Context-sensitive Personal Space for Dense Crowd Simulation

Omar Hesham and Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
omar.hesham@carleton.ca, gwainer@sce.carleton.ca

ABSTRACT

Real-time simulation of dense crowds is finding increased use in event planning, congestion prediction, and threat assessment. Existing particle-based methods assume and aim for collision-free trajectories. That is an ideal -yet not overly realistic- expectation, as near-collisions increase in dense and rushed settings compared to typically sparse pedestrian scenarios. This paper presents a method that evaluates the immediate personal space area surrounding each entity to inform its pathing decisions. While personal spaces have traditionally been modeled as having fixed radii, they actually often change in response to the surrounding context. For instance, in cases of congestion, entities tend to share more of their personal space than they normally would, simply out of necessity (e.g. leaving a concert or boarding a train). Likewise, entities travelling at higher speeds (e.g. strolling, running) tend to expect a larger area ahead of them to be their personal space. We illustrate how our agent-based method for local dynamics can reproduce several key emergent dense crowd phenomena; and how it can be efficiently computed on consumer-grade graphics (GPU) hardware, achieving interactive frame rates for simulating thousands of crowd entities in the scene.

Author Keywords

Crowd animation; Personal space; GPGPU; Interactive.

ACM Classification Keywords

I.6.5 [Simulation and Modeling]: Model Development;
I.3.7 [Computer Graphics]: Animation;

1 INTRODUCTION

Dense crowd simulation is an area of research concerned with assessing and predicting the motion of large groups of people within a limited physical space. The applications range from use in gaming and film production, to designing public spaces and assessing quality of occupancy, to the safety-critical analysis of the potential for stampedes and crowd crushes.

If you ask two people to walk the same path, they will display deviations from each other. Ask the same person to walk the same path twice, and you would still get deviations from one walk to another. Human motion is seemingly non-deterministic, and hence, pedestrian simulation will always be an exercise in abstraction.

When simulating high-density pedestrian traffic, congestion, or a mass gathering event (e.g. at a concert), macroscopic methods that rely on aggregate parameters (bringing a sense of determinism through bounded stochasticity) can be very effective when analyzing collective motion results, such as rate of egress and density distribution over an area. Because they do not rely on simulating individual entities, those methods are often efficient enough to accommodate large-scale simulations of thousands of entities. By contrast, microscopic methods can reproduce the intricate details of every individual's trajectory, at an increased computational cost. However, as computing hardware continues to evolve to provide performance gains through parallelism and portability through power efficiency, microscopic simulation is becoming increasingly accessible to designers, architects, and event planners, allowing them to readily assess the risks and focus stakeholder efforts around potential congestion issues.

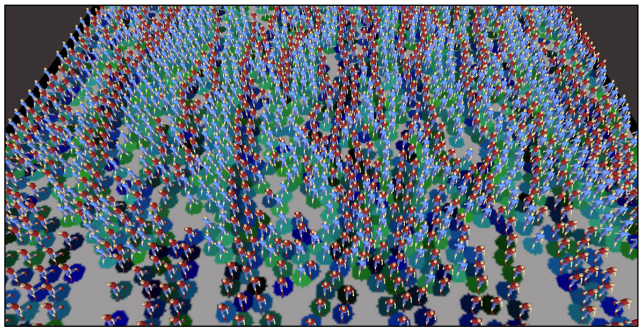

Figure 1. Organically formed lanes in dense bidirectional flow; 2000 entities simulated at 130fps on a consumer-grade desktop.

Here, we present a contribution to the development of micro-scopic methods catered to dense crowd phenomenon (e.g. Figure 1). The relevant background is discussed in sections 2 and 3. Section 4 outlines our contributions and section 5 illustrates our method's results and real-time performance. Finally, the discussion in section 6 reflects on the method's limitations, and hosts opportunities for further discussion and future work.

2 BACKGROUND

Like other physically-based phenomena that exhibit complex interactions between entities over time, crowd motion could only be practically simulated using numerical methods rather than analytical solutions. There are different granularities of motion abstraction depending on the target use case.

SimAUD 2017 May 22-24 Toronto, Canada

Historically, the earliest methods were macroscopic in nature, simulating aggregate behavioural parameters, rather than actual individual trajectories in the scene. They were based on adapting existing fluid simulation models to incorporate aggregate human motion parameters. They were typically computed over an Eulerian grid [20, 25] to provide computational stability and high performance. This granularity of simulation was sufficient to assess and validate collective motion parameters such as egress rate and density distribution over a given scene layout.

Network optimization techniques were also adapted to simulate occupant movement within a predefined multi-compartment environment [15]. Each compartment is treated as a graph node that might represent a section within a room, a hallway, or even an entire building. The edges connecting those nodes would represent the capacity of pedestrians moving between one node to another. By utilizing classic optimization techniques such as finding the shortest path or detecting the max flow, designers could focus their efforts on areas of potential pedestrian bottlenecks better.

To this day, macroscopic crowd methods remain popular in engineering and design applications due to their computational efficiency and the ability to provide a great deal of insight into aggregate crowd dynamics, especially for large-scale projects which involve high crowd counts [6, 13].

With ever-increasing hardware capabilities and improved modeling methodology, the ability to simulate individual entity-to-entity interactions became possible. Unlike macroscopic methods, microscopic methods simulate entities as individual agents with localized rulesets whose emergent behaviour matches that of the aggregate results of macroscopic methods (and more importantly, reality). In microscopic methods, local-neighbourhood interaction rules can have significant effects on the emergent global behaviour.

Some of the earliest examples of this modeling philosophy include Cellular Automata (CA) and the closely related Lattice Boltzmann (LBM) models [1]. In CA methods, the space is typically divided into a uniform grid, where every cell can either be available, have an entity, or represent an obstacle. Every cell's future state is then determined based on the states of the cells in its local neighbourhood. CA crowd models were rapidly developed and adopted, thanks to their parallel-friendly processing and native visualization (every cell is both the computational unit and the visual representation). Nevertheless, this grid-based Eulerian evaluation with discretized stepping and finite directions of motion does not faithfully reflect the fluidity of human motion trajectories.

Lagrangian methods, typically implemented in the form of free-moving particles, perform their computations in-place, avoiding the fixed-grid problem of Eulerian evaluation. Successful efforts in this area were introduced by Reynold's particle swarm model [21] and Helbing's social forces crowd model [10]. Examples of later variations include HiDAC, which incorporates psychological profiles and pushing behaviour [18]; and physical models, which incorporate the simulation of individual locomotive limbs to generate each entity's motion [2].

A fundamental element of Lagrangian-based methods is neighbourhood detection, the process of identifying each entity's neighbours. This is the primary cost differential when compared to Eulerian evaluation where neighborhoods are typically predefined and directly accessible. Certain data structures can be used to accelerate the neighbourhood search through recursive subdivision (e.g. Octrees). Other structures include the Voronoi diagram, which can be used to limit the search area and accelerate neighbourhood detection using GPUs [4, 22]. Beyond identifying neighbours, centroidal particle dynamics can also utilize Voronoi diagrams to compute each entity's response to violations of its personal space [11].

Human motion has been empirically shown to be anticipatory in nature [19]. People continually scan their environment for potential collision events and enact local maneuvers to avoid those predicted events. Agent-based models built on this principle include Reciprocal Velocity Obstacle (RVO) [5], and a velocity-space optimization model (ORCA) [27].

Other efforts try to mimic the human vision-to-motion feedback cycle, by rendering a 1D [16] or 2D [17] depth map from each entity's perspective, and emulating how humans change their trajectory based on that information alone. Vision-based approaches are unique in their realistic depiction of data encapsulation. That is, they realistically model how an entity does not have direct access to its neighbors' state variables; it can only interpret what it can glean from the depth information in its own perspective. Alas, the computational and memory costs of representing each entity's viewpoint can become prohibitive for large-scale simulation.

This paper adopts the centroidal particles approach in [24, 11] for their realistic depiction of personal space compression in congested settings. The following section briefly describes their basic approach, then outlines our contributions, which add anticipatory collision avoidance, and offloads more CPU workload to the graphics card (GPU) for increased performance and higher frame rates.

3 CENTROIDAL PARTICLE DYNAMICS

Centroidal particle dynamics (CPD) for crowd simulation assume that every entity knows its global trajectory or vector [24, 11]. That is, barring any other dynamic entities in the scene, following the global path will lead each entity to its target location in optimal time. Global pathing algorithms, such as A*, are typically used for such broad scale pathing and already take into account the large-scale static elements of the scene (e.g. walls, doorways, obstacles, etc.). CPD methods then enact local rules to attempt to maneuver around

the surrounding dynamic entities in the scene, with the least deviation possible from the ideal global path.

3.1 Personal Space

The basis for CPD crowds lies in explicitly modeling and evaluating every entity's personal space (PS). Studies in France and North America have shown that the average adult PS is ~0.8m evenly around the center of the entity when idling, and ~0.5m when in motion [19, 29]. These numbers vary slightly across cultures [30], and CPD methods can adapt to PS compression around barriers to motion in moving crowds, or points of interest in static crowds (e.g. closer to the stage at a concert) [11]. When two entities approach each other, they equally share (or violate) each other's personal space. CPD methods model local dynamics by having each entity attempt to be at the virtual center of mass (or centroid) of the unviolated portion of its PS, attempting to restore its preferred PS area over time.

The Centroidal crowds in [11] model this attempt in the form of a linear force in the direction of the centroid (Figure 2). The personal space definition of shared space can be geometrically represented by constrained centroidal Voronoi tessellation [24]. This tessellation does not need to happen pair-wise; it can be computed over the entire domain of simulated entities (including obstacles), accounting for the aggregate infringement of each entity's personal space. This global tessellation is called the Personal Space Map (PSM).

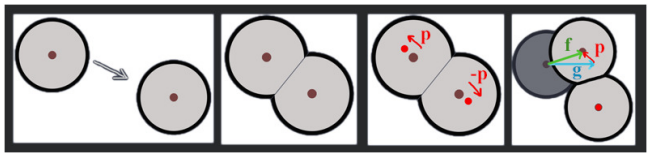

Figure 2. The net force (*f*) experienced by an entity is a linear combination of the global pathing force (*g*), and the penalty force (*p*) which falls along the direction of the new centroid [11].

3.2 Personal Space Map

The global PSM is used to tessellate the space and accelerate the nearest neighbours search [22] by precisely placing and orienting a short-range ray-marching probe in the most optimal position for neighbor detection, without requiring an exhaustive radial search. By contrast, the PSM, illustrated in Figure 3, is used in [11] to avoid the need for any nearest neighbor search explicitly. It does so by relying on the CPD equivalent of data encapsulation: in dense congested scenarios, every entity's local dynamics are informed only by the information within its expected personal space. Hence, regardless of who the entity neighbours are or what obstacle caused the reduction in personal space, every entity will enact the proper local dynamics by only evaluating its expected PS. This high locality also translates to data-parallelism, which could be exploited for acceleration onto a GPU. CPD methods also allow for the modification of the PS base shape (called *PS footprint* in [11]). The modifications include changing the size to accommodate a variety of different entity profiles; reducing the relative Voronoi influence of PS to indicate a more vulnerable pedestrian (e.g. child); and applying an influence map to completely customize how the centroid is calculated (the map is convolved with the integral of the PS area).

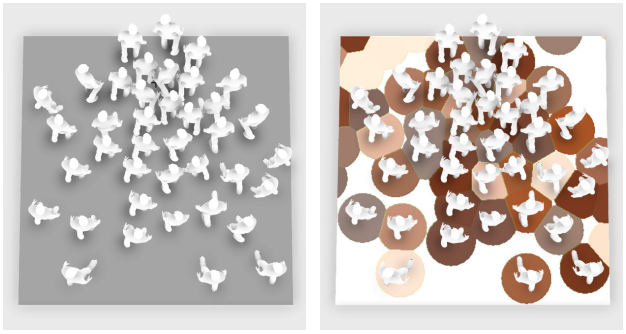

Figure 3. A small section of a larger crowd. Right: the crowd's PSM shown as an underlay. PS colors simply encode entity IDs.

We take advantage of this flexibility by proposing our own modifications and contributions to the CPD base PS shapes, and present an implementation that utilizes the GPU compute capabilities of consumer devices (desktop and mobile alike).

4 CONTEXT-SENSITIVE PERSONAL SPACES

This section outlines our contributions to the CPD model, towards more realistic simulation of dense multi-directional flow.

4.1 Asymmetric PS Weighting

Currently, CPD methods use a PS shape that is evenly weighted around the entity, based on the empirical studies in [19] that demonstrate this fact. However, when an entity has its personal space infringed upon outside of its vision, the entity would unrealistically sense this infringement and react as if it had eyes in its back, so to speak.

Our first modification is the use of a multi-area kernel (shown in Figure 4) that splits the shape into two key areas: i) PS area that affects both the entity and its neighbours; and ii) PS area that only affects surrounding neighbours. This asymmetrical shape is an intuitive change that brings about some implications:

- The net separation between entities in motion remains at the ideal ~1m (twice the 0.5m PS radius). The only difference here is that instead of equally sharing the responsibility, the entity with visibility will now shoulder most of the responsibility and corrective efforts to maintain that distance. This is analogous to a driver maintaining a safe distance from the vehicles ahead (there might be some collaboration, but it is mostly that driver's responsibility).
- Because a single entity (the one in the back) is maintaining most of the separation distance, the severity of PS compression around areas of congestion is reduced.

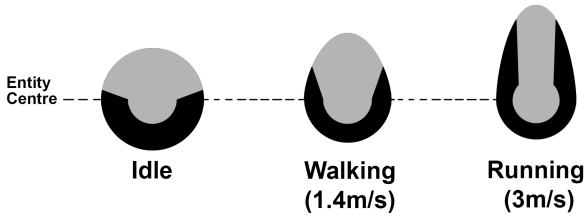

Figure 4. Proposed PS shape: light area affects the entity and its neighbours; dark area only affects neighbors. Lengthening of PS kernel is towards direction of motion as demonstrated.

Additionally, the suggested extension of personal space proportional to the entity velocity was implemented, where the personal space extends by ~0.4m per m/s. This extension is slight for walking speeds (~1.4m/s), but quite noticeable at running or cycling speeds (> 3m/s). Figure 4 illustrates the multipart modification also in relation to speed, to account for the narrowing focus of speeding entities. This is how collision *anticipation* was incorporated into the simulation.

4.2 Resistance to Non-Optimal Bearings

In the linear combination of forces illustrated in Figure 2, we added a resistance element to centroidal forces opposing the optimal global path/objective. This was inspired by the energy-minimization goals set in ORCA [27], and it has reduced the "springiness" of near-miss collisions in pedestrian crossings significantly (especially in bidirectional flow). Therefore, even if the centroid is pointing the entity to face away from the goal -because that is what is locally optimal- the entity will resist this change and instead attempt to wait until more favourable centroidal forces are available.

4.3 Hardware Acceleration via GPU Shaders

Previous CPD methods implemented the PSM using a constrained Voronoi diagram over a discretized surface. The idea was to render every entity PS as a 3D cone viewed from the top [12], and the visible pixels after any intersection will represent the remaining available personal space. This utilization of the graphics pipeline allowed Voronoi-based proximity detection [4] and CPD methods such as [11] to achieve interactive frame rates for thousands of 2D entities in the scene.

In our attempt to accelerate the CPD's PSM computation, the CPU was initially found to be the primary bottleneck, due to the repeated rendering calls made for each entity cone. Each render call came with graphics 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 relatively slow system bus. Unfortunately, this feature could not be naively used for PSM computation because of the dynamic PS shapes, especially with our introduction of velocity-dependent extensions (Figure 4).

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 programmatically 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, comfort speed, etc.) and lets the GPU generate the appropriate voronoidal PS shapes per entity. This reduction in CPU render calls has improved the simulation framerate, as will be shown in Section 5

Furthermore, in order 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).

Sample source code and GLSL shaders are available at http://cell-devs.sce.carleton.ca/publications/ .

5 RESULTS

This section demonstrates how our proposed agent-based method can reproduce several emergent crowd phenomena. Figure 5 illustrates the top view of our simulation of bidirectional flow of a dense crowd (1000 entities) in a corridor. The resulting interlocking pattern is born out of each entity's desire to take the path of least resistance; and in bidirectional scenarios, this simply comes down to avoiding oncoming traffic. As entities traveling in the same direction leave an empty space behind them, the centroid of similarly oriented entities become attracted to fill the void. Hence the appearance of chains or lanes amidst the crowd.

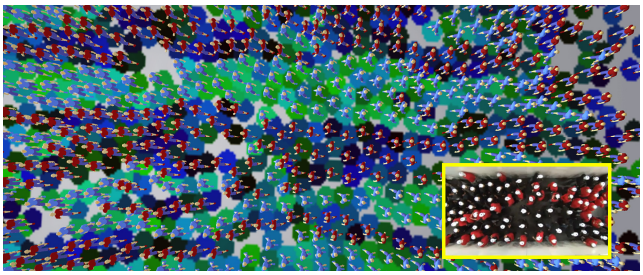

Figure 5. The emergent lane formation produced by our method in a dense bidirectional flow scenario with visually similar forking/joining patterns to those observed in reality (bottom right shows a still frame of real footage [28] of bidirectional flow in a corridor). The entities in both our simulation and the real footage are color-coded to indicate direction of motion.

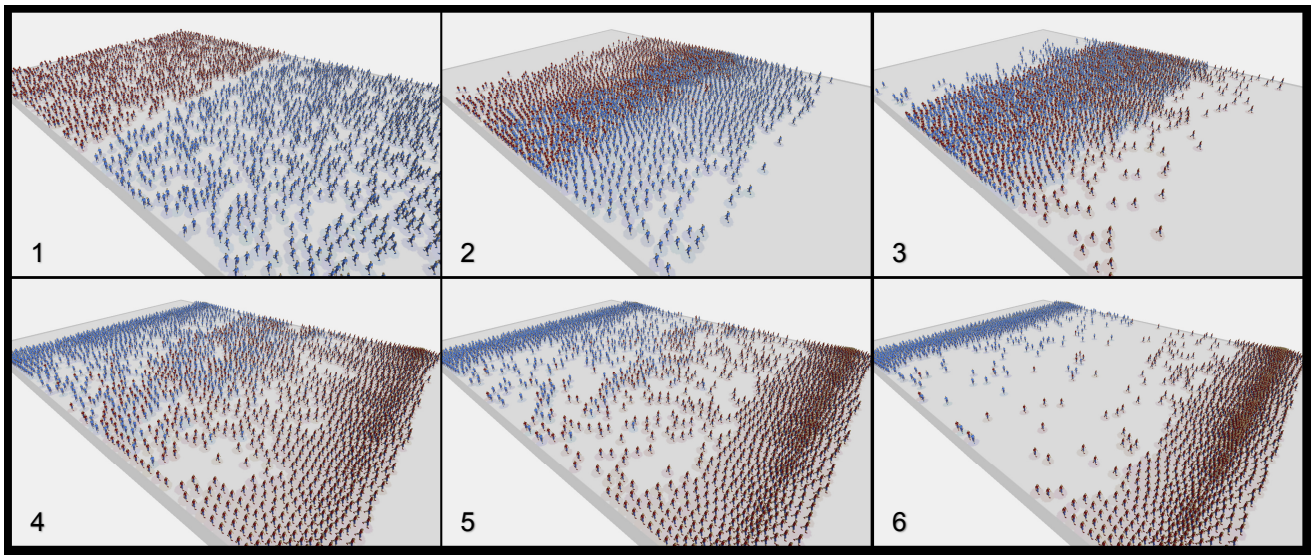

Figure 6. Artificial scenario used for performance testing: blue entities heading north; red entities heading south.

5.1 Simulation Setup

All simulations were run with discrete-time integration, using a quantum of 100ms per frame. Each pixel length represented 10cm of physical space.

The entity PS base radii were kept at 7 pixels (plus 1 centroidal) to achieve the ~0.8m PS idle radius. The shaders described in Section 4.3 were implemented using OpenGL Shading Language (GLSL). Parameters were randomized across the crowd, including the PS radius, comfort speed, and centroidal effect (which alters how aggressive/lenient an entity is about restoring its personal space).

Children were given the same PS radius as adults, but rendered with weaker Voronoi cones (i.e. further away from the PSM top view camera) to reflect the increased the chance of being overpowered by adult personal spaces, or being swept away by strong crowd flow in dense settings.

5.2 Performance

Our testbed consisted of three representative consumer-grade devices with various CPU-GPU configurations:

- Mid-range Desktop: Intel Core i5+ Nvidia GTX1060 GPU
- Laptop: Intel Core i5 (with integrated HD Graphics 4000)
- Mobile: Nexus 6P + Qualcomm Adreno 430 GPU

Table 1 summarizes the average framerate of simulating bidirectional flow over a 600x900 PSM (effectively, a 60m corridor) while varying the number of pedestrians in the scene (Figure 6). The performance gain from implementing our GPU shaders is noticeable, at ~2.6x throughout. While the Android device was capable of simulating higher crowd counts, it was no longer at interactive framerates (< 1fps).

	Non-Instanced Rendering + 2D Sprites			With GPU Shaders + 3D Sprites	
#Entities	**Desktop GTX1060**	**Laptop Corei5**	**Nexus 6P (Android)**	**Desktop GTX1060**	**Gains**
100	160	105	30	450	2.8x
250	125	87	23	338	2.7x
500	90	62	18	266	3.0x
1,000	58	41	13	134	2.3x
1,500	44	31	10	121	2.8x
2,000	33	26	7	89	2.7x
5,000	16.5	12	3	47	2.8x
10,000	10.7	7	-	25	2.3x
15,000	7.6	4.5	-	20	2.6x
20,000	6.2	3	-	14	2.3x

Table 1. Simulation performance in frames per second.

Currently, the simulation performance shows a dependence on the simulation range area. For instance, the 2000 entities in Figure 1 are in a 50x80m corridor (500x800 PSM pixels), which naturally results in faster PSM rendering than the larger PSM area in the Figure 6/Table 1 benchmark. The granularity of PSM pixels can be adjusted to reach performance targets, at the cost of reduced PS fidelity.

Ideally, the simulation performance would be dependent only on the crowd count. We believe the current overhead of essentially simulating empty spaces can be overcome (or hidden) by utilizing multi-threaded CPU rendering calls, as will be possible in upcoming graphics API standards, such as Vulkan [31], the direct successor of OpenGL.

Given the 100ms time quantum, every 10 frames represent 1 second of simulation time. Hence, our algorithm produces faster-than-real-time simulation results for up to 20,000 entities in the scene, and maintains interactive framerates for even higher counts. Performance can be further increased on machines equipped with workstation-grade GPUs.

5.3 Visualization

The shader computation allowed room for real-time 3D sprite rendering. Our simulation can be visualized using existing methods, which generate smoothly rigged and GPU-animated characters [22] at interactive frame rates. For demonstration purposes, we opted instead to create low-cost rigidly-rigged multi-part characters primarily composed of simple primitives, and procedurally animated walk cycles. Animating these composites is only a matter of adjusting a handful of transform matrices (position, orientation, scale, etc.) per entity, rather than fully animating each vertex through smoothed skeletal rigging. The resulting 3D sprites are sufficient for rapid prototyping and iteration, consuming on average 15-30% of each frame time. The results presented in Table 1 include this 3D visualization stage.

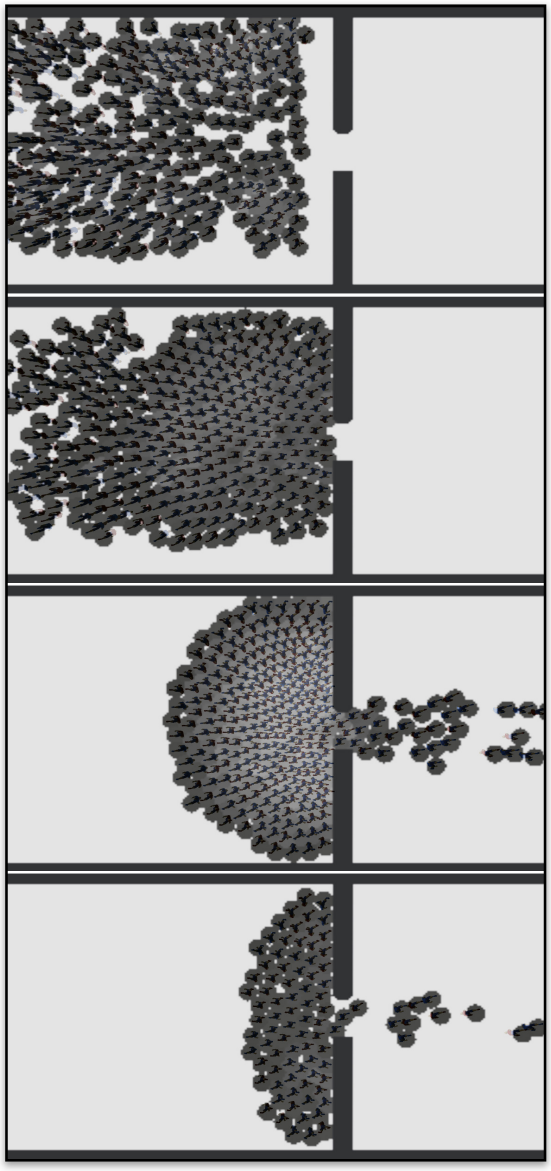

Figure 7. Crowds attempting to pass through a narrow gateway typically begin to arch around that bottleneck.

The PSM can be directly shown as an underlay to visualize the personal spaces used for centroidal force calculations. They can also be used for quick local density visualization by measuring the ratio of the violated area to the ideal PS footprint. Figure 7 illustrates a common emergent crowd phenomenon (arching around pathway bottlenecks), with noticeable compression of personal space near the exit.

This effect aligns with observed PS compression in both moving and static crowds (Figure 8). In addition to arching, densely packed crowds tend to display petal-like formations (as each entity attempts to be situated behind the midpoint of two entities ahead). This increases the entity's visibility of the point of interest (or global path destination), and results in more compact space-filling.

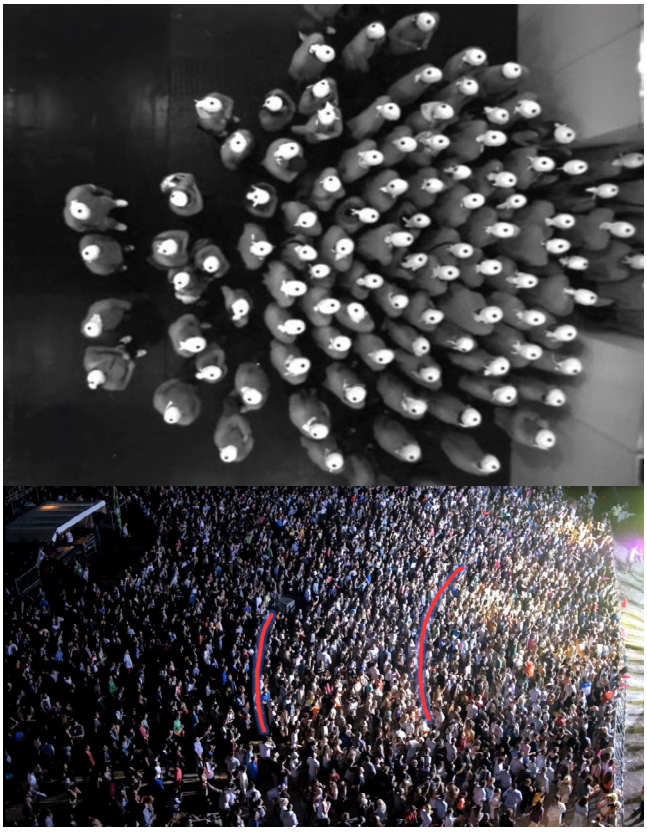

Figure 8. Arching, gradual PS compression, and petal-like formations observed in crowds during egress [28] (top) and while stationary at a concert (bottom). These effects agree with our simulation results.

More experiments, videos, and results can be viewed on the project page at: http://cell-devs.sce.carleton.ca/publications/.

6 LIMITATIONS AND OPPORTUNITIES

Implementation Platform

Our implementation and visualization so far were built on the Java-based Processing framework. Processing is good for rapid prototyping of graphical applications, providing an accessible and cross-platform set of tools for window, file,

and UI management, while simultaneously providing access to low-level OpenGL calls. Also, being Java-based meant that it could be ported directly to Android's platform. Since most of the work is done by GPU shaders, there would not have been much to gain from attempting to use Android's native C++ (NDK) environment.

Although Processing is great for prototyping, it might not be the ideal *deployment* solution. Our method can be made into a plugin for middle-ware engines, depending on the context. For instance, in the film and gaming industry, a worthwhile effort would be to port this to the Fabric Engine [8], which can be run in standalone mode or be made to communicate directly with the scene graphs of major 3D animation packages (e.g. Autodesk Maya and 3ds Max) without rewriting the code for each target platform. The potential overhead when adopting middleware, however, is always an issue that requires careful evaluation.

Heterogeneous and Multi-layered Crowds

The centroidal force indicates a locally preferred bearing and direction of motion for the entity to restore its personal space. However, acting on that centroidal "suggestion" is left up to the entity and its constraints. Human motion is quite flexible with the ability to turn in-place if needed. To extend the simulation to heterogeneous entities sharing the same space, we can still compute the centroidal forces as we did with humans, but the mechanics of following that centroidal "suggestion" might differ (e.g. for strollers, shopping carts, and vehicles).

Those other entities will consume the same rules about personal space, but execute those maneuvers under their own physical constraints (e.g. shopping carts might have a turning radius compared to a human's ability to turn on the spot). To complement this effort, other methods for computing the Voronoi PSM must be tested, since the scene might include lengthy entities whose centroid is no longer a concentric point, but a spine segment. In this case, the jump flooding technique might be a good alternative to Voronoi cones [23].

Another feasible improvement on CPD involves separating gaze from body orientation for situations when, for example, an entity is crossing the road or paying attention to a loud event or scream. The gaze could be computed in its own PSM layer, and it would become another parameter in the context-sensitive personal space adjustments.

Flocking

By design, CPD methods don't produce grouping or flocking behaviour as they focus on dynamics within a personal space. The entities are assumed to be individualistic with their own target destination in mind. This was intentional, to study the effects of centroidal forces in isolation.

By manipulating the global pathing vectors being input into CPD methods, it would be possible to augment (or be augmented by) behaviours such as the ones studied in SAFEgress [3], to account for each entity's familiarity with the environment, social attitudes, and herd dynamics (leader-follower, social order, etc.). This concept also extends to simulating families and friends trying to stay together when at a large gathering or outdoors event.

Validation

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 behaviour, it cannot yet be reliably used for safety-critical applications. That would require further validation against in-lab scenarios [28] and statistical analysis. There are global statistical properties that can be checked (e.g. governing distributions [14]) and local similarity indices for targeted analysis of smaller areas of interest (e.g. [9]). 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. Simulation then becomes an important tool for preplanning barriers and other crowd control measures to prevent such awful disasters in what are otherwise peaceful gatherings [7, 26].

7 CONCLUSION

We presented an improvement to the centroidal particle dynamics (CPD) model that addresses several subtle problems in dense crowd simulation. The contributions include a context-aware personal space kernel that adjusts to the bearing of the entity, its velocity, and destination heading, resulting in a more realistic response to personal space violations and collision anticipation.

The presented implementation aides in scaling the algorithm by utilizing geometry and vertex shaders to offload the computationally demanding personal space map (PSM) and centroidal calculations onto graphics hardware (GPU). This allows for the animation of 5000+ 3D entities at interactive framerates on consumer-grade hardware.

The agent-based simulation produces several visually convincing emergent results for crowd crossings, dense bidirectional flow, and arching near hallway bottlenecks.

ACKNOWLEDGMENTS

This research has been partially funded by NSERC and the first author further supported by an OGS (Ontario Graduate Scholarship). The authors thank members of the Advanced Real-time Simulation lab for their participation and support, and the anonymous reviewers for their insightful feedback.

REFERENCES

1. Bandini, S., Manzoni, S., & Vizzari, G. Crowd Modeling and Simulation. *Innovations in Design & Decision Support Systems in Architecture and Urban Planning*, 105-120 (2006).

2. Brogan, D.C., Metoyer, R.A., and Hodgins, J.K. 1998. Dynamically simulated characters in virtual environments. *IEEE computer graphics and applications* 18, 5, 58–69.
3. Chu, M.L., Parigi, P., Law, K., and Latombe, J.-C. 2014. SAFEgress: A Flexible Platform to Study the Effect of Human and Social Behaviors on Egress Performance. *SimAUD Proceedings*, SCS 4:1–4:8.
4. De Gyves, O., Toledo, L., and Rudomín, I. 2013. Proximity Queries for Crowd Simulation Using Truncated Voronoi Diagrams. Proceedings of Motion on Games, *ACM*, 87–92.
5. den Berg, J. van, Lin, M., and Manocha, D. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. Robotics and Automation, 2008. *ICRA*, 1928–1935.
6. Duives, D.C., Daamen, W., and Hoogendoorn, S.P. 2013. State-of-the-art crowd motion simulation models. *Transportation Research Part C: Emerging Technologies* 37, 0, 193–209.
7. Fruin, J.J. 1993. The causes and prevention of crowd disasters. *Engineering for Crowd Safety* 1, 10.
8. Fabric Software Inc. Fabric Engine. http://fabricengine.com/.
9. Guy, S.J., van den Berg, J., Liu, W., Lau, R., Lin, M.C., and Manocha, D. 2012. A Statistical Similarity Measure for Aggregate Crowd Dynamics. *ACM transactions on graphics* 31, 6, 190:1–190:11.
10. Helbing, D., Farkas, I., and Vicsek, T. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803, 487–490.
11. Hesham, O. and Wainer, G. 2016. Centroidal Particles for Interactive Crowd Simulation. *Proc. SummerSim, Society for Computer Simulation International*, 7:1–7:8.
12. Hoff, K.E., III, Keyser, J., Lin, M., Manocha, D., and Culver, T. 1999. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. *Proceedings of the 26th CGIT*, ACM, 277–286.
13. Huerre, S., Lee, J., Lin, M., and O'Sullivan, C. 2010. Simulating Believable Crowd and Group Behaviors. *ACM SIGGRAPH ASIA 2010 Courses*, ACM.
14. Karamouzas, I., Skinner, B., and Guy, S.J. 2014. Universal power law governing pedestrian interactions. *Physical review letters* 113, 23, 238701.
15. Kisko, T.M., Francis, R.L., and Nobel, C.R. 1998. Evacnet4 user's guide. *University of Florida*.
16. Moussaïd, M., Helbing, D., and Theraulaz, G. 2011. How simple rules determine pedestrian behavior and crowd disasters. *Proc. of the National Academy of Sciences of the United States of America* 108, 17.
17. Ondřej, J., Pettré, J., Olivier, A.-H., and Donikian, S. 2010. A Synthetic-vision Based Steering Approach for Crowd Simulation. *ACM transactions on graphics* 29, 4, 123:1–123:9.
18. Pelechano, N., Allbeck, J.M., and Badler, N.I. 2007. Controlling Individual Agents in High-density Crowd Simulation. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics SCA*, 99–108.
19. Pettré, J., Ondřej, J., Olivier, A.-H., Cretual, A. & Donikian, S. 2009. Experiment-based Modeling, Simulation and Validation of Interactions between Virtual Walkers. *ACM SIGGRAPH/Eurographics SCA* 189–198.
20. Peschl, I.A.S.Z. 1971. Passage Capacity of Door Openings in Panic Situations. *BAUN*.
21. Reynolds, C.W. 1999. Steering behaviors for autonomous characters. *Game developers conference*, 763–782.
22. Rivalcoba, I. and Ruiz, S. 2013. GPU generation of large varied animated crowds. *Computación y Sistemas* 17, 3, 365–380.
23. Rong, G. and Tan, T.-S. 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. *Proc. Interactive 3D graphics and games*, ACM, 109–116.
24. Secord, A. 2002. Weighted Voronoi Stippling. *Proceedings of NPAR*, ACM, 37–43.
25. Smith, R. A. Volume Flow Rates of Densely Packed Crowds. *Engineering for Crowd Safety* (1993).
26. Special Events Contingency Planning. 2005. *FEMA*.
27. van den Berg, J., Guy, S.J., Lin, M., and Manocha, D. 2011. Reciprocal n-Body Collision Avoidance. In: C. Pradalier, R. Siegwart and G. Hirzinger, eds., *Robotics Research*. Springer Berlin Heidelberg, 3–19.
28. Zhang, J., Klingsch, W., Schadschneider, A., and Seyfried, A. 2011. Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. *arXiv* [physics.soc-ph].
29. Gérin-Lajoie, M., Richards, C.L., and McFadyen, B.J. 2005. The negotiation of stationary and moving obstructions during walking: anticipatory locomotor adaptations and preservation of personal space. *Motor control* 9, 3, 242–269.
30. Chattaraj, U., Seyfried, A., and Chakroborty, P. 2009. Comparison of Pedestrian Fundamental Diagram. *Advances in Complex Systems 12*, 03, 393–40.
31. Khronos Vulkan Working Group, 2016. Vulkan 1.0.37 - A Specification. *Khronos Group*, 11-18.