SPH crowds: Agent-based crowd simulation up to extreme densities using fluid dynamics

Wouter van Toll, Thomas Chatagnon, Cédric Braga, Barbara Solenthaler, Julien Pettré

Computers & Graphics (2021)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SPH crowds: Agent-based crowd simulation up to extreme densities using fluid dynamics

Wouter van Toll[a,*], Thomas Chatagnon[a], Cédric Braga[a], Barbara Solenthaler[b], Julien Pettré[a]

[a] *Univ Rennes, Inria, CNRS, IRISA, France*
[b] *ETH Zürich, Switzerland*

ARTICLE INFO

Article history:
Received 1 April 2021
Received in final form 6 June 2021
Accepted 12 June 2021
Available online 18 June 2021

Keywords: Crowd simulation, Fluid dynamics *2010 MSC:* 68T42, 76M28

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 Smoothed Particle Hydrodynamics (SPH), a particle-based method that is popular for fluid simulation. We integrate SPH into the crowd simulation loop by treating each agent as a fluid particle. The forces of SPH (for pressure and viscosity) then augment the usual navigation behavior and contact forces per agent. We extend the standard SPH model with a dynamic rest density per particle, which intuitively controls the crowd density that an agent is willing to accept. We also present a simple way to let agents blend between individual navigation and fluid-like interactions depending on the SPH density.

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. Also, density-based blending between collision avoidance and SPH improves the simulation of mixed-density scenarios. 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.

DOI: https://doi.org/10.1016/j.cag.2021.06.005.

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 or 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 [1], and new types of collective motion such shockwaves may occur [2].

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

*Corresponding author.
e-mail: wouter.van-toll@inria.fr (Wouter van Toll)

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)* [3] 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' physical interactions 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, and we analyze the effects of parameters.

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. To make SPH more suitable for crowd simulation, we add the concept of a dynamic rest density per particle, we present a specific way to handle 2D polygonal obstacles, and we allow agents to blend between behaviors depending on the SPH density.
- 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, a dynamic rest density, and density-dependent behavior 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, dynamic personal rest densities, and density-dependent blending), and the first that applies it to *extremely dense* crowds of $\geq$4 P/m^2. Furthermore, we thoroughly analyze the system's parameters and real-time performance.

1.3. Extension of previous work

This paper is an extension of a conference publication [4] that presented the SPH-enhanced crowd simulation and several results. Next to an improved organization, this extended version contains the following new content:

- In Section 4.3, we present an improved way to handle static obstacles (such as walls) in the SPH component of the simulation. Instead of using a particle approximation, our method uses the 2D polygon representation of obstacles that is common for crowd simulation. This has several advantages, including easier integration into existing crowd simulators, and fewer parameters to manage.
- In Section 4.4, we present a simple yet effective way to transition from individual behavior (e.g. collision avoidance) to fluid-like physical interactions (SPH) depending on the crowd density. This enables microscopic crowd simulation of mixed-density scenarios.
- In Sections 5 and 6, we now use collision-avoidance methods that are more representative of the state of the art, and we describe their mathematical details. These new methods impact our results and analysis in Sections 6.1 and 6.2.
- In Section 6.2, we show a new experiment that demonstrates the advantage of density-based behavior blending.
- In Section 6.3, we perform a more thorough analysis of parameter effects, including more visual results.

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 [5, 6].

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 [7, 8, 9].

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 [10, 11], velocity selection [12, 13, 14], or vision [15]. They usually assume that collisions can indeed be avoided. Some algorithms prevent collisions by letting agents stop when necessary [16]. 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. [17] 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. [18] have added physical interaction forces (such as pushing) to simulate a crowded pilgrimage scenario. Hesham and Wainer [19] have given agents a dynamic *personal space* that adapts to their current speed.

In this paper, we let agents experience forces computed by a fluid-dynamics simulation method (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.

Stüvel et al. [20] 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.

Other work has focused on giving agents density-dependent behavior. Best et al. [21] have used crowd density information to update the *preferred velocity* of agents, allowing them to actively avoid dense areas. Van Goethem et al. [22] have adapted an agent's preferred velocity to the motion of surrounding agents, to improve the crowd flow at high densities. This bears similarities to the concept of viscosity in fluid dynamics.

Our work contains a different density-dependent aspect: a mechanism to let agents *blend* between collision avoidance and fluid-like interactions depending on the (SPH) density. This is particularly useful for *mixed-density* scenarios, where low-to-medium densities require collision avoidance, while high-to-extreme densities are controlled by fluid dynamics. A similar blending scheme was used by Narain et al. [23]; compared to their work, we generalize the concept to density-based blending between arbitrary (combinations of) algorithms. An algorithm for density-dependent behavior [21, 22] could be added to the 'low-to-medium density' end of such a blended system.

2.2. Dense crowds and fluid dynamics

In contrast to microscopic algorithms, *macroscopic* algorithms simulate the crowd as a whole [24, 1], using a grid in which each cell prescribes the optimal walking speed and direction. Popularized by Treuille et al. [25], the concept has been successfully used to simulate extreme-density crowds [23] and turbulence effects [26]. These extensions are based on the laws of fluid mechanics, motivated by the popular belief that dense crowds bear similarities to fluids [1].

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 [27, 3], SPH has become a popular fluid-simulation method for computer graphics, starting with the work by Müller et al. [28]. A recent tutorial by Koschier et al. [29] 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 [30, 31], 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, and they do not describe how their SPH computations handle obstacles.

Weiss et al. [32] 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. [33] 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 existing 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. Also, to make SPH more suitable for crowds, we introduce a dynamic rest density per particle, and we present a simple particle-free way to handle obstacles. Finally, we study a *mixed-density* scenario where agents need to blend between individual and fluid-like behavior. All in all, we give a more complete explanation of how SPH can lift agent-based crowd simulation to extreme densities.

3. Preliminaries: Smoothed Particle Hydrodynamics

This section repeats the foundations of Smoothed Particle Hydrodynamics (SPH), based on the version by Müller et al. [28]. 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. Many fluid simulations focus on the following *Navier-Stokes equation*, which describes the conservation of momentum in an incompressible fluid:

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

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 the density ρ_i, pressure ∇p_i, and velocity fluctuation $\nabla^2 \mathbf{v}_i$ for each particle.

3.2. SPH approximation equation

In SPH, any quantity A at a position $\mathbf{r}$ is approximated by a weighted sum over all particles:

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

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. 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. It is possible to use different kernel functions W for different purposes. Section 5.2 will list the specific functions used in our implementation.

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.

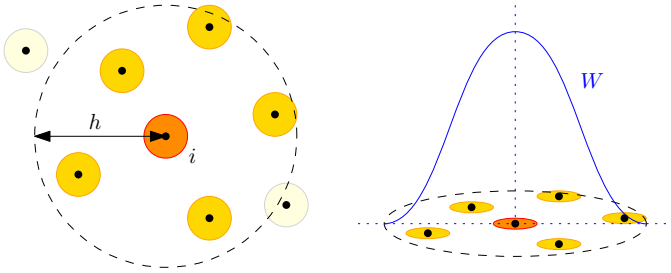


Fig. 1. Principle of an SPH kernel. Left: 2D view of a particle i and its neighbors. All particles within the radius h are considered by SPH. Right: Perspective view including the kernel function W. A particle's contribution to any quantity (such as the density) depends on the distance to particle i.

3.3. Computing density, pressure, and forces

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

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

Thus, the density at any position $\mathbf{r}$ is approximated by 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. [28] suggest the following adaptations of Eq. (3):

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

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

3.4. Obstacle handling

One difficulty of SPH lies in how to treat *obstacles*, so that particles near obstacles perceive a meaningful density. A common solution, which we used in our previous work as well [4], is to place 'boundary particles' at regularly sampled positions inside obstacles. In each simulation frame, these boundary particles compute the SPH density and pressure just like other particles, but they do not move.

Particle-based obstacle handling is conceptually easy and it works sufficiently well for many applications. Possible disadvantages are its computational overhead and a lack of accuracy. For our specific application to crowds, there are several more disadvantages. First, boundary particles impose design choices

that need to be tuned per scenario, such as the number of particles and their SPH parameters. Suboptimal settings can lead to artifacts near obstacles (e.g. overcrowding or gaps), which are more problematic for crowds than for purely visual fluid simulations. Second, it is unfortunate to require different obstacle representations for different tasks. Compared to particles in a fluid, agents perform many more tasks (such as collision avoidance) for which they are already using a 2D polygon representation of obstacles. It would be ideal to use the same representation in the SPH part of the simulation.

Several other methods for SPH obstacle handling have been presented in recent years. For example, Bender et al. [34] have proposed *volume maps*: a grid in which the obstacle contribution to SPH kernels is precomputed per grid cell. This solves the problems of accuracy and parameter tuning, but it still uses a separate obstacle representation that we prefer to avoid.

We refer the reader to the survey by Koschier et al. [29] for a more complete overview of SPH obstacle handling. In Section 4.3, we will present a specific method for handling the 2D polygonal obstacles that are typical for crowd simulation.

3.5. 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 ρ_i and p_i *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. Combined simulation loop

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. Next to collision avoidance, agents may also want to satisfy other local criteria, such as staying in a group with other agents. Each agent can have its own criteria and its own algorithms for satisfying them: this is a key advantage of agent-based simulation. This leads to a simulation loop where each iteration (or *frame*) 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, compute a *preferred velocity* $\mathbf{v}_i^{\text{pref}}$ that does not consider the agent's neighbors yet. This velocity can point either straight to the goal $\mathbf{g}_i$ or along a precomputed path.
3. For each agent i, compute an acceleration $\mathbf{a}_i$ that satisfies the desired local criteria (such as collision avoidance), based on $\mathbf{v}_i^{\text{pref}}$ and the agent's neighbors. This step can contain multiple substeps, depending on the local criteria and the algorithms used.
4. 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).
5. Move all agents using a time-integration method of choice.

To integrate SPH into this 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 algorithms. Concretely, the following actions should be added after steps 1 and 3 of the loop:

1B. For each agent i, compute the SPH density ρ_i (Eq. (4)), the personal rest density ρ_i^0 (to be discussed in Section 4.2), and the pressure p_i (Eq. (5)).

3B. 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. Dynamic personal rest density

To make the SPH model more suitable for crowds, we make a 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. In the crowd simulation loop, this rest density should be updated between the density ρ_i and the pressure p_i, so that the pressure is always based on an up-to-date rest density.

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). We propose the following simple option: each agent i maintains 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:

$$\hat{\rho}_i := \left(1 - \frac{\Delta t}{T^\rho}\right) \cdot \hat{\rho}_i + \frac{\Delta t}{T^\rho} \cdot \rho_i \,. \tag{8}$$

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 a predefined range $[\rho^{0,\min}, \rho^{0,\max}]$. In our experiments, we will treat $\rho^{0,\max}$ as a parameter while keeping $\rho^{0,\min}$ and T^ρ constant. Section 6.1 will show that $\rho^{0,\max}$ (the *maximum rest density*) is an intuitive parameter for controlling the density of the crowd.

4.3. SPH obstacle handling

As mentioned in Section 3.4, the most common ways to handle *obstacles* in SPH have several disadvantages for our purpose. We now present an obstacle-handling approach that is based directly on the 2D polygonal obstacles in the environment. Our technique borrows several concepts from volume maps [34], but without requiring a grid representation.

We assume that the environment's obstacles are given as a set of non-overlapping 2D polygons. Agents can then treat each boundary segment of each polygon separately. We will refer to these boundary segments as *walls*. Let $\{O_k\}_{k=0}^{M-1}$ be the set of all M walls. This representation is common in crowd simulation: for example, walls are already commonly used as input for collision avoidance and contact forces.

4.3.1. Density contribution of obstacles

Overall, we extend the SPH density equation (Eq. (4)) so that it combines the usual contributions of particles (agents) with the contributions of obstacles:

$$\rho_i \approx \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h) + \sum_k \rho_k^{\text{wall}}(\mathbf{r}_i) \,. \tag{9}$$

We will now define the density contribution ρ_k^{wall} for a single wall O_k. First, an agent i should only take O_k into account if O_k lies (partly or fully) inside the kernel radius, i.e. inside the disk δ_i with centroid $\mathbf{r}_i$ and radius h. When this is the case, we assume that all points in δ_i *behind* O_k are obstacle points. Let P_k be this set of points, i.e. the set of all points inside δ_i that are invisible from $\mathbf{r}_i$ due to O_k. Fig. 2 shows an example. Let $a(P_k)$ be the surface area of P_k in square meters. This area can be computed via simple geometric operations.

To compute the contribution of O_k to ρ_i, we need to approximate an integral over all points in P_k. We employ two approximations inspired by volume maps [34]. First, we assume that the density inside P_k is equal to the agent's current rest density ρ_i^0. Second, we condense all of P_k into a single representative point $\mathbf{r}_k^\rho$. This allows for the following simplifications:

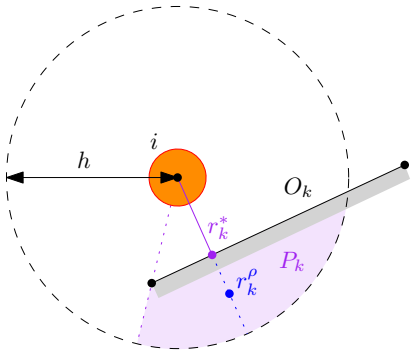


Fig. 2. To handle a wall O_k in SPH, we assume that O_k occupies the area P_k (highlighted in purple) inside the kernel of agent i. We choose a point r_k^ρ that represents the total contribution of O_k to density.

$$\begin{aligned}
\rho_k^{\text{wall}}(\mathbf{r}_i) &= \int_{P_k} \rho(\mathbf{x}) W(\mathbf{r}_i - \mathbf{x}, h)\, d\mathbf{x} \\
&\approx \rho_i^0 \int_{P_k} W(\mathbf{r}_i - \mathbf{x}, h)\, d\mathbf{x} \\
&\approx \rho_i^0\, W(\mathbf{r}_i - \mathbf{r}_k^\rho, h) \int_{P_k} d\mathbf{x} \\
&= \rho_i^0\, a(P_k)\, W(\mathbf{r}_i - \mathbf{r}_k^\rho, h) \,.
\end{aligned} \tag{10}$$

The representative point $\mathbf{r}_k^\rho$ should be chosen in such a way that its kernel weight $W(\mathbf{r}_i - \mathbf{r}_k^\rho, h)$ is a good approximation of the average kernel weight inside P_k. Therefore, the best choice for $\mathbf{r}_k^\rho$ is kernel-dependent. For smooth kernels (such as the one used in our implementation), we suggest to choose $\mathbf{r}_k^\rho$ as the point exactly between the *nearest* obstacle point $\mathbf{r}_k^*$ and the edge of the kernel, as shown in Fig. 2.

4.3.2. Pressure contribution of obstacles

To define the SPH pressure force $-\nabla p_i$ with obstacles included, we apply a very similar process. We extend the SPH pressure-force equation (Eq. (6)) with a force per wall:

$$-\nabla p_i \approx -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) + \sum_k \mathbf{f}_k^{\text{wall}}(\mathbf{r}_i) \,. \tag{11}$$

To define the pressure force $\mathbf{f}_k^{\text{wall}}$ exerted by a wall O_k, we condense the obstacle region P_k into a representative point $\mathbf{r}_k^p$, and we further assume that the pressure inside P_k is equal to p_i. This 'pressure mirroring' is used in volume maps as well [34]. This leads to the following equation for $\mathbf{f}_k^{\text{wall}}$:

$$
\begin{aligned}
\mathbf{f}_k^{\text{wall}}(\mathbf{r}_i) &= -\int_{P_k} \rho(\mathbf{x})\, \frac{p_i + p(\mathbf{x})}{2\rho(\mathbf{x})}\, \nabla W(\mathbf{r}_i - \mathbf{x}, h)\, d\mathbf{x} \\
&\approx -\int_{P_k} \frac{p_i + p_i}{2}\, \nabla W(\mathbf{r}_i - \mathbf{x}, h)\, d\mathbf{x} \\
&= -p_i \int_{P_k} \nabla W(\mathbf{r}_i - \mathbf{x}, h)\, d\mathbf{x} \\
&\approx -p_i\, \nabla W(\mathbf{r}_i - \mathbf{r}_k^p, h) \int_{P_k} d\mathbf{x} \\
&= -p_i\, a(P_k)\, \nabla W(\mathbf{r}_i - \mathbf{r}_k^p, h)\ .
\end{aligned}
\tag{12}
$$

The representative point is $\mathbf{r}_k^p$ technically kernel-dependent again. However, we obtain good results by using $\mathbf{r}_k^p = \mathbf{r}_k^\rho$, even though we use very different kernels for the density and the pressure force (see Section 5.2). This suggests that this uniform choice will work sufficiently well in most cases.

4.3.3. Relation to other methods

Another polygon-based technique is used by Golas et al. [26] in their macroscopic simulation of stress and turbulence in crowds. For the center $\mathbf{r}_c$ of each cell c in a grid, they compute the fraction of the neighborhood of $\mathbf{r}_c$ that is occupied by obstacles. They compute the density at $\mathbf{r}_c$ using agents only, and they scale the result by this obstacle occupancy. In comparison, we compute the contribution of obstacles more accurately, by using kernels and representative points per wall.

The main difference between our technique and volume maps [34] is that a volume map stores the SPH contributions of obstacles in a precomputed grid. By contrast, we compute this obstacle contribution per agent in each frame. Although this may be computationally heavier, it works directly with the data that agents in a crowd are already using, without requiring additional storage. Our method can therefore be more easily integrated into a microscopic crowd simulation.

4.4. Density-dependent behavior blending

Finally, we present a simple mechanism for letting an agent *blend* between (active) navigation and (passive) interaction forces depending on the SPH density. In practice, a fixed combination of local navigation with SPH (as presented in Section 4.1) does not always work well. For example, when using an advanced collision-avoidance algorithm such as RVO [12], agents will avoid collisions at all costs, and high-density scenarios do not get the chance to occur. In such cases, we need to model explicitly how a crowd becomes more 'fluid-like' as the density increases. This allows for the microscopic simulation of *mixed-density* scenarios.

To describe our blending concept generically, let a *profile* be a combination of local navigation algorithms, SPH parameters, and contact-force parameters. Thus, a profile is a specific way to fill in Steps 1B, 3, 3B, and 4 of the simulation loop for a single agent (see Section 4.1). In each frame of the simulation loop, the task of a profile for agent i is to compute an overall acceleration vector $\mathbf{a}_i$.

Now, instead of using a constant profile per agent, let us give each agent i a *sequence* of h profiles with corresponding density values, $\left\{\left(\Pi_i^j, \rho_i^j\right)\right\}_{j=0}^{h-1}$, ordered by their density values ρ_i^j in increasing order. At any moment in the simulation, let $\mathbf{a}_i^j$ be the acceleration that profile Π_i^j would produce. Using the current SPH density ρ_i, we define the agent's behavior as a linear interpolation of the nearest profiles:

- If $\rho_i \le \rho_i^0$, then $\mathbf{a}_i = \mathbf{a}_i^0$.
- Otherwise, if $\rho_i > \rho_i^{h-1}$, then $\mathbf{a}_i = \mathbf{a}_i^{h-1}$.
- Otherwise, $\rho_i^j < \rho_i \le \rho_i^{j+1}$ for some value of j. In this case, $\mathbf{a}_i = (1-\kappa)\mathbf{a}_i^j + \kappa \mathbf{a}_i^{j+1}$, where $\kappa = (\rho_i - \rho_i^j)/(\rho_i^{j+1} - \rho_i^j)$.

Note that using a single profile ($h = 1$) yields standard behavior without any density-dependent blending. To interpolate between collision avoidance and SPH, it suffices to employ two profiles ($h = 2$) where Π_0 uses collision avoidance and Π_1 uses SPH. We will show an example of this in our experiments.

This paper focuses on enabling agent-based crowd simulation at extreme densities. We will therefore blend specifically between individual behavior and fluid-like interactions. However, we have described the concept of blending more generically for completeness. We expect that density-dependent interpolation of algorithms can be useful for more purposes in future work.

5. Implementation and settings

We have integrated SPH into UMANS [35], 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 4.1. The simulation loop uses forward Euler integration. We will now discuss all relevant settings and implementation details.

In general, it is important to note that each agent i is approximated by a disk with radius D_i. This approximation is used for contact forces and in one of the collision-avoidance algorithms that we employ (RVO). However, it is not used in SPH and the other collision-avoidance method (social forces).

For simplicity, we do not include any form of global path planning in our experiments. Thus, the preferred velocity $\mathbf{v}_i^{\text{pref}}$ of an agent i always points straight to the goal $\mathbf{g}_i$. The inclusion of path planning would introduce even more variables that would make our results difficult to analyze.

5.1. Multiple framerates

To produce stable results at high densities, several simulation aspects require a higher simulation framerate than the 10 FPS that is common for lower-density crowd simulations. However, for performance reasons, the usual low framerate is still preferred for nearest-neighbor queries (Step 1) and for the RVO collision-avoidance algorithm (part of Step 3 in certain navigation profiles). We therefore use two different Δt values: a small value Δt_{fine} that is used for most steps, and a larger value Δt_{coarse} for Step 1 and for RVO. Our simulation loop uses Δt_{fine}

overall, but a coarse step re-uses its result from the previous frame whenever less than Δt_{coarse} seconds have passed. For an efficient implementation, Δt_{coarse} should be an exact multiple of Δt_{fine}. Section 6.4 will show that this combination of frequencies enables real-time simulations of large dense crowds.

5.2. SPH details

For the SPH smoothing kernel W, we use the different functions suggested by Müller et al. [28], adapted to a 2D domain [36]. More precisely, we use 2D versions of the 'Poly6' kernel for density, the 'spiky' kernel for pressure, and Müller's kernel for viscosity:

$$W^p(\mathbf{r}, h) = \frac{4}{\pi h^8} \begin{cases} (h^2 - \|\mathbf{r}\|^2)^3, & \text{if } \|\mathbf{r}\| < h \\ 0, & \text{otherwise;} \end{cases} \tag{13}$$

$$\nabla W^p(\mathbf{r}, h) = -\frac{30}{\pi h^5} \cdot \frac{\mathbf{r}}{\|\mathbf{r}\|} \begin{cases} (h - \|\mathbf{r}\|)^2, & \text{if } \|\mathbf{r}\| < h \\ 0, & \text{otherwise;} \end{cases} \tag{14}$$

$$\nabla^2 W^{\text{visc}}(\mathbf{r}, h) = \frac{360}{29\pi h^5} \begin{cases} h - \|\mathbf{r}\|, & \text{if } \|\mathbf{r}\| < h \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

For all kernels, we will use $h = 1$ m as the kernel support radius. Recall that an SPH particle takes all neighbors and walls within this radius into account. Using $h = 1$ m yields a good balance between smoothness and performance.

Also, a commonly used technique in SPH is to *ignore negative pressure* [29]. 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.

5.3. Collision avoidance

We will use two different collision-avoidance methods that are representative for the methods typically used in crowd simulation. In both methods, we let agents consider all neighboring agents and walls within a radius of 5 m.

5.3.1. Social force model

The first collision-avoidance method is the *social force model* by Helbing and Molnár [10]. Here, an agent i is steered by a goal-reaching force and by avoidance forces, where each neighboring agent and wall imposes its own force:

$$\mathbf{f}_i = \mathbf{f}_i^{\text{goal}} + \sum_{j \neq i} \mathbf{f}_{ij}^{\text{av,ag}} + \sum_k \mathbf{f}_{ik}^{\text{av,obs}} \,. \tag{16}$$

The goal-reaching force is defined as:

$$\mathbf{f}_i^{\text{goal}} = K^{\text{goal}} \cdot (\mathbf{v}_i^{\text{pref}} - \mathbf{v}_i)/\tau \,, \tag{17}$$

where K^{goal} is a scalar and τ is a *relaxation time* in seconds. For the avoidance forces, many variants have been proposed in literature. We use the original force definitions from 1995 [10] with small adjustments. The avoidance force imposed by an agent j is:

$$\mathbf{f}_{ij}^{\text{av,ag}} = -\nabla_{\mathbf{r}_{ij}} V_0 \, e^{-b/\sigma} \,, \tag{18}$$

where $b = \frac{1}{2}\sqrt{\left(\left\|\mathbf{r}_{ij}\right\| + \left\|\mathbf{r}_{ij} + \mathbf{v}_{ij}T\right\|\right)^2 - \left\|\mathbf{v}_{ij}T\right\|^2}$, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. The scalars V_0, σ, and T are parameters.

Compared to the original version of this model [10], which we also used in our previous work [4], we now use the relative velocity $\mathbf{v}_{ij}$ rather than the absolute velocity $\mathbf{v}_j$. This adjustment greatly improves the behavior of agents. It has been used in most later variants of the social force model as well [37].

The avoidance force imposed by a wall O_k is:

$$\mathbf{f}_{ik}^{\text{av,obs}} = -\nabla_{\mathbf{r}_{ik}} U_0 \, e^{-\|\mathbf{r}_{ik}\|/R} \,, \tag{19}$$

where $\mathbf{r}_{ik} = \mathbf{r}_i - \mathbf{r}_k^*$, and $\mathbf{r}_k^*$ is the point of O_k that is nearest to $\mathbf{r}_i$. The scalars U_0 and R are parameters.

Finally, any avoidance force is scaled down whenever the corresponding neighbor or nearest wall point lies outside a certain field of view. Specifically, we multiply an agent force $\mathbf{f}_{ij}^{\text{av,ag}}$ by 0.5 if the angle between $\mathbf{v}_i$ and $\mathbf{r}_{ij}$ is larger than 100 degrees. Obstacle forces are scaled analogously. This scaling occurs in the original model as well [10].

To ensure stability at high densities, the social force model needs to use the fine simulation step size Δt_{fine}.

5.3.2. RVO

The second collision-avoidance method that we use is *Reciprocal Velocity Obstacles (RVO)* [12], a popular algorithm that is representative of *velocity-based* collision avoidance. In RVO, each agent computes a new velocity by randomly sampling many candidate velocities $\mathbf{v}'$, and then choosing the candidate that minimizes a cost function:

$$C(\mathbf{v}') = \left\|\mathbf{v}' - \mathbf{v}_i^{\text{pref}}\right\| + w \,/\, TTC(i, 2\mathbf{v}' - \mathbf{v}_i) \,, \tag{20}$$

where w is a parameter, and $TTC(i, \mathbf{u})$ is the time to the first collision that agent i will encounter if it uses velocity $\mathbf{u}$. This time-to-collision calculation uses the disk representations of agents, and it assumes that each neighboring agent j maintains its current velocity $\mathbf{v}_j$. The concept of time to collision is used in many collision-avoidance algorithms [35].

Let $\mathbf{v}^* = \arg\min_{\mathbf{v}'} C(\mathbf{v}')$ be the best candidate velocity. The agent will accelerate towards $\mathbf{v}^*$ during Δt_{coarse} seconds:

$$\mathbf{a}_i = (\mathbf{v}^* - \mathbf{v}_i)/\Delta t_{\text{coarse}} \,. \tag{21}$$

Agents perform RVO at the coarse simulation step size Δt_{coarse}. This is common for velocity-based collision avoidance: it does not suffer from the same stability issues as social forces, and a higher framerate would be too computationally expensive.

5.4. Contact forces

For contact forces between colliding agents, we implement the model by Helbing et al. [17] that is commonly used throughout literature. Similarly to avoidance forces in the social force model, contact forces are computed per neighboring agent and wall. However, contact forces use the disk approximation of agents, just like RVO (and unlike social avoidance forces). The contact force imposed by a neighboring agent j is defined as:

$$\mathbf{f}_{ij}^{\text{c,ag}} = K^{\text{ag}} \cdot \max(0, D_i + D_j - \left\|\mathbf{r}_{ij}\right\|) \cdot \frac{\mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|} \,, \tag{22}$$

where K^{ag} is a parameter. The contact force imposed by a neighboring wall O_k is defined as:

$$\mathbf{f}_{ik}^{\text{c,obs}} = K^{\text{obs}} \cdot \max(0, D_i - \|\mathbf{r}_{ik}\|) \cdot \frac{\mathbf{r}_{ik}}{\|\mathbf{r}_{ik}\|}, \tag{23}$$

where K^{obs} is a parameter. A contact force is always $\mathbf{0}$ if there is no collision with the corresponding agent or obstacle.

5.5. Navigation profiles and parameter settings

We combine SPH, collision avoidance, and contact forces to create the following different *profiles* for our experiments:

- *SF*: Agents use only the social force model, and SPH is disabled. For contact forces, we use $K^{\text{obs}} = K^{\text{ag}} = 1000$. The parameters of the social force model are usually tuned per scenario [37, 38]. For its goal-reaching and agent-avoidance forces, the settings from the original article [10] work sufficiently well: $K^{\text{goal}} = 1$, $\tau = 0.5$, $V_0 = 2.1$, $\sigma = 0.3$, and $T = 2$. For its *obstacle*-avoidance forces, the originally suggested settings do not allow a smooth evacuation through the narrow bottleneck of Section 6.1. We use $U_0 = 2.1$ and $R = 0.1$ instead.
- *RVO*: Agents use only the RVO algorithm, and SPH is disabled. Contact-force settings are the same as in the *SF* profile. For RVO, we use $w = 1$ as suggested by its authors [12].
- *SF+SPH* and *RVO+SPH*: Same as the previous two profiles, but with SPH enabled, and with weaker contact forces: $K^{\text{ag}} = 50$, $K^{\text{obs}} = 200$. Default SPH settings: $k = 200$, $\mu = 0$, $\rho^{0,\text{min}} = 0$, $\rho^{0,\text{max}} = 5$, $T^{\rho} = 0.1$.
- *SPH*: Agents use SPH and the goal-reaching component of the social force model, without any collision avoidance. Default settings: same as for *SF+SPH* and *RVO+SPH*.
- *SF→SPH*: Agents perform density-dependent behavior blending (see Section 4.4) using the profile sequence $\{(SF, 2), (SPH, 4)\}$. Thus, agents use the *SF* profile when $\rho_i \leq 2$, they use the *SPH* profile when $\rho_i > 4$, and otherwise they interpolate between the two.
- *RVO→SPH*: Agent perform density-dependent blending using the profile sequence $\{(RVO, 2), (SPH, 4)\}$.

Our preceding conference publication [4] also included a profile that used only goal reaching, so neither collision avoidance nor SPH. We will omit this profile now, as it would never be useful for any scenario where agents actively navigate towards a goal. Instead, we will focus on showing how SPH can improve simulations with commonly used settings.

5.6. Other simulation settings

As explained in Section 5.5, the variables of our experiments will be either the contact-force scalar K^{ag} or several SPH parameters (k, μ, $\rho^{0,\text{max}}$), depending on whether an agent profile uses SPH. All other simulation settings will be treated as constants. Aside from profile-specific constants, we will use the following simulation settings throughout all experiments:

- The timesteps Δt_{fine} and Δt_{coarse}. We will use $\Delta t_{\text{coarse}} = 0.1$ s, a common value for crowd simulation. We will use $\Delta t_{\text{fine}} = 0.02$ s, the largest divisor of Δt_{coarse} that satisfies the CFL stability criteria [29] under our SPH settings. This also leads to a simulation framerate of 50 FPS that is commonly used for physics in game engines.
- 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'. 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 s_{pref}, maximum speed s_{max}, and maximum acceleration a_{max}. We will use $s_{\text{pref}} = 1.4$ m/s, $s_{\text{max}} = 1.8$ m/s, and $a_{\text{max}} = \infty$. The latter is required by the traditional contact-force model, which produces unstable results when the acceleration is capped.

6. Experiments and results

This section demonstrates our model in various scenarios. 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 for agents that do not respond to any SPH forces. In our figures, we will color the agents based on either their velocity or their SPH density. These coloring schemes are shown in Fig. 3. We also encourage the reader to watch the supplementary video of this paper, which shows several of our results in motion.

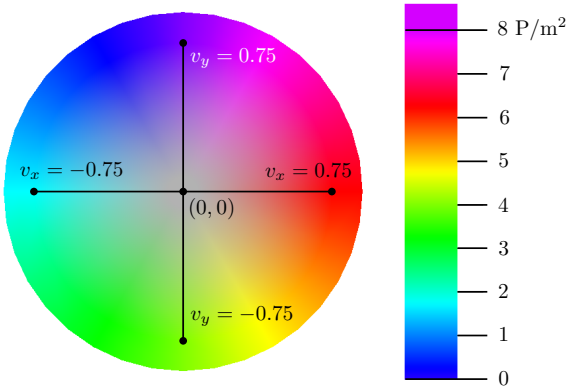
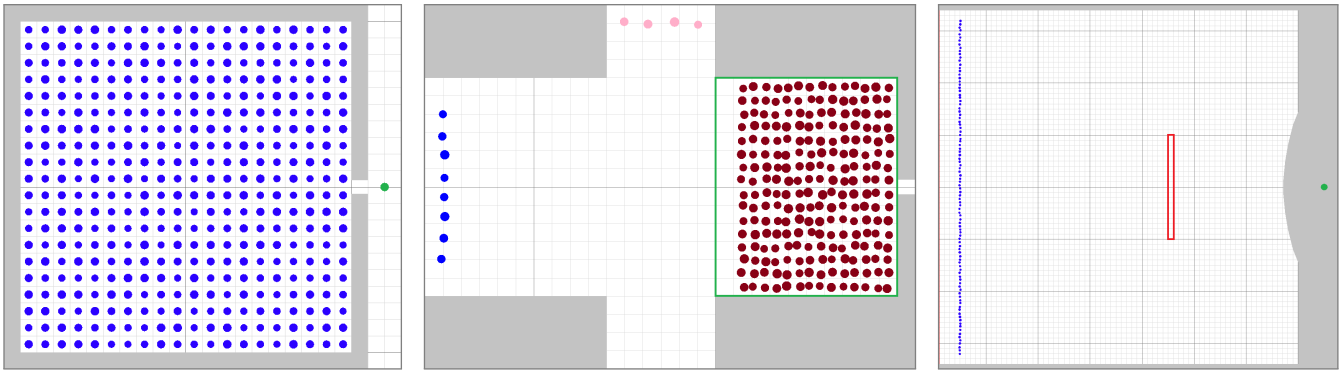


Fig. 3. The two agent-coloring schemes used in this paper. Left: Coloring based on an agent's current velocity. Right: Coloring based on an agent's current SPH density.

6.1. Scenario 1: Room evacuation

In our first scenario, summarized in Fig. 4(a), 400 agents need to exit a room of 20×20 m through a 0.8 m-wide doorway. We set the goal of each agent to a point 1m beyond the exit, and we remove an agent when it is within 0.5 m of this goal. This scenario is motivated by real-world experiments that have been performed in recent years; see e.g. Garcimartín et al. [39].



(a) Room evacuation (Section 6.1) (b) Crossing and bottleneck (Section 6.2) (c) Concert (Section 6.3)

Fig. 4. The start configurations of our experiments. The grid squares measure 1×1 m. (a) 400 agents (shown in blue) need to exit a room. Their goal is shown in green. (b) 224 agents (brown) need to exit the room via the right. On the left, 8 agents (blue) with the same goal appear every two seconds. At the top, 4 downward-moving agents (pink) appear each second. The blue agents need to avoid the pink agents to reach the bottleneck. We will report the average SPH density of agents inside the green rectangle. (c) 100 agents (blue) are added per second during 100 seconds, with an unreachable goal (green) on the stage on the right. After 150 seconds, all agents inside the red rectangle push forward during 0.5 seconds.

Various publications have focused on simulating this scenario, using several variants of social forces and contact forces [17, 37, 40]. We do not claim that SPH produces better behavior than the state of the art. Instead, we intend to show that SPH facilitates generating crowds of a particular density, as it provides more intuitive density control than the usual force parameters.

We run this scenario with all profiles from Section 5.5. In the profiles *without* SPH, we vary the contact-force scalar K^{ag}. In the profiles *with* SPH, we vary the maximum rest density $\rho^{0,\text{max}}$. In each run, we measure the average (SPH) crowd density among all agents after 15 seconds, and we measure the average number of agents that reach the goal per second (the 'flow rate'). The latter measurement starts when the first agent reaches the goal, and it ends when 350 agents have reached the goal. We exclude the last 50 agents because both collision-avoidance methods have problems evacuating the final agents: their evacuations slow down strongly towards the end, and in the case of social forces, the last two agents never pass through the exit. Thus, the last part of such a simulation is not representative of the entire run.

In real-world evacuation experiments, researchers have measured flow rates between 2 and 4 people per second [39]. The range of 'realistic' values for the crowd density is less clear. Evacuation experiments tend to start at a density of around 4 P/m^2, but densities of up to 10 P/m^2 can occur in certain scenarios [39]. Therefore, it is important that a simulation can reproduce a wide range of densities.

6.1.1. Results

Fig. 5 shows each crowd after 15 seconds, using density coloring. Table 1 reports the aforementioned statistics.

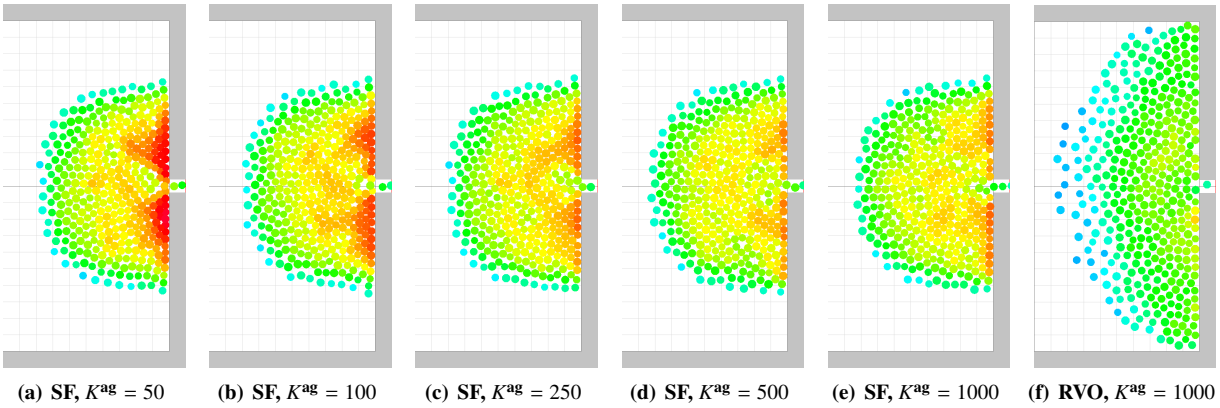
With the *SF* profile (Figs. 5(a)–5(e)), the agents carefully approach the bottleneck, but there are collisions among the agents closest to the wall. K^{ag} then controls how strongly these agents push each other away. Figs. 5(a)–5(e) show that this leads to slightly different densities, but Table 1 indicates that these differences are small. One interesting effect is that stronger contact forces decrease the flow rate because they lead to more and longer temporary clogging near the exit. More specifically, two agents can get stuck when they try to pass through the exit at the same time, and then stronger contact forces (i.e. less compressible agents) cause this conflict to last longer.

As announced in Section 4.4, the RVO collision-avoidance algorithm is intent on avoiding collisions at all costs. With the *RVO* profile (Fig. 5(f)), agents therefore avoid a high-density situation altogether. This leads to a very slow evacuation with a low flow rate. Because collisions are always avoided, we do not report on different values of the contact-force scalar K^{ag}.

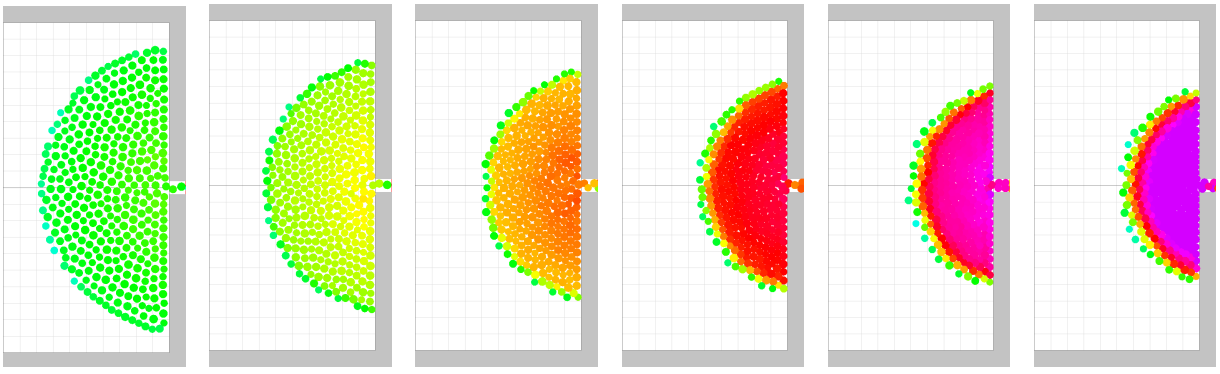
With the *SPH* profile (Figs. 5(g)–5(l)), $\rho^{0,\text{max}}$ controls the maximum density that the crowd wishes to attain. Depending on $\rho^{0,\text{max}}$, agents automatically keep distance from one another, or they allow their disks to overlap. Table 1 shows that the average crowd density is indeed close to $\rho^{0,\text{max}}$. Also, because the contact forces are now very weak, the clogging at the exit observed for *SF* does not occur here. However, for high values of $\rho^{0,\text{max}}$, the flow rate becomes unrealistically high.

When we combine SPH with collision avoidance in a straightforward way (i.e. without density-dependent blending), SPH can help limit the density in the crowd. For the *SF+SPH* profile (Figs. 5(m)–5(o)), a low value of $\rho^{0,\text{max}}$ ensures that the density in the crowd remains low. As $\rho^{0,\text{max}}$ increases, we approach the results of the regular *SF* profile. For *RVO+SPH* (Figs. 5(p)–5(r)), we see a similar effect. A low $\rho^{0,\text{max}}$ leads to a lower density than when using RVO only, but a high $\rho^{0,\text{max}}$ does not lead to a *higher* density than with RVO only, because the collision-avoidance algorithm prevents this. Furthermore, due to RVO, the flow rate remains undesirably low; the combination with SPH does not change this.

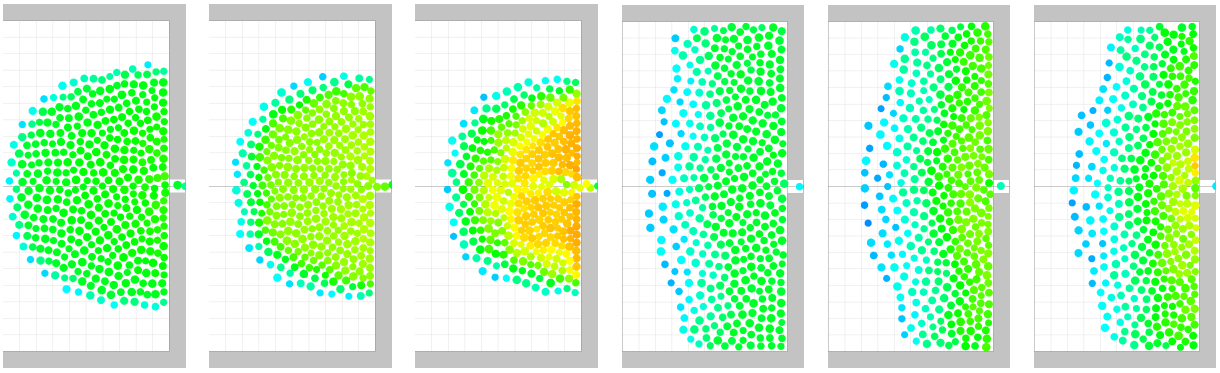
Finally, with the density-dependent behavior blending of *SF→SPH* and *RVO→SPH* (Figs. 5(s)–5(x)), SPH overtakes collision avoidance as the crowd density increases. As such, the crowd compresses to a density that can be expected of $\rho^{0,\text{max}}$, just like in the profile with SPH only. At the boundary of the crowd, the motion of agents is still dominated by the collision-avoidance algorithm.



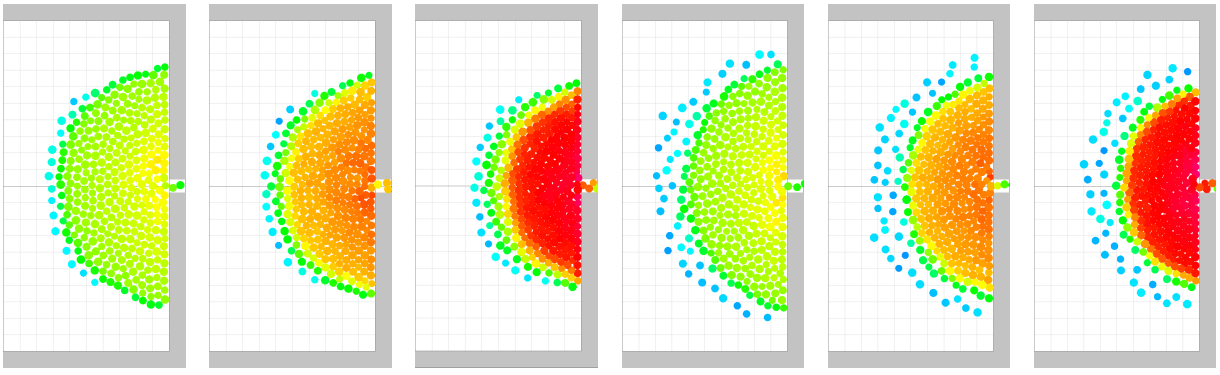
(a) SF, $K^{ag} = 50$ (b) SF, $K^{ag} = 100$ (c) SF, $K^{ag} = 250$ (d) SF, $K^{ag} = 500$ (e) SF, $K^{ag} = 1000$ (f) RVO, $K^{ag} = 1000$



(g) SPH, $\rho^{0,max} = 3$ (h) SPH, $\rho^{0,max} = 4$ (i) SPH, $\rho^{0,max} = 5$ (j) SPH, $\rho^{0,max} = 6$ (k) SPH, $\rho^{0,max} = 7$ (l) SPH, $\rho^{0,max} = 8$



(m) SF+SPH, $\rho^{0,max}$=3 (n) SF+SPH, $\rho^{0,max}$=4 (o) SF+SPH, $\rho^{0,max}$=5 (p) RVO+SPH, $\rho^{0,max}$=3 (q) RVO+SPH, $\rho^{0,max}$=4 (r) RVO+SPH, $\rho^{0,max}$=5



(s) SF→SPH, $\rho^{0,max}$=4 (t) SF→SPH, $\rho^{0,max}$=5 (u) SF→SPH, $\rho^{0,max}$=6 (v) RVO→SPH, $\rho^{0,max}$=4 (w) RVO→SPH, $\rho^{0,max}$=5 (x) RVO→SPH, $\rho^{0,max}$=6

Fig. 5. Snapshots of the evacuation experiment. Each subfigure shows a particular simulation after 15 s, using density coloring. (a)-(e) Social forces only. (f) RVO only. (g)-(l) SPH only. (m)-(o) Social forces with SPH. (p)-(r) RVO with SPH. (s)-(u) Blending between social forces and SPH. (v)-(x) Blending between RVO and SPH.

Table 1. Results of the room evacuation experiment (Section 6.1). Columns 2 to 4 show the number of agents that reach the goal, the average SPH density among all agents at t = 15s (with standard deviations), and the average number of evacuees per second (up to the 350th evacuated agent).

Profile	Specific settings	#Evac. agents	Avg. density (P/m^2) after 15s	Avg. flow rate (P/s)
SF	$K^{\text{ag}} = 50$	398	4.39 [1.05]	2.70
	$K^{\text{ag}} = 100$	398	4.32 [1.00]	2.36
	$K^{\text{ag}} = 250$	398	4.25 [0.92]	2.01
	$K^{\text{ag}} = 500$	398	4.19 [0.85]	1.49
	$K^{\text{ag}} = 1000$	398	4.21 [0.85]	1.29
RVO	$K^{\text{ag}} = 1000$	400	3.23 [0.81]	0.97
SPH	$\rho^{0,\max} = 3$	400	3.27 [0.27]	3.08
	$\rho^{0,\max} = 4$	400	4.21 [0.45]	4.17
	$\rho^{0,\max} = 5$	400	5.09 [0.64]	4.89
	$\rho^{0,\max} = 6$	400	5.89 [0.90]	5.80
	$\rho^{0,\max} = 7$	400	6.61 [1.25]	6.45
	$\rho^{0,\max} = 8$	400	7.23 [1.60]	7.20
SF+SPH	$\rho^{0,\max} = 3$	398	3.03 [0.39]	2.34
	$\rho^{0,\max} = 4$	398	3.70 [0.74]	2.63
	$\rho^{0,\max} = 5$	398	4.13 [1.01]	2.73
	$\rho^{0,\max} = 6$	398	4.24 [1.17]	2.73
	$\rho^{0,\max} = 7$	398	4.28 [1.18]	2.78
	$\rho^{0,\max} = 8$	398	4.23 [1.23]	2.79
RVO+SPH	$\rho^{0,\max} = 3$	400	2.61 [0.52]	1.43
	$\rho^{0,\max} = 4$	400	2.96 [0.81]	1.42
	$\rho^{0,\max} = 5$	400	3.00 [0.91]	1.43
	$\rho^{0,\max} = 6$	400	3.04 [0.92]	1.43
	$\rho^{0,\max} = 7$	400	3.04 [0.92]	1.44
	$\rho^{0,\max} = 8$	400	3.06 [0.89]	1.44
SF→SPH	$\rho^{0,\max} = 4$	400	4.07 [0.65]	3.72
	$\rho^{0,\max} = 5$	400	4.79 [0.98]	4.43
	$\rho^{0,\max} = 6$	400	5.32 [1.34]	4.85
RVO→SPH	$\rho^{0,\max} = 4$	400	3.85 [0.95]	3.49
	$\rho^{0,\max} = 5$	400	4.58 [1.30]	4.39
	$\rho^{0,\max} = 6$	400	5.20 [1.70]	4.43

6.1.2. Discussion

The results *without* blending show that SPH can help limit the crowd density. The maximum rest density $\rho^{0,\max}$ is an in intuitive parameter for this, whereas the contact-force scale K^{ag} and the parameters of social forces and RVO have a less obvious meaning. However, adding SPH does not suddenly allow *higher* densities that are already prevented by collision avoidance. This is particularly visible for the *RVO+SPH* profile, where RVO prevents high-density situations altogether. To explicitly allow higher densities, one possibility is to interpolate from collision avoidance to SPH via *density-dependent blending*. We will show another example of this in the next scenario.

We do acknowledge that this is merely one example, and that there are many other parameters that we could have varied, e.g. those of the social force model and of density-dependent blending. Even with such additional experimentation, it would be difficult to say in general whether an SPH-enriched model is less or more *realistic*. As mentioned earlier, real-world measurements for evacuations vary strongly in terms of density and flow rate [39]. However, the main benefit of SPH lies in *density control* via the intuitive parameter $\rho^{0,\max}$. 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. It can also help improve our understanding of how exactly crowds become more similar to fluids as the density increases.

6.2. Scenario 2: Crossing and bottleneck

Table 2. Results of the 'crossing with bottleneck' experiment (Section 6.2). Column 2 shows the average SPH density at t = 45s in the rectangle of Fig. 4(b), and its standard deviation. Column 3 shows the average numer of evacuees per second during the first minute.

Profile	Avg. density (P/m^2) after 45s	Avg. flow rate (P/s) in first 60s
SF	4.00 [1.09]	1.71
RVO	3.21 [0.77]	1.18
SPH	4.14 [1.33]	4.72
SF+SPH	3.81 [1.27]	2.85
RVO+SPH	3.05 [0.80]	1.71
SF→SPH	4.11 [1.77]	4.56
RVO→SPH	3.83 [1.55]	4.49

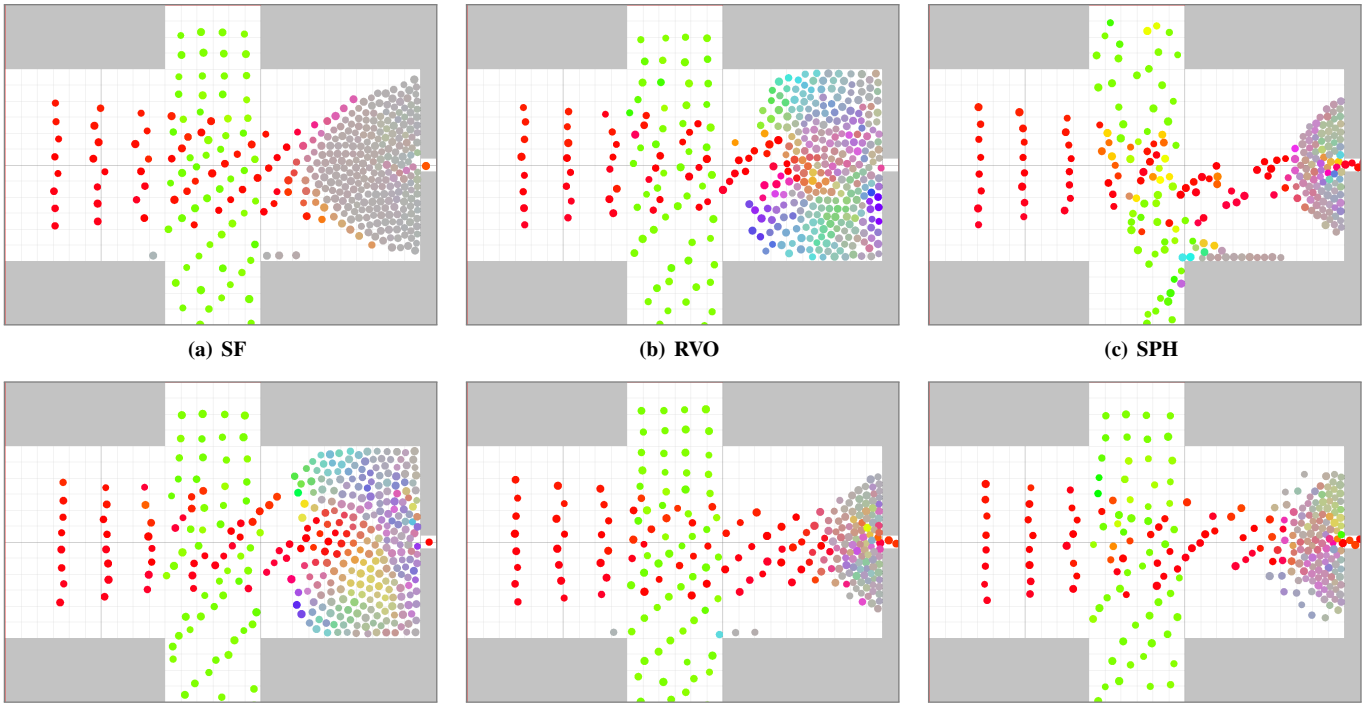
Next, we simulate a mixed-density scenario that combines a room evacuation with a crowd crossing. Fig. 4(b) shows the setup of this scenario. On the right-hand side, a crowd attempts to leave a room through a narrow exit, just like in Section 6.1. On the left, we periodically insert new agents who need to leave through the same exit. However, these agents first need to avoid a crossing flow of agents that are moving vertically. The horizontally-moving agents therefore need to perform both collision avoidance *and* a high-density evacuation.

The purpose of this experiment is to show the importance of density-dependent blending, which lets agents use the right behavior at the right time. We run the scenario with all agent profiles, using their default settings (see Section 5.5). Most notably, $\rho^{0,\max} = 5$ for all profiles that include SPH. To keep the discussion focused, we will not vary the settings per profile.

Fig. 6 shows snapshots of each simulation after 45 seconds. Table 2 reports quantitative results: the average SPH density after 45 seconds among agents inside the rectangle displayed in Fig. 4(b), and the average flow rate during the first 60 seconds. Although these numbers are difficult to compare to those of Section 6.1 (due to the differences between these scenarios), they facilitate the following discussion.

The *SF* profile is not capable of letting all agents pass the crossing. In Fig. 6(a), some of the vertically-moving agents have been pushed too far aside and are now stuck against the lower walls. An advanced collision-avoidance algorithm such as RVO is needed here. However, while the *RVO* profile (Fig. 6(b)) indeed handles the crossing adequately, it produces a very low flow rate at the bottleneck, just like in our first experiment. The *SPH* profile (Fig. 6(c)) offers control over the crowd density at the bottleneck, but it does not contain any collision avoidance, so it cannot handle the crossing.

When using a static combination of collision avoidance and SPH, the crowd's motion seems less chaotic than when using collision avoidance only. For example, with the *RVO+SPH* profile (Fig. 6(d)), agents coming from the left blend more smoothly into the group of agents waiting at the exit. This leads to slightly lower densities and a higher flow rate. Still, as in Section 6.1, the maximum rest density $\rho^{0,\max}$ is not achieved.



(a) SF (b) RVO (c) SPH

(d) RVO+SPH (e) SF→SPH (f) RVO→SPH

Fig. 6. Snapshots of the 'crossing and bottleneck' experiment (Section 6.2). Each subfigure shows the simulation with a particular agent profile after 45 seconds, using velocity coloring.

Finally, the *SF→SPH* and *RVO→SPH* profiles (Figs. 6(f) and 6(f)) blend between collision avoidance and SPH depending on the density. The horizontally-moving agents first use social forces or RVO to avoid the crossing crowd flow, and then they interpolate to SPH as they join the dense crowd on the right. In the case of *RVO→SPH*, we obtain successful collision avoidance at the crossing *and* density control at the bottleneck.

This is a simple example to show the benefit of density-dependent behavior in mixed-density scenarios. In future work, we plan to extend this concept to larger scenarios with tens of thousands of agents. As we will explain in Section 7, simulating such scenarios in *real-time* will require more improvements on a technical level. Our next and final experiment will focus purely on SPH again.

6.3. Scenario 3: Concert with shockwave

Finally, we simulate a large dense crowd attending a concert, where some of the attendees initiate a *crowd shockwave*. This is a phenomenon very specific to extreme-density crowds, where a pushing motion propagates through the crowd and causes people to be transported involuntarily. This effect has been observed and recorded in real life, such as at a concert in 2005.[1]

We will use SPH to simulate this shockwave effect. The purpose of our experiment is to investigate the role of SPH parameters on the wave, and to show that SPH and contact forces combined yield better behavior than either component on its own. With this purpose in mind, we focus on the *SPH* agent profile only, and we will vary its parameter settings per run. Once again, we refer to our supplementary video as well; many of our results are better observed in motion.

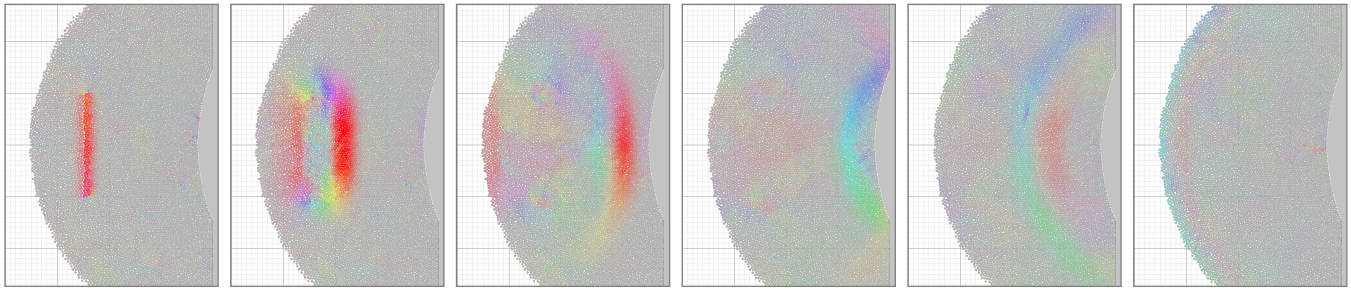
Fig. 4(c) shows our simulation set-up. We add 10,000 agents (in rows of 100 agents per second) with a goal position on the stage on the right. The agents will try to get as close to the stage as possible, but the stage itself is an obstacle. We set the goal-reaching scalar K^{goal} to 0.1, so that agents do not move to the stage too aggressively. Eventually, the crowd converges to a shape concentrated around the stage. 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 letting them ignore all contact and SPH forces. The agents keep these settings for 0.5 s and then revert to their old settings.

6.3.1. Controlling the crowd: Gas constant and rest density

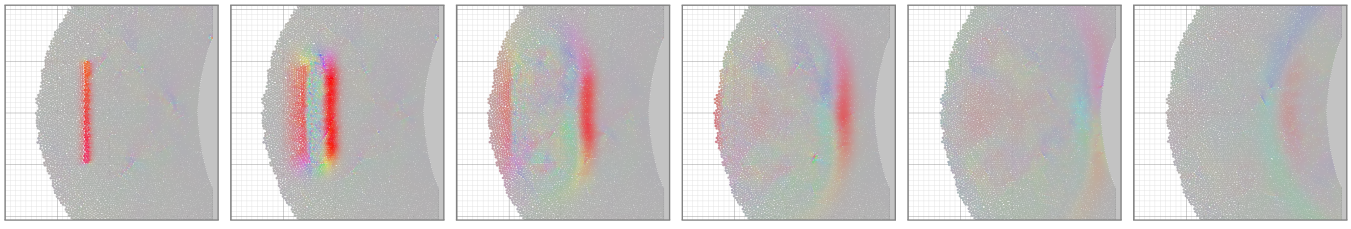
Fig. 7 shows how the shockwave propagates through the crowd using several SPH parameter variations. In each version, a clearly visible wave moves towards the stage, bounces off, and then moves backward. This is consistent with real-world recordings [2]. For the baseline simulation in Fig. 7(a), we use the standard *SPH* profile but with viscosity included ($\mu = 5$).

Figs. 7(b) and 7(c) show that the *gas constant* k affects both the speed and the size of the wave. With a higher gas constant, the wave propagates faster, and therefore the wavefront is thicker, as more agents are experiencing the push at the same time. With a lower gas constant, the wave is slower and therefore thinner. However, the range of useful values for k is limited. If k is too high, the simulation may require a smaller

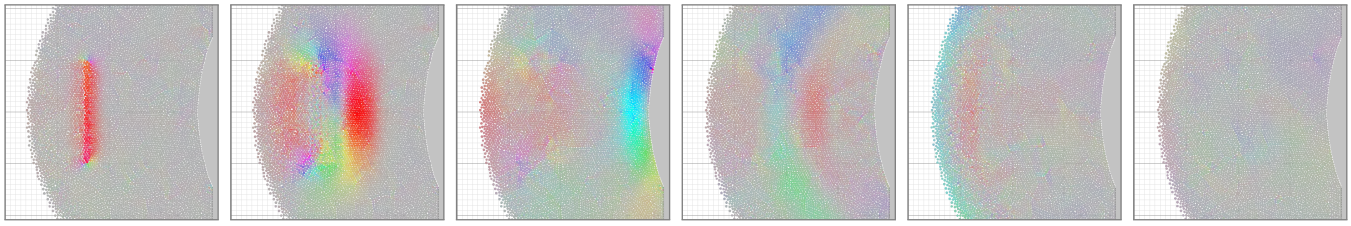
[1] See `https://youtu.be/BgpdmAtbhbE` for footage of the crowd at this concert. See Bottinelli and Silverberg [2] for an analysis of this video.



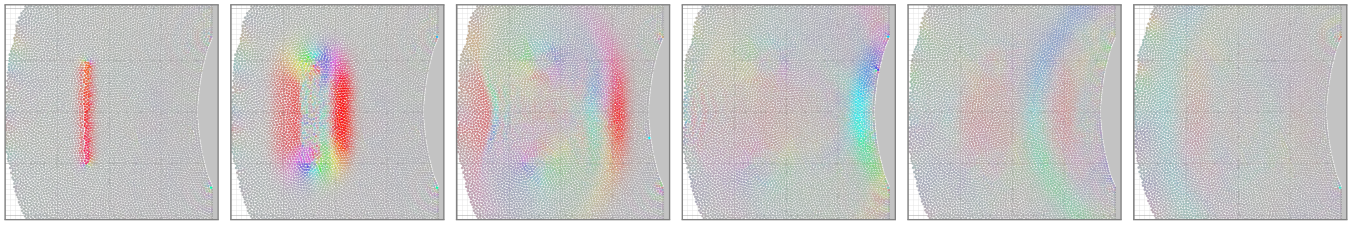
(a) $\rho^{0,\max} = 5$, $k = 200$, $\mu = 5$ **(baseline)**



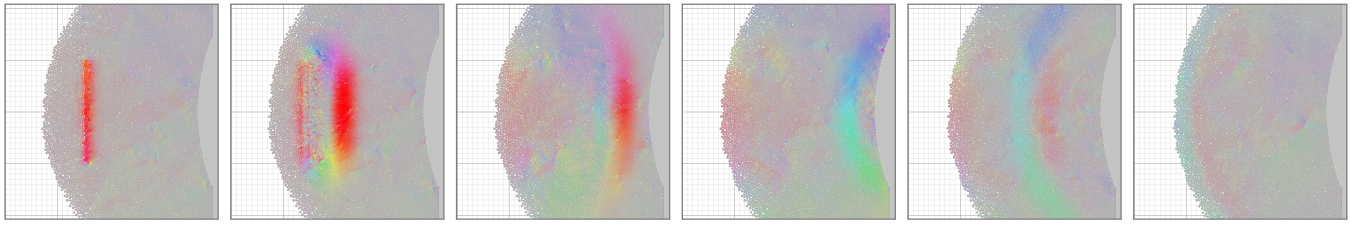
(b) $k = 50$



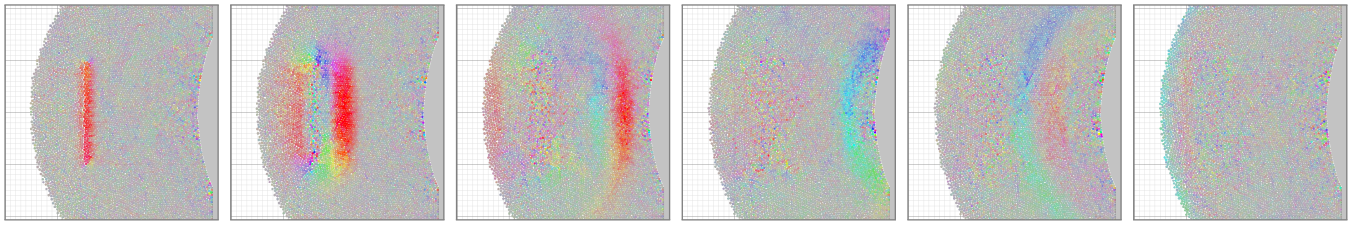
(c) $k = 500$



(d) $\rho^{0,\max} = 4$



(e) $\rho^{0,\max} = 6$



(f) $\mu = 0$

Fig. 7. Snapshots of the concert experiment (Section 6.3). The shockwave is triggered at $t = 150$ **s. Each row shows the crowd at (from left to right)** $t = 150.3$, 151, 152, 153, 154, **and** 156 **s, using velocity coloring. (a) Baseline version of the simulation. The other rows differ from this simulation by one parameter. (b)-(c) Using a smaller or larger gas constant** k**. (d)-(e) Using a smaller or larger maximum rest density** $\rho^{0,\max}$**. (f) Without viscosity.**

timestep Δt_{fine} to remain stable. If k is too low, the density near the stage far exceeds $\rho^{0,\max}$ because SPH can no longer 'defeat' the goal-reaching and contact forces.

Figs. 7(d) and 7(e) show that the *maximum rest density* $\rho^{0,\max}$ affects the crowd density, while only barely affecting the wave. Overall, SPH enables the simulation of the same shockwave at various crowd densities. The density and the wave can be controlled separately via $\rho^{0,\max}$ and k, respectively. These parameters could therefore be tuned to match a simulation with real-world footage.

6.3.2. Stability: Viscosity and dynamic rest density

A small viscosity force (here: $\mu = 5$) improves the crowd's stability. When using $\mu = 0$ (Fig. 7(f)), the same wave still occurs, but many agents show 'jittery' motion, especially after having been exposed to the wave. This follows the consensus that the viscosity force makes SPH more stable [29].

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. 8. The latter artifact occurs because a static rest density ρ^0 is only truly reached *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 as well.

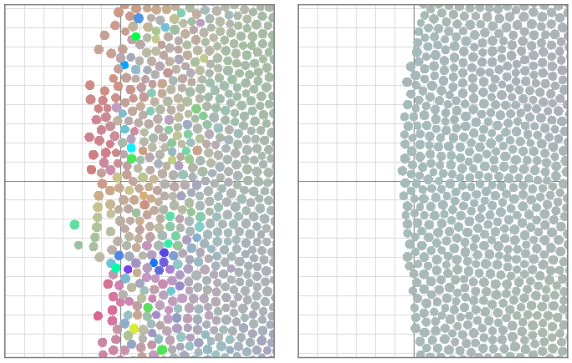


Fig. 8. Left: A static rest density (here: $\rho^0 = 4$) yields chaotic motion at the boundary of the crowd. Right: A dynamic rest density (here: $\rho^{0,\text{min}} = 0, \rho^{0,\text{max}} = 4$) gives stable results.

6.3.3. The role of contact forces

So far, we have deliberately used contact forces and SPH forces side by side. We now briefly discuss the importance of both types of forces in this scenario.

Let us first disable SPH completely ($k = 0, \mu = 0$), so that agents only experience a goal-reaching force and contact forces. As in the other SPH-less agent profiles, we use $K^{\text{obs}} = 1000$, and K^{ag} is now the main parameter for controlling the crowd. Fig. 9 shows the shockwave for two different values of K^{ag}.

These simulation variants manage to generate a shockwave as well, but they have several disadvantages. First, just like in Section 6.1, the contact-force scale K^{ag} affects the crowd density in an unintuitive way, so it is difficult to simulate a crowd of a particular density. Second, K^{ag} affects both the density and the wave at the same time. Third, agents shake quite heavily, especially after exposure to the wave, just like in an SPH simulation without viscosity. In short, without SPH, the simulation is less stable, and the wave's behavior is an unintuitive result of the contact forces.

Conversely, let us now disable inter-agent contact forces ($K^{\text{ag}} = 0$) and rely fully on SPH. This allows particles to overlap considerably more near the boundary of the crowd and near obstacles. This amount of overlap may be reasonable for *fluid* simulations where the particles themselves are not directly rendered. However, it is not desirable for *crowd* simulations where the particles represent human bodies. Also, in this scenario, using $K^{\text{ag}} = 0$ leads to errors in which the boundary of the crowd deforms, as shown in Fig. 10. 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. These observations suggests that contact forces actually make SPH more stable as well, and not just the other way around.

6.4. Performance analysis

We conclude this section by looking into the running time of our system. In the concert scenario with 10,000 agents, using the baseline settings from Fig. 7(a), we measure the computation times between $t = 150$ and $t = 180$, i.e. from the moment the wave starts.

6.4.1. Overview

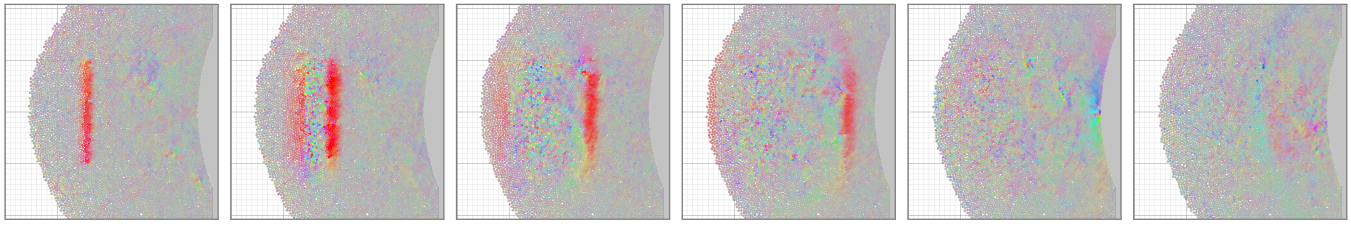
The average running time *per frame* (the 'frame time') of the concert scenario is 4.94 ms. This is less than the simulation timestep Δt_{fine} (20 ms), so the simulation runs in real-time.

For real-time applications, Δt_{fine} should ideally be as large as possible without making the crowd unstable. As mentioned earlier, our use of $\Delta t_{\text{fine}} = 0.02$ (i.e. 50 frames per second) satisfies the stability criteria for SPH, and it is consistent with the framerate of many physics engines. Coarser framerates sometimes gave unstable results, both with and without SPH. This makes sense because we simulate tightly-packed particles.

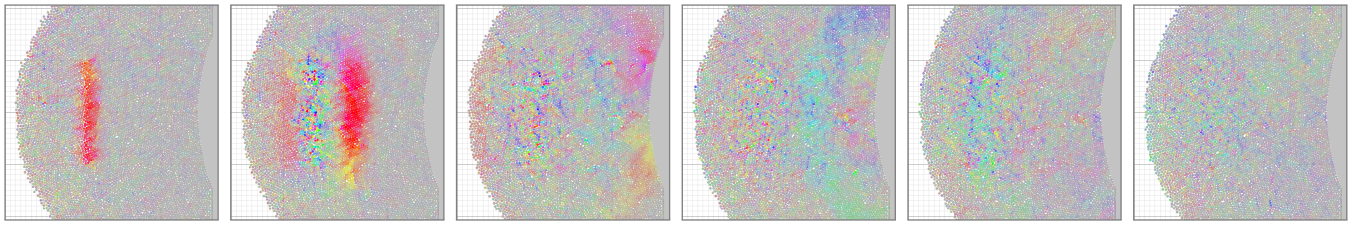
6.4.2. Details and scalability

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), acceleration (5), and the new velocity and position (6). Tasks 1 and 2 use Δt_{coarse} (i.e. they occur once per five frames); the rest uses Δt_{fine}. Task 5 includes all sources of acceleration: SPH, the goal-reaching force, and contact forces. Note that the *tasks*



(a) $K^{\text{ag}} = 200$, $K^{\text{obs}} = 1000$



(b) $K^{\text{ag}} = 1000$, $K^{\text{obs}} = 1000$

Fig. 9. Snapshots of the concert shockwave as in Fig. 7, but without using SPH, and using different values of the agent contact force scalar K^{ag}.

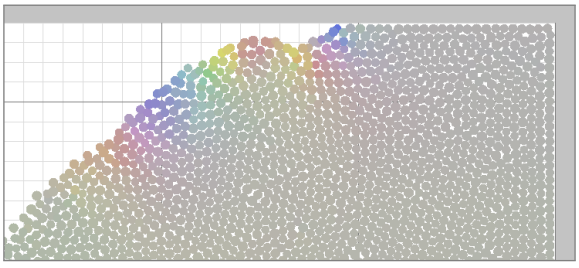


Fig. 10. Without inter-agent contact forces ($K^{\text{ag}} = 0$), we observe deformations at the boundary of the crowd.

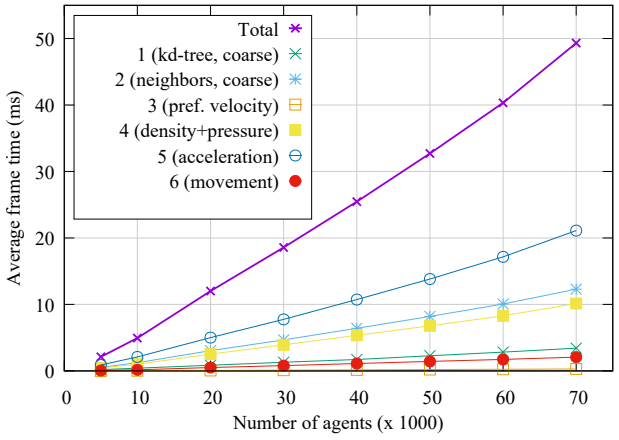


Fig. 11. The relation between the average frame time and the number of agents in the concert scenario.

in this implementation are not exactly the same as the *steps* in the theoretical simulation loop of Section 4.1.

Fig. 11 shows the results. Constructing the *kd*-tree (task 1) is fast, but querying it to find neighbors (task 2) is expensive. These tasks are unavoidable in any agent-based simulation. Using Δt_{coarse} instead of Δt_{fine} keeps their impact manageable. Tasks 3 and 6 take constant time per agent. Finally, tasks 4 and 5 compute certain quantities per agent by looping over its neighbors. Thus, their performance depends on the number of neighbors per agent, and therefore on the crowd density.

With 30,000 agents, the average frame time (18.57 ms) is just under Δt_{fine} (20 ms). Recall from Section 6.3 that the baseline simulation uses $\rho^{0,\max} = 5$, leading to a crowd density of approximately 5 P/m^2. These results suggest that the system can simulate an extreme-density crowd with tens of thousands of agents in real-time.

7. Discussion

Overall, our results demonstrate that SPH can indeed lift agent-based crowd simulation to extreme densities. Compared to standard SPH, we make particles (i.e. agents) move less erratically by giving them a dynamic rest density and by combining SPH with (weak) traditional contact forces. When using an intelligent collision-avoidance algorithm such as RVO, density-dependent blending lets agents interpolate from collision avoidance to SPH as the density increases. This makes the system suitable for mixed-density scenarios. Finally, SPH is also suitable for simulating shockwaves in extreme-density crowds. The parameters of SPH control the behavior of a shockwave more intuitively than contact forces do.

We have not explicitly compared our SPH-enhanced simulation to a system with a macroscopic fluid simulation model, or to a fully macroscopic crowd simulation. In theory, macroscopic (grid-based) simulations can be faster at high densities, as they perform certain calculations per grid cell instead of per agent. Behaviorally, with the appropriate settings, both paradigms should be capable of simulating the same effects. We therefore believe that it is not useful to discuss differences in behavior between methods, such as in the recent work of Yuan et al. [33]. We do not claim that SPH can simulate certain effects better than other techniques. The main advantage of SPH is that it can easily be *combined* with agent-based simulation, using a one-on-one correspondence between agents and SPH particles.

Regarding the performance measurements in Section 6.4, note that the concert scenario does not include an advanced collision-avoidance algorithm such as RVO. Such an algorithm

would greatly increase the running time of the simulation. However, at extreme densities with ample physical interactions, successful collision avoidance is not feasible, and we argue that an RVO-like algorithm is not worth the computational effort. A challenge for future work is to simulate *mixed-density* scenarios with tens of thousands of agents in real time, by automatically adapting simulation settings to the local density. Density-dependent blending is a first step, but other simulation settings may need to become adaptive too, such as the nearest-neighbor search radius (which affects performance in many ways) and possibly even the timestep Δt_{fine}. After all, $\Delta t_{\text{fine}} = 0.02$ s is specifically required for extreme-density scenarios, whereas a timestep of 0.1 s is more common for lower densities.

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

We have deliberately not made any claims about the *realism* of our simulations. An underlying question is how to compare a crowd simulation to reality. For high-density crowds, one could compare overall statistics such as walking speeds, densities, flow rates, and the relations between these values (i.e. fundamental diagrams). It is generally possible to tune simulation parameters to optimize such statistics [41]. The parameters of SPH are also useful in this regard. However, an additional problem is that real-world footage of high-density crowds is difficult to obtain, let alone to process into usable data. For example, in a video of a concert, it is usually not possible to reliably extract the trajectories of individual people. We therefore require new techniques for analyzing such footage.

Finally, we acknowledge that real-world crowds contain many features that we have not considered, such as social groups and individual variability in behavior. Simulating such features is beyond the scope of this paper. However, SPH has a clear place in the simulation loop, and it only concerns the physical interactions and fluid-like phenomena at high densities. It can therefore trivially be combined with other techniques that operate on other levels.

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 crowds behave similarly to fluids, and where standard collision handling is not sufficient. Previously, such scenarios were almost exclusively 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. To make the standard SPH model more suitable for crowd simulation, we have extended it with a dynamic personal rest density per particle, we have presented a simple new way to handle 2D polygonal obstacles, and we have added a mechanism to interpolate between navigation and fluid-like behavior according to density.

We have demonstrated our model in a number of 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 simulation of extreme-density *and* mixed-density crowds. However, as explained in Section 7, simulating large mixed-density crowds (e.g. with tens of thousands of agents or more) in real time remains technically challenging. To ensure real-time performance for very large crowds, many simulation settings will need to become density-dependent. Our density-based blending mechanism can be useful in this regard: we expect that this will see more applications in future work.

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.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 899739 (CrowdDNA).

References

[1] Hughes, RL. The flow of human crowds. Annu Rev Fluid Mech 2003;35(35):169–182.

[2] Bottinelli, A, Silverberg, JL. Can high-density human collective motion be forecasted by spatiotemporal fluctuations? arXiv:180907875 2018;.

[3] Gingold, RA, Monaghan, JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. Mon Not Roy Astron Soc 1977;181(11):375–389.

[4] van Toll, W, Braga, C, Solenthaler, B, Pettré, J. Extreme-density crowd simulation: combining agents with Smoothed Particle Hydrodynamics. In: Proc. ACM SIGGRAPH Conf. Motion, Interaction and Games. 2020, p. 11:1–11:10.

[5] Pelechano, N, Allbeck, JM, Kapadia, M, Badler, NI. Simulating Heterogeneous Crowds with Interactive Behaviors. CRC Press; 2016.
[6] Thalmann, D, Musse, SR. Crowd Simulation. 2 ed.; Springer; 2013.
[7] van Toll, W, Jaklin, N, Geraerts, R. Towards believable crowds: A generic multi-level framework for agent navigation. In: ASCI.OPEN. 2015,.
[8] Curtis, S, Best, A, Manocha, D. Menge: A modular framework for simulating crowd movement. Collective Dynamics 2016;1(A1):1–40.
[9] Kielar, PM, Biedermann, DH, Borrmann, A. MomenTUMv2: A modular, extensible, and generic agent-based pedestrian behavior simulation framework. Tech. Rep. TUM-I1643; TU München, Institut für Informatik; 2016.
[10] Helbing, D, Molnár, P. Social force model for pedestrian dynamics. Physical Review E 1995;51(5):4282–4286.
[11] Karamouzas, I, Skinner, B, Guy, SJ. Universal power law governing pedestrian interactions. Phys Rev Lett 2014;113:238701:1–5.
[12] van den Berg, JP, Lin, MC, Manocha, D. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In: Proc. IEEE Int. Conf. Robotics and Automation. 2008, p. 1928–1935.
[13] van den Berg, JP, Guy, SJ, Lin, MC, Manocha, D. Reciprocal n-body collision avoidance. In: Proc. Int. Symp. Robotics Research. 2011, p. 3–19.
[14] Guy, SJ, Chhugani, J, Curtis, S, Dubey, P, Lin, M, Manocha, D. PLEdestrians: A least-effort approach to crowd simulation. In: Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation. 2010, p. 119–128.
[15] Dutra, TB, Marques, R, Cavalcante-Neto, JB, Vidal, CA, Pettré, J. Gradient-based steering for vision-based crowd simulation algorithms. Comput Graph Forum 2017;36(2):337–348.
[16] Pelechano, N, Allbeck, JM, Badler, NI. Controlling individual agents in high-density crowd simulation. In: Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation. 2007, p. 99–108.
[17] Helbing, D, Farkas, I, Vicsek, T. Simulating dynamical features of escape panic. Nature 2000;407:487–490.
[18] Kim, S, Guy, SJ, Hillesland, K, Zafar, B, Gutub, AAA, Manocha, D. Velocity-based modeling of physical interactions in dense crowds. The Visual Computer 2015;31:541–555.
[19] Hesham, O, Wainer, G. Context-sensitive personal space for dense crowd simulation. In: Proc. Symp. Simulation for Architecture and Urban Design. 2017,.
[20] Stüvel, SA, Magnenat-Thalmann, N, Thalmann, D, van der Stappen, AF. Torso crowds. IEEE Trans Vis Comput Graphics 2016;23(7):1823–1837.
[21] Best, A, Narang, S, Curtis, S, Manocha, D. DenseSense: Interactive crowd simulation using density-dependent filters. In: Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation. Eurographics Association; 2014, p. 97–102.
[22] van Goethem, A, Jaklin, NS, Cook IV, AF, Geraerts, R. On streams and incentives: A synthesis of individual and collective crowd motion. In: Proc. 28th Int. Conf. Computer Animation and Social Agents. 2015, p. 29–32.
[23] Narain, R, Golas, A, Curtis, S, Lin, MC. Aggregate dynamics for dense crowd simulation. ACM Trans Graph 2009;28:1–8.
[24] Maury, B, Roudneff-Chupin, A, Santambrogio, F. A macroscopic crowd motion model of gradient flow type. Math Mod Meth Appl S 2010;20(10):1787–1821.
[25] Treuille, A, Cooper, S, Popović, Z. Continuum crowds. ACM Trans Graph 2006;25:1160–1168.
[26] Golas, A, Narain, R, Lin, MC. Continuum modeling of crowd turbulence. Physical Review E 2014;90(4):042816.
[27] Lucy, LB. A numerical approach to the testing of the fission hypothesis. The Astronomical Journal 1977;82(12):1013–1024.
[28] Müller, M, Charypar, D, Gross, M. Particle-based fluid simulation for interactive applications. In: Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation. 2003, p. 154–159.
[29] Koschier, D, Bender, J, Solenthaler, B, Teschner, M. Smoothed Particle Hydrodynamics techniques for the physics based simulation of fluids and solids. In: Eurographics 2019 Tutorials. 2019,.
[30] Tantisiriwat, W, Sumleeon, A, Kanongchaiyos, P. 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. 2007, p. 261–268.
[31] Vetter, C, Oetting, L, Ulrich, C, Rung, T. SPH simulations of pedestrian crowds. In: Proc. 6th Int. Spheric Workshop. 2011, p. 261–268.
[32] Weiss, T, Jiang, C, Litteneker, A, Terzopoulos, D. Position-based multi-agent dynamics for real-time crowd simulation. In: Proc. 10th Int. Conf. Motion in Games. 2017, p. 10:1–10:8.
[33] Yuan, Y, Goñi-Ros, B, Bui, HH, Daamen, W, Vu, HL, Hoogendoorn, SP. Macroscopic pedestrian flow simulation using Smoothed Particle Hydrodynamics. Transp Res Part C Emerg Technol 2020;111:334 – 351.
[34] Bender, J, Kugelstadt, T, Weiler, M, Koschier, D. Volume maps: An implicit boundary representation for SPH. In: Motion, Interaction and Games. 2019,.
[35] van Toll, W, Grzeskowiak, F, López, A, Amirian, J, Berton, F, Bruneau, J, et al. Generalized microscopic crowd simulation using costs in velocity space. In: Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games. 2020,.
[36] Sjöström, K. Computational fluid dynamics in 2D game environments. Master's thesis; Umeå University; 2011.
[37] Helbing, D, Johansson, A. Pedestrian, crowd and evacuation dynamics. In: Encyclopedia of Complexity and Systems Science. 2009, p. 6476–6495.
[38] Johansson, A, Helbing, D, Shukla, PK. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. Advances in Complex Systems 2007;10(supp02):271–288.
[39] Garcimartín, Á, Parisi, DR, Pastor, JM, Martín-Gómez, C, Zuriguel, I. Flow of pedestrians through narrow doors with different competitiveness. J Stat Mech Theory Exp 2016;(4):043402.
[40] Parisi, DR, Dorso, SO. Morphological and dynamical aspects of the room evacuation process. Physica A 2007;385:343–355.
[41] Wolinski, D, Guy, SJ, Olivier, AH, Lin, M, Manocha, D, Pettré, J. Parameter estimation and comparative evaluation of crowd simulations. Comput Graph Forum 2014;33(2):303–312.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at `https://doi.org/10.1016/j.cag.2021.06.005`.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Wouter van Toll: Conceptualization, Methodology, Investigation, Formal analysis, Software, Validation, Writing - original draft, Writing - review & editing, Visualization. Thomas Chatagnon: Conceptualization, Methodology, Formal analysis, Investigation, Writing - review & editing. Cédric Braga: Conceptualization, Methodology, Formal analysis, Investigation. Barbara Solenthaler: Conceptualization, Methodology, Writing - review & editing. Julien Pettré: Conceptualization, Methodology, Validation, Writing - review & editing, Supervision, Project administration, Funding acquisition.