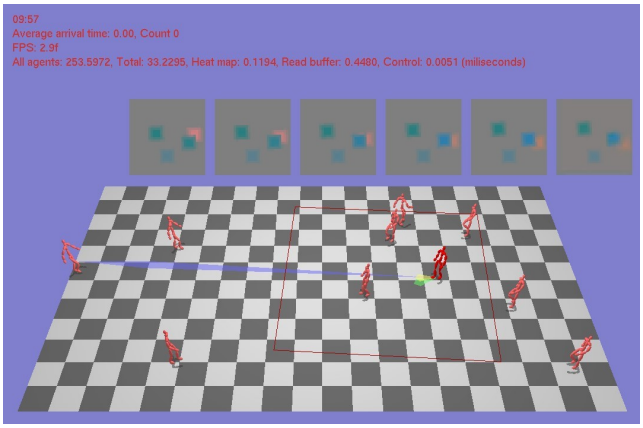

Fig. 4. Example of generated crowd simulation. Five previous and one estimated velocity field map of the selected red agent is displayed. The blue arrow indicates the target position of the agent. The green arrow indicates the desired velocity. The yellow arrow indicates the total force applied to the agent.

The force for avoiding collisions with obstacles and walls is computed similarly as

$$\mathbf{f}_o = \sum_{j<O} -k_o \left(1 - \frac{|\mathbf{p}_j - \mathbf{p}|}{d_o}\right) \frac{\mathbf{p}_j - \mathbf{p}}{|\mathbf{p}_j - \mathbf{p}|}, \tag{3}$$

where d_o and k_o are the distance and scale parameters, respectively. O is a group of obstacles and walls within a distance d_o, and $\mathbf{p}_j$ denotes the closest positions of the obstacles and walls. We used $d_c = 1.0$ and $k_c = 2.0$.

Finally, the total force applied to the agent was computed as follows:

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

The velocity $\mathbf{v}$ and position $\mathbf{p}$ of the agent are updated according to the applied force and time step Δt in the simulation. The velocity of each agent is limited to its maximum speed s_{max}. In our experiment, s_{max} values between $0.2\mathrm{m/s}$ and $0.5\mathrm{m/s}$ were randomly assigned to each agent.

Figure 4 presents an example of a generated crowd simulation. The walking motions of the agents were generated by applying a cycle of a walking motion sequence to their positions and orientations.

IV. Experimental Results and Discussion

In this section, we present the experimental results. The results of the estimated velocity field map, crowd simulation, and computational efficiency are described and discussed.

A. Model Training

We train our LSTM model using crowd animation data of pedestrians on streets [12]. For comparison, we used the same crowd animation data as in the previous method [18]. From the crowd animation data, we generated 1200 sets of training examples. We used TensorFlow [19] with the RMSprop optimization method and Tanh activation function. We set the batch size to 10 and the number of epochs to 100.

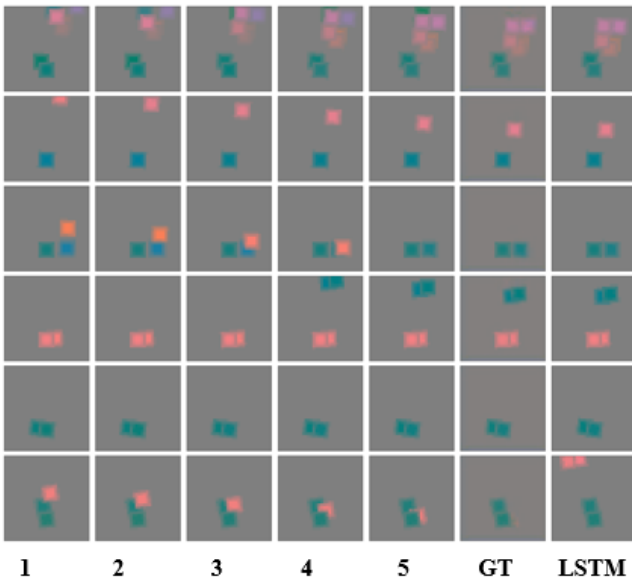

Fig. 5. Results of estimated velocity field maps of the LSTM model. Each row represents an example that contains five previous velocity field maps (1-5), ground truth (GT) of the next velocity field map, and the estimated result by our LSTM model (LSTM).

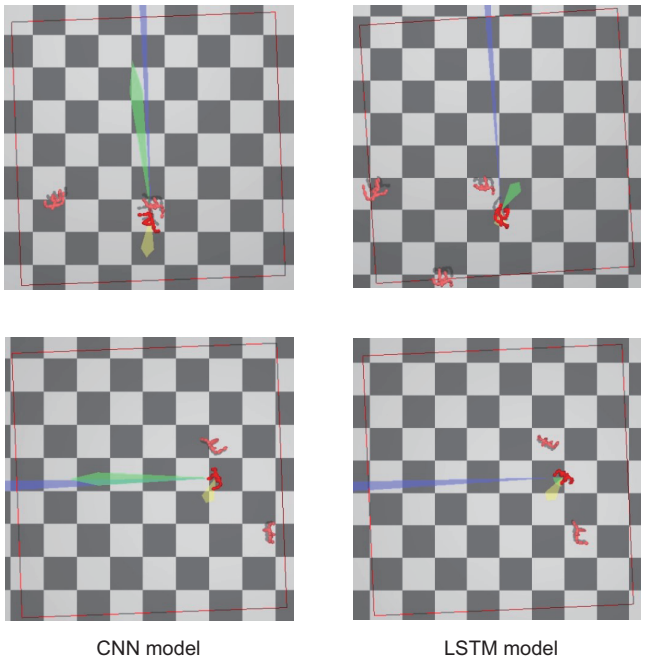

Fig. 6. Results of crowd simulation. Each row represents an example that contains the results of the CNN model [18] and our LSTM model in similar simulations. The blue arrow indicates the target position of the agent. The green arrow indicates the desired velocity. The yellow arrow indicates the total force that is applied to the agent.

For comparison, the CNN model was also trained using the same crowd animation data as described in [18].

B. Results of Estimated Velocity Field Maps

To evaluate the trained model, we applied it to examples used for training and compared the estimated velocity field maps with the ground-truth images. Six examples of estimated velocity field maps are shown in Figure 5.

From the results, it can be seen that all estimated velocity field maps were similar to the ground-truth images. The agents that disappeared in the ground truth images also disappeared in the estimated images in the No. 1 (first row), No. 3 (third row), and No. 6 (sixth row). In the estimated images of results No. 2 (second row) and No. 4 (fourth row), the agents moved to the same positions as in the ground-truth images. In the No. 5 (fifth row) result, there are no obvious changes in the color and positions of the agents in the estimated images, because the two agents always maintained their moving directions and velocities in the previous frames.

C. Results of Crowd Simulation

Using the trained model, we ran a crowd simulation on a street scene that was similar to crowd animation data [12]. Each agent appeared on the left or right edge of the scene. The target position was set at the opposite edge of the scene, and the maximum number of agents in the scene was maintained at a specified number.

The crowd simulation results are shown in Figure 6. We compared the results of the CNN model [18] with those of our LSTM model under the same initial conditions. These results show that our LSTM model properly determines the heading directions and improves the collision avoidance ability of the agents. In these examples, the red agents adjusted their moving directions to avoid collision with other agents in the LSTM model, whereas they slightly collided with each other in the CNN model.

D. Evaluation of Crowd Simulation

Although it is difficult to quantitatively evaluate the naturalness of the crowd simulation results, we compared the number of collisions estimated with the CNN model [18] and our LSTM model. The results are presented in Table I. Our LSTM model decreased the number of collisions compared to that with the CNN model. These results indicate that the agents controlled by our LSTM model selected proper heading directions and avoided collisions with other agents.

E. Computational Efficiency

The proposed method is simple and operates in real-time on a standard computer. We measured average execution times for a desktop computer with an Intel Core i5 -8400 2.8-GHz CPU and an NVIDIA GeForce GTX 1050 Ti GPU. The average time required to control an agent in one frame using the LSTM model was 52 ms. Our model requires more computational time than that with, the CNN model [18], because five previous velocity field maps must be generated to estimate the next velocity field map.

In this simulation, the LSTM model was used for each agent in each frame. The simulation can be run in real-time using a small number of agents, as shown in Figure 4. By using the LSTM model to determine the desired velocity in some

TABLE I
NUMBER OF COLLISIONS IN CROWD SIMULATION WITH THE CNN MODEL [18] AND OUR LSTM MODEL.

number of arrived agents	collision count		
	social force model	CNN model [18]	LSTM model
10	9	1	1
20	21	8	1
30	25	10	3
40	31	13	7
50	39	15	10
60	41	19	10

intervals instead of every frame, our method can be applied for real-time simulations with a large number of agents. This approach is reasonable, considering that humans require time to react to surrounding situations and change their heading directions.

F. Discussion

In this study, we introduced velocity field maps and LSTM neural network in crowd simulation. Our method achieved better results than a previous method [18] that used grayscale heat maps and CNNs. In our experiments, we used five previous velocity field maps after testing different numbers and intervals. In future work, further analysis of efficiency with different parameters is expected. Although we did not include obstacles in our crowd simulation, our method should be able to handle both fixed obstacles and moving agents.

As discussed in [18], the choice of the pixel size of the velocity field map and spatial size mapped to the velocity field map are important. Although we used the same size as in the previous method for comparison, these sizes affected the results of the crowd simulation. For example, in our experiments, the agents cannot be considered outside the $8\mathrm{m} \times 8\mathrm{m}$ space. Selecting an appropriate size or combining multiple sizes of the velocity field maps is a topic for future work. Additionally, although we constructed a single trained model from all the agents, such as in crowd animations, it was possible to train separate models for different types of behaviors of groups. The construction and experiments of different trained models will also be part of future work.

V. CONCLUSION

In this paper, we propose a novel method that uses a velocity field map and an LSTM neural network to control agents to avoid collisions with nearby agents considering their previous trajectories. Our results show that our approach can simulate natural movements of crowd efficiently. They also show that our method presents improvements over a previous method using a similar approach [18]. Our future work includes further analysis of efficiency of our method with different parameters, determining an appropriate size or combining multiple sizes of the velocity field maps, and training separate models for different types of behaviors of groups.

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] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87*, 1987, pp. 25–34.

[3] A. Sud, E. Andersen, S. Curtis, M. C. Lin, and D. Manocha, "Real-time path planning in dynamic virtual environments using multiagent navigation graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 3, pp. 526–538, 2008.

[4] M. Kapadia, A. Beacco, F. Garcia, V. Reddy, N. Pelechano, and N. I. Badler, "Multi-domain real-time planning in dynamic environments," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2013*, 2013, pp. 115–124.

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

[6] 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.

[7] A. Best, S. Narang, S. Curtis, and D. Manocha, "Densesense: Interactive crowd simulation using density-dependent filters," in *ACM SIGGRAPH/Eurographics Symposium on Computer animation (SCA) 2014*, 2014, pp. 97–102.

[8] J. P. Julien Bruneau, "Eacs: Effective avoidance combination strategy," *Computer Graphics Forum*, vol. 36, no. 8, pp. 108–122, 2017.

[9] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha, "Simulating heterogeneous crowd behaviors using personality trait theory," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2011*, 2011, pp. 43–52.

[10] S. J. Lee and Z. Popović, "Learning behavior styles with inverse reinforcement learning," *ACM Transactions on Graphics (ACM SIGGRAPH 2010)*, vol. 29, no. 4, p. Article No. 122, 2010.

[11] 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.

[12] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[13] B. Yersin, J. Maïm, J. Pettré, and D. Thalmann, "Crowd patches: populating large-scale virtual environments for real-time applications," in *Symposium on Interactive 3D graphics and games (I3D) 2009*, 2009, pp. 207–214.

[14] A. Bera, S. Kim, and D. Manocha, "Efficient trajectory extraction and parameter learning for data-driven crowd simulation," in *Graphics Interface 2015*, 2015, pp. 65–72.

[15] 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.

[16] C. D. Boatright, M. Kapadia, J. M. Shapira, and N. I. Badler, "Generating a multiplicity of policies for agent steering in crowd simulation," *Computer Animation and Virtual Worlds*, vol. 26, no. 5, pp. 483–494, 2015.

[17] M. Shamsul, M. Oshita, T. Noma, M. S. Sunar, F. M. Nasir, K. Yamamota, and Y. Honda, "Making decision for the next step in dense crowd simulation using support vector machine," in *ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI) 2016*, 2016, pp. 281–287.

[18] M. Oshita, "Agent navigation using deep learning with agent space heat map for crowd simulation," *Computer Animation and Virtual Worlds*, vol. 30, no. 3-4, pp. 1–12, 2019.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/