Extreme-Density Crowd Simulation: Combining Agents with Smoothed Particle Hydrodynamics

Wouter van Toll
wouter.van-toll@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

Cédric Braga
cedric.de-almeida-braga@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

Barbara Solenthaler
solenthaler@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Julien Pettré
julien.pettre@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

ABSTRACT

In highly dense crowds of humans, collisions between people occur often. It is common to simulate such a crowd as one fluid-like entity (macroscopic), and not as a set of individuals (microscopic, agent-based). Agent-based simulations are preferred for lower densities because they preserve the properties of individual people. However, their collision handling is too simplistic for extreme-density crowds. Therefore, neither paradigm is ideal for all possible densities.

In this paper, we combine agent-based crowd simulation with the concept of Smoothed Particle Hydrodynamics (SPH), a particle-based method that is popular for fluid simulation. Our combination augments the usual agent-collision handling with fluid dynamics when the crowd density is sufficiently high. A novel component of our method is a dynamic rest density per agent, which intuitively controls the crowd density that an agent is willing to accept.

Experiments show that SPH improves agent-based simulation in several ways: better stability at high densities, more intuitive control over the crowd density, and easier replication of wave-propagation effects. Our implementation can simulate tens of thousands of agents in real-time. As such, this work successfully prepares the agent-based paradigm for crowd simulation at all densities.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; **Real-time simulation**; **Motion path planning**; **Intelligent agents**.

KEYWORDS

crowd simulation, dense crowds, smoothed particle hydrodynamics

ACM Reference Format:
Wouter van Toll, Cédric Braga, Barbara Solenthaler, and Julien Pettré. 2020. Extreme-Density Crowd Simulation: Combining Agents with Smoothed Particle Hydrodynamics. In *Motion, Interaction and Games (MIG '20), October 16–18, 2020, Virtual Event, SC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3424636.3426896

MIG '20, October 16–18, 2020, Virtual Event, SC, USA

ACM ISBN 978-1-4503-8171-0/20/10…$15.00
https://doi.org/10.1145/3424636.3426896

1 INTRODUCTION

The real-time simulation of human crowds has many applications, from computer games to safety-critical crowd studies. With an increasing number of crowded events occurring worldwide, there is an increasing desire to simulate *dense* crowds in particular. Simulation can help understand the behavior of such crowds, and it can help predict and prevent dangerous situations.

In this paper, we consider crowds of *extreme density*, containing 4 people per square meter (P/m^2) or more. These densities can be observed on specific occasions such as concerts, pilgrimage gatherings, and evacuations. At extreme densities, there is ample physical contact between people, the crowd's motion bears similarities to the motion of fluids [Hughes 2003], and new types of collective motion such shockwaves may occur [Bottinelli and Silverberg 2018].

For these reasons, most simulations of extreme-density crowds are *macroscopic*: they model the crowd as one fluid-like continuum, without considering the properties and goals of individual people. These models are less suitable for lower densities where individuality is required. Also, their grid representation of the environment limits their scalability to large or complex geometry.

By contrast, *microscopic* (or *agent-based*) methods simulate each person as an individual agent. This is preferred for lower densities where people can follow their own plans. However, agent-based models try to *avoid collisions* between agents. Any collisions that do occur are resolved simplistically, e.g. by treating agents as disks and applying contact forces when they overlap. At extreme densities where collisions are inevitable, the simulation is determined mostly by these contact forces. If the forces are too strong, the agents' disks are incompressible, and the simulation can easily reach a deadlock or become unstable. If the forces are too weak, the disks become *too* compressible, which can result in unrealistically high densities.

1.1 Motivation: Combining Agents and SPH

It is common to choose either microscopic or macroscopic simulation depending on the scenario: there is not one ideal simulation method for all crowd densities. In this paper, we aim to combine the advantages of both paradigms, by enriching agent-based simulation with *Smoothed Particle Hydrodynamics (SPH)* [Gingold and Monaghan 1977] to improve the crowd's behavior at high densities.

SPH is a simulation method in which *particles* move according to forces caused by differences in density and velocity. It is popular in the computer-graphics community for the simulation of fluids. SPH uses the same physical principles as macroscopic methods, but it discretizes the continuum into particles. This is easy to combine with agent-based crowd simulation. If we treat each agent as an SPH particle, SPH can yield 'fluid-like' behavior at extreme densities, without sacrificing the individuality of agents at lower densities. The combination of agents and SPH also intuitively handles *transitions* between low and high densities, without the need to switch between microscopic and macroscopic algorithms. Finally, because the concept of density in SPH translates intuitively to *crowd density*, SPH can prevent the crowd density from becoming too high.

In short, SPH can *augment* agent-based crowd simulation to handle extremely dense scenarios where contact forces do not suffice. In this paper, we present this augmentation, we demonstrate its effectiveness in two example scenarios, and we analyze the effect of parameters on both behavior and performance.

1.2 Goals and Contributions

The main contributions of this paper are the following:

- In Section 4, we show how to integrate SPH into any agent-based crowd simulation. We also extend standard SPH with the concept of a dynamic rest density per particle.
- In Section 6, we show via experiments that SPH can improve the behavior of agent-based crowd simulation at extreme densities. SPH makes the simulation more stable, it gives better control over the crowd density, and it can reproduce fluid-like wave effects at various densities. We also show that contact forces and a dynamic rest density make the standard SPH model more suitable for crowd simulation. Our implementation can simulate large crowds in real-time.

Of course, we are not the first to apply fluid-simulation concepts to crowd simulation; see Section 2.2 for an overview. However, to our knowledge, this is the first paper that generically *integrates* SPH into the agent-based simulation paradigm (using contact forces and personal rest densities), and the first that applies it to *extremely dense* crowds of ≥ 4 P/m^2. Furthermore, we thoroughly analyze the system's parameter settings and real-time performance.

2 RELATED WORK

Crowd simulation is an active research topic containing aspects of motion planning, computer animation, artificial intelligence, and more. Several books give a good overview of the field [Pelechano et al. 2016; Thalmann and Musse 2013].

2.1 Agent-based Crowd Simulation

Agent-based crowd-simulation algorithms model each person in the crowd as an intelligent agent with individual properties and goals. This paradigm subdivides the navigation task of an agent into multiple 'levels', such as planning a global path to the goal, following this path smoothly, and avoiding collisions with other agents. Various efficient implementations of this 'multi-level' concept exist [Curtis et al. 2016; Kielar et al. 2016; van Toll et al. 2015].

A particularly popular research topic is *collision avoidance*: ensuring that all agents move towards their goals without colliding. To solve this problem efficiently, it is common to approximate agents by disks. Most collision-avoidance algorithms are based on forces [Helbing and Molnár 1995; Karamouzas et al. 2014], velocity selection [Guy et al. 2010; van den Berg et al. 2011], or vision [Dutra et al. 2017]. They usually assume that collisions can indeed be avoided. Some algorithms prevent collisions by letting agents stop when necessary [Pelechano et al. 2007]. Others accept that collisions may occur, and they resolve them via contact forces between the agents' disk representations. The popular contact-force model by Helbing et al. [2000] suffices for most applications. However, we will show that it is not ideal for extremely dense crowds.

To improve agent-based simulation of dense crowds, researchers have looked into more detailed models of contact between individuals. Kim et al. [2015] have added physical interaction forces (such as pushing) to simulate a crowded pilgrimage scenario. Hesham and Wainer [2017] have given agents a dynamic *personal space* that adapts to their current speed. Stüvel et al. [2016] focused on *navigation* through dense crowds, using a Voronoi diagram with capsule-shaped agents as its sites. While successful, their model is too involved to simulate very large crowds in real-time.

In this paper, we let agents experience forces computed by a fluid-mechanics model (SPH). These forces can easily coexist with the concepts of personal space and pushing. However, we will show that SPH alone can already reach interesting results.

2.2 Dense Crowds and Fluid Mechanics

In contrast to microscopic algorithms, *macroscopic* algorithms simulate the crowd as a whole [Hughes 2003; Maury et al. 2010], using a grid in which each cell prescribes the optimal walking speed and direction. Popularized by Treuille et al. [2006], the concept has been successfully used to simulate extreme-density crowds [Narain et al. 2009] and turbulence effects [Golas et al. 2014]. These extensions are based on the laws of fluid mechanics, motivated by the popular belief that dense crowds bear similarities to fluids [Hughes 2003].

Macroscopic methods simulate crowd dynamics 'from above' without considering the differences between people. This is especially suitable for high-density crowds. At lower densities, though, *microscopic* simulation is preferred because it can model the properties and goals of each person (agent). Furthermore, because macroscopic methods use a (dense) grid overlay, they are less scalable to large environments. We are therefore interested in applying 'macroscopic-like' physics equations to the agent-based paradigm, to make agent-based simulation suitable for all crowd densities.

The Smoothed Particle Hydrodynamics (SPH) model meets this demand. Originally developed for astrophysics simulations [Gingold and Monaghan 1977; Lucy 1977], SPH has become a popular fluid-simulation method for computer graphics, starting with the work by Müller et al. [2003]. A recent tutorial by Koschier et al. [2019] gives a good overview of SPH and its implementation considerations.

Contrarily to macroscopic methods, SPH simulates a substance as a set of moving *particles*. Each particle experiences forces caused by the density and velocity around its current position. We will show that this concept is ideal for simulating extreme-density crowds.

SPH has been suggested before for agent-based crowd simulation [Tantisiriwat et al. 2007; Vetter et al. 2011], but only as an alternative

for collision avoidance at low densities, and not as a way to model the fluid-like behavior of dense crowds. Also, these publications do not discuss parameter settings or computational efficiency.

Weiss et al. [2017] have simulated crowds using Position-Based Dynamics (PBD), an alternative to SPH that directly controls the positions of particles instead of applying forces. However, using this for crowds requires e.g. collision-avoidance algorithms to be reformulated. SPH can be plugged into an existing crowd simulation more easily. Weiss et al. also did not study extremely dense crowds.

Closest to our work, Yuan et al. [2020] have recently shown the potential of SPH for crowd simulation in more detail. However, they studied the crowd purely macroscopically, without considering the properties of individual agents. They also did not compare or combine SPH with contact forces, and they did not consider extreme densities. By contrast, we fit SPH into the agent-based paradigm (thus maintaining individuality), and we study extremely dense crowds where both SPH and contact forces are important.

Compared to all previous work that combines SPH and crowd simulation, we focus on extreme-density crowds, we provide more insight into parameter settings, and we assess the system's computational performance. We also extend SPH with a dynamic rest density per particle, to make the model more suitable for crowds.

3 SMOOTHED PARTICLE HYDRODYNAMICS

This section repeats the foundations of Smoothed Particle Hydrodynamics (SPH), based on the version by Müller et al. [2003]. These equations work in both 3D and 2D, but it is useful to remember that we will apply them to a 2D domain. Our particles will represent individual people and the environment will be a 2D plane.

3.1 Overview

In general, the goal of fluid simulation is to accurately approximate the dynamics of a fluid. These are given by the *Navier-Stokes equations*, such as the following for conservation of momentum:

$$\rho \frac{D\mathbf{v}}{Dt} = \rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{G} \tag{1}$$

Here, μ is the viscosity of the fluid, and $\mathbf{v}$ (velocity), ρ (density), and p (pressure) are continuous fields with values that change over time. The equation depends on the *pressure force* $-\nabla p$ caused by differences in pressure, the *viscosity force* $\mu\nabla^2\mathbf{v}$ caused by fluctuations in velocity, and *external forces* $\mathbf{G}$ such as gravity.

A simulation method needs to discretize these continuous fields. While macroscopic methods divide the *environment* into grid cells and compute the required values per cell, SPH divides the *fluid* into particles and computes the values per particle. (These two approaches are respectively referred to as Eulerian and Lagrangian.) In SPH, a particle with index i has a mass m_i and a variable position $\mathbf{r}_i$ and velocity $\mathbf{v}_i$. At any point in time, assume that the density $\rho(\mathbf{r})$ and pressure $p(\mathbf{r})$ can be computed for any position $\mathbf{r}$. Let $\rho_i = \rho(\mathbf{r}_i)$ and $p_i = p(\mathbf{r}_i)$ be the current density and pressure at the position of particle i. The acceleration $\mathbf{a}_i$ of the particle is then:

$$\mathbf{a}_i = \frac{-\nabla p_i + \mu \nabla^2 \mathbf{v}_i}{\rho_i} + \mathbf{G} \tag{2}$$

The details of SPH concern how to compute ρ_i, ∇p_i, and $\nabla^2 \mathbf{v}_i$.

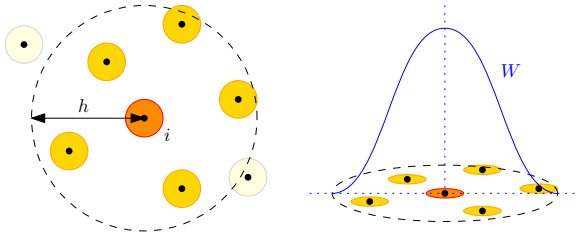


Figure 1: Principle of an SPH kernel. Left: 2D view of a particle i and its neighbors. All particles within the radius h are considered by SPH. Right: Perspective view including the kernel function W. A particle's contribution to any quantity (such as the density) depends on the distance to particle i.

3.2 SPH Approximation Equation

For any quantity A at a position $\mathbf{r}$, SPH defines $A(\mathbf{r})$ as a weighted sum over all particles:

$$A(\mathbf{r}) = \sum_j m_j \frac{A(\mathbf{r}_j)}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \tag{3}$$

In practice, $\mathbf{r}$ will almost always be the position $\mathbf{r}_i$ of a particle i, because we want to compute certain quantities per particle. However, the equation theoretically works for any position.

The *smoothing kernel* $W(\mathbf{r}, h)$ is a weight function that usually decreases as the length of its argument $\mathbf{r}$ increases. This ensures that particles that are farther away have a smaller influence on the quantity being computed. It is possible to use different functions W for different purposes; we will provide the details of this later. The scalar h is the *support radius* of W. It usually indicates that $W(\mathbf{r}, h) = 0$ whenever $\|\mathbf{r}\| \geq h$. Fig. 1 summarizes this principle.

The gradient $\nabla A(\mathbf{r})$ and the Laplacian $\nabla^2 A(\mathbf{r})$ can be obtained from Eq. (3) by replacing W by ∇W and $\nabla^2 W$, respectively.

3.3 Computing Density, Pressure, and Forces

Applying Eq. (3) to the density ρ gives the following equation:

$$\rho(\mathbf{r}) = \sum_j m_j \frac{\rho(\mathbf{r}_j)}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) = \sum_j m_j \cdot W(\mathbf{r} - \mathbf{r}_j, h) \tag{4}$$

Thus, the density at a position $\mathbf{r}$ is a weighted sum of the masses of all particles near $\mathbf{r}$. This holds for the particle densities $\rho_i = \rho(\mathbf{r}_i)$ as well. From ρ_i, we can compute the pressure p_i as

$$p_i = k(\rho_i - \rho^0) \tag{5}$$

where k is a constant (traditionally a physical gas constant that depends on temperature) and ρ^0 is the *rest density* that the simulated substance tries to attain. Note that densities above ρ^0 can still occur, due to external forces $\mathbf{G}$ acting on the system.

With the density and pressure in place, SPH can compute the pressure force $-\nabla p_i$ and the viscosity force $\mu\nabla^2\mathbf{v}_i$, leading to an acceleration $\mathbf{a}_i$ per particle via Eq. (2). To compute $-\nabla p_i$ and $\mu\nabla^2\mathbf{v}_i$, Müller et al. [2003] suggest the following adaptations of Eq. (3):

$$-\nabla p_i = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{6}$$

$$\mu \nabla^2 \mathbf{v}_i = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{7}$$

3.4 Simulation Loop

In a basic SPH simulation loop, each iteration does the following:

(1) For each particle i, find the neighboring particles within distance h of $\mathbf{r}_i$.
(2) For each particle i, compute the density ρ_i (Eq. (4)) and the pressure p_i (Eq. (5)).
(3) For each particle i, compute the forces $-\nabla p_i$ (Eq. (6)) and $\mu\nabla^2\mathbf{v}_i$ (Eq. (7)) and finally the acceleration $\mathbf{a}_i$ (Eq. (2)).
(4) Move all particles using a time-integration method of choice.

It is important to compute all density and pressure values *before* computing any forces, so that all particles use the same information.

This process is repeated over time, usually with a fixed timestep Δt. The next section will show how to integrate this SPH loop into an agent-based crowd-simulation loop.

4 SPH-ENHANCED CROWD SIMULATION

As explained in Section 1, a 'traditional' agent-based crowd simulation is not suitable for extreme densities due to its simplistic collision handling. In this section, we describe how to enrich any agent-based simulation with SPH to improve this aspect. We will keep the discussion as general as possible. Section 5 will describe the implementation and settings used for our experiments.

4.1 Outline of Agent-based Crowd Simulation

An agent-based crowd simulation consists of agents with individual properties and goals. The environment is usually a 2D plane with polygonal obstacles. Each agent i has a current position $\mathbf{r}_i$, a current velocity $\mathbf{v}_i$, and a goal position $\mathbf{g}_i$, and it tries to reach this goal (possibly by following a global path) while avoiding collisions with other agents and obstacles. For efficient collision prediction, each agent is approximated by a disk. Next to collision avoidance, agents may also want to satisfy other navigation criteria, such as following a path, staying in a group with other agents, and possibly more.

Generally, let us say that the navigation criteria per agent are combined into an overall *navigation policy* $\mathcal{P}_i$. At any point in time, the purpose of $\mathcal{P}_i$ is to compute an acceleration vector $\mathbf{a}_i$ that will move agent i while satisfying all navigation criteria as well as possible. In a classical crowd simulation, $\mathcal{P}_i$ consists of at least path following and collision avoidance. Each agent can have its own policy: this is a key advantage of agent-based simulation. This leads to a simulation loop where each iteration does the following:

(1) For each agent i, find the neighboring agents and obstacles within a certain distance of interest.
(2) For each agent i, use the navigation policy $\mathcal{P}_i$ to compute an acceleration $\mathbf{a}_i$, keeping the agent's neighbors in mind.
(3) For each agent i, compute the contact forces $\mathbf{F}_i^c$ imposed by all agents and obstacles that are currently colliding with i. Update the acceleration as $\mathbf{a}_i := \mathbf{a}_i + \mathbf{F}_i^c/m_i$, where m_i is the agent's mass (which is usually set to 1).
(4) Move all particles using a time-integration method of choice.

Step 2 may consist of several smaller steps depending on the complexity of the policy. These steps could include finding a preferred velocity $\mathbf{v}_{\text{pref}}$ that sends the agent forward along a global path, finding a velocity close to $\mathbf{v}_{\text{pref}}$ that avoids collisions, and so on. These details do not matter in our discussion.

4.2 Adding SPH to the Simulation

To integrate SPH into this crowd-simulation loop, the idea is to treat each agent as an SPH particle (in 2D), and to add SPH forces as additional sources of acceleration per agent. An equivalent interpretation is to see the simulation as SPH, but with an 'external force' $\mathbf{G}$ (see Eq. (2)) that is particle-dependent and that is produced by an agent's navigation policy.

4.2.1 Updated Simulation Loop. To add SPH to the simulation loop of Section 4.1, the following should be added *between* steps 3 and 4:

(a) For each agent i, compute the SPH density ρ_i (Eq. (4)) and the SPH pressure p_i (Eq. (5)).
(b) For each agent i, compute the SPH forces (Eqs. (6) and (7)) and update the acceleration as $\mathbf{a}_i := \mathbf{a}_i + \frac{-\nabla p_i + \mu\nabla^2\mathbf{v}_i}{\rho_i}$.

This yields a crowd-simulation loop that combines navigation behavior, traditional contact forces, and SPH forces, all of which can be scaled and/or disabled by modifying the appropriate parameters.

4.2.2 Dynamic Personal Rest Density. To make the SPH model more suitable for crowds, we make one notable change to it: we give each agent i a *personal rest density* ρ_i^0 that is allowed to *change over time*. The motivation for this is that we are not simulating a fluid with fixed physical properties, but a crowd of conscious individuals that can accept changes in the density around them.

The rest density ρ_i^0 can be interpreted as the crowd density that agent i is currently willing to accept. (As usual, though, the actual density can become higher due to other forces in the system.) It can be defined in various ways, such as the average density that agent i has recently perceived, or the density that it expects for the near future (e.g. based on knowledge of the environment). Our implementation (see Section 5.1) will approximate the recent average density in a memory-friendly way. We will clamp this value to a *maximum rest density* $\rho^{0,\max}$, which will be an intuitive parameter for controlling the density of the crowd.

5 IMPLEMENTATION AND SETTINGS

We have integrated SPH into UMANS [van Toll et al. 2020], an open-source framework for microscopic crowd simulation. We have extended its simulation loop to compute the SPH density, pressure, and forces as outlined in Section 4.2. The loop uses forward Euler integration. We will now discuss all relevant settings and details.

5.1 SPH Details

5.1.1 Kernel Functions. For the SPH smoothing kernel W, we use the different functions suggested by Müller et al. [2003], adapted to a 2D domain [Sjöström 2011]. More precisely, we use 2D versions of the 'Poly6' kernel for density, the 'spiky' kernel for pressure, and Müller's kernel for viscosity:

$$W^{\rho}(\mathbf{r},h) = \frac{4}{\pi h^8}\begin{cases}(h^2-\|\mathbf{r}\|^2)^3, & \text{if } \|\mathbf{r}\|<h\\ 0, & \text{otherwise}\end{cases} \tag{8}$$

$$\nabla W^{p}(\mathbf{r},h) = -\frac{30}{\pi h^5}\cdot\frac{\mathbf{r}}{\|\mathbf{r}\|}\begin{cases}(h-\|\mathbf{r}\|)^2, & \text{if } \|\mathbf{r}\|<h\\ 0, & \text{otherwise}\end{cases} \tag{9}$$

$$\nabla^2 W^{\text{visc}}(\mathbf{r},h) = \frac{360}{29\pi h^5}\begin{cases}h-\|\mathbf{r}\|, & \text{if } \|\mathbf{r}\|<h\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

5.1.2 Rest Density. For the dynamic rest density ρ_i^0 per agent, we let each agent i maintain an approximation $\hat{\rho}_i$ of its average SPH density over the last T^ρ seconds, where T^ρ is a parameter. This $\hat{\rho}_i$ is updated per frame as an exponential moving average:

$$\hat{\rho}_i \coloneqq \left(1 - \frac{\Delta t}{T^\rho}\right) \cdot \hat{\rho}_i + \frac{\Delta t}{T^\rho} \cdot \rho_i \tag{11}$$

(Compared to the *true* average over T^ρ seconds, this value can be computed in a more memory-friendly way, so it is more suitable for large crowds.) We then define the rest density ρ_i^0 as $\hat{\rho}_i$ clamped to the range $[\rho^{0,\min}, \rho^{0,\max}]$. In our experiments, we will treat $\rho^{0,\max}$ as a modifiable parameter while keeping $\rho^{0,\min}$ and T^ρ constant.

5.1.3 Other Considerations. A commonly used technique in SPH is to *ignore negative pressure* [Koschier et al. 2019]. This prevents unnecessary clustering of particles in low-density areas. In our case, whenever an agent i perceives a density ρ_i below its current rest density ρ_i^0, we use $-\nabla p_i = \mathbf{0}$ as the pressure force.

One difficulty of SPH lies in how to treat (static) polygonal *obstacles*, so that particles near obstacles perceive a meaningful density. An easy approach, which our implementation will use as well, is to place 'boundary particles' at regularly sampled positions inside all obstacles. In each iteration of the simulation loop, these boundary particles compute the SPH density and pressure, but they do not compute any forces and they do not move. Note that collision detection (for the traditional contact forces $\mathbf{F}_i^c$) is still based on the original polygonal obstacles; it is unrelated to these particles.

Finally, to produce stable results, SPH usually requires a higher framerate than a regular agent-based crowd simulation. Some tasks of our simulation loop, such as finding neighbors (Step 1) and running navigation policies (Step 2), do not need the high framerate required by SPH. We therefore use a large timestep Δt_{coarse} for these two tasks and a smaller timestep Δt_{fine} for the others. Our simulation loop uses Δt_{fine} overall, but it lets Steps 1 and 2 re-use their result from the previous frame whenever less than Δt_{coarse} seconds have passed. Section 6.3 will show that this combination of frequencies enables real-time simulations of large dense crowds.

5.2 Navigation Policy and Contact Forces

In our experiments, the navigation policy $\mathcal{P}_i$ of an agent i consists of *goal reaching* and optionally *collision avoidance*. The goal-reaching force is defined as follows:

$$\mathbf{F}_i^{\text{goal}} = K^{\text{goal}} \cdot (\mathbf{v}_i^{\text{pref}} - \mathbf{v}_i)/\tau \tag{12}$$

where $\mathbf{v}_i^{\text{pref}}$ is the *preferred velocity* that sends the agent towards its goal at its preferred speed, K^{goal} is a scalar for the force's strength (usually 1), and τ is a *relaxation time* (usually 0.5 seconds). For collision avoidance, we use the UMANS implementation of social forces [Helbing and Molnár 1995]; we will omit the details here.

For *contact forces* between colliding agents, we implement the model by Helbing et al. [2000] that is commonly used throughout literature. The total contact force applied to agent i is the sum of all contact forces imposed by other agents and by obstacles:

$$\mathbf{F}_i^c = \sum_{j \neq i} \mathbf{F}_{ij}^c + \sum_O \mathbf{F}_{iO}^c \tag{13}$$

Each agent i is approximated by a disk with radius D_i. The contact force imposed by a neighboring agent j is defined as:

$$\mathbf{F}_{ij}^c = K^{\text{ag}} \cdot \max(0, D_i + D_j - \|\mathbf{r}_{ij}\|) \cdot \mathbf{r}_{ij} / \|\mathbf{r}_{ij}\| \tag{14}$$

where $K^{\text{ag}} \geq 0$ is a parameter and $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$. The contact force imposed by a neighboring obstacle O is defined as:

$$\mathbf{F}_{iO}^c = K^{\text{obs}} \cdot \max(0, D_i - \|\mathbf{r}_{iO}\|) \cdot \mathbf{r}_{iO} / \|\mathbf{r}_{iO}\| \tag{15}$$

where $K^{\text{obs}} \geq 0$ is a parameter and $\mathbf{r}_{iO}$ is the vector to $\mathbf{r}_i$ from the nearest boundary point of O. Note that a contact force is always $\mathbf{0}$ if there is no collision with the corresponding agent or obstacle.

5.3 Parameter Settings

The *variables* of our experiments will be the force scalars (k, μ, K^{ag}, K^{obs}) and the maximum rest density $\rho^{0,\max}$. All other parameters will be treated as *constants*. These are the following:

- The parameters for computing the dynamic rest density ρ_i^0 per agent (see Section 5.1.2). We will use $\rho^{0,\min} = 0$ P/m^2 and $T^\rho = 0.1$ s, leaving $\rho^{0,\max}$ as a variable.
- The SPH kernel support radius h. A particle takes neighbors within this radius into account. We will use $h = 1$ m, which yields a good balance between smoothness and performance.
- The timesteps Δt_{fine} and Δt_{coarse}. We will use $\Delta t_{\text{coarse}} = 0.1$ s (a common value for crowd simulation) and $\Delta t_{\text{fine}} = 0.02$ s (the largest value that consistently gave stable results).
- The goal-reaching parameters: strength K^{goal} and relaxation time τ. We will use $\tau = 0.5$ and $K^{\text{goal}} = 1$, except in Section 6.2 where agents switch between weak and strong goal reaching ($K^{\text{goal}} = 0.1$ and $K^{\text{goal}} = 1$, respectively).
- The parameters of the social-force model. For ease of comparison, we use the default settings suggested by its authors.
- The disk radius D_i and mass m_i per agent. A logical setting for the mass is 1, so that the unit of density can be interpreted as 'persons per square meter'. For SPH boundary particles, we will use $m_i = 1$ and $D_i = 0.24$ m. For agents, to avoid symmetry, we will use a uniformly random radius $D_i \in [0.215, 0.265]$ and a corresponding mass $m_i = (D_i/0.24)^2$.
- The remaining agent parameters in UMANS: preferred speed, maximum speed, and maximum acceleration. We will use $s_{\text{pref}} = 1.4$ m/s, $s_{\max} = 1.8$ m/s, and $a_{\max} = 5.0$ m/s^2.

6 EXPERIMENTS AND RESULTS

This section demonstrates our model in two extreme-density scenarios. We test various combinations of SPH, collision avoidance, and contact forces. We perform all experiments on a Windows 10 device with an Intel Core i7-7920HQ CPU, using 6 parallel threads.

For comparative purposes, we always compute the SPH density for all agents, even when these values are not used by the simulation. In our figures, we will color the agents based on either their velocity or their SPH density. These coloring schemes are shown in Fig. 2. We also encourage the reader to watch the supplementary video of this paper, which shows our results in motion.

6.1 Scenario 1: Room Evacuation

6.1.1 Description. In our first scenario, summarized in Fig. 3(a), 400 agents need to exit a room of 20×20 m through a 0.8 m-wide

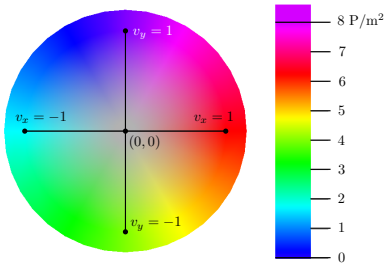


Figure 2: The two agent-coloring schemes used in this paper. Left: Coloring based on an agent's current velocity. Right: Coloring based on an agent's current SPH density.

doorway. We set the goal of each agent to a point 1m beyond the exit, and we remove an agent when it is within 0.5m of this goal. This scenario is motivated by real-world experiments that have been performed in recent years; see e.g. Garcimartín et al. [2016].

We intend to show that SPH facilitates generating crowds of a particular density, that it improves over standard contact forces, and that it is even successful without collision avoidance. We therefore run this scenario both with and without SPH, and both with and without social forces (SF). All runs use contact forces (CF) to some extent. In the runs *without* SPH (i.e. in CF and CF+SF), we vary the contact-force scalar K^{ag}, and we set $K^{\text{obs}} = 500$. In the runs *with* SPH (i.e. in CF+SPH and CF+SF+SPH), we vary the maximum rest density $\rho^{0,\max}$, and we use weak contact forces ($K^{\text{ag}} = 50$, $K^{\text{obs}} = 200$) and fixed SPH constants ($k = 200$, $\mu = 0$).

6.1.2 Discussion of Results. Fig. 3 shows each crowd after 15 s, and Table 1 reports several statistics per run. Its rightmost metric ('average flow rate') is computed as the average number of agents that reach the goal per second. This is comparable to the flow rate J that is often measured in real-world experiments. For evacuations under non-competitive conditions, researchers have measured flow rates of up to 4 people per second [Garcimartín et al. 2016].

In the CF simulation variants (Figs. 3(b)–3(e)), the agents simply move forward until they collide, and then K^{ag} controls how strongly they push each other away. However, K^{ag} impacts several statistics at the same time. With *weak* contact forces, the agents successfully leave the room, but the crowd achieves very high densities, and the flow rate is much higher than has been observed in real-world experiments. With *strong* contact forces, the density and flow rate remain manageable, but the exit eventually gets blocked when several agents approach it at the same time. (This can be seen in Table 1: not all agents reach the goal.) There is no value for K^{ag} that limits the density and flow rate without blocking the exit.

In the CF+SF variants (Figs. 3(f)–3(g)), social forces let agents approach a crowded area more carefully. Also, the flow rates are more realistic because agents are less selfish in using the exit. However, the relation between K^{ag} and the crowd density is still not intuitive, and strong contact forces ($K^{\text{ag}} = 500$) still cause a deadlock.

When SPH is added (i.e. in CF+SPH and CF+SF+SPH), $\rho^{0,\max}$ controls the maximum density that the crowd wishes to attain. Table 1 shows that the average density is usually lower than $\rho^{0,\max}$. Depending on $\rho^{0,\max}$, agents automatically keep distance from one another, or they allow their disks to overlap. This is clearly visible in Figs. 3(j)–3(u). Furthermore, deadlocks never occur because the contact forces are always sufficiently weak. These observations hold for both CF+SPH and CF+SF+SPH, i.e. both without and with social forces. Just like before, social forces change the way in which agents approach a crowded area, and they prevent unrealistic flow rates in the narrow passage. However, even the CF+SPH variant has configurations where the flow rate is manageable.

These results show in particular that SPH can help avoid unrealistically high densities. The rest density $\rho^{0,\max}$ is an in intuitive parameter for controlling the density of the crowd, whereas the contact-force scale K^{ag} has a less obvious meaning. Our supplementary video shows another effect that strengthens this argument. With SPH, the density remains roughly *constant during the evacuation* as the crowd becomes smaller. Without SPH, the density near the exit depends greatly on the size of the crowd.

Thus, given an estimate of the density in a real crowd, SPH makes it easier to simulate a crowd of a comparable density. This is an important step towards the synchronization between simulation and reality. We do acknowledge that this is merely one scenario, and that it is difficult to say in general whether an SPH-enriched model is less or more *realistic*. The real-world data that has been gathered for evacuations varies strongly in terms of density and flow rate [Garcimartín et al. 2016].

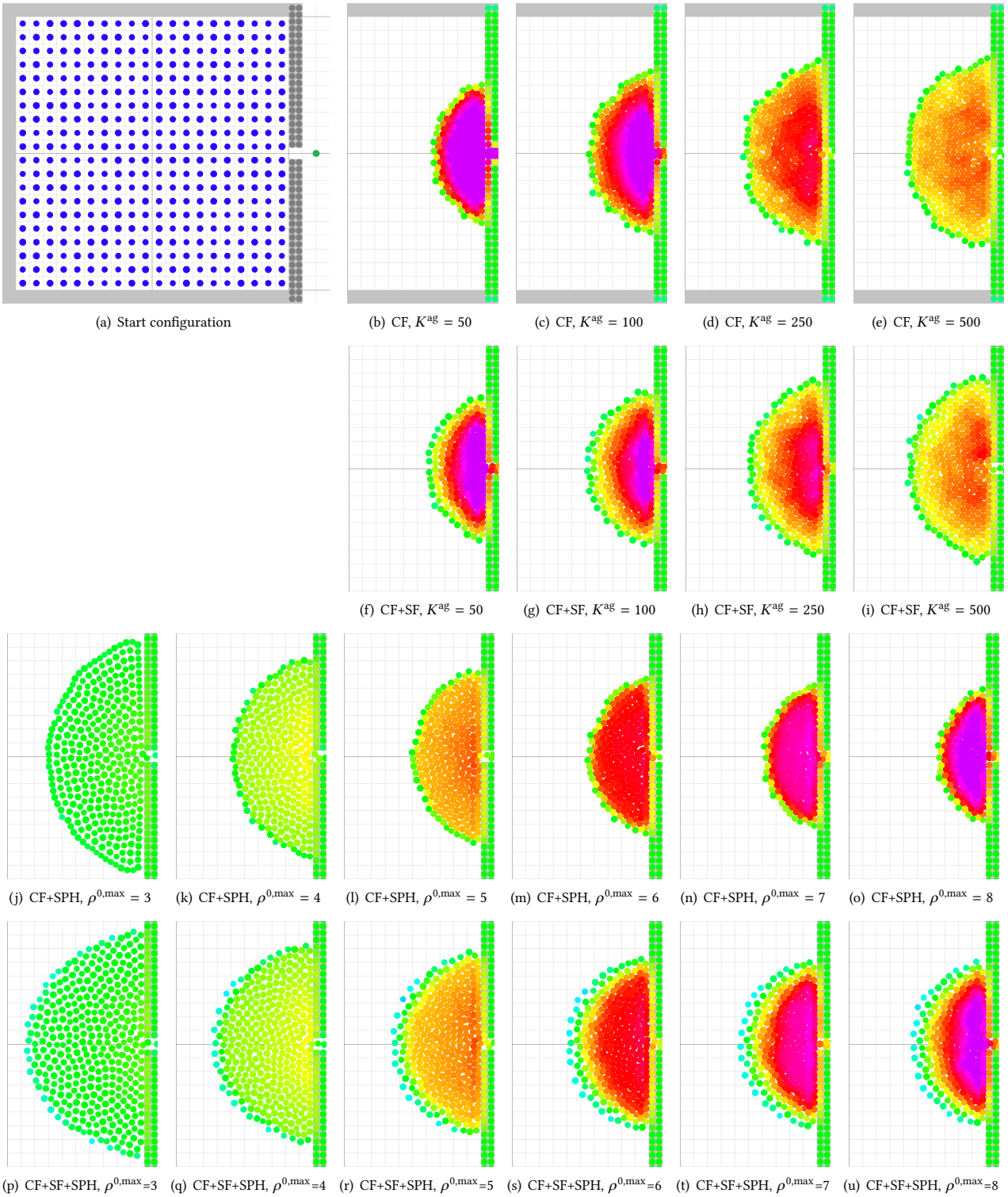
6.2 Scenario 2: Concert with Shockwave

6.2.1 Description. Next, we simulate a large dense crowd attending a concert. The purpose of this experiment is to show that SPH and contact forces combined yield better behavior than either component on its own. Also, SPH can help simulate *crowd shockwaves*: a pushing motion that propagates through the crowd, where people are involuntarily transported. This phenomenon has been observed and recorded in real life, such as at a concert in 2005.[1]

[1]See *https://youtu.be/BgpdmAtbhbE* for footage of the crowd at this concert. See Bottinelli and Silverberg [2018] for an analysis of this video.

Table 1: Results of the 400-agent evacuation. Columns 2 to 4 show the number of agents that reach the goal, the avg. SPH density among all agents at $t = 15$s (with standard deviations), and the avg. number of evacuated agents per second.

Simulation variant	Specific settings	#Evacuated agents	Avg. density (P/m²) after 15s	Avg. flow rate (P/s)
CF	$K^{\text{ag}} = 50$	400	7.13 [2.46]	6.63
	$K^{\text{ag}} = 100$	400	5.63 [1.55]	4.88
	$K^{\text{ag}} = 250$	371	4.67 [0.87]	3.70
	$K^{\text{ag}} = 500$	216	4.37 [0.67]	3.52
CF+SF	$K^{\text{ag}} = 50$	398	6.47 [2.22]	2.48
	$K^{\text{ag}} = 100$	398	5.82 [1.67]	2.48
	$K^{\text{ag}} = 250$	398	5.43 [1.11]	2.25
	$K^{\text{ag}} = 500$	254	4.94 [0.78]	4.18
CF+SPH	$\rho^{0,\max} = 3$	400	2.95 [0.34]	2.50
	$\rho^{0,\max} = 4$	400	3.64 [0.56]	3.26
	$\rho^{0,\max} = 5$	400	4.20 [0.81]	4.03
	$\rho^{0,\max} = 6$	400	4.67 [1.08]	4.71
	$\rho^{0,\max} = 7$	400	5.04 [1.37]	5.29
	$\rho^{0,\max} = 8$	400	5.32 [1.64]	5.84
CF+SF+SPH	$\rho^{0,\max} = 3$	398	3.18 [0.33]	2.12
	$\rho^{0,\max} = 4$	398	4.07 [0.57]	2.51
	$\rho^{0,\max} = 5$	398	4.83 [0.89]	2.80
	$\rho^{0,\max} = 6$	398	5.42 [1.25]	2.84
	$\rho^{0,\max} = 7$	398	5.80 [1.57]	2.89
	$\rho^{0,\max} = 8$	398	6.02 [1.83]	3.28



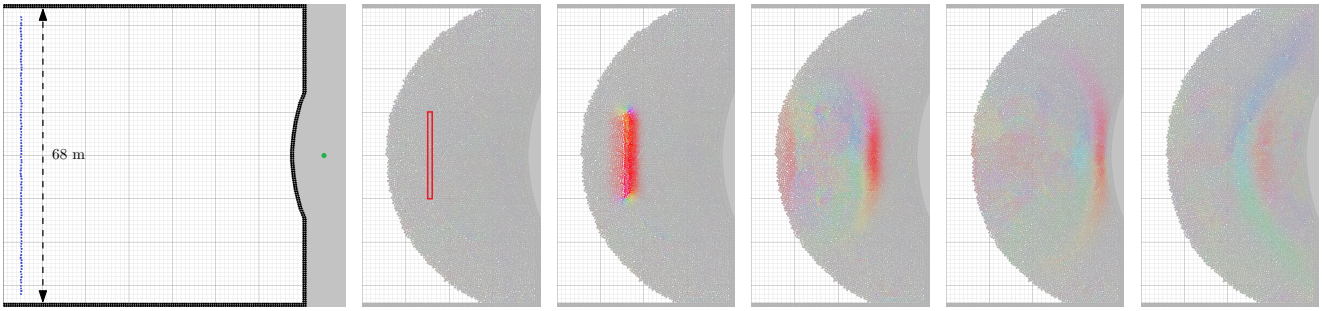
(a) Start configuration (b) CF, $K^{ag} = 50$ (c) CF, $K^{ag} = 100$ (d) CF, $K^{ag} = 250$ (e) CF, $K^{ag} = 500$

(f) CF+SF, $K^{ag} = 50$ (g) CF+SF, $K^{ag} = 100$ (h) CF+SF, $K^{ag} = 250$ (i) CF+SF, $K^{ag} = 500$

(j) CF+SPH, $\rho^{0,max} = 3$ (k) CF+SPH, $\rho^{0,max} = 4$ (l) CF+SPH, $\rho^{0,max} = 5$ (m) CF+SPH, $\rho^{0,max} = 6$ (n) CF+SPH, $\rho^{0,max} = 7$ (o) CF+SPH, $\rho^{0,max} = 8$

(p) CF+SF+SPH, $\rho^{0,max}=3$ (q) CF+SF+SPH, $\rho^{0,max}=4$ (r) CF+SF+SPH, $\rho^{0,max}=5$ (s) CF+SF+SPH, $\rho^{0,max}=6$ (t) CF+SF+SPH, $\rho^{0,max}=7$ (u) CF+SF+SPH, $\rho^{0,max}=8$

Figure 3: Evacuation experiment. (a) 400 agents (in blue) need to exit a room. Their goal position is shown in green. SPH obstacle particles are shown in dark gray. The remaining subfigures show the simulation after 15 s, using density coloring. Each row shows its own simulation variant, with several values for its main parameter. (b)-(e) Contact forces only. (f)-(k) Contact forces and social forces. (j)-(o) Contact forces and SPH. (p)-(u) Contact forces, social forces, and SPH.



(a) Start configuration (b) t = 150s (c) t = 150.6s (d) t = 152s (e) t = 153s (f) t = 155s

Figure 4: Concert scenario with 10,000 agents. (a) For 100 seconds, we add 100 SPH-enriched agents (in blue) per second, with a goal position on the right (in green). Boundary particles are shown in black. (b) After 150 seconds, all agents in the red box push forward during 0.5 seconds. (c)-(f) With velocity coloring, a wave propagating through the crowd is clearly visible.

Fig. 4(a) shows our set-up. We add 10,000 agents (in rows of 100 agents per second) with a goal position on the stage (on the right side). The agents will try to get as close to the stage as possible, but the stage itself is an obstacle. We set the goal-reaching scalar K^{goal} to 0.1, so that agents do not move to the stage too aggressively. The SPH and contact-force parameters will vary per run. For simplicity, we do not use social forces in this experiment.

Eventually, the crowd converges to a shape concentrated around the stage; see Fig. 4(b) for an example. After 150 seconds, we let all agents in a given rectangle of 1×20 m push forward without paying attention to the rest of the crowd. We do this by setting their K^{goal} to 1 and by ignoring any contact and SPH forces. The agents keep these settings for 0.5 s and then revert to their old settings.

6.2.2 Discussion of Results. Fig. 4 shows the result when using SPH ($\rho^{0,\max} = 5$, $k = 100$, $\mu = 3$) and weak contact forces ($K^{\text{ag}} = 50$, $K^{\text{obs}} = 200$). With these settings, a wave moves towards the stage, bounces off, and then moves backward. This is consistent with real-world recordings, albeit with a different scale and duration [Bottinelli and Silverberg 2018]. Our supplementary video shows that the wave occurs for other values of $\rho^{0,\max}$ as well.

If we replace SPH by stronger contact forces, we can produce a comparable wave, but with three clear disadvantages. First, we have (again) poor control over the crowd density. Second, the contact-force scale K^{ag} affects both the density and the wave's speed at the same time. Third, when agents have just experienced the wave, they shake heavily as they move back to a stable position. The crowd is more stable with SPH enabled, mostly thanks to the *viscosity* force which we will discuss shortly. Fig. 5 shows this difference. The video of this paper contains more results.

In short, SPH enables the simulation of a shockwave at various crowd densities. Without SPH, the results are less stable, and the wave's behavior is an unintuitive result of the contact forces.

6.2.3 Effect of Parameters. The parameters of SPH control the density of the crowd and the properties of the shockwave. We will now discuss the effect of these parameters, as well as the importance of (weak) contact forces. Once again, we refer to our supplementary video for more details; the results are better observed in motion.

With different values of $\rho^{0,\max}$, the crowd attains a different density, but the wave propagation still occurs, and it maintains the same speed. With a higher gas constant (e.g. $k = 1{,}000$), the wave propagates faster and the wavefront is thicker. With a lower gas constant, the wave is slower and thinner. However, if k is very low, the density near the stage far exceeds $\rho^{0,\max}$ because SPH can no longer 'defeat' the goal-reaching and contact forces.

As mentioned earlier, a small viscosity force ($\mu = 3$) improves the crowd's stability. With $\mu = 0$, the wave still occurs, but many agents show 'jittery' motion, especially after having been exposed to the wave; see Fig. 5 again. This follows the consensus that the viscosity force makes SPH more stable [Koschier et al. 2019].

So far, we have deliberately still used contact forces next to SPH forces. Disabling inter-agent contact forces ($K^{\text{ag}} = 0$) would allow particles to overlap completely, which may be reasonable for *fluid* simulations, but not for *crowd* simulations where particles represent human bodies. Also, in this scenario, setting K^{ag} to 0 leads to errors in which the boundary of the crowd deforms, as shown in Fig. 6. We could only reduce this effect by using a much smaller timestep (e.g. $\Delta t_{\text{fine}} = 0.002$ s), which is not practical for real-time crowd simulation. This observation suggests that contact forces actually make SPH more stable as well, and not just the other way around.

Our *dynamic rest density* automatically adapts to the situation at hand. As such, one simulation can contain areas of different densities, and the rest density in an area can change over time. One advantage of this is that agents near the boundary of the crowd stand still, whereas a static rest density leads to agents repeatedly 'splashing' in and out of the crowd, as shown in Fig. 7. The latter artifact occurs because a static rest density ρ^0 is only truly reached

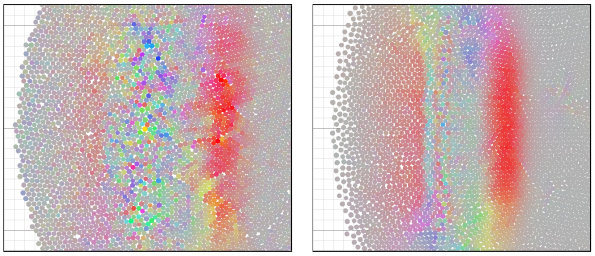


Figure 5: Left: With contact forces only, the crowd becomes unstable after exposure to the wave. Right: With SPH (including viscosity), this effect is strongly reduced.

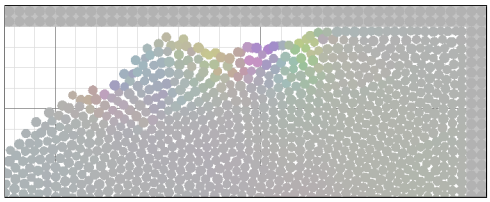


Figure 6: Without inter-agent contact forces ($K^{\mathbf{ag}} = 0$), we observe deformations at the boundary of the crowd.

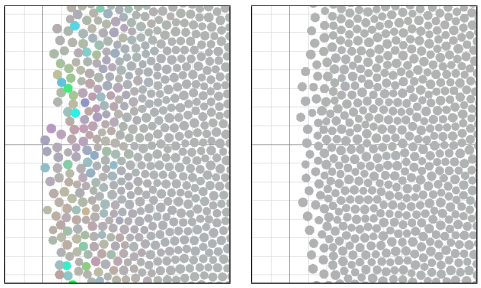


Figure 7: Left: A static rest density (here: $\rho^0 = 4$) yields chaotic motion at the crowd's boundary. Right: A dynamic rest density ($\rho^{0,\mathbf{min}} = 0, \rho^{0,\mathbf{max}} = 4$) gives stable results.

inside the dense crowd; around the border, the density is lower, leading to larger differences in pressure and stronger forces. Thus, a dynamic per-agent rest density improves the simulation's stability.

6.3 Performance Analysis

We conclude this section by looking into the running time of our system. In the concert scenario (with 10,000 agents and 840 boundary particles), we measure the computation times between $t = 150$ and $t = 180$, i.e. from the moment the wave starts.

6.3.1 Overview. The average running time *per frame* (the 'frame time') of the concert scenario is 5.51 ms. This is less than the simulation timestep Δt_{fine} (20 ms), so the simulation runs in real-time.

For real-time applications, Δt_{fine} should ideally be as large as possible without making the crowd unstable. Preliminary experiments have led to our use of $\Delta t_{\text{fine}} = 0.02$ s (i.e. 50 frames per second). This framerate is also used by many physics engines. Coarser framerates gave unstable results, both with and without SPH. This makes sense because we simulate tightly-packed particles.

6.3.2 Details. To gain insight into which aspects of the simulation are the most demanding, we also measure the average frame time *per task* of the simulation. Furthermore, to see how these times scale with the number of agents, we measure the frame times for various crowd sizes, by running variants of the concert scenario where fewer or more agents are inserted per second.

Per simulation frame, the first task (1) is to compute a *kd*-tree of all agent positions. The remaining tasks compute (on six parallel threads) the following for all agents: neighbors (2), preferred velocity to the goal (3), SPH density and pressure (4), non-contact forces (5), contact forces (6), and the new velocity and position (7). Tasks 1 and 2 use Δt_{coarse} (i.e. they occur once per five frames); the rest uses Δt_{fine}. Task 5 includes both SPH and goal-reaching forces.

Fig. 8 shows the results. Constructing the *kd*-tree (task 1) is fast, but querying it to find neighbors (task 2) is rather expensive. These

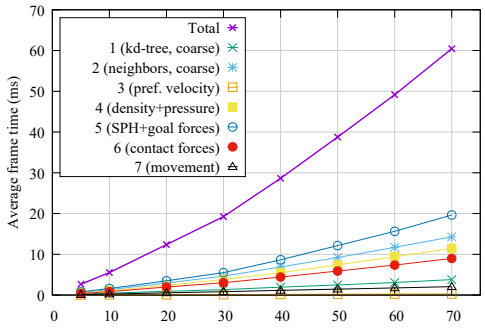


Figure 8: The relation between the avg. frame time and the number of (non-boundary) agents in the concert scenario.

tasks are unavoidable in any agent-based simulation. Using Δt_{coarse} instead of Δt_{fine} keeps their impact manageable.

Tasks 3 and 7 take constant time per agent. Tasks 4–6 compute certain quantities per agent by looping over its neighbors. Thus, the performance of these tasks depends on the number of neighbors per agent, and therefore on the crowd density. If we were to add collision avoidance, it would belong to the same category of tasks.

With 30,000 agents, the average frame time (19.25 ms) is just under Δt_{fine} (20 ms). Recall from Section 6.2.2 that the maximum rest density is 5 P/m^2 in this scenario. Our results suggest that the system can simulate an extremely dense crowd with tens of thousands of agents in real-time. Again, the frame time will increase if agents perform more tasks, such as collision avoidance.

7 DISCUSSION

We have not explicitly compared our simulation to a macroscopic model. Macroscopic crowd simulations can be faster, as they simulate a grid rather than a full crowd. However, Yuan et al. [2020] have already shown that SPH has behavioral advantages. Because they did not yet consider contact forces or a dynamic rest density, we are confident that these advantages persist.

In our extreme-density scenarios, it was sufficient to use only SPH, contact forces, and the standard social-force model. However, more intelligent *collision avoidance* may be necessary in lower-density areas, and *global path planning* is important in scenarios with more obstacles. Combining all this with SPH is conceptually easy (see Section 4), but maintaining real-time performance may be challenging, especially if the collision-avoidance algorithm is computationally heavy. An interesting idea would be to interpolate from collision avoidance to SPH as the density increases.

Our SPH implementation can be improved in various ways. For example, replacing boundary particles by more sophisticated obstacle handling [Bender et al. 2019] will make scenarios easier to set up, and it will most likely make the simulation even faster.

In the context of shockwave propagation, we have discussed the influence of contact forces and parameter settings. However, this discussion was non-exhaustive, and it is not yet clear how it extends to other scenarios. We intend to perform a more general parameter study of SPH-based crowd simulation in future work.

Finally, we have not made any claims about the *realism* of our simulations. Comparing a crowd simulation to reality is a fairly young topic of research. To our knowledge, any state-of-the-art

concepts in this area are only suitable for lower-density crowds, often requiring 'perfect' motion data of individual people. This data cannot be reliably extracted from high-density crowd footage. We therefore require new techniques for analyzing such footage, and new metrics for quantifying the behavior of dense crowds.

8 CONCLUSIONS AND FUTURE WORK

This paper has shown how to enrich agent-based crowd simulation with the concept of Smoothed Particle Hydrodynamics (SPH), a particle-based method for simulating fluid dynamics. SPH makes a crowd simulation more suitable for extreme-density scenarios where standard collision handling is not sufficient. Previously, such scenarios were modelled with macroscopic methods that simulate the crowd as a whole. By contrast, SPH fits in the microscopic paradigm where each person has individual property and goals.

SPH can be added to any agent-based simulation by treating each agent as an SPH particle. We have extended the standard SPH model with a dynamic personal rest density per particle. This makes the model more suitable for crowds of conscious individuals.

We have demonstrated our model in two scenarios, one of which features a propagating shockwave. Compared to using only contact forces, SPH gives intuitive control over the crowd density, and it can improve the simulation's stability. Conversely, (weak) contact forces also improve the stability of SPH, so they remain relevant for crowd simulation. Our implementation can simulate extreme-density crowds with tens of thousands of agents in real-time.

To our knowledge, we are the first to integrate SPH into agent-based crowd simulation in a general way. This makes the agent-based paradigm suitable for all crowd densities, which removes the need to choose between microscopic and macroscopic simulation.

Our current implementation shows that SPH indeed has merit for the real-time simulation of extreme-density crowds. However, we have not yet considered any scenarios where *collision avoidance* or *global path planning* is crucial. Combining all these navigation tasks in real-time requires further research and development.

We believe that SPH is also useful for *measuring the density* in a (real or virtual) crowd. Compared to grid-based density computation, SPH is smoother and resolution-independent. Compared to methods based on the Voronoi diagram of all agents (or people), SPH is easier to implement. Thus, even without using any SPH *forces*, the SPH *density* can already be useful for analytical purposes. We intend to apply this idea to existing and future datasets.

Finally, as discussed in Section 7, *comparing high-density crowd simulations to reality* is a future-work topic with many unanswered questions. It requires more insight into the effects of simulation parameters, new ways to quantify the behavior of dense crowds, and better techniques for extracting meaningful data from dense-crowd footage. We hope that these developments eventually lead to a system that automatically calibrates a simulation to reality.

REFERENCES

Jan Bender, Tassilo Kugelstadt, Marcel Weiler, and Dan Koschier. 2019. Volume maps: An implicit boundary representation for SPH. In *Motion, Interaction and Games*. Article 26, 10 pages.

Arianna Bottinelli and Jesse L. Silverberg. 2018. Can high-density human collective motion be forecasted by spatiotemporal fluctuations? *arXiv:1809.07875* (2018).

Sean Curtis, Andrew Best, and Dinesh Manocha. 2016. Menge: A modular framework for simulating crowd movement. *Collective Dynamics* 1, A1 (2016), 1–40.

Teofilo B. Dutra, Ricardo Marques, Joaquim B. Cavalcante-Neto, Creto A. Vidal, and Julien Pettré. 2017. Gradient-based steering for vision-based crowd simulation algorithms. *Comput. Graph. Forum* 36, 2 (2017), 337–348.

Ángel Garcimartín, Daniel R. Parisi, Jose M. Pastor, César Martín-Gómez, and Iker Zuriguel. 2016. Flow of pedestrians through narrow doors with different competitiveness. *J. Stat. Mech. Theory Exp.* 4 (2016), 043402.

Robert A. Gingold and Joseph J. Monaghan. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. Roy. Astron. Soc.* 181, 11 (1977), 375–389.

Abhinav Golas, Rahul Narain, and Ming C. Lin. 2014. Continuum modeling of crowd turbulence. *Physical Review E* 90, 4 (2014), 042816.

Stephen J. Guy, Jatin Chhugani, Sean Curtis, Pradeep Dubey, Ming Lin, and Dinesh Manocha. 2010. PLEdestrians: A least-effort approach to crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*. 119–128.

Dirk Helbing, Illés Farkas, and Tamás Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407 (2000), 487–490.

Dirk Helbing and Péter Molnár. 1995. Social force model for pedestrian dynamics. *Physical Review E* 51, 5 (1995), 4282–4286.

Omar Hesham and Gabriel Wainer. 2017. Context-sensitive personal space for dense crowd simulation. In *Proc. Symp. Simulation for Architecture and Urban Design*. Article 19, 8 pages.

Roger L. Hughes. 2003. The flow of human crowds. *Annu. Rev. Fluid Mech.* 35, 35 (2003), 169–182.

Ioannis Karamouzas, Brian Skinner, and Stephen J. Guy. 2014. Universal power law governing pedestrian interactions. *Phys. Rev. Lett.* 113 (2014), 238701:1–5. Issue 23.

Peter M. Kielar, Daniel H. Biedermann, and André Borrmann. 2016. *MomenTUMv2: A modular, extensible, and generic agent-based pedestrian behavior simulation framework*. Technical Report TUM-I1643. TU München, Institut für Informatik.

Sujeong Kim, Stephen J. Guy, Karl Hillesland, Basim Zafar, Adnan Abdul-Aziz Gutub, and Dinesh Manocha. 2015. Velocity-based modeling of physical interactions in dense crowds. *The Visual Computer* 31 (2015), 541–555. Issue 5.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2019. Smoothed Particle Hydrodynamics techniques for the physics based simulation of fluids and solids. In *Eurographics 2019 Tutorials*.

Leon B. Lucy. 1977. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal* 82, 12 (1977), 1013–1024.

Bertrand Maury, Aude Roudneff-Chupin, and Filippo Santambrogio. 2010. A macroscopic crowd motion model of gradient flow type. *Math. Mod. Meth. Appl. S.* 20, 10 (2010), 1787–1821.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*. 154–159.

Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. 2009. Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.* 28 (2009), 1–8. Issue 5.

Nuria Pelechano, Jan M. Allbeck, and Norman I. Badler. 2007. Controlling individual agents in high-density crowd simulation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*. 99–108.

Nuria Pelechano, Jan M. Allbeck, Mubbasir Kapadia, and Norman I. Badler. 2016. *Simulating Heterogeneous Crowds with Interactive Behaviors*. CRC Press.

Kalle Sjöström. 2011. *Computational fluid dynamics in 2D game environments*. Master's thesis. Umeå University.

Sybren A. Stüvel, Nadia Magnenat-Thalmann, Daniel Thalmann, and A. Frank van der Stappen. 2016. Torso crowds. *IEEE Trans. Vis. Comput. Graphics* 23, 7 (2016), 1823–1837.

Weerawat Tantisiriwat, Arisara Sumleeon, and Pizzanu Kanongchaiyos. 2007. A crowd simulation using individual-knowledge-merge based path construction and Smoothed Particle Hydrodynamics. In *Proc. 15th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*. 261–268.

Daniel Thalmann and Soraia R. Musse. 2013. *Crowd Simulation* (2 ed.). Springer.

Wouter van Toll, Fabien Grzeskowiak, Axel López, Javad Amirian, Florian Berton, Julien Bruneau, Beatriz Daniel, Alberto Jovane, and Julien Pettré. 2020. Generalized microscopic crowd simulation using costs in velocity space. In *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games*. 10 pages.

Wouter van Toll, Norman Jaklin, and Roland Geraerts. 2015. Towards believable crowds: A generic multi-level framework for agent navigation. In *ASCI.OPEN*.

Adrien Treuille, Seth Cooper, and Zoran Popović. 2006. Continuum crowds. *ACM Trans. Graph.* 25 (2006), 1160–1168. Issue 3.

Jur P. van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. In *Proc. Int. Symp. Robotics Research*. 3–19.

Christin Vetter, Lars Oetting, Christian Ulrich, and Thomas Rung. 2011. SPH simulations of pedestrian crowds. In *Proc. 6th Int. Spheric Workshop*. 261–268.

Tomer Weiss, Chenfanfu Jiang, Alan Litteneker, and Demetri Terzopoulos. 2017. Position-based multi-agent dynamics for real-time crowd simulation. In *Proc. 10th Int. Conf. Motion in Games*. 10:1–10:8.

Yufei Yuan, Bernat Goñi-Ros, Ha H. Bui, Winnie Daamen, Hai L. Vu, and Serge P. Hoogendoorn. 2020. Macroscopic pedestrian flow simulation using Smoothed Particle Hydrodynamics. *Transp. Res. Part C Emerg. Technol.* 111 (2020), 334 – 351.