IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 October 2024, accepted 11 November 2024, date of publication 18 November 2024,
date of current version 29 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3501093

RESEARCH ARTICLE

Massive Crowd Simulation With Parallel Computing on GPU

VINCENZO LOMBARDO, DAVIDE GADIA, AND DARIO MAGGIORINI
Department of Computer Science, University of Milan, 20133 Milan, Italy
Corresponding author: Davide Gadia (davide.gadia@unimi.it)

ABSTRACT The ability to simulate realistic crowds is a highly sought-after capability in the fields of entertainment (video games, movies), urban planning and evacuation simulations. Traditional approaches to crowd simulation rely on heavy Central Processing Unit (CPU) computation. This approach has limitations in terms of scalability and performance, which are solvable with the use of Graphics Programming Units (GPUs) and parallel computing techniques. In fact, the development of Compute Shaders on GPU allows the execution of general-purpose operations alongside traditional rendering tasks within real-time applications. This paper aims to contribute to the current literature on crowd simulation methods by developing a real-time simulation model that integrates and expands several techniques from literature, adapted and optimized to exploit GPU computing capabilities. The proposed model incorporates continuous representations for crowds in order to simulate human movement and decision-making. The achieved results demonstrate a high level of scalability and efficiency. The implemented techniques and optimizations allow the model to handle a significant number of agents while maintaining real-time performances to achieve reduced simulation time and good user experience. Stress tests showcase that the proposed model significantly outperforms other macroscopic models, maintaining a stable frame rate of 60 FPS when simulating 20,000 agents even on mid-range systems intended for personal use.

INDEX TERMS Crowd simulation, GPU computing, video games, real-time.

I. INTRODUCTION

The field of crowd simulation offers a multitude of potential applications, ranging from the real-time rendering of agents in video games to optimization of evacuation plans during building design.

Achieving high levels of realism and complexity in crowd behavior necessitates the incorporation of a substantial number of individual agents within the simulation. However, despite its potential benefits, research focusing on large-scale crowd simulations remains limited, with the majority of projects favoring smaller-scale scenarios that employ a greater number of rules and intricate individual behaviors.

Current approaches and models for crowd simulation predominantly rely on CPU computation. While multi-threading offers some improvement, inherent limitations in processing units and scheduling often hamper the simulation of exceptionally large crowds. Several attempts have been made to push these boundaries. While some have succeeded, they often involve undesirable compromises, such as requiring the large computational power of distributed systems or unconventional and customized hardware. In the last years, hardware advancements have led to a terrific increase in GPU parallel computation and programmability. One of the first fields to adopt these powerful approaches was physics simulation, which involves complex differential equations, notoriously difficult to be handled efficiently by the CPU. Furthermore, as GPU computing power continues to rise, the ability to perform general-purpose operations alongside traditional rendering tasks has become a reality. This has spurred the development of parallel alternatives of existing algorithms, through the use of Compute Shaders, not only within the video game industry, but also in heavy computational tasks in fields such as crowd simulation.

The associate editor coordinating the review of this manuscript and approving it for publication was Jesus Felez.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

Since its inception, crowd simulation has flourished with extensive research endeavors. Particularly intriguing is the investigation of human group behavior across diverse environments, incorporating psychological factors such as perceived danger, and exploring varying intra-group and inter-group dynamics. While these elements elevate the realism of simulations, they often result in computationally intensive models capable of representing only a limited number of individuals, thus hindering the observation of emergent behaviors within larger crowds. The design of a crowd simulation model entails careful consideration of several key features. Their combined configuration ultimately determines the nature and outcome of the simulation. These features can be conceptualized as distinct modules that, when integrated, contribute to the comprehensiveness and realism of a model based on observed human behavior. However, it is crucial to recognize that the computational complexity associated with a fully comprehensive model necessitates thoughtful weighting of each feature in accordance with its purpose. We define a complete crowd simulation model as the combination of:

- Global planner: it proposes the high-level direction that each agent must follow to reach its goal. Due to the complexity of the algorithms, the technique has limited access to environmental information.
- Local obstacles avoidance: it corrects the direction toward goals proposed by the global planner, in order to avoid collisions between agents and obstacles.
- Behavioral modeling: it applies the logic about the behavior, quirks and subjective elements that can influence the motion of individuals.
- Motion synthesis: it animates the assets of the agents according to the final movement obtained.

This paper presents a novel crowd simulation model implemented entirely on a GPU. The model prioritizes two key features: global planning and collision avoidance. Additionally, it leverages parallel computing to incorporate per-agent behavioral modeling. Individual agents navigate towards their goals based on calculated optimal paths, while also accounting for unique characteristics and fine-grained interactions with neighboring agents. This work emphasizes the advancements achievable in crowd simulation by combining two established macroscopic techniques: Potential Fields [1] and Aggregate Dynamics [2]. Both techniques are proposed with optimized and efficient GPU-based solutions. This approach paves the way for real-time simulation models that seamlessly integrate large-scale crowd behaviors with individual agent actions. Additionally, many of the approaches employed in the proposed model make use of continuous equations to describe crowd motion rules, drawing inspiration from fluid behavior. Fluids are one of the most extensively studied physical phenomena for simulation purposes, with a wealth of research conducted over the years. This has led to the development of highly efficient solutions, particularly for real-time applications and GPU-based algorithms, making them well-suited for the techniques introduced here. Parallelizing these approaches offers the potential for significant improvements.

This research project prioritized the following goals:

- GPU-based large-scale crowd simulation: to explore the feasibility of converting established macroscopic techniques into parallel versions for GPU-accelerated large-scale crowd simulations.
- Adaptability: to develop an elastic model which is able to scale to diverse outdoor environments.
- Performance optimization: to enhance system saturation and to support customer-grade hardware by optimizing the model's performance and efficiency.
- Hybrid model and realism: to improve overall simulation quality and realism by combining techniques from different crowd simulation classes to create a hybrid model.
- Efficient pair-wise algorithm: to design a highly efficient GPU-based pairwise algorithm for managing microscopic behaviors.
- Motion synthesis: to introduce a basic yet effective motion synthesis features to evaluate the model's performance with 3D animated agents.

This paper is structured as follows. In Section II we provide an overview of crowd simulations approaches and solutions based on parallel computing. In Section III, we describe the main components of the proposed model. In Section IV, the mathematical background of the Potential Fields [1] technique is discussed along with the proposed approach and the optimizations introduced to achieve efficient parallel algorithms. Following, in Section V, we widely describe the techniques composing the collision avoidance method. Section VI presents the results of the proposed model on different test scenes. Finally, in Section VII we draw some conclusions and we discuss on future work and achievable improvements.

A. ACRONYMS AND NOTATION

For sake of clarity, the following section provides a concise overview of the acronyms, symbols, and mathematical notations adopted in this paper.

1) ACRONYMS

FPS	Frame Per Second.
SFM	Social Force Model.
ORCA	Optimal Reciprocal Collision Avoidance.
RVO	Reciprocal Velocity Obstacle.
SPH	Smoothed Particle Hydrodynamics.
VO	Velocity Obstacle.
OCEAN	Openness, Conscientiousness, Extraversion, Agreebleness, Neuroticism.
PEN	Psychoticism, Extraversion, Neuroticism.
UIC	Unilateral Incompressibility Constraint.
HLSL	High Level Shading Language.
SIMT	Single Instruction Multiple Threads.
CCR	Continuous Crowd Representation.

FFM	Fast Marching Method.
FIM	Fast Iterative Method.
IFIM	Improved Fast Iterative Method.
VAT	Vertex Animation Texture.

2) NOTATION

$G \in \mathbb{R}^2$	Coordinates of the agents' goal.
f	Speed field.
$\vec{x}$	Discrete individuals' motion.
θ	Direction of the agent's motion.
$\vec{n}_\theta$	Unit vector along θ.
g	Discomfort field.
Π	Set of all the paths from x to G.
$P \in \Pi$	Minimizing path to G.
$\alpha \in [0, 1]$	Unit cost field weight related to path length.
$\beta \in [0, 1]$	Unit cost field weight related to the required time.
$\gamma \in [0, 1]$	Unit cost field weight related to discomfort.
C	Unit cost field.
Φ	Potential function.
$\nabla\Phi$	Gradient function of Φ.
ρ	Crowd density field.
$\vec{v}$	Average velocity field.
$\vec{v}_i$	Scalar velocity of individual i.
w_x	Interpolation weight for ρ.
x_i	Position of an agent.
$\bar{\rho}$	Threshold density contribution of the splatting method.
$(P_x, P_y) \in \mathbb{R}^2$	Coordinates of the agent's position.
λ	Splatting decay of the splatting method.
f_T	Topological speed field.
$f_{\min}$	Minimum speed of agents.
$f_{\max}$	Maximum speed of agents.
$h(x) \in \mathbb{R}$	Height map of the environment.
$s_{\min}$	Minimum slope evaluated by the speed field.
$s_{\max}$	Maximum slope evaluated by the speed field.
$f_{\vec{v}}$	Crowd flow speed.
r	Radius of the estimated distance traversed by agents in a single simulation step.
$\eta_{\min}$, $\eta_{\max}$	Thresholds for the density environment classification.
T	Size of the environment.
τ	Agents per cell ratio.
δx, δy	Size of a grid cell.
$\rho_{\max}$	Maximum density compression value.
$d_{\min}$	Minimum interpersonal distance.
$\vec{\boldsymbol{u}}$	Corrected velocity field.
p	Scalar pressure field.
ν	Kinematic viscosity of a fluid.
$\vec{F}$	External forces acting on a fluid.
n	Number of simulated agents.

II. RELATED WORK

The field of crowd simulation encompasses a diverse range of models, categorized into three main groups [3] based on their purpose, inherent characteristics and level of detail: microscopic, mesoscopic, and macroscopic.

Beyond the core functionalities of a crowd simulation model, another crucial property influencing both the simulation and the model itself is the type of crowd being represented: homogeneous or heterogeneous. This distinction hinges on the number and types of goals the crowd pursues. A homogeneous crowd is defined as a group where individuals share a single goal or a set of very similar goals. Conversely, a heterogeneous crowd is one where individuals possess diverse goals. This heterogeneity significantly impacts the design and development process. Algorithms within the model need to be dynamic and adaptable, capable of adjusting based on the currently evaluated goal of each agent. Additionally, heterogeneous crowd simulations offer the potential to assign distinct behaviors or identities to individual agents, further enriching the simulated crowd's complexity.

This section explores the diverse types of crowd simulation models and their applications. We will follow the classification used by Yang et al. [3] and delve into the most popular approaches associated with each category.

A. MICROSCOPIC MODELS

Microscopic models prioritize the intricate representation of each individual agent interactions with their surroundings. They incorporate meticulously crafted micro-behaviors, often employing an agent-based approach where each simulated individual follows a distinct set of rules. This methodology can lead to exceptionally realistic and granular crowd behavior that factors in socio-psychological influences. However, the computational burden associated with processing these rules for each agent limits microscopic models to simulating relatively smaller crowds.

Early crowd simulation models, like the Social Force Model (SFM) [4], [5], focused on interactions between individuals and the environment, incorporating socio-psychological factors.

Collision avoidance techniques became popular, with notable examples including Optimal Reciprocal Collision Avoidance (ORCA) [6] and Reciprocal Velocity Obstacle (RVO) [7]. These models calculate the ''Velocity Obstacle'' (VO) space: a set of velocities that would cause a collision between two agents. Then, they calculate and choose a safe velocity outside this space. This algorithm is applied for each pair of agents to compute the crowd motion.

Agent-based models emerged as an evolution of collision avoidance techniques. These models combine them with global knowledge techniques to achieve complex simulations. The ClearPath model [8] is an example. It improves the accuracy of VO models, integrating parallel algorithms for large-scale crowd simulations with a microscopic approach.

Research has yielded diverse models with innovative combinations of these techniques. Examples include the Lagrangian formulation in Position-based Dynamic Systems [9] and cooperative behaviors like pushing and pulling introduced by Wong et al. [10].

Velocity-based models, introduced by Ondřej et al. [11], focus on perception-action cycles. These models use a virtual camera to gather information about the environment and calculate metrics like bearing angles and time-to-interaction to guide collision avoidance.

While microscopic models offer flexibility due to their intricate rule sets, their computational complexity increases exponentially. Even with GPU acceleration, the performance of microscopic simulations can be limited by the numerous rule combinations that individual agents must process based on their context. While some GPU-based microscopic models were proposed [12], [13], they often exhibit limitations in terms of agents count and overall performance compared to other GPU techniques used in crowd simulation. Because the focus of our research is on large-scale simulation and increased realism, we have not considered at the moment the introduction of a full microscopic model in our proposed system. Rather, we have opted for a hybrid model, where we integrated microscopic behaviors inside a macroscopic model.

B. MESOSCOPIC MODELS

Crowd simulation models can dig deeper than simple collision avoidance. Mesoscopic models have gained significant traction within the scientific community due to their unique approach of simulating group dynamics and complex, human-like behaviors. These models operate at a higher level of abstraction with respect to the microscopic ones. They focus on groups of simulated entities rather than individual agents. This shift allows to incorporate psychological analysis at both global and local emotional levels, enhancing the realism of crowd behavior.

Among the mesoscopic models, dynamic group models focus on the behavior, psychology and relationships within social groups to achieve a more realistic portrayal. This theory, introduced by Lewin [14], defines a social group as individuals who interact and share information. These models explore both intra-group (within-group) and inter-group (between-group) dynamics [15]. Intra-group dynamics consider roles like leaders who define group movement and followers who simply tag along. Inter-group dynamics examine interactions when different groups cross paths. Pedestrian walking patterns are a common scenario studied using these models, as groups often form pairs or triples while walking [16], [17].

Individual-group interactions are another area of study. Bruneau et al. [18] simulated behaviors like ''going through'' and ''going around'' using a combination of the Principle of Minimum Energy and the RVO model. This research highlights how group density, direction and appearance influence these interactions.

Furthermore, a burgeoning subcategory within mesoscopic models specifically addresses the formation of interactive groups. This is particularly relevant in fields like video game development and manipulation studies. These models offer flexibility in manipulating crowd formations based on user input. They bridge the gap between mesoscopic and macroscopic models, handling larger crowds than typical mesoscopic models while focusing on user-defined goals rather than complex social psychology simulations. One example is ''CrowdBrush'' [19]: a real-time crowd manipulation framework that allows designers to edit crowds through a user interface, resulting in high-quality animations. Subsequent models have expanded on these tools and effects, increasing crowd realism with various mathematical approaches like Delaunay triangulation [20] and Laplacian matrices [21].

Mesoscopic models, from the social psychology perspective, focus on simulating individual traits and emotional contagion, both important aspects of human behavior. These models represent emotions and individual psychology, drawing from various psychological theories. Two prominent examples are the OCEAN model [22] and the PEN model [23]. By incorporating these personality descriptions, crowd simulations can model emotion contagion. Several adaptations of these models have been proposed, such as the OCC model [24] and the PAD model [25], which categorize human personalities for simulation purposes. An example is the proposal of Guy et al. [26], which uses the PEN model to simulate a heterogeneous crowd with different individual personalities influencing the behavior of each agent. Kim et al. [27] built upon this by integrating real-world data and stress factors to create more diverse crowd behaviors. Xu et al. [28] used an adaptation of the RVO model with emotion contagion to simulate emergency evacuations considering different personalities and hazards.

Mesoscopic approaches often rely on detailed individual representations, which can pose a challenge for efficient parallel implementation. While these models represent a promising area of research, our research prioritizes achieving a high-efficiency crowd simulation model using GPU computing. Our proposed model successfully incorporates microscopic behaviors without compromising the overall complexity, demonstrating the feasibility of the integration of mesoscopic techniques. A more comprehensive consideration of mesoscopic approaches will be investigated in future developments of the proposed project.

C. MACROSCOPIC MODELS

Macroscopic models excel at simulating large-scale crowd scenarios. They prioritize estimating individual movement through its relation to observable physical phenomena like potential fields or fluid dynamics. Simulations leverage a continuously updated logical grid or data structure that parti-

tions the space. Higher resolution grids enhance accuracy but increase complexity. Compared to other models, macroscopic approaches provide less detailed representations of individual local behavior. This approach stems from the primary reliance on global methods and spatial approximations to dictate movement, prioritizing the efficient simulation of denser and larger crowds. However, macroscopic models offer the distinct advantage of integrating more complex techniques for both global planning and collision avoidance. While sacrificing some individual realism, these models allow for more refined and computationally efficient simulations.

Within the macroscopic class, continuum models represent the most promising set of techniques. They are founded upon the theory of Pedestrian Flow Dynamics introduced by Hughes [29]. This theory proposes leveraging continuous flow equations to model both crowd movement and individual behaviors. A landmark application of this theory emerged in the Continuum Crowds technique proposed by Treuille et al. [1]. This technique demonstrated the vast potential of this class of models by implementing an efficient large-scale approach. It exploits potential fields to define a global planner based on density and velocity, coupled with discomfort fields to guide the simulated population. The Continuum Crowd method has significantly influenced the literature, inspiring extensive research efforts that have expanded upon its concepts. This has led to the development of an entire subfield within the domain of macroscopic crowd simulation. An example is the Interactive Navigation Field method by Patil et al. [30]. This approach allows users to define crowd directions and routes through input guidance fields. These fields contribute to the overall potential field used for agent motion computation.

Another research direction, inspired by continuum models, utilizes fluid dynamics equations and flow simulations to represent crowd movement. A notable example is the Aggregate Dynamics model by Narain et al. [2], which adopts a collision avoidance system based on a continuous representation of the crowd. To facilitate this, an incompressibility constraint called Unilateral Incompressibility Constraint (UIC) is employed. Fluid dynamics has proven to be a highly effective approach for simulating crowd behavior, particularly in high-density situations where crowds exhibit fluid-like characteristics. Consequently, researchers have explored various fluid dynamics techniques for crowd simulation, including Smoothed Particle Hydrodynamics (SPH). Yuan et al.'s pioneering work [31] introduced SPH to crowd simulation. The subsequent research of van Toll et al. [32] advanced this approach by combining SPH fluid dynamics with agent-based collision avoidance techniques like SFM [4], [5] and RVO [7].

The research presented in this paper focuses on macroscopic crowd simulation techniques due to their potential for GPU-friendly implementations. The Potential Fields [1] and Aggregate Dynamics [2] techniques, selected as a basis for our research, employ 2D crowd representations and homogeneous calculations. The exploitation of the potential efficiency of these methods within the context of the GPU allows us to achieve a high-performance crowd simulation model. Further extensions to this model are possible through the introduction of more complex techniques and features based on Potential Fields and Aggregate Dynamics methods, which build upon the optimizations introduced in our study.

D. HYBRID MODELS

Several research efforts have explored hybridizing macroscopic and agent-based models in order to leverage the advantages of both the approaches and to introduce individualistic behaviors. A notable example is the dualistic model proposed by Saeed et al. [33], which employs an agent-based component for individual movement while governing collective actions through a dedicated group module. This group module incorporates ''service points'' where the agents of the entire group pause to replenish energy before resuming their shared goals. While offering individually assigned goals and real-time path-finding capabilities, the model struggles to achieve a frame rate useful to interactive applications, restricting its application to offline simulations.

Computational efficiency remains a crucial aspect of large-scale crowd simulations. Skrzypczak and Czarnul [34] addressed this challenge by exploring various implementation strategies for optimal hardware utilization. Their work analyzed the strengths and weaknesses of CPU-only and GPU-only implementation, ultimately demonstrating the superior efficiency of a hybrid approach. This approach focused on balancing workload between CPUs and GPUs through a model combining the microscopic SFM [4], [5] with the macroscopic spatial discretization technique. While the model achieved excellent results, the showcased computations relied on a high-performance system with multiple CPUs and GPUs, limiting its applicability on more standard hardware configurations.

Large-scale crowd simulations often require simplifying crowd behavior to reduce computational demands. BioClouds [35] addresses this by representing groups as ''clouds'' instead of individual agents. It builds upon BioCrowds [36], another group-based model, but adds a ''zoom-in'' function to observe individual behavior within specific areas, achieving a multi-level simulation. This approach allows for studying crowd behavior on a large scale while still observing individual nuances in specific areas. High-performance computing has also been explored: Malinowski and Czarnul [37] used distributed caching with non-volatile RAM, achieving impressive agent counts. However, their approach requires a custom and uncommon hardware configuration, thus limiting a broader application of the method.

Analysing literature, crowd simulation research traditionally favors CPU-based models due to their simpler implementation. However, GPUs offer the potential for significantly faster simulations, especially for large-scale scenarios. The decline in research on macroscopic CPU-

based models in favor of mesoscopic ones, coupled with the growing capabilities of GPUs for real-time applications, has opened new avenues for GPU-based large-scale crowd simulation.

This paper proposes a new research direction by leveraging existing GPU-optimized techniques and combining established macroscopic models: potential fields for global planning, and aggregate dynamics for collision avoidance. This combination, rarely explored before, holds promise for efficient large-scale crowd simulations. Additionally, the extensive research in GPU computing on optimizing simulations of physical phenomena, such as fluid dynamics, can be exploited. Following Stam's seminal work [38] on 2D fluid simulation using Navier-Stokes equations on CPUs, Harris adapted it for GPUs [39] and Crane et al. [40] extended it to 3D, demonstrating efficient algorithms still used today for their balance of visual quality and computational cost. These advancements in GPU fluid simulation sparked interest in applying similar techniques to macroscopic crowd simulation models. While macroscopic techniques offer computational efficiency, they have to compromise on simulation realism. To address this issue, our research leverages GPU computing of the proposed model to extend the standard capabilities of the macroscopic approach. The optimized efficiency enables the introduction of microscopic behaviors, enhancing control over the simulated crowd. This integration of microscopic behaviors transforms the model into a hybrid crowd simulation model.

III. SCIENTIFIC CONTRIBUTION

This paper introduces a novel crowd simulation model based on a pipeline architecture. This model extends existing techniques and incorporates refinements in each phase to achieve real-time performance. The pipeline architecture is particularly well-suited for crowd simulations because they can be decomposed into independent features. Large-scale crowd simulations typically consist of two main features: global planning, which determines individual agent velocities to guide them towards their goals, and collision avoidance, which refines these velocities to prevent collisions with structures, obstacles, and other agents.

The proposed architecture model is composed by 4 steps:

1) Global Planner: it computes the intended movement for each individual. It considers a Continuous Crowd Representation (CCR) of the crowd updated at every iteration.
2) Continuous Crowd Update: the CCR is updated with the velocities proposed by the Global Planner. It produces input data for the Collision Avoidance phase.
3) Collision Avoidance: it is in charge to apply corrections to the proposed movement in order to reduce the density of the crowd in areas where agents are closer than the minimum interpersonal distance.
4) Crowd Discretization: the crowd is converted again to an agent-based, or discrete, representation.

A detailed scheme of the model architecture, presenting all phases from initialization to simulation steps, is shown in Fig. 1. The proposed method is not based on a simple conversion of existing crowd simulation techniques to GPU, but it improves and optimizes these models for parallel processing using Compute Shaders. This optimization unlocks the full potential of the GPU Single Instruction Multiple Threads (SIMT) architecture.

Specifically, our proposal introduces a GPU-based solver for global path planning via potential fields. This solver incorporates optimizations to ensure real-time performance. The collision avoidance module is also enhanced with a new pressure projection strategy inspired by GPU fluid dynamics approaches. This strategy enables efficient crowd motion correction. Finally, to address the limitations of prior techniques due to numerical approximations, a GPU-friendly pairwise test is implemented.

A. ARCHITECTURE OVERVIEW

The architecture reported in Fig. 1 is composed by a number of sequential phases. We report here a brief overview of the operations performed during the initialization and at each simulation step.

The initialization phase involves calculating the grid size using an adaptive method (marked as *(1)* in Fig. 1, and described in Section IV-C5). Agent positions, unique maximum speeds (*(2)*, Sec V-B4), and goals are then assigned. Following this, goals are created within the environment and speed, obstacle, and unit cost fields are generated. These fields contribute to the calculation of the static potential fields (*(3-A)*, Sec. IV-C).

During each simulation step, the static potential field is combined with agent velocities and a discomfort field to produce global planning directions (*(3-B)*, Sec. IV-C). These directions are then converted into the CCR (*(4)*, Sec. IV-B) to be used by the aggregate dynamics collision avoidance algorithm (*(5-A)*, Sec. V-A). In parallel, the pairwise test calculates and applies forces to correct the position of agents violating the minimum interpersonal distance with nearby agents (*(5-B)*, Sec. V-B). Finally, these corrections are combined, and the agents' updated velocities are obtained from the discretized CCR (*(4)*, Sec. IV-B) and adjusted by the agents unique max speed (*(2)*, Sec. V-B4).

The inputs and outputs for each phase are the position and velocity of the agents in the case of the agent representation, while the crowd is represented by the density and velocity fields in the continuous representation.

IV. GLOBAL PLANNING

Treuille et al. [1] introduced a novel approach to crowd simulation using continuum theory. Their model, known as the Potential Fields model or Continuum Crowds, leverages differential equations and potential fields to create a global planning system. This system efficiently considers dynamic obstacles and interpersonal distance without explicitly requiring classical collision avoidance mechanisms.

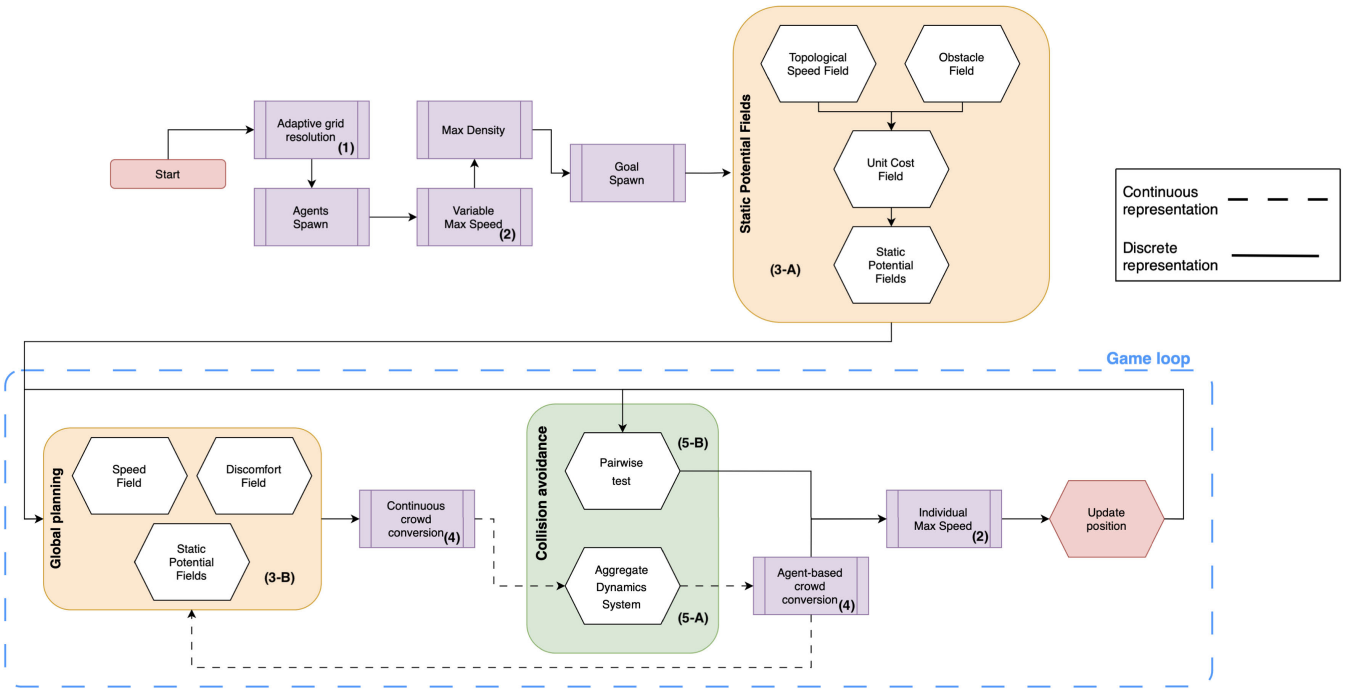


FIGURE 1. Detailed model architecture, exposing every operations performed during the initialization and at each simulation step. An overview of the architecture is provided in Section III-A.

This macroscopic model is primarily designed for simulating large, homogeneous crowds. However, it can also handle heterogeneous simulations by dividing the crowd into goal-specific groups. Each group has a unique potential field for navigation, enabling the simulation of multiple goals simultaneously. This approach works effectively, but it becomes computationally expensive for heterogeneous crowds with many groups, offering only a minor performance improvement over agent-based models as the number of groups increases.

The core concept of the Potential Fields model is to simplify crowd motion as a particle energy minimization problem. This allows the simulation to be treated as a continuous problem, leading to an efficient global planning system. This approach surpasses agent-based models, which often split path planning into local and global components, limiting achievable realism. The introduced trade-off was an efficient model for large crowds at the expense of homogeneous behavior. Similar to other macroscopic models, performance is heavily influenced by the number of agents and the size of the simulation environment, usually represented by grids. Larger grids lead to more expensive simulations.

This section delves into the global planning technique of potential fields [1]. Section IV-A provides a description of the technique from a mathematical standpoint. Following this, Section IV-B explores the concept of a continuous crowd and the computations involved in achieving it. Finally, Section IV-C offers a detailed explanation of the parallel transposition of the technique, including the proposed improvements for the solver algorithm.

A. MATHEMATICAL DESCRIPTION

The underlying logic of this technique is based on the following assumptions:

1) Each individual aims to reach a specific coordinate inside the given environment $G \in \mathbb{R}^2$.
2) The individuals' motion is always set to the max speed allowed by the current position and surrounding environment. This speed is defined as the scalar speed field f such that
$$\vec{x} = f(x, \theta)\vec{n}_\theta$$
where x is the current agent's position, θ is the direction of the agent's motion, and $\vec{n}_\theta = [\cos\theta, \sin\theta]^T$ represents the unit vector in the direction of θ.
3) The discomfort field g is defined as the scalar field that introduces a weight to the cost of movement in the directions available to the agent, allowing it to prefer the best paths.
4) Let Π be the set of all the paths from x to the goal. Assuming that the speed field f, the discomfort field g and the goal G are fixed, the chosen path $P \in \Pi$ minimizes the following formula:
$$\alpha \int_P ds + \beta \int_P dt + \gamma \int_P g\, dt$$
where $\alpha, \beta, \gamma \in [0, 1]$ are the weights for each component. We can redefine the previous formula considering the relationship $ds = f\, dt$:
$$\int_P \frac{\alpha f + \beta + \gamma g}{f} ds = \int_P C\, ds$$
where we define C as the scalar *unit cost field*.

We define the potential function $\Phi : \mathbb{R}^2 \to \mathbb{R}$ to describe the cost of a path to the goal from the given coordinate. Akin to many path-finding techniques, this algorithm seeks to minimize the path cost. It achieves this by minimizing the potential function Φ and then calculating the path direction by following the opposite of the gradient function $\nabla\Phi$:

$$\Phi(x) = \begin{cases} 0 & \text{if } x = G \\ \|\nabla\Phi(x)\| = C & \text{otherwise} \end{cases}$$

The equation $\|\nabla\Phi(x)\| = C$ can be related to a particular type of first-order, nonlinear partial differential equation named Eikonal equation, used to model wave propagation within a medium.

The theorem of the calculus of variation, discussed and proven by Kimmel and Sethian [41], guarantees that all optimal paths follow the direction of the gradient. Finally, the velocity of the individuals are defined as:

$$\vec{x} = -f(x, \theta)\frac{\nabla\Phi(x)}{\|\nabla\Phi(x)\|}$$

B. CONTINUOUS CROWD REPRESENTATION

Due to the spatial variations in crowd characteristics, traditional agent-based representations have been proven to be inadequate. To address this limitation, the model employs a Continuous Crowd Representation (CCR). This representation maps the entire environment onto the position and current velocity of each individual agent, resulting in a continuous description of the crowd (Fig. 1, marked as *(4)*). The CCR consists of two crucial fields: a crowd density field ρ reflecting the spatial distribution of individuals, and an average velocity field $\vec{v}$ describing the overall movement direction:

$$\rho = \sum_i \rho_i = \sum_i w_x(x_i)$$
$$\vec{v} = \frac{\sum_i \rho_i \vec{v}_i}{\rho}$$

where w_x is the *interpolation weight* proposed by a splatting method, x_i is the current position of the i-th agent, and $\vec{v}_i$ is the current velocity of the i-th agent.

The population of these continuous representation fields necessitates a specific splatting algorithm that fulfills certain criteria:

1) Fields must guarantee the continuity of the position of agents, to ensure the presence of consistent values over the entire domain.
2) A threshold value $\bar{\rho}$ is introduced. Each individual can contribute a minimum value of $\bar{\rho}$ to the corresponding field coordinate from its position, while in neighboring coordinates its contribution cannot exceed the threshold.

The algorithm proposed in [1] finds the closest cell with respect to the agent's position $(P_x, P_y) \in \mathbb{R}^2$, and then it defines the contributions to this cell and the nearest ones as:

$$\begin{cases} \Delta x = P_x - \lfloor P_x \rfloor \\ \Delta y = P_y - \lfloor P_y \rfloor \end{cases}$$
$$\rho_A = \min(1 - \Delta x, 1 - \Delta y)^\lambda \quad \rho_B = \min(\Delta x, 1 - \Delta y)^\lambda$$
$$\rho_C = \min(\Delta x, \Delta y)^\lambda \quad \rho_D = \min(1 - \Delta x, \Delta y)^\lambda$$

where λ is a parameter determining the *splatting decay*, and ρ_i are the interpolation weights for the involved cells, as shown in Fig. 2. This method also satisfies the first condition if the threshold is defined as $\bar{\rho} = 1/2^\lambda$.

C. PROPOSED PARALLEL IMPLEMENTATION

Following both the global planning and the collision avoidance steps, the CCR needs to be converted back into a discrete agent-based representation. Subsequent steps, including the final integration method that updates individual positions, require per-agent data.

The CCR is typically stored in a two-dimensional (or equivalent) data structure. In a GPU implementation, 2D textures offer optimal performance. To convert this data back to individual agent information (global position and movement velocity), filtering methods can be applied. Given the assumed continuity of the representation, a natural choice is bi-linear interpolation. This technique ensures retrieval of the most appropriate data from the fields for each agent based on their position.

1) SPEED FIELD

The speed field defines the maximum achievable movement speed for agents throughout the environment. In this paper, we adopt a different approach compared to Treuille et al. [1] where potential field values were influenced by crowd density. Inspired by the real-time optimization strategies of Lu et al. [42], this component is replaced by a static field (Fig. 1, marked as *(3-A)*). This static field is solely determined by ground conditions, simplifying calculations and improving performance. We define this particular field as the topological speed field f_T:

$$f_T(x, \theta) = f_{\max} + \left(\frac{\nabla h(x) \cdot \vec{n}_\theta - s_{\min}}{s_{\max} - s_{\min}}\right)(f_{\min} - f_{\max})$$

where $\nabla h(x) \cdot \vec{n}_\theta$ is the slope relative to the direction θ, while $s_{\min}$ and $s_{\max}$ define respectively the minimum and maximum slopes. This modification enables pre-computation of potential fields, and it eliminates the need for recalculation at each frame, thereby enhancing efficiency by up to $\approx 60\%$.

Real-world crowd movement and velocity are also influenced by the overall motion of the crowd. This mirrors the concept of self-advection in fluids, where the velocity of the fluid transports objects, densities, and other properties along its flow, but the fluid itself is also affected, creating a feedback loop. Similarly, in crowd simulations, the velocity of the crowd can influence its own movement and evolution. To account for this, a second type of speed, the crowd flow $f_{\vec{v}}$,

is incorporated and considered during the calculation of the global planning strategy for individual agents:

$$f_{\vec{v}}(x,\theta) = \vec{v}(x + r\vec{n}_\theta) \cdot \vec{n}_\theta$$

where $\vec{v}$ is the velocity field and r is the radius of the estimated next position. The simulation step involves recalculating the speed field to incorporate the influence of crowd flow (Fig. 1, marked as *(3-B)*). To isolate this influence from the base topological speed, the local density at each agent's position (from the density field) is evaluated. Using a pair of thresholds, $\eta_{\min}$ and $\eta_{\max}$, the environment is classified into sparse, moderately dense, and highly dense:

$$f(x,\theta) = \begin{cases} f_T(x,\theta) & \text{if } \rho \le \eta_{\min} \\ f_{\vec{v}}(x,\theta) & \text{if } \rho \ge \eta_{\max} \\ \omega\, f_{\vec{v}}(x,\theta) + (1-\omega) f_T(x,\theta) & \text{otherwise} \end{cases}$$

$$\omega = \left(\frac{\rho(x + r\vec{n}_\theta) - \eta_{\min}}{\eta_{\max} - \eta_{\min}} \right)$$

The speed field defines the maximum permissible velocity magnitude for an agent in each cardinal direction (left, right, forward, and backward) within its local area. This results in an *anisotropic* field, where a set of values is associated with each coordinate. For GPU efficiency, in the proposed implementation we use a 2D floating-point texture where each channel (red, green, blue, and alpha) represents the speed in north, east, south, and west directions respectively. This approach is also applied to the unit cost field, as it depends on the agent's movement direction.

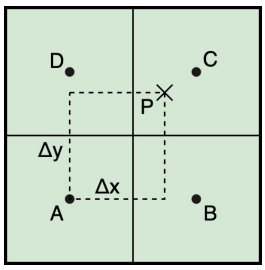


FIGURE 2. Visualization of involved cells during the splatting method [1].

2) DISCOMFORT FIELD

While the static potential field removes the need for a discomfort field, the proposed implementation incorporates this concept in a modified way. The discomfort field still influences agent movement, discouraging paths that intersect with obstacles and steering away from congested areas. The field is generated by merging two separate fields: the Obstacle Field and the Future Density Field. The former represents the presence and influence of static obstacles within the environment. The latter estimates the agent density distribution based on current position and movement direction. After the generation, the two fields are merged to create the discomfort field. This discomfort field is then incorporated as a weighting factor for the static potential fields. The potential function Φ resembles a "funnel" surface with a single global minimum at the goal (Fig. 3). To preserve this crucial property, the discomfort field needs adaptation. This is achieved by scaling discomfort values proportionally to their distance from the goal. This scaling replicates the funnel-like distribution and it allows for effective weighting of the static potential fields. While this approach only partially replicates dynamic potential fields, it generates an emergent behavior: individuals tend to cluster and move in groups.

3) OBSTACLE FIELD

Effective obstacle avoidance remains essential for realistic crowd simulation. Therefore, methods to represent both partial and complete obstruction within the environment are mandatory. Consequently, a scalable method is needed to handle any quantity of obstacles regardless of their shape or numerosity. This paper adapts the Inside-Outside Voxelization algorithm, a technique proposed by Crane et al. [40] in the context of 3D fluid simulation. This technique is also based on the Stencil Shadow Volume algorithm [43]. This method represents obstacles within the environment using a 3D volumetric grid (voxel grid). The Stencil Test, a GPU functionality, plays a crucial role in this algorithm. It uses the stencil buffer to distinguish between three cases: objects entirely outside the environment, objects entirely inside, and objects partially intersecting the environment. Leveraging these case distinctions, the algorithm selectively modifies the voxel grid. This ensures that only the relevant obstacle areas, corresponding to interactable regions, are represented.

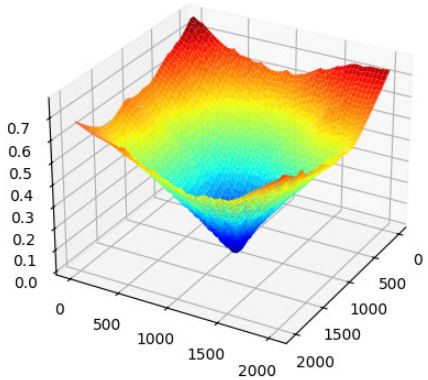


FIGURE 3. Funnel-shaped surface of the potential field.

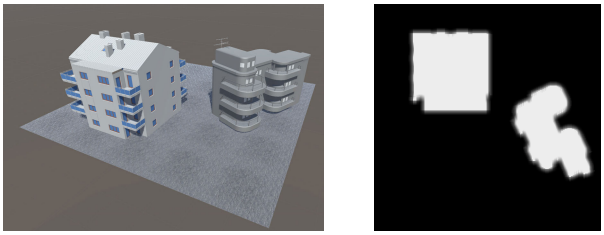
Since obstacle representation within a 2D space is inherently simpler, the implemented algorithm draws inspiration from the layer-slicing strategy employed for the 3D approach. To achieve this selective representation, the algorithm employs a virtual camera with an orthogonal projection. The camera is positioned at the center of the environment, looking down, at a specific height to capture only relevant obstacles. The camera frustum is extended by the size of the entire environmental area. This ensures that the resulting grid

accurately reflects the obstacles that can potentially obstruct agent movement (Fig. 4).

This obstacle mapping algorithm, unfortunately, generates binary textures with sharp edges around obstacles, contradicting the assumed continuity in the crowd representation. The proposed solution to address this issue and enhance simulation realism is the introduction of a Gaussian blurring near these edges. This allows agents to ''perceive'' obstacles before reaching them, improving their behavior. Furthermore, since the obstacle field represents static elements in the environment, it can be pre-computed during initialization and it does not require recalculation during each simulation update, contributing to its efficiency.

4) FUTURE DENSITY FIELD

To enable agents to proactively avoid high-density areas, potential fields are employed. However, using only the current density field is insufficient as it solely reflects the present crowd distribution. Therefore, an additional density field is introduced, representing an estimated future agent distribution. This estimation is achieved by simply considering the current velocities of all agents and extrapolating their positions forward in time using an integration step of a few seconds. The chosen time step for calculating the future density field needs to be adjusted based on the ratio of environment size to grid size. Coarser grids necessitate a larger time step to capture a meaningful change in the density field, as each cell represents a larger area.



(a) Scenario with complex buildings. (b) Resulting obstacle field.

FIGURE 4. Example of the adaptive algorithm introduced for the generation of obstacle fields.

5) ADAPTIVE GRID RESOLUTION

Grid size is a critical parameter in grid-based algorithms, significantly impacting the overall performance. The ability to dynamically adjust and minimize grid size without compromising result quality can lead to substantial improvements. In the context of the techniques considered for the proposed model, an adaptive method for field size should consider both the number of simulated agents and the environment's dimensions (Fig. 1, marked as *(1)*). Coarse grids, while computationally efficient, would result in higher field values and provide less accurate information about crowd distribution.

The proposed adaptive method considers ratio distributions about the number of agents and terrain size to provide the optimal grid size:

$$S = \lfloor k\sqrt[4]{T^2N} + (1-k)\sqrt{\tau N} \rfloor$$

where T is the size of the environment, and τ is the desired agents per cell ratio. The parameter $k \in [0, 1]$ is the interpolation factor that can be used to balance the relationship between terrain and agents ratios. This method exhibits a logarithmic performance trend as the number of simulated agents increases (Fig. 5), confirming the advantages of continuous representations.

While a large environment is necessary to accommodate the agents and generate valid field representations, this can lead to uniform overcrowding across the entire area and less realistic simulations. Therefore, careful parameter tuning based on simulation results is crucial to provide optimal grid sizes.

D. EIKONAL SOLVER

The potential function Φ is defined by the Eikonal equation, introduced by Bruns [44], that describes wave propagation in a medium. Due to its complexity, this equation cannot be solved directly and it requires numerical approximations and specialized algorithms. This highlights the computationally intensive nature of the potential fields model. Following the construction of the unit cost field, the calculation of Φ_M for each cell M necessitates an approximation using finite differences. This involves identifying the adjacent cells with lower costs in both axes:

$$m_x = \arg\min_{i\in\{W,E\}}\{\phi_i + C_{M\to i}\}$$
$$m_y = \arg\min_{i\in\{N,S\}}\{\phi_i + C_{M\to i}\}$$

To approximate the Eikonal equation, the upwind discretization scheme by Godunov and Bohachevsky [45] is employed. This popular finite difference method transforms the Eikonal equation into a quadratic one, facilitating calculations:

$$\frac{(\phi_M - \phi_{m_x})^2}{C_{M\to m_x}} + \frac{(\phi_M - \phi_{m_y})^2}{C_{M\to m_y}} = 1$$

Once the entire potential function field is defined, its gradient $\nabla\Phi$ can be computed by taking the difference with neighboring cells in the upwind direction:

$$\nabla\Phi = \left(\frac{\partial\Phi}{\partial x}, \frac{\partial\Phi}{\partial y}\right) = \left(\frac{\Phi_{i+1,j} - \Phi_{i-1,j}}{2\delta x}, \frac{\Phi_{i,j+1} - \Phi_{i,j-1}}{2\delta y}\right)$$

where δx and δy are the sizes of a grid cell. This gradient information is then used to determine crowd movement towards the goal.

1) OVERVIEW OF NUMERICAL SOLVERS FOR THE EIKONAL EQUATION

Due to data dependency and to the nature of the problem, potential fields require the choice and application of numerical solvers. Treuille et al. [1] originally adopted the Fast Marching Method (FMM) [46], similar to the Dijkstra

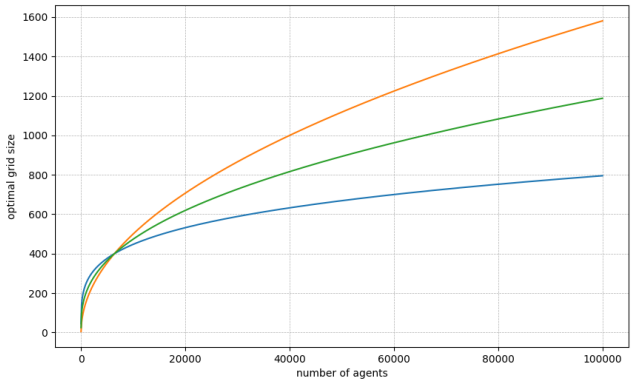


FIGURE 5. Optimal grid size (green) for a given number of agents in a terrain of size $2,000 \times 2,000$, obtained from the interpolation (set to $k = 0.5$) between the estimations from terrain (orange) and agents per cell ($\tau = 25$) ratios (blue).

algorithm. This iterative approach evaluates adjacent cells in the unknown set, selecting the lowest-cost option until reaching the goal. However, this type of algorithm is poorly suited for parallel computing, prompting the implementation of a different solver in this work.

The Fast Iterative Method (FIM) [47] represents a significant advancement in the parallelization of the FMM. Traditional CPU-based solvers for the Eikonal equation impose a specific computation order on cells. This often necessitates sorting steps and additional data structures, hindering efficiency. In contrast, the FIM aims to minimize data dependencies. By relying only on information from immediate neighbor cells, FIM enables parallel computation of distant potential values. This significantly improves overall performance. The upwind discretization scheme requires at least one neighboring cell to converge before calculating the value of the target cell. The core concept of FIM involves ''expanding'' active cells, i.e., the cells satisfying the convergence requirement, in each direction. FIM operates through two interleaved steps: potential calculation for active cells and their convergence check, which expands the active cell list. This sequential dependence hinders parallel execution. Furthermore, the lack of a fixed iteration count due to the ''complete after convergence'' design complicates implementation and increases computational cost.

Huang [48] analyzed and optimized the FIM for potential field calculation, aiming at maintaining its benefits while addressing stability issues. The Improved Fast Iterative Method (IFIM) aims to address the convergence issues and improve the stability of the original FIM algorithm. The key improvement lies in introducing a ''remedy step''. This step postpones recalculating cell convergence until the entire grid has known values, potentially accelerating the activation of the remaining cells. The algorithm utilizes three sets:

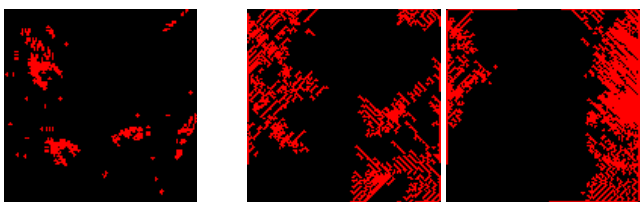
- Active Set: it contains unknown points to be calculated in the next iterations.
- Source Set: it stores points with already calculated values.
- Remedy Set: used during the remedy step to hold cells with potentially improvable values.

The update step follows a similar process as the original FIM. Once the Active Set is empty, the remedy step begins. The convergence test is performed again on the entire grid. The detected cells violating the condition are added to the Remedy Set. Then, cells in the Remedy Set are updated until they definitely converge. Since these updates might impact neighboring cells, they are also added to the Remedy Set for recalculation in subsequent iterations (Fig. 6).

2) PROPOSED IMPROVEMENTS TO IFIM SOLVER

In this paper we enhance stability and determinism in the GPU implementation of IFIM for calculating potential fields. The original FIM and its extention IFIM both suffer from non-deterministic execution, leading to unpredictable performance spikes. This is a significant drawback for real-time applications. The key challenge lies in the algorithm's reliance on data synchronization after each iteration to check for grid convergence. This synchronization is highly inefficient on GPUs. The solution proposed in this paper leverages the fact that the Godunov and Bohachevsky [45] approximation for potential calculation always produces a finite value. Therefore, we simplify the algorithm by eliminating the convergence check and remedy step during the update phase. This ensures immediate cell convergence after the first potential calculation, potentially accelerating the process.

Furthermore, the update scheme of the IFIM activates cells progressively starting from the goal (Fig. 7). With the simplification, the entire grid can be activated in a predictable number of iterations, resulting in a time complexity of $O(n)$ for an $n \times n$ grid. The remedy step is still performed after the update phase. However, experiments show that convergence for most cells is achieved within a limited number of iterations (between n and $2n$). This allows to set a maximum number of iterations for the remedy step, ensuring near-deterministic completion. In rare cases with a few remaining non-convergent cells, a small performance penalty is considered an acceptable trade-off for faster overall execution. Additionally, grid size can be factored in to adjust the number of remedy step iterations for optimal performance.



(a) Detected cells added to Remedy Set.

(b) Propagation of active cells.

FIGURE 6. Active cells during the course of iterations in the remedy step of the original IFIM algorithm [48].

V. COLLISION AVOIDANCE

In crowd simulations, local collision avoidance ensures that agents steer clear of obstacles and other agents. Our proposed model presents a novel approach by combining two complex techniques: potential fields for global planning and aggregate dynamics for collision avoidance. Their complementary roles within the simulation make this combination feasible. Additionally, leveraging GPUs for computations allows for efficient implementation.

The proposed collision avoidance method comprises two steps: aggregate dynamics, a primary method using a macroscopic approach based on CCR, and a pairwise test. The pairwise test compensates for potential numerical errors caused by the spatial approximations inherent to both macroscopic techniques. It ensures a minimum interpersonal distance between agents to prevent overlapping. The computational cost of this additional step is carefully analyzed and reduced through optimized parallel algorithms.

The following sections delve into a detailed discussion of the two aforementioned collision avoidance techniques. Section V-A focuses on the aggregate dynamics approach, providing both the mathematical foundation and the proposed solutions for its GPU implementation. Subsequently, Section V-B examines the final pairwise test, including a comprehensive explanation of the optimization steps leading to the proposed optimal solution.

FIGURE 7. Propagation of active cells through iterations during the update phase, showing the expansion scheme of the original IFIM algorithm [48].

A. AGGREGATE DYNAMICS

Dense crowds pose unique challenges in crowd simulation due to the limited individual movement freedom caused by close interpersonal distances. This results in a collective behavior resembling a single, unified system. When agents are sufficiently spaced, their movement can be approximated as a compressible fluid. However, in very dense scenarios, the physical limitations of interpersonal space make the crowd effectively incompressible. This shift in compressibility presents a significant modeling challenge. In order to exploit fluid motion in highly dense crowd scenarios, Narain et al. [2] propose to directly integrate local collision avoidance techniques into the CCR. This integration is achieved by incorporating fluid motion constraints within the governing equation, specifically the Unilateral Incompressibility Constraint (UIC) (Fig. 1, marked as *(5-A)*).

1) MATHEMATICAL DESCRIPTION

It is important to acknowledge that the simulated crowd does not precisely represent a physical fluid. It does not exhibit the behavior of either a purely incompressible or purely compressible fluid. Instead, Narain et al. [2] propose a novel definition: a "*unilaterally incompressible fluid*". This essentially represents an intermediate situation between the two extreme cases. This definition necessitates a constraint on the fluid's density. This constraint limits the density to a maximum value $\rho_{\max}$, which represents the maximum compression the crowd can undergo. This constant is actually defined from the minimum interpersonal distance $d_{\min}$ between agents, resulting in the following relationship:

$$\rho \leq \rho_{\max} = \frac{4\alpha}{d_{\min}^2}$$

where $d_{\min}$ is the minimum interpersonal distance between agents and $\alpha \in [0, 1]$ is an user-controlled parameter.

To guarantee the validity of the UIC, the velocity field $\vec{v}$ is adjusted in order to ensure that the density remains within permissible limits inside the environment. This correction process aims to maintain the crowd density within a valid range while still allowing individuals to move towards their goals. The corrected velocity field $\vec{\boldsymbol{u}}$ is calculated under the assumption that individuals move at their maximum speed $v_{\max}$ without encountering collisions:

$$\vec{\boldsymbol{u}} = v_{\max} \frac{\vec{v} - \nabla p}{\|\vec{v} - \nabla p\|}$$

where $p \geq 0$ represents a scalar *pressure* component.

This pressure is subjected to the constraints of:

$$\begin{cases} \rho < \rho_{\max} \implies p = 0 \\ p > 0 \implies \nabla \cdot \vec{\boldsymbol{u}} = 0 \end{cases} \quad \longleftrightarrow \quad p > 0 \implies \rho = \rho_{\max}$$

These equations model the behavior of the crowd density by introducing a non-negative pressure field. This pressure acts as a balancing force: it actively limits the density to its maximum value $\rho_{\max}$ when it approaches this threshold. Conversely, when the density value falls below $\rho_{\max}$, the pressure becomes negligible, allowing for free movement within the flow.

When the flow density is low, the pressure is absent and the influence of the UIC solver is smaller. The influence of the UIC solver dynamically adapts according on crowd density. In low-density scenarios, where the pressure field is negligible, the global planning takes primary responsibility for determining individual velocities. To seamlessly transition between this dominance and the pressure-driven behavior in denser regions, an interpolation formula is considered:

$$\vec{\boldsymbol{u}}_i = \vec{v}_i + \frac{\rho(x_i)}{\rho_{\max}} (\vec{\boldsymbol{u}}(x_i) - \vec{v}_i)$$

where $\vec{v}_i$ is the global planning velocity of the i-th agent, $\vec{\boldsymbol{u}}_i$ is the output velocity for the i-th agent, and $\vec{\boldsymbol{u}}(x_i)$ is the interpolated velocity from the continuous corrected velocity field.

2) PARALLELIZATION CHALLENGES

The approach by Narain et al. [2] considers a Quadratic Programming solver [49] for pressure calculation, which is effective on CPUs but it is not efficient in parallel implementations. This poses a challenge for adapting it to the GPU-centric model presented here. The core of this technique lies in applying pressure to the crowd flow whenever the density exceeds the threshold $\rho_{\max}$ in specific areas. This pressure balances the flow and reduces overcrowding, ultimately preventing collisions between individuals. However, calculating pressure for a fluid with an incompressibility constraint presents a challenge.

In fluid simulation, the concept of incompressibility is a fundamental assumption used to define the Navier-Stokes equations, as detailed by Stam [38]. These equations utilize pressure, derived from the fluid's velocity field, to govern the behavior of the fluid within the environment. A technique called pressure projection is employed to relax and expand compressed areas. The method proposed by Harris [39] achieved efficient and realistic fluid simulations on GPUs, surpassing alternative solutions in terms of both performance and computational cost. Therefore, our model utilizes Harris' method to calculate the pressure field required for applying UIC within the aggregate dynamics approach.

3) NAVIER-STOKES EQUATIONS

Understanding the pressure solver necessitates a preliminary discussion of the underlying physical and mathematical principles. A correctly simulated fluid velocity field allows us to describe fluid behavior using the Navier-Stokes equations for incompressible and homogenous flow. These equations, based solely on the velocity field, rely on the assumption that the simulated fluid volume remains constant within any sub-region at any given time, and its density is uniform throughout space. This assumption aligns perfectly with crowd simulation, where constant density can be linked to the constraint of a fixed number of agents, mirroring the core concept of the aggregate dynamics technique. With these constraints in place, the simulated fluid can be accurately represented by the vector velocity field $\vec{v}(x,t)$, and a scalar pressure field $p(x,t)$, provided their initial state at time $t = 0$ is known. The evolution of the fluid is then described with the following Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial \vec{v}}{\partial t} = \underbrace{-(\vec{v}\cdot\nabla)\vec{v}}_{\text{advection}} - \underbrace{\dfrac{1}{\rho}\nabla p}_{\text{pressure}} + \underbrace{\nu\nabla^2\vec{v}}_{\text{diffusion}} + \underbrace{\vec{F}}_{\text{external forces}} \\ \nabla\cdot\vec{v} = 0 \end{cases}$$

where ρ is the constant fluid *density*, ν is the *kinematic viscosity* and $\vec{F}$ represents any external forces that act on the fluid.

The second equation represents the zero divergence condition, that is the aforementioned incompressibility assumption. To fulfill the zero divergence condition, we employ the Helmholtz-Hodge Decomposition Theorem (proven and extensively discussed by Chorin et al. [50]). This theorem allows us to decompose the vector velocity field $\vec{w}$ into a sum of individual vector fields:

$$\vec{w} = \vec{v} + \nabla p \quad \longrightarrow \quad \vec{v} = \vec{w} - \nabla p$$

In particular, the $\vec{v}$ component is divergence-free and it represents the pressure-corrected velocity field to achieve.

Applying the divergence operator to the decomposed second equation of Navier-Stokes, we define the Poisson-pressure equation:

$$\nabla\cdot\vec{w} = \nabla\cdot(\vec{v}+\nabla p) = \underbrace{\nabla\cdot\vec{v}}_{=0} + \nabla^2 p \implies \nabla^2 p = \nabla\cdot\vec{w}$$

Solving this equation yields the pressure field for the flow. By leveraging its gradient, we can then correct the velocity field obtained from the CCR. This enforces UIC and it achieves local collision avoidance within the crowd simulation.

4) PRESSURE SOLVER

Poisson equations can be solved using an iterative relaxation technique. This method approximates the solution by treating the equation as a sparse linear system of the form $A\vec{x} = \vec{b}$. In this system, A is a matrix, $\vec{x}$ is the unknown vector we seek, and $\vec{b}$ is a constant vector. The iterative relaxation technique progressively refines an initial guess for $\vec{x}$ until it converges to a sufficiently accurate solution. A Poisson equation can be solved by the Laplacian operator in the following form:

$$x_{i,j}^{k+1} = \frac{x_{i-1,\,j}^{k} + x_{i+1,\,j}^{k} + x_{i,\,j-1}^{k} + x_{i,\,j+1}^{k} + \alpha\, b_{i,j}}{\beta}$$

where α and β are constants, and k is the iteration number. In the case of the Poisson-pressure equation:

- x represents the pressure
- $b = \nabla\cdot\vec{w}$, which is the divergence of the velocity field
- $\alpha = -(\delta x)^2$, where δx is the size of the grid cells
- $\beta = 4$

Several solvers were proposed in the literature to calculate this type of equation, but only a few are actually efficient in the field of real-time applications. The research by Amador and Gomes [51] compares CPU and GPU implementations of popular solvers. Their analysis identifies iterative Jacobi and Gauss-Seidel solvers as the most performant within the SIMT architecture of GPUs. Both solvers employ the iterative relaxation technique but they differ in how they utilize previously calculated values. Gauss-Seidel leverages already updated cells, reducing memory access compared to Jacobi, which requires reading and writing from separate buffers. Additionally, Gauss-Seidel converges faster, requiring fewer iterations. However, Gauss-Seidel introduces potential race conditions (conflicting access to the same memory location) on GPUs. To avoid this, our model adopts the *red-black* variant. This variant divides the grid into red and black cells, performing updates on alternating colors to guarantee consistency in the neighbouring grid cells of the computed one. Due to the absence of user-defined synchronisation

barriers within kernels (i.e., GPU functions), a single iteration must be split into two kernel executions. While iterative solvers typically require 40-80 iterations for a good solution, their low cost per iteration keeps the overall computational cost manageable. In the context of our model, the Gauss-Seidel solver with the red-black variant guarantees pressure field convergence within 40 iterations.

Iterative pressure solvers rely on an iterative relaxation technique that exploits the concept of fluid divergence. As these values are crucial for pressure calculation, an intermediate step is required to determine the Divergence Field. By employing the finite difference method's divergence approximation, we can calculate the divergence of the fluid for each cell based on the velocity field:

$$\nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = \frac{u_{i+1,j} - u_{i-1,j}}{2\delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\delta y}$$

where $\vec{u} = (u, v)$ is a vector field, while δx and δy are the bi-dimensional cell sizes.

B. PAIRWISE TEST

Due to the approximations in potential fields and aggregate dynamics, interpersonal distance cannot be reliably guaranteed. This calls for a final pairwise collision test in the simulation pipeline. This test involves checking if each agent's updated position violates the minimum distance $d_{\min}$ from all other agents. This step ensures smooth crowd movement and proper individual distribution within the environment (Fig. 1, marked as *(5-B)*).

The high complexity of the pairwise test, NP-hard due to its n^2 nature, necessitates optimizations for efficient GPU implementation. Parallel implementation is challenging because a naive approach evaluating all n^2 agent pairs would require a quadratic number of processors and exclusive memory, leading to poor performance.

The SIMT architecture executes the same instructions across threads, but conditional branching within the kernel splits and slows down performance by forcing threads to sequential computation. In GPU computing, *divergence* measures this performance penalty caused by branching within a warp (i.e., group of threads running in parallel). Minimizing divergence, along with processor count and memory usage, is crucial for an efficient parallel algorithm. The pairwise test calculates velocity corrections only for pairs violating the minimum agent distance. While statistically only a few pairs require correction, this still results in high kernel divergence and it hinders performance. Since this inherent issue cannot be entirely eliminated, optimizations should focus on minimizing other factors like processor usage and memory access patterns.

1) PAIRS REPETITION

Enforcing minimum interpersonal distance involves applying a repulsive force proportional to the distance between an agent and any other agent found within a threshold. The force direction aligns with their relative position, mimicking a pushing effect. The final force applied to an agent is the sum of all such forces from violating pairs.

A naive approach would assign a single agent pair (i, j) to each thread. This requires the computation of the distance between the agents and the application of a correction if needed. However, this requires n^2 memory locations for n agents, leading to poor performance for large-scale simulations.

The first optimization avoids redundant calculations by recognizing that evaluating (i, j) is the same as evaluating (j, i). This allows a single thread to apply the force to both agents, effectively halving the number of required processors. Additionally, the algorithm can be redesigned so that each agent i computes pairs (i, j), where $j = i + 1, \ldots, n$.

To address the quadratic memory usage, a trade-off is introduced. Only n threads are dispatched (i.e., GPU instancing of the requested number of threads), each iterating through all assigned pairs for its corresponding agent. This allows for a linear data structure. However, since exclusive write access is no longer guaranteed, an atomic (synchronized) operation is needed when calculating the force for the other agent, impacting performance. Despite this, the reduced resource requirements lead to a significantly improved performance compared to the naive approach (as depicted in Fig. 8(a)).

2) LOAD BALANCING

In GPU computing, the slowest thread determines the overall execution time of a parallel algorithm. Analyzing the previous version of the pairwise test, we observe that each thread i iterates from agent i to agent n, calculating distances. Thread 0 has the most computations (n pairs), resulting in a linear time complexity. However, GPUs have many low-power processing units compared to CPUs. An algorithm where a single thread performs many iterations suffers from poor performance. Splitting calculations across multiple threads for each agent, however, would be highly complex with the current approach.

Therefore, a different method is proposed, based on the following observation: unlike the first thread, the last thread has no pairs to evaluate, causing performance divergence. This inefficiency can be addressed through load balancing techniques. By fixing threads to evaluate a maximum number of pairs equal to $n/2$, they can be divided into two groups, based on their IDs i. In the first group, threads with $i < n/2$ must compute the pairs (i, j) with $j \in (i, i + n/2]$. In the second group, the remaining threads must compute the pairs (i, j) with $j \in (i, n]$. Threads in the second group are also responsible to compute the pairs $(spec(i), j)$ with $j \in (spec(i) + n/2, n]$, where $spec(x) = n - 1 - x$

Threads that previously had fewer pairs to evaluate are assigned the remaining pairs from their *specular* thread during the second half of execution (illustrated in Fig. 8(b)). The *specularity* relationship, defined by the $spec(x)$ function, identifies pairs of threads with complementary workloads. This allows for even distribution of pair calculations, ensuring

all pairs are evaluated while balancing the number of iterations per thread. The resulting algorithm achieves a time complexity of $O(n/2)$.

3) LOOP UNROLLING

Having achieved good time complexity, the optimization process focused on reducing instructions per thread. Leveraging the fixed-length iterations, the algorithm splits thread execution into sets. In the optimal case, each thread evaluates a single pair. A thread grid of $n \times n/2$ is dispatched, exploiting the two-dimensional IDs: the x-coordinate identifies the agent, while the y-coordinate the iteration. Consequently, the load balancing strategy is preserved (Fig. 8(c)).

While there is an increase in synchronized atomic operations to enforce minimum distance (due to multiple threads accessing the same agent memory location), this is a worthwhile trade-off. This final version further improves performance.

4) COMBINING MICROSCOPIC BEHAVIORS

The proposed optimizations for the pairwise test kernel significantly improve frame rate and reduce execution cost compared to the initial version. Leveraging these performance gains, the model incorporates a more complex collision avoidance rule as an alternative correction to the pairwise test. Since the existing potential field global planner tends to group agents into lanes, the new rule complements this behavior. Additionally, it aims to smooth out crossover interactions between agents with opposing motion directions. By considering the current velocity of each agent, the new rule produces a more refined collision avoidance movement that factors in the direction of movement. This strafe, or ''dodge-like'', behavior is achieved by adapting the existing repulsive force: the relative movement between agents determines a corrective force that alters their directions, shifting one agent to the right and the other to the left (Fig. 9). This new strategy is implemented alongside the original pairwise test correction in a ''wrapper'' fashion. This modular approach allows an easy introduction of additional strategies that can be combined or adapted based on the simulation context. These strategies can be used to model and introduce microscopic behaviors within large-scale crowd simulations. Importantly, the optimized pairwise test ensures these refinements do not affect the overall performance, enabling the final simulation to benefit from increased freedom in crowd modeling while maintaining high efficiency.

Leveraging the performance optimizations, the model was extended with another additional layer of crowd behavior to explore its limits and enhance realism. To reduce the determinism of existing techniques, a unique maximum speed was assigned to each agent, chosen randomly within a defined range. This introduces speed variations among individuals (Fig. 1, marked as *(2)*). The results show an overall slowdown due to the variable speeds and the tendency of agents to form lines. The limited prediction of high-density regions, caused by the introduced trade-offs of static potential fields, prevents agents from significantly deviating from the optimal path, leading to congestion in certain areas.

VI. SIMULATION RESULTS

To evaluate performance and classify the model practically, various tests have been performed across different environments and scenarios.

To prove the downward scalability of our solution, the preliminary experiments have been performed using an Apple MacBook Pro 14'' 2021 with Apple Silicon M1 Pro (CPU: 10 cores, GPU: 16 cores), and 32 GB RAM. Then, to evaluate the model's performance on a more powerful setup, we also carried out the experiments using a desktop PC with a CPU Intel i7 6700K, a GPU ASUS Dual RTX 4060Ti 8 GB GDDR6, and 32 GB RAM. In the following sections we discuss the tests performed with the first hardware configuration.

The software has been implemented using Unity3D version 2022.3.4f1. In particular, the development and testing of the project leveraged HLSL for the development of Compute Shaders. Because HLSL is adopted in several development environments and game engines, the proposed model can be easily ported to other platforms, or implemented as a standalone application.

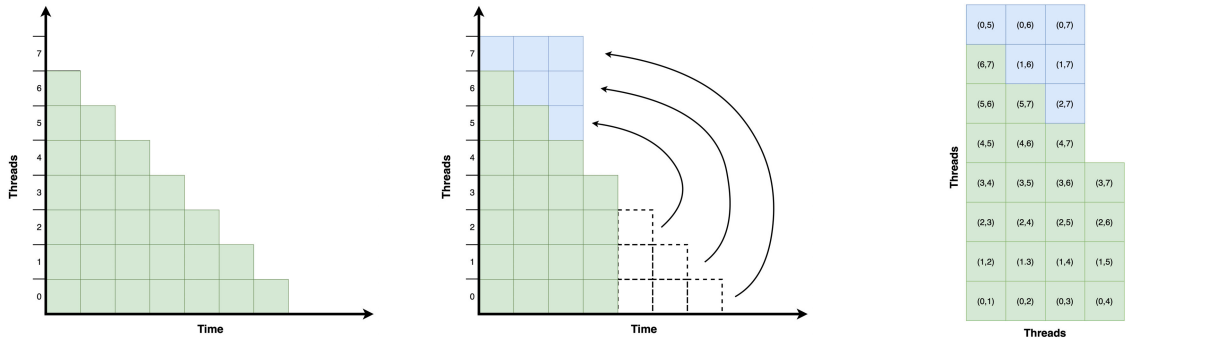
A comprehensive testing procedure was performed to evaluate the proposed crowd simulation model across different scenarios. This assessment aimed to showcase its strengths, and to identify potential limitations to guide us toward future improvements. Testing focused on the following key aspects:

- Environmental complexity: from simple and flat ''terrains'' to difficult and rugged ''terrains'', useful to verify the adaptability of the model to the terrain conditions and their consequences on the speed and paths of agents.
- Obstacle density: sparse and dense obstacle configurations, including structured elements, were employed to assess the effectiveness of obstacle avoidance and global planner-generated paths.
- Scalability with agents count: performance under increasing numbers of agents was measured to understand the behavior of the model in case of large-scale simulations.
- Multi-goal behavior: while not part of the original design, this additional feature was implemented to observe crowd behavior under scenarios with various individual goals.

In the tests, the global planner is set to its static version, and the configuration for the unit cost field weights are:

- Path length $\alpha = 0.157$
- Time required $\beta = 0.884$
- Discomfort $\gamma = 0.94$

This configuration prioritizes paths that avoid elevation changes in the potential field, even if they are not necessarily the shortest. This trade-off between total walking time and obstacle avoidance ensures that the path prioritizes safety while still considering overall efficiency.



(a) Evaluation of pairs per thread over time, without repetition of pairs.

(b) Evaluation of pairs per thread over time, showing how the load is balanced between threads.

(c) Mapping of dispatched threads with their assigned pair of agents (i, j), during pairwise test execution.

FIGURE 8. Optimization steps of the parallel pairwise test.

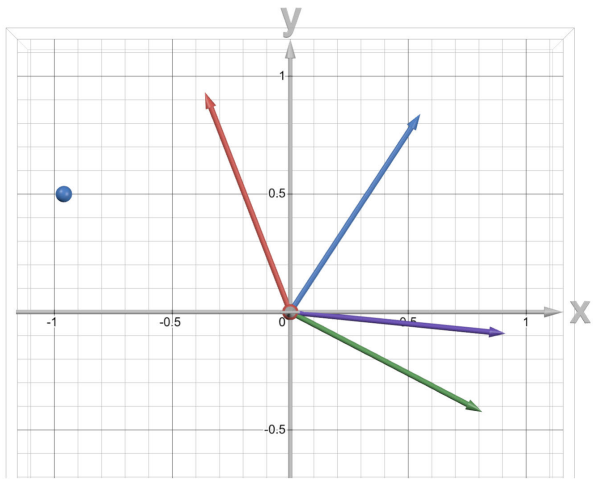


FIGURE 9. Application of the strafe movement strategy for collision avoidance. When an agent (blue dot) violates the minimum interpersonal distance with another agent (grey dot at the origin), the pairwise test calculates a repulsive force (represented by the green vector). This force is then used by the strafe motion strategy to compute a correction vector (purple vector). The correction vector takes into account the current velocity of the evaluated agent (red vector) and results in a final movement that steers the blue agent to its right side (blue vector).

In each experiment we start with an empty map and agents are added in groups of 1,000. After adding a group, when the system reaches a steady state, we measure the computational time required by an update and the frame per seconds observed by the engine. The second performance index is the number of times all agents can be updated every second by the implementation of the simulation model; not to be confused with frames per second on the screen, which is the final result of the rendering pipeline and other computational modules of the game engine. Of course, the higher the value of the frames per second in the engine, the better the visual experience for the user. Once we have both performance information, we iterate with the next group until we reach 500,000 agents; that is, in our case, system saturation.

The supplementary material accompanying this paper includes a video that provides a concise overview of our work. The video delves into the model's architecture and implemented techniques, and showcases a selection of clips from the simulations we conducted.

A. TESTING SCENARIOS

This section details the three implemented scenarios chosen to represent potential real-world applications of our model.

1) HIGHLANDS

The Highlands scenario simulates a large, open area with varied terrain and simple obstacles like cubes and spheres. The terrain size for the tests is set to $2,000 \times 2,000$ meters, and the interpolation factor of the adapting grid resolution is set to $k = 0.5$.

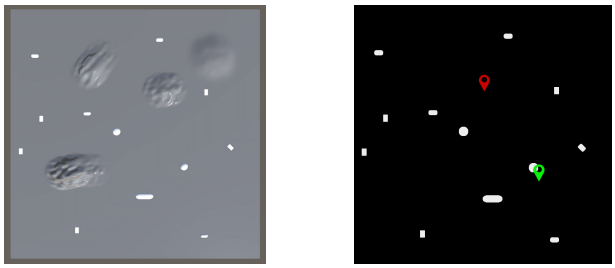
The purpose of this scenario is to evaluate the navigation abilities of an agent, emphasizing effective movement around terrain features and obstacles. A complete view of the scenario and the resulting obstacle field is shown in Fig. 10.

Individuals are guided by a global planner to circumvent elevated areas or safely descend slopes while considering terrain-imposed speed limitations. Obstacles are introduced to assess their impact on path-finding and collision avoidance techniques. The scenario highlights the effectiveness of potential fields in handling sparse obstacles and large maps. Agents navigate around obstacles and elevated terrain by minimizing travel distance and time to assigned goals (Fig. 13).

In a multi-goal setting, multiple potential fields are managed for each goal, enabling seamless navigation (Fig. 14). Up to 500,000 agents can be randomly spawned and assigned to goals during simulation initialization.

2) ARENA

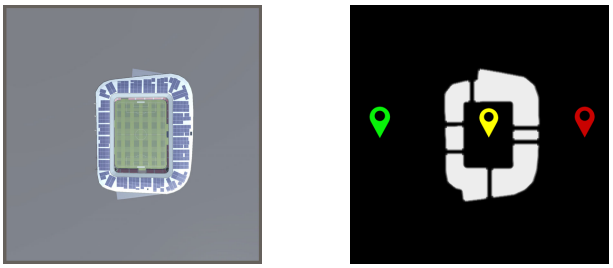
The Arena scenario (Fig. 11), inspired by the Eden Arena in Prague [52], presents a soccer stadium with numerous tunnel-like entrances, and a flat terrain in the center. Mirroring the terrain size of the Highlands scenario ($2,000 \times 2,000$ meters), Arena features a single obstacle but demands higher path-planning precision. To address this, the adaptive grid prioritizes agent count over terrain size, resulting in an



(a) Rendering of the environment.

(b) Resulting obstacle field, and the position of the goal in single-goal (red) and multi-goal (red and green) simulations.

FIGURE 10. Render of the Highlands scenario and positions of the goals in the two simulation modes.



(a) Rendering of the environment.

(b) Resulting obstacle field, and the position of the goal in single-goal (yellow) and multi-goal (red and green) simulations.

FIGURE 11. Render of the Arena scenario and positions of the goals in the two simulation modes.

interpolation factor of $k = 0.722$. This choice leads to higher grid sizes, allowing the model to manage details easier.

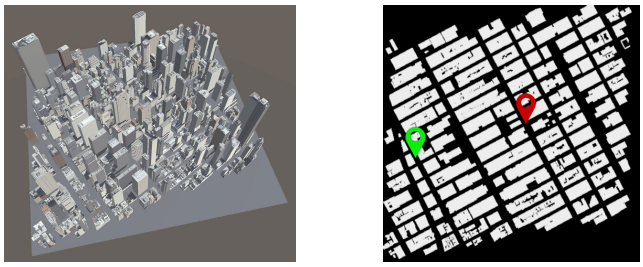
Agents aim to reach the stadium's center (Fig. 15), allowing us to observe their behavior under crowded conditions near the entrances and their path-finding accuracy in confined spaces.

Multi-goal simulations within the stadium environment provided another opportunity to analyze agent interactions during encounters with opposing movement, in particular on how the model handles navigating the narrow passages of the stadium entrances (Fig. 16).

3) MANHATTAN

To assess the model's performance in a more complex and realistic setting, a crowd simulation was conducted within a Manhattan-like scenario [53]. This environment presents a significant challenge due to the high density of obstacles, requiring the model to calculate optimal paths across street intersections and avoid collisions with structures (Fig. 12 and Fig. 17). Terrain size for this scenario is set to $2,025 \times 2,025$ meters, according to the city environment size and distribution of its obstacles. Following the statements for the Arena scenario, the interpolation factor of the adaptive grid resolution is set to $k = 0.759$.

Multi-goal simulations, where two goals are positioned in opposing areas within the environment, demonstrate successful crowd division into lines on each street (Fig. 18). These results underline the effectiveness of the proposed model. Global planning efficiently defines agent paths, while collision avoidance effectively manages interactions between agents moving in opposite directions.



(a) Rendering of the environment.

(b) Resulting obstacle field, and the position of the goal in single-goal (red) and multi-goal (red and green) simulations.

FIGURE 12. Render of the Manhattan scenario and positions of the goals in the two simulation modes.

B. PERFORMANCE ANALYSIS

The model exhibits robustness to scenario complexity and density. As shown in Tab. 1 and in Fig. 19 processing time scales nearly linearly with the number of agents, suggesting efficient handling of large crowds even on consumer-grade hardware. On the laptop test machine, simulations show that our implementation can manage up to 20,000 agents while achieving a stable 60 frames per second or more (Fig. 20). 60 frames per second are recognized as the minimum requirement for fast-paced video games. Lowering this requirement to 30 fps is allowing for more than 30,000 agents to be simulated in real time by our laptop.

A stress tests with 500,000 agents has also been performed. This test yielded an average GPU time of 2445.1 milliseconds per frame. Despite the low frame rate, this result greatly outperforms existing models in literature. A comparative analysis is challenging due to the diversity of approaches and goals among the different models. Even within the same class, models can vary widely, limiting meaningful comparisons. Indeed, crowd simulation models differ in goals, techniques, and testing hardware, making it difficult to find a relevant number of published methods with common features. For comparison, we selected recent works describing models with partial similarities with our method. The hybrid model by Saeed et al. [33] shares similarities with our pairwise algorithm for collision avoidance. The PaCS model by Zhao et al. [54] is a parallel field-based crowd simulation model. Finally, the hybrid model by Sudkhot et al. [55] uses RVO [7] for collision avoidance and supports 3D rendering. The empirical analysis of the performance results provided in the studies indicates that these models require approximately twice the computational time for agent counts in the same

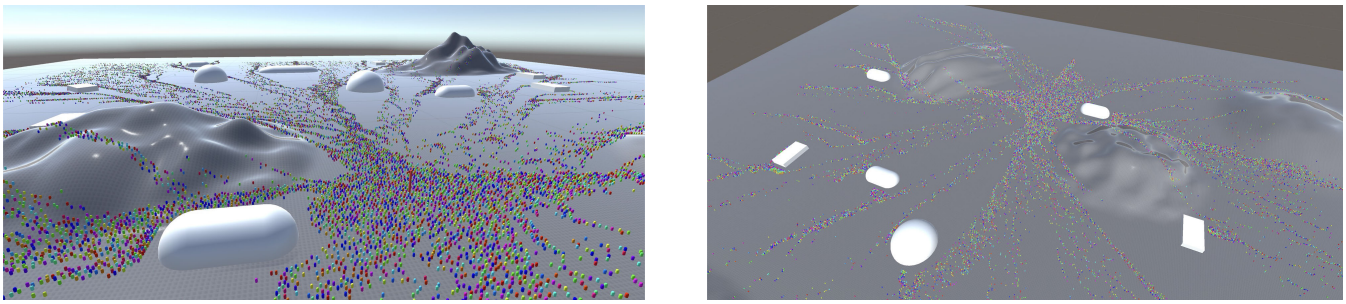


FIGURE 13. Distribution of individuals near the goal in a simulation of 30,000 agents in the Highlands scenario. The position of the assigned goal is represented by the red column. The parameters of the model are set to: $\alpha = 0.157$, $\beta = 0.884$, $\gamma = 0.94$. The average frame-rate achieved during this simulation is 39.5 FPS on the laptop test machine.

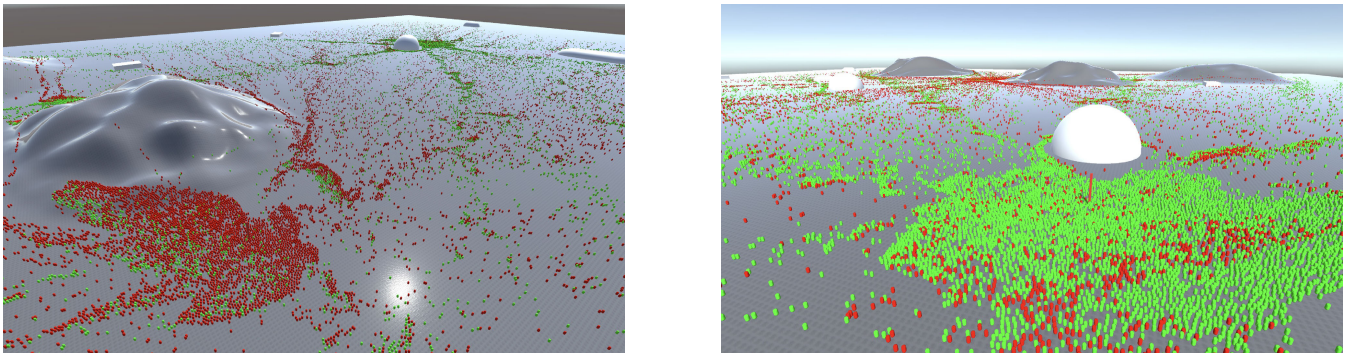


FIGURE 14. Multi-goal simulation with 50,000 agents in the Highlands scenario. The parameters of the model are set to: $\alpha = 0.157$, $\beta = 0.884$, $\gamma = 0.94$. The average frame-rate achieved during this simulation is 21 FPS on the laptop test machine.

TABLE 1. Average results of different simulation instances in the testing scenarios on the laptop test machine.

Agents	Average FPS			Average GPU time (ms)		
	Highlands	Arena	Manhattan	Highlands	Arena	Manhattan
1,000	212.85	228.52	171.82	4.70	4.38	5.82
2,000	208.70	228.05	168.21	4.79	4.39	5.95
3,000	195.43	210.46	166.58	5.12	4.75	6.00
5,000	156.58	165.25	146.74	6.39	6.05	6.81
7,000	132.54	150.93	132.10	7.54	6.63	7.57
10,000	99.47	119.04	89.44	10.05	8.40	11.18
20,000	59.48	62.16	56.24	16.81	16.09	17.78
30,000	39.11	38.03	38.17	25.57	26.30	26.20
50,000	20.85	19.74	20.47	47.96	50.67	48.85
75,000	12.79	12.42	12.86	78.18	80.54	77.77
100,000	7.32	7.16	7.37	136.66	139.68	135.70
200,000	2.17	2.21	2.29	461.20	452.13	436.84
300,000	0.99	1.05	1.09	1013.45	951.69	916.80
500,000	0.39	0.41	0.42	2533.94	2418.89	2382.32

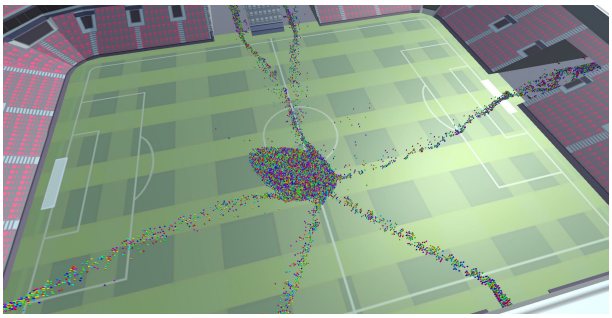
order of magnitude. Both hybrid models and the parallel field-based model exhibit lower performance or rendering challenges.

On a side note, the Manhattan scenario exhibits relatively lower performance with fewer agents due to the denser obstacles and detailed objects that need a more complex rendering stage. However, for simulations with a higher number of agents, the model's behavior in this scenario becomes consistent with the other two test environments.

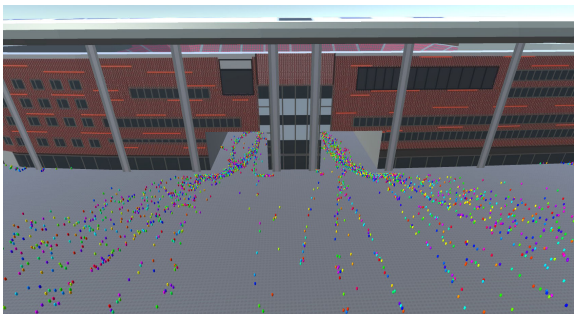
The stress tests conducted on the more powerful desktop machine confirmed the model's linear scaling performance. Fig. 21 compares the processing time for the Highlands scenario: we achieved an additional 50% reduction in processing time compared to the laptop test machine. Similar results were observed on the other scenarios, confirming the extremely positive performance of the proposed model on modern desktop GPUs.

C. DISCUSSION

Observing the experiments unfolding allows also to collect interesting observations about the adopted methodologies. In particular, when heterogeneous crowds are being simulated.

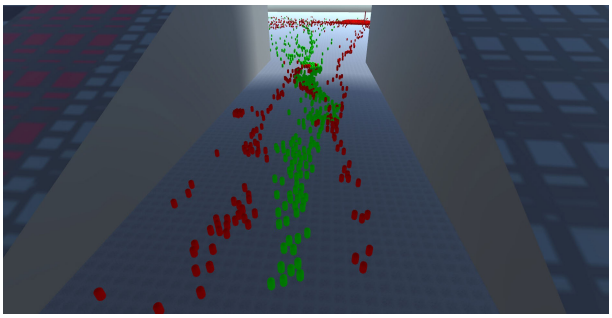


(a) Stadium field and crowd approaching the goal.



(b) Crowd entering the stadium. Within the entrance tunnels, agents show difficulty in using the available space due to spacial approximation of the obstacle field.

FIGURE 15. Simulations of 50,000 agents in the Arena scenario, with model's parameters set to: $\alpha = 0.157$, $\beta = 0.884$, $\gamma = 0.94$. The average frame-rate achieved during this simulation is 19.7 FPS on the laptop test machine.



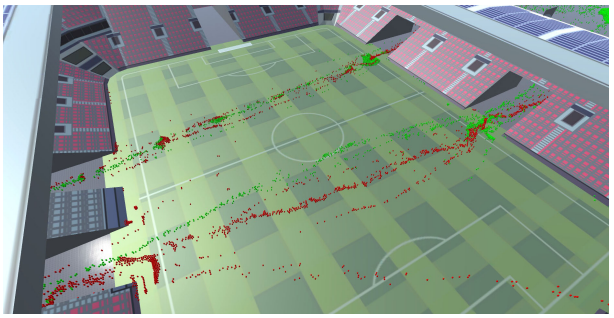
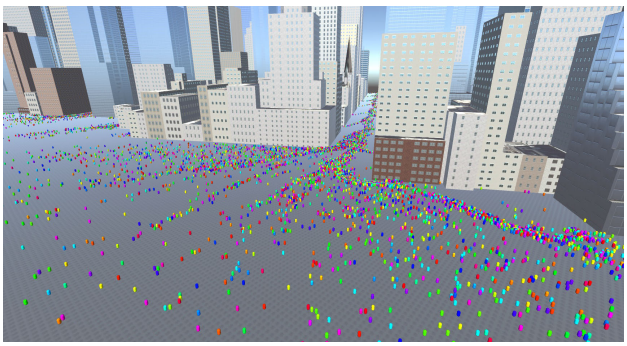
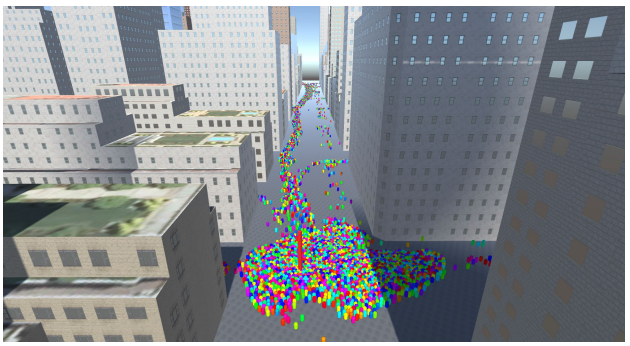


FIGURE 16. Multi-goal simulation of 50,000 agents in the Arena scenario, with model's parameters set to: $\alpha = 0.157$, $\beta = 0.884$, $\gamma = 0.94$. The average frame-rate achieved during this simulation is 20 FPS on the laptop test machine.



(a) Simulation depicting crowd gathering to a narrowing passage.



(b) View close to the goal, showing the large group at the destination and the crowds arriving from a widening.

FIGURE 17. Simulations of 50,000 agents in the Manhattan scenario, with model's parameters set to: $\alpha = 0.157$, $\beta = 0.884$, $\gamma = 0.94$. The average frame-rate achieved during this simulation is 20.6 FPS on the laptop test machine.

Both Potential Fields and Aggregate Dynamics techniques are designed for homogeneous crowds, leading to challenges in managing interactions between different crowds. Specifically, collision avoidance provided by aggregate dynamics becomes dominant in high-density areas, causing significant congestion due to the pressure created by opposing agent movement. It seems that congestion arises near intersections experiencing high traffic volumes. Limiting the correction influence over the Global Planning, with the attempt of preventing complete standstill, helps to resolve the congestion. However, this process remains slow in highly dense areas (Fig. 22). Conversely, in low-density areas, the strafe motion strategy applied by the pairwise test effectively manages interaction, forming bi-directional lines. This, combined with the line formation behavior of the potential fields technique, results in well-defined and structured movement towards goals, with agents visibly arranged in lines (Fig. 23).

While this approach generally produces satisfactory simulations, certain limitations become apparent. The spatial approximation used for obstacle detection leads to

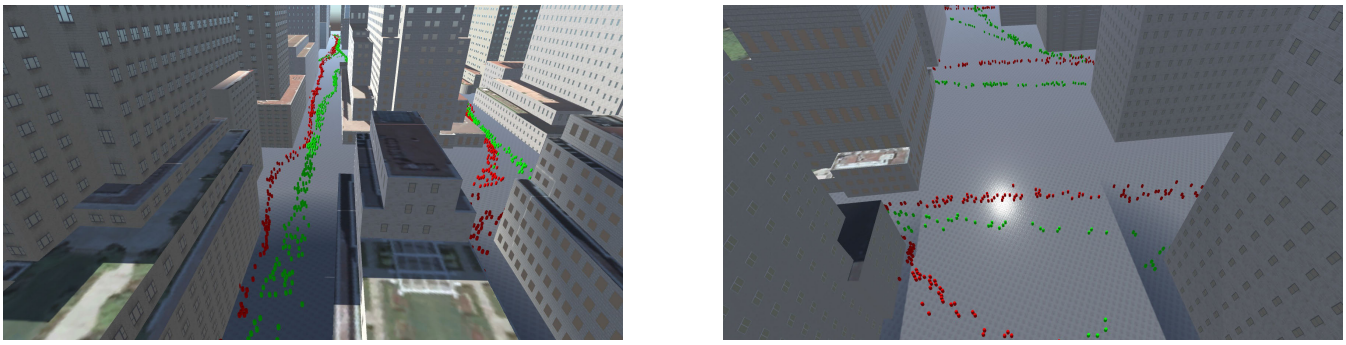


FIGURE 18. Individuals belonging to different goals in Manhattan scenario moving in lines, correctly avoiding collisions and structures. Multi-goal simulation of 50,000 agents, with model's parameters set to: $\alpha = 0.157$, $\beta = 0.884$, $\gamma = 0.94$. The average frame-rate achieved during this simulation is 20.5 FPS on the laptop test machine.

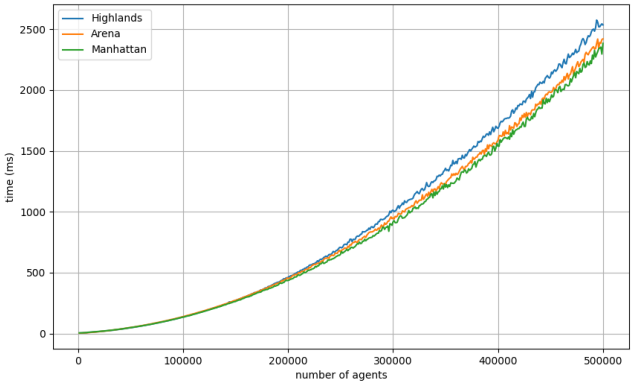


FIGURE 19. Graph of elapsed GPU computational time performance for each scenario on the laptop test machine.

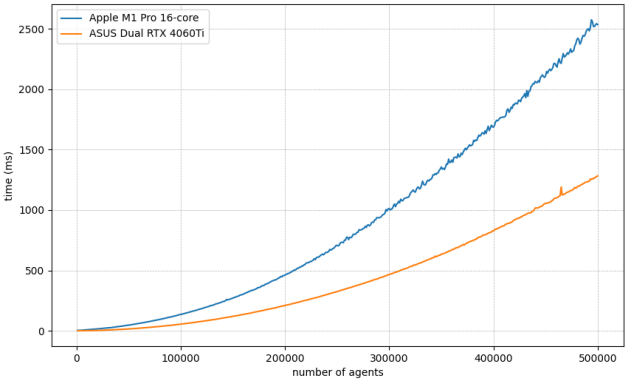


FIGURE 21. Graph of the elapsed frame-rate comparison between the two test machines, conducted on the Highlands scenario.

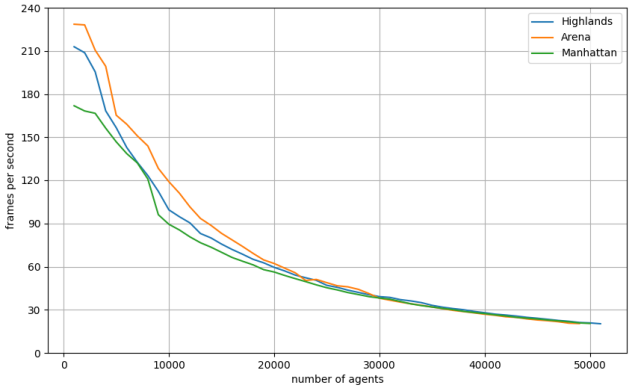


FIGURE 20. Graph of elapsed frame-rate for each scenario on the laptop test machine.

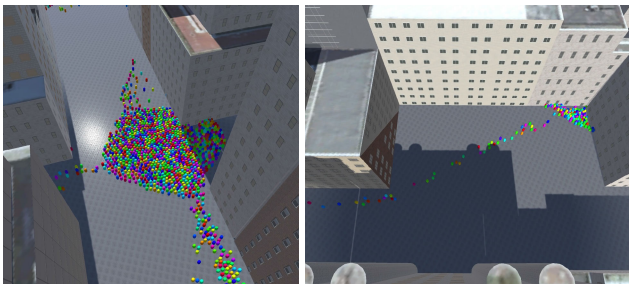


FIGURE 22. Examples of crowd congestion and wrong navigation to dead-end alleys in Manhattan scenario.

inaccuracies when dealing with small details. This results in agents not fully utilizing available space, maintaining unnecessary distances, and contributing to overcrowding (Fig. 15(b)). The obstacle field blur radius, which controls the smoothing distance, partially addresses this issue (see Sec. IV-C3). In Fig. 24 we demonstrate the impact of different spatial approximation configurations on agent behavior. Despite varying grid sizes, agents tend to follow similar paths. While improvements in agent distribution and obstacle avoidance can be achieved, the increased computational cost associated with larger grids does not justify the choice. Future research should focus on enhancing the model's performance, allowing better control over the resulting simulation. The crowd's behavior is primarily influenced by the static potential field, which does not consider crowd density for distribution balancing. Increasing interpersonal distance can potentially improve agent distribution. Additionally, due to the inherent nature of potential field construction and optimization-related approximations, optimal path values may not propagate correctly across the field, leading to dead-end alleys and misguided agent navigation.

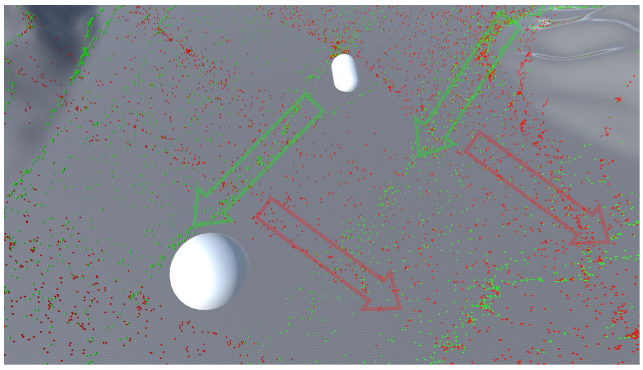
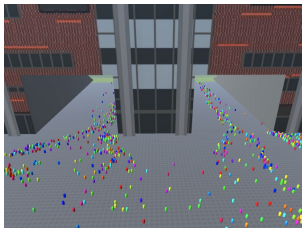
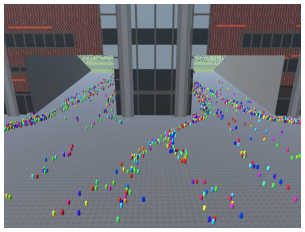


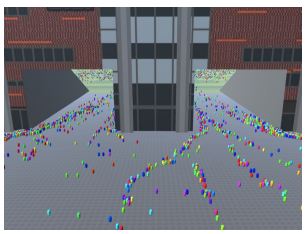
FIGURE 23. Lines formation of agents moving toward their respective goals in the Highlands scenario.



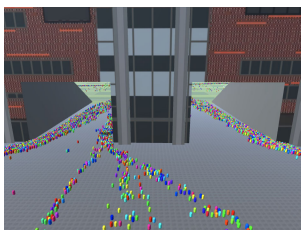
(a) Simulation with grid size set to 300 while the obstacle field blur radius is set to 1 cell. The average frame-rate achieved is 28.6 FPS.



(b) Simulation with grid size set to 800 while the obstacle field blur radius is set to 3 cells. The average frame-rate achieved is 22.6 FPS.



(c) Simulation with grid size set to 1200 while the obstacle field blur radius is set to 5 cells. The average frame-rate achieved is 17.7 FPS.



(d) Simulation with grid size set to 2000 while the obstacle field blur radius is set to 10 cells. The average frame-rate achieved is 10.1 FPS.

FIGURE 24. Simulations in the Arena scenario showing the agents behavior with different configurations for the spatial approximation. The simulations were conducted on the laptop test machine.

Further testing within the Manhattan scenario revealed additional insights about this issue. The path planning of potential fields can be improved in environments with dense obstacles by considering two factors. First, designing environments with obstacles aligned along the axes and diagonals leverages the technique's inherent bias towards movement along these directions. Second, relaxing the time weight in the unit cost field prioritizes obstacle avoidance over path length, ensuring the potential fields effectively propagate values even in complex layouts. An example of the achieved improvements is shown in Fig. 25.

For sake of clarity about the effects of the unit cost field weights, in Fig. 26 we show three different simulations obtained by relaxing each weight. To emphasise the resulting effects, we decided to frame the higher mountain in the

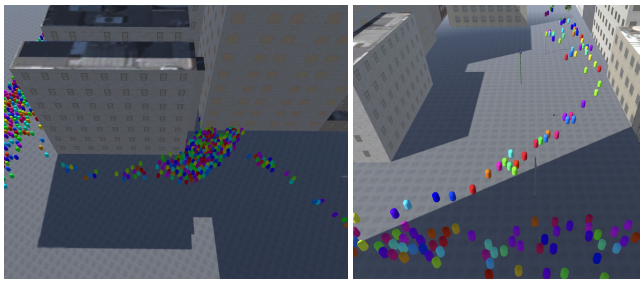


FIGURE 25. Comparison between the first version of the Manhattan scenario (left) and the axis-aligned version (right), showing the achieved improvements in path planning. In the axis-aligned version, the unit cost field parameter related to the required time for paths is set to $\beta = 0.051$.

Highlands scenario. Heights are the most influencing obstacles during the construction of the paths in the potential fields. Their slopes simplifies the visualization of the agents behavior during its descend and how they avoid the heights obstacles. Introducing a time weight relaxing, the goal attainment is prioritized over the obstacle avoidance. Consequently, agents tend to spread more evenly, reducing the time needed to reach their destination. Agents demonstrate a preference for crossing heights rather than completely circumventing them, minimizing directional changes. Conversely, relaxing the path length weight results in simulations where agents are encouraged to prioritize obstacle avoidance. However, this can lead to longer, but more realistic paths. Finally, simulations presenting a discomfort weight relaxing results in a slower agent response to obstacles. The discomfort field data provides most of the information about obstacles and heights. Limiting its influence in potential fields reduce the obstacles forecasting along paths by the agents. For instance, agents placed on the opposite side of hills relative to their goal tend to climb rather than circumnavigate by moving backward.

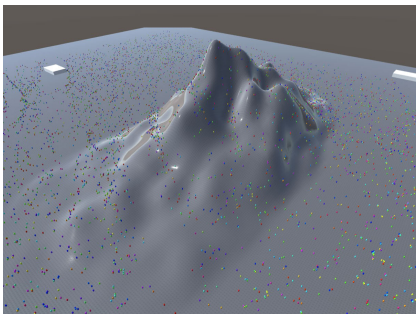
D. ADDING ANIMATIONS IN CROWD SIMULATION

This research prioritizes scalability and efficiency in crowd simulation. While crowd simulation applications often incorporate animations for enhanced realism and visualization, our model aims to cover new ground by introducing motion synthesis capabilities for animated 3D agents. This extension expands the model's potential applications and paves the way to future enhancements of our crowd simulation technology.

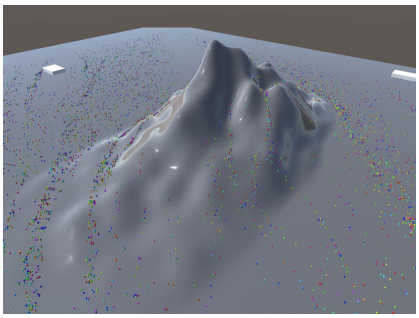
1) VERTEX ANIMATION TEXTURE

Traditionally, animations were managed and updated by the CPU. However, advancements in GPU computational power have enabled GPUs to directly manage animations, improving performance.

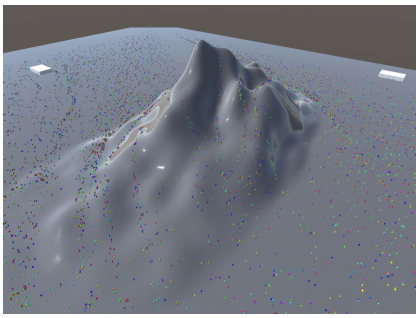
In our model, we adopted a popular animation approach based on *texture baking*: the Vertex Animation Texture (VAT) [56], also known as Morphing Animation. This involves precomputing the model's position at keyframes and storing vertex positions as colors in a texture pixel on the GPU.



(a) Crowd simulation with time weight relaxing.

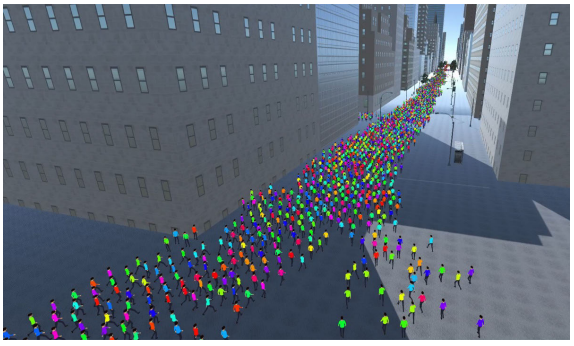


(b) Crowd simulation with path length weight relaxing.



(c) Crowd simulation with discomfort weight relaxing.

FIGURE 26. Simulations performed on the Highlands scenario, using different configurations of unit cost field weights. In these simulations, the tested weight is relaxed by minimizing its value, in contrast to the other weights, which are maximized.



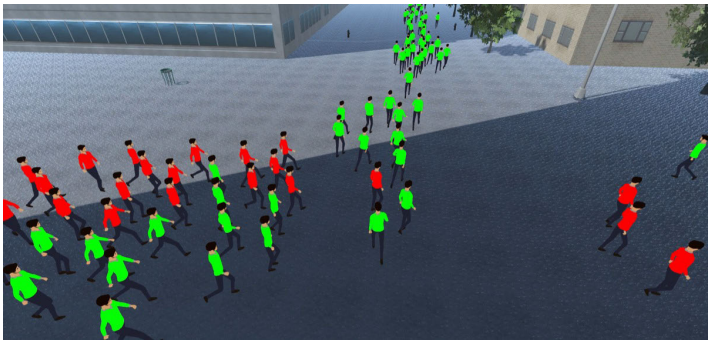


FIGURE 27. Single-goal (left) and multi-goal (right) simulations of 20,000 3D animated agents in the Manhattan scenario (axis-aligned version), with model's parameters set to: $\alpha = 0.148$, $\beta = 0.051$, $\gamma = 1$. The average frame-rate achieved during this simulation is 22.8 FPS on the laptop test machine.

During rendering, the Vertex Shader reads vertex positions from the animation texture based on the requested keyframe.

This technique is highly efficient for scenes with multiple instances of the same 3D model. Crowd simulations with a few 3D models can significantly benefit from this approach.

The complexity of the 3D model also impacts simulation performance. Models with high polygon counts require more rendering and animation time, increasing the computational cost.

2) PERFORMANCE ANALYSIS

We performed stress tests to evaluate the efficiency of our crowd simulation model when incorporating 3D animated agents. The animated agents were rendered using a mesh with 2606 polygons, significantly more complex than the initial 3D capsule model.

Simulations with animated agents are more computationally demanding, but the model successfully produces high-quality results with a large number of agents. While the model currently limits agents to resting or walking animations, introducing motion synthesis capabilities offers promising avenues for future development. This would enable more complex animation management, including shape blending and smoother transitions between animations.

Fig. 27 showcases examples of single and multi-goal simulations in the Manhattan scenario. The supplementary material includes a video focusing on 3D animated simulations, providing a brief overview of the techniques and showcasing clips from the Manhattan scenario. This scenario proved to be the most engaging environment for these tests.

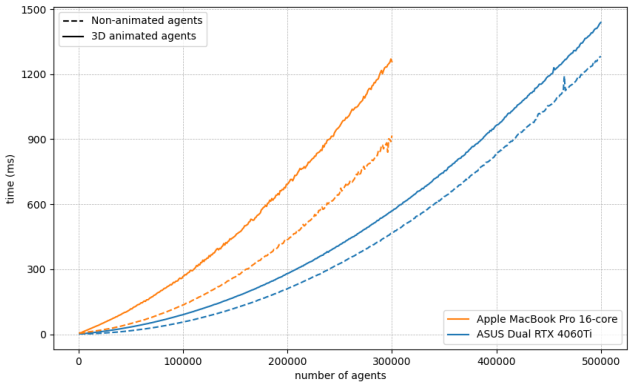


FIGURE 28. Graph of the elapsed GPU computational time comparison between the two test machines, conducted on the Manhattan scenario (axis-aligned version) and using 3D animated models for agents. For purposes of comparison, the results of non-animated simulations were also included.

Fig. 28 compares the processing times for simulations with 3D animated agents in the Manhattan scenario, highlighting the performance differences between the two testing machines. As we can observe from the figure,

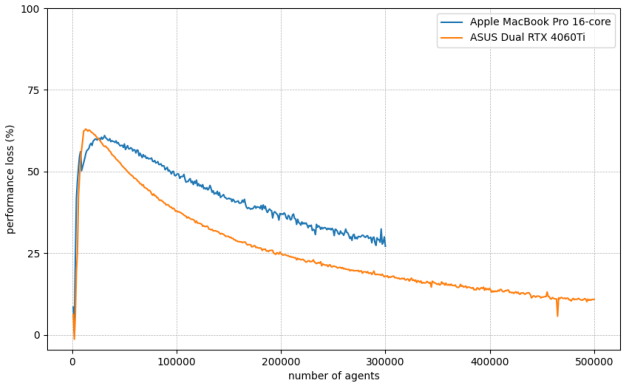


FIGURE 29. Graph of the elapsed performance loss (calculated as the ratio of GPU computational times) between simulations using non-animated and animated 3D models for the agents.

adding animation is contributing with a computational cost that is mostly constant. Finally, in Fig. 29 we show the performance loss between simulations performed on both the test machines, using non-animated models and animated models for the agents. Since the added computational cost of animation is quite constant (as shown in Fig. 28), its relative contribution is more significant with a lower number of agents.

VII. CONCLUSION

This paper introduces a high-performance crowd simulation model utilizing GPU-based parallel computing.

The model primarily focuses on homogeneous crowds due to its core techniques (potential fields and aggregate dynamics). However, promising multi-goal simulation results suggest potential for simulating heterogeneous crowds by introducing diverse goals and individual behaviors. The successful attainment of high performance in simulations with both test machines, comprising an Apple laptop and a desktop PC equipped with a modern GPU, serves to illustrate the scalability of the proposed model on different hardware configurations. The outcomes of this study provide a promising foundation for future developments and extensions. The key to this performance lies in effectively combining macroscopic and microscopic techniques using GPU computation. The proposed custom GPU-based pairwise test enables efficient and accurate agent interaction algorithms. Additionally, the compatibility of macroscopic techniques with the SIMT architecture facilitates efficient utilization of complex solvers.

The efficiency of the proposed model makes it ideal for various applications. In the entertainment industry, it can simulate dynamic crowds in video games (enemies, fleeing civilians) and enhance VFX/CGI for movies (large crowds, armies, etc.). Additionally, for offline rendering with longer processing times, the model allows the easy integration of new rules and strategies to create more complex crowd behaviors. Beyond entertainment, research applications can leverage the model's capabilities. Traffic analysis for optimizing city road systems and studying building/city evacuation plans are two promising areas. While evacuation plan simulations might require further model refinement, this area holds significant potential and it could be a key driver for future model improvements. By leveraging HLSL and Compute Shaders for the GPU implementation of the core model, the proposed methods can be readily adapted and integrated into other game engines and platforms. This design choice enhances the portability of our model and its potential for use in diverse applications.

A. FUTURE WORK

The proposed model exhibits significant promise in its ability to manage animations and more complex meshes. The next steps may focus on extending the motion synthesis feature for animated meshes, leveraging existing high-efficient GPU-based techniques and their compatibility with the model's architecture. In addition to the introduction of multiple meshes for the agents, the capacity to implement a multitude of animations with precise blending to attain a more realistic movement is one of the subsequent stages for the further development of this work.

Pairwise collision avoidance algorithms can be further enhanced. For example, more complex and precise techniques like RVO [7] or ORCA [6] can be adopted. These techniques generate corrective velocities without compromising performance thanks to the proposed optimizations.

Further performance gains can be achieved by optimizing the pairwise test, which represents the more computationally expansive step in the pipeline. For example, a promising approach is the use of space partitioning to divide the crowd into tiles with similar agent counts. This would allow to efficiently exclude irrelevant agents from pairwise calculations, significantly reducing the computational burden. However, implementing space partitioning as a parallel algorithm poses challenges due to its complexity.

Moreover, multi-level fields can be introduced to improve obstacle avoidance and the navigation within tight spaces. These detailed fields provide higher precision in specific areas, with agents switching between them as needed.

The most ambitious and complex future direction is enabling 3D environment handling, such as multi-storey buildings.

REFERENCES

[1] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1160–1168, Jul. 2006, doi: 10.1145/1141911.1142008.

[2] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–8, Dec. 2009, doi: 10.1145/1618452.1618468.

[3] S. Yang, T. Li, X. Gong, B. Peng, and J. Hu, "A review on crowd simulation and modeling," *Graph. Models*, vol. 111, Sep. 2020, Art. no. 101081. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070320300242

[4] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, no. 5, pp. 4282–4286, May 1995, doi: 10.1103/physreve.51.4282.

[5] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, Sep. 2000, doi: 10.1038/35035023.

[6] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal *n*-body collision avoidance," in *Robotics Research*. Heidelberg, Germany: Springer, 2011, pp. 3–19.

[7] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.

[8] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 177–187, doi: 10.1145/1599470.1599494.

[9] T. Weiss, A. Litteneker, C. Jiang, and D. Terzopoulos, "Position-based multi-agent dynamics for real-time crowd simulation (MiG paper)," 2018, *arXiv:1802.02673*.

[10] S.-K. Wong, Y.-H. Chou, and H.-Y. Yang, "A framework for simulating agent-based cooperative tasks in crowd simulation," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games*. New York, NY, USA: Association for Computing Machinery, May 2018, pp. 1–10, doi: 10.1145/3190834.3190839.

[11] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," in *Proc. ACM SIGGRAPH papers*. New York, NY, USA: Association for Computing Machinery, Jul. 2010, pp. 1–9, doi: 10.1145/1833349.1778860.

[12] J. Was, H. Mróz, and P. Topa, "GPGPU computing for microscopic simulations of crowd dynamics," *Comput. Informat.*, vol. 34, no. 6, pp. 1418–1434, Mar. 2016.

[13] G. Payet, "Developing a massive real-time crowd simulation framework on the gpu," M.S. thesis, Univ. Canterbury, Canterbury, U.K., 2016.

[14] K. Lewin, "The research center for group dynamics at Massachusetts institute of technology," *Sociometry*, vol. 8, no. 2, p. 126, May 1945. [Online]. Available: http://www.jstor.org/stable/2785233

[15] K. L. Dion, "Group cohesion: From 'field of forces' to multidimensional construct.," *Group Dyn., Theory, Res., Pract.*, vol. 4, no. 1, pp. 7–26, Mar. 2000, doi: 10.1037/1089-2699.4.1.7.

[16] Q. Ji, F. Wang, and T. Zhu, "VPBS: A velocity-perception-based SFM approach for crowd simulation," in *Proc. Int. Conf. Virtual Reality Visualizat. (ICVRV)*, Sep. 2016, pp. 317–324.

[17] I. Karamouzas and M. Overmars, "Simulating and evaluating the local behavior of small pedestrian groups," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 3, pp. 394–406, Mar. 2012.

[18] J. Bruneau, A.-H. Olivier, and J. Pettré, "Going through, going around: A study on individual avoidance of groups," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 4, pp. 520–528, Apr. 2015.

[19] B. Ulicny, P. D. H. Ciechomski, and D. Thalmann, "Crowdbrush: Interactive authoring of real-time crowd scenes," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation (SCA)*, 2004, p. 243, doi: 10.1145/1028523.1028555.

[20] M. Xu, Y. Wu, Y. Ye, I. Farkas, H. Jiang, and Z. Deng, "Collective crowd formation transform with mutual information–based runtime feedback," *Comput. Graph. Forum*, vol. 34, no. 1, pp. 60–73, Aug. 2014, doi: 10.1111/cgf.12459.

[21] S. Takahashi, K. Yoshida, T. Kwon, K. H. Lee, J. Lee, and S. Y. Shin, "Spectral-based group formation control," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 639–648, Mar. 2009, doi: 10.1111/j.1467-8659.2009.01404.x.

[22] J. S. Wiggins, *The Five-Factor Model of Personality: Theoretical Perspectives*. New York, NY, U.S.A: Guilford Press, 1996.

[23] H. J. Eysenck, *Dimensions of Personality*. Oxford, England: Kegan Pau, 1947.

[24] A. Ortony, G. L. Clore, and A. Collins, *The Cognitive Structure of Emotions*. Cambridge, U.K.: Cambridge Univ. Press, 1988.

[25] A. Mehrabian, "Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament," *Current Psychol.*, vol. 14, no. 4, pp. 261–292, Dec. 1996, doi: 10.1007/bf02686918.

[26] S. J. Guy, S. Kim, M. C. Lin, and D. Manocha, "Simulating heterogeneous crowd behaviors using personality trait theory," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*. New York, NY, USA: Association for Computing Machinery, Aug. 2011, pp. 43–52, doi: 10.1145/2019406.2019413.

[27] S. Kim, S. J. Guy, D. Manocha, and M. C. Lin, "Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games*. New York, NY, USA: Association for Computing Machinery, Mar. 2012, pp. 55–62, doi: 10.1145/2159616.2159626.

[28] M. Xu, X. Xie, P. Lv, J. Niu, H. Wang, C. Li, R. Zhu, Z. Deng, and B. Zhou, "Crowd behavior simulation with emotional contagion in unexpected multihazard situations," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 3, pp. 1567–1581, Mar. 2021.

[29] R. L. Hughes, "A continuum theory for the flow of pedestrians," *Transp. Res. B, Methodol.*, vol. 36, no. 6, pp. 507–535, Jul. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261501000157

[30] S. Patil, J. van den Berg, S. Curtis, M. C. Lin, and D. Manocha, "Directing crowd simulations using navigation fields," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 2, pp. 244–254, Feb. 2011.

[31] Y. Yuan, B. Goñi-Ros, H. H. Bui, W. Daamen, H. L. Vu, and S. P. Hoogendoorn, "Macroscopic pedestrian flow simulation using smoothed particle hydrodynamics (SPH)," *Transp. Res. C, Emerg. Technol.*, vol. 111, pp. 334–351, Feb. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X19310125

[32] 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," *Comput. Graph.*, vol. 98, pp. 306–321, Aug. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849321001205

[33] R. A. Saeed, D. R. Recupero, and P. Remagnino, "Simulating crowd behaviour combining both microscopic and macroscopic rules," *Inf. Sci.*, vol. 583, pp. 137–158, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521011488

[34] J. Skrzypczak and P. Czarnul, "Efficient parallel implementation of crowd simulation using a hybrid CPU+GPU high performance computing system," *Simul. Model. Pract. Theory*, vol. 123, Feb. 2023, Art. no. 102691. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22001605

[35] A. Da Silva Antonitsch, D. H. M. Schaffer, G. W. Rockenbach, P. Knob, and S. R. Musse, "Bioclouds: A multi-level model to simulate and visualize large crowds," in *Advances in Computer Graphics*, M. Gavrilova, J. Chang, N. M. Thalmann, E. Hitzer, and H. Ishikawa, Eds., Cham, Switzerland: Springer, 2019, pp. 15–27.

[36] A. D. L. Bicho, R. A. Rodrigues, S. R. Musse, C. R. Jung, M. Paravisi, and L. P. Magalhães, "Simulating crowds based on a space colonization algorithm," *Comput. Graph.*, vol. 36, no. 2, pp. 70–79, Apr. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849311001713

[37] A. Malinowski and P. Czarnul, "Multi-agent large-scale parallel crowd simulation with NVRAM-based distributed cache," *J. Comput. Sci.*, vol. 33, pp. 83–94, Apr. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877750318307221

[38] J. Stam, "Stable fluids," in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn.-SIGGRAPH*. Boston, MA, USA: Addison-Wesley, 1999, pp. 121–128, doi: 10.1145/311535.311548.

[39] M. Harris, "Fast fluid dynamics simulation on the GPU," in *Proc. ACM SIGGRAPH Courses*. New York, NY, USA: Association for Computing Machinery, 2005, p. 220, doi: 10.1145/1198555.1198790.

[40] K. Crane, I. Llamas, and S. Tariq, "Real-time simulation and rendering of 3D fluids," in *GPU Gems 3*, H. Nguyen, Ed., Boston, MA, USA: Addison-Wesley, 2007, ch. 30.

[41] R. Kimmel and J. A. Sethian, "'Optimal algorithm for shape from shading and path planning," *J. Math. Imag. Vis.*, vol. 14, no. 3, pp. 237–244, May 2001, doi: 10.1023/a:1011234012449.

[42] G. Lu, L. Chen, and W. Luo, "Real-time crowd simulation integrating potential fields and agent method," *ACM Trans. Model. Comput. Simul.*, vol. 26, no. 4, pp. 1–16, Mar. 2016, doi: 10.1145/2885496.

[43] T. Heidmann, "Real shadows real time," *Iris Universe*, vol. 18, pp. 28–31, 1989. [Online]. Available: https://cir.nii.ac.jp/crid/1570009750005293312

[44] H. Bruns, *Das Eikonal*, vol. 21, no. 5. Leipzig: S. Hirzel, 1895.

[45] S. K. Godunov and I. Bohachevsky, "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics," *Matematičeskij sbornik*, vol. 47, no. 3, pp. 271–306, 1959.

[46] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Trans. Autom. Control*, vol. 40, no. 9, pp. 1528–1538, Sep. 1995.

[47] W.-K. Jeong and R. T. Whitaker, "A fast iterative method for eikonal equations," *SIAM J. Sci. Comput.*, vol. 30, no. 5, pp. 2512–2534, Jan. 2008.
[48] Y. Huang, "Improved fast iterative algorithm for eikonal equation for GPU computing," 2021, *arXiv:2106.15869*.
[49] Z. Dostál and J. Schöberl, "Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination," *Comput. Optim. Appl.*, vol. 30, no. 1, pp. 23–43, Jan. 2005, doi: 10.1007/s10589-005-4557-7.
[50] A. J. Chorin, J. E. Marsden, and A. Leonard. (1979). *A Mathematical Introduction To Fluid Mechanics*. [Online]. Available: https://api.semanticscholar.org/CorpusID:119966220
[51] G. Amador and A. Gomes, "Linear solvers for stable fluids: GPU vs CPU," in *Proc. 17th EncontroPortugues de ComputacaoGrafica (EPCG09)*, A. Coelho and A. P. Cláudio, Eds., 2021, pp. 1–10.
[52] *Arena Scenario*. Accessed: Apr. 19, 2024. [Online]. Available: https://sketchfab.com/3d-models/eden-arena-8f1caa3250424e8e8769a405f1194468
[53] Geopipe, Inc. *Manhattan Scenario*. Accessed: Apr. 19, 2024. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/urban/real-new-york-city-vol-1-208247
[54] H. Zhao, T. Guo, W. Tong, H. Yin, and Z. Liu, "PaCS: A parallel computation framework for field-based crowd simulation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 12659–12670, Nov. 2023.
[55] P. Sudkhot, K. W. Wong, and C. Sombattheera, "Collision avoidance and path planning in crowd simulation," *ICIC Exp. Lett.*, vol. 17, no. 1, pp. 13–24, 2022.
[56] L. O. Vasconcelos and A. Sato. (2020). *Texture Animation: Applying Morphing and Vertex Animation Techniques*. [Online]. Available: https://medium.com/tech-at-wildlife-studios/texture-animation-techniques-1daecb316657

VINCENZO LOMBARDO received the B.S. degree in computer science from the University of Palermo, in 2020, and the M.S. degree in computer science from the University of Milan, in 2024, where he is currently pursuing the Ph.D. degree. In recent years, he worked on developing GPU-accelerated simulation models for various physical phenomena (e.g., fluid simulation), manipulation of computer graphics meshes, VR applications, and procedural content generation for video games. His research interests include GPU computing, video game programming, real-time computer graphics, and distributed systems.

DAVIDE GADIA received the M.Sc. and Ph.D. degrees in computer science from the University of Milan, Italy, in 2003 and 2007, respectively. He is currently an Associate Professor with the Department of Computer Science, University of Milan. He is with the Playlab fOr inNovation in Games (PONG) Laboratory. His research interests include video game programming, procedural content generation for computer graphics and video games, game engine architectures, and VR.

DARIO MAGGIORINI received the M.S. and Ph.D. degrees in computer science from the University of Milan, in 1997 and 2002, respectively. He is currently an Associate Professor with the Department of Computer Science, University of Milan, and the Co-Founder of the PONG Playlab fOr inNovation in Games (PONG) Laboratory. In the past, he performed research on QoS for multi-service IP-based networks, large network services scalability, multimedia transmission, mobile services, and opportunistic networks. Since 2011, he has been focusing his expertise on research for novel distributed architectures for entertainment applications.

Open Access funding provided by 'Università degli Studi di Milano' within the CRUI CARE Agreement