Computers & Graphics 78 (2019) 12–22

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on Motion in Games 2017

Position-based real-time simulation of large crowds

Tomer Weiss[a,*], Alan Litteneker[a], Chenfanfu Jiang[b], Demetri Terzopoulos[a]

[a] Computer Science Department, University of California, Los Angeles, USA
[b] Computer and Information Science Department, University of Pennsylvania, USA

ARTICLE INFO

Article history:
Received 27 July 2018
Revised 9 October 2018
Accepted 9 October 2018
Available online 31 October 2018

Keywords:
Crowd simulation
Position-based dynamics
Collision avoidance

ABSTRACT

We introduce a crowd simulation method that runs at interactive rates for on the order of a hundred thousand agents, making it particularly suitable for use in games. Our new method is inspired by Position-Based Dynamics (PBD), a fast physics-based animation technique in widespread use. Individual agents in crowds are abstracted by particles, whose motions are controlled by intuitive position constraints and planning velocities, which can be readily integrated into a standard PBD solver, and agent positions are projected onto constraint manifolds to eliminate colliding configurations. A variety of constraints are presented, enabling complex collective behaviors with robust agent collision avoidance in both sparse and dense crowd scenarios.

1. Introduction

Crowd simulation has become commonplace in visual effects for movies and games. However, the real-time simulation of numerous agents in virtual environments and the simulation of interactions among agents continues to attract the attention of researchers [1,2]. A large number of crowd simulation algorithms have been proposed, mostly focusing on specific requirements, such as defining the scenario to be simulated, modeling the crowd to be simulated, computing the movements of the characters, and rendering the characters. We address the task of computing the movements of characters; i.e., in each time step of the simulation, each character must determine in which direction to move such that realistic individual and collective behaviors result. While approaches such as the social force model [3] and the power law model [4] can yield some realistic crowd behaviors, they often require elaborate numerical treatments to remain stable and robust, especially for dense crowds (Fig. 1).

We show how Position-Based Dynamics (PBD) [5] can be adapted as an alternative algorithm for simulating both dense and sparse crowds, providing a high level of control and stability. Due to its simplicity, robustness, and speed, PBD has recently become popular in physics-based computer animation, particularly in the interactive environments of computer games. In view of the success of PBD in simulating various physical phenomena such as solid and fluid materials in real-time, our work further extends the approach to crowd simulation, ideally for use in games and other interactive applications.

Our objective is a numerical framework for crowd simulation that is robust, stable, and easy to implement. Due to the flexibility of PBD in defining positional constraints among particles, our framework provides a new platform for artistic design and control of agent behaviors in crowd modeling and animation.[1] For example, with positional constraints we prevent agents from colliding and encourage collective crowd behavior. Furthermore, a PBD approach provides an unconditionally stable implicit scheme. Even though it may not always converge to the solution manifold, a nonlinear Gauss–Seidel-like constraint projection enables the algorithm to produce satisfactory results with modest computational cost suitable for real-time applications. Additionally, the resulting numerical scheme is easy to implement and does not require the solution of linear systems of equations.

This paper, which is an extended version of [7], makes the following contributions:

- We show how crowds can be simulated within the PBD framework by augmenting it with non-passive agent-based planning velocities.
- We adopt the position-based frictional contact constraints of granular materials to model local collision avoidance among nearby agents. An XSPH viscosity term is also added to approximate coherent and collective group behavior.

* Corresponding author.
E-mail addresses: tweiss@cs.ucla.edu, tomer.weiss84@gmail.com (T. Weiss).

[1] See [6] for a closely-related PBD approach to layout synthesis.

https://doi.org/10.1016/j.cag.2018.10.008

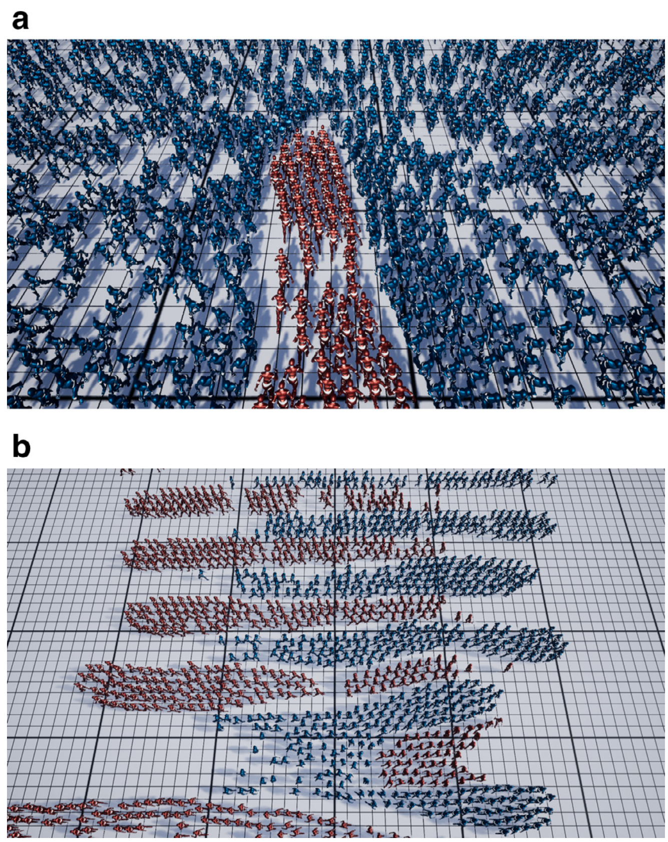


Fig. 1. Our PBD-based crowd simulation method animates dense groups of agents at interactive rates.

- We develop a novel long-range collision avoidance constraint to deal with anticipatory collisions. Our model permits the natural development of agent groups.
- We demonstrate "multi-species" crowd coupling by supporting spatially varying Lagrangian physical properties.

The remainder of the paper is organized as follows: Section 2 surveys relevant prior work on crowd simulation and PBD. Section 3 overviews our algorithmic approach and presents the algorithmic details and detailed constraint design. We present our simulation results in Section 4. Section 5 discusses our method's limitations and future work, and Section 6 presents our conclusions.

2. Related work

2.1. Crowd simulation

Among various considerations, collision avoidance between agents in a moving crowd remains challenging and time consuming. Collision avoidance algorithms can be classified into discrete and continuum approaches [8]. Continuum approaches [9,10] have proven efficient for large-scale dense crowds, but are less suitable for sparse crowds. Force-based discrete approaches, such as the recently proposed power-law model [4], are well suited for sparse crowds, but can be computationally expensive and may require smaller time steps due to explicit time integration. We focus on achieving efficient collision avoidance that is stable and natural in both sparse and dense distributions of agents.

Continuum, macroscopic, approaches—such as 'Continuum Crowds' [9], a crowd model that uses continuum mechanics to simulate pedestrian flow—are particularly suitable for dense, homogeneous crowds and complex environments [11]. Unfortunately, the traditional regime of pure continuum models tends to smooth out local agent behaviors, because the agents are represented by particles carried by force fields. This can lead to unrealistic agent behaviors, with agents accelerating and changing direction without respecting realistic biomechanical limitations. These shortcomings motivated research on hybrid methods. For example, Narain et al. [10] simulated dense crowds with a hybrid, Eulerian–Lagrangian particle-in-cell approach and the unilateral incompressibility constraint (UIC), which has proven to be an effective assumption for crowds. Subsequently, frictional forces were taken into account in modeling crowd turbulence [12,13], which is essential in extra high-density scenarios. This has also inspired us to treat dense agent collisions with a frictional contact model similar to dry sand simulation [14]. Golas et al. [8] proposed a hybrid scheme for simulating both high-density and low-density crowds with seamless transitions.

Many researchers have proposed local force-based models [15–18]. In most of these models, individual agents are simulated, and the crowds naturally form by agent interactions. Typically, the force that determines collision avoidance behavior is a function of inter-agent distance. However, the function is typically scenario-dependent and hardly easy to choose. Recently, Karamouzas et al. [4] proposed a collision avoidance approach whose motivation is the experimental observation that humans do not avoid collisions according to a specific distance threshold, but rather by the estimated time to collision, whose anticipatory behavior energy follows a power law. Similar to concurrent work [19], our method also employs time-to-collision for local agent collision avoidance. However, whereas the method of Karamouzas requires a computationally costly global matrix assembly and solve for each time step, ours is local and independent for each agent, allowing parallelism, which yields a fast, real-time frame rate for up to 100,000 agents.

As an alternative to forces, reciprocal velocity obstacles were proposed for multi-agent navigation [20]. Agents avoid collisions by choosing a velocity that lies outside the velocity obstacles of other agents. Guy et al. [21] demonstrated a parallel velocity obstacles framework for collision avoidance. Ren et al. [22] augment velocity obstacles with velocity connections to keep agents moving together, thus allowing more coherent behaviors. Guy et al. [23] and He et al. [24] simulated crowds based on the least effort principle. Yeh at el. [25] introduced composite agents for complex agent interactions. Bruneau and Pettré [26] presented a mid-term planning system to fill in the gap between long-term planning and short-term collision avoidance. In the work of Kim et al. [27], multi-agent simulation and physical interaction with obstacles were nicely combined to generate interesting new behaviors.

Most of the aforementioned algorithms extrapolate the agents' positions to a future time step, and then deterministically plan agent behavior. However, stochastic approaches are also possible. Burstedde et al. [28] suggested a cellular automata approach to pedestrian dynamics, where the simulation space is discretized into a grid, and pedestrians move from one grid cell to another based on transition probabilities. Kim et al. [29] proposed a statistical inference approach to predict agent paths. WarpDriver [30] models agent interactions as space-time collision probabilities—agents sample intersections between their trajectory and possible upcoming collisions and select the trajectory that has the lowest collision probability.

2.2. Position-Based Dynamics

Position-Based Dynamics (PBD) was proposed by Müller et al. [5] to quickly simulate deformable objects for applications in which simulation speed and robustness take priority over physical realism, such as in computer games. The method works by directly and iteratively computing the positions of objects based on a set of positional constraints. The easily implementable computation involves local Gauss–Seidel projections for each constraint.

Since then, similar position-based simulation methods have been introduced, whose common property is that they do not explicitly compute physical quantities such as momentum and forces, but instead work directly with positions. One such method is Nucleus, developed by Stam [31]. Both PBD and Nucleus have become popular in physics-based animation because of their simplicity and robustness, and they have been implemented in commercial animation frameworks; Autodesk's Maya uses Nucleus as a constraint solver for cloth animation, and NVIDIA PhysX implements a version of PBD. This popularity has inspired work on expanding the simulation capabilities of these methods. Macklin et al. [14] presented a unified PBD solver for various natural phenomena, including fluids and smoke. Barreiro [32] added velocity conformation constraints to allow simulation of viscous fluids. XPBD was proposed recently to eliminate the iteration count and time step dependence of PBD [33].

Even though PBD traditionally defines geometric constraints among particles, it can also approximate force responses from continuum mechanics. Bender et al. [34] formulated continuum energies as PBD constraints. The close relationship between PBD and popular continuum-mechanics-based discretization was further explored in recent work on optimization-based methods for real-time animation [35,36]. Bouaziz et al. [37] introduced projective dynamics, a position-based method that is a more physically principled modification to PBD, and Narain and colleagues [38] showed that projective dynamics is a special case of the Alternating Direction Method of Multipliers (ADMM), which is a general-purpose optimization algorithm. A more complete survey of PBD is provided by Bender et al. [39,40].

3. Algorithm

3.1. Overview

Algorithm 1 outlines our simulation loop (the comments therein indicate sections of the paper that explain the key steps), which is similar to that for PBD, with several modifications. Each agent i, where $i = 1, 2, \ldots, N$, is represented as a fixed-sized particle with position $\boldsymbol{x}_i \in \mathbb{R}^2$ and velocity $\boldsymbol{v}_i \in \mathbb{R}^2$. To represent multiple "species" of agents, we treat each particle as a circle with radius r_i and mass m_i. When stepping from time n to time $n+1$ in a conventional PBD simulation loop for passive physical simulations, a forward Euler position prediction is first computed as $\boldsymbol{x}_i^* = \boldsymbol{x}_i^n + \Delta t(\boldsymbol{v}_i^n + \Delta t \boldsymbol{f}_{\text{ext}}(\boldsymbol{x}_i^n))$, where $\boldsymbol{f}_{\text{ext}}$ represents external forces such as gravity. In position-based crowds, $\boldsymbol{x}_i^*$ must be computed differently, taking into account the velocity planning of each agent. In particular, we compute $\boldsymbol{x}_i^*$ based on a blending scheme between a preferred velocity and the current velocity $\boldsymbol{v}_i^n$. With this alone, particles would passively advect in the velocity field, completely ignoring the existence of other particles. PBD defines constraint functions on the desired locations of the particles. Both equality constraints $C_k(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = 0$ and inequality constraints $C_k(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) \geq 0$ are supported. Hence, the task is to search for a correction $\Delta\boldsymbol{x}_i$ such that $\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^* + \Delta\boldsymbol{x}_i$ satisfies the constraints. The correction is multiplied by a stiffness $k \in [0, 1]$ associated with the constraint type, which provides flexibility in controlling the magnitude of the constraint correction. After the new positions are computed, the agent velocities can be updated as $\boldsymbol{v}_i^{n+1} = (\boldsymbol{x}_i^{n+1} - \boldsymbol{x}_i^n)/\Delta t$. This update guarantees stable agent velocities as long as the constraint projection is stable.

Our position-based formulation includes several modifications to the standard PBD scheme as well as additional constraints for short-range and long-range collision avoidance between agents, as described in the following sections.

Algorithm 1 Position-based crowd simulation loop.

for all agent i **do** ▷ §3.2
  calculate $\boldsymbol{v}_i^p$ from the velocity planner
  calculate a blending velocity $\boldsymbol{v}_i^b$ from $\boldsymbol{v}_i^p$ and $\boldsymbol{v}_i^n$
  $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^n + \Delta t \boldsymbol{v}_i^b$
end for
for all agent i **do**
  find neighboring agents $S_i = \{s_{i_1}, s_{i_2}, \ldots, s_{i_m}\}$
end for
while iteration count < max stability iterations **do**
  for all agent i **do**
    compute position correction $\Delta\boldsymbol{x}_i$ ▷ §3.3
    $\boldsymbol{x}_i^n \leftarrow \boldsymbol{x}_i^n + \Delta\boldsymbol{x}_i$
    $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^* + \Delta\boldsymbol{x}_i$
  end for
end while
while iteration count < max iterations **do**
  for all agent i **do**
    compute position correction $\Delta\boldsymbol{x}_i$ ▷ §3.3, §3.4, §3.5
    $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^* + \Delta\boldsymbol{x}_i$
  end for
end while
for all agent i **do**
  $\boldsymbol{v}_i^{n+1} \leftarrow (\boldsymbol{x}_i^* - \boldsymbol{x}_i^n)/\Delta t$
  Add XSPH viscosity to $\boldsymbol{v}_i^{n+1}$ ▷ §3.8
  Clamp $\boldsymbol{v}_i^{n+1}$ ▷ §4.7
  $\boldsymbol{x}_i^{n+1} \leftarrow \boldsymbol{x}_i^*$
end for

3.2. Velocity blending

Agent level roadmap velocity planning describes high-level agent behaviors. Local behavior may be influenced by factors such as social or cognitive goals, while global behavior may be specified by a particular walking path. Roadmap planning is an orthogonal component to our constraint-based approach.

In the physics-based simulation of solids and fluids, particles generally retain their existing velocities. In particular, as demonstrated in [37], the implicit Euler time integration of a physical system can be formulated as an minimization problem that balances the 'momentum potential' $\|\boldsymbol{M}^{1/2}(\boldsymbol{x} - (\boldsymbol{x}^n + \Delta t \boldsymbol{v}^n))\|_F^2 / 2\Delta t^2$ and other potential energies, where $\boldsymbol{M}$ is the mass matrix. In multi-agent crowd simulation, it is similarly more desirable to include the inertial effect before predicting an agent's desired velocity. Denoting the preferred velocity given the planner as $\boldsymbol{v}_i^p$, we calculate the agent velocity $\boldsymbol{v}_i^b$ as a linear blend between $\boldsymbol{v}_i^p$ and the current velocity $\boldsymbol{v}_i^n$, as follows:

$$\boldsymbol{v}_i^b = (1 - \alpha)\boldsymbol{v}_i^n + \alpha \boldsymbol{v}_i^p, \tag{1}$$

where $\alpha \in [0, 1]$ is the velocity blending parameter. We set $\alpha = 0.0385$ in all our simulations. A more adaptive choice, such as the density-based blending factor in [10], can also be used in our framework.

3.3. Short-range interaction

As in standard position-based methods, we model short-range, local particle contacts using an inequality distance constraint:

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| - (r_i + r_j) \geq 0, \tag{2}$$

where r_i and r_j are the radii of agents i and j. Since each constraint in a PBD-based framework has an associated stiffness, we resolve the constraint with a stiffness of 1.0 in order to prevent the agent disks from unrealistically overlapping in the next time step.

3.4. Long-range collision

Karamouzas et al. [4] describe an explicit force-based scheme for modeling crowds. We design a similar scheme in the form of a position-based constraint. As in their power law setting, the leading term is the time to collision τ, defined as the future time when two disks representing particles i and j will touch each other. As in [4], it can be shown that

$$\tau = \frac{b - \sqrt{b^2 - ac}}{a}, \tag{3}$$

where

$$a = \frac{1}{\Delta t^2} \|\boldsymbol{x}_i^* - \boldsymbol{x}_j^*\|^2, \tag{4}$$

$$b = -\frac{1}{\Delta t} (\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_i^* - \boldsymbol{x}_j^*), \tag{5}$$

$$c = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 - (r_i + r_j)^2. \tag{6}$$

No potential energies associated with forces are required in our framework. To facilitate collision-free future states, we directly apply a collision-free constraint on future positions. Recall that in our simulation loop, the predicted position of particles i and j in the next time step are

$$\boldsymbol{x}_{i,j}^* = \boldsymbol{x}_{i,j}^n + \Delta t \boldsymbol{v}_{i,j}^b, \tag{7}$$

where $\boldsymbol{v}_{i,j}^b$ is defined in (1), and the subscripts indicate that the above equation is defined exclusively in the context of particles i and j.

We estimate a future collision state between i and j using τ. We first compute the exact time to collision using (3). Valid cases are those with $0 < \tau < \tau_0$, where τ_0 is a fixed constant, set to $\tau_0 = 20$ in our experiments unless noted otherwise. After pruning out the invalid cases, we process the remaining colliding pairs in parallel (Section 4.1). We define $\hat{\tau} = \Delta t * \lfloor \tau / \Delta t \rfloor$, where $\lfloor \cdot \rfloor$ denotes the floor operator. This simply clamps τ to provide a discrete time slightly before the predicted contact. With $\hat{\tau}$, we have

$$\hat{\boldsymbol{x}}_{i,j} = \boldsymbol{x}_{i,j}^n + \hat{\tau} \boldsymbol{v}_{i,j}^b. \tag{8}$$

Note that $\hat{\boldsymbol{x}}_{i,j}$ are similar to $\boldsymbol{x}_{i,j}^n$ in the traditional collision constraint case (2) and are still in a collision free state. Stepping forward will cause the actual penetration. We denote the colliding positions with

$$\tilde{\boldsymbol{x}_{i,j}} = \boldsymbol{x}_{i,j}^n + \tilde{\tau} \boldsymbol{v}_{i,j}^b, \tag{9}$$

where $\tilde{\tau} = \Delta t + \hat{\tau}$. We enforce a collision free constraint on $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. Note that $\tilde{\boldsymbol{x}}_{i,j}$ is a function of $\boldsymbol{x}_{i,j}^*$; therefore, it is still essentially a constraint on $\boldsymbol{x}_{i,j}^*$. Due to its anticipatory nature, high stiffness on this constraint is not necessary. Since the particles represent agents, we want to prevent unrealistic agent locomotion that might result from a positional correction. To that end, instead of a full overly-stiff correction for the impending collision, we multiply the correction by a stiffness of $k \exp(-\hat{\tau}^2/\tau_0)$, where k is a user-specified constant.

Other than [4], previous work such as OpenSteer [16] proposed a similar long-range collision avoidance scheme. However, our approach differs in several aspects. First, OpenSteer's avoidance behavior is distance dependent, and is activated starting from a certain pairwise distance. Second, for resolving a potential collision site, OpenSteer fully adjusts the agent's velocity, while our method takes into account time-to-collision and PBD stiffness as inputs to the magnitude of the agent's correction. Time-to-collision was shown to be a better predictor for collision avoidance behavior than pairwise distance (see Section 2.1).

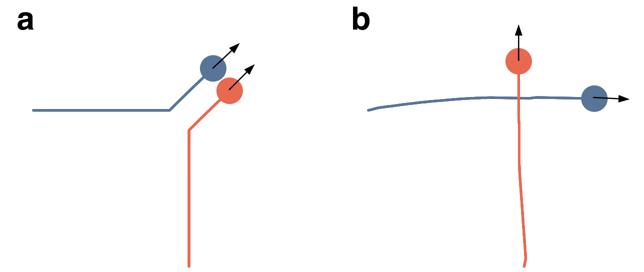


Fig. 2. Predictive collision avoidance model. (a) Starting with particles at $\boldsymbol{x}_i^n$ and $\boldsymbol{x}_j^n$, PBD estimates their positions $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_j^*$ at the next time step. We estimate a discrete time to collision $\hat{\tau}$ using their trajectories. This results in $\hat{\boldsymbol{x}}_{i,j} = \boldsymbol{x}_{i,j}^n + \hat{\tau} \boldsymbol{v}_{i,j}$. When further advanced in time by Δt, particles collide at $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. (b) Projecting these collision constraints resolves the collision between $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. (c) We compute the relative displacement $\boldsymbol{d}$ from time $\hat{\tau}$ to $\tilde{\tau}$. (d) $\boldsymbol{d}$ is decomposed into contact normal $\boldsymbol{d}_n$ and tangential $\boldsymbol{d}_t$ components. (e) The tangential contribution of the relative displacement is distributed to $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_j^*$, thus avoiding the collision.

3.5. Collision avoidance model

A traditional collision response is not always satisfactory and realistic (Fig. 3a), since agents do not make any long-range attempt to avoid the upcoming collision. Furthermore, the long-range collision constraint proposed in Section 3.4 can slow down the agents. Fig. 2 illustrates this process—in a typical long-range response, agents correct their motions according to the contact normal $\boldsymbol{d}$, a vector that has a component $\boldsymbol{d}_n$ in the direction opposite to the agent's trajectory. This behavior is often undesirable in dense scenarios like those shown in Fig. 1.

To amend such undesirable behavior, we present a novel collision avoidance model. We observed that the tangential component of long-range collision response is often desired, effectively causing the agents to divert sideways in response to the predicted collision. Hence, our avoidance model preserves only the tangential movement in such collisions. The total relative displacement is calculated as

$$\boldsymbol{d} = (\tilde{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i) - (\tilde{\boldsymbol{x}}_j - \hat{\boldsymbol{x}}_j), \tag{10}$$

which may be decomposed into contact normal and tangential components, as follows:

$$\boldsymbol{d}_n = (\boldsymbol{d} \cdot \boldsymbol{n})\boldsymbol{n}, \tag{11}$$

$$\boldsymbol{d}_t = \boldsymbol{d} - \boldsymbol{d}_n, \tag{12}$$

where $\boldsymbol{n} = (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)/\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|$ is the contact normal. We preserve only the tangential component in the positional correction to $\boldsymbol{x}_{i,j}^*$. This provides an avoidance behavior (Fig. 3b) and prevents agents from being pushed back in a dense flow.

3.6. Frictional contact

Researchers have proposed that for medium and high densities the motion of crowds can be approximately modeled using techniques inspired from fluid dynamics and granular flows [41–43], and these granular and fluid analogies have inspired crowd simulation algorithms [12,13]. While our method also builds on these ideas, experiments conducted by Seyfried et al. [44] demonstrating a relationship between the velocity and the density of agents in a crowd motivated us to offer a degree of control on how much agents slow down in high-density simulations. The short-range constraint (Section 3.3) does not allow such control. Even though

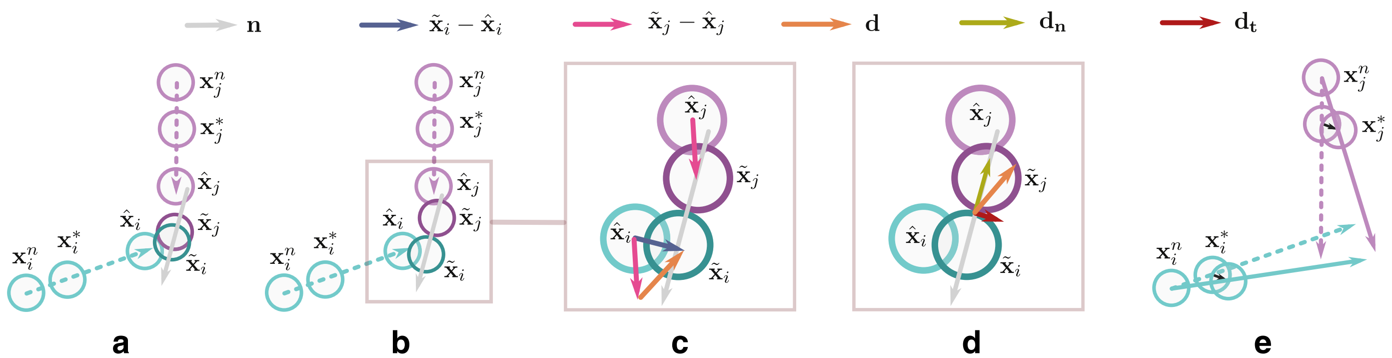


Fig. 3. Orthogonal crossing. (a) With the traditional collision constraint (2), agents collide and are unable to pass. (b) Agents that employ our avoidance model successfully cross paths without collision.

the constraint resolves possible collisions between agents, it does not slow down agents by limiting their tangential displacement.

Consequently, following the PBD constraint for granular material proposed by Macklin et al. [14], we incorporate a contact friction constraint between pairs of neighboring agents, as follows: During a time step, if agents are predicted to overlap, we resolve the potential collision by projecting particles a distance d given by the conventional collision constraint (2). Then, we calculate the total relative displacement (10) in the time step, and decompose it into the contact normal (11) and tangential (12) components. Assuming that the agents have similar mass, we add $0.5\boldsymbol{d}_t \min(1, \mu\Delta x\|\boldsymbol{d}\|)$, where $\mu \in [0, 1]$, to the conventional correction for each particle. This correction limits the tangential movement, contrary to our approach in the avoidance model (Section 3.5).

3.7. Maximum speed limiting

After the constraint solve, we further clamp the maximum speed of the agents to better approximate real human capabilities. In our implementation (Algorithm 1), we limit the magnitude of the agent's velocity to a maximum value. Alternatively, we achieved similar results by clamping the maximum acceleration correction allowed per the constraint type, instead of just clamping the entire movement in the time step. This may be desirable when we wish to preserve full positional correction for certain constraint types, such as the short-range contact, but limit the allowed positional correction of other types, such as the long-range collision avoidance. This form of clamping is roughly equivalent to modifying a constraint's stiffness and τ_0.

3.8. Cohesion

To encourage more coherent agent motions, we add artificial XSPH viscosity [45,46] to the updated agent velocities. Specifically,

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + c\sum_j (\boldsymbol{v}_i - \boldsymbol{v}_j) W(\boldsymbol{x}_i - \boldsymbol{x}_j, h), \tag{13}$$

where $W(\boldsymbol{r}, h)$ is the Poly6 kernel for SPH [46]. In our simulation, for agents represented by discs with radius 1, we use $h = 7$ and $c = 217$.

3.9. Walls and obstacles

Agents can interact with walls and other static obstacles in the environment. To prevent agents from locomoting into walls and other static obstacles, we employ a traditional collision response (2) between the agent's predicted position and the nearest point on the obstacle. The obstacle's collision point is assigned infinite mass, so that any positional correction applies solely to the agent.

4. Experiments and results

4.1. Setup and parameter settings

We implemented our framework in CUDA on an Nvidia GeForce GT 750M GPU. We set $\Delta t = 1/48$ s for all the experiments (2 substeps per frame). We solve each constraint group in parallel, employing a Jacobi solver, with a delta averaging coefficient of 1.2. To find neighboring agents, we use two hash-grids with different cell sizes for short and long-range collisions. This is more efficient than using one grid for both, since the long-range grid covers a bigger collision radius. Each grid is constructed efficiently and in parallel. See [14,47] for additional details.

In our simulations, we use 1 stability iteration to resolve contact constraints possibly remaining from the previous time step, and 6 iterations in the constraint solve loop. Additional iterations can increase stability and smoothness albeit at increased computational cost.

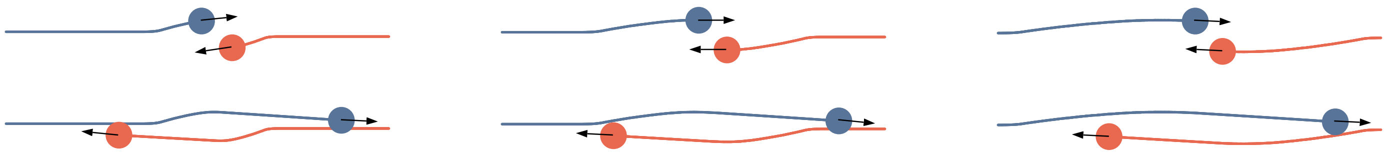
For agent rendering and online locomotion synthesis, we used Unreal Engine 4.15. Clamping the agent's skeletal positional acceleration and rotational velocity allows smoother locomotion. Additionally, we applied a uniform motion scaling of about 30. The motion rendered is at approximately 5 times the simulation rate.

We demonstrated the robustness of our position-based framework in a variety of scenarios. To simplify the experiment setup and unless otherwise stated, we modeled all agents using a disk with radius 1, and use the same width for our humanoid agents in the rendering stage. For smoother motion, we allow an expansion of the agent's disk radius by 5% during collision checks. Unless otherwise stated, we chose $\tau_0 = 20$ for all our experiments. The larger the value of τ_0, the earlier the agents adjust their trajectories to avoid each other (Fig. 4). For each benchmark, we used a simple preferred velocity planner, where the preferred velocity of each agent points to the closest user-scripted goal. We also slightly varied the agent's preferred velocity around a mean of 1.4, to achieve more realistic simulation that corresponds to field studies of pedestrian walking speed [48]. Table 1 presents timing information. Table 2 provides a complete list of parameter values used in the experiments.

4.2. Benchmarks and analysis

4.2.1. Sparse passing (low count, long-range collision)

We experimented with two groups of agents locomoting towards each other (Fig. 6). The agents in each group are positioned in a loose grid formation with an initial separation distance. To avoid collisions, the agents use the constraint of Section 3.4. In this scenario, the agents organize into narrow lanes and pass each other easily.



(a) $\tau_0 = 6$ (b) $\tau_0 = 12$ (c) $\tau_0 = 20$

Fig. 4. Agents locomoting past each other, showing the effect of the time-to-collision τ_0 on their motions. The larger the value of τ_0, the earlier the agents begin maneuvering according to the collision avoidance model (Section 3.5).

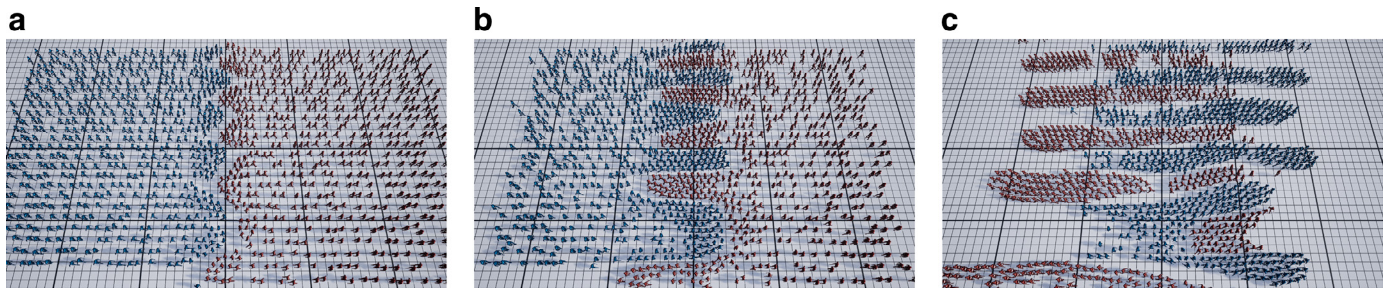


Fig. 5. Groups of agents passing each other using the avoidance model. (a) Agents organize into boundary fronts in preparation for collision avoidance. (b) Agents huddle in noticeable alternating thick lanes. (c) Agents successfully pass each other.

Table 1
Timings (excluding rendering times). All experiments use $\Delta t = 1/48$, with 6 iterations per time step. LR: long-range collision constraint; A: avoidance model constraint.

	# Agents	LR	A	ms/frame
Sparse passing	1,600	On	–	11.27
Sparse passing	1,600	–	On	11.61
Dense, low count	1,600	On	–	12.03
Dense, low count	1,600	–	On	11.34
Dense, high count	10,032	On	–	14.06
Dense, high count	10,032	–	On	13.63
Bears and rabbits	1,152	–	On	11.86
Dense ellipsoid	1,920	–	On	10.06
Circle	250	On	–	11.09
Circle	1,000	On	–	12.35
Two crossing groups	210	On	–	9.25
Four crossing flows	800	On	–	8.13
Proximal behavior	50	On	–	10.12
Proximal behavior	50	–	On	10.13
Target locomotion	192	On	–	10.42
Bottleneck	480	–	–	11.99
Bottleneck	3,600	–	–	17.76
Bottleneck	100,048	–	–	43.66

4.2.2. Sparse passing (low count, avoidance)

This scenario is identical to Experiment 4.2.1, but the agents employ the constraint of Section 3.5 to avoid collisions. In this scenario, the agents form thicker lanes (Fig. 5), which separate into subgroups.

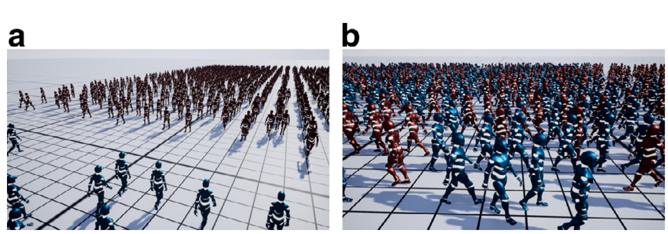


Fig. 6. Two groups of agents exchanging positions. (a) The groups approaching each other. (b) Collisions are avoided using the long-range collision avoidance constraint.

4.2.3. Dense passing (low count, long-range collision)

A total of 1600 agents are split into two groups, with a separating distance of 2.5 (Fig. 7a). We used a higher and denser crowd of agents. To avoid collision, the agents employ the constraint of Section 3.5. Because of the dense agent setting, the two agent groups do not easily pass each other, and some bottleneck

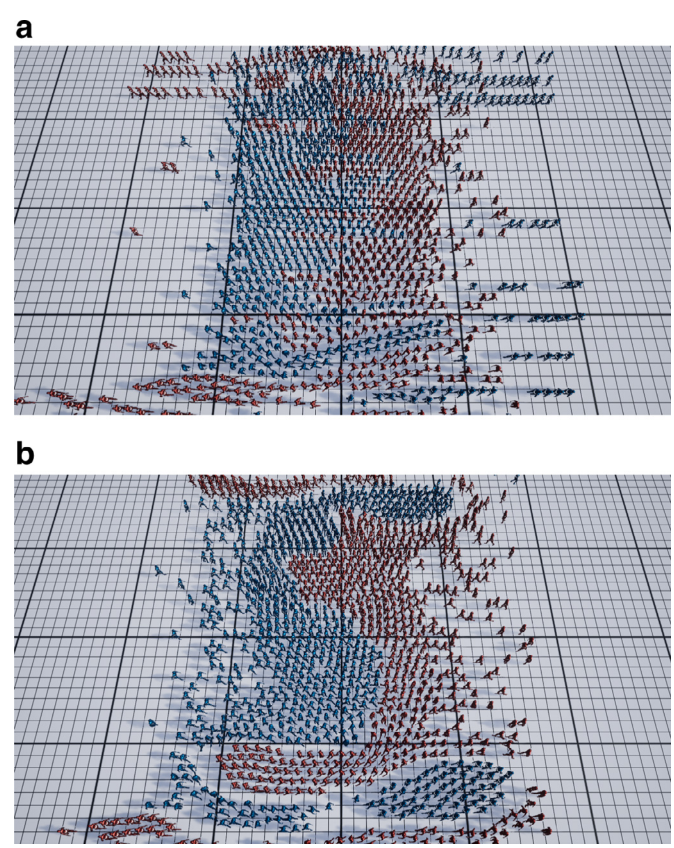


Fig. 7. High-density agent simulation. (a) Long-range collision. (b) Avoidance model.

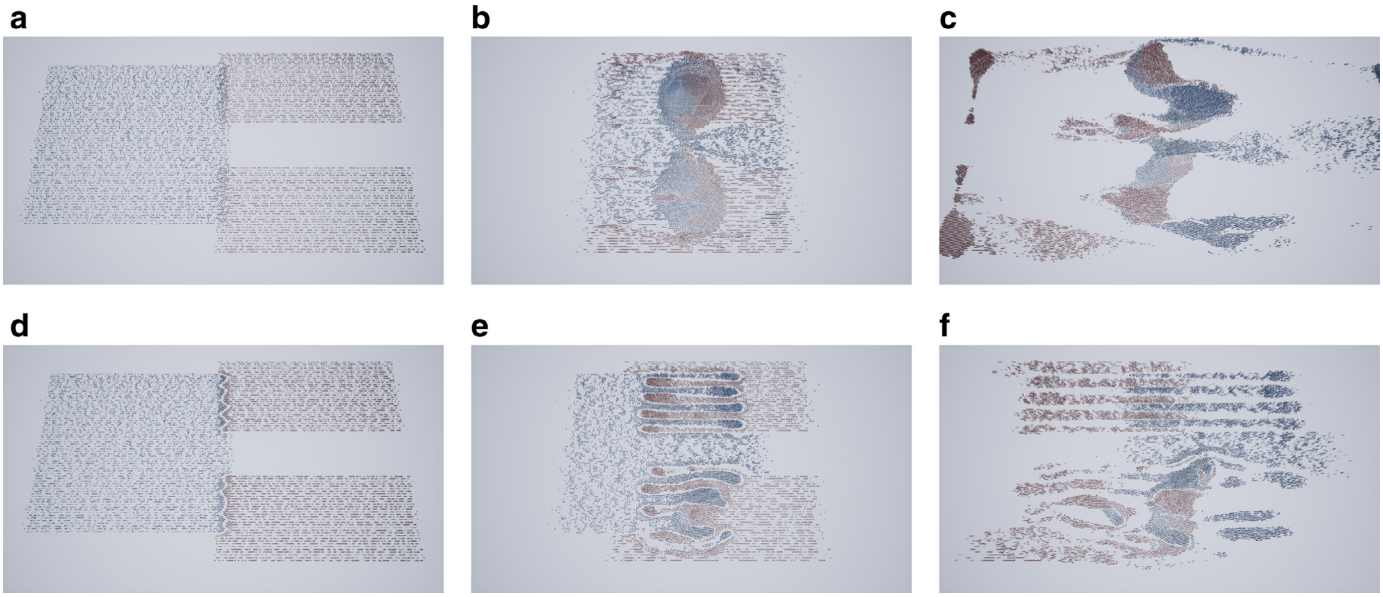


Fig. 8. High density and high agent count. (a)–(c) Groups of agents avoid each other using long-range collision. (d)–(f) Using the avoidance model.

groups are formed. Eventually, the agents pass, avoiding unrealistic collisions.

4.2.4. Dense passing (low count, avoidance)

This experimental setup is identical to Experiment 4.2.3. To avoid collision, the agents employ the constraint of Section 3.5. In this scenario, the agents form thicker lanes, which separate into subgroups (Fig. 7b).

4.2.5. Dense passing (high count, long-range collision)

A total of 10,032 agents are split into two groups (Fig. 8a–c) with a separating distance of 3.5.

4.2.6. Dense passing (high count, avoidance)

This experiment setup is identical to Experiment 4.2.5. To avoid collision, the agents employ the constraint of Section 3.5. In this scenario, the agents form thicker lanes, which separate into subgroups (Fig. 8d–f).

4.2.7. Bears and rabbits

In this experiment, we showcased how a Lagrangian PBD scheme may be employed to model agents of different sizes (Fig. 9). We simulated a group of rabbits passing through a group of bears, totaling 1152 agents. The rabbits had size 1.0, while the bears had a size ranging from 2.5 to 4.0. Since bears are less inclined than rabbits to change their paths, we assigned the bears a mass that is approximately 30 times greater than that of the rabbits.

4.2.8. Dense ellipsoid

This simulation comprises 1920 agents. To reach their goals, an ellipsoid-shaped group of agents (Fig. 10) with an initial separation distance of 3.3 must locomote through a larger, rectangular group of agents with a separation distance of 3.0. Throughout the simulation, the small group retains its shape and it successfully passes the larger group.

4.2.9. Circle

In this scenario, agents are arranged in circle formation (Fig. 13). Each agent has the goal of locomoting to the antipodal position on the circle. The paths of all agents to their destination passes through the center of the circle, which allows us to observe how they avoid collision despite an increasing number of possible neighbor contacts, all with varied paths. We tested this scenario with the long-range collision constraint and observed a minor increase in the computational cost between 250 and 1000 agents. In both cases, agents converged and formed one group in the center. After a few seconds, the group rotated, with many of the agents breaking free of the group and locomoting to their goals (Fig. 11).

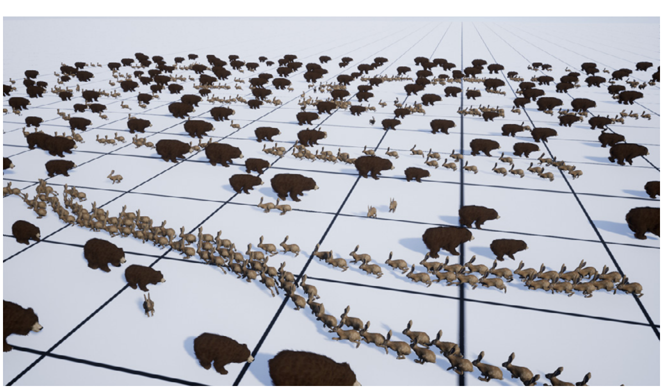


Fig. 9. A group of smaller agents (rabbits) passing through a group of larger ones (bears).

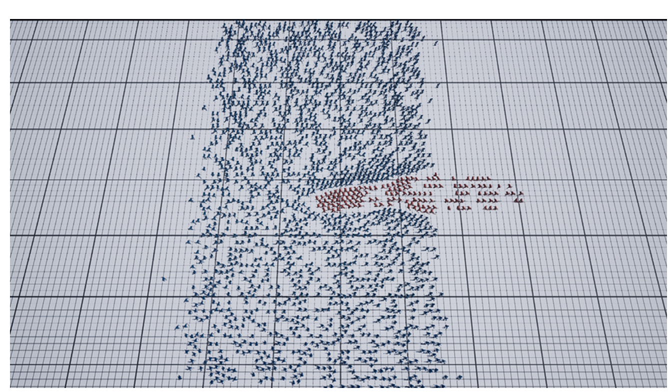


Fig. 10. A small ellipsoid shaped group passing through a larger group.

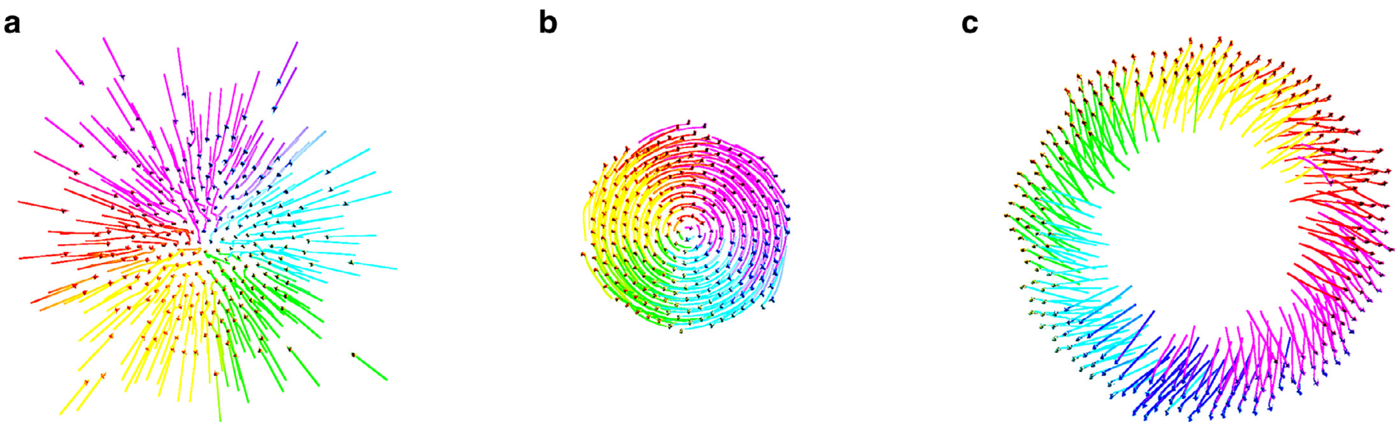


Fig. 11. Circle of 250 agents. (a) Agents locomote towards their antipodal position goals on the circle. Multiple potential collisions between agents are avoided. (b) Agents converge on the center of the circle. All agents have slowed down; however, the entire group eventually rotates, thus avoiding a stalemate. (c) Agents successfully reach their goals.

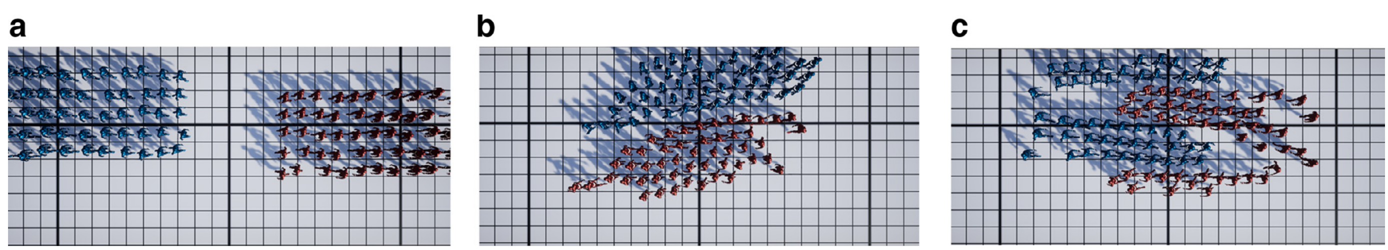


Fig. 12. Proxemic group behavior. (a) Initial state. (b) Agents avoid each other using the long-range collision model, while creating lanes. (c) Agents avoid each other using the avoidance model.

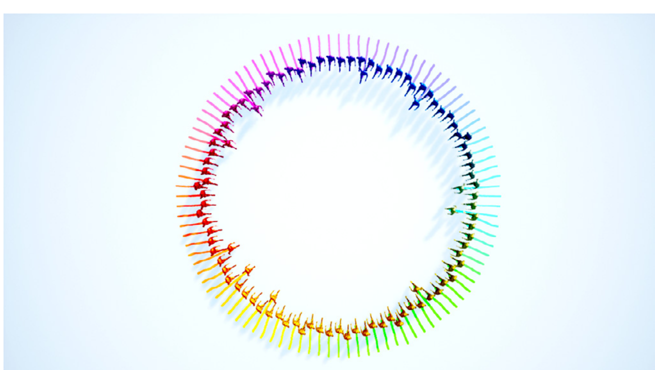


Fig. 13. Starting conditions of the circle experiment. Agents have different preferred velocities, around a mean of 1.4 (Table 2).

Table 2
Default parameter values used in the experiments.

Parameter	Value	Refer to
# Solver Iterations	6	Section 4.1
# Stability Iterations	1	Section 4.1
Delta averaging coefficient	1.2	Section 4.1
Time step	1/48	Section 4.1
Time-to-collision τ_0	20	Section 3.4
Short-range stiffness	1.0	Section 3.3
Long-range stiffness	0.24	Section 3.4
Avoidance model stiffness	0.24	Section 3.5
Nearest neighbor radius expansion	5%	Section 4.1
Maximum acceleration	5.1	Section 4.1
Velocity blending parameter	0.0385	Section 3.2
Preferred velocity	1.4	Section 4.1
Friction parameter μ	0.21	Section 3.6
XSPH c_{poly6}	7	Section 3.8
XSPH h_{poly6}	217	Section 3.8

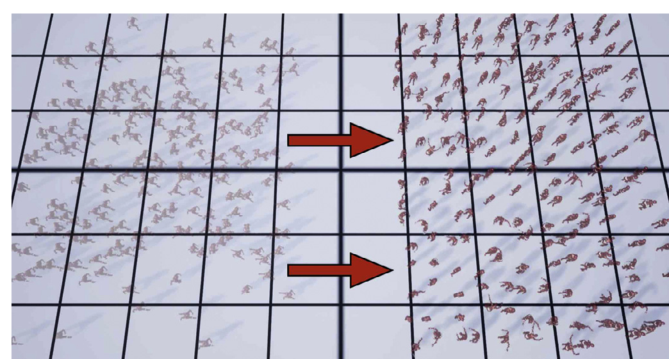


Fig. 14. Target locomotion. The goal is based on the initial position, which is translated to the right, and then randomly exchanged with another agent. The result is multiple intersecting paths. Using the long-range collision avoidance constraint, agents reach their respective goals with no noticeable slowdown.

4.2.10. Proximal behavior, avoidance model

Two groups of 50 agents start in tightly packed formations, and must pass each other in a narrow corridor with limited collision avoidance space (Fig. 12b). This benchmark demonstrates that our novel avoidance model creates proxemic behavior in agent groups [24].

4.2.11. Proximal behavior, long-range collision

Here, we used the same initial conditions as in Experiment 4.2.10. We observed lane formation and splitting of the original group (Fig. 12a).

4.2.12. Target locomotion, long-range collision

192 agents start in a uniform random grid setting at a separation distance of 5.5 (Fig. 14). The locomotion targets are in a similar but translated grid pattern, randomly perturbed with additive

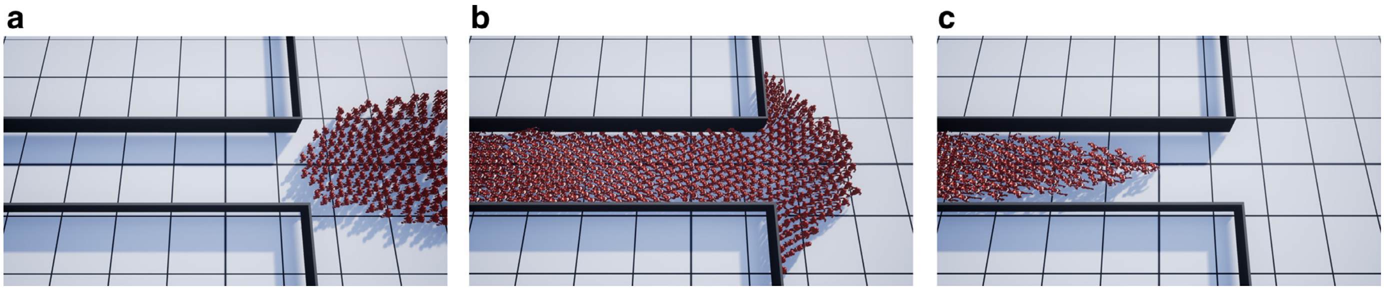


Fig. 15. A group of agents passing through a narrow corridor. (a) Agents huddle on approaching corridor's entrance. (b) A semi-circular arch forms as agents enter a narrow corridor. (c) Agents successfully exit.

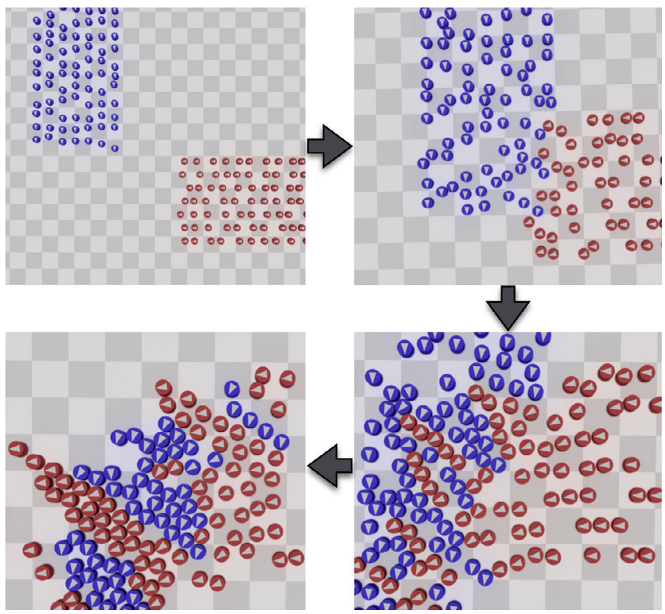


Fig. 16. Two orthogonally crossing groups. Top left: Initial configuration, with the red agents locomoting leftward and the blue agents locomoting downward. Top right: Agents approach each other, and start to divert slightly in response. Bottom right: The groups meet, and the agents of each group start forming alternating lanes. Bottom left: Self-organized into diagonal lanes, agents cross each other, mitigating congestion. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

uniformly distributed random noise. Agents are able to reach their respective goal with minimal interference.

4.2.13. Bottleneck

We demonstrated our method on a bottleneck scenario with varying number of agents (Fig. 15). Agents must pass through a narrow corridor to reach their goal. In this scenario, we observed jamming and arching near the corridor's entrance, as well as the formation of pockets, a phenomena observed in realistic crowds, which was also reported in [13,23].

4.2.14. Two orthogonally crossing groups

Two groups of 110 agents orthogonally cross each other (Fig. 16). The leading agents in both groups slow down and deviate from their course, attempting to avoid the oncoming collisions. Subsequently, agents form clusters that approximate diagonal lanes, successfully crossing each other, a phenomenon that is also observed in real crowds [49].

4.2.15. Four diagonally crossing flows

In a scenario of four crossing agent flows (Fig. 17) with the locomotion goal of each flow at the opposite corner, the main difficulty is that the agents can potentially become stuck in congestion at the intersection of the flows. Additionally, we want agents to avoid excessive deviation from their locomotion goals. Using the long-range collision avoidance constraint, agents mange to achieve their locomotion goals and avoid major congestion. We also observed vortex-like behavior once the flows meet, which is a form of lane formation, similar to [9].

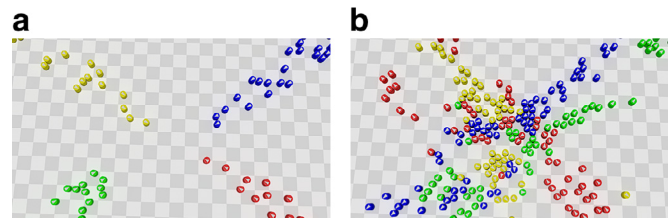


Fig. 17. Four diagonally crossing agent flows. (a) As the flows approach at the center, agents start diverting to mitigate a temporary congestion that develops. (b) Agents continue to make progress, albeit at a slow pace.

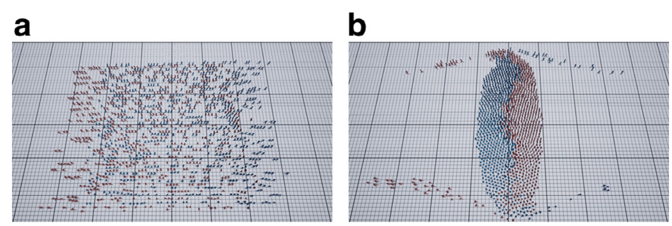


Fig. 18. Explicit force-based power law [4]. (a) In a sparse setting, most agents successfully avoid collisions (at most 3 agent pairs colliding). (b) In a dense setting, the agents collide, overlap, and are unable to pass smoothly.

4.3. Comparison

The method of Karamouzas et al. [4] is considered a state-of-the-art model for explicit force-based modeling of pedestrian behavior, and it has been validated against human behavior. We implemented this method based on code obtained from the authors. For our comparison, we chose the same parameter settings and time step as in our method (Section 4.1). Using 1344 agents, we performed experiments in the following two settings (Fig. 18):

4.3.1. Crowd passing (sparse)

For the sparse setting (Fig. 18a), we used a separating distance of approximately 4.5 between agents. Agents preformed well and avoided collisions, managing to pass with minimal interference to the opposing group. Lane patterns emerged.

4.3.2. Crowd passing (dense)

In the dense setting (Fig. 18b), we used a separation distance of approximately 3.3. For major parts of the simulation, agents were

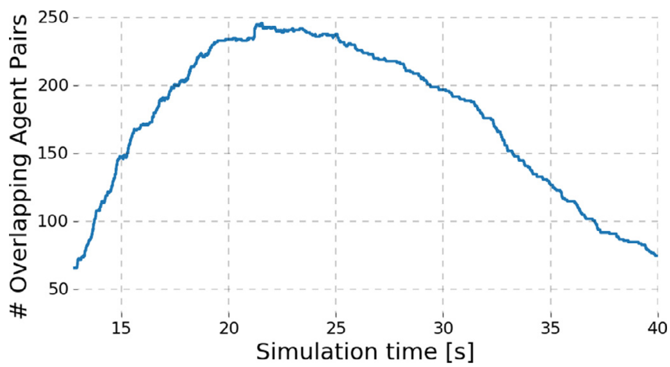


Fig. 19. Number of overlapping agent pairs in the crowd passing (dense) experiment, with agent collision avoidance using the method in [4]. The two groups meet around second 5. Note that out of 1344 agents, there are a significant number (up to 246) of colliding agent pairs by second 15 and later.

not able to maintain their trajectories or avoid collisions with the opposing group. Some of these collisions were not resolved, leading to unrealistic behavior for most of the simulation (Fig. 19).

Both the above experiments ran interactively, averaging 12.12 ms/frame and 13.74 ms/frame for the sparse and dense scenarios, respectively. Increasing the number of agents or the density of agents resulted in slower run-times. From these experiments, we noticed that the power law method does not provide a collision-free model for dense crowds. Nevertheless, careful parameter tuning or smaller time steps may help, albeit at the expense of efficiency hence usability.

5. Limitations and future work

Our approach has several limitations. First and foremost, modeling agents as simple particles hardly aspires to simulate real pedestrians, unlike several other notable efforts on multi-human simulation [50–52].

Even though ours is a simple and stable crowd simulation framework, it requires a certain amount of parameter tuning to maximize realism (see Table 2). Designing metrics to evaluate the realism of crowd simulations is a problem in and of itself, and it is outside the scope of our present work. Investigating this topic in future work would call for a further quantitative analysis of time-to-collision and other anticipatory metrics.

Currently, our method employs a simple navigational scheme for planning each agent's velocity in the next time step. The planner directs agents to their locomotion goal without considering static objects or other agents. Furthermore, the current navigational scheme does not consider directing agents to other, longer paths towards their goals that might nevertheless be shorter in time. Replacing this component with a dynamic path planning scheme should lead to more realistic simulation results.

Since PBD is a deterministic framework, there is in principle a chance agents will not be able to avoid precisely head-on collisions; however, we did not observe any such cases in our experiments. Such collisions may be averted by adding a small stochastic component to the agent's trajectory [53].

Our position-based approach allows simple integration into an existing PBD framework. By adding new constraints, our robust, parallel framework can easily incorporate more complex crowd behaviors with minimal run-time cost. We also plan to explore other constraints, such as clamping the magnitude of turning and backward motion of agents, which we believe will yield more realistic crowd motion. Finally, experimenting with different online locomotion synthesis methods can lead to more interesting agent interactions.

6. Conclusion

We have presented a discrete algorithm for simulating crowds that is inspired by Position-Based Dynamics. First, we showed how to adapt preexisting PBD constraints to control the motion and short-range collision avoidance behavior of each agent. Second, we proposed a novel long-range collision avoidance constraint within the PBD framework. Third, we demonstrated interactive frame rates for up to 100,000 agents simulated in CUDA on an Nvidia GeForce GT 750M GPU. Finally, our method easily integrates with readily available visual effects and gaming tools that employ PBD.

Our solution is flexible and produces interesting patterns and emergent crowd behavior with no user intervention, such as groups of agents passing each other seamlessly, as well as the spontaneous formation of traffic lanes and subgroups of agents. We demonstrated our crowd simulation algorithm on groups of agents of various sizes, arranged in varying densities, using different mixtures of PBD constraints.

By varying constraint parameters, an animator using our approach can easily control the motions of virtual crowds. Such qualities are available generally with larger time step choices than with existing explicit time integration schemes for crowd simulation. Thus, our method offers interactive, collision-free crowd simulation with guaranteed stability and easy controllability.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cag.2018.10.008.

References

[1] Thalmann D. Crowd simulation. Wiley Online Library; 2007.
[2] Pelechano N, Allbeck J, Badler N. Virtual crowds: methods, simulation, and control. Synthesis lectures on computer graphics and animation. Morgan & Claypool Publishers; 2008.
[3] Helbing D, Molnar P. Social force model for pedestrian dynamics. Phys Rev E 1995;51(5):4282.
[4] Karamouzas I, Skinner B, Guy S. Universal power law governing pedestrian interactions. Phys Rev Lett 2014;113:238701.
[5] Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. J Vis Commun Image Represent 2007;18(2):109–18.
[6] Weiss T, Litteneker A, Duncan N, Nakada M, Jiang C, Yu L-F, et al. Fast and scalable position-based layout synthesis. IEEE Trans Vis Comput Graph 2018:1–13. doi:10.1109/TVCG.2018.2866436. (Early Access)
[7] Weiss T, Jiang C, Litteneker A, Terzopoulos D. Position-based multi-agent dynamics for real-time crowd simulation. In: Proceedings of the tenth ACM international conference on motion in games (MIG); 2017. p. 10:1–10:8. ISBN 978-1-4503-5541-4.
[8] Golas A, Narain R, Lin M. Hybrid long-range collision avoidance for crowd simulation. In: Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games I3D; 2013. p. 29–36.
[9] Treuille A, Cooper S, Popović Z. Continuum crowds. ACM Trans Graph 2006;25(3):1160–8.
[10] Narain R, Golas A, Curtis S, Lin M. Aggregate dynamics for dense crowd simulation. ACM Trans Graph 2009;28(5):122:1–122:8.
[11] Jiang H, Xu W, Mao T, Li C, Xia S, Wang Z. Continuum crowd simulation in complex environments. Comput Graph 2010;34(5):537–44.
[12] Helbing D, Johansson A, Al-Abideen H. Dynamics of crowd disasters: an empirical study. Phys Rev E 2007;75(4):046109.
[13] Golas A, Narain R, Lin M. Continuum modeling of crowd turbulence. Phys Rev E 2014;90:042816.
[14] Macklin M, Müller M, Chentanez N, Kim T. Unified particle physics for real–time applications. ACM Trans Graph 2014;33(4):153:1–153:12.
[15] Reynolds C. Flocks, herds and schools: a distributed behavioral model. Comput Graph 1987;21(4):25–34.
[16] Reynolds C. Steering behaviors for autonomous characters. In: Proceedings of the game developer conference, 1999; 1999. p. 763–82.
[17] Helbing D, Farkas I, Vicsek T. Simulating dynamical features of escape panic. Nature 2000;407(6803):487–90.
[18] Karamouzas I, Heil P, van Beek P, Overmars M. A predictive collision avoidance model for pedestrian simulation.. In: Proceedings of the ninth international conference on motion in games, 9; 2009. p. 41–52.
[19] Karamouzas I, Sohre N, Narain R, Guy S. Implicit crowds: optimization integrator for robust crowd simulation. ACM Trans Graph 2017;36(4):136:1–136:13.

[20] Van den Berg J, Lin M, Manocha D. Reciprocal velocity obstacles for real–time multi-agent navigation. In: Proceedings of the international conference on robotics and automation. IEEE; 2008. p. 1928–35.
[21] Guy S, Chhugani J, Kim C, Satish N, Lin M, Manocha D, et al. Clearpath: highly parallel collision avoidance for multi-agent simulation. In: Proceedings of the symposium on computer animation SCA; 2009. p. 177–87.
[22] Ren Z, Charalambous P, Bruneau J, Peng Q, Pettré J. Group modeling: a unified velocity-based approach. Comput Graph Forum 2017;36:45–56.
[23] Guy S, Chhugani J, Curtis S, Dubey P, Lin M, Manocha D. Pledestrians: a least--effort approach to crowd simulation. In: Proceedings of the symposium on computer animation SCA; 2010. p. 119–28.
[24] He L, Pan J, Narang S, Manocha D. Dynamic group behaviors for interactive crowd simulation. In: Proceedings of the symposium on computer animation; 2016. p. 139–47.
[25] Yeh H, Curtis S, Patil S, van den Berg J, Manocha D, Lin M. Composite agents. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation. Eurographics Association; 2008. p. 39–47.
[26] Bruneau J, Pettré J. Energy-efficient mid-term strategies for collision avoidance in crowd simulation. In: Proceedings of the symposium on computer animation; 2015. p. 119–27.
[27] Kim S, Guy S, Manocha D. Velocity-based modeling of physical interactions in multi-agent simulations. In: Proceedings of the symposium on computer animation; 2013. p. 125–33.
[28] Burstedde C, Klauck K, Schadschneider A, Zittartz J. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. Physica A Stat Mech Appl 2001;295(3–4):507–25.
[29] Kim S, Guy SJ, Liu W, Wilkie D, Lau RW, Lin MC, et al. Brvo: predicting pedestrian trajectories using velocity-space reasoning. Int J Robot Res 2015;34(2):201–17.
[30] Wolinski D, Lin MC, Pettré J. Warpdriver: context-aware probabilistic motion prediction for crowd simulation. ACM Trans Graph 2016;35(6):164:1–164:11.
[31] Stam J. Nucleus: towards a unified dynamics solver for computer graphics. In: Proceedings of the IEEE conference on computer-aided design and computer Graphics. IEEE; 2009. p. 1–11.
[32] Barreiro H, García-Fernández I, Alduán I, Otaduy MA. Conformation constraints for efficient viscoelastic fluid simulation. ACM Trans Graph 2017;36(6):221:1–221:11.
[33] Macklin M, Müller M, Chentanez N. XPBD: position-based simulation of compliant constrained dynamics. In: Proceedings of the ninth international conference on motion in games (MIG); 2016. p. 49–54. ISBN 978-1-4503-4592-7.
[34] Bender J, Koschier D, Charrier P, Weber D. Position-based simulation of continuous materials. Comput Graph 2014a;44:1–10.
[35] Liu T, Bargteil A, O'Brien J, Kavan L. Fast simulation of mass-spring systems. ACM Trans Graph 2013;32(6):209:1–7.
[36] Wang H. A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. ACM Trans Graph 2015;34(6):246:1–246:9.
[37] Bouaziz S, Martin S, Liu T, Kavan L, Pauly M. Projective dynamics: using constraint projections for fast simulation. ACM Trans Graph 2014;33(4):154:1–154:11.
[38] Narain R, Overby M, Brown G. ADMM $\supseteq$ projective dynamics: fast simulation of general constitutive models. In: Proceedings of the symposium on computer animation; 2016. p. 21–8.
[39] Bender J, Müller M, Otaduy M, Teschner M, Macklin M. A survey on position-based simulation methods in computer graphics. Comput Graph Forum 2014;33(6):228–51.
[40] Bender J, Müller M, Macklin M. A survey on position-based dynamics. In: Proceedings of the Eurographics tutorials. Eurographics Association; 2017. p. 1–31.
[41] Helbing D, Farkas IJ, Molnar P, Vicsek T. Simulation of pedestrian crowds in normal and evacuation situations. Pedestr Evac Dyn 2002;21(2):21–58.
[42] Hughes RL. A continuum theory for the flow of pedestrians. Transp Res Part B Methodol 2002;36(6):507–35.
[43] Hughes RL. The flow of human crowds. Annu Rev Fluid Mech 2003;35(1):169–82.
[44] Seyfried A, Steffen B, Klingsch W, Boltes M. The fundamental diagram of pedestrian movement revisited. J Stat Mech Theory Exp 2005;2005(10):P10002.
[45] Schechter H, Bridson R. Ghost SPH for animating water. ACM Trans Graph 2012;31(4):61:1–61:8.
[46] Macklin M, Müller M. Position based fluids. ACM Trans Graph 2013;32(4):104:1–104:12.
[47] Green S. Particle simulation using CUDA. NVIDIA Whitepaper, 2010, p. 1–12.
[48] Knoblauch R, Pietrucha M, Nitzburg M. Field studies of pedestrian walking speed and start-up time. Transp Res Record J Transp Res Board 1996;1538:27–38.
[49] Cividini J, Appert-Rolland C, Hilhorst H-J. Diagonal patterns and chevron effect in intersecting traffic flows. EPL Europhys Lett 2013;102(2):20002:1–6.
[50] Shao W, Terzopoulos D. Autonomous pedestrians. Graph Models 2007;69(5–6):246–74.
[51] Yu Q, Terzopoulos D. A decision network framework for the behavioral animation of virtual humans. In: Proceedings of the symposium on computer animation SCA; 2007. p. 119–28. ISBN 978-1-59593-624-0.
[52] Huang W, Terzopoulos D. Door and doorway etiquette for virtual humans. IEEE Trans Vis Comput Graph 2018:1–16. doi:10.1109/TVCG.2018.2874050. Early Access
[53] Forootaninia Z, Karamouzas I, Narain R. Uncertainty models for TTC-based collision avoidance. In: Proceedings of the conference on robotics: science and systems; 2017.