Explicit Modeling of Personal Space for Improved Local Dynamics in Simulated Crowds

OMAR HESHAM and GABRIEL WAINER, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada

Crowd simulation demands careful consideration in regard to the classic trade-off between accuracy and efficiency. Particle-based methods have seen success in various applications in architecture, military, urban planning, and entertainment. This method focuses on local dynamics of individuals in large crowds, with a focus on serious games and entertainment. The technique uses an area-based penalty force that captures the infringement of each entity's personal space. This method does not need a costly nearest-neighbor search and allows for an inherently data-parallel implementation capable of simulating thousands of entities at interactive frame rates. The algorithm reproduces personal space compression around motion barriers for moving crowds and around points of interest for static crowds.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; **Simulation by animation**; • **Human-centered computing** → **Visualization**; • **Computing methodologies** → **Animation**; **Collision detection**;

Additional Key Words and Phrases: Agent-based modeling, crowd pedestrian models, personal space, GPU

ACM Reference format:
Omar Hesham and Gabriel Wainer. 2021. Explicit Modeling of Personal Space for Improved Local Dynamics in Simulated Crowds. *ACM Trans. Model. Comput. Simul.* 31, 4, Article 23 (July 2021), 28 pages.
https://doi.org/10.1145/3462202

1 INTRODUCTION

Real-time simulation of dense crowds is finding increased use in the context of event planning, congestion prediction, and threat assessment. Crowd simulation algorithms have also been used in the entertainment and serious gaming industries. Existing particle-based methods assume and aim for collision-free trajectories. This is an ideal—yet not overly realistic—expectation, as near-collisions increase in dense and rushed settings compared with typically sparse pedestrian scenarios.

While theoretical accuracy is desired for planning complex emergency scenarios, the goals of our research favor other objectives: visual believability, a certain degree of artistic control, and fluid simulated scenarios. In cases of high density and near hot spots of interest, handling personal space can become a major issue to deal with. The compression of personal space is accumulative; the barriers then create the conditions for exceeding the critical crowd densities. The objective is to

This research was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) (code N/A).
Authors' addresses: O. Hesham and G. Wainer, Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Dr., Ottawa, ON, K1S 5B6, Canada; emails: omar.hesham@carleton.ca, gwainer@sce.carleton.ca.

1049-3301/2021/07-ART23 $15.00
https://doi.org/10.1145/3462202

Fig. 1. Crowd abstraction hierarchy.

define a simple mechanism to obtain advanced close interaction simulations that improve existing methods to deal with close-range collision avoidance and personal space management.

This research assumes a hierarchy that divides the crowd modeling and simulation process into three levels: cognitive, global path computation, and local interaction, as illustrated in Figure 1.

The cognitive model focuses on broad decision-making, such as deciding where the target location is, and interacting with entity goals and personality traits that could alter such decisions (e.g., taking the stairs instead of the elevator, or following a parent during an evacuation instead of taking the nearest exit). Once the target location is decided, the global path is computed based on the spatial structure of the mostly static environment and finds an optimal path according to some cost function (typically, a short path or a least congested one). Finally, the local interaction module further modifies the path to navigate around and avoid collision with minor dynamic obstacles: other pedestrians, gates, and doorways (generally following the optimal path and not straying too far from it). This hierarchy encourages separation of concerns and allows further experimentation and mixing of components and solutions from various sources. *Our contribution focuses on improved local dynamics in the modeling of personal space; we model personal space explicitly and obtain specific emergent behavior that other methods cannot reproduce easily. This specific type of emergent behavior does not need to be programmed explicitly.* The method we propose, called the Centroidal Particle Dynamics method (CPD), adds anticipatory collision avoidance and can offload CPU workload to graphics cards for increased performance and higher frame rates.

CPD uses an area-based penalty force that allowed us to obtain good results using consumer-grade graphics hardware. The model produces specific *emergent behavior found in known crowd phenomena* (in particular, lane formation in bidirectional flow, regular banding around areas of congestion, petal formation in crowds, stop-and-go behavior, concentration around points of attention, collision avoidance) *without the need for introducing specific rules* in the model to achieve any of those specific behaviors. In addition, still compression effects can be found (i.e., accumulative reduction of personal space near areas of high congestion and around points of interest). CPD can generate these varied behaviors with ease as well as static/near-static crowds. In these cases, previous methods that rely on relative velocities between entities have failed (as they have a difficult time distinguishing between static entities at varying distances from a barrier or an area of collective interest). For instance, Figure 2 shows how the area-based force of CPD can aggregate such compression of personal space in high-density stationary crowds without defining such behavior explicitly. Furthermore, the area force is customizable through graphical parameters that are designer friendly for creative control and for the introduction of non-homogeneity into the system. This local interaction force can be integrated with existing global pathing schemes or used to augment the calculations of other local avoidance methods.

We will discuss our contribution to the development of microscopic methods catered to dense crowd phenomena. We conducted qualitative analysis with real-world data (due to our objectives stated earlier, quantitative analysis is outside the scope of this research). The method

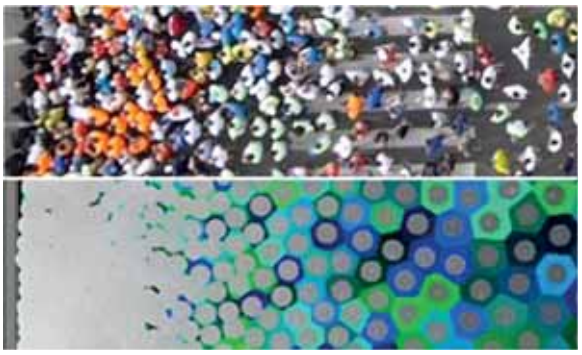

Fig. 2. An "invisible force": gradual compression of personal space among a static crowd near areas of interest or barriers to desired motion. Marathon start line (top); simulated result (bottom).

has potential for real-world emergency scenarios or large crowd models in urban environments although a thorough quantitative analysis should be conducted before its applicability in these cases. Similarly, the hierarchical organization of crowd abstraction presented in Figure 1 allows integration with path planning and cognitive models. We will show how CPD improves the quality of individual interactions. Another aspect is the specific emergent behavior found in the execution of the method under different conditions: lane formation in bidirectional flows, compression of space in congested areas, petal formation, and other patterns in stationary crowds, without explicitly programming any of these specific behaviors.

The relevant background is discussed in Section 2. Section 3 presents our proposed method. Section 4 illustrates our method's results and real-time performance. The discussion in Section 5 reflects on the method's limitations and hosts opportunities for further discussion and future work. We discuss the relevant background next, followed by a geometric description of our proposed method and a discretized graphics pipeline implementation. Lastly, we highlight some key results before briefly discussing opportunities for improvement, including thoughts on the conversion from a discrete-time evaluation to a more efficient discrete-event model.

2 BACKGROUND

Human motion is seemingly non-deterministic; hence, pedestrian simulation will always be an exercise in abstraction. Modeling and Simulation (M&S) of human behavior and decision-making is one of the most complex in M&S—crowds are no exception. *Dense* crowd M&S focuses on predicting the motion of large groups of humans within a limited physical space. The abstraction of motion dynamics based on the generalization of observed phenomena is necessary to obtain computable results [8, 18]. This section presents a brief overview of the multitude of methods developed to tackle this problem.

Early efforts to simulate crowd motion took a *macroscopic* approach and viewed the crowd as a continuous fluid-like field. The behavior was generated using finite element solvers to evolve aggregate density and velocity fields in time [19]. Later versions showed stop-and-go waves and bottleneck clogging [55]; methods such as Continuum Crowds [53] delivered large-scale results at interactive frame rates, suitable for animation, gaming, or training. Other macroscopic methods took an operational research approach: they adapted network optimization techniques to simulate occupant movement within a predefined multi-compartment environment [21, 52] where each graph node represented a building section and links represented the capacity of pedestrian nodes. Using optimization, designers focused on areas of potential bottlenecks. Overall, macroscopic methods remain popular due to their computational efficiency and their ability to provide

insight into aggregate crowd dynamics, especially for large-scale scenes with high crowd counts [8].

Unlike macroscopic methods, *microscopic* methods simulate entities as individual agents with localized rulesets whose emergent global behavior resembles reality. Some of the earliest examples include Cellular Automata (CA) and Lattice Boltzmann (LBM) models [1, 2]. In CA, a space is discretized into a uniform lattice of tillable shapes that form a consistent neighborhood pattern. A global clock triggers a simultaneous update of all cells, where the next state of each cell is determined by the state of the cells in its neighborhood [57]. Cell-DEVS provides a discrete-event parallel modeling and simulation approach to solving microscopic pedestrian flow problems [12].

Eulerian evaluation with discretized stepping and finite directions of motion cannot faithfully reflect the fluidity and details of human motion. Lagrangian methods, typically implemented as free-moving particles, perform their computations in-place. Successful efforts were introduced by Helbing's social forces [14], or HiDAC, which incorporates psychological profiles and pushing [34]. When taken to an extreme, microscopic algorithms could opt to simulate the joints of every entity (e.g., legs on a human or pedals on a bike) to generate mechanically accurate locomotion [5]. A key element of Lagrangian methods is neighborhood detection every time advance, which is the primary cost when compared with Eulerian evaluation, in which neighborhoods are predefined and accessible. Certain data structures can be used to accelerate this stage, primarily involving a spatial tessellation such as an Octree or a Voronoi diagram, which is used to limit the search area and can accelerate neighborhood detection using GPUs [42, 43].

Human motion is empirically shown to be anticipatory [37, 38]: we scan the environment for potential collisions and enact local maneuvers to avoid them. Agent-based models were built on this principle—including Reciprocal Velocity Obstacle (RVO) [3], ORCA [50], or learning agent models [25, 26]—in particular, those that can use live crowd data to adjust their behavior in real time [27]. Other efforts try to mimic vision-to-motion by rendering a one-dimensional (1D) [29] or two-dimensional (2D) [33] depth map from each entity's perspective and emulating how we adjust trajectories based on parallax and depth perception.

Our research introduces a new method based on the physiological origins of personal space, which arises primarily from two factors:

—Biological: When a person crosses the boundaries of personal space, the amygdala (an area in the brain) stimulates the sympathetic nervous system, provoking *fight or flight* behavior. This results in personal space, which is a mechanism that helps keep people safe from danger and gives them time to react in case of danger. This applies to strangers not close to the person (people with brain damage or defects in the amygdala have difficulty discerning appropriate personal space).

—Social: Personal space is also called the *safe zone*, a specific area around a person that, if breached, results in an uncomfortable feeling. The safe zone varies per the individual and the society in which the individual lives. When in crowds, we are not as uncomfortable around each other, as we handle the situation by temporarily dehumanizing people around us [51] until we spot an escape route [13]. These personal space bubbles start forming between the ages of 3 and 4 years and they have a fixed size around adolescence. Bubbles are socially and culturally constructed, and they are combined with the fear triggered by the amygdala.

We proposed the Centroidal Particles approach [15, 16], a new Lagrangian approach for realistic depiction of personal space compression in congested settings, which is based on the two ideas just discussed: the social (safe zone) and biological (fight or flight) personal spaces. The methods try to reconstruct this basic human behavior for individuals and combine thousands of individuals, generating scenarios in which behavior emerges. The application target is in film, gaming, and

training applications. Although safety-critical applications such as civil planning, crowd control, and large-scale event threat assessment could benefit from a dense-crowd simulation method such as ours, we do not yet endorse the use of CPD, as further analysis is required for those applications. Other state-of-the-art methods that share our target applications include WarpDriver [59], ORCA [50], RVO [3], Social Forces [14], and position-based [54] crowds. Microscopic methods typically excel in sparse-crowd simulation. RVO and ORCA have enjoyed success in real-time multimedia/gaming engines. However, they struggle to reproduce convincing trajectories, *particularly in dense bidirectional flow scenarios, due to emergent lane rigidity or artificial congestion.* One of the causes is a rigid 1D separation distance between entities. In contrast, CPD uses a compressible 2D area. To tackle dense crowds, one approach is to model aggregates of local crowd flow instead of the trajectories of individual entities [31]. While this approach allows for real-time simulation of thousands of entities at interactive frame rates, it can create the appearance of overly coordinated motion among local pockets of the crowd. Other approaches include energy minimization to reduce the effort cost over a given pedestrian's entire trajectory, short-range stochastic motion-prediction based on prior collision experiences [28], and position-based dynamics that adapt existing fluid and soft-body physics solvers for use in crowd simulation [50]. Implicit Crowds is a particularly interesting recent development that allows for smooth trajectories using much larger timesteps than is required from typical numerically simulated crowds [20]. An important difference between this state-of-the-art method and CPD is that we are not explicitly querying the neighborhood of each pedestrian (including its basic force parameters, direction vectors, distance between pedestrians), and combine them to compute the final force acting on a pedestrian. Each neighbor must be queried. Instead, we use a rendering algorithm to compute individual kernels for each pedestrian (which can run in parallel—for instance, using GPUs, as in our proposed implementation). Another important difference with current research is that obstacles are also computed in the same visual space: when we compute personal space, we no longer must worry if the space is reduced because of an obstacle or a person: the computation is agnostic as to why the space is reduced. In recent years, different authors proposed methods for quantitative evaluation for studying the fidelity of crowd models. As explained earlier, this evaluation is outside the scope of this research, as our current objective is to obtain realistic visual results and complex emerging behavior matching real scenarios for animation and training. Applying our research to real-world crowd management scenarios would require exhaustive validation with quantitative studies. For instance, ProactiveCrowd [24] provides a method and tool to compare trajectory similarity between agents and human data adopting the Longest Common Subsequence metric. Similarly, Wolinsky et al. [60] define an evaluation framework for parameter estimation, although the scope of their study is not sufficient for our case studies. They use around 150 agents and we are interested in experiments on specific emergent behavior in close-range interaction with a minimum of 2,000 agents and up to over 100,000 agents, as shown in Section 4. The Fundamental Diagram [30], which focuses on density and collisions, would provide metrics on the average frame computation; nevertheless, the authors do not discuss how they conduct real-world validation (they validate three reproductions on live pedestrians, using the same dataset we have used for Section 5). The work in [56] introduces a new semantic-level crowd evaluation metric, which can be used to analyze quantitative results. The method can be used to study path planning. This method is not useful for our person-to-person interactions in dense crowds. As discussed in Figure 1, we count on a high-level path planning method such as the one presented and evaluated quantitatively in [56], which introduces the idea of a trending path. Their quantitative analysis focuses on discovering latent path patterns. They state that their "…method does not directly measure individual trajectories thus does not reflect individual visual similarities," which is the main objective of CPD. An integration of the CPD and a path planning method such as the one in [56], with detailed quantitative

analysis for the overall path planning and interaction, is an interesting research topic for future investigation.

According to the established wisdom in crowd M&S literature (extensively surveyed in [47]), different metrics from those discussed earlier, when pursued and measured properly, would improve the confidence in using our proposed CPD algorithm. However, the very concept of data validation in the context of crowd simulation is not without its criticism. The fire safety literature [41] brings attention to the absence of an international standard for verification and validation of evacuation models and that "validity" is still ambiguous and can have different meanings and rigor for different experts. They argue that the problem is further compounded by the M&S literature's tendency to "validate" against data "outside their original context of use" (e.g., building evacuation data used to validate ship or stadium evacuation). Another critique comes from the field of neurocomputing, which argues that collision avoidance methods should take macroscopic statistical "truths" into account when deriving microscopic models: instead of calibrating abstract model parameters and "hoping" to validate the model by achieving certain macroscopic properties that match the statistical data, it is argued that those learned macroscopic truths should be known to the microscopic model beforehand, thus, guaranteeing the desired emergent macroscopic properties. This is a bit too restrictive, in our view, and it encourages a model that departs from the way that actual pedestrians process their surrounding stimuli and make collision-avoidance decisions locally. Their recommended approach means that each virtual pedestrian would have more knowledge about the surrounding aggregate dynamics than the real pedestrian entity it supposedly models.

Our primary application target requires the simulation to perform at real-time (simulation time advances at least as fast as wall clock time) or at interactive frame rates (10 fps, as user interaction experiments in the context of software usability have shown that 100 ms response time was perceived as fluid or instantaneous feedback to user actions [32]). To be specific, the type of motion we aim to simulate pertains to the update of each pedestrian's position (the pedestrian's center of mass) in scenarios of large-count, high-density crowds. This situation arises commonly (but not exclusively) at stadiums, concerts, busy shopping malls, mass protests, and during building evacuations. The sheer number of pedestrians in the scene, reaching tens or hundreds of thousands (or even millions, such as the crowds gathered at past US presidential inaugural events [39]), presents a computational challenge for methods that share our goals. Separately, the high density presents a modeling challenge, where local agent-based rules would reproduce global motion phenomenon, as observed.

Our interpretation of a dense crowd is based on the report on contingency planning by the United States Federal Emergency Management Agency (FEMA) [23]. From the perspective of a pedestrian, FEMA categorizes crowd densities as follows:

— ~2.3 m^2 (25 ft^2)/pedestrian: Normal walking speed and comfortable maneuverability.
— ~0.9 m^2 (10 ft^2)/pedestrian: Restricted movements and noticeably slower speeds.
— ~0.5 m^2(5 ft^2)/pedestrian: Shuffling gait; calm motion as a group; difficult to overtake others.
— <0.3 m^2 (3 ft^2)/pedestrian: Brushing and close contact with surrounding entities.
— <0.2 m^2 (2 ft^2)/pedestrian: Dangerous density with potential crushing injury.

For our purposes, a *dense* area is one with 0.3 to 0.9 m^2 per pedestrian. At less than 0.3 m^2, it is considered a contact-collision with possible injury. We note that CFD does not simulate the physics of contact collisions or friction among pedestrians as prior rigid/soft-body simulation literature is already capable in this area and it is outside our scope of research. Rather, we are interested in the dynamics of local *collision avoidance attempts in dense crowds* and the emerging motion patterns.

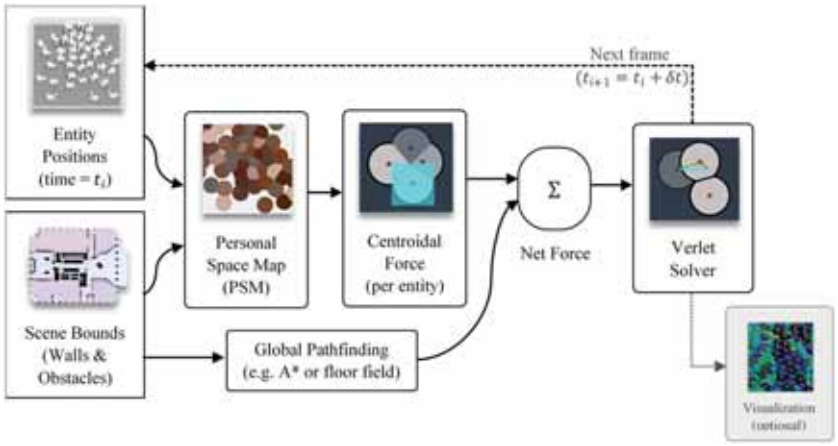

Fig. 3. Overview of proposed pedestrian update cycle.

3 CENTROIDAL PARTICLES

The Centroidal Particle Dynamics (CPD) method is a personal area-based microscopic method that produces physical interactions, pressure propagation waves, and compression of personal space due to the crowd's collective attraction to an area of interest, in which it behaves as a compressible fluid even when the crowd is stationary.

To simulate local pedestrian interactions, we first compute a combined personal space violation map from which we can compute reactive penalty forces. The way each entity contributes to the personal space can be modified and parameterized visually through weight maps. Finally, the new location for each entity is computed by integrating the net acceleration force iteratively over several frames. Figure 3 shows an overview of the method. In the first step, we use entity (pedestrian) positions and scene bounds and construct a model of a *Personal Space Map* (PSM). In this model, the personal space (PS) is computed to find out whether the PS areas of two entities overlap (called *sharing/violating* the PS of each other). The generation of the PSM is a global operation that explicitly tessellates the scene's area to map unshared PS areas to each entity. The next step is using a model that represents each pedestrian examining one's immediate surrounding, calculate how much PS is violated, and reacting. To do so, a new geometric center (called the *centroid*) is computed using the unviolated PS. As well, obstacles are rendered in the same distance field space. We generate a vector pointing the pedestrian to this new centroid (the *Centroidal Force*), which points to the place where the pedestrian will regain the most amount of PS possible. Then, we integrate it with other forces (which can have different weights): global path, friction, proximity to family members, and the like. Lastly, we apply an advection model due to the net acceleration experienced by each pedestrian, which is integrated using a numerical solver (we use a Verlet–symplectic integrator, a semi-implicit integration method with the computational efficiency of an explicit solver, such as explicit Euler, and numerical stability, such as an implicit solver, without introducing additional energy and providing long-term stability and reducing error propagation [4]).

We now discuss some details on each of the steps.

(a) Personal Space Map: To be considered unviolated, any point in each entity's PS must be closer to that entity than to any other entity. This concept evokes the tessellations produced by the Voronoi diagram. In fact, our definition of shared spaces can be geometrically represented by a truncated Voronoi tessellation [17] that does not need to be computed pairwise: it can represent

Fig. 4. (a) Subsection of a crowd; (b) PSM; (c) PSM visualized as an underlay.

an aggregated account of all PS violations of an entity in its neighborhood. The resulting PSM outlines all PS violations across the crowd (Figure 4).

The PSM is defined as a partitioning of a 2D plane G (the ground) with obstacles (walls, gates, vehicles, etc.) and pedestrians. After the PSM is built, every point g in G will belong to only one entity (e.g., pedestrian, obstacle, or unoccupied space), as illustrated in Figure 4. Let T denote the many-to-one mapping that partitions plane G. All points in G are considered *unoccupied* to represent all of the empty space available for any pedestrian to traverse (i.e., $T(g) = 0$ for all $g \in G$). Let us assume that each pedestrian is assigned a unique ID from 1 to n. Finally, obstacles can be explicitly defined by the modeler—for example, the set $B \subset G$—which denotes the areas that the pedestrian needs to avoid. Additionally, scene geometry can be projected onto G as if viewed orthogonally from the top. Most scene geometry is already in 2D (e.g., architectural floor plans), but any 3D geometry (e.g., columns) needs to be projected explicitly onto the PSM for collision avoidance. To project 3D meshes onto G, the following transform is applied per vertex:

$$Proj_G(v) = v - (\vec{n}_G \cdot v) \times v,$$

where $\vec{n}_G$ is the plane's unit normal vector and $v=(v_x,v_y,v_z)$ is a vertex position that has a height v_y between 0 m (ground) and 3 m (max human height) and does not explicitly belong to a ceiling element. Then, for every point $g \in G$ that falls within the polygons formed by $Proj_G(v)$, we set $g \in B$. Hence, the set B contains all boundary points in space G that were defined explicitly by the modeler along with all of the scene obstacle projections. If we denote $d(a,b)$ as the Euclidean distance between any two points a and b on G, then the tessellation is:

$$T(g) = \begin{cases} i, & (d(g, p_i) < r_i) \wedge (d(g, p_i) < d(g, p_j)) \wedge g \notin B \\ -1, & \text{for all } g \in B \text{ (overrides all other cases)} \\ 0, & \text{otherwise} \end{cases},$$

where $i, j \in \{1, \ldots, n\}$; p_i is the position of pedestrian i, and r_i is the radius of i's PS. We do not explicitly differentiate between obstacles and assign them an *obstacle* value (−1). The entire PSM tessellation process is memoryless and gets reconstructed on every timestep. **In doing so, the PSM can account for dynamic obstacles** in the scene, such as revolving doors. The normal vector used in the projection step could be altered to accommodate basic inclines, uneven terrain, and stairs.

(b) Centroidal Force: The PSM defines the overall PS that the pedestrians occupy. Each entity needs to view only its personal neighborhood (within radius r_i) to compute quantities related to its PS violations. We compute the PS *centroid*, that is, the new center of mass of the current PS, which is the average position of all unviolated points in the PS area, to mimic people moving in the direction of the centroid to regain the most PS possible. Each entity is represented by a particle in the 2D plane surrounded by a contiguous PS footprint (based on [7, 13], we use ~0.8 m evenly around the center). When entities get close to each other, we assume that the PS is shared equidistantly: the closer they get, the more they (equally) violate the other's PS. Experiments also have shown

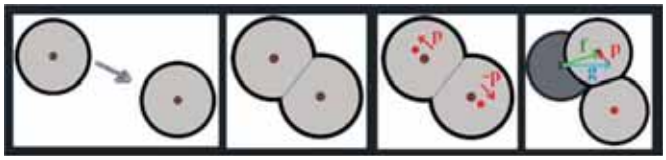

Fig. 5. Formation of new centroids during collisions.

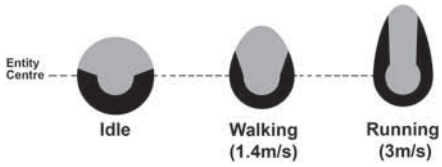

Fig. 6. PS shape: Light area affects the entity and neighbors; dark area affects only neighbors.

that there is a short delay in human response to react and enact collision avoidance (~150–350 ms). We use an area-based penalty force that reacts to the PS violation iteratively, attempting to restore the preferred PS area over time. To be considered unviolated, any point in an entity's PS must be closer to that entity than any other. This is like the Centroidal Voronoi Tessellation (CVT) relaxation algorithm [17]; we then overlay the combined PS onto the shared PSM, which aggregates and outlines all PS violations. Given the PSM, each entity independently computes the current unviolated area's centroid (in CVT relaxation, the particle simply moves to the new centroid, whereas we use the vector from the center of the original footprint to this new centroid), as illustrated in Figure 5.

We can see that there is a primary directional force that moves the entity towards its destination (following the global pathing shown in Figure 1). We then compute the net force f as a linear combination of the global pathing force g and the penalty force p, which falls along the direction of the new centroid. The penalty force produces the reactive behavior of our crowd. The magnitudes of those forces and their stochastic variations are adjusted to create a smooth transition to the entity's desired speed, on average, 1.4 ± 0.24 m/s [7, 13].

We can modify the local dynamics by changing the footprint's geometry or using a map to influence its weight. The PS footprint can be artificially varied over time or in response to events (e.g., a fire alarm evacuation) or in proximity to points of interest (e.g., slowing down when window shopping or near interesting booths at a busy exhibit hall). The shape can also reflect the entity type (e.g., adult, child, stroller). The footprint can also accept a weight map that through simple convolution varies the impact of the PS infringement—for instance, placing a slightly heavier weight on the right to indicate a preference for taking the "right lane" when encountering oncoming traffic.

Based on empirical results [37], CPD uses a PS shape evenly weighted around the entity. Even if an entity has its PS infringed upon outside of its vision, the individual reacts to pressure or sound. We also provide the option of using a multi-area kernel that splits the shape into two key areas: (i) the PS area that affects both the entity and its neighbors; and (ii) the PS area that affects only surrounding neighbors (Figure 6). Now, the net separation between entities remains at ~1 m, but the entity with visibility will shoulder most of the corrective efforts to maintain the distance. The entity in the back maintains most of the separation distance; thus, the severity of PS compression around areas of congestion is reduced. Additionally, we have an extension proportional to the entity velocity: the PS extends by ~0.4 m per m/s. This extension is slight for walking speeds (~1.4 m/s), but noticeable at running or cycling speeds (>3 m/s). Figure 6 illustrates this multipart

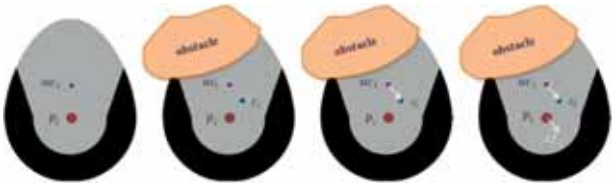

Fig. 7. Steps to compute the centroidal force.

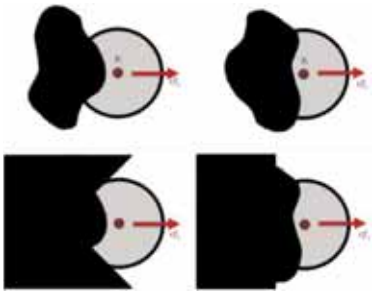

Fig. 8. Centroidal force on different types of obstacles.

modification, accounting for the narrowing focus of speeding entities, providing collision *anticipation*.

Taking these aspects in consideration, the equation for computing the centroid is

$$c_i = \left(\frac{\iint_{PS_i} x \cdot \lambda_i(x,y) \cdot w_i(x,y)\ dxdy}{A_i(s)}, \frac{\iint_{PS_i} y \cdot \lambda_i(x,y) \cdot w_i(x,y)\ dxdy}{A_i(s)} \right),$$

$A_i(s)$ = Euclidean area of active region of PS_i given speed s,

$$\lambda_i(x,y) = \begin{cases} 1, & sample_G(x,\ y) = i, \\ 0, & \text{otherwise}, \end{cases}$$

$$w_i(x,y) = \text{mask sampling function} = \begin{cases} 1, & (x,\ y) \in active\ PS_i \\ 0, & \text{otherwise}, \end{cases}$$

The original centroid of PS_i's active region (uc_i) is not guaranteed to be at the entity's position (pi). Thus, the new centroidal force (illustrated in Figure 7) is computed as $cf_i = c_i - uc_i$.

We employ the notion of the Fundamental Diagram [45, 46] in which pedestrian speeds, on average, vary inversely to their local density. Empirical data have shown that the Fundamental Diagram differs depending on the context of the crowd (e.g., indoors vs. crosswalk) and across cultures. We use it to determine the desired speed for each individual entity dynamically throughout the simulation.

As the centroidal force is an aggregate measure of PS violations, regardless of the obstacle's curvature (axis-aligned, convex, concave, etc.), the centroidal vector will point away from PS violations, which allows collision avoidance with obstacles of arbitrary shapes (e.g., vegetation, furniture, etc.). Pedestrians perform an iteration of the Lloyd algorithm [22] on truncated Voronoi cells representing PS. When an entity has all of its PS unviolated, the centroid is simply the entity's current position. Let $PS_i \subset G$ denote pedestrian i's unviolated PS region. When another entity/obstacle invades i's PS, the new centroid position $c_i = (\tilde{x}, \tilde{y})$ is computed, as seen in Figure 8. The actual implementation of these integrals boils down to a couple of weighted summations over a 2D grid of PSM pixels.

(c) Global Path Finding: The global path is one that, absent any other dynamic entities in the scene, would guide an entity to its destination efficiently (e.g., shortest time, cost, etc.). The only obstacles considered by the global path are static scene elements. The literature is saturated with methods in this area (e.g., A*, floor fields, AI navigation trees [9]). Thus, we assume this to be an input into our net force integrator. If the scene obstacles are defined over a graph, then a method such as A* [63] would work well, as it efficiently computes such global paths per entity and can be updated often. For large-scale simulations with thousands of pedestrians with relatively few possible global targets within the scene (e.g., only dozens of shops in a mall), a single global floor map per target is more efficient. The map is essentially a 2D gradient field that points an entity to the direction it needs to follow to reach the global target. See [2, 53] for example implementations of this floor field.

CPD is a local dynamics method that does not place limitations on the kind of global pathing algorithm used. This is a direct illustration of the *separation of concerns* discussed earlier in Figure 1. However, as this research focuses on the local dynamics layer, we experimented with the simplest forms of global pathing (to reduce the influence of "intelligent" path finding over CPD independently). Some of the experiments to be discussed later discarded global paths entirely to evaluate the emergent behavior from local dynamics alone (e.g., aimless crowds experiencing overcrowding and stationary crowds at a concert). The global pathing forces we used are time invariant; they rely only on the current position of the entity to find the "next" step along the global path. Prior to the force integration step, we computed a resistance to centroidal forces that opposed the entity's global path/objective, inspired by the energy-minimization goals set in ORCA. It reduced the "springiness" of near-miss collision in our pedestrian crossings significantly. Even if the local centroidal force is pointing the entity to face away from the global path, the entity will *resist* this change and attempt to wait until more favorable centroidal forces are available.

(d) Net Force: The total force experienced by each entity is a weighted sum of the local forces and the global pathing direction. In ideal conditions with a single entity in the scene, it would simply follow the current global path to the destination. However, the local forces enact collision-avoidance maneuvers with their surrounding environment and entities. The net force calculation is

$$nf_i = \alpha\ cf_i + \beta\ gf(p_i) + \gamma\ uf_i,$$

where $gf(p_i)$ is the global path vector given i's current position in the scene and α, β, and γ are scalar weights to parametrize the behavior of the entity. An aggressive pedestrian has low α and high β, emphasizing the pedestrian's own global path with little regard for local PS violations (indicated by cf). This is essentially how our close-range pedestrian behavior can be calibrated according to reference trajectory data. We have empirically arrived at a set of parameter values and illustrated their results (in general, we use $\alpha = 0.7$; $\beta = 0.2$; and $\gamma = 0.2$). Lastly, the velocity resulting from the time integration of these forces and their stochastic variations are clamped to stay within the entity's desired speed. We used the average desired speed for walking pedestrian of 1.4 ± 0.24 m/s [37, 38].

It is possible to further refine the parameters and automate their calibration using context-specific reference trajectory data. Existing parameter estimation platforms that capture path planning and collision-avoidance behavior from footage/mocap data could be used to perform looped optimization (or machine learning) to optimize the simulation model's parameters to best fit with the input data. Automating calibration would allow modelling of behavior that changes across cultures, event types, age groups, and unforeseen contexts. However, such platforms encompass the overall motion of the crowd, not just local collision avoidance. This is beyond the scope of this research.

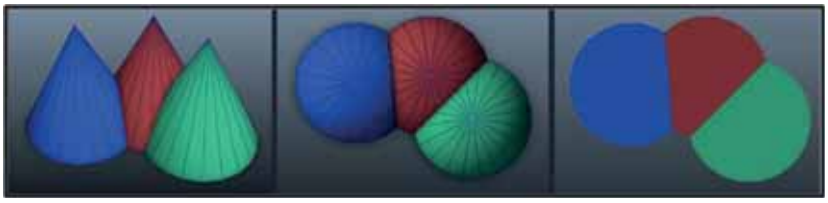

Fig. 9. PSM 2D bitmap: Truncated Voronoi cells created by a top-view orthogonal projection of 3D cones.

4 CPD IMPLEMENTATION

The 2D nature of the PS kernels and their response forces allows us to reason about the proposed method from a Computational Geometry lens, invoking visual data structures and algorithms to implement a computable version of CPD. This section outlines a prototype implementation developed throughout the course of this research. The presented implementation produces online simulation results for thousands of entities in the scene.

4.1 Discrete-Space Implementation of CPD using Truncated Voronoi Disks

Voronoi tessellation has been used to accelerate spatial queries such as nearest neighbor search [11, 43] or global path planning [36]. Recently, it has been used to identify a discrete set of candidate local vectors to pursue, essentially, along one of the neighboring Voronoi cell edges [40, 61]. This approach (which was developed concurrently with our own research) shares the simplicity and intuitiveness of our method but does not explicitly model an entity's PS or its compression; thus; it is unsuitable for dense crowd scenarios or largely stationary ones (e.g., a concert). Our proposed use of the truncated Voronoi diagram as an analog for compressible personal space is unique, as is our visual parameterization through dynamic weight maps.

A spatially discrete version of the continuous PSM presented earlier can be created using a truncated Voronoi diagram whose cells are bounded by a certain distance from their sites. We use the GPU-accelerated computation method for Voronoi diagrams introduced in [17] to produce the PSM, followed by custom shaders to compute the centroidal forces per entity and integrate them with other global forces before finally solving them for timestep δt. In contrast with the continuous-space method described, the simulation space is implemented as a discretized 2D grid (a bitmap). We use textured 3D cones to represent the entities and their personal space, with the tip of the cone representing the PS center, and the base representing its outer edges. In effect, the height along the surface of the cone encodes the distance to the center of the entity. When rendered from an orthographic top view (free of any perspective distortion) facing the tips, two cones will overlap at precisely the points that are equidistant to both entities [17]. Figure 9 shows this procedure.

By encoding the entities as geometric primitives and using the GPU's depth buffer to obtain the PSM tessellation as discrete pixels quickly, we are left with a globally shared data structure (the PSM bitmap) that allows each entity to compute its relative centroid and resulting penalty forces in a data-parallel fashion (Figure 10). The entities do not need to conduct a costly nearest-neighbor search, as they consume and interact with the set of pixels representing their PS in the PSM.

To differentiate between the rendered cones, they are colored using a one-to-one reversible hash map (a function of the unique entity IDs). We reserve the first 10 colors in the 24-bit RGB space for obstacles and debugging purposes and use the rest to uniquely color code each entity. The reverse lookup (which is also a constant cost function) enables any entity to identify the ID of another entity directly infringing on its pixel space. Theoretically, we could simulate more than 16 million entities on a 24-bit RGB PSM surface and potentially billions of pedestrians on higher

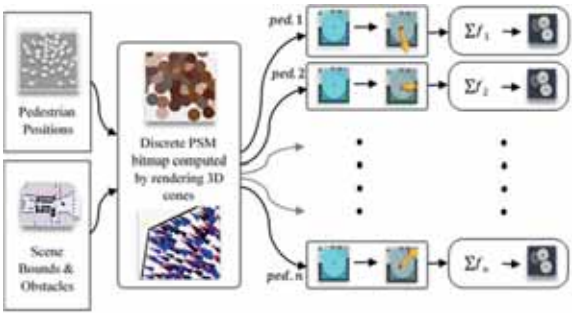

Fig. 10. Data-parallel implementation of CPD.

bit-depth surfaces. Nevertheless, practically speaking, 16 million entities could only be afforded a single pixel each on a 4k × 4k PSM map as a best case-scenario. The cones act as a bounded signed distance field around each pedestrian; by rendering the cones, we compute the distance field for the entire crowd all at once. In using these 3D cones, which can be parameterized to include different kinds of pedestrians, the Z-buffer will generate a correct kernel for each pedestrian without a global neighborhood computation algorithm, as discussed earlier. We offload this computation to the GPU by rendering the 3D cone opposed to traditional methods that need to compute each near-neighbor force.

To compute the true center of PS footprints that have an influence map, vary by time, or are proportional to the entity's speed, we first calibrate each PS shape for bias and then adjust the cone tip and texture accordingly. Without this step, an asymmetrical PS area that, for instance, elongates with speed will experience an ever-increasing force in the direction of travel. Instead, the violation-free bias should be considered to detect true violations of the PS area. The mass of the individuals can be modeled by adjusting the height of the cones. The heavier the individual is (less likely to be affected by force), the closer the cone should be to the camera. Essentially, the lighter individuals would have to exert more force to make up for their increased distance from the camera and infringe on the heavier individual's space.

There are two ways to aggregate the centroid from the PSM, either:

(a) Per-pixel: If there is overcrowding resulting in significant PS overlap and it is more economical to count the visible pixels and aggregate the results using reverse hash lookup.
(b) Per-entity: In sparser scenarios, it is more efficient to skip the PSM's empty pixels and simply count the unviolated pixels near each entity.

A simple heuristic we used in our implementation checks: if $(|entities| \times \pi r^2) > (1.2 \times PSM\ area)$ perform (a); else perform (b), where r is the average PS radius. We take each PSM pixel to represent 10 × 10 cm. Thus, our PS radii can range from 8 to 10 pixels, as sizes vary in a crowd.

We began the exploration of this centroidal area force in the context of crowd simulation due to the natural limits of human acceleration and velocity. While it might be tempting to directly use this force for physics-based simulation of fluids, soft-bodies (such as clay), and granular solids (such as sand), the unpredictable, often chaotic, and extreme accelerations experienced within those bodies require more careful consideration. It would be difficult to dismiss the nearest neighbor search entirely if our "implicit collision" response force were to be used for physics-based simulation. However, in its current form, the effort presented in this article certainly opens the possibility for less critical applications, such as film, gaming, and immersive **virtual reality (VR)** experiences.

Fig. 11. (a) Local density estimated per entity; (b) global low-pass filter applied.

4.2 Scalability

GPUs are efficient when it comes to massively data-parallel processing, but they also have built-in memory limitations on texture sizes and, by extension, the size of the PSM maps that we can compute for a given scene. A possible route is to use a courser grid to represent the scene, at the cost of simulation fidelity. A more reasonable approach in this instance is tiling. Regardless of the shape and complexity of the simulation plane, it can be divided into smaller tiles that fit within the supported texture sizes by the GPU. The overhead in this case will be in cross-tile communication (for entities that fall near the edge of one tile but with a PS area that spills into a neighboring tile).

A straightforward approach that minimizes communication across tiles is to have them overlap their computation space by an amount equal to the average entity PS radius. This means that entities near the tile edges will likely have their forces computed twice (once in each tile). However, the locality of data access within a given tile means that the duplicated computations are still more efficient that the cost of texture-switching and CPU-synchronization required for cross-tile communication. Like how we skip empty pixels in sparse crowds, we extend the concept by skipping entire tiles if they are empty, and processing only the occupied ones. These tiles can be dispatched for simultaneous processing across multiple GPUs if available. Furthermore, the recent development of low-overhead graphics APIs such as Khronos Vulkan and Microsoft DirectX12, which allow and encourage the multi-threaded dispatching of render-calls [48] supports the scalability potential of our method. CPU threads could concurrently launch and synchronize GPU-computable PSM tiles without costly context-switching (a required cost for the ubiquitous graphics APIs: OpenGL/WebGL/DX11).

4.3 Density Estimation

Computing local density is important to conduct flow rate analysis and aggregate PS violation measures. It is possible to obtain an estimation of the local crowd density near an entity by leveraging the existing PSM to compute a density value d_i (entities/m^2) as the reciprocal of the unviolated portion of that entity's PS area. Once known, we render another PSM pass. However, this time, we color every entity i using the estimated d_i, where brighter values correspond to higher densities. We then smooth the discontinuities using a Gaussian blur filter (a standard and parallel-friendly utilization of the graphics pipeline. The resulting smooth density field is shown in Figure 11.

It might be argued that better density estimation methods exist, such as those found in physics simulations using Soft Particle Hydrodynamics (SPH) that rely on cubic kernels to accurately converge to a true density continuum [35]. However, we opted for the Gaussian kernel applied to our PSM with a radius of half a footprint's radius in effective pixels. Not only is the result sufficient for our purposes (an estimation of density) but it also helps that it *is not* too precise, since the Fundamental Diagram is only an aggregate of human behavior that hides individual variations and human inaccuracies that would naturally occur from an entity subconsciously assessing its

surrounding. Furthermore, the Gaussian filter is linearly separable, meaning that we can blur all the rows first using a 1D Gaussian, then blur the columns to obtain the final 2D Gaussian blur. This optimizes the computation of the convolution from a quadratic $O(n\text{-}m^2)$ computation to a more linear $O(n\text{-}m)$, where n is the number of pixels in space, and m is the radius of the filtering kernel.

4.4 Hardware-Acceleration via GPU Shaders

So far, we have implemented the PSM using a constrained Voronoi diagram over a discrete surface. The idea is to render every entity PS as a 3D cone viewed from the top, and the pixels visible after all cone intersections will represent the remaining available personal space. This utilization of the graphics pipeline allowed us to achieve interactive frame rates for thousands of entities in the scene [15]. In our attempt to accelerate the CPD's PSM computation, the CPU was initially found to be the primary bottleneck due to the repeated cone rendering calls made for each entity. Each render call comes with API overhead and CPU-to-GPU memory transfer costs. Modern graphics APIs have features that allow instanced rendering. The CPU would send the shape information only once, along with a point cloud of instance locations. Then, the GPU would perform the replication on chip without needing to communicate again with the CPU over the slow system bus. Unfortunately, this feature could not be naïvely used for PSM computation because of the potentially differing PS shapes, especially with the introduction of velocity-dependent elongation in direction of travel.

With nothing to "instance," we opted instead to develop Geometry Shaders that dynamically generate the PS shapes on the GPU. Geometry Shaders are part of the modern graphics processing pipeline that can procedurally generate new meshes and geometry that the CPU did not initially send. Our geometry shaders accept a point cloud of entity positions along with an array of entity attributes (e.g., current velocity, bearing, desired speed) and lets the GPU generate the appropriate voronoidal PS shapes per entity. This reduction in CPU calls has improved the frame rate. Furthermore, to compute each entity's new centroid position, we opted for a vertex shader (run once per entity, in parallel) that computes the available PS space (and the violated space, by omission) by sampling the previously created PSM (which was input into the vertex shader as a texture). This further resulted in performance gains that improved scalability and significantly reduced the bottlenecks at higher crowd counts (10,000+ entities in the scene).

5 EMERGENT CROWD DYNAMICS

In this section, we explore the global crowd dynamics that emerge from the locally defined agent-based rules of CPD. The simulations illustrate various scenarios of dense crowds and associated obstacles; our focus is on emergent crowd phenomena in high-density scenarios. We first describe the general simulation setup and then discuss each phenomenon. Although we already indicated that a detailed quantitative evaluation of the model is outside the scope of this research, we conducted detailed quantitative and qualitative analysis of the results presented in this section. We obtained a dataset included in [20] that was used for exhaustive studies of the simulation results. The dataset includes pedestrian motion real-world data from laboratory experiments; the detailed traces were used to validate all of the results of this research.

Our virtual world is defined as a 3D *scene* containing a flat *ground* on which the motion of *entities* (pedestrians) is simulated. They share their ground space with static scene *obstacles* (e.g., walls), dynamic elements (e.g., gates), and dynamic accessories (e.g., strollers and shopping carts). The invisible computational "backend" of the simulated world includes the PSM top-projection camera and the rendered PSM that it generates every timestep. Ideally, the ground shares the same aspect ratio as the PSM being computed on it to ensure isotropic computational fidelity regardless of

Fig. 12. Compression of personal space and petal-like emergent pattern.

which direction the pedestrians are facing in the scene. We maintain an agent-based approach, in which each entity encapsulates its own data and never has direct access to neighbor entities' data.

All simulations were run with discrete-time integration, using a quantum of 0.1 s per frame after finding minimal improvements in quality with a finer time delta. Each pixel side length represented 10 cm of physical space. For each scenario, we initialized entity positions, orientations, and randomized a few parameters such as weight, size, and the base desired speed (1.4 ± 0.24 m/s). The entity PS base radii were kept at 9 pixels (plus 1 center) to achieve the ∼0.8 to 1.0 m PS idle radius. The shaders described earlier were implemented using the OpenGL Shading Language (GLSL). Parameters were randomized across the crowd, including the PS radius, desired speed, and force parameters (α, β, and γ) which alters how aggressive or lenient an entity gets about restoring its own personal space and violating others'. Children were given the same PS radius as adults but rendered with weaker Voronoi cones (i.e., farther away from the PSM top view camera) to reflect the increased the chance of being overpowered by adult personal spaces or getting swept away by strong crowd flow in dense settings [10, 54]. The obstacles could be procedurally drawn during runtime or loaded from a bitmap (e.g., architectural floor plans). We focus on the emerging behavior producing four of such phenomena that are commonly studied:

—Lane formation in bidirectional flow
—Compression of personal space in areas of congestion and near areas of crowd interest
—Petal-like space filling
—Inverse correlation between an entity's surrounding density and its speed

The last effect is perhaps the easiest to reason about: in denser areas, a pedestrian tends to move slower. That relationship has been recorded, analyzed, and formalized, culminating in what is known as the Fundamental Diagram [45], a macroscopic measure of the crowd's density-speed profile.

Another phenomenon is the emergent petal-like space-filling pattern, in which each entity stands roughly behind the midpoint between the shoulders of two entities right in front of it (e.g., Figure 12, which shows a snapshot from an in-lab crowd-capture experiment [64], demonstrating both the compression of personal space near the congestion and the petal-like space-filling emergent pattern). Each entity optimizes the utility of the shared limited space [66], as there is less space compared with entities standing directly behind each other. As well, the entity has improved visibility of its target motion vector (i.e., it can better see where it is heading or has a better view of a point of interest, such as the stage at a concert). Figure 13 also demonstrates the compression of personal space near a congested doorway. This shows an in-lab trajectory capture experiment [47] in which two crowds of the same count are asked to evacuate through the narrow exits. The scenario on the left has no guiding barriers, while the scenario on the right utilized barriers to passively shape the crowd.

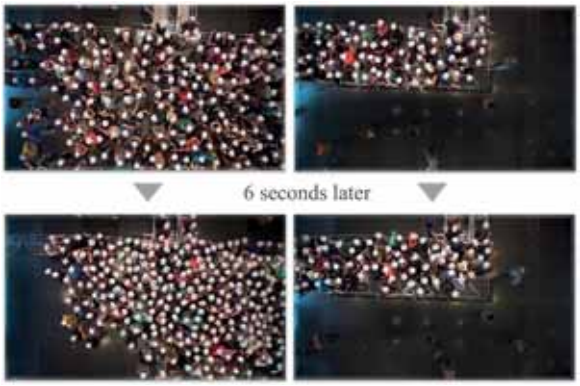

Fig. 13. Two crowd evacuations through narrow exits: without barriers and with barriers.

Existing microscopic crowd models struggle to reproduce these effects, as they model personal space as a 1D rigid separation distance, leading to artificial congestion and jamming, while captured crowd data such as what we see in these experiments indicate otherwise. CPD, instead, uses a compressible 2D area-based model of each entity's personal space; this local avoidance model can recreate such emergent compression effects in both dynamic and stationary crowds. A simple change in the design of passive barriers results in dramatically differing density profiles in a largely stationary crowd as it trickles through the narrow exit gates. This illustrates the accumulative nature of personal space compression and how restricting the possible angles of such accumulation can reduce the compression. Personal space compression is not just a matter of personality or personal preferences; the barriers in the environment also play a role. Note that at these densities, we are not considering psychological or personality-driven factors [49]. We are observing the nearly biomechanical collision avoidance response vectors that pedestrians tend to exhibit in such scenarios.

As discussed earlier, state-of-the-art methods such as RVO and ORCA, as well as Implicit Crowds, end up with incompressible artificial congestion that does not match how people tend to gradually concede their personal space to ensure a continuous motion. As these methods are based on analysis in the velocity space [20], they cannot reproduce compression in largely stationary crowds (e.g., near a concert stage or at a marathon start line). CPD provides an improved local collision avoidance model that reproduces emergent dense crowd phenomena, including lane formation in bidirectional flow, compression of personal space near congestion and near areas of crowd interest, which current microscopic methods struggle to reproduce correctly.

Other existing crowd simulation tools produce this kind of emergent behavior, but their profile is not adequate when compared with reference material, as will be shown in the next sections. CPD can produce queues and lane formation that match the reference source data, including thin snaking lanes with organic forks and joins (whereas existing methods cannot generate such behavior, and instead produce straight lines of pedestrians or lines clumped up, including lanes 3–4 people wide). State-of-the-art methods also must explicitly program these behaviors (which emerge naturally using CPD); thus, the computation stops being local and instead one must force region-based heuristics. In the case of CPD, there is no explicit programming of these behaviors, which are also parameterizable locally: we can adjust the profile for running/walking speeds as discussed earlier and can change the level of aggressiveness while keeping all computations local and not needing regional control for any of these. The computations are done at the level of each pedestrian and during the computation of their interactions with other close pedestrians. Existing

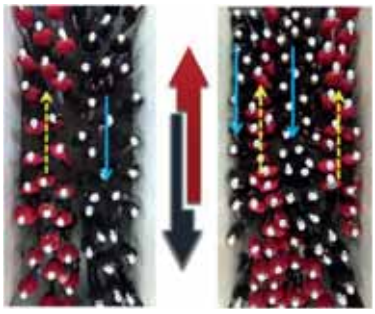

Fig. 14. Bidirectional flows: left—stable lanes emerging; right—unstable lanes.

tools normally use non-local modifications and stylistic control for achieving this kind of behavior, which emerges using CPD.

5.1 Bidirectional Flow

Bidirectional flow is a common scenario for hallways and corridors where pedestrians exhibit two dominant and opposing directions of motion. One of the well-studied observations of bidirectional flow is the emergent lane formation [66]. Lane formation is an innate pedestrian optimization strategy to minimize resistance due to oncoming traffic. The apparent "interlocking" pattern is born out of each entity's desire to take the path of least resistance. In bidirectional scenarios, this simply comes down to avoiding oncoming traffic. This is a macroscopic phenomenon that is emergent from microscopic local dynamics (i.e., there are no globally controlled explicit "laning rules").

The phenomenon of self-organized lane formation has been observed in bidirectional flows of real crowds in general and has also been confirmed in dense crowds. Figure 14 shows an in-lab study in which lanes are formed as a global phenomenon that emerges from local optimization decisions [64]. Lanes emerge from the collective motion of the crowd, with each entity trying to optimize its path, finding vectors of least resistance to its intended target, and avoiding head-on collisions with oncoming traffic. On the left, the crowd was not asked to seek a specific exit point along the width of the corridor. On the right, unstable lanes emerge when pedestrians explicitly seek a specific target exit. Existing simulation methods struggle to reproduce unstable lanes, as with each footstep. Instead, in CPD, each entity tries to minimize the deviation from its target path and the potential for collision with oncoming traffic. Following behind another entity that shares a similar direction of its motion will result in the least pathing disruption as they progress in the same general direction down the corridor. When this local agency ripples across the crowd, lanes start to emerge and form.

Figure 15 shows a bidirectional scenario in which our method is capable of reproducing lane formation using the symmetrical PS kernel. The figure shows a natural lane formation during a bidirectional flow simulation of 2,000 entities on a 600×800 PSM grid.

When we use the asymmetric PS kernel (recall that it considers the entity's vision cone, shifting the bulk of the collision avoidance response force to the entity with vision of the shared PS violation), the observed CPD emergent lanes exhibit less congestion near lane forking and branching spots. We also have a better resemblance to the lanes formed in real-life bidirectional flow. This is illustrated in Figure 16, which shows a top view of our simulation of bidirectional flow of a dense crowd (1,000 entities) in a wide corridor, in which we see similar branching/merging patterns to those observed (the bottom right shows a still frame of real footage [56] of bidirectional flow in

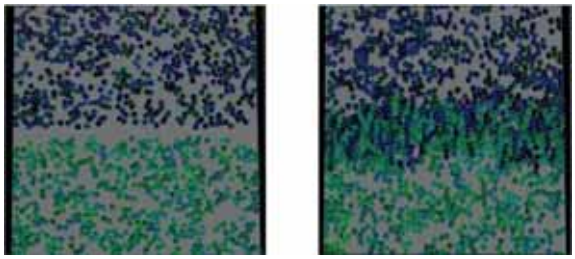

Fig. 15. Natural lane formation in bidirectional flow simulation (blue entities headed south).

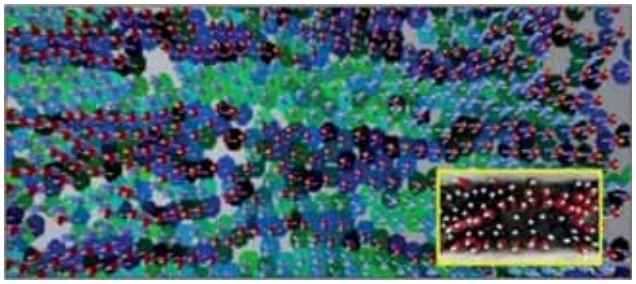

Fig. 16. Emergent lane formation produced by CPD pedestrians in a dense bidirectional flow scenario.

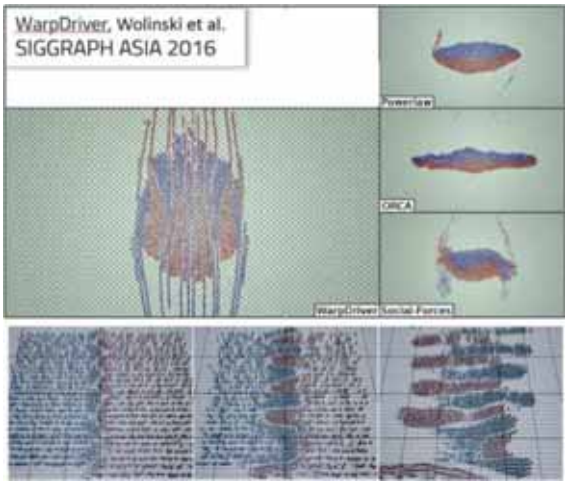

Fig. 17. Other real-time crowd simulation algorithms. Images captured from WarpDriver DOI: https://bit.ly/2R6w1tQ, [34] and https://bit.ly/2s6athT (*for academic use only).*

a corridor). The entities in both our simulation and the real footage are color-coded to indicate direction of motion (East-West).

Figure 17 shows how the state-of-the-art methods (WarpDriver [59], ORCA [50], RVO [3], Social Forces [14], and position-based [58] crowds) that share our application target of real-time crowd simulation reproduce bidirectional flow. When compared with our CPD pedestrians discussed earlier, those methods either fail or struggle to reproduce believable lane formation in dense crowds.

Position-based crowds is another recent method that struggles with reproducing organic laning—their motion trajectories are either too aggregated or suffer from excessive artificial

Fig. 18. (a) Pedestrian footage in a narrow bidirectional corridor. (b) Trace of trajectories [66]. (c) Simulated trajectories with CPD.

congestion. When considering the average lane width, CPD was able to reproduce thin and piercing lanes that exhibit branch-and-merge patterns throughout the simulation timeline. WarpDriver can reproduce thin lanes but they appear too axis-aligned (or "straight") as opposed to the real footage's more organic and curved branch-and-merge patterns. Social Forces, ORCA, and Power Law fail to exhibit any laning at such high densities due to their reliance on a rigid 1D uncompressible separation distance, resulting in overly collision-averse congested areas rather than smooth lane formation that our compressible 2D PS model can reproduce. Position-based crowds succeed in forming dense lanes, but they are unrealistically wide and eventually clump into congested and aggregated motion patterns, whereas groups of entities appear to move precisely in unison during lane formation.

There is a lack of agreed on metrics to measure that statement quantitatively. Cross-sectional velocity distribution charts [44] and average velocity grids [6] have been previously proposed to quantify the flow. However, in our view, they aggregate away many of the microscopic details of the flow dynamics that motivate us. Instead, we propose to do a cursory analysis of the observed quality of the emergent lanes compared with live footage. Then, we examine the microscopic trajectory traces, which paint a richer picture of flow dynamics. The quality of the emergent bidirectional lanes can also be assessed over a period of simulation. Diagrams that trace the trajectory of each pedestrian can capture an overall snapshot of such dynamic patterns over time.

Figure 18 shows a trace of the trajectory of a real bidirectional flow scenario in a 3.6-m-wide corridor obtained from [66]. The participants in this experiment were randomly assigned an x-axis target on the other end of the corridor, leading to emergence of lanes, as apparent from the trajectory trace, and are termed *unstable* as they vary spatially and temporally. Larger-sized experiments are difficult to construct and coordinate (e.g., this experiment required over 300 volunteers). Simulation, instead, is a useful tool here.

The trace in the figure shows how our model was able to replicate the emergent lane formation, appearing nearly congruent to the observed trace. We were able to reproduce better laning behavior when compared with state-of-the-art tools. This was not only done by observation—we also obtained the detailed traces of the original experiments and verified the similarities between them. Given further rigorous statistical validation, our method could facilitate urban design and safety-critical planning for large-scale crowded events [29]. As the method deals with local interactions, our analysis focuses on the study of local results, as discussed in this section.

As demonstrated in lab experiments on bidirectional flow [20], lanes emerge from the collective motion of the crowd, with each entity trying to optimize its path, finding vectors of least resistance to its intended target and avoiding head-on collisions with oncoming traffic. Our method mimics

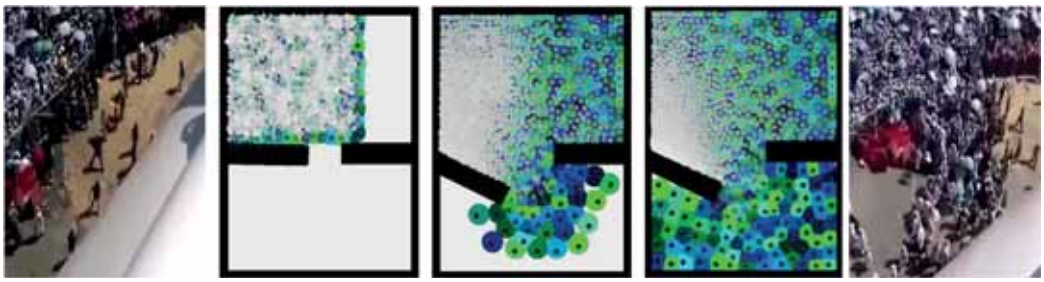

Fig. 19. Penalty forces in action in an overcrowded area.

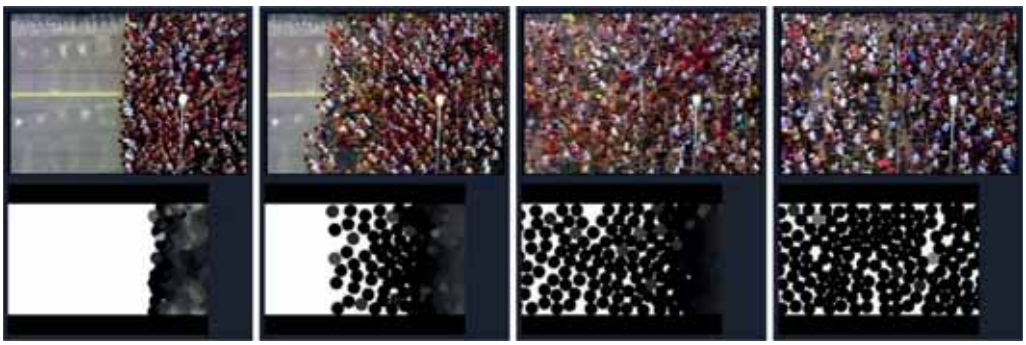

Fig. 20. Gradual release of density-dependent velocity. Snapshots of a marathon start and CPD.

these behaviors with precision. We used the information in the dataset to study the correctness of the simulation results generated by the model, discussed in this section.

5.2 Local Density and Personal Space

In sparse scenarios, the centroidal response force acts like a microscopic method whose goal is to avoid collision with nearby entities. In denser cases, where personal space compressibility is observed, it starts displaying macroscopic qualities that mimic waves and energy propagation among the crowd's individuals. At less than 3 ft^2 of personal space per entity, the crowd reaches a critical density [23], where entities are subjected to enough pressure to cause significant discomfort and injury, with potential injury at <2 ft^2. This is a concern in large-event contingency planning; simulation can help identify pockets of potentially unsafe accumulation and overcrowding of attendees [10]. Figure 19 shows such behavior in a virtual crowd of 1,000 entities in a confined area. We can see how members of the crowd react to the fact that they are overcrowded. Penalty forces are enough to cause an overcrowded room to diffuse to a more comfortable equilibrium.

Requiring neither a cognitive model nor a global pathing scheme, the centroidal response force can restore the compressed crowd to a more comfortable distribution naturally and gradually, filling the room if necessary, until every entity reaches a suitable local density. This can be seen in real life when crowds are held behind barriers and a gate opens, allowing the otherwise compressed crowd to immediately diffuse through, fan out, and reclaim their personal space. Those on the outer edge of the crowd have less density to deal with, so they have the greatest freedom to accelerate to higher speeds compared with those that are relatively slowed or still trapped in higher densities.

This same effect can be observed at the beginning of a marathon. As shown in Figure 20, the wave of delayed acceleration is the result of those at the front of the race having the advantage of lower density ahead of them, allowing them to sprint ahead sooner while those behind are

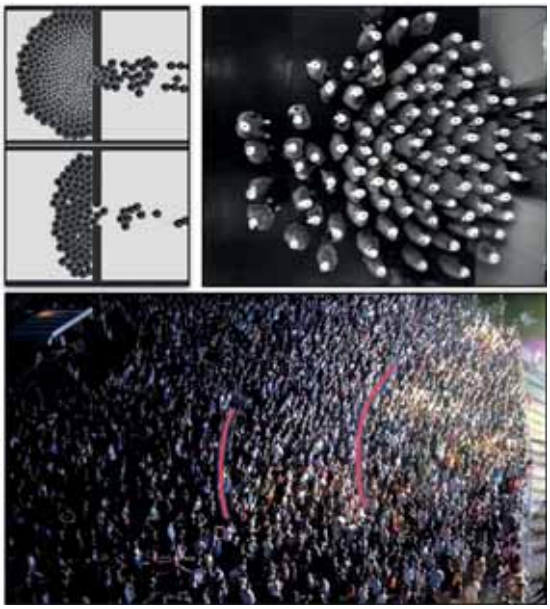

Fig. 21. Arching patterns.

stuck to less competitive speeds until the wave catches up to them eventually and the speeds equalize. Unlike the compressed room in Figure 19, the marathon scenario has a global path vector along the marathon track applied evenly to all contestants. However, the centroidal response force reproduces the observed emergent acceleration wave effect.

5.3 Observable Patterns in Stationary Crowds

Figure 21 illustrates the phenomenon of personal space compression in largely stationary crowds. The figure shows a bottleneck arching pattern (top left), which displays arching, gradual PS compression, and petal-like formations observed in crowds during egress [56] (top right) and while stationary at a concert (bottom). This is a challenging effect to simulate in many state-of-the-art methods because of their reliance on relative velocities and optimization in velocity space, which would have a difficult time distinguishing between static entities at varying distances from a barrier or an area of collective interest. Our area-based force can accumulate the compression, through time-iterative energy transfer, in high-density stationary crowds. This effect aligns with observed PS compression in both moving and static crowds in Section 5.2. In addition to arching, huddled crowds tend to display petal-like formations (as each entity attempts to be situated behind the midpoint of the two entities ahead). That increases the entity's visibility of the point of interest (or global path destination), and results in an overall more compact space filling, as it attempts to equalize the number of neighbors surrounding each entity. The PSM can be directly shown as an underlay to visualize the personal spaces used for centroidal force calculations. It also provides local density visualization. The figure shows a common emergent crowd phenomenon (arching around pathway bottlenecks), with noticeable compression of personal space near a narrow exit.

5.4 Simulation Performance

CPD design is adequate to be computationally parallel friendly; thus, the method can be used in parallel computing devices. The results show interactive frame rates with thousands of entities in

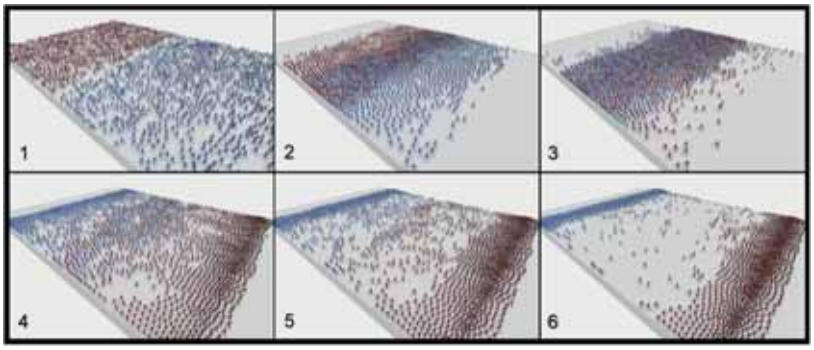

Fig. 22. Artificial scenario used as a case study. Entities are color-coded by motion direction.

Table 1. Simulation Performance in Frames per Second (fps)

	CPU&CPU Non-instanced Rendering			**GPU&CPU Instanced Rendering GPU Shaders**	
# Entities	Desktop GTX1060	Laptop Core i5	Nexus 6P (Android)	Desktop GTX1060	Speedup (compared with CPU&CPU)
100	160	105	30	450	2.8×
250	125	87	23	3	2.7×
500	90	62	18	266	3.0×
1,000	58	41	13	134	2.3×
1,500	44	31	10	121	2.8×
2,000	33	26	7	89	2.7×
5,000	16	12	3	47	2.8×
10,000	10	7	-	25	2.3×
15,000	7	4	-	20	2.6×
20,000	6	3	-	14	2.3×

the scene, as discussed in this section. The main metric of interest is the average frame rate. The case study presented in Figure 22 uses our model to simulate bidirectional flow over a 600 × 900 PSM (effectively, a 60-m corridor) while varying the number of pedestrians in the scene.

The case study shows how the algorithm can provide effective animation features (expressed as the number of frames per second, which gives an accurate idea of the quality of visualization for serious games and entertainment). The experiment includes many active entities; all of the individuals in the scene are moving (in contrast to more static scenarios, for instance, those caused by bottlenecks); this provides an opportunity to study a pedestrian scenario with a high level of activity.

Table 1 shows the results obtained for this scenario using three representative consumer-grade devices: a mid-range desktop computer (4GHz Quadcore CPU + NVIDIA GTX970 GPU), a laptop computer (Intel Core i5 with integrated HD Graphics 4000), and a smartphone (Nexus 6P with Qualcomm Adreno 430 GPU).

Each simulation cycle involves two major phases: PSM construction (i.e., cone rendering) and per-entity forces computation (centroidal, global, net, etc.). Our engine, used in [15, 16], was built on Processing 3.0 (a Java-based graphics framework) and supported two computation modes: (a) CPU&CPU: CPU OpenGL call per cone render; CPU-computed forces; and (b) GPU&CPU: GPU

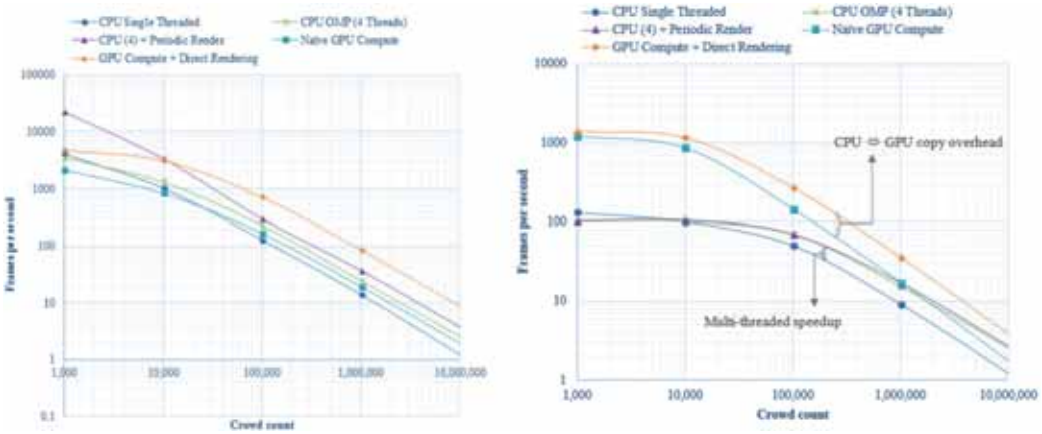

Fig. 23. (a) Log chart of PSM (i.e., PS cone rendering). (b) Log chart of full CPD cycle.

instancing to concurrently render all cones; CPU-computed forces. While the Android device was capable of simulating higher crowd counts, it was no longer able to do so at interactive frame rates (i.e., 8 fps or higher). The performance gain from implementing the GPU shaders for instanced cone rendering is noticeable, at ∼2.6× throughout. Given the 100-ms time quantum, every 10 frames represented 1 second of simulation time. Hence, this Java implementation produced faster-than-real-time simulation for up to 20,000 entities in the scene and maintained reasonably interactive frame rates for even higher counts. Later iterations of the engine were developed using C++17 and OpenGL 4.3, removing the dependency on Processing and on a Java Virtual Machine (JVM) at runtime. For this engine, the PSM construction phase (i.e., cone rendering) is always GPU-instanced. However, the force computation phase can be executed in a variety of ways. Force computation per entity involves querying the local PSM area for PS violations, computing the response centroidal force(s), and advecting positions.

Figure 23 charts CPD's computational performance. We use CPD++ (Kaveri QuadCore+GTX970 with /O2 compile) and a PSM: 600900. The figures show the computational performance in frames per second, plotted against the scene's crowd count.

The C++ engine implements from the algorithm in Section 3 several computation options or modes:

- CPU Single Threaded: The global PSM is fetched from the GPU into RAM, and a CPU loop iterates over every entity to compute its forces and advect its position. New positions are sent back to the GPU (visualization) and made ready for the next frame's computation.
- CPU OMP (4 Threads): A multi-threaded implementation utilizing OpenMP 2.0. Once the PSM is fetched from the GPU into RAM, OpenMP splits the workload over the available CPU threads. In this case, we tested a quad-core CPU; hence, we utilized 4 threads. Lastly, the new positions are sent back to the GPU, as done in the single-threaded mode.
- CPU (4) + Periodic Render: Rendering at higher frequencies than supported by the screen is essentially wasteful. Hence, we use the same OpenMP multi-threaded computation, but only perform the visualization step at courser timesteps. For example, if you can compute forces at 2,000 fps, visualization on a 60-Hz monitor will be performed every ∼16 ms.
- Naïve GPU Compute: After the PSM computation, an OpenGL 4.3 compute shader containing the force computation logic is dispatched to the GPU. In this case, instead of iterating over each entity in a loop, the shader launches a compute thread per entity.

- GPU-Compute + Direct Rendering: In this mode, the PSM construction shader (i.e., cone rendering) and the forces compute shader share access to the same buffer of entity data, eliminating all GPU-to-CPU and GPU-to-GPU copies entirely, including the initial fetch of the PSM to the CPU, because the CPU is not involved in the force computation at all.

Of interest to note from the figures is that the overhead of launching multiple threads in the CPU OMP Compute mode is too expensive compared with the single-threaded CPU-compute option for relatively small crowd counts (less than 10,000 entities). But the speedup due to CPU multi-threading does improve with higher crowd counts. Periodic rendering improves the multi-threaded CPU mode's performance in the PSM computation phase but not in the overall CPD computation. This typically indicates that the PSM computation is not the bottleneck in the overall CPD algorithm (a pleasant by-product of formulating neighborhood queries as a GPU-accelerated cone rendering problem). Instead, it appears that the centroidal force computation is the primary overall bottleneck and the GPU modes provide a clear performance speedup in the CPD's overall computations.

With zero-copy direct rendering enabled, the GPU can provide approximately 2× speedup compared to the Naïve GPU option. The speedup is the result of removing the copying of crowd data between the CPU and the GPU through the system bus. At the higher end of workloads, it seems that the GPU's 4-GB memory (VRAM) limitations become a limiting factor where the crowd computation must be dispatched to the GPU in multiples of 4 GB compared with the CPU's native access to 32-GB RAM and more through an SSD-backed virtual memory space.

Regarding scalability, across this engine's modes, the computational costs appear negatively linear in the log-log scale, with the caveat being that crowd counts exceeding a system's available memory (e.g., RAM or VRAM) would perform the force update over multiple passes and might require PSM tiling. The artificial workloads presented of up to 10 million entities were successfully simulated on a machine with 32 GB RAM and 4 GB VRAM (GPU memory).

The resolution of the PSM in pixels can be adjusted to reach higher performance at the cost of reduced fidelity. Ideally, performance would depend on the crowd count rather than PSM resolution. We believe that the current overhead of essentially simulating empty PSM spaces can be overcome (or hidden) by utilizing multi-threaded CPU rendering calls, as will be possible in upcoming low-overhead graphics API standards such as Vulkan [42], the direct successor of OpenGL. For reference, ORCA could simulate 5,000 agents at 140 fps, while WarpDriver simulated 5,000 agents in real time (15–20 fps @ 50 ms timestep) [50, 59], and Continuum Crowds ran 10,000 agents at 50 fps [53].

6 CONCLUSION

We presented **centroidal particle dynamics (CPD)**, an agent-based short-range collision avoidance model for pedestrians in dense crowds. We showed our model's ability to reproduce emergent phenomena that show congruence to real pedestrian trajectory data and explained our performant implementations for simulating high-density crowds. To further the trust in CPD for use in critical and safety-oriented applications, the model's parameters will require further calibration, incorporating data-validation methods to enhance trust in the CPD model.

Our explicit 2D approach to modeling personal space meant that it can be edited and modified visually and intuitively (e.g., culling the front of a PS cone for pedestrians distracted on cellphones). Additionally, the PSM computation allows for arbitrary shapes, affording high flexibility of scene walls, obstacles, and barrier designs, a favorable property when simulating crowd motion in architectural and urban design contexts. The inherent compressibility of our PS model meant that it accommodates dense scenarios correctly as opposed to existing methods that treat PS as a rigid 1D separation distance, leading to artificial congestion and unnecessary clogging of pathways.

While this lightweight force can produce a variety of visually convincing emergent crowd behavior on its own, it is equally suitable for integration with existing particle-based methods if desired. For more serious applications that rely on data-driven calibration and rigorous accuracy requirements, the idea of encoding local interactions as geometric primitives requires further study. A worthy pursuit is the geometric encoding of velocity-space collision avoidance schemes, such as RVO optimization, which allow for calibration and the accurate reproduction of kinetic trajectories of microscopic pedestrian interactions. The challenge in describing the RVO scheme geometrically is in its empirically motivated assumption that collision avoidance is a shared effort between nearby entities, thus requiring that entities know about and share information with their nearest neighbors, something we have been actively avoiding in our performance-minded GPU implementation. The method leads to parallel computation, opening the floor to future work that could include detailed overhead analysis, improved calculations for large densities, and explicit measuring of the different components used in calculating tessellation. The method does not compete with traditional methods such as RVO or social forces; instead, it could be combined with them. A classic tool can be used for path planning, long-range collision avoidance, and CPD can be used for handling personal space forces. Our design would allow such combinations, as we deal with local interactions.

Civil safety and threat assessment applications stand to benefit the most from dense-crowd research. Although our method uses empirically driven parameters to produce visually convincing aggregate behavior, it cannot yet be reliably used for safety-critical applications. That would require validation against in-lab scenarios [30, 65, 66] and statistical analysis. There are global statistical properties that can be checked (e.g., governing distributions [37, 38]) and local similarity indices for targeted analysis of smaller areas of interest. We echo our earlier assertion that regardless of which method is used, crowd simulation is essentially an exercise in abstraction with no "ground truth" to converge on, yet the increase in accuracy is a worthwhile pursuit, considering the potential applications. A particularly challenging and motivating use case is the prevention of crowd stampedes and crushes.

Could our proposed close-range PS model complement those methods by allowing them to have compression of personal space? Perhaps so—that would be a useful avenue to investigate and experiment with (e.g., RVO vs. RVO+CPD). This possibility of integrating CPD into existing crowd methods was part of its design from the beginning, in which we focused our attention on addressing the close-range avoidance dynamics first, in a localized agent-based manner, and with high-performance implementations in mind that leave room for other components of the crowd path planning model (e.g., medium-range collision avoidance via RVO).

REFERENCES

[1] A. Abdelghany, H. Mahmassani, K. Abdelghany, H. Al-Ahmadi, and W. Alhalabi. 2018. Incidents in high-volume elongated crowd facilities: A simulation-based study. *SIMULATION Trans of SCS* 95, 9 (2018), 823–843

[2] S. Bandini, S. Manzoni, and G. Vizzari. 2006. Crowd modeling and simulation. In *Innovations in Design & Decision Support Systems in Architecture and Urban Planning*. Springer, 105–120.

[3] J. van den Berg, M. Lin, and D. Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. *IEEE Intl Conf on Robotics and Automation*, Pasadena, CA.

[4] G. Van Den Bergen and D. Gregorius. 2010. *Game Physics Pearls*. CRC Press.

[5] D. C. Brogan, R. A. Metoyer, and J. K. Hodgins. 1998. Dynamically simulated characters in virtual environments. *IEEE Comput Graph Appl* 18, 5 (1998), 58–69.

[6] S. Chandra and A. Kumar Bharti. 2013. Speed distribution curves for pedestrians during walking and crossing. *Procedia—Soc Behav Sci* 104, 660–667.

[7] U. Chattaraj, A. Seyfried, and P. Chakroborty. 2009. Comparison of pedestrian fundamental diagram across cultures. *Adv Complex Syst* 12, 03 (2009), 393–405.

[8] D. C. Duives, W. Daamen, and S. P. Hoogendoorn. 2013. State-of-the-art crowd motion simulation models. *Transp Res Part C Emerg Tech* 37, 193–209.

[9] C. Eastman, J. K. Lee, Y. S. Jeong, I. Anthony, R. Li, I. Li, A. Dey, J. Forlizzi, S. Railsback, S. Lytinen, S. Jackson, R. Volk, J. Stengel, F. Schultmann, H. Lee, H. Oh, Y. Kim, T. G. Kim, M. Gatti, P. Cavalin, S. B. Neto, C. Pinhanez, C. Santos, D. Gribel, and A. Appel. 2015. Simulating human behavior in not-yet built environments by means of event-based narratives. *Aut in Constr. v.* 38 (2015), 109–127.

[10] J. Fruin. 1993. The causes and prevention of crowd disasters. In *1st International Conference on Engineering for Crowd Safety*. London, UK.

[11] O. De Gyves, L. Toledo, and I. Rudomín. 2013. Proximity queries for crowd simulation using truncated Voronoi diagrams. *Proc of Motion on Games* 65, 87–92.

[12] A. Al-Habashna and G. Wainer. 2016. Modeling pedestrian behavior with Cell-DEVS: Theory and applications. *SIMULATION* 92, 2 (2016), 117–139.

[13] E. T. Hall. 1990. *The Hidden Dimension*. Penguin Random House.

[14] D. Helbing, I. Farkas, and T. Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803:5, 487–490.

[15] O. Hesham and G. Wainer. 2021. Advanced models for centroidal particle dynamics: Short-range collision avoidance in dense crowds. *SIMULATION, Transactions of SCS*. First published April 16, 2021. https://doi.org/10.1177/00375497211003126.

[16] O. Hesham and G. Wainer. 2017. Context-sensitive personal space for dense crowd simulation. In *Proceedings of the Symposium for Architecture and Urban Design (SimAUD'17), Toronto, ON.*

[17] K. Hoff, J. Keyser, M. Lin, D. Manocha, and T. Culver. 1999. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of the 26th Conference on Computer Graphics and Interactive Techniques—SIGGRAPH '99*. Los Angeles, CA.

[18] S. Huerre, J. Lee, M. Lin, and C. O'Sullivan. 2010. Simulating believable crowd and group behaviors. *ACM SIGGRAPH ASIA 2010 Courses, Seoul, Korea.* 13:1–13:92.

[19] R. Hughes. 2003. The flow of human crowds. *Ann Rev Fluid Mech* 35, 1 (2003), 169–182.

[20] I. Karamouzas, B. Skinner, and S. J. Guy. 2014. Universal power law governing pedestrian interactions. *Phys Rev Lett* 113, 23 (2014), 238701.

[21] T. Kisko, R. Francis, and C. Nobel. 1998. *Evacnet4 User's Guide*. University of Florida.

[22] S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans Inf Theory* 28, 2 (1982), 129–137.

[23] Federal Emergency Management Agency. 2010. Job Aids Manual. Special events contingency planning. Apr. 2010.

[24] L. Luo, C. Chai, J. Ma, S. Zhou, and W. Cai. 2018. ProactiveCrowd: Modeling proactive steering behaviours for agent-based crowd simulation. *Comp Graphics Forum* 37, 1 (2018), 375–[8.

[25] F. Martinez-Gil, M. Lozano, and F. Fernández. 2014. MARL-Ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups. *Simul Model Pract Th* 47, 259–275.

[26] F. Martinez-Gil, M. Lozano, and F. Fernández. 2012. Calibrating a motion model based on reinforcement learning for pedestrian simulation. In Kallmann M., Bekris K. (eds.), *Motion in Games (Lecture Notes in Computer Science)*, Vol. 7660, Springer, Berlin.

[27] Mitsubishi. 2016. *Real-time crowd-congestion estimation system manual*. Mitsubishi Electric Corporation Information Technology R&D Center.

[28] B. Mohler, W. Thompson, S. Creem-Regehr, H. Pick, and W. Warren. 2007. Visual flow influences gait transition speed and preferred walking speed. *Exp Brain Res* 181, 2 (2007), 221–228.

[29] M. Moussaïd, D. Helbing, and G. Theraulaz. 2011. How simple rules determine pedestrian behavior and crowd disasters. *Proc Natl Acad Sci* 108, 17 (2011), 6884–6888.

[30] S. Narang, A. Best, S. Curtis, and D. Manocha. 2015. Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. *PLoS One.* 10, 4 (2015).

[31] R. Narain, A. Golas, S. Curtis, and M. C. Lin. 2009. Aggregate dynamics for dense crowd simulation. In *ACM SIGGRAPH Asia 2009*, Yokohama, Japan.

[32] J. Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann, 1994.

[33] J. Ondřej, J. Pettré, A. H. Olivier, and S. Donikian. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Trans Graph* 29, 4 (2010), 123, 1–9.

[34] N. Pelechano, J. Allbeck, and N. Badler. 2007. Controlling individual agents in high-density crowd simulation. *SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA.

[35] F. I. Pelupessy, W. E. Schaap, and R. van de Weygaert. 2003. Density estimators in particle hydrodynamics. *Astron Astrophys* 403, 2 (2003), 389–398.

[36] J. Pettré, H. Grillon, and D. Thalmann. 2007. Crowds of moving objects: Navigation planning and simulation. *IEEE International Conference on Robotics and Automation*. Rome, Italy.

[37] J. Pettré, D. Wolinski, and A. H. Olivier. 2012. Velocity-based models for crowd simulation. In *Pedestrian and Evacuation Dynamics*. 1065–1078.

[38] J. Pettré, J. Ondřej, A. H. Olivier, A. Cretual, and S. Donikian. 2009. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'09)*. New Orleans, LA.
[39] J. Gillian. 2017. "Inaugural crowd sizes ranked | PolitiFact," *Politifact*, 2017.
[40] Y. Qu, Y. Xiao, J. Wu, T. Tang, and Z. Gao. 2018. Modeling detour behavior of pedestrian dynamics under different conditions. *Phys A Stat Mech Its Appl* 492, 1153–1167.
[41] J. Xie, K. Chen, T. Kwan, and Q. Yao. 2020. Numerical simulation of the fire emergency evacuation for a metro platform accident. *SIMULATION, Trans of SCS* 97, 1 (2020), 19–32.
[42] G. Rong and T. Tan. 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *Symposium on Interactive 3D Graphics and Games*, Boston, MA.
[43] I. Rudomin, B. Hernández, O. De Gyves, L. Toledo, I. Rivalcoba, and S. Ruiz. 2013. GPU generation of large varied animated crowds. *Comput y Sist* 17, 3 (2013), 365–380.
[44] M. Saberi, K. Aghabayk, and A. Sobhani. 2015. Spatial fluctuations of pedestrian velocities in bidirectional streams: Exploring the effects of self-organization. *Phys A: Stat Mech.* 434, 120–128.
[45] A. Seyfried, B. Steffen, W. Klingsch, T. Lippert, and M. Boltes. 2005. *Steps toward the fundamental diagram — empirical results and modelling. Pedestrian Evacuation Dynamics.* 377–390.
[46] A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes. 2005. The fundamental diagram of pedestrian movement revisited. *J Stat Mech Theory Exp* (2005) P10002.
[47] X. Shi, Z. Ye, N. Shiwakoti, and Z. Li. 2015. A review of experimental studies on complex pedestrian movement behaviors. In *Proceedings of the 15th COTA International Conference of Transportation Professionals*, Beijing, China.
[48] J. Shiraef. 2016. *An exploratory study of high performance graphics application programming interfaces. Master's Thesis.* University of Tennessee at Chattanooga.
[49] A. Sieben, J. Schumann, and A. Seyfried. 2017. Collective phenomena in crowds - where pedestrian dynamics need social psychology. *PLoS One.* 12, 6 (2017), 1–19.
[50] J. Snape, S. J. Guy, D. Vembar, A. Lake, and M. Lin. 2012. Reciprocal collision avoidance and navigation for video games. In *Game Developers Conference*, San Francisco, CA.
[51] R. Sommer. 2008. *Personal Space; Updated, the behavioral basis of design.* Bosko Books.
[52] S. A. Tashrifullahi and M. A. Hassanain. 2013. A simulation model for emergency evacuation time of a library facility using EVACNET4. *Struct Surv* 31, 2 (2013), 75–92.
[53] A. Treuille, S. Cooper, and Z. Popović. 2006. Continuum crowds. *ACM Trans Graph* 25, 3 1160–1168.
[54] S. A. Turris, A. Lund, and R. R. Bowles. 2014. An analysis of mass casualty incidents in the setting of mass gatherings and special events. *Disaster Med Public Health Prep* 8, 2 (2014), 143–149.
[55] M. Twarogowska, P. Goatin, and R. Duvigneau. 2014. Macroscopic modeling and simulations of room evacuation. *Appl Math Model.* 38, 24 (2014), 5781–5795.
[56] H. Wang, J. Ondřej, and C. O'Sullivan. 2017. Trending paths: A new semantic-level metric for comparing simulated and real crowd data. *IEEE Trans on Vis and Comp Gr* 23, 5 (2017), 1454–1464.
[57] S. Wang, M. Van Schyndel, G. Wainer, and V. Rajus. 2012. DEVS-based building information modeling and simulation for emergency evacuation. In *Winter Simulation Conference*. Berlin, Germany.
[58] T. Weiss, C. Jiang, A. Litteneker, and D. Terzopoulos. 2017. Position-based multi-agent dynamics for real-time crowd simulation. In *Proceedings of the 10th International Conference on Motion in Games*, Barcelona, Spain.
[59] D. Wolinski, M. C. Lin, and J. Pettré. 2016. WarpDriver: Context-aware probabilistic motion prediction for crowd simulation. *ACM Trans Graph* 35, 6 (2016), 1–11.
[60] D. Wolinski, S. Guy, A. H. Olivier, M. C. Lin, D. Manocha, and J. Pettré. 2014. Parameter estimation and comparative evaluation of crowd simulations. *Computer Graphics Forum* 33, 303–312.
[61] Y. Xiao, Z. Gao, Y. Qu, and X. Li. 2016. A pedestrian flow model considering the impact of local density: Voronoi diagram based heuristics approach. *Transp Res Part C Emerg Tech* 68, 566–580.
[62] L. Yilmaz, S. J. E. Taylor, R. Fujimoto, and F. Darema. 2014. Panel: The future of research in modeling & simulation. *Proceedings of the 2014 Winter Simulation Conference*, Savannah, GA.
[63] W. Zeng and R. L. Church. 2009. Finding shortest paths on real road networks: The case for A*. *Int J Geogr Inf Sci* 23, 4 (2009), 531–543.
[64] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried. 2011. Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions. *J Stat Mech Theory Exp.* 2011 (2011), P06004.
[65] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried. 2012. Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. *J Stat Mech Theory Exp*, 2 (2012), P02002.
[66] J. Zhang and A. Seyfried. 2013. Empirical characteristics of different types of pedestrian streams. *Procedia Eng.* 62 (2013), 655–662.

Received January 2019; revised February 2021; accepted March 2021