accurate
INSTITUTE FOR CROWD SIMULATION

Technical Report: AI Methods for Crowd Simulation

Introduction

In today's rapidly evolving urban environments, the safe and efficient design of large buildings and public spaces has never been more critical. As cities grow and events increase in scale, managing crowd movement, ensuring public safety, and optimizing space usage present enormous challenges. Traditionally, planners have relied on manual methods—often time-consuming, inflexible, and prone to human error. However, the growing complexity of these projects demands more advanced solutions. Digital technologies, especially crowd simulation tools, have emerged as key enablers in transforming the way these challenges are addressed. Crowd simulation allows for the precise modeling of human movement within various environments, enabling planners to predict and mitigate potential risks long before construction or event execution. Yet, despite these advancements, current simulation technologies often suffer from long processing times and unrealistic behavioral outputs, limiting their broader adoption.

The ACSAI project addresses these limitations by integrating state-of-the-art Artificial Intelligence (AI) methods into crowd simulation software. To achieve this goal, we have explored three different approaches: The Convolutional Neural Network (CNN) approach for automatic simulation results from our previous project BEYOND, the Reinforcement Learning (RL) approach for more realistic agent behavior and the Dynamic Mode Decomposition (DMD) approach for faster simulation computation times. In this report the different approaches are explained and discussed.

Through the different innovative AI-driven approaches, ACSAI is ready to improve the field of crowd management and provide robust and reliable tools for architects, security planners and authorities.

Our simulation software *crowd:it*

Crowd simulation enables the virtual modeling of evacuations, allowing for the analysis of potential bottlenecks. Key metrics such as evacuation times and congestion durations can be calculated and contextualized to assess safety performance. These parameters provide critical support for safety planners, enabling them to evaluate different safety scenarios, compare alternative designs, and make informed decisions about compensatory measures. By incorporating crowd simulation, planning becomes

more resilient and reliable; safety can be objectively validated, even when deviations from standard building regulations occur. Beyond evacuation planning, crowd simulation also offers valuable insights into the general operational efficiency of buildings or event spaces, providing information about service levels under normal conditions.

The most modern crowd simulation model on the market is our product *crowd:it*. Planners can import the geometry and specify destinations, along with the number of individuals and their movement order and simulate different scenarios. Based on the microscopic data, crowd:it computes and visualizes key metrics like evacuation times or congestion durations. crowd:it utilizes a microscopic, agent-based simulation model, where each pedestrian is simulated individually. The simulation assigns different properties to each pedestrian, for example a different velocity and torso size. To advance the simulation, each step of a pedestrian that is needed to reach a given destination is simulated one after the other.

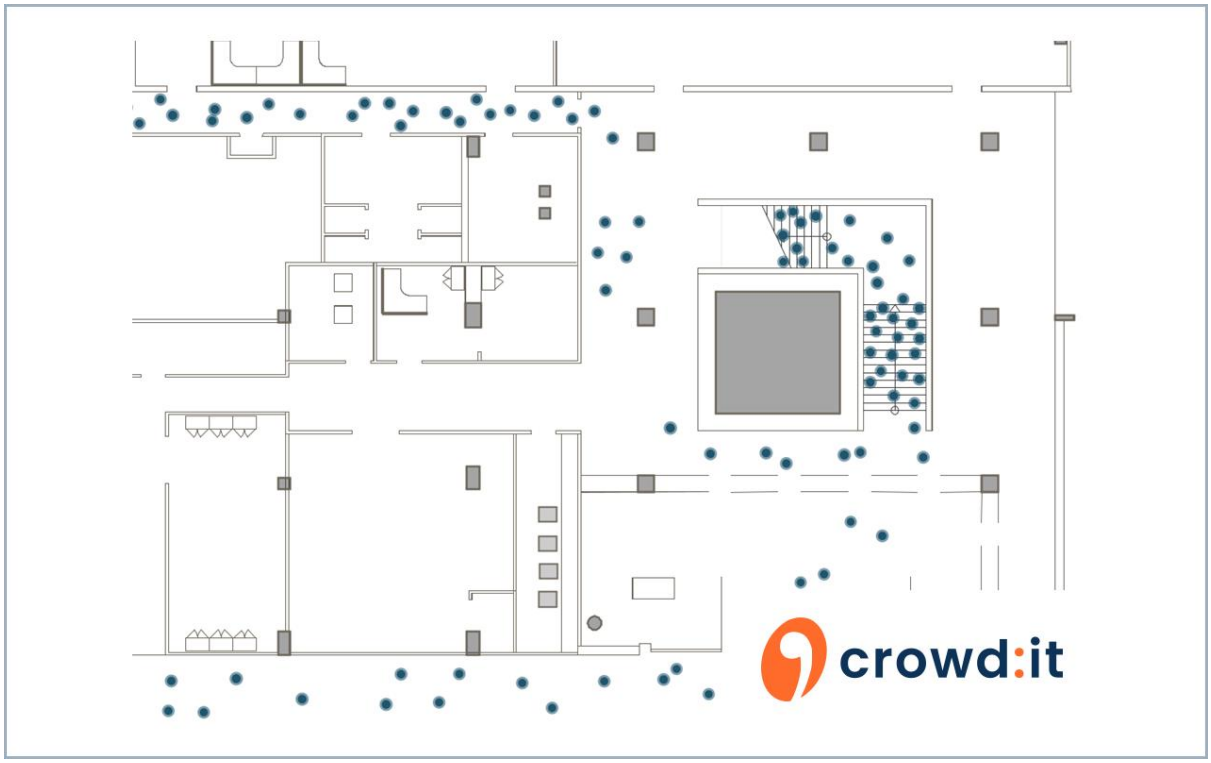

Figure 1: Snippet from the simulation software crowd:it, including a floor plan and blue circles representing the position of individual agents.

The **Optimal Steps Model** marks a significant advancement in crowd simulation. Unlike traditional models borrowed from other disciplines, this model is rooted in the core

principle of human movement. The simulation starts with the basic unit of movement—a single step. Each pedestrian has a dynamic step length that adjusts according to their speed, direction, and proximity to obstacles or other individuals. This allows each step to adapt to real-time environmental conditions.
Through a dynamic navigation field, agents continually reassess their optimal route based on the changing circumstances, treating other agents as obstacles and, if necessary, opting for a longer but ultimately faster path to their destination.

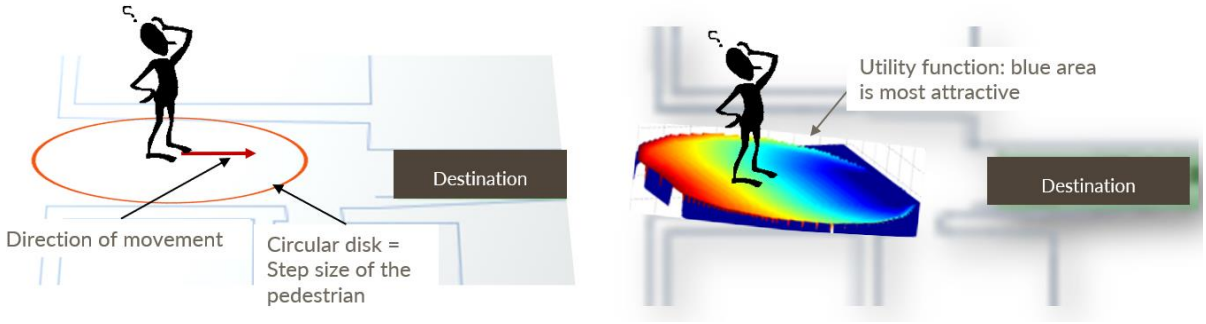

Figure 2: Visualization of how the next step of an individual agent is simulated in crowd:it. Within a circular disk given by the current maximal step size and position of the agent, the model searches the point with the best value of the utility function given by the navigation field.

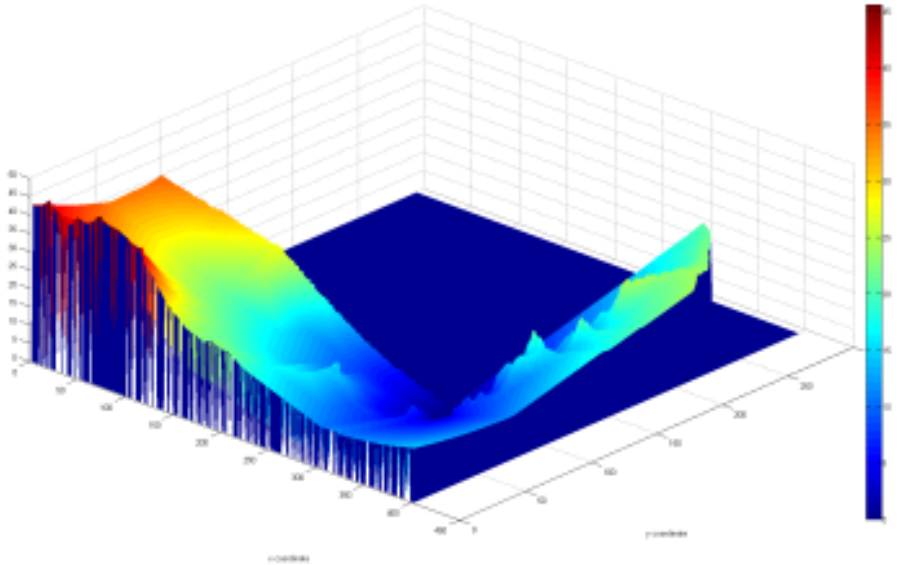

Figure 3: Visualization of the navigation field. It takes into account distance from destination as well as repulsion from obstacles and other people.

Different Machine Learning Approaches

Convolutional Neural Network

Our initial plan for the ACSAI project was to transfer the results from our research project BEYOND into a ready-for-market product. In BEYOND, we researched how Convolutional Neural Networks (CNN) could predict the results of our crowd simulation in order to achieve significantly smaller analysis times. The goal was to deeply integrate the crowd simulation into the planning process of railroad stations and enable direct support of the design process and extensive comparison of variants. A neural network was developed and trained with the help of pre-calculated simulation data. The trained CNN should give answers at the push of a button, whether a design is compliant or not. This would accelerate the process enormously since it quickly sorts out designs that are not feasible. Thus, only pre-checked design variants would be simulated for final confirmation and optimization. With this approach, we hoped to significantly accelerate the planning process and reduce the risk of planning mistakes.

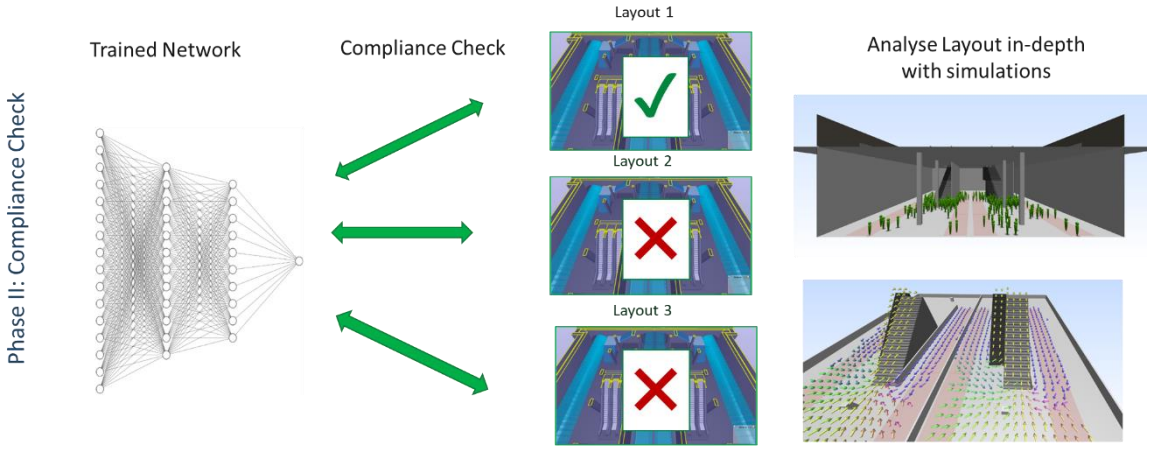

Figure 4: The idea of our CNN approach is to use neural networks for quick checks, whether variants of building layouts complied with the crowd management goals.

Although we still consider this a promising approach, we encountered significant delays in its implementation. While it was relatively straightforward to apply the method to predefined models in the early phases of a building project, generalizing it to other models in a cost-effective manner proved challenging due to the wide variety of building designs.

The primary limitation of this approach lies in the need to train the NN with a specific building model before it can be used for predictions. This requirement makes it impractical to create a generalized NN model capable of analyzing any building without prior preparation.

As a result, we concluded that further research is necessary to address these challenges on a more fundamental level. Currently, the Georg Nemetschek Institute at the Technical University of Munich (TUM) is exploring this issue through the FORWARD project, investigating which machine learning models are best suited for automating the prediction of pedestrian flows.
Until we receive better results from this method, we explored other AI approaches that are discussed within the scientific community. Techniques such as Reinforcement Learning (RL) and Dynamic Mode Decomposition (DMD) were applied with distinct objectives: RL can be utilized to enhance the realism of agent behavior, while DMD can be employed to reduce computational time and accelerate simulations.

Reinforcement Learning

Model description and approach

Reinforcement Learning (RL) is an area of machine learning that aims to train intelligent agents to take purposeful actions in a dynamic environment. As crowd:it simulated the actions of individual pedestrians in dynamic crowds, RL seems a perfect fit for making agent behavior more flexible and allowing the agents to act more intelligently in complex situations. Such situations include pedestrians that encounter each other from different directions in narrow pathways.

With these goals in mind, we developed an approach for AI-driven step calculation of our agents using RL. The RL model takes observations of the environment as an input and produces an output that defines the next action of an agent. According to the consequences of the action, the network receives a reward or punishment and the next state of the environment. This is particularly suited for our application because there is no single objectively correct step in any situation that we could compare the network output to, but we can evaluate the consequences of an arbitrary step and thus provide the model with rewards so that it can learn useful agent behavior.
Whenever an agent is about to take a step, an image is produced that captures the current state of the simulation, as shown in Figure 5. This is fed into the neural network which then outputs an action that defines the size and direction of the agent's next step (see Figure 6). This action is realized by the agent in the simulated scenario. If the agent gets closer to the destination, they receive a positive reward. If they collide with an obstacle or other agent, the reward is negative. Inaction or implausible movements are also punished.

After collecting these experiences, the model is afterwards optimized using the Proximal Policy Optimization (PPO) algorithm. It entails two Deep Neural Networks for the actor and the critic, both of which are built with the Res-Net 50 architecture using pretrained parameters.
In contrast to the CNN approach, our reinforcement learning approach is not dependent on external training data. Instead, we produce the training data ourselves and only need to design appropriate training scenarios. Due to the focus on the perspective of individual agents, these do not have to capture the broad space of possible buildings.

Implementation

The starting point for our reinforcement learning project was a workshop with an external AI expert, the German consulting company kineo ai. Within one week, they shared important insights about reinforcement learning with us. Together, we defined the observation space, the action space and the reward function for our use case. Subsequently, we implemented the required additions to our software and set up the training pipeline. For the start, we chose simple scenarios so that the model could learn the basic behaviour before being fine-tuned to more complex situations.
An uncommon aspect of this application of RL is that multiple agents are controlled simultaneously within the same training scenario, as each agent must learn to account for the movements of others in its decision-making process. This introduces additional complexity, making it particularly challenging to achieve the desired learning outcomes. As a result, we have not yet reached a level of agent behavior that is suitable for inclusion in our product. However, we remain confident in the potential of this approach and have several ideas for further improvements, including refining how observations are generated and redesigning the reward function to better guide learning.

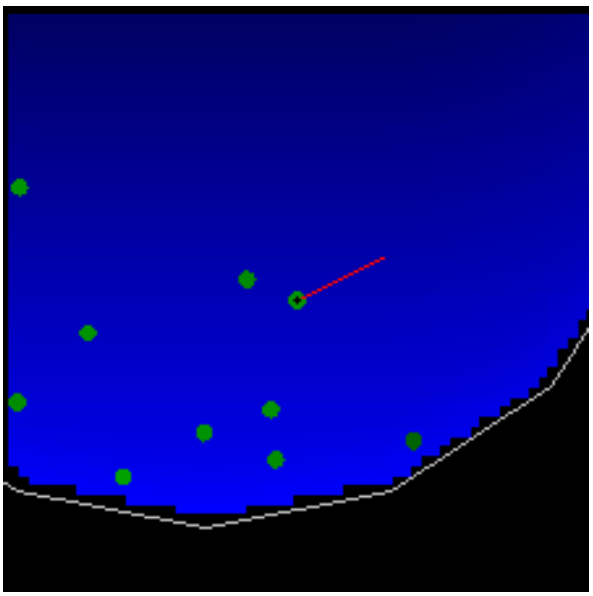

Figure 5: The observation space includes walls (white line), the agents (green dots, saturated according to their velocity) and the current direction of the agent that will take the next step (red line).

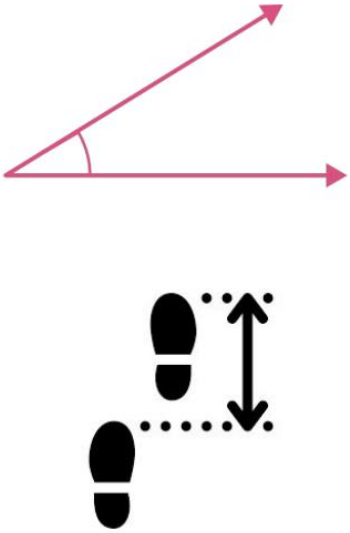

Figure 6: The action space consists of the angle, relative to the current direction, and the step length, relative to the maximal step length of the agent.

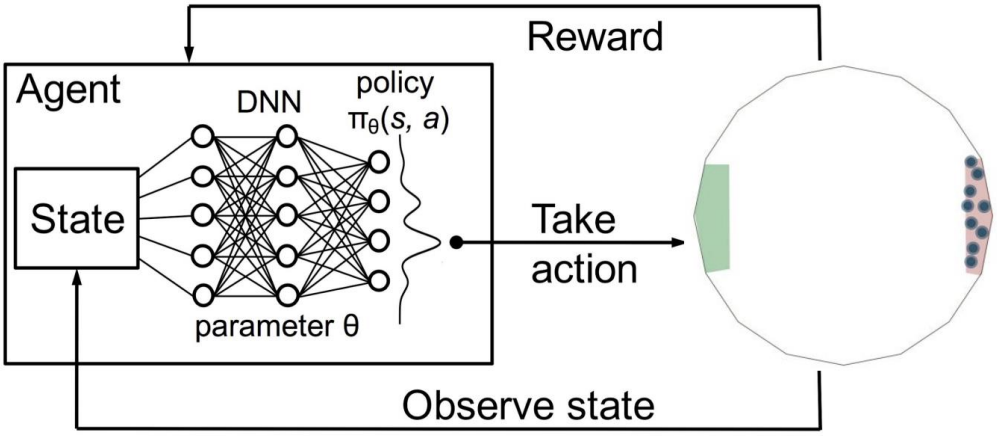

Figure 7: Our reinforcement learning model for controlling agents in a scenario observes the state of the scenario and produces an action, which is in turn executed by the agent in the scenario. It is trained according to rewards for the consequences of the actions.

Dynamic Mode Decomposition

Model description and approach

Dynamic Mode Decomposition (DMD) is a machine learning algorithm that predicts dynamic systems, operating on a time series of data. Crowd simulations model just such a dynamic system that evolves over time. This makes DMD a possible application for crowd simulations. This section describes how we implemented DMD and an extension of the DMD algorithm, Extended Dynamic Mode Decomposition (EDMD), in order to improve the performance of the crowd:it simulation kernel.

One factor that requires particularly high performance in this simulation model is the dynamic flooding: Suppose a pedestrian is placed in a scenario and needs to walk to a given destination. To simulate the individual steps of the pedestrian, the distance from each point to the destination needs to be calculated. This calculation must take into account the changing environment, in particular other pedestrians, and therefore needs to be recalculated on a regular basis. We implemented DMD to reduce the frequency of recalculation, saving time and energy.
Given a time series that describes how a dynamic system evolved in the past, DMD predicts how it will evolve in the future. Therefore, it is not suited for entirely replacing dynamic flooding. Instead, it operates in conjunction with dynamic flooding: In the beginning of the simulation, the regular dynamic flooding algorithm is used to calculate a time series of data. At a later point, DMD takes the given time series as input for its prediction. At this point, dynamic flooding is no longer required. In practice, it is necessary to evaluate the output of DMD on a regular basis. If it diverges from the results of dynamic flooding (which serves as „ground truth“), dynamic flooding once again is utilized to calculate a new time series of data. This process is visualized in Figure 8. While DMD is active, the error grows until it reaches a certain threshold. Then, dynamic flooding is activated once again and the error decreases. This approach allows to achieve an approximation of the dynamic flooding while reducing the computational effort.

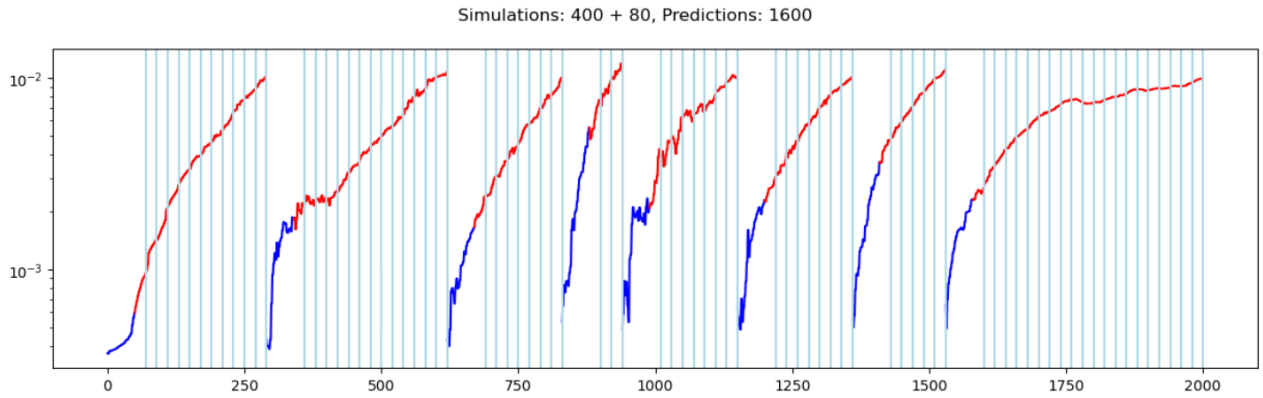

Figure 8: The line indicates how the error of DMD evolves over time. If the line is blue, dynamic flooding is active in the simulation. If the line is red, DMD is used.

Implementation

First, we evaluated if DMD is suitable for our use case and feasible to implement. For this purpose, we supervised a project seminar with students at the University of Applied Science in Munich, conducted by Prof. Dr. Köster. The ideas of these seminars is that students work together with industry partners to implement new scientific approaches in real world applications. In cooperation with accu:rate, the students implemented DMD and EDMD with regard to the dynamic flooding use case. They evaluated the results both for large and small, complex scenarios. The evaluation provided satisfactory results, such that we decided to pursue this approach further.

As a next step, we supervised a bachelor thesis that aimed to expand the proof of concept from the seminar as a working implementation in the crowd:it software. We granted the student access to our code base So that implementations could be carried out both by the student and by our developers. This method allowed us to fully integrate the approach described above in crowd:it.

The bachelor student evaluated the performance of the implementation on a basic scenario in crowd:it. The results showed that DMD can effectively learn and predict the navigation fields with comparable results to conventional dynamic flooding. Unfortunately, the work did not manage to decrease the overall better calculation time: Despite the theoretical efficiency of DMD, the additional steps for data transmission and processing proved to be time-consuming, so that the expected acceleration could not be achieved. Shorter calculation times could only be achieved at the expense of the simulation results.

In order to create real impact for our simulation software, especially in larger scenarios, the communication between Java (crowd:it) and Python (DMD) needs be further optimized. Furthermore it would be useful to explore the integration of Extended

Dynamic Mode Decomposition (EDMD) in order to increase the accuracy of the predictions. Extended Dynamic Mode Composition (EDMD) is a rather new extension to DMD that has been examined since 2015. It aims to improve DMD in nonlinear systems.

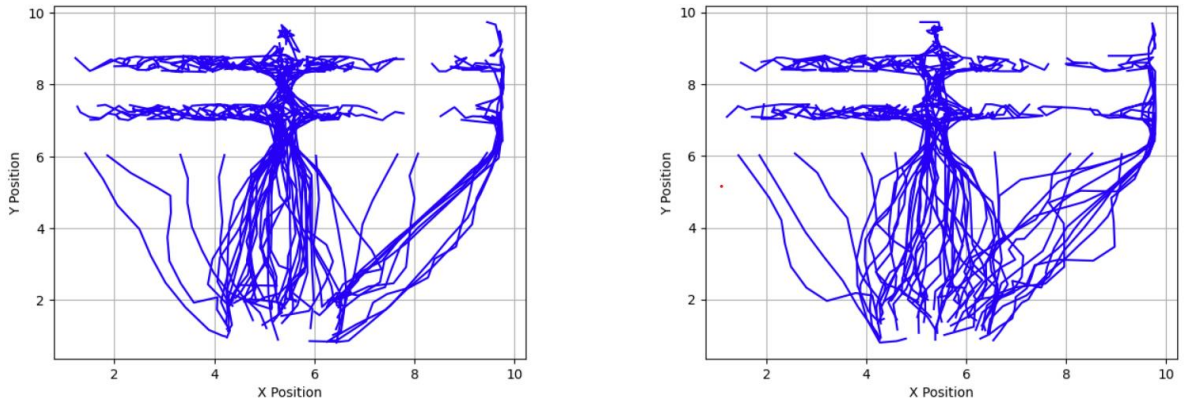

Figure 9: Traces of pedestrians in an exemplary scenario, based on conventional dynamic flooding (left) and DMD (right). This shows that DMD can approximate dynamic flooding close enough so that the resulting behavior is only changed marginally.

Conclusion

With convolutional neural networks, reinforcement learning and dynamic mode decomposition, we have been pursuing three different approaches with different goals. This shows that the new possibilities for improving simulation software provided by artificial intelligence are manifold. In fields like crowd simulation, where the new methods are not yet applied in products, innovative companies must try different approaches until they find which one works for their specific applications.

Entering a new territory brings new challenges. In our case, this was especially the lack of real-world training data that slowed down the CNN approach. Therefore, we pursued approaches that do not require acquiring such data. Still, as we have seen in the case of reinforcement learning, it is hard to predict how easily training the models leads to the desired results. Machine learning models are not comprehensible in the same way as other types of simulation code and applying them to new use cases often includes a fair amount of trial-and-error.

This trial-and-error process in transferring scientific insights to practical applications poses a big risk for small companies like ours. From a business point of view, we are bound to invest our limited resource in such a way that it directly affects our revenue. Appointing resources to experimental projects with uncertain outcomes therefore requires external funding. The ACSAI program was crucial for us in pursuing the realization of the AI approaches.