TaiCrowd: A High-Performance Simulation Framework for Massive Crowd

Xiaoyu Guan Yihao Li Tianyu Huang*
School of Computer Science, Beijing Institute of Technology, Beijing, CN

Abstract

Massive crowd simulation offers valuable insights into human behavior, with applications in urban planning, public safety, computer animation, and beyond. However, traditional methods that treat crowds as individual agents face significant scalability challenges as crowd sizes grow, resulting in performance bottlenecks. While there are advanced animation engines capable of supporting the efficient rendering of large-scale crowds, existing crowd simulation frameworks often face limitations in scalability and flexibility. In response to these challenges, we present TaiCrowd, an open-source crowd simulation framework. TaiCrowd employs a generic parallel approach specifically designed for agent-based methods, filling the current gap for a scalable, efficient, and user-friendly crowd simulation framework. Experiment results show that TaiCrowd achieves a substantial 60-fold improvement for crowd simulations involving hundred thousand individuals compared to other simulation frameworks, outperforming existing frameworks and enabling real-time simulation for large-scale crowd management. TaiCrowd can be accessed from *https://github.com/Worter623/TaiCrowd*.

Keywords: Crowd simulation, High-performance computation, Open-source framework, Physics-based computation

1. Introduction

Massive crowd simulation not only provides insights into human behavior but also facilitates the creation of realistic virtual environments, which are critical for applications ranging from emergency evacuation planning to immersive gaming experiences[18]. However, traditional simulation methods face efficiency constraints as crowd scales increase. This limitation stems from the conventional treatment of crowds as collections of individual agents, a methodology that excels in capturing heterogeneous crowd dynamics but faces challenges in scalability[17, 29]. The computational demands imposed by massive crowds can strain existing frameworks, hindering the seamless integration of different simulation methods across diverse platforms. While advanced animation engines like Unreal Engine 5 and Unity offer powerful tools for rendering large-scale crowds, there remains a critical gap: the lack of an integrated, scalable, and user-friendly crowd simulation framework that seamlessly complements these rendering platforms. To address these challenges, we propose TaiCrowd, aims to bridge the gap between advanced rendering engines and scalable crowd simulation methods, offering a practical and high-performance solution for massive crowd simulations.

Taichi[10] is a programming language that excels in handling parallel computation, with sparse data structures, ideal for scenarios like massive crowd simulations with sparse spatial configurations. Through our dedicated design of system architecture and data structures, TaiCrowd aims to meet the urgent demand for a scalable, efficient, and user-friendly crowd simulation framework. The main contributions of this paper are as follows:

- We introduce a generic parallel approach for high-performance massive crowd simulation. This approach achieves 60-fold improvement for agent-based crowd simulations with hundred thousand individuals.
- We propose TaiCrowd, an open-source crowd simulation framework. TaiCrowd not only caters to performance requirements but also provides a user-friendly solution for massive crowd simulation.
- We showcase the TaiCrowd's capability and identify its optimal configurations for high-performance massive crowd simulation on limited hardware, setting the stage for future advancements.

The remainder of this paper is structured as follows: In section 2, we discuss previous work on crowd simulation and Taichi. We analyze existing crowd simulation frameworks, identifying their characteristics and limitations. In section 3, we describe details of our simulation framework TaiCrowd. In section 4, we evaluate TaiCrowd's capabilities. In section 5, we review our work and directions for future research.

*corresponding author, Email: huangtianyu@bit.edu.cn

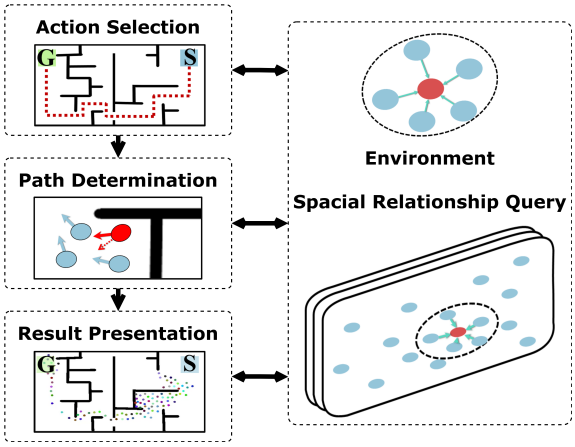

Figure 1. An abstraction of crowd simulation based on three operational layers.

2. RELATED WORK

2.1. Simulating crowds

Generally, the crowd simulation problem can be decomposed into three layers as illustrated in Figure 1: action selection, path determination, and result presentation. Firstly, specify initial positions, goals, and schedules, computing a base plan to reach goals under specified constraints. Secondly, adapt the base plan to unforeseen scenarios arising from the dynamic environment and individual characteristics. This process, facilitated by the locomotion model, addresses collisions and integrates local interactions, ensuring safety and smoothness. Finally, visualize or analyze the simulation results. Each operational layer requires interaction with the simulation environment, conducting spatial relationship queries during each simulation frame to obtain neighborhood information for agents.

This architecture is not a novel abstraction; instead, it has gained widespread adoption and substantial representation in previous works, solidifying its status as a classic model for agent-based crowd simulation[17, 20]. While certain techniques diverge in their approach to simulating massive crowds, such as Hughes[12] employing continuum theory and Treuille et al.[26] formulating crowds with dynamical potential fields, they tend to treat crowds as a cohesive, continuous entity. Crowd movement regulated by potential fields or fluid dynamics, etc., often overlook underlying details and individual-level interactions among agents and environment[29, 27]. In contrast, the agent-based method stands out for its ability to simulate diverse crowd behaviors, showcasing individual-level interactions and crowd heterogeneity[15, 19, 4]. However, as the size of the crowd increases, the computational demand grows linearly, impacting the efficiency of the agent-based method. Parallel computing can distribute the simulation calculations for each agent across different computing hardware, facilitating massive crowd simulations[30, 24].

2.2. Existing crowd simulation frameworks

Extensive research has been conducted on crowd simulation frameworks. Proprietary simulation frameworks like AutoPed[21] and commercial software such as PTV Viswalk hinder detailed analysis and the integration of new methodologies. Table 1 lists major open-source crowd simulation frameworks developed in recent years. However, these frameworks do not fully meet the concurrent requirements of scalability, efficiency, and user-friendliness.

Both FDS[16] and JuPedSim[28] focus on microscopic crowd dynamics within the realm of evacuation scenarios, potentially limiting their broader applicability and extensibility. PedsimRos[25], while adaptable to various scenarios, faces constraints in extensibility as it solely supports the social force model[8]. Addressing scalability concerns, Vadere[14] aims to compare and validate locomotion models, while SocNavBench[2] focuses on providing real-world pedestrian data and a suite of metrics for evaluating navigation algorithms. Both prioritize facilitating experimentation and analysis of new methodologies rather than high-performance crowd simulation, diverging from the emphasis of this paper.

In alignment with the objectives of our work, SteerSuite[23], ADAPT[22], Menge[5, 6], and MomenTUM[13] provide a comprehensive framework for dynamically simulating a wide range of scenarios and behaviors. ADAPT provides narrative scene configuration through Unity plugins, however, is tightly coupled with Unity and exhibits efficiency issues, supports approximately 150 agents with full fidelity at interactive frame rates. The rest frameworks only provide simple GUI and I/O APIs, requiring the specification of scene information using configuration files such as XML files or WKT files, which can be bothering for massive crowd simulation. None of these frameworks supports directly importing an overhead view picture as a simulation scene, hindering user-friendliness.

2.3. Taichi: high-performance computing language

Taichi is a Python-based domain-specific language designed for high-performance parallel computation. Its just-in-time compilation and support for sparse data structures make it particularly suited for TaiCrowd's architecture. Leveraging Taichi's strengths, TaiCrowd utilizes hierarchical data structures and sparse spatial configurations to enable scalable simulations on modest hardware.

Building on Taichi's success in domains such as particle simulation and soft robotics[9, 3, 11], TaiCrowd addresses a critical gap by providing a scalable, efficient, and user-friendly crowd simulation framework.

Table 1. Open-source crowd simulation frameworks in recent years. The statistics for "Latest Update" are based on the last update time of the official code download source and the latest reply time on the GitHub repository's issues accurate as of February 2025. "Conf." is short for configuration files.

Simulator	Language	Latest Update	Locomotion Model			Input / Output API		
			Rule	Force	Speed	Conf.	Picture	CSV
SteerSuite[23]	C++	2019	×	✓	✓	✓	×	×
ADAPT[22]	C++,C#	2021	×	✓	×	✓	✓	✓
SocNavBench[2]	Python	2023	×	✓	✓	✓	✓	×
MomenTUM[13]	Java	2023	✓	✓	×	✓	×	×
Menge[5, 6]	C++	2023	×	✓	✓	✓	×	×
Vadere[14]	Java	2023	✓	✓	✓	✓	✓	×
PedsimRos[25]	C++	2024	×	✓	×	✓	×	×
FDS[16]	Fortran	2025	×	✓	×	✓	×	✓
JuPedSim[28]	C++, Python	2025	×	✓	✓	✓	×	×
TaiCrowd	Taichi	2025	✓	✓	✓	✓	✓	✓

3. TAICROWD'S ARCHITECTURE

This section will delve into the details of TaiCrowd. In Section 3.1, we will demonstrate how TaiCrowd implements the proposed generic agent-based parallel architecture through several typical agent-based methods. Section 3.2 will expound on the generic parallel approach tailored for agent-based simulation methods. Finally, Section 3.3 will elucidate the implementation of each module in TaiCrowd.

3.1. Architecture design

Traditional agent-based crowd simulation methods face significant performance bottlenecks as the size of the crowd grows. While advanced animation engines excel in rendering large-scale crowds, there lack an scalable, efficient, and user-friendly simulation framework for integration. TaiCrowd addresses these challenges by implementing a generic parallel approach specifically designed for agent-based methods. This approach, outlined in Algorithm 1, forms the basis for achieving high-performance parallelization in agent-based crowd simulation methods. The bold content in the algorithm signifies innovations distinguishing it from the conventional agent-based crowd simulation paradigm.

Figure 2 illustrates how TaiCrowd aligns agent-based methods with the parallel paradigm defined in Algorithm 1. TaiCrowd has integrated several prominent agent-based simulation models as both demonstrations and representatives. These models encompass the steering model for rule-based crowd simulation, the social force model for force-based crowd simulation, and the ORCA (Optimal Reciprocal Collision Avoidance) model[1] for crowd simulation based on velocity space.

Algorithm 1: Parallel Agent-based Crowd Simulation Method

```
ScenarioInitialization()
while not reached specified simulation time do
    do in parallel for each agent
        StateMachineUpdate()
        DesiredVelocityComputation()
        NeighborhoodInformationUpdate()
        FeasibleVelocityComputation()
        AgentPositionUpdate()
    Visualization()
```

TaiCrowd's computation pipeline is demonstrated in Figure 3. In addition to TaiCrowd's pre-implemented modules designed for immediate utilization, we facilitate the effortless substitution of user-defined implementations to promote ease in experimentation and comparison.

In addition to its utilization of the Taichi language for parallel computation, TaiCrowd embodies a paradigm shift in crowd simulation through a series of innovative architecture design, distinguishing TaiCrowd from mere implementations of existing methodologies:

- **State machine architecture** enables nuanced behavioral transitions, allowing simulated individuals to dynamically respond to diverse environmental stimuli with realistic and contextually appropriate actions.

- **Optimized neighborhood search mechanism** facilitates swift and accurate interaction calculations among individuals, enhancing simulation efficiency without compromising accuracy.

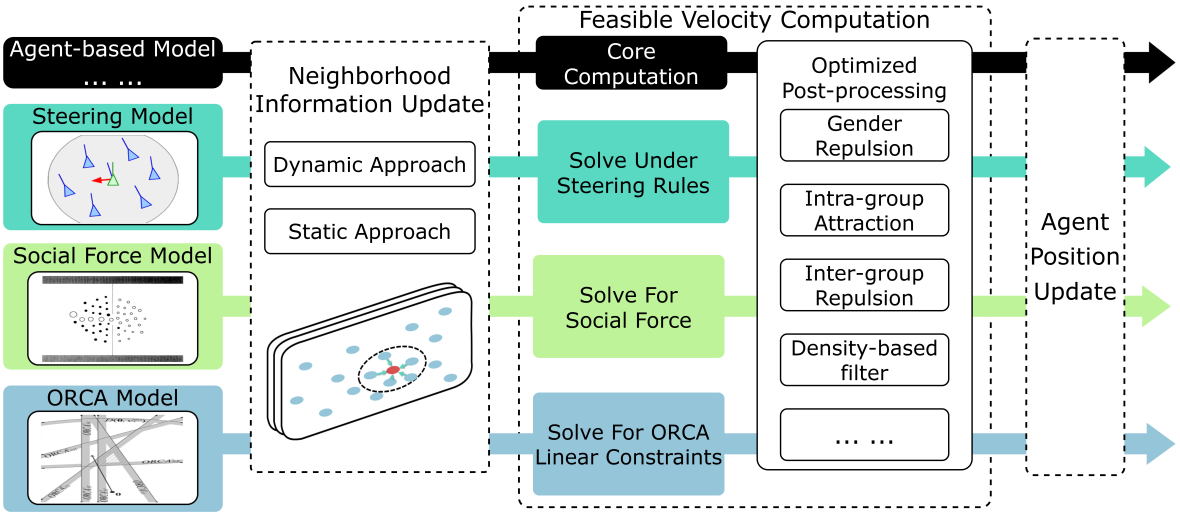

Figure 2. Integration of agent-based methods in TaiCrowd's parallel architecture.

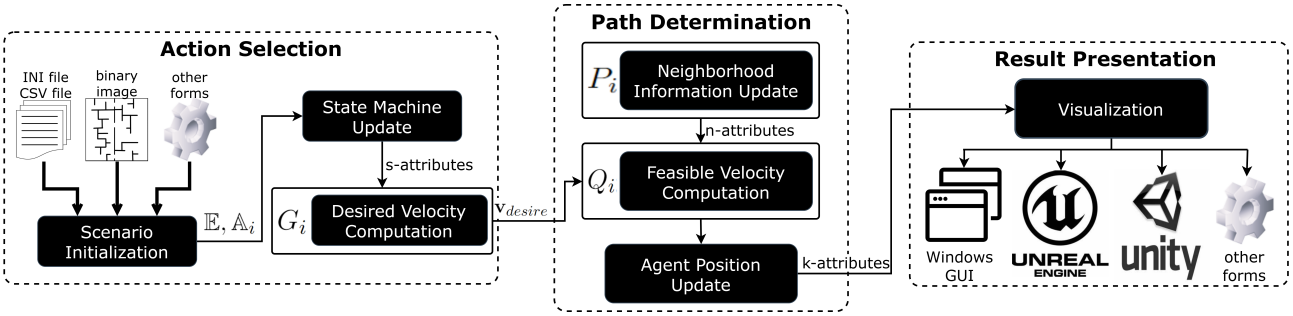

Figure 3. Computation pipeline of TaiCrowd. Modules are shown in black boxes. White dotted boxes show how TaiCrowd's modules correspond to the three abstraction layers.

- **Optimized post-processing** supports advanced effects, such as grouping dynamics, personality-specific behaviors, and emotional contagion, without hindering simulation efficiency.
- **User-friendly input-output interfaces** provide intuitive controls, fostering ease of use and customization for users of varying expertise levels.

Through these advancements, TaiCrowd stands as a testament to innovation in crowd simulation, offering a holistic solution that addresses the evolving needs of simulation practitioners across various domains.

3.2. Formalized definition

To facilitate parallel computation in agent-based methods, the overall process of crowd simulation is defined as iteratively solving an Initial Value Problem at each simulation frame:

$$\dot{\mathbf{x}}_i(t) = \mathbf{v}_i(t) = \mathbf{F}_i(t, \mathbb{E}, \mathbb{A}_i(t)), \tag{1}$$

where $\mathbf{x}_i(t), \mathbf{v}_i(t)$ represents the position and velocity of agent i at time t, $\mathbb{A}_i(t)$ denotes the individual attributes of agent i at time t, and $\mathbb{E}$ denotes the static simulation environment information, like obstacle positions and map size. Align with the abstraction in section 2.1, the key computation function $\mathbf{F}_i$ is decomposed into three conceptual functions as illustrated in Equation 2. It is noteworthy that the functions G_i, P_i, Q_i are designed to be executed by each agent.

$$F_i(t+1) = Q_i(t, \mathbb{E}, P_i(t, \mathbb{A}_i), G_i(t, \mathbb{E}, \mathbb{A}_i)). \tag{2}$$

The subsequent content will provide a detailed exploration of the taxonomy of $\mathbb{A}_i$ and the definitions of G_i, P_i, and Q_i.

$\mathbb{A}_i$ is categorized as vector:

$$\mathbb{A}_i = \begin{bmatrix} \mathbf{x}_i & \mathbf{v}_i & \mathbf{n}_i & \mathbf{s}_i \end{bmatrix}^T, \tag{3}$$

where $\mathbf{x}_i, \mathbf{v}_i \in \mathbb{R}^2$. The vector $\mathbf{k}_i = \begin{bmatrix} \mathbf{x}_i & \mathbf{v}_i \end{bmatrix}^T$ is referred to as the key attributes, or **k-attributes**. TaiCrowd currently assumes that crowd simulation is performed in a two-dimensional space. This simplification is sufficient for most applications. $\mathbf{n}_i$ represents the dynamic neighborhood information, such as the relative positional details of other

agents around agent i:

$$\mathbf{n}_i = \{\bigcup_j \begin{bmatrix}\mathbf{x}_j & \mathbf{v}_j\end{bmatrix} \mid j \in \mathbb{C}_i\}, \tag{4}$$

where $\mathbb{C}_i$ represents the risk disk of agent i. $\mathbf{n}_i$ is referred to as the neighborhood attributes, or **n-attributes**. The remaining attributes, encompassing the agent's heterogeneous characteristics such as goals, constraints, social identity, and emotions, are equally crucial as they collectively contribute to generating more realistic and diverse simulation effects. This part of agent attributes is referred to as the supplementary attributes, or **s-attributes**.

Defining G_i, P_i, and Q_i is equivalent to elucidating the computational process of crowd simulation, which will be presented according to the three hierarchical layers outlined in Figure 1.

In the action selection layer, the computation function G_i calculates the instantaneous desired velocity for agent i based on time, the agent's current position, and its goal:

$$G_i : t, \mathbb{E}, \mathbb{A}_i \rightarrow \mathbf{v}_{desire}. \tag{5}$$

In the path determination layer, we innovatively designed this layer to consist of two functions to fully isolate the computation pipeline for each agent for high-performance parallelism. Firstly, we define

$$P_i : t, \mathbb{A} \mapsto \mathbb{A}'_i, \tag{6}$$

where $\mathbb{A}'$ represents the vector updated through the function P_i. To be more specific, P_i updates the n-attributes for agent i based on time and k-attributes of other agents, formulated as

$$\mathbf{n}_i(t+1) = P_i(\mathbb{A}(t)). \tag{7}$$

Secondly, we define

$$Q_i : t, \mathbb{E}, \mathbb{A}'_i, \mathbf{v}_{desire} \rightarrow \mathbf{v}_{feasible}. \tag{8}$$

Function Q_i adapts the ideal velocity $\mathbf{v}_{desire}$ for agent i based on its state at time t, and outputs a feasible velocity, which equals to $\mathbf{v}_i(t)$ in Equation 1.

In the result presentation layer, we dynamically update visualization results or outputs $\mathbb{A}$ to file for subsequent data analysis, according to user requirements.

As $\mathbb{E}$ represents static and unchanging environmental information, and considering that the k-attributes for each agent remain constant until the result presentation layer, the only variable being modified throughout the entire crowd simulation task is $\mathbb{A}_i$. Assuming $\mathbb{E}$ as an implicit parameter, the key function F_i can be formulated as the composition of G_i, P_i, Q_i as in Equation 2. The position of agent i at each frame is then determined by solving for $\mathbf{x}_i(t)$.

3.3. Modules implementation

Scenario initialization module INI files can be used to globally configure TaiCrowd, enabling or disabling specific modules, setting the simulation timestep, and choosing from TaiCrowd's native provided locomotion models. For specifying the simulation environment, TaiCrowd offers multiple implementations. This includes creating simple obstacle walls by specifying the start and end points of obstacle segments, generating convex obstacles with a vertex list, and inputting binary pictures for more intricate simulation environments such as mazes. To specify the initial conditions of agents, options include providing an explicit list of each agent's position, velocity, and target, setting k-attributes through two-dimensional numpy arrays, or importing a CSV file containing such information. To facilitate comparisons between simulations, TaiCrowd natively includes several optional predefined classical simulation scenarios.

State machine update module TaiCrowd offers this module as an optional feature to simplify the description of crowd behavior by dividing intricate motion scenarios into a finite set of states and transition conditions. TaiCrowd allows users to define arbitrary states and transition conditions. In many cases, the basic states are already self-consistently implemented in the feasible velocity computation module with mature locomotion models. In such cases, the state machine module can be used to define extended states or specify a series of distinct targets.

Desired velocity computation module In the desired velocity computation module, TaiCrowd aims to support the paradigm of global path planning plus local navigation in crowd simulation. In cases where global path planning is not required, the functionality of this module is simply to calculate the vector from the current position of each agent to its target. TaiCrowd incorporates the global path planning algorithm mainly based on A*[7], preventing short-sighted congestion in crowd movement. We enhance the performance by parallelizing the steps of "extracting the highest-priority node q from the open list" and "expanding the neighboring nodes of q".

Neighborhood information update module TaiCrowd employs neighborhood information update approaches which can be categorized into dynamic and static methods.

The dynamic approach utilizes two Taichi Dynamic SNode fields, parallelly traversing all agent k-attributes, and recording the number and identifiers of agents within each grid.

The static approach pre-allocates memory for Taichi Pointer SNode fields before program execution and utilizes

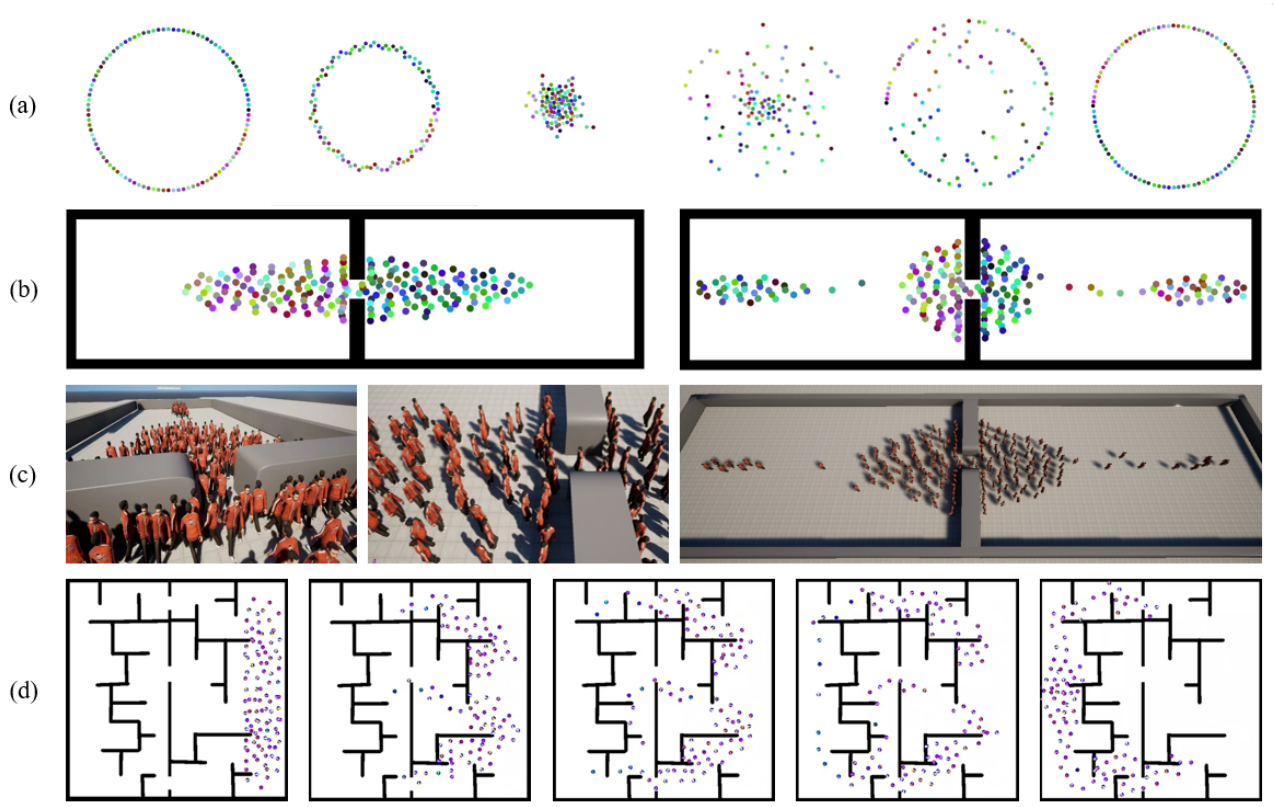

Figure 4. TaiCrowd simulation scenarios, (a)Scene Circle with windows GUI, (b)Scene Doorway with windows GUI, (c)Scene Doorway with Unreal Engine, (d)Crowd evacuation scenarios at major events in China with Unreal Engine.

parallel prefix sum algorithms to compute neighborhood information. This method capitalizes on a crucial property: each agent at any given moment belongs to only one grid, ensuring that the sum of agents contained in all grids is equal to the total number of agents in the crowd. By employing a Taichi field of the same size as the total number of agents, this method records the identifiers of agents within each grid in row-major order. Calculating the starting and ending positions of agents corresponding to each grid in this field reveals the number of agents and their corresponding identifiers within each grid. This approach achieves efficient computations on GPU though sacrificing some memory space.

Feasible velocity computation module The feasible velocity computation module computes a feasible velocity that maximally satisfies the desired velocity while adhering to the rules dictated by the current state. Users can utilize the pre-loaded representative crowd simulation models in TaiCrowd, as shown in Figure 2, or develop custom models using the Taichi language. This flexibility enables research into how changes in locomotion models affect overall simulation.

Agent position update module The agent position update module is responsible for parallelly solving $\mathbf{x}_i$ based on Equation 1.

Visualization module The visualization module is capable of graphically representing, exporting, or analyzing the spatial dynamics of the crowd in each simulation frame. TaiCrowd provides real-time simulation visualization and supports data-driven crowd simulation with professional rendering engines using exported data.

TaiCrowd predefines various simulation evaluation metrics, primarily categorized into two categories. The first category evaluates the realism and feasibility of the simulation, including metrics such as the count of pedestrian collisions, arrival rates, and average travel time. The second category evaluates the performance of simulation, encompassing metrics such as simulation time, and memory usage. TaiCrowd supports extensibility for personalized customization of evaluation metrics based on specific analytical requirements.

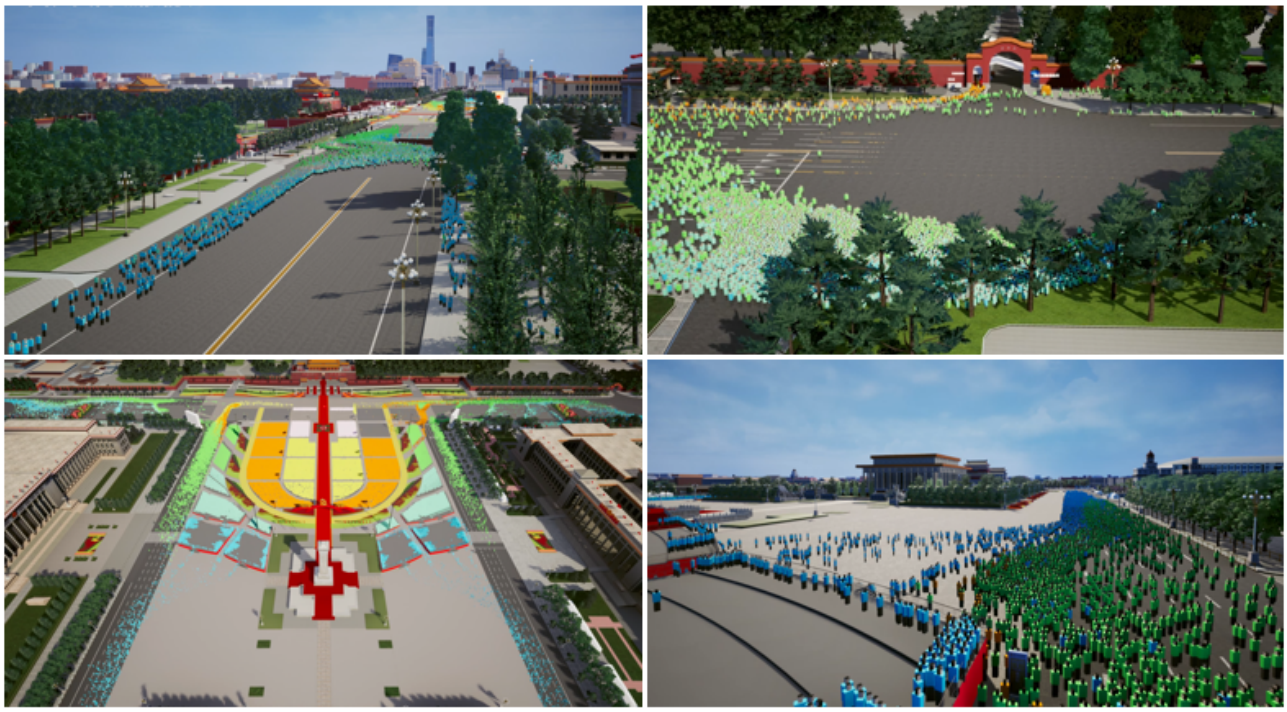

Figure 5. Crowd evacuation scenarios at major events in China with TaiCrowd.

4. APPLICATION AND EVALUATION

4.1. Experiment scenario

The performance data were measured on a standard desktop PC with GPU RTX 3070, an Intel i7 3.4GHz processor with 32GB of memory.

In our experiments, for environment representation, both rule-based and force-based feasible velocity computation models utilized the common grid-based approach. The obstacle information for each grid is stored in the two-dimensional Taichi field, as illustrated in Figure 6. For models based on velocity space, an obstacle structure is defined to record information for each obstacle vertex in the two-dimensional plane. This information is then stored in Taichi Struct fields.

Concerning the representation of crowd attributes, we employ the AOS (Array of Structures) approach to tightly organize each agent's k-attributes and s-attributes within the Taichi Dense SNode. As for storing the n-attributes of the crowd, we utilize the Taichi Dynamic SNode, which facilitates dynamic addition and removal of elements similar to a C++ vector.

4.2. Simulation results

TaiCrowd facilitates real-time visualization of simulation results directly through the Taichi GUI. We employ coloured disks to signify different types of crowds. The angle of rotation of the indicative triangle in the center of

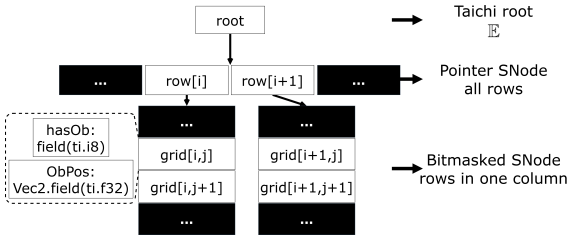

Figure 6. TaiCrowd's data representation approach, white box indicates active nodes, $grid[i, j]$ represents the grid at row i and column j.

the circle indicates the direction of the force at the current moment. The RGB color value of the circle represents the magnitude of the force. Figure 4 provides several simulation example.

For more precise control over crowd motion animations, one can utilize the data-driven crowd simulation approach with professional rendering engines, using the crowd k-attributes data exported from TaiCrowd. Figure 5 illustrates a practical application of TaiCrowd in evacuation scenario.

TaiCrowd empowers users to define complex individual behaviors with ease, while maintaining high performance. We illustrate this capability by incorporating personality-specific behavior and grouping dynamics within TaiCrowd.

We refer the readers to our Github repository to view the complete simulation results.

- **Scene doorway** In Scene Doorway, 200 agents are po-

Table 2. Likert scale ratings (1–5) for Realism, Dynamic Coherence, and Applicability by Experts Across Scenarios."A." represents the average score given by the three experts for each scenario.

Metric	Scene Doorway				Evacuation Scenario				Grouping Dynamics				Metric Average
	E1	E2	E3	A.	E1	E2	E3	A.	E1	E2	E3	A.	
Realism	4	4	3	**3.67**	5	5	4	**4.67**	4	5	4	**4.33**	**4.22**
Dynamic Coherence	4	4	3	**3.67**	4	5	4	**4.33**	3	4	4	**3.67**	**3.89**
Applicability	4	4	4	**4.00**	5	5	5	**5.00**	4	5	4	**4.33**	**4.44**

sitioned on either side of a narrow and long corridor, to traverse through a narrow door to the opposite side. In this scenario, the noticeable queue blockages near the narrow doorway create challenges for obstacle avoidance and interactions among dense agents.

- **Scene circle** In Scene Circle, 200 agents form a circle, aiming to exchange positions with agent along the diagonal. This scene involves no obstacles, and the crowd density becomes extremely high when people exchange positions in the middle of the circle, requiring complex collision detection among agents.
- **Scene maze** In Scene Maze, 100 agents are initialized in the lower right corner of a complicated maze, and their goal is to get through the maze to the exit on the left. This scenario shows that TaiCrowd is capable of handling global navigation and environmental obstacles at a high computational speed.
- **Evacuation scenario** This scenario serves as a case study for using TaiCrowd to simulate and compute evacuation plans for large-scale events involving nearly 70,000 spectators. The simulation precisely calculates the optimal exit routes and times for the crowd, with the results visually rendered in Unreal Engine, demonstrating TaiCrowd's effectiveness.
- **Personality-specific behavior** This scenario involves 100 agents forming a 10x10 queue and moving towards a destination on the right side of an open space. Each agent is assigned a unique gender, personality, and physical fitness level at random. Male agents are represented by blue disks, while female agents are depicted in green. The depth of color indicates the agent's speed capability. This demonstrates TaiCrowd's ability to simulate individual behaviors and interactions within a structured group with little computational cost.
- **Grouping dynamics** In this scenario, 60 agents traverse through Scene Maze, with several subgroups evident within the crowd. Agents within the same subgroup, connected by lines of the same color, exhibit a tendency to move more closely together. This showcases TaiCrowd's ability to simulate grouping dynamics and effortlessly analyze the influence of social groups on crowd behavior.

User study To evaluate the authenticity of TaiCrowd's simulation results, we conducted a user study with three domain experts in crowd simulation. The study focused on three core scenarios, with each expert assessing them on three dimensions using a 1–5 Likert scale, along with optional open-ended feedback. The full questionnaire is available in our source code.

The Likert scale results as shown in Table 2 demonstrated a strong performance by TaiCrowd across all dimensions. The highest score, 4.44/5, was achieved for applicability among three metrics, affirming TaiCrowd's potential for real-world applications regarding large-scale crowd simulation. Experts praised TaiCrowd's scalability and quality across scenarios while suggesting improvements in path optimization under dense crowds and adding new use cases. The study confirms TaiCrowd's authenticity and practical potential, with future evaluations requiring broader user studies.

4.3. Performance evaluation

We compare TaiCrowd with several actively maintained and recently updated open-source frameworks in alignment with the objectives of our work, including Menge, JuPedSim, and Vadere, in terms of efficiency improvements across different simulated crowd scales. The feasible velocity is computed with the force-based method. All performance statistics are the average computation time over 20 tests. The results are shown in Table 3 and Figure 7.

Regardless of simulation frameworks, there is a slight decrease in Scene Doorway's FPS performance, reflecting the additional computational load incurred by the queue blockage and collision with static obstacles. In both scenarios, TaiCrowd has enhanced crowd simulation efficiency, showing even more remarkable effectiveness as the simulated crowd size exceeds 5,000. In general, TaiCrowd outperforms the comparison frameworks across various crowd scales, achieving a 5-fold improvement in the thousand-individual tier, a 30-fold enhancement in the ten-thousand-

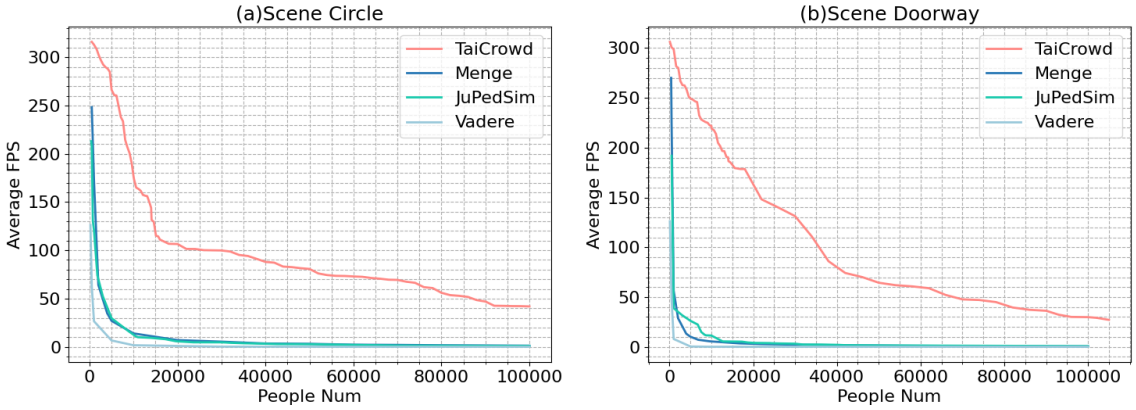

Figure 7. Efficiency comparison between TaiCrowd and other frameworks.

Table 3. Performance statistics(ms) of TaiCrowd and other frameworks in different simulation scenarios. Bold indicates better results. A "-" sign indicates that the value of this item is bigger than 10,000 and is not collected.

Agent	Scene Circle-Figure 4(a)							
Num	0.5k	1k	5k	10k	30k	50k	70k	100k
Vadere	15.87	37.04	151.52	625.0	5000.0	-	-	-
JuPedSim	4.81	8.26	33.33	76.92	222.22	370.37	769.23	1111.11
Menge	4.03	5.78	37.04	71.43	185.19	312.5	500.0	833.33
TaiCrowd	**3.16**	**3.19**	**3.76**	**5.65**	**10**	**12.35**	**14.49**	**23.81**
Agent	Scene Doorway-Figure 4(b)(c)							
Num	0.5k	1k	5k	10k	30k	50k	70k	100k
Vadere	20.83	126.58	2500.0	-	-	-	-	-
JuPedSim	5.21	25.64	38.46	90.91	312.5	666.67	833.33	1250.0
Menge	4.44	17.86	100.0	181.82	434.78	714.29	1111.11	1666.67
TaiCrowd	**3.32**	**3.34**	**4.02**	**4.5**	**7.63**	**15.63**	**20.83**	**33.33**

individual tier, and a remarkable 60-fold improvement in the hundred-thousand-individual tier.

The exceptional performance highlighted above positions TaiCrowd as an efficient framework for massive crowd simulations, emphasizing its robust crowd simulation capabilities. Moreover, as shown in Table 1, TaiCrowd provides various feasible velocity computation models, I/O APIs, and optional modules, surpassing other simulation frameworks in scalability and user-friendliness. Considering the aforementioned aspects, TaiCrowd can be characterized as a scalable, efficient, and user-friendly crowd simulation framework, facilitating researchers in analyzing simulation outcomes and algorithm performance.

4.4. Performance analysis

We analyze the performance improvements of TaiCrowd by focusing on three key modifications that enable better parallelization and enhanced integration with Taichi:

- **Parallelization of agent-based computation** TaiCrowd uses fully parallelized agent-based computations, allowing agents to update their states simultaneously without synchronization delays. By leveraging Taichi's parallel loops across CPU and GPU threads, we achieved a significant reduction in simulation time (see Table 3 and Figure 7).

- **Data structures and memory layouts optimization** We designed data structures aligned with a sparse, hierarchical memory layout(see Section 3.2), reducing memory footprint and improving data access efficiency (Figure 6). Custom quantization techniques store attributes efficiently, further enhancing performance.

- **Optimized neighborhood search mechanism** The neighborhood search optimization reduces computational overhead during updates, enabling efficient spatial queries in parallel, which maintains accuracy even

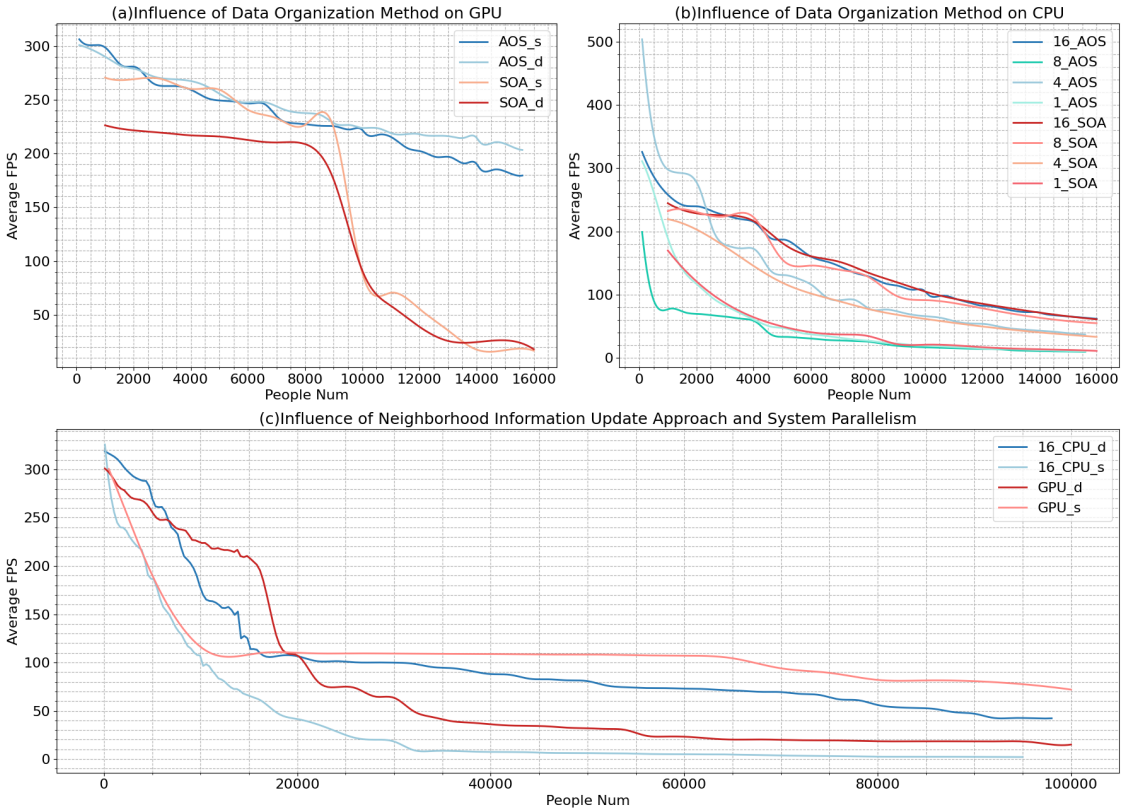

Figure 8. Configuration analysis in TaiCrowd using social force model in Scene Circle. The prefix number represents the number of CPU threads, the suffix "d" represents the dynamic neighborhood information update approach and the suffix "s" represents the static neighborhood information update approach. Figure(c) use the AOS data organization method and dynamic neighborhood information update approach on GPU.

in dense crowd scenarios (Section 3.3).

These improvements are rooted in our system architecture and design, ensuring that TaiCrowd's performance is not language-dependent and can be implemented in other similar programming languages without sacrificing efficiency.

4.5. Configuration analysis

We conducted a series of experiments to identify the optimal configuration for massive crowd simulation using the configurable parameters provided by TaiCrowd.

As depicted in Figure 8(a) and Figure 8(b), the impact of the data organization method on performance is more pronounced on the GPU than on the CPU, owing to the peculiarities of GPU memory. In general, the AOS method enhances cache hit rates, leading to a substantial improvement in simulation efficiency compared to the SOA (Structures of Array) method; the parallelism of GPU surpasses that of multi-threaded CPU.

Figure 8(c) compares the performance of different feasible velocity computation models natively supported by TaiCrowd. TaiCrowd's parallel optimization for various feasible velocity computation models significantly accelerates simulation efficiency.

Based on the aforementioned experimental results, when conducting massive parallel crowd simulations with TaiCrowd, it is suitable to use the GPU, adopt a static neighborhood information update approach, and implement AOS data organization for computation.

5. CONCLUSION

Traditional agent-based crowd simulation excels in capturing diverse crowd dynamics but struggles with scalability. This paper introduces TaiCrowd, an open-source framework addressing the need for a scalable, efficient, and user-friendly crowd simulation solution. Utilizing the general parallel approach we proposed for agent-based methods, TaiCrowd achieves significant performance improvements, with a 5-fold enhancement in the thousand-individual tier, 30-fold in the ten-thousand-individual tier, and an impressive 60-fold improvement in the hundred-thousand-individual tier. Best practices for TaiCrowd include GPU utilization, the static neighborhood information

update approach, and the AOS data organization method. TaiCrowd's evolution holds promise for addressing a wider range of crowd simulation challenges, fostering innovation, and facilitating the integration of diverse modelling approaches.

The future of TaiCrowd involves exploring real-world video comparisons to validate its ability to model real-life crowd behaviors. Beyond agent-based methods, we aim to adapt TaiCrowd's architecture for compatibility with other massive crowd simulation methods, expanding its versatility.

Acknowledgement

This study was supported by the National Science and Technology Major Project under Grant 2022ZD0120204.

References

[1] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed autonomous robotic systems: The 10th international symposium*, pages 203–216. Springer, 2013. 3

[2] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction*, 11:1–24, 2022. 2, 3

[3] Z. Chen, S. Zhang, S. Luo, F. Sun, and B. Fang. Tacchi: A pluggable and low computational cost elastomer deformation simulator for optical tactile sensors. *IEEE Robotics and Automation Letters*, PP:1–8, 03 2023. 3

[4] A. Colas, W. van Toll, K. Zibrek, L. Hoyet, A.-H. Olivier, and J. Pettré. Interaction fields: Intuitive sketch-based steering behaviors for crowd simulation. In *Computer Graphics Forum*, volume 41, pages 521–534. Wiley Online Library, 2022. 2

[5] S. Curtis, A. Best, and D. Manocha. Menge: A modular framework for simulating crowd movement. *Collective Dynamics*, 1:1–40, 2016. 2, 3

[6] M. Diamanti and H. H. Vilhjálmsson. Extending the menge crowd simulation framework: visual authoring in unity. In *Proceedings of the 22nd ACM International Conference on Intelligent Virtual Agents*, pages 1–3, 2022. 2, 3

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022. 6

[8] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 2

[9] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019. 3

[10] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019. 1

[11] Y. Hu, J. Liu, X. Yang, M. Xu, Y. Kuang, W. Xu, Q. Dai, W. T. Freeman, and F. Durand. Quantaichi: a compiler for quantized simulations. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021. 3

[12] R. L. Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002. 2

[13] P. M. Kielar, D. H. Biedermann, and A. Borrmann. Momentumv2: A modular, extensible, and generic agent-based pedestrian behavior simulation framework. In *Computer Science, Engineering*, 2016. 2, 3

[14] B. Kleinmeier, B. Zönnchen, M. Gödel, and G. Köster. Vadere: An open-source simulation framework to promote interdisciplinary understanding. *arXiv preprint arXiv:1907.09520*, 2019. 2, 3

[15] H. Kolivand, M. S. Rahim, M. S. Sunar, A. Z. A. Fata, and C. Wren. An integration of enhanced social force and crowd control models for high-density crowd simulation. *Neural Computing and Applications*, 33:6095–6117, 2021. 2

[16] T. Korhonen, S. Hostikka, S. Heliövaara, H. Ehtamo, and K. Matikainen. Integration of an agent based evacuation simulation and the state-of-the-art fire simulation. In *Proceedings of the 7th Asia-Oceania symposium on fire science & technology*, pages 20–22, 2007. 2, 3

[17] M. Lemonari, R. Blanco, P. Charalambous, N. Pelechano, M. Avraamides, J. Pettré, and Y. Chrysanthou. Authoring virtual crowds: A survey. In *Computer Graphics Forum*, volume 41–2, pages 677–701. Wiley Online Library, 2022. 1, 2

[18] Y. Li, T. Huang, Y. Liu, and G. Ding. A spatio-temporal hierarchical model for crowd formation planning in large-scale performance. *IEEE Access*, 8:116685–116694, 2020. 1

[19] L. Luo, C. Chai, J. Ma, S. Zhou, and W. Cai. Proactive-crowd: Modelling proactive steering behaviours for agent-based crowd simulation. In *Computer Graphics Forum*, volume 37, pages 375–388. Wiley Online Library, 2018. 2

[20] C. W. Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, pages 763–782. Citeseer, 1999. 2

[21] W. Shao and D. Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH Eurographics symposium on Computer animation*, pages 19–28, 2005. 2

[22] A. Shoulson, N. Marshak, M. Kapadia, and N. I. Badler. Adapt: the agent development and prototyping testbed. In *Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games*, pages 9–18, 2013. 2, 3

[23] S. Singh, M. Kapadia, P. Faloutsos, and G. Reinman. An open framework for developing, evaluating, and sharing steering algorithms. In *Motion in Games: Second International Workshop, MIG 2009, Zeist, The Netherlands, November 21-24, 2009. Proceedings 2*, pages 158–169, 2009. 2, 3

[24] J. Skrzypczak and P. Czarnul. Efficient parallel implementation of crowd simulation using a hybrid cpu+ gpu high performance computing system. *Simulation Modelling Practice and Theory*, 123:102691, 2023. 2

[25] Social Robotics Lab, University of Freiburg. Pedsim ros. Last accessed on 2024-02-01. 2, 3

[26] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3):1160–1168, 2006. 2

[27] W. van Toll, T. Chatagnon, C. Braga, B. Solenthaler, and J. Pettré. Sph crowds: Agent-based crowd simulation up to extreme densities using fluid dynamics. *Computers & Graphics*, 98:306–321, 2021. 2

[28] A. K. Wagoum, M. Chraibi, J. Zhang, and G. Lämmel. Jupedsim: an open framework for simulating and analyzing the dynamics of pedestrians. In *3rd Conference of Transportation Research Group of India*, volume 12, 2015. 2, 3

[29] S. Yang, T. Li, X. Gong, B. Peng, and J. Hu. A review on crowd simulation and modeling. *Graphical Models*, 111:101081, 2020. 1, 2

[30] X. Zhao, L. Lyu, C. Lyu, and C. Ji. A new parallel simulation method for massive crowd. *Procedia computer science*, 147:283–287, 2019. 2