Centroidal Particle Dynamics: An Explicit Model of Pedestrian Personal Space for the Simulation of Short-Range Collision-Avoidance and Emergent Motion Patterns in Dense Crowds

by

Omar Hesham

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

Computer simulation of dense crowds is finding increased use in event planning, congestion prediction, and threat assessment. State-of-the-art particle-based crowd methods assume and aim for collision-free trajectories. That is an idealistic yet not overly realistic expectation, as near-collisions increase in dense and rushed settings compared to typically sparse pedestrian scenarios. We propose Centroidal Particle Dynamics (CPD), a method that explicitly models the compressible *personal space* area surrounding each entity (~0.8m-1.0m radius) to inform its local pathing and collision avoidance decisions. While personal space has traditionally been modeled as a fixed radius, the reality is that it often changes 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. passing through a crowded gate 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 -unoccupied- space. We illustrate how our proposed agent-based method for local dynamics can reproduce several key emergent dense crowd phenomena at the microscopic level (e.g. emergent lane formation in bidirectional flow and arching near congested gateways) with higher congruence to real trajectory data and with more visually convincing collision avoidance paths than the existing state-of-the-art.

We further show how CPD can be efficiently computed on consumer-grade graphics hardware (GPU), achieving interactive frame rates when simulating thousands of crowd entities in the scene, thus making it suitable and ready-for-use in our target applications of entertainment and interactive media projects (i.e. film, gaming, and educational media). Lastly, we discuss crowd motion validation, and how to increase confidence in CPD, potentially making it also suitable for use in safety-critical applications, including urban design, evacuation analysis, and crowd safety planning.

Acknowledgements

This work would not be possible without the dedication, mentorship, honesty, friendship, and trust that Dr. Gabriel Wainer has provided throughout my years at Carleton University's Advanced Real-time Simulation Lab. He had enormous confidence in me at times when I had none; and he inspired me to learn, take risks, improve, and teach. Thank you, professor.

Publications emanating from this work were supported by funds from NSERC. Special thanks to my undergraduate and master's supervisor, Dr. Chris Joslin, who was the first to invite me to the world of graphics research and academia; and later encouraged and funded my pursuit of admission to a PhD program. He will always be my mentor and a source of inspiration.

Thank you to the faculty, admin, and staff members at Carleton's School of Information Technology, and the Department of Systems and Computer Engineering, and the Faculty of Graduate and Postdoctoral Affairs for supporting me throughout my graduate studies and teaching duties.

To my lab mates Ala'a, Baha, Cristina, Ife, Sixuan, Michael, Jan, Joseph, and the many simulation-curious geeks I've shared weekly seminars with; thank you for your support and pleasant company.

To my friends Chris, Sina, Anas, Allen, Vanessa, Patrick, Jonathan, Guillaume, and to my dear siblings Mahmoud, Yasser, Noha, Moustafa, and Karim. You all make my life better in unique ways; and I'm grateful for your patience with all the long selfish cave-mode sessions I took over the years to work on this dissertation. I owe you all a dinner or two.

Mom and dad, you bought me a license for Poser 3D when I was 12 just because I was curious; now look what that's done!

Thank you. This is for you.

Table of Contents

List of Acronyms

PS	Personal Space
PSM	Personal Space Map
CPD	Centroidal Particle Dynamics
CVT	Centroidal Voronoi Tessellation
DEVS	Discrete-Event System Specification
RVO	Reciprocal Velocity Optimization

List of Tables

List of Figures

Chapter 1

Introduction

Dense crowd simulation is an area of research concerned with assessing and predicting the motion paths of large groups of people within a limited physical space. Applications range from presenting crowds in gaming and film production, to designing public spaces and assessing their quality of occupancy, to the safety-critical analysis of the potential for stampedes and crowd crushes [1]. In this work, we present our effort to model and simulate the collision-avoidance exhibited by individuals in high-density crowds. We focus on simulating the avoidance and steering decisions that each entity (i.e. human) makes in response to the position of other entities or obstacles in their immediate surroundings (span of roughly an arm's length). We use the term "personal space" to refer to that short-ranged area of interest; and we further define it and its simulation dynamics throughout the following chapters.

The paths of two pedestrians walking from the same starting positions and reaching the same destination positions often display slight deviations from each other. Ask the same person to walk the same path twice, and you would still get deviations from one trajectory to another. Human motion is seemingly non-deterministic, and pedestrian path simulation is currently an exercise in imprecise abstraction.

When simulating the movement of high-density pedestrian traffic (e.g. at outdoors festivals, concerts, or mass gathering events), macroscopic methods that rely on aggregate parameters (bringing a sense of determinism through bounded stochasticity) can be very effective when analyzing collective motion metrics, such as rate of evacuation and expected human density distribution over an area. Because they do not rely on simulating individual entities, those macroscopic methods are often efficient enough to

accommodate large-scale simulation of the position update of thousands of crowd members. However, as hardware continues to evolve to provide increased parallelism and power efficiency, microscopic methods that can reproduce the intricate details of every single individual's trajectory and cognitive state are becoming increasingly accessible to designers, architects, and event planners to readily assess the risks and focus stakeholder efforts around potential congestion issues.

1.1 Topic Motivation

The Modeling and Simulation (M&S) of virtual crowds has found a certain appeal as an illustrative tool in urban and architectural projects. It allows designers and engineers to visualize the utility of their project's space and facilitates the feedback, discussion, and decision-making among all stakeholders, be they technical, business-oriented, or otherwise. A more demanding tier of crowd simulation can be seen in the entertainment and serious gaming industries. While theoretical accuracy is desired in this context, practical concerns favour other objectives: visual believability, stability in an unpredictable environment, and the performance to meet the real-time requirements of interactive experiences. Creative applications further demand a certain degree of artistic control. In a similar vein, serious gaming, which often involves interactive scenarios for training, education, and social purposes, tends to aim for model accuracy but ends up favoring performance that maintains a fluid simulated scenario for the learner. Advances in parallel computing and ever-increasing hardware affordability are helping close the gap towards accuracy in interactive simulation, including on low-power and mobile devices.

The most demanding tier of crowd simulation applications comprises civil safety analysis tools, contingency planning, and military applications. The simulation of emergency evacuation procedures, prediction of traffic bottlenecks, and shaping of pedestrian flow by manipulating access point allocations and doorway designs, are all examples of design-stage activities in this tier. They help the authorities assess threats and plan preventative measures that minimize the risk of disasters in large

crowded events such as outdoor music festivals, public forums, and sporting events [2], [3]. Due to their critical nature, these models typically require a high degree of rigorous validation, data-driven calibration, and conformity with existing modeling standards such as High-Level Architecture (HLA) or with industrial formats such as Building Information Modeling (BIM) [2].

This diversity of application requirements induced an equally rich variety of approaches to the modeling and abstraction of crowd behaviour. According to our own assessment, a hierarchy of systems divides the abstractions into three inter-operating levels: a cognitive model, a global pathing model, and a local interaction model. The cognitive model is tasked with broad decision-making, such as deciding where the target location is, and interacting with the entity's goals and personality traits that could alter such decisions (e.g. following a parent during an evacuation instead of taking the nearest exit). Once the target location is decided, the global pathing module analyzes the spacial structure of the static environment and finds an optimal path according to some cost minimization (e.g. taking the stairs vs. the elevator); but typically, just the shortest path or least congested one. Finally, the local interaction module further modifies the path to navigate around and avoid collision with minor dynamic obstacles, which include other pedestrians, gates, and doorways, while still generally following the optimal path and attempting not to stray too far away from it.

For a quick analogy:

- Cognitive model: a car driver deciding on a destination and entering it into their GPS device;
- Global pathing model: the navigation service (e.g. Google Maps) would suggest optimal route(s) to reach the chosen destination;
- Local dynamics model: finally, the driver has the responsibility and control over the local maneuverability of the car (e.g. lane switching, overtaking slower cars, collision avoidance, etc.);

Figure 1.1 Dense pedestrian activity at Shibuya crossing, Tokyo, Japan.
(via: JTNO -Official Tourism Guide for Japan Travel).

While there are crowd simulation methods that blur the lines and attempt to solve more than one level simultaneously (e.g. Continuum Crowds [4]), our hierarchical view encourages the separation of concerns (e.g. the GPS navigation model is not expected to also be a self-driving local avoidance model), and allows further experimentation and mixing of components and solutions from various sources.

This dissertation focuses on the modeling of local dynamics of dense crowds, with the assumption that the cognitive and global pathing models are adequately modeled by other methods and can feed their desired destination into our local dynamics model (more on force integration in Chapter 3).

1.2 Problem Statement

In high density crowds, pedestrians often make collision avoidance and trajectory correcting decisions within close proximity to other pedestrians or obstacles. Given the real-world benefits of accurate microscopic trajectories of pedestrians, as contextualized in the previous section, the state-of-the-art in

crowd simulation methods does not accurately reproduce the emergent dense crowd trajectories and phenomenon, such as unstable bidirectional flow and density gradients near gateways and bottlenecks. Those individual pedestrian trajectories are, by definition, microscopic models of a crowd's collective motion; and thus, many macroscopic methods, which aggregate and hide the details of individual pedestrians cannot address our problem statement. A review of related work is presented in Chapter 2, along with a gap analysis in Section 2.3 specifically exploring those shortcomings in microscopic methods.

1.3 Contributions and Publications

Our focus in on simulating motion paths (i.e. trajectory of entity's center of mass) adjustments in response to changes in an entity's short-range environment (roughly an arm's length); and particularly for high density scenarios. As will be expanded upon in Chapters 2 and 3, at this level of density, we do not consider psychological factors. To this end, we present a novel area-based penalty force that operates on the premise of personal-space-perseveration; and we demonstrate promising results obtained from simulation on consumer-grade graphics hardware. The model successfully reproduces empirically known crowd phenomenon such as lane formation in bidirectional flow and arching around areas of congestion, where state-of-art either struggles or fails completely. Figures 1.1 and 1.2 show examples of dense crowd scenarios we're tackling. We aim to provide a local collision avoidance model customizable through graphical parameters that are designer-friendly for creative control and facilitate 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 further scope the dissertation's goal by specifically targeting film, gaming and education media as the application context we aim to deploy our algorithm in.

Figure 1.2 A crowded outdoors event with dense bidirectional traffic.
(via: Locke Street Festival, Hamilton, Canada).

The author's background in multimedia and design encouraged the exploration of applying system-theoretic formalisms (such as PDEVS [5][6]) into gaming and interactive graphics contexts. Particularly, exploring how an event-driven approach might improve the performance of graphics engines. The appeal of this approach came from two key features of discrete-event systems: (i) The asynchronous execution of components/submodels where processing power is only spent on those elements responding to an event (internally scheduled or externally received); and (ii) the ability to use a continuous timeline, rather than a fixed timestep as is common in current gaming, graphics, and multimedia engines/frameworks.

In summary, our proposed Centroidal Particle Dynamics (CPD) method is an explicit 2D model of the dynamics of response to violations of personal space. It is implemented through autonomous Lagrangian agents, which emergently recreate global phenomenon observed in dense crowds with higher congruency to real-life trajectory data than existing state-of-the-art methods in real-time crowd simulation for film, gaming, and interactive educational media.

This research effort has produced several publications on interactive crowd simulation and visualization, which have generally been well-received by the simulation community:

- Michael Van Schyndel, Omar Hesham, Gabriel Wainer, and Brandon Malleck. **2016**. *Crowd Modeling in the Sun Life Building.* In Proceedings of the Symposium on Simulation for Architecture & Urban Design (SimAUD '16).
    - Highlight: presented a case study on the practical use of cell-based crowd simulation for urban planning; and a critique of offline 3D visualization.
- Omar Hesham and Gabriel Wainer. **2016**. *Centroidal particles for interactive crowd simulation.* In Proceedings of the Summer Computer Simulation Conference (SCSC '16). Society for Computer Simulation International, San Diego, CA, USA, 8 pages.
    - Highlight: introduced centroidal particles and analyzed preliminary results.
    - *Awarded*: Best Paper - Summer Computer Simulation Conference (SCSC).
    - *Awarded*: Best Paper Overall - SummerSim Multiconference.
- Omar Hesham and Gabriel Wainer. **2017**. *Context-sensitive Personal Space for Dense Crowd Simulation.* In Proceedings of the Symposium on Simulation for Architecture & Urban Design (SimAUD '17).
    - Highlight: improved personal space model and doubled GPU performance.
    - *Awarded*: Best Student Paper.
- Omar Hesham, Princy, Walter Aburime, Shashi, Ziyad Rabeh, and Gabriel Wainer. **2018**. *Observed Behaviour in Simulated Close-range Pedestrian Dynamics.* In Proceedings of the Symposium on Simulation for Architecture & Urban Design (SimAUD '18).
    - Highlight: scenario studies (e.g. congestion in a narrow hallway due to handful of idle pedestrians).
- Bruno St-Urbain, Omar Hesham, and Gabriel Wainer. **2018**. *A Cell-DEVS Visualization and Analysis Platform.* In Proceedings of the Summer Computer Simulation Conference (SCSC '18).

- Highlight: visualization and statistical analysis platform for simulated grid data using portable web technologies (HTML/CSS/JScript). The same platform was used to visualize and analyze cell-based crowds.

The tools and framework developed throughout this research were further utilized in courses at University of Buenos Aires and at Carleton University, where graduate students studied specific scenarios (e.g. congestion in narrow hallway due to handful of idle pedestrians). More on student work is discussed in Chapter 5.

1.4 Thesis Document Organization

This thesis presents our contributions to crowd modeling and simulation. The relevant background is discussed in Chapter 2, the proposed method and a GPU-accelerated performant implementation are presented in Chapter 3, followed by a discussion of the results and their real-time performance in Chapter 4. We conclude in Chapter 5 by reflecting on the proposed method's validation, limitations, and opportunities for future work.

Chapter 2

Background

Like other physically-based phenomena which exhibit complex interactions between multiple entities over time, pedestrian motion could only be practically expressed and simulated using numerical, rather than analytical, methods. Generally, crowd simulation models are of the form a differential equation over time. There are then different granularities of motion abstraction depending on the use-case and its requirements such as performance targets, validation metrics, and the acceptable error margins. The choice of numerical method could also vary depending on the availability of limited resources, such as processing power, memory constraints, and source data availability. In this chapter, we overview past and current crowd simulation methods relevant to our pursuit of improved local collision avoidance in dense crowds for film, gaming, and training applications.

2.1 Related Work

Replicating human decision-making is a highly ambitious endeavor, never mind simulating an entire crowd of them. To this end, the abstraction of motion dynamics by generalizing observed phenomenon is necessary to achieving a computable result. This section presents a brief overview of the multitude of methods developed to tackle this problem. For a more detailed analysis, the reader is referred to the critical assessment done in [1], [7].

2.1.1 Macroscopic Crowd Models

Historically, the earliest crowd simulation methods were macroscopic in nature, simulating aggregate behavioural patterns, 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 [8]–[10] to provide computational stability and high performance. This granularity of simulation was sufficient to assess and validate collective motion parameters such as egress (i.e. evacuation) rate and density distribution over a given scene layout. Flow-based methods have since evolved, with notable modern contributions such as Continuum Crowds [4] delivering visually convincing large-scale results at interactive frame rates, suitable for animation, gaming, training, and educational media.

Network optimization techniques have also been adapted to simulate occupant movement within a predefined multi-compartment environment [11], [12]. Each compartment is treated as a graph node that might represent a section within a room, a hallway, or even an entire building. The choice of what a node represents depends on the desired granularity of describing the spatial structure of the model. The edges connecting those nodes would then represent the capacity of pedestrians moving between one node to another. Safety engineers, architects, and event planners who are concerned about occupant experience could then utilize classic optimization techniques such as Dijkstra's shortest path [13] or max-flow detection [14] in order to focus their limited resources and efforts on areas of potential pedestrian bottlenecks.

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 (e.g. building or stadium evacuation) which involve potentially high crowd counts [1], [15].

2.1.2 Microscopic Crowd Models

With ever-increasing hardware capabilities and improved modeling methodology, the ability to simulate individual entity-to-entity interactions have become computationally viable. The aim with microscopic methods is to simulate the individual agents with localized rulesets and whose emergent

behaviour is intended to match 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 [16]. In CA methods, the space is typically divided into a uniform grid, where every cell can either be available, occupied by 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 inherent visualization (every cell is both the computational unit and the visual representation). Nevertheless, grid-based Eulerian evaluation of agent dynamics using discretized stepping and finite directions of motion does not faithfully reflect the fluidity of human motion trajectories.

The numerical solution of Eulerian methods requires a spatially-fixed set of sampling and solving points, typically a regular grid. By contrast, Lagrangian methods, which are typically implemented in the form of free-moving particles, are spatially unrestricted and can perform their numerical computations (e.g. neighbourhood sampling, collision detection, advection, etc.) in-place, avoiding the finite and limited degrees of motion exhibited by Eulerian models. Successful efforts in this area were introduced by Reynold's flocking model [17] and Helbing's social forces crowd model [18]. Later variations include HiDAC [19], which seeks to incorporate psychological profiles and physical pushing behaviour. When taken to an extreme, microscopic algorithms could opt to simulate the individual joints of every entity's anatomy (e.g. legs on a human, pedals on a bike, etc.) in order to generate a biomechanically accurate locomotion towards a given location [20].

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 [21], [22].

Pedestrian trajectories have been empirically shown to be anticipatory in nature [23]. 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) [24], and a velocity-space optimization model (ORCA) [25]. Modifications to ORCA allow it to exhibit density-dependent behaviour [26], and data-driven motion profiles [27]. While those methods typically operate on linear path and velocity predictions, methods like WarpDriver [28] perform a non-linear prediction of the upcoming optimal path for a relatively short imminent time frame (within a handful of seconds). The non-linear prediction can be visualized as a short-range area of optimal path prediction values, and the entity can "learn" to adjust the shape of that prediction based on past experiences. The final choice of a single advection direction from within that predicted area comes down to a combination of factors including the entity's objective functions and the state of the surrounding environment (i.e. other entities and obstacles). Ideally, entities would not have access to the predicted paths of their surrounding neighbours, in order to mimic the true encapsulation of such data in real pedestrians.

Other efforts try to mimic the human vision-to-motion feedback cycle, by rendering a 1D [29] or 2D [30] 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 primarily 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.

2.2 Discrete-event Simulation

The state-of-the-art in microscopic crowd simulation favors discrete-time Lagrangian modeling [1], [7], [31]. But it could be worthwhile to pursue a discrete-event model. The key differentiator here is the model's ability to perform state updates only when necessary. That is, instead of calculating the motion of the all the entities synchronously at regular time intervals (i.e. discrete-time), each entity would instead asynchronously evaluate its own state transitions in response to *events* (i.e. discrete-event); which could either be internally scheduled timed events, or events due to external sources such as collision, a sensory stimulus, or a cognitive decision. Thus, for example, if a passive pedestrian's neighbourhood doesn't change, no compute cycles would be wasted on updating that pedestrian's state. This potential saving in computational cost is one reason to consider discrete-event modeling of crowds.

The benefit of discrete-event modeling is perhaps more apparent for server-client type of systems. For instance, to simulate an ATM machine that only changes its state upon interaction with a bank client, it would be quite wasteful to update its state synchronously with the rest of the simulated world in a discrete-time fashion. Instead, a discrete-event simulation that advances time in response to events is more suitable in this case. In the context of crowd simulation, discrete-event systems have been explored in macroscopic and microscopic models [16], [32], [33], and [34] (Figure 2.1); most of which rely on an Eulerian numerical solution and an axis-aligned grid-like tessellation of the scene layout.

It remains to be seen what benefits or challenges that a discrete-event approach would bring to Lagrangian microscopic crowd modeling, although promising efforts already exist for the discrete-event simulation of particle-based physics [35]–[37].

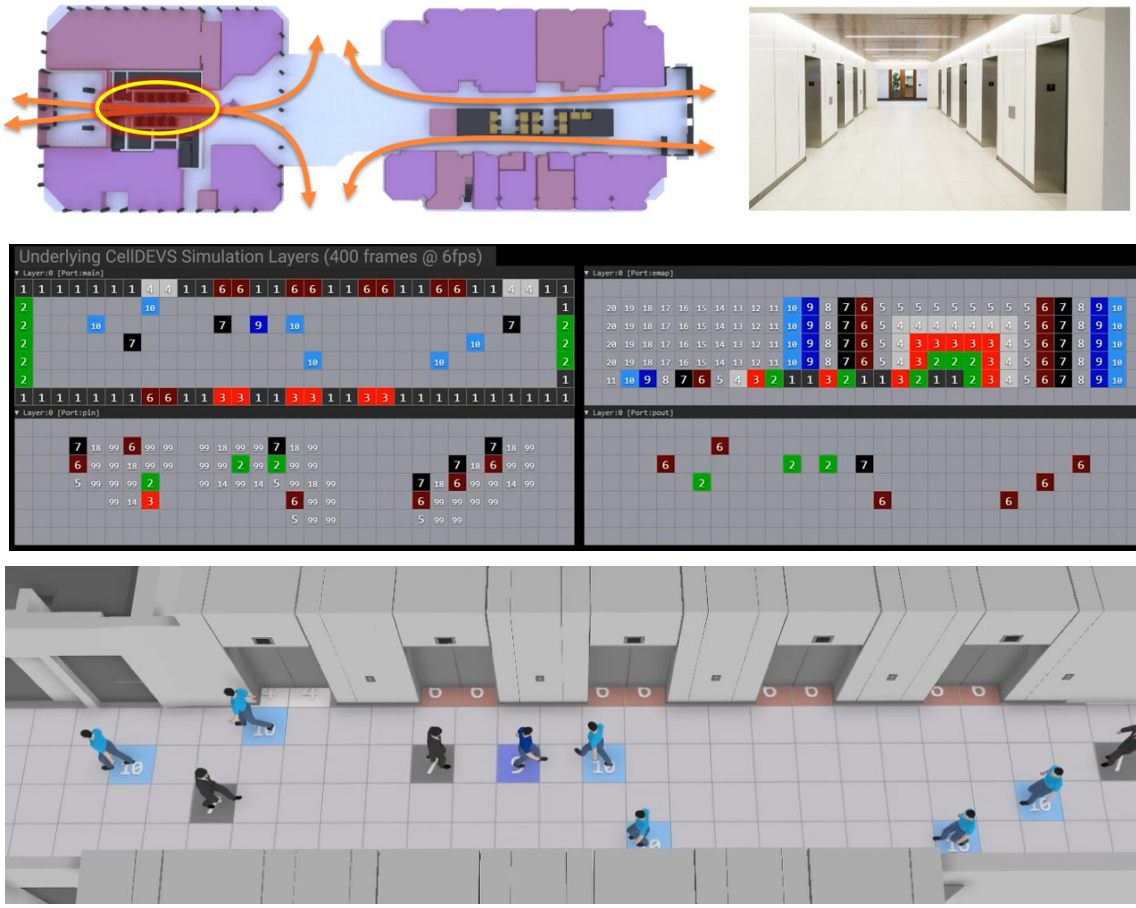


Figure 2.1 Cellular Discrete Event System Specification (Cell-DEVS [103]) used in [34] to model and simulate an elevator corridor of concern to the tenants of a commercial building (Sun Life, downtown Ottawa, Canada) which expected a rush of new foot traffic due to the opening of a public transit rail station in its underground. Top: the reference corridor of concern contextualized within the rest of the building and demonstrating its bottleneck potential. Middle: The layers of state variables used in the Cell-DEVS discrete-event simulation. Bottom: a 3D visualization of the underlying Eulerian simulation.

However, there is always a trade-off. With the aforementioned potential computational savings, there is a cost to maintaining the list of all scheduled events (internal events), coordinating the message passing between entities (messages that trigger "external events"), and the synchronized global termination detection (how to efficiently communicate to all the computationally asynchronous entities in the scene that all state updates in the current time-step have terminated and that it is safe to proceed to the next time step or event). Such challenges have been the subject of study in distributed

computing literature in general [38]–[40] and in discrete-event parallel simulation in particular [19], [41]–[43]. So, the question to ask when considering switching from discrete-time to discrete-event modelling then becomes: are the potential computational savings from an event-driven asynchronous state update policy worth the overhead of discrete-event simulation (timeline correctness, event book-keeping, and state transition coordination)? This particular question remains to be studied in the context of pedestrian and crowd simulation.

2.3 Crowd Density and Dissertation Focus

As this dissertation focuses on the simulation of dense crowds, let us clarify our notion of a *dense* crowd and how it differs from a sparse crowd. First, we outline the particular categorization of densities and their implications on pedestrian safety; then we discuss the global motion patterns and phenomenon that seem to consistently emerge in dense crowd scenarios, many of which will also be revisited in Chapter 4's discussion of our simulation results.

Studies have determined that conditions leading to crowd crushes and stampede disasters in what are otherwise peaceful gatherings can typically be traced back to the mismanagement of crowd flow, ultimately exceeding *critical crowd densities* [3], [19], [44]. Our interpretation of a dense crowd is based on the United States Federal Emergency Management Agency (FEMA)'s report on contingency planning [45]. From the perspective of a pedestrian, FEMA categorizes crowd densities as follows:

- $\sim 25\ \mathrm{ft}^2/\mathrm{pedestrian}$: normal walking speed and comfortable maneuverability.
- $\sim 10\ \mathrm{ft}^2/\mathrm{pedestrian}$: restricted movements and noticeably slower speeds.
- $\sim 5\ \mathrm{ft}^2/\mathrm{pedestrian}$: shuffling gait; calm motion as a group; difficult to overtake others.
- $\sim < 3\ \mathrm{ft}^2/\mathrm{pedestrian}$: brushing and close-contact with surrounding entities.
- $\sim < 2\ \mathrm{ft}^2/\mathrm{pedestrian}$: dangerous density with potential crushing injury.

For our purposes, a *dense* area of the crowd is one with $3\ \mathrm{ft}^2$ to $10\ \mathrm{ft}^2$ per pedestrian. At less than $3\ \mathrm{ft}^2$, it is considered a contact-collision with possible injury. We do note that our proposed method does not simulate the physics of contact-collisions or friction among pedestrians as prior art in rigid/soft-body simulation literature is already quite 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 global motion patterns.

Our primary application target is the simulation of crowds in film, gaming, and education/training scenarios. This requires the simulation to perform at real-time (simulation time advances at least as fast as wall clock time) or at interactive framerates (10fps, as user interaction experiments in the context of software usability have shown that 100ms response time was perceived as fluid or instantaneous feedback to user actions [46]). To be specific, the type of motion we aim to simulate pertains to the update of each pedestrian's position (their 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 [47]), presents a computational challenge for methods that share our goals. Separately, the high density presents a modeling challenge; where local agent-based rules would presumably reproduce global motion phenomenon, as observed in reality. We focus on the reproduction of four such phenomena that are commonly studied in crowd simulation literature:

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

Figure 2.2 A snapshot from an in-lab crowd capture experiment [89], demonstrating both the compression of personal space near the congestion, and the petal-like space-filling emergent pattern.

The last effect is perhaps the easiest to reason about: in denser areas, a pedestrian tends to move slower. In fact, that relationship has been recorded, analyzed and formalized; culminating in what is now known as the Fundamental Diagram [48], a macroscopic measure of the crowd's density-speed profile. Chapter 5 further discusses the use of this macroscopic measure as a metric for validation purposes.

Another phenomenon is the emergent petal-like space-filling pattern, where each entity stands roughly behind the midpoint between the shoulders of two entities right in front of it (e.g. Figure 2.2). One explanation is that each entity is optimizing the utility of the shared limited space [49]. Being packed in such a formation, there is less space being wasted compared to entities standing directly behind each other. Another reason might be the entity's improved visibility of its target motion vector (i.e. it can better see where it's heading; or have a better view of a point interest like the stage at a concert).

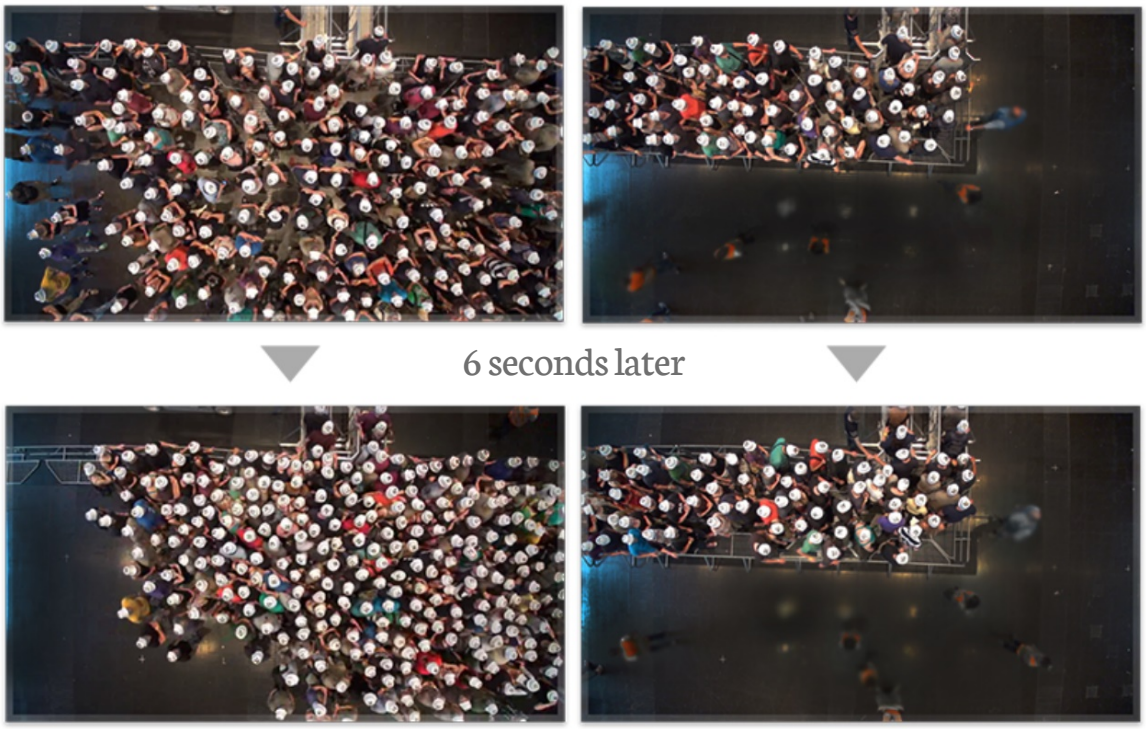


Figure 2.3 An in-lab trajectory capture experiment [81], where two crowds of the same count are asked to evacuate through the narrow exits. The left scenario is without guiding barriers, while the right scenario utilized barriers to passively shape the crowd.

Figure 2.2 also demonstrates the compression of personal space near the congested doorway. Existing microscopic crowds struggle to reproduce this effect due to their modeling of personal space as a 1D rigid separation distance, leading to artificial congestion and jamming, when captured crowd data indicates otherwise. We instead will propose, in Chapter 3, a compressible 2D area-based model of each entity's personal space, and will demonstrate in Chapter 4 how that local avoidance model is able to recreate such emergent compression effects in both dynamic and stationary crowds, in comparison to the struggles and jams produced by 1D rigid separation models. Personal space compression is not just a matter of personality or personal preferences; the barriers in the environment also play a strong role.

For example, Figure 2.3 shows how 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

Figure 2.4 A stationary crowd exhibiting the gradual decrease of personal space near a point of interest; which in this case is the marathon start line. Similar patterns can be observed near a live concert stage. (Licensed from iStock).

exit gates. This illustrates the accumulative nature of personal space compression and how restricting the possible angles of such accumulation can reduce the compression. The compression of personal space can also be observed in largely stationary crowds, such as at concerts or at marathon start lines (Figure 2.4). We note that at these densities, we are not considering psychological or personality-driven factors. We are simply observing the nearly biomechanical collision avoidance response vectors that pedestrians tend to exhibit in such scenarios. That biomechanical nature of personal space preservation is further explored in Chapter 3.

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 2.5 illustrates a color-coded in-lab bidirectional scenario where the entities are not forming lanes due to any explicit laning rules, cultural norms, or barriers in the scene, but rather it is a global phenomenon that emerges from local optimization decisions. Consider that with each foot step, an entity wants to minimize the deviation

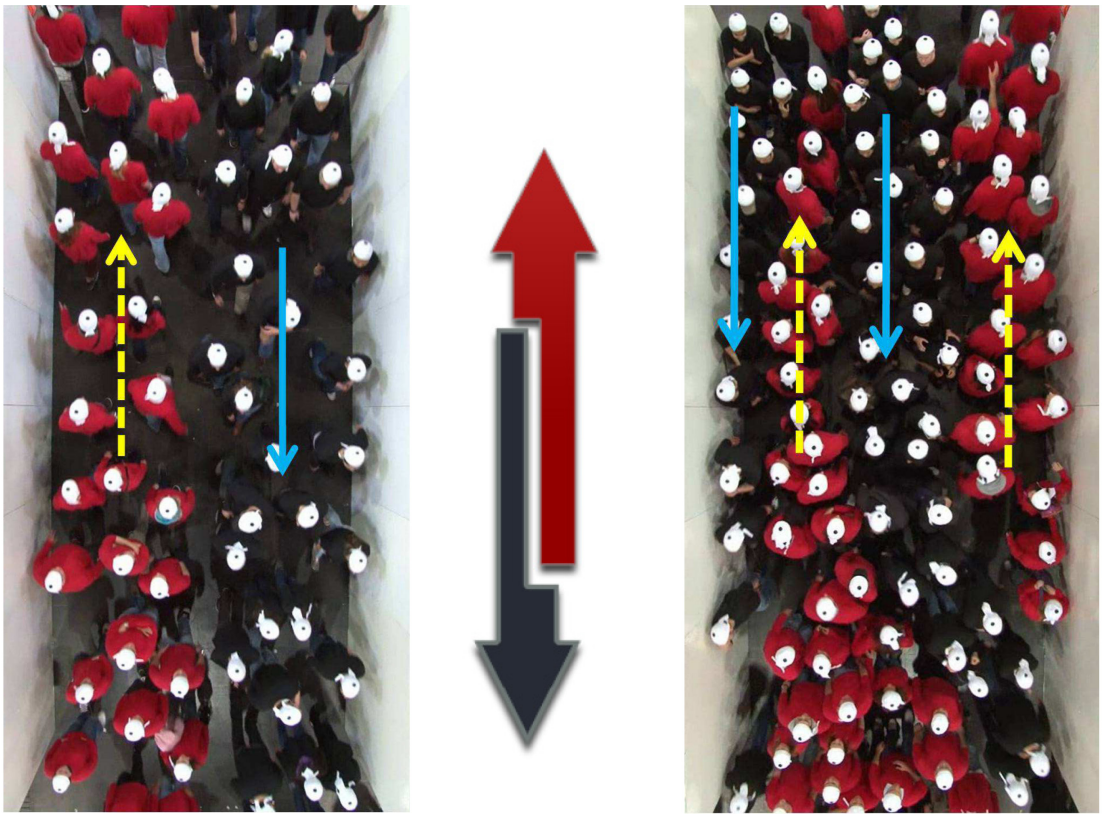


Figure 2.5 As demonstrated in lab experiments on bidirectional flow [89], lanes emerge from the collective motion of the crowd, with each entity trying to optimize its path, finding vectors of least resistance to its intended target and avoiding head-on collisions with on comping traffic. Left: *stable* lanes emerge when entities are not asked to seek a specific exit point along the width of the corridor. Right: *unstable* lanes emerge when entities explicitly seek a specific target exit point. Existing simulation methods struggle to reproduce unstable lanes.

from its target path (the one it would've taken had their been no dynamic obstacles in its way), and it wants to minimize the potential for collision with oncoming traffic. Thus, 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. As this local agency ripples across the crowd, lanes start to emerge and form globally.

The ability to correctly predict and simulate this effect in large-count high-density scenarios is challenging, but if done correctly, it can aid in contingency planning and occupant safety preparations

Figure 2.6 Example of a large-scale crowded event with dense, predominantly bidirectional, pedestrian traffic (Oktoberfest, Munich. Licensed from Intrepix/shutterstock.com).

at large-scale events, such as the busy festival scenario shown in Figure 2.6. As we learned earlier, barriers in the environment can have a positive effect on crowd density, and a virtual simulation tool can allow the experimentation (or at least cursory exploration) of barrier designs and different placements of points-of-interest for optimal occupant experience and safety.

However, it is important to keep in mind that barriers can also be misused (intentionally or otherwise) to actually cause severe harm to the crowd. If the net pedestrian influx flowing into the scene is not properly managed, then barriers can have an adverse effect as they start to behave as density funnels, which eventually keep trapping more and more entities within their bounds. And as we recall that the compression of personal space is accumulative; the barriers then create the conditions for exceeding the

critical crowd densities (i.e. less than 2 ft^2/pedestrian) potentially leading to injury or loss of life. This negative and often tragic outcome has occurred in crowded large events [2], [3] which were otherwise peaceful but involved the mismanagement of crowd influx, which let barriers like the seating area fences in the Hillsborough stadium (at Sheffield, England; 1989) and the tunnels at the Love Parade festival (at Duisburg, Germany; 2010) create exactly the kind of density funnel that led to injuries and fatalities due to crowd crushes and stampedes. In Chapter 5, we present a few opportunities for future work that would enable the application of our proposed method in such safety-critical applications.

Microscopic methods typically excel in sparse crowd simulation. RVO and ORCA in particular have enjoyed success in real-time multimedia/gaming engines [50]. However, they struggle to reproduce convincing trajectories for dense crowds. To tackle dense crowds, one approach is to model aggregates of local crowd flow instead of the trajectories of individual entities [51]. While this approach allows for real-time simulation of thousands of entities at interactive framerates, 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 [52], short-range stochastic motion-prediction based on prior collision experiences [53], position-based dynamics which adapt existing fluid and soft-body physics solvers for use in crowd simulation [54]. Implicit Crowds is a particularly interesting recent development that allows for smooth trajectories using much larger time steps than is required from typical numerically simulated crowds [55].

As will be demonstrated in Chapter 4, many of those methods end up with incompressible artificial congestion that does not match how people tend to gradually concede their personal space in an attempt to ensure as continuous of a motion as possible. And because many of those methods are based on analysis in the velocity-space [56], they cannot reproduce that compression phenomenon in largely

stationary crowds (e.g. near a concert stage, or at a marathon start line, as shown in Figure 2.4). We revisit the shortcomings of those existing methods in Chapter 4's discussion of our simulation results.

To summarize, we are seeking an improved local collision avoidance model that reproduces emergent dense crowd phenomena and is suitable for use in real-time simulation of thousands of entities in film, gaming, and educational media. Those phenomena include lane formation in bidirectional flow, compression of personal space near congestions and near areas of crowd interest; which current microscopic methods struggle to reproduce correctly. Chapter 3 will outline our proposed collision avoidance model, while the evaluation of our results (and comparison with state-of-the-art) is explored in Chapter 4.

Chapter 3

Centroidal Particle Dynamics

The intuition that launched this research was the simple hypothesis that close-range collision avoidance is fundamentally about finding the optimal direction which, when taken, would allow the entity to regain more of its personal space than by stepping in any other direction. To simulate close-range crowd dynamics, we propose a personal space (PS) preserving method we call Centroidal Particle Dynamics (CPD) [22]. As a variant of the social forces crowd simulation model [18], the CPD method models close-range interactions of pedestrians in dense crowds by explicitly asking them to step in the direction that would best maintain and attempt to regain the personal space in their vicinity (~0.8m-1.0m). The study of this intangible "personal space" often occurs under the umbrella of proxemics, a sub-field of social sciences that is focused on non-verbal spatial communication [57].

Earlier, in Chapter 1, we stated that overall human path planning simulation is fundamentally an exercise in stochastic abstraction, due to the apparent non-determinism of steering decisions and other sociopsychological factors which would become too complex to use as a basis for our dense crowd model. However, when we focus our attention on the avoidance behaviour at very short-ranges (the aforementioned PS area), the reference system we're trying to model turns out to be a lot more biomechanical and deterministic than the systems deriving long-range (global) and medium-range (within vision distance) path planning decisions. Medical studies have discovered the specific cause for the phenomenon of personal space preservation, proving that it originates and is regulated by the Amygdala, the fear center of the brain [58]. That is to say that the preservation of immediate personal space is a fear response rooted in physiology, not psychology. The studies confirmed their hypothesis by observing the lack of personal space preservation in patients that had damaged or missing Amygdalae.

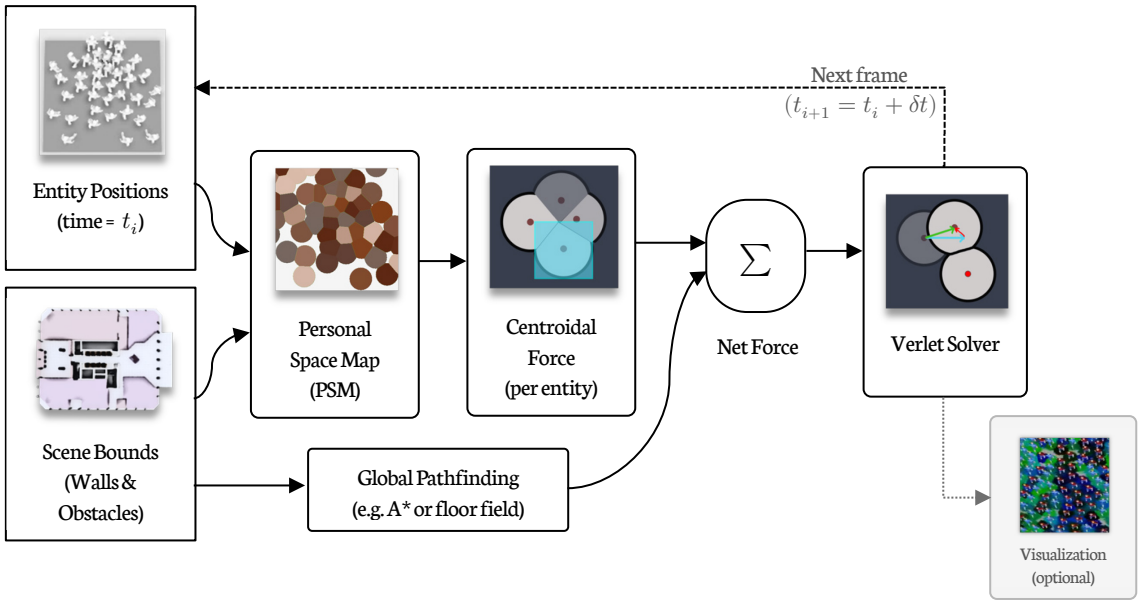


Figure 3.1 Overview of proposed pedestrian update cycle (per time step δt).

One evolutionary explanation indicates that personal spaces are a mechanism for subconsciously affording us (and mammals in general) a buffer of time to react to potentially negative outcomes, especially near strangers and in large crowds. In an interesting display of "nature vs. nurture", the specific PS radius varies across cultures and social settings [59], but the shared biological origin could explain the near universal radius of ~0.8m-1.0m.

3.1 Overview

An overview of the proposed method and its subcomponents is shown in Figure 3.1. We start by using the entity (pedestrian) positions to construct a *Personal Space Map* (PSM). Each entity is surrounded by an area of personal space (PS); and when two entities get close to enough to each other that their PS areas overlap, then we refer to that as sharing (or violating) the personal spaces of each other. The generation of the PSM is then a global operation that explicitly tessellates the scene's ground area to map (e.g. via color-coding) the currently unshared personal space (PS) areas to each entity it belongs to. Each pedestrian can then examine the local area in their immediate surrounding (~0.8m-1m) to calculate

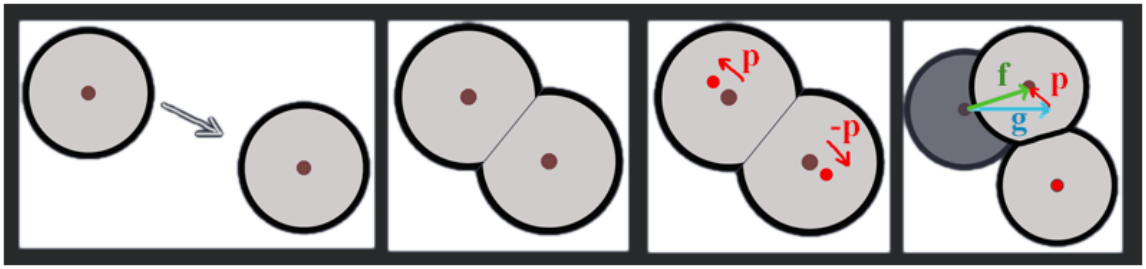


Figure 3.2 Sketch of the proposed force: net force (f) experienced by an entity is a linear combination of the global pathing force (g), and a penalty force (p) which falls along the direction of the new centroid.

how much of their personal space was violated and the appropriate response. This is done by computing the new geometric center (or centroid) of the currently unviolated personal space. A vector pointing the pedestrian to this new centroid is called the *Centroidal Force*. In essence, this force vector, if followed, will allow the pedestrian to regain the most amount of personal space when compared to stepping in any other direction.

Once the local (centroidal) force is computed, it is integrated with other forces relevant to the pedestrian (e.g. global path, friction, maintaining proximity to nearby family members). Such forces can be given weights, which can be treated as the parameters of the overall pedestrian simulation, as will be covered in section 3.6. Lastly, the advection due to the net acceleration experienced by each pedestrian is integrated using a typical numerical solver. We opt for a Verlet (symplectic) integrator as a happy medium between an explicit solver's computational efficiency, and an implicit solver's energy conservation and numerical stability [60]; with no additional constraints, time- or variable-wise.

3.2 Personal Space (PS)

Each entity is represented by a particle in the 2D plane surrounded by a continuous area of personal space (PS). Studies in France and North America have shown that the average adult PS is ~1.0m evenly around the center of the entity [23]. This number vary slightly across cultures [57], [59] within ~0.8m-1.0m. In addition, as the entity gains velocity, there is a further elongation of the personal space in the

Figure 3.3 A subsection of a larger crowd. Middle: the crowd's 2D personal space map (PSM). Right: the PSM visualized as an underlay.

direction of travel that is directly proportional to the speed. That is, humans typically expect an increasingly empty area ahead of them as they gain speed.

When two entities get within close proximity to each other, we assume they equally share (or violate) each other's personal space. We also assume, as experiments have shown in [23], [61], that there's a short delay to human response, requiring a brief time (~150-350ms) to react to events in their surroundings and enact their collision avoidance manoeuvres. From this view, we propose an area-based penalty force that reacts to the PS violation in an iterative manner, attempting to restore the preferred PS area over several frames. The idea, briefly sketched in Figure 3.2, is formally explained in following subsections.

Typically, there's a primary directional force that moves the entity towards its destination, and it is given by the global pathing scheme. The assumption is that the entity would reach its goal if it follows that vector, given that there are no obstacles in its path. The penalty force is added to the global pathing force, resulting in the reactive behaviour of our crowd.

3.3 Personal Space Map (PSM) Construction

During the simulation, for any point in a given entity's PS to be considered unviolated, it must be closer to that entity than to any other entity. The concept of sharing a space equidistantly as such evokes the

tessellations produced by the Voronoi Diagram. In fact, our definition of shared space spaces can be geometrically represented by a truncated Voronoi tessellation [62]. This tessellation does not need to be computed pair-wise; it can instead represent an aggregated account of all PS violations that an entity experiences in its local neighbourhood. We call the resulting global map the personal space map (PSM), which clearly outlines all PS violations across the crowd. An example PSM is briefly sketched in Figure 3.3.

The PSM is defined as a tessellation or partitioning of a 2D plane G (i.e. the ground) on which scene obstacles exist (walls, gates, barriers, vehicles, etc.) and pedestrians traverse. After the PSM construction is done, every point g in plane G, will belong to one and only one entity (e.g. pedestrian_13, obstacle, or unoccupied space), as briefly illustrated in Figure 3.3. Let T denote the many-to-one mapping that tessellates plane G.

To start, all points $g \in G$ are considered *unoccupied* (or numerically 0) to represent all the empty space available for any pedestrian to traverse:

$$T(g) = \ 0, \quad \text{for all } g \in G.$$

As mentioned earlier, pedestrians will be accounted for by performing a constrained Voronoi tessellation with pedestrian positions as the Voronoi *sites*, and the personal space (PS) radius as the constraint. Let's assume that each pedestrian is assigned a unique ID from 1 to n. If we denote $d(a, b)$ as the Euclidean distance between any two points a and b on plane G, then the tessellation becomes:

$$T(g) = \begin{cases} i, & (d(g, p_i) < r_i) \ \wedge \ (d(g, p_i) < d(g, p_j)) \ \forall \ i \neq j \\ 0, & \text{otherwise} \end{cases},$$

where $i, j \in \{1, \dots, n\}$; p_i is the position of pedestrian i, and r_i is the radius of pedestrian i's personal space (PS).

Finally, scene obstacles can be explicitly defined by the modeler, for example set $B \subset G$ which denotes areas that the pedestrian needs to avoid. Additionally, scene geometry can be projected onto G as if viewed from orthogonally from the top. Most scene geometry are already in 2D form (e.g. architectural floor plans), but any 3D geometry (e.g. columns or vehicles) would need to be explicitly projected onto the PSM for ground-level collision avoidance. To project 3D meshes onto G, the following transform can be 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 0m (ground) and 3m (reasonable 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 the boundary points in space G that were defined explicitly by the modeler along with all the scene obstacle projections. When tessellating G, we -currently- don't explicitly differentiate between different obstacles, and hence assign them to be an *obstacle* (or numerically -1):

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

The entire PSM tessellation process is memoryless and gets reconstructed every time step in the same fashion. 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. Simulation of crowd motion in multi-storey buildings is possible by combining the normal vector modification with a layered PSM (one per floor) and an expected overlap in their stairs region. Alternatively, the stair regions could be simulated on separate PSMs altogether, if desired.

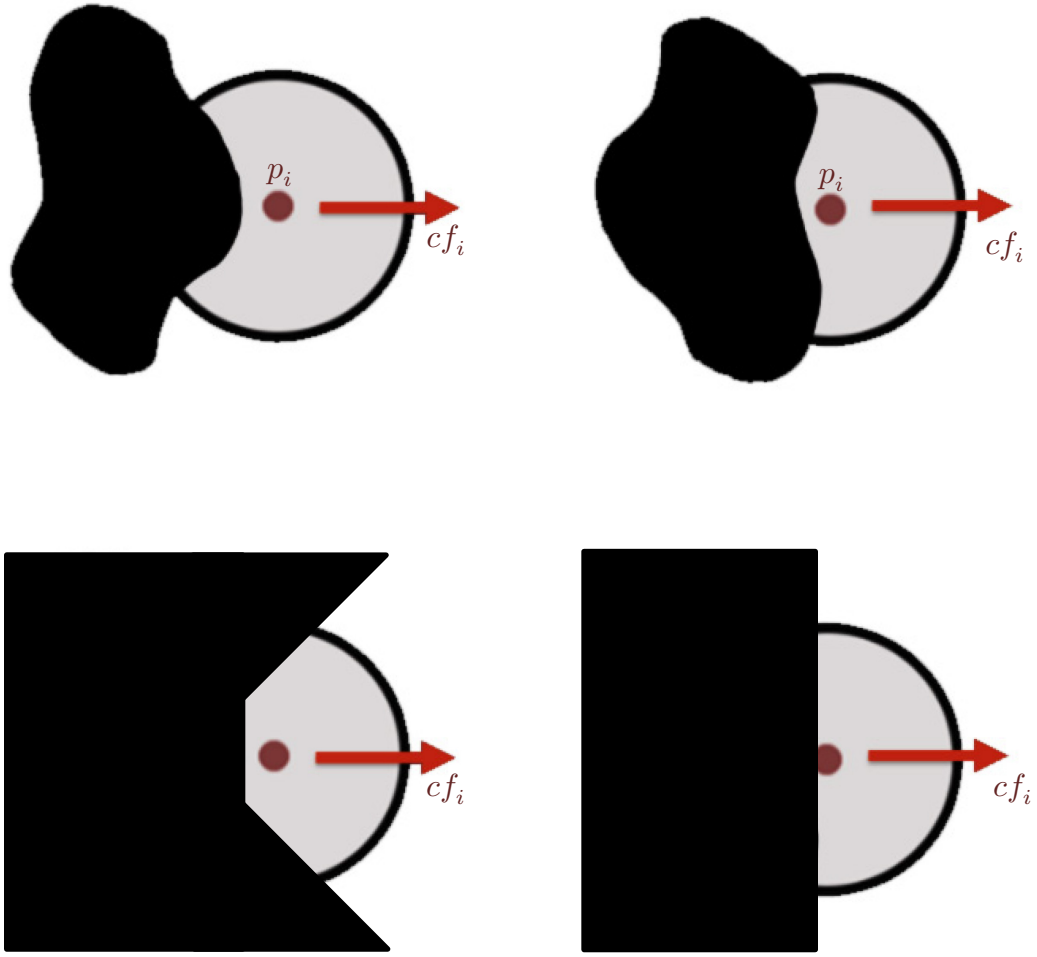


Figure 3.4 Because the centroidal force is an aggregate measure of PS violations, then regardless of the obstacle's curvature (axis-aligned, convex, concave, etc.), the centroidal vector $\boldsymbol{cf_i}$ should correctly point away from PS violations. Architecture and urban design simulation professionals can benefit from our method's flexibility which allows collision-avoidance with obstacles of arbitrary shapes (e.g. vegetation, furniture pieces, columns, etc.)

3.4 PS Centroids (Centroidal Force)

The PSM paints us a picture of the overall personal space (PS) that the pedestrians occupy. Each entity only needs to view its personal neighbourhood (within radius r_i) to compute quantities related to its personal space violations. One quantity we are particularly interested in is the PS *centroid*, i.e. the new center of mass of the current PS, which is mathematically the average position of all unviolated points in the PS area. The intuition here is that a person would try to move in the direction of the centroid in order to regain the most PS possible. We could view it as the direction of the vector avoiding the most violations of PS (Figure 3.4). In effect, pedestrians will be performing an iteration of the Lloyd algorithm [63], but on truncated Voronoi cells representing personal spaces. Normally, when an entity has all of its personal space unviolated, the centroid is simply the entity's current position. Let $PS_i \subset G$ denote pedestrian i's original completely unviolated personal space region. When another entity or scene obstacle encroaches on i's personal space, the new centroid position $c_i = (\tilde{x}, \tilde{y})$ is found using surface integrals:

$$c_i = \left(\frac{\iint_{PS_i} x \cdot \lambda_i(x,y)\ dx\,dy}{A_i}, \frac{\iint_{PS_i} y \cdot \lambda_i(x,y)\ dx\,dy}{A_i} \right),$$

$$A_i = \text{euclidean area of } PS_i = \iint_{PS_i} 1\ dx\,dy\ ,$$

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

where $sample_G(x,y)$ returns the current value of the PSM at $(x,y) \in G$. The centroidal force cf_i is then simply the vector from the pedestrian position to this new centroid:

$$cf_i = c_i - p_i$$

The actual implementation of the above integrals boils down to a couple of weighted summations performed over a 2D grid of PSM pixels, as discussed in section 3.7.

3.4.1 Asymmetric PS Weighting

Entities with the homogenous kernel 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. So far, the personal space (PS) kernel is a uniformly weighted homogenous area, which was based on the empirical studies conducted in [23]. However, we can further modify the local dynamics by changing the footprint's geometry or using a weight map. The first such modification is a kernel mask (or weight map), as shown in Figure 3.5, that splits the PS shape into two non-overlapping regions: i) an *active* PS space that affects both the entity and its neighbours; and ii) a *passive* PS space that only affects surrounding neighbours. This asymmetric PS shape has a couple of implications on the local dynamics:

- The net separation distance between a pair of entities in motion remains the same as before. The difference now is that instead of equally sharing the responsibility (i.e. response force), the entity with visibility of the PS sharing will shoulder most of the responsibility and corrective efforts to maintain that distance. This is analogous to a driver of a vehicle in the back being largely responsible for maintaining a safe distance from the vehicles ahead.
- This shift in responsibility, where one entity (one in the back) will experience a stronger response force, reduces the overall severity of PS compression around areas of congestion.

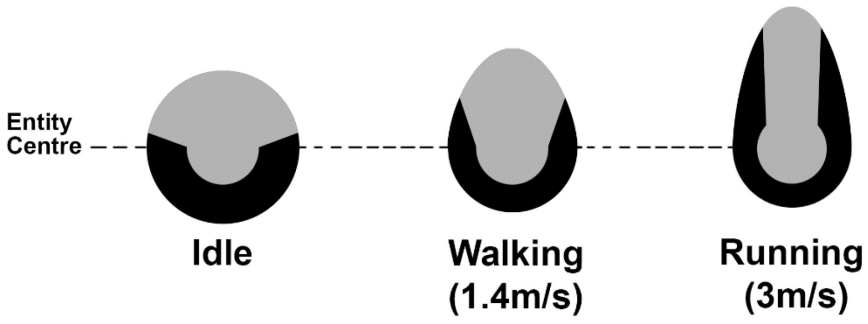


Figure 3.5 Proposed PS kernel shape: active (light) region affects the entity and its neighbours; passive (dark) only affects neighbors. Lengthening of PS kernel is towards direction of motion as demonstrated.

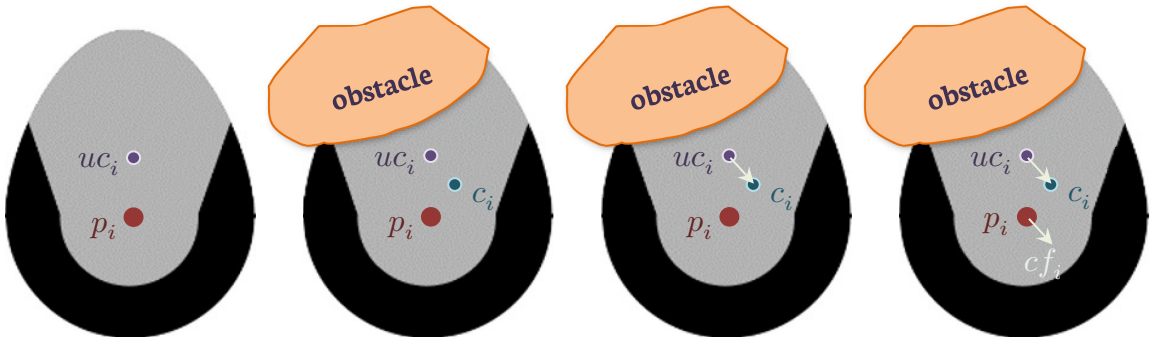


Figure 3.6 Example of steps to compute the centroidal force ($\boldsymbol{cf_i}$) when a walking entity experiences $\boldsymbol{PS}$ violation; given the entity's position ($\boldsymbol{p_i}$), unbiased centroid ($\boldsymbol{uc_i}$), and new centroid ($\boldsymbol{c_i}$).

Additionally, [23]'s suggested elongation of personal space proportional to the entity velocity was incorporated, 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 3.5 illustrates the base kernel reduction also in relation to speed, to account for the narrowing focus of speeding entities. The equation for computing the centroid now becomes:

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

$$A_i(s) = \text{euclidean area of active region of } PS_i \text{ given speed } s\ ,$$

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

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

The original centroid of PS_i's active region (uc_i) is now no longer guaranteed to be at the entity's position (p_i). So, the new centroidal force (illustrated in Figure 3.6) is computed as:

$$cf_i = c_i - uc_i\,.$$

3.5 Global Path Finding

We describe the global path as the path 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 such as walls and road blocks. The literature is saturated with methods in this area (e.g. A*, floor fields, AI navigation trees in games) [64] and so we assume this as an input into our net force integrator, as shown in Figure 3.1. If the scene obstacles are defined over a graph, then a method like A* path finding [65] would work well in our real-time environment, as it efficiently computes such global paths per entity and can be updated every reasonable interval (say 5 seconds) throughout the simulation. But for large scale simulations involving thousands of pedestrians with relatively few possible global targets within the scene (e.g. only dozens of shops within an event, as shown in Figure 2.6), a single global floor map per target is more computationally 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 [4], [44] for example implementations of this floor field.

Our proposed local dynamics method does not place limitations on the kind of global pathing algorithm used. This is a direct illustration of the *separation of concerns* discussed in the hierarchy in Chapter 1. However, as most of the contributions in this proposed thesis lies in the local dynamics layer, we wanted to first use the simplest forms of global pathing in order to reduce the influence of any "intelligent" path finding and evaluate our proposed centroidal dynamics as independently as possible. A sanity-check, if you will, before proceeding with more complex path finding techniques. In some experiments in Chapter 4 we discard global paths entirely to evaluate the emergent behaviour from local dynamics alone (e.g. aimless crowds experiencing overcrowding, and stationary crowds at a concert). The global pathing forces used in this thesis are time-invariant; they rely only on the current position of the entity in order to find the "next" step along the global path. Let's call this force $gf(p_i)$.

Prior to the force integration step, we compute a resistance to centroidal forces that oppose the entity's global path/objective. This was inspired by the energy-minimization goals set in ORCA [24], and it has reduced the "springiness" of near-miss collision in our pedestrian crossings significantly (especially in bidirectional flow). Therefore, even if the local centroidal force is pointing the entity to face away from the global path goal -because that is what is locally optimal- the entity will *resist* this change and attempt to wait until more favourable centroidal forces are available. This resistance force is called uf_i.

3.6 Net Force

The total force experienced by each entity is a weighted sum of the local forces (including the centroidal force) and the global pathing direction. In ideal conditions with a single entity in the scene, it would simply follow the current global path direction to get to its destination. However, with other entities in the scene, the local forces are necessary to enact collision avoidance maneuvers with their surrounding environment (other entities + obstacles). The net force calculation will be of the form:

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

where $gf(p_i)$ is the global path vector given i's current position in the scene (e.g. using A* or a floorfield); and α, β, and γ are scalar weights to parametrize the overall behaviour of the entity. For instance, an aggressive pedestrian might have low α and high β values, hence emphasizing their own global path with little regard for local PS violations (indicated by cf). This is essentially how our close-range pedestrian behaviour can be calibrated according to reference trajectory data. At this current stage, we've empirically arrived at a set of parameter values and illustrated their emergent results in Chapter 4. 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 comfort speed. On average, a comfort speed for walking pedestrian is $1.4 \pm 0.24\mathrm{m/s}$ [23] and is what we've used in our model.

It is possible to further refine the aforementioned parameters and automate their calibration using context-specific reference trajectory data. An interested party might be able to use existing parameter estimation platforms (e.g. [62]) which capture path planning and collision avoidance behaviour directly from footage/mocap data and perform looped optimization (or machine learning) to optimize any given simulation model's parameters to best fit with the input data. Having an automated calibration system would allow our pedestrian simulation to model behaviour that changes across cultures, event types, age groups, and unforeseen pedestrian contexts. However, such platforms encompass the overall motion of the crowd, and not just the local collision avoidance. As this is beyond the scope of this dissertation, more is discussed in Chapter 5.

3.7 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. The section outlines one possible implementation, developed throughout the course of this dissertation. The presented implementation produces real-time simulation results for thousands of entities in the scene (more on performance in Chapter 4).

3.7.1 Voronoi Diagrams in Prior Art

In the context of Lagrangian crowds, the Voronoi tessellation has been used to accelerate spatial queries like nearest neighbour search [21], [66], or developing a navigation mesh for global path planning [67]. Recent developments suggest the use of Voronoi diagrams to identify a discrete set of candidate local vectors to pursue, essentially, along one of the neighbouring Voronoi cell edges [68], [69]. While their approach (published concurrently to our own publications [22], [70]) shares the simplicity and intuitiveness of our thesis, it does not explicitly model an entity's personal space nor its compression, thus it's unsuitable for dense crowd scenarios or largely stationary ones (e.g. at a concert). To the best of

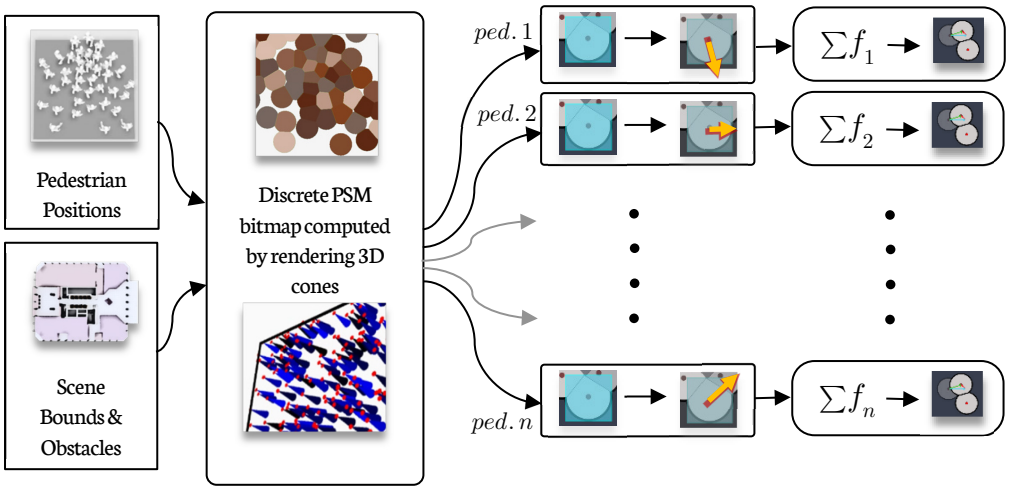


Figure 3.7 Proposed data-parallel implementation of Figure 3.1.

our knowledge, our proposed use of the truncated Voronoi diagram as an analog for compressible personal space is novel, and so is our intuition-motivated visual parameterization through dynamic weight maps, as proposed in section 3.4.1.

3.7.2 Discrete-Space Implementation of CPD using Truncated Voronoi Disks

A spatially-discrete version of the continuous PSM presented in section 3.3 can be created using a truncated Voronoi diagram, whose cells are bounded by a certain distance from their sites. To achieve the high performance desired in interactive applications, we propose the use of GPU-accelerated computation of Voronoi diagrams in [62] to produce the PSM, followed by custom shaders designed to compute the centroidal forces per entity and integrate them with other global forces before finally solving them for time step δt. Figure 3.7 outlines the procedure. In contrast with the continuous-space method described in section 3.4, the simulation space is implemented as a discretized 2D grid (a bitmap).

Textured 3D cones are used 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

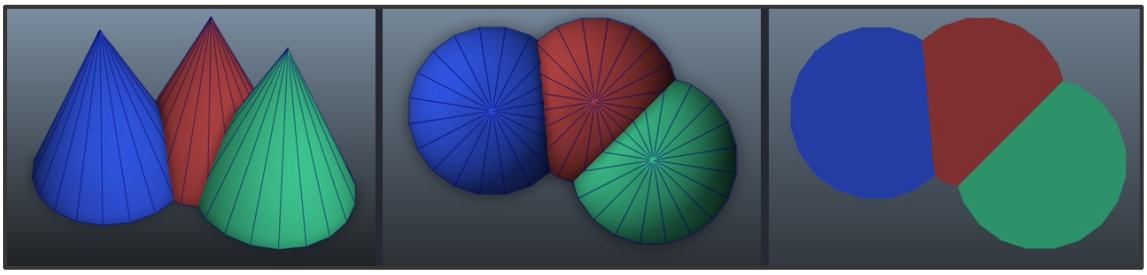


Figure 3.8 The PSM is implemented as a 2D bitmap composed of truncated Voronoi cells created by a top-view orthogonal projection of 3D cones; underneath an obstacles overlay.

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 [62]. Figure 3.8 visualizes this procedure. By encoding the entities as geometric primitives and using the GPU's depth buffer to quickly obtain the PSM tessellation as discrete pixels, 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 3.7). The entities do not need to conduct any additional costly nearest neighbor search, as they simply consume and interact with the set of pixels representing their PS in the PSM.

In order to differentiate between the rendered cones, they're colored using a one-to-one reversible hash map that is a function of the unique entity IDs. For our purposes, we reserve the first ten colors in the 24-bit RGB space for obstacles and debugging purposes then utilize the rest to uniquely color code each entity. The reverse lookup (which is also a constant cost function) enables any entity to directly identify the ID of another entity infringing on its pixel space. See Appendix A. for a sample hash code.

Theoretically, we could simulate more than 16 million entities on a 24-bit RGB PSM surface, and potentially billions of pedestrians on higher bit-depth surfaces. Though, practically speaking, 16 million entities could only be afforded a single pixel each on a 4k×4k PSM map, as a best case-scenario. More on such scalability issues is discussed in section 3.7.3.

To compute the true center of PS footprints that have an influence map, vary by time, or are proportional to the entity's speed (as discussed in the section 3.4), 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 in order to detect true violations of the PS area.

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

a) Per-pixel aggregation - if there is overcrowding resulting in significant PS overlap, and then it is more economical to only count the visible pixels and aggregate the results using the reverse hash lookup.

b) Per-entity aggregation - in sparser scenarios, it's 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 10×10cm, so our PS radii can range from 8 to 10 pixels, as sizes vary in a crowd. See Appendix A. for a sample implementation of both options.

3.7.3 Scalability

GPUs are highly 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 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. The 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 neighbouring 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). But 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. Similar to how we skip empty pixels in sparse crowds, we extend the concept by skipping entire tiles if they're empty, and only processing 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 like Khronos Vulkan and Microsoft DirectX12 which allow and encourage the multi-threaded dispatching of render-calls [71] supports the scalability potential of our presented method. CPU threads could concurrently launch and synchronize GPU-computable PSM tiles without costly context-switching (a required cost for the currently ubiquitous graphics APIs: OpenGL/WebGL/DX11).

3.7.4 Density Estimation

Computing a local density is an important feature to be able to conduct flow rate validation 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, but this time color every entity i using the estimated d_i, where brighter values correspond to higher densities. We then smooth the discontinuities using a

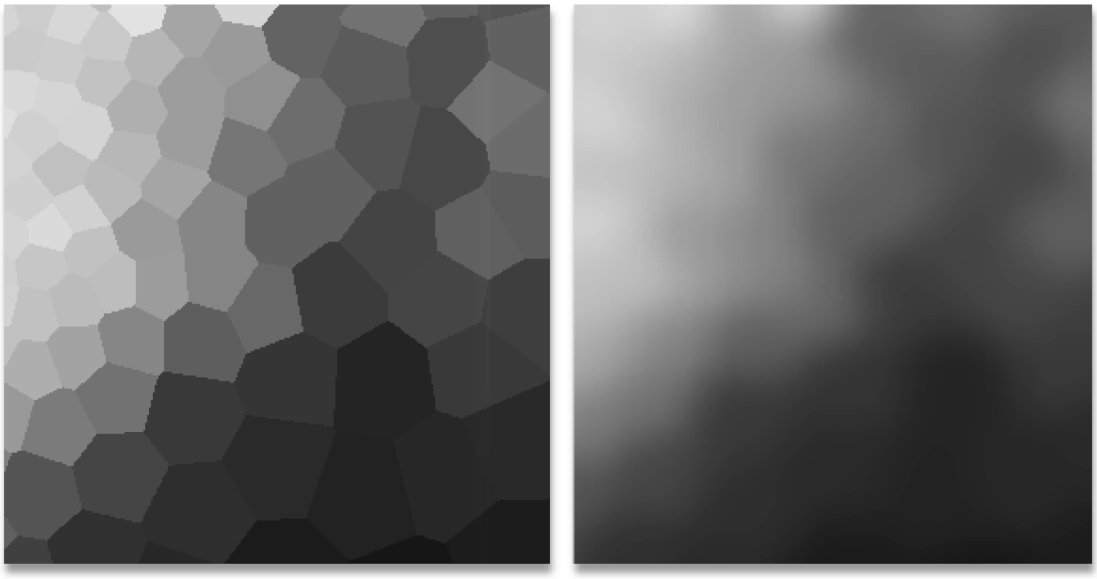


Figure 3.9 Local density is estimated per entity (left) then a global low-pass filter (right) is applied to obtain a smooth "sampling-friendly" field.

Gaussian blur filter; a very standard and parallel-friendly utilization of the graphics pipeline. The resulting smooth density field is shown in Figure 3.9.

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 [72]. However, we opted for the simpler 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 *isn't* too precise since the fundamental diagram is only an aggregate of human behaviour 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 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 $\mathrm{O}(nm^2)$ computation to a more linear $\mathrm{O}(nm)$, where n is the number of pixels in space, and m is the radius of the filtering kernel.

3.7.5 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 [22]. 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 relatively slow system bus. Unfortunately, this feature could not be naively 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, comfort speed, etc.) and lets the GPU generate the appropriate voronoidal PS shapes per entity. This reduction in CPU calls has improved the framerate, as shown in Chapter 4.

3.8 Visualization

The shader computation allowed room for real-time 3D sprite rendering. Our simulation could potentially be visualized using existing character generation methods, which generate smoothly rigged and GPU-animated characters [22] at interactive frame rates. One could also import the crowd trajectories into game engines for interaction with other non-crowd elements.

To demonstrate a self-contained simulation and visualization environment for this dissertation, we opted 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 skeletal smoothed rigging. The resulting 3D sprites are sufficient for rapid prototyping and iteration, with visualization overall consuming on average 15-30% of each frame time.

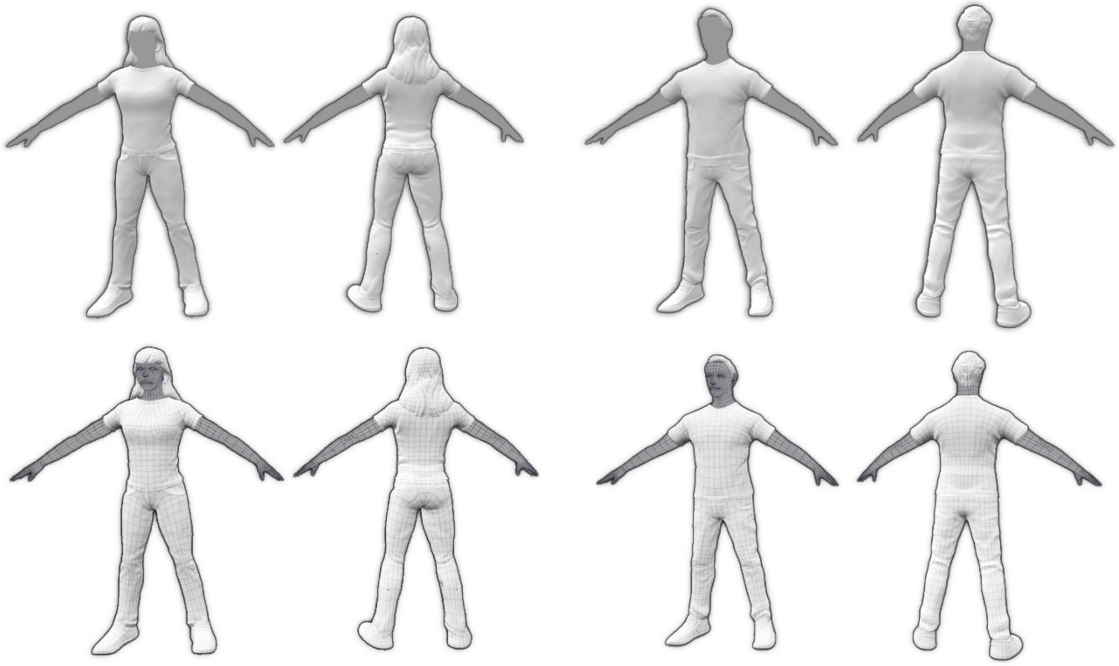


Figure 3.10 3D character meshes used for pedestrians. The polygonal topology has been optimized to create convincing contours (silhouettes) while reducing polygonal counts. Additional detail can be added through texturing and hair mesh variations.

Chapter 4

Emergent Crowd Dynamics

Centroidal Particle Dynamics (CPD), as proposed in Chapter 3, is an agent-based collision-avoidance method; and in this chapter, we explore the global crowd dynamics that emerge from such locally-define agent-based rules. The simulations illustrate various scenarios of dense crowds and associated obstacles; with a particular focus on emergent crowd phenomena seen in high density scenarios (as discussed in Chapter 2). The chapter first describes the general simulation setup (sections 4.1 and 4.2) then discusses each phenomenon in the remaining sections. Throughout the chapter, we compare CPD's emergent results to other methods; before summarizing those comparisons in Table 4.1.

4.1 World Definition

Our virtual world is defined as a 3D *scene* containing a flat *ground* on which the motion of a *crowd* of *pedestrians* (or *entities*) will be simulated. The pedestrians share their ground space with static scene *obstacles* (e.g. walls), dynamic elements (e.g. gates), and dynamic pedestrian accessories (e.g. strollers, shopping carts, etc.). The invisible computational "backend" of the simulated world includes the PSM top-projection camera and the rendered PSM it generates every time step. The motion of every entity is solved using a symplectic verlet solver, which delivers an acceptable level of conservation of energy at fast computing speeds. Verlet integration is a form of semi-implicit integration [60], which performs nearly as fast as the explicit Euler, but without introducing additional energy, and hence providing long-term stability in the scene and reducing error propagation over simulation time.

Ideally, the ground shares the same aspect ratio as the PSM being computed on it to ensure isotropic computational fidelity regardless of which direction the pedestrians are facing in the scene. We

maintain an agent-based approach, where each entity encapsulates its own data and never has direct access to neighbour entities' data.

4.2 Simulation and Model Setup

All simulations were run with discrete-time integration, using a quantum of $0.1\mathrm{s}$ per frame as we did not notice an improvement in quality by going with a finer time delta. Each pixel side length represented $10\mathrm{cm}$ of physical space. For each scenario, we initialize entity positions, orientations, and randomize a few parameters such as weight, size, and a base comfort walking speed ($1.4 \pm 0.24\mathrm{m/s}$). For demonstration and testing purposes we use simple global pathing schemes such as A*, floor maps, or global vectors to target positions [65].

The entity PS base radii were kept at 9 pixels (plus 1 center) to achieve the ~0.8m-1.0m PS idle radius. The shaders described in section 3.7 were implemented using the OpenGL Shading Language (GLSL); a sample of which can be found in Appendix A. Parameters were randomized across the crowd, including the PS radius, comfort speed, and force parameters ($\alpha, \beta,$ 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. further 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 [2], [3]. The obstacles could be procedurally drawn during runtime; or loaded from a bitmap (e.g. architectural floor plans). Regardless of the source, the RGB24 color space (8 bits each with range of values 0-255) used for the final PSM is split as follows:

- Obstacles: (0, 0, 0) to (0, 0, 5)
- Debug: (0, 0, 6) to (0, 0, 10)
- Pedestrians: (0, 0, 11) to (255, 255, 255)
    - Calculated using a one-to-one map (Appendix A.).

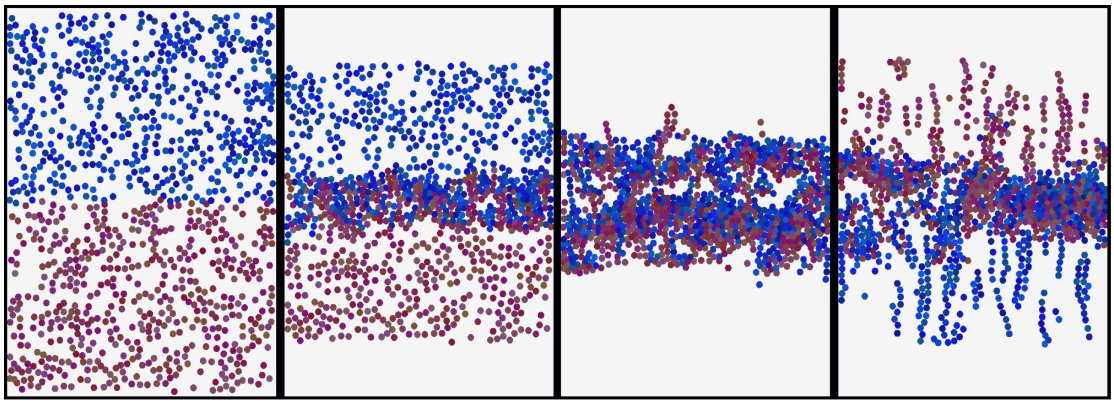


Figure 4.1 Natural lane formation during a bidirectional flow simulation of 1000 entities on a 600×800 PSM grid (blue entities headed south; red headed north).

4.3 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 [73]. 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; and 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" anywhere).

Figure 4.1 shows a bidirectional scenario where our method is capable of reproducing lane formation using the symmetrical PS kernel from section 3.2. But when utilizing the asymmetric PS kernel from section 3.4.1 (which, to recall, takes into account 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, and have a better resemblance to the lanes formed in real-life bidirectional flow. This is illustrated Figure 4.2, which shows a top view of our simulation of bidirectional flow of a dense crowd (1000 entities) in a wide corridor.

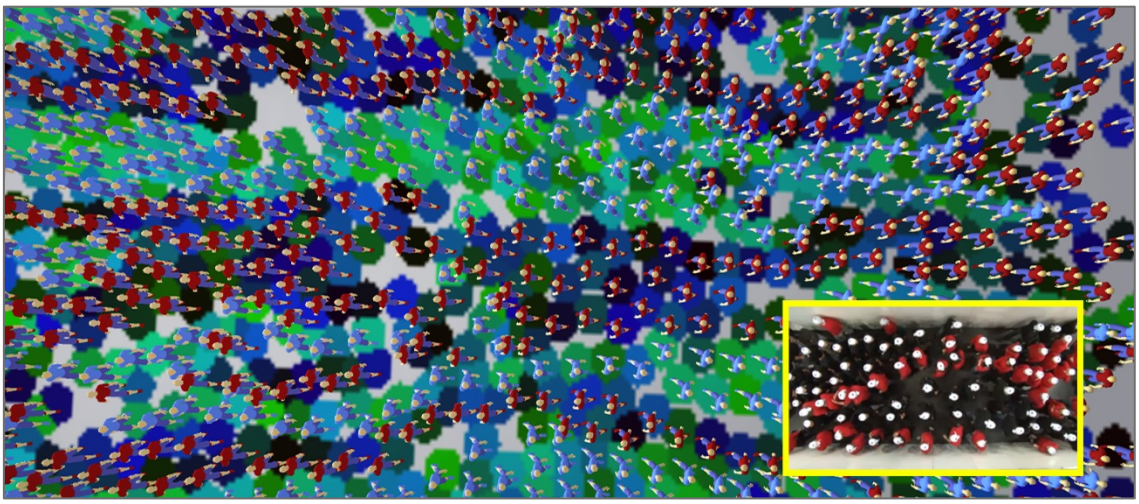


Figure 4.2 The emergent lane formation produced by CPD pedestrians in a dense bidirectional flow scenario with visually similar branching/mergin patterns to those observed in reality (bottom right shows a still frame of real footage [89] of bidirectional flow in a corridor). The entities in both our simulation and the real footage are color-coded to indicate direction of motion (east-west).

Figure 4.3 and Figure 4.4 shows bidirectional flow of the state-of-the-art methods (WarpDriver [28], ORCA[50], RVO [24], [25], [50], Social Forces [18], and Position-based [54] crowds) that share our application target of real-time crowd simulation for film, gaming, and serious education. When compared to our CPD pedestrians (Figure 4.1, Figure 4.2), those methods either fail or struggle to reproduce lane formation in dense crowds. There is a lack of agreed upon metrics to measure that statement quantitatively. Cross-sectional velocity distribution charts [74] and average velocity grids [75] have been previously proposed to quantify the flow; but in our view they aggregate away many of the microscopic details of the flow dynamics that motivated us to pursue this topic in the first place. Instead, we'll first do a cursory analysis of the observed quality of the emergent lanes compared to real footage, then examine the microscopic trajectory traces, which paint a richer picture of flow dynamics.

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 is able to 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. SocialForces, ORCA, and PowerLaw 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're unrealistically wide and eventually clump into congested and awkwardly aggregated motion patterns, where groups of entities appear to move precisely in unison during lane formation.

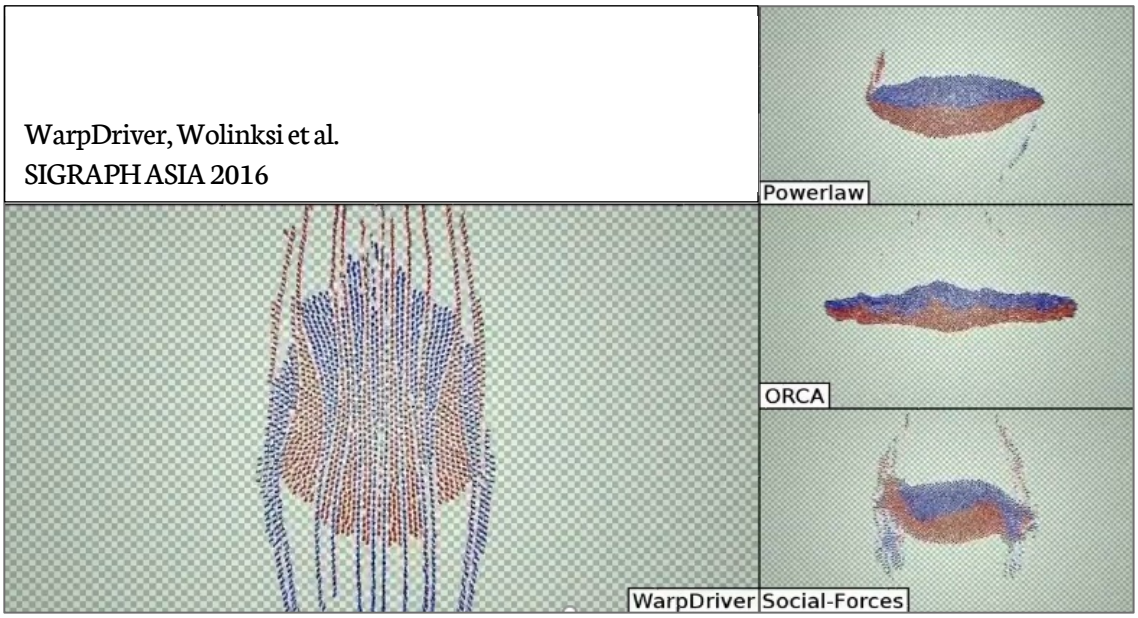


Figure 4.3 WarpDriver [28] (left) and other real-time crowd simulation algorithms [18], [25], [56] struggle to produce organic laning seen in real footage compared to our results in Figures 4.1- 4.2. Image captured from WarpDriver video: https://www.youtube.com/watch?v=WPPrIz39wQk (*for academic use only*).

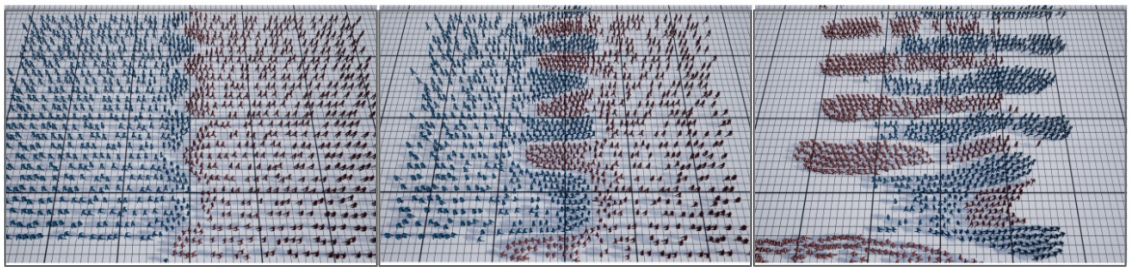


Figure 4.4 Position-based crowds [54] is another recent method that struggles with reproducing organic laning. Their motion trajectories are either too aggregated or suffer from excessive artificial congestion. Image reproduced from [31] *(for academic use only)*. Video available at: https://www.youtube.com/watch?v=gTGCVP4Amtc

The quality of the emergent bidirectional lanes can also be assessed over a time period of simulation; and diagrams that trace the trajectory of each pedestrian can capture an overall snapshot of such dynamic patterns over time. Figure 4.5, obtained from [49], shows a trace of the trajectory of a real bidirectional flow scenario in a 3.6m wide corridor. 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* [49] as they vary spatially and temporally. Larger-sized experiments are difficult to construct and coordinate (e.g. this narrow corridor experiment required over 300 volunteers). So, it is reasonable to expect that the data capture of a bigger bidirectional flow scenario (like the one in Figure 2.6) under controlled lab conditions would be challenging.

Simulation can become a useful tool here. The trace shown in Figure 4.6(left) illustrates how our model was able to replicate the emergent lane formation pattern in a 3.6m virtual hallway, appearing nearly congruent to Figure 4.5's observed trace. We then extrapolated the CPD scenario by modeling a wider 6.5m virtual corridor. Given further rigorous statistical validation (see discussion in Chapter 5), our method could facilitate urban design and safety-critical planning for large-scale crowded events [29].

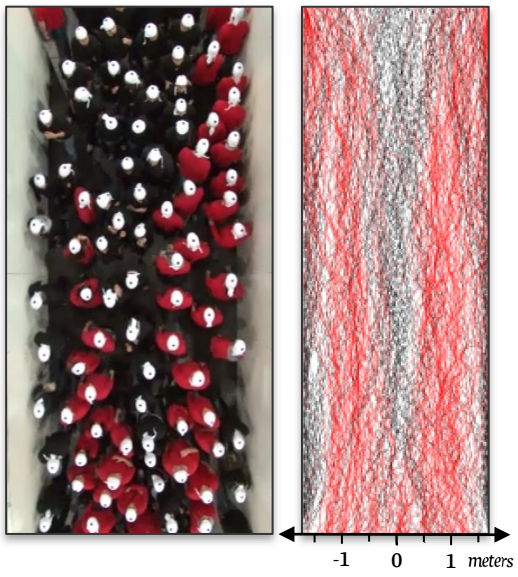


Figure 4.5 Trace of real pedestrian trajectories in a narrow bidirectional corridor (obtained from [49]).

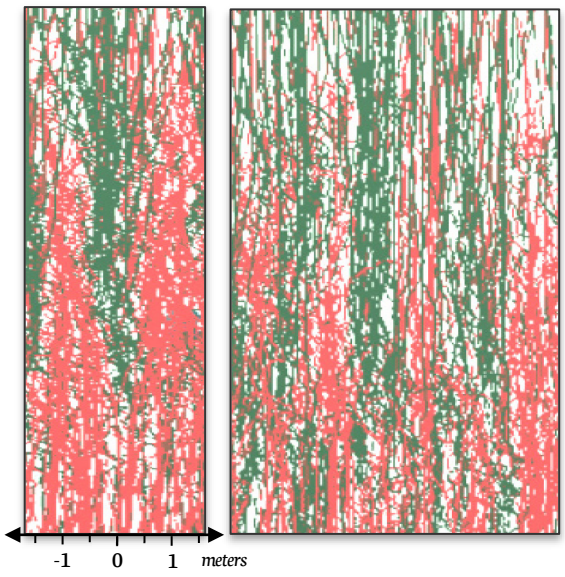


Figure 4.6 Trace of our CPD virtual pedestrians simulated in a narrow 3.6m (left) and a wide 6.5m (right) bidirectional corridors.

4.4 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, while 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 [45], 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, and simulation can help identify pockets of potentially unsafe accumulation and overcrowding of attendees [3]. In Figure 4.7, a virtual crowd of 1000 entities in a confined space is reacting to the fact that they are over-crowded. Requiring neither a cognitive model nor a global pathing scheme, the centroidal response force is sufficiently able to naturally and gradually restore the compressed crowd to a more comfortable distribution, filling the room if necessary, until every entity reaches a suitable local density (assuming a cooperative crowd with no disruptive members, as we'll examine in later sections). 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

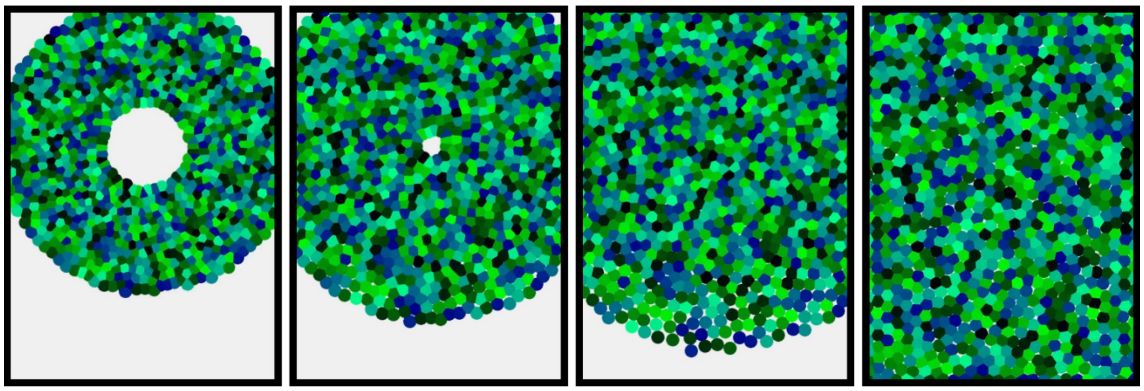


Figure 4.7 Penalty forces are sufficient to cause an overcrowded room (left) to diffuse to a more comfortable equilibrium (right), where any further motion would not improve the situation.

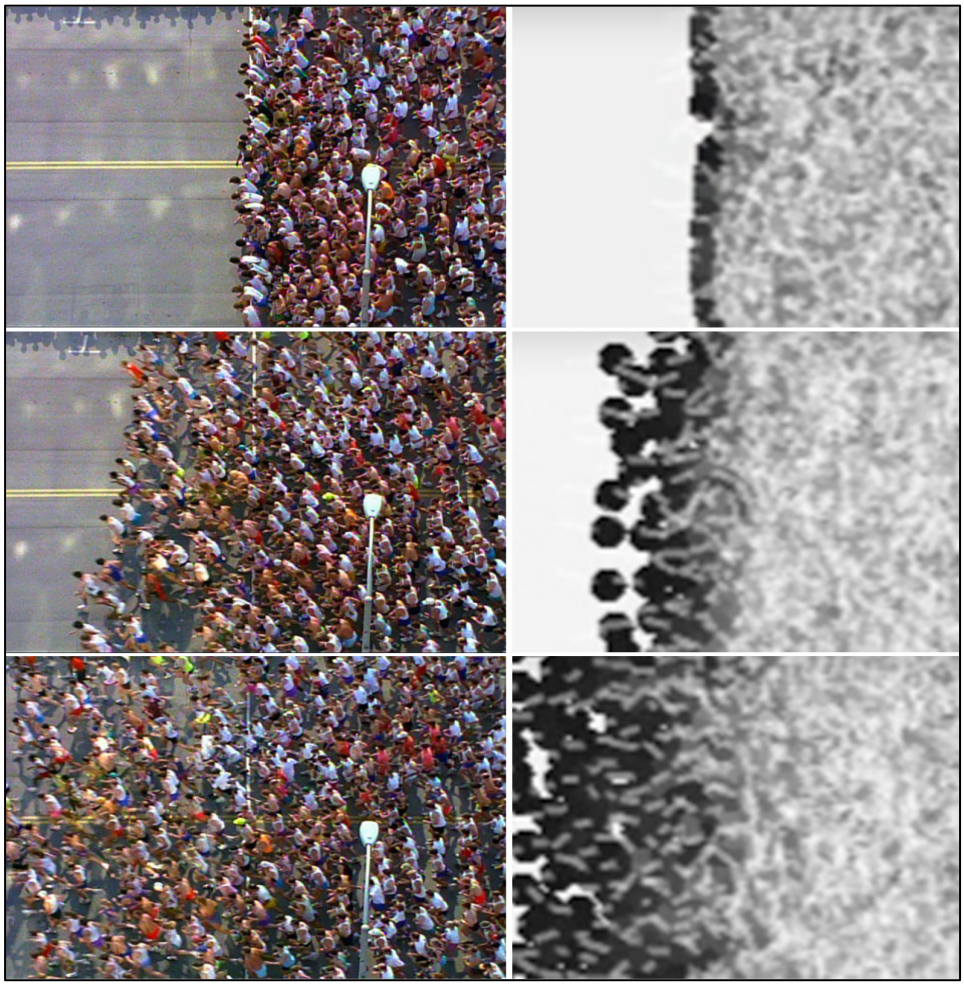


Figure 4.8 The gradual release of density-dependent velocity. Left: snapshots of marathon start; Right: our simulated result.

density to deal with, so they have the greatest freedom to accelerate to higher speeds compared to 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 4.8, 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 stuck to less competitive speeds, until the wave catches up to them eventually and the speeds equalize. Unlike the compressed room in Figure 4.7, the marathon scenario has a global path vector along the marathon track applied evenly to all contestants; but the centroidal response force reproduces the observed emergent acceleration wave effect.

4.5 Observable Patterns in Stationary Crowds

Figure 4.9 and Figure 4.10 illustrate the phenomenon of personal space compression in largely stationary crowds. 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 (as cited in section 2.3) which would have a difficult time distinguishing between static entities at varying distances from a barrier or an area of collective interest.

As demonstrated in Figure 4.9, our area-based force is able to 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 (Figure 4.10). 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

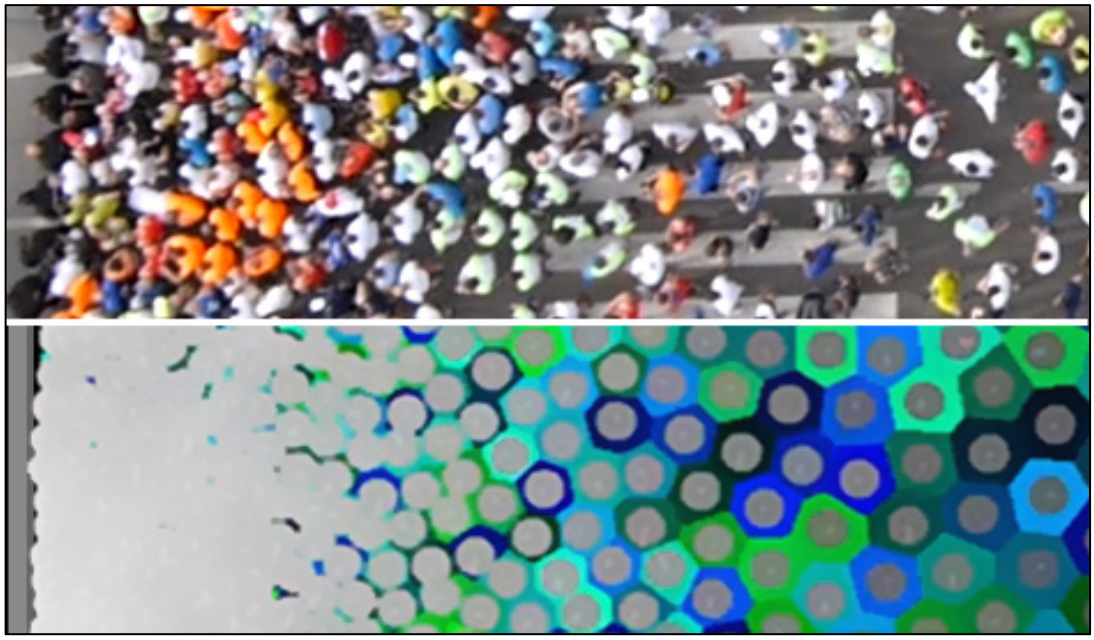


Figure 4.9 The gradual compression of personal space among a static crowd near an area of interest, a marathon start line (top); our simulated result (bottom).

of neighbours surrounding each entity. The PSM can be directly shown as an underlay to visualize the personal spaces used for centroidal force calculations. They also provide local density visualization, as discussed in section 3.7.4. Figure 4.10 illustrates a common emergent crowd phenomenon (arching around pathway bottlenecks), with noticeable compression of personal space near a narrow exit.

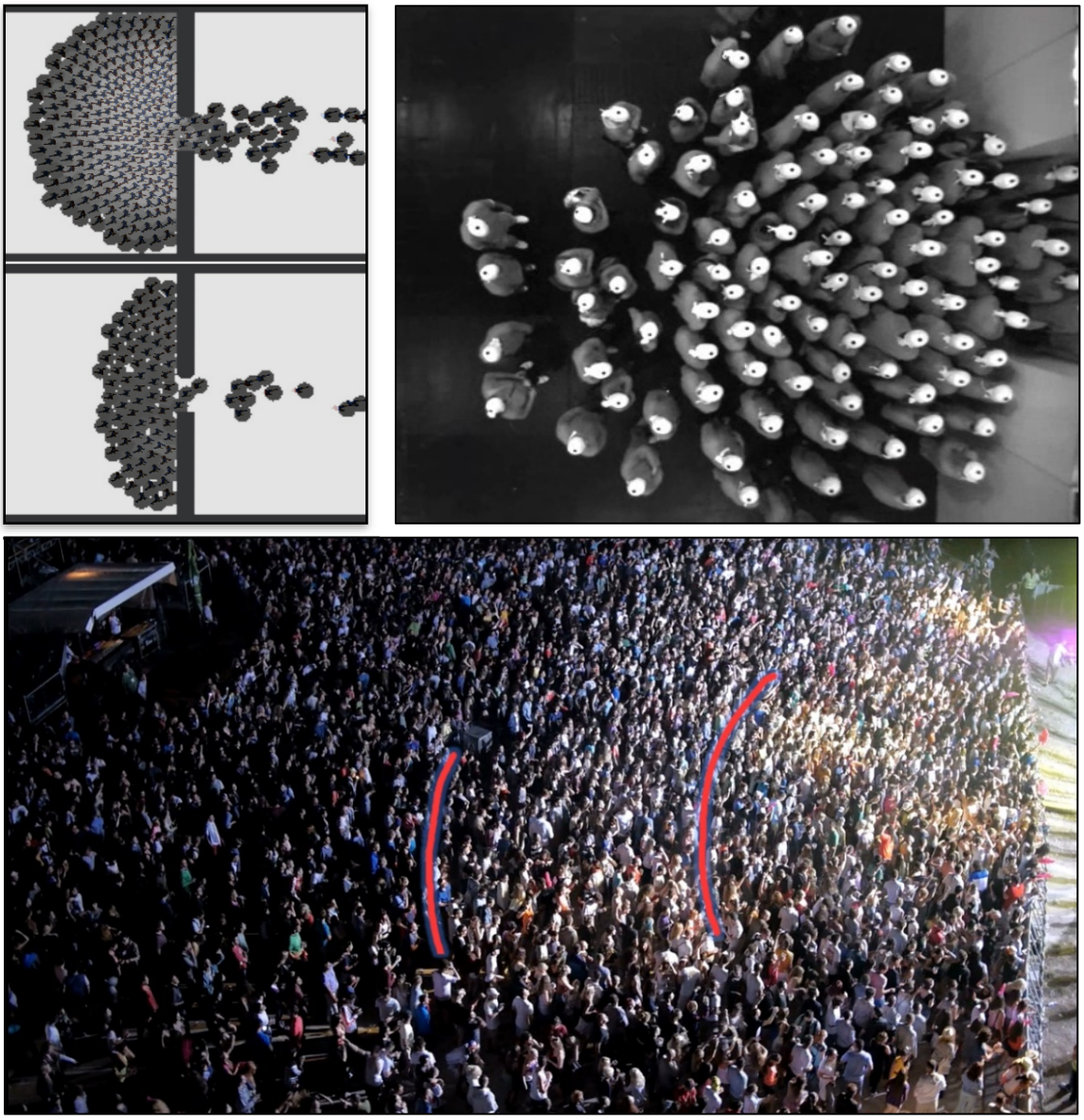


Figure 4.10 While our bottleneck arching pattern (top-left) could be improved, we're still able to display arching, gradual PS compression, and petal-like formations observed in crowds during egress [89] (top-right) and while stationary at a concert (bottom).

Simulation Method:		**Social Forces**	**Velocity-space Optimization (RVO/ORCA/ Implicit)**	**Statistical Inference / Probabilistic Model (WarpDriver)**	**Position-based Crowd Dynamics**	**Centroidal Particle Dynamics (CPD)**
		[18]	[24], [25], [55]	[28]	[54]	
Emergent Phenomenon	Relevant Figure(s)					
Unstable lanes in Bidirectional Flow	Figures 4.2-4.6					√
Stable lanes in Bidirectional Flow	Figures 4.2-4.6		Reproducible in sparse crowds; fails/congests at high densities	√	√	√
Gradual loss of personal space near gateways and bottlenecks	Figure 4.10					√
Arching near gateways and bottlenecks	Figure 4.10	√	√	√	√	√
Gradual loss of personal space near points of interest in stationary crowds	Figure 4.9					√
Arching near points of interest in stationary crowds	Figure 4.10	√				√

Table 4.1 A summary of the reproducibility of emergent trajectories and phenomena observed in crowds of *high-density* (as defined in section 2.3); comparing our CPD method to other microscopic crowd simulation approaches in the literature.

4.6 Simulation Performance

One of the objectives of CPD's design is to be computationally parallel-friendly allowing for interactive and high-performance frame-rates for thousands of entities in the scene. Our benchmarking consisted of three representative consumer-grade devices with various CPU-GPU configurations:

- Mid-range Desktop: 4GHz Quadcore CPU + Nvidia GTX970 GPU
- Laptop: Intel Core i5 (with integrated HD Graphics 4000)
- Mobile: Nexus 6P with Qualcomm Adreno 430 GPU

Table 4.2 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 4.11). Recall from Section 3.7 that each simulation cycle involves two major phases: PSM construction (i.e. cone rendering), and per-entity forces computation (centroidal, global, net, etc.). Our first engine, used in publications [20] and [53], was built on Processing 3.0 (a Java-based graphics framework) and supported two computation modes:

- CPU&CPU: CPU OpenGL call per cone render; CPU-computed forces
- GPU&CPU: GPU instancing to concurrently render all cones; CPU-computed forces

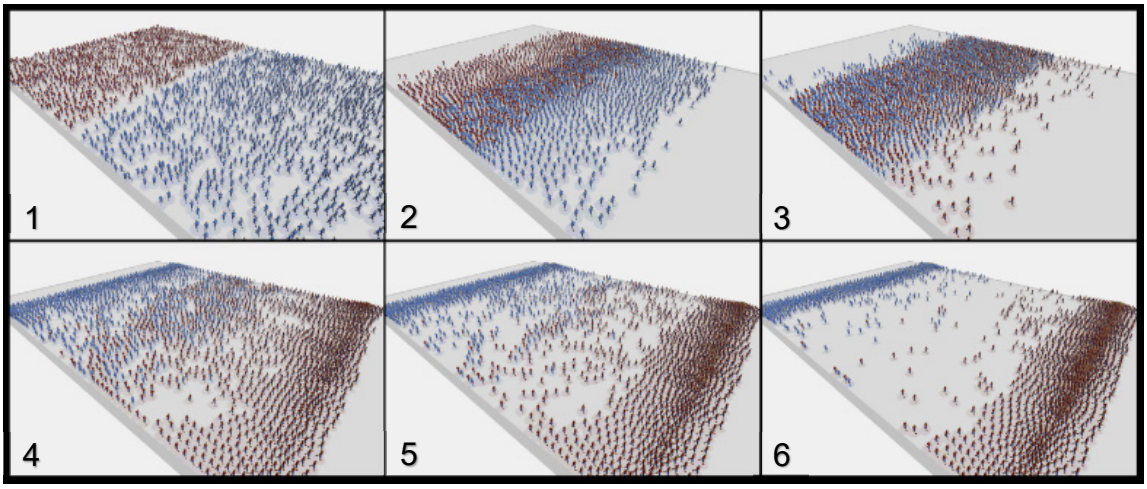


Figure 4.11 Artificial scenario used for benchmarking. Entities are color-coded by motion direction.

	CPU&CPU Non-Instanced Rendering			GPU&CPU Instanced Rendering GPU Shaders	
#Entities	Desktop GTX1060	Laptop Core i5	Nexus 6P (Android)	Desktop GTX1060	Speedup (compared to CPU&CPU)
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	12	3	47	2.8x
10,000	10	7	-	25	2.3x
15,000	7	4	-	20	2.6x
20,000	6	3	-	14	2.3x

Table 4.2 Simulation performance in *frames per second* (fps).

While the Android device was capable of simulating higher crowd counts, it was no longer able to do so at interactive framerates (i.e. 8fps or higher). The performance gain from implementing the GPU shaders (discussed in section 3.7.5) for instanced cone rendering is noticeable, at ~2.6x throughout. Given the 100ms 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 framerates for even higher counts.

Later iterations of the engine were developed using C++ (specifically, C++17 [77]) and OpenGL 4.3 [78], removing the dependency on Processing and on a Java Virtual Machine (JVM) at runtime. For this engine, the PSM construction (i.e. cone rendering) phase is always GPU-instanced; but 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 through the time step delta. Figure 4.12 and Figure 4.13 chart CPD's computational performance.

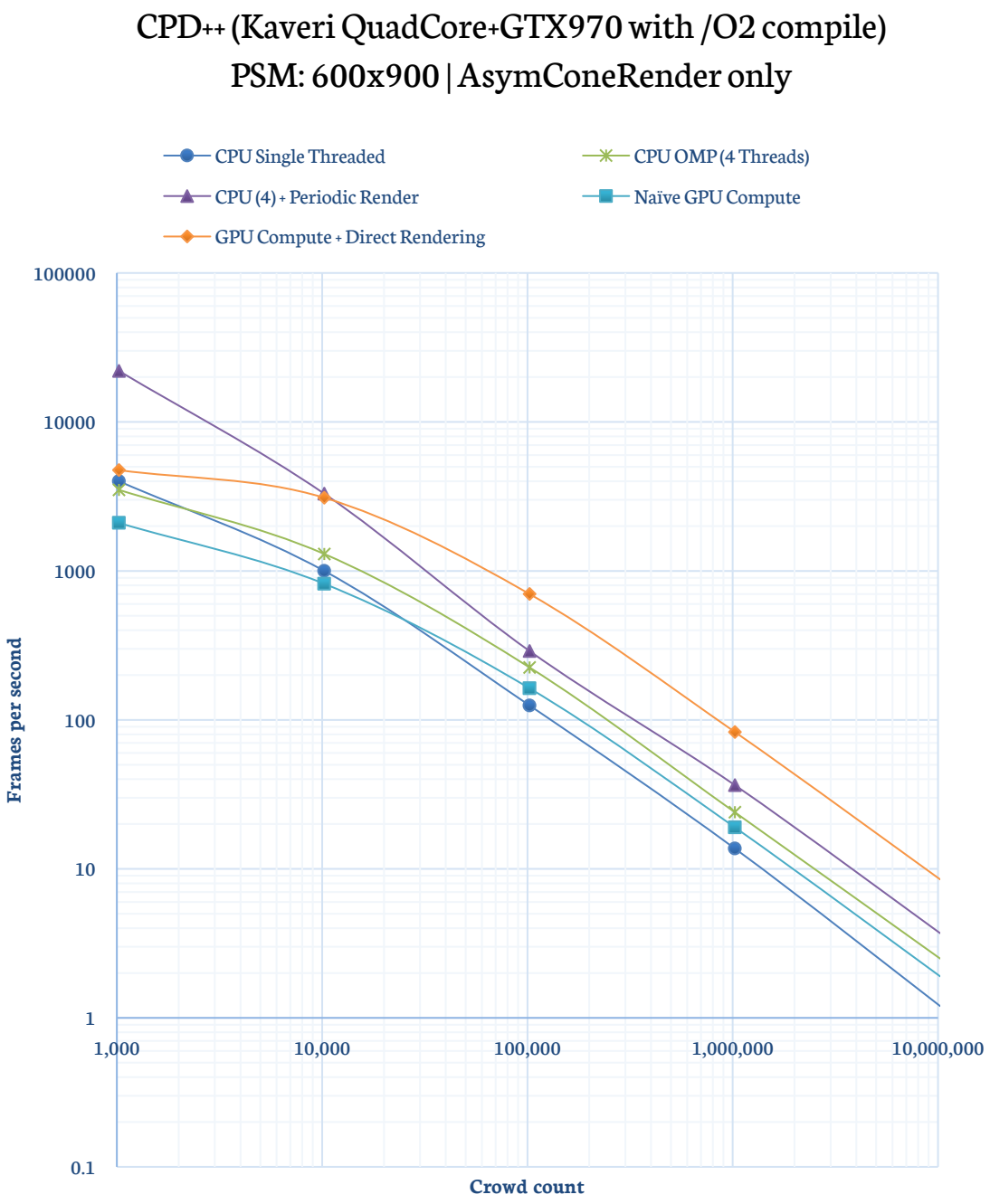


Figure 4.12 Log-log chart of PSM (i.e. PS cone rendering) computational performance in frames-per-second, plotted against the scene's crowd count.

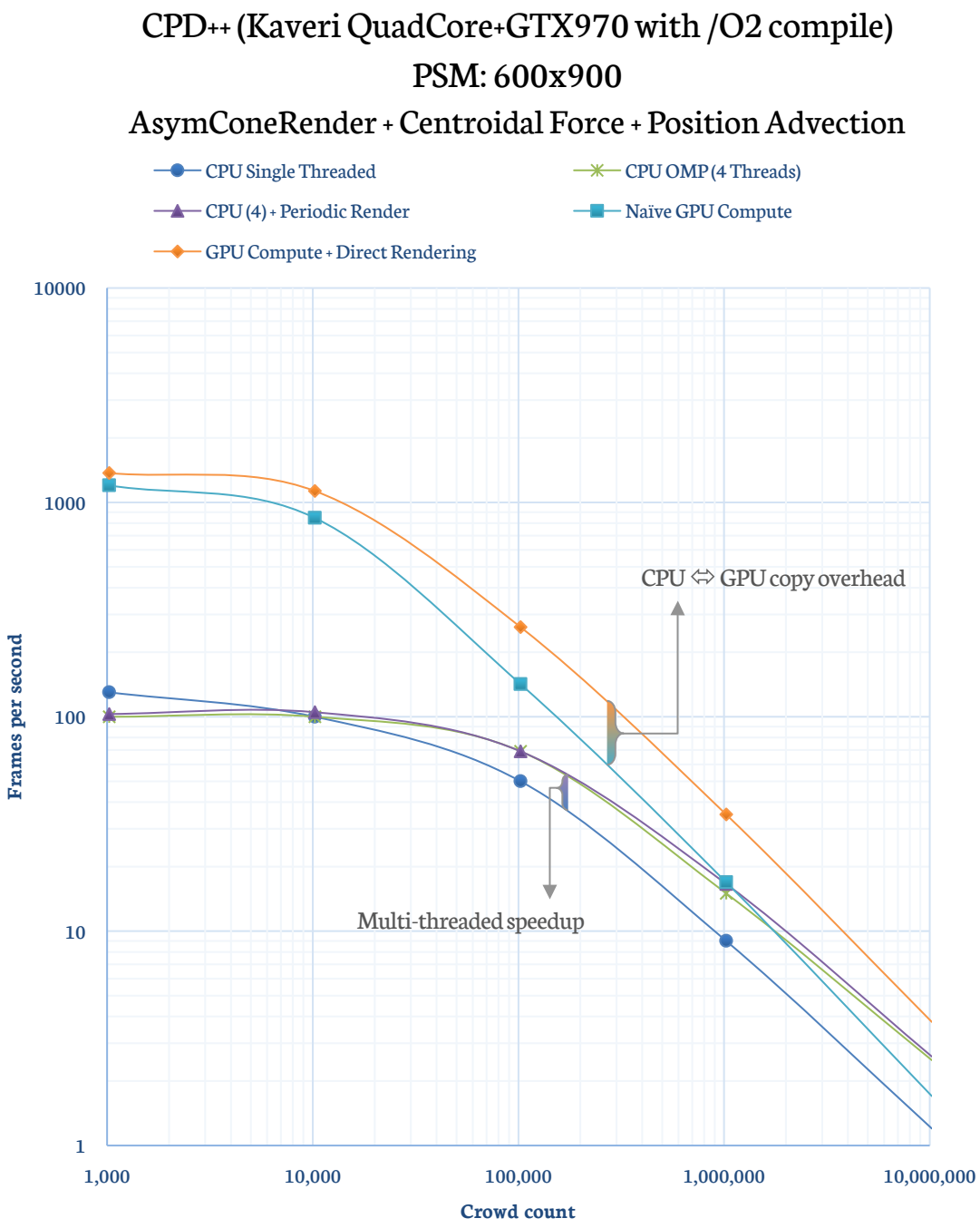


Figure 4.13 Log-log chart of full CPD cycle in frames-per-second, plotted against the scene's crowd count.

The C++ engine implements the algorithm in Figure 3.7 through several computation options or modes. They are broken down as follows:

- CPU Single Threaded – 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 for current frame's visualization; and made ready for next frame's PSM computation.
- CPU OMP (4 Threads): a multi-threaded implementation utilizing the OpenMP 2.0 standard. Once the PSM is fetched from the GPU into RAM; OpenMP splits the loop workload over the available CPU-threads. In this case we tested a quad-core CPU, and hence utilized 4 threads. Lastly, the new positions are sent back to the GPU as done in the single threaded mode.
- CPU (4) + Periodic Render: this mode recognizes that rendering at higher frequencies than supported by the display screen is essentially wasteful. So, this mode implements the same OpenMP multi-threaded computation, but only performs the visualization step at courser timesteps than the force computations. E.g. if you're able to compute forces @2000 fps, then the visualization on a typical 60Hz monitor will only be performed every ~16ms.
- 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 compute 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. Thus 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 Figure 4.13 is that the overhead of launching multiple threads in the CPU OMP Compute mode is too expensive compared to the single threaded CPU-compute option, for relatively

small crowd counts (less than 10,000 entities), but that the speedup due to CPU multi-threading does improve with higher workloads (crowd count). Periodic rendering improves the multi-threaded CPU mode's performance in the PSM computation phase (Figure 4.12) but not in the overall CPD computation (Figure 4.13). This typically indicates that the PSM computation is not the bottleneck in the overall CPD algorithm (a pleasant by-product of formulating neighbourhood 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 CPD's overall computations. With zero-copy direct rendering enabled, the GPU is able to provide approximately 2x speedup compared to the Naïve GPU option, as annotated in Figure 4.13. 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 the GPU's 4GB memory (VRAM) limitations become a limiting factor where the crowd computation has to be dispatched to the GPU in multiples of 4GB, compared to the CPU's native access to 32GB 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 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, as described in section 3.7.3. The artificial workloads presented in Figure 4.13 of up to 10 million entities were sufficiently simulated on a machine with 32GB RAM and 4GB VRAM (GPU memory).

The resolution of the PSM in pixels can be adjusted to reach higher performance targets, at the cost of reduced PS fidelity. Ideally, the simulation performance would depend solely on the crowd count rather than PSM resolution. We believe 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 [71], the direct successor of OpenGL.

For reference, ORCA was capable of simulating 5000 agents at 140fps, while WarpDriver simulated 5000 agents in real-time (15-20fps @ 50ms time step delta)[50][28], and Continuum Crowds ran 10,000 agents at 50fps [4].

4.7 Non-Homogenous Behaviour

Up to this point, we've been assuming that the agents are individual, cooperative and behaviourally homogenous; they also shared little to no relationship between each other. We've illustrated how phenomenon like lane formation, which appear to be organized, are in fact just emergent global behaviours due to each entity pursuing entirely individualistic pedestrian dynamics. In this section, we observe the results of simulated scenarios that involve intentional organization (e.g. pedestrians travelling in groups such as a tour group, or a family staying close together at a crowded festival). In those cases, the entities require additional social forces to be accounted for when computing each entity's net force (section 3.6). Having said that, we assume that overall, the personal space (centroidal) forces and their Amygdala-driven dynamics don't change just because people have organized into groups; every entity still wants to maintain a reasonable personal space within its surrounding.

The difficulty in simulating such scenarios is the lack of real data or trackable footage that focuses on micro-grouping configurations, especially in controlled experiment environments, to help us calibrate and further validate the simulated results. Such large-scale crowd data capture projects with varied obstacle scenarios are in demand [79].

The rest of this chapter documents CPD's ability to simulate scenarios that exhibit micro-grouping, competitive pathing, distracted entities, and uncooperative behaviour, as published in [80].

4.7.1 Disruptive Micro-grouping

In this scenario, a crowd of 990 entities is in bidirectional flow through a 40m corridor. Ten special entities are then grouped by implementing Boids flocking rules [17] of separation, cohesion, and alignment. These forces are fed to the special entities' net force calculation (section 3.6).

The special group underwent two simulation tasks:

a) Starting from the sideline, the group is asked to move *across* the bidirectional stream to reach the opposite end together (as shown in Figure 4.14).

b) Starting from the north, the group is asked to move *along* the bidirectional stream to reach the south.

The average time it took the group to complete task a) is 83.75 seconds. Task b) took 68 seconds. This confirms the intuitive notion that going across the established flow of a dense crowd will be slower, as the group has to either wait for openings to cross or force their way to disrupt the bidirectional flow. While opposing flow enjoyed lane formation as an emergent optimization strategy, micro-groups flocking across the corridor did not display any particular flow-optimizing behaviour, further explaining the delay in performing task (a).

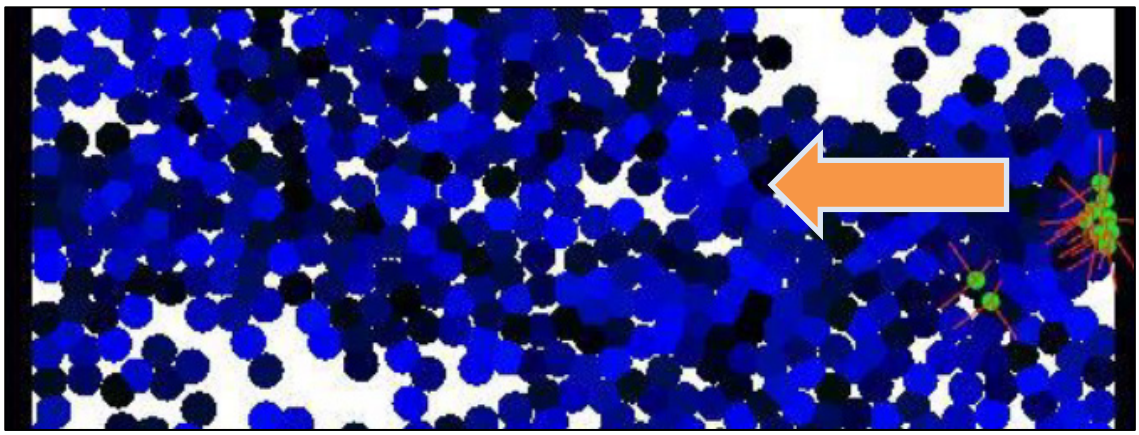


Figure 4.14 The ten grouped entities (green-coded) are explicitly grouped to stay together using the Boids flocking rules as they traverse across the scene (task (a)). The rest of the crowd is in north-south bidirectional motion.

The disparity between tasks (a) and (b) did not noticeably change for larger micro-groups of more than ten individuals, however, as the groups got smaller, nearing individualistic behaviour, the disparity between the two tasks was significantly reduced, and almost imperceptible in groups of two. The immediately observable explanation is that smaller groups can seize on smaller "openings" available to cross amidst the dense crowd. Additionally, task (b) is limited by the emergent bidirectional flow rate, which at high enough densities effectively equalizes movement speed for large portions of the crowd. In other words, even through task (b) seems easier, overtaking people ahead of you in a very dense crowd is quite difficult; hence the reduction in disparity between tasks (a) and (b) as the group size decreases.

4.7.2 Competitive Pathing

In this subsection, we demonstrate an experiment that illustrates the potential use of CPD in an architectural/urban design context to address anticipated dense-crowd issues.

The scenario is an artificial setup, where entities are initially arranged equally around a ring. Each entity's target is to arrive at the opposite side of the ring. There are no other global paths and no organized grouping. This artificial setup is designed to test an algorithm's ability to handle the least optimal configuration: all pedestrians are headed into each other, and all are competing for the center of the ring to reach the other side in the shortest path possible. Such scenarios are not too far off from reality. Indeed, major crossing such as Shibuya Crossing in Japan can display such a massively competitive pedestrian scenario.

Figure 4.15 shows the results of the crowd motion at various time instances. Recall that the default parameter values were: $\alpha = 0.9;\ \beta = 0.25$; and $\gamma = 0.1$. We created 3 variations of the crowd, as shown in Figure 4.15:

a) High aggression: $\alpha = 0.7;\ \beta = 0.3;\ \gamma = 0.4$. Here, the entities display higher-than-default drive towards the final destination (β) and lesser regard for the personal space violations (α). Additionally, the entities are highly resistant (γ) to paths that deviate from the optimal route (i.e. straight through the center of the ring). Hence, we see heavy congestion and a pattern where the red entities pierce through the blue entities to get to the other side. All entities share the same parameters; the colors only there to help visualize the overall effect.

- Avg. density experienced by all entities: 3.3 ft²/ pedestrian
- Peak density: 1.75 ft²/ pedestrian

b) Low aggression: $\alpha = 0.8;\ \beta = 0.2;\ \gamma = 0.1$. Here, the entities display higher regard for personal space violations than the aggressive entities in (a). They're also more receptive to deviating from the optimal path.

- Avg. density experienced by all entities: 3.4 ft²/ pedestrian
- Peak density: 1.92 ft²/ pedestrian

c) A round obstacle is inserted at the center of the ring, with the entities maintaining their low aggression parameters.

- Avg. density experienced by all entities: 3.3 ft²/ pedestrian
- Peak density: 2.5 ft²/ pedestrian

As if gently guided by this new obstacle, a cyclone pattern quickly forms and facilitates the crowd's collaborative turning motion. It might be counter-intuitive to think that an obstacle would ease traffic, but this is an example where architectural design can experiment with ways to help guide flow without explicitly designating single-way lanes.

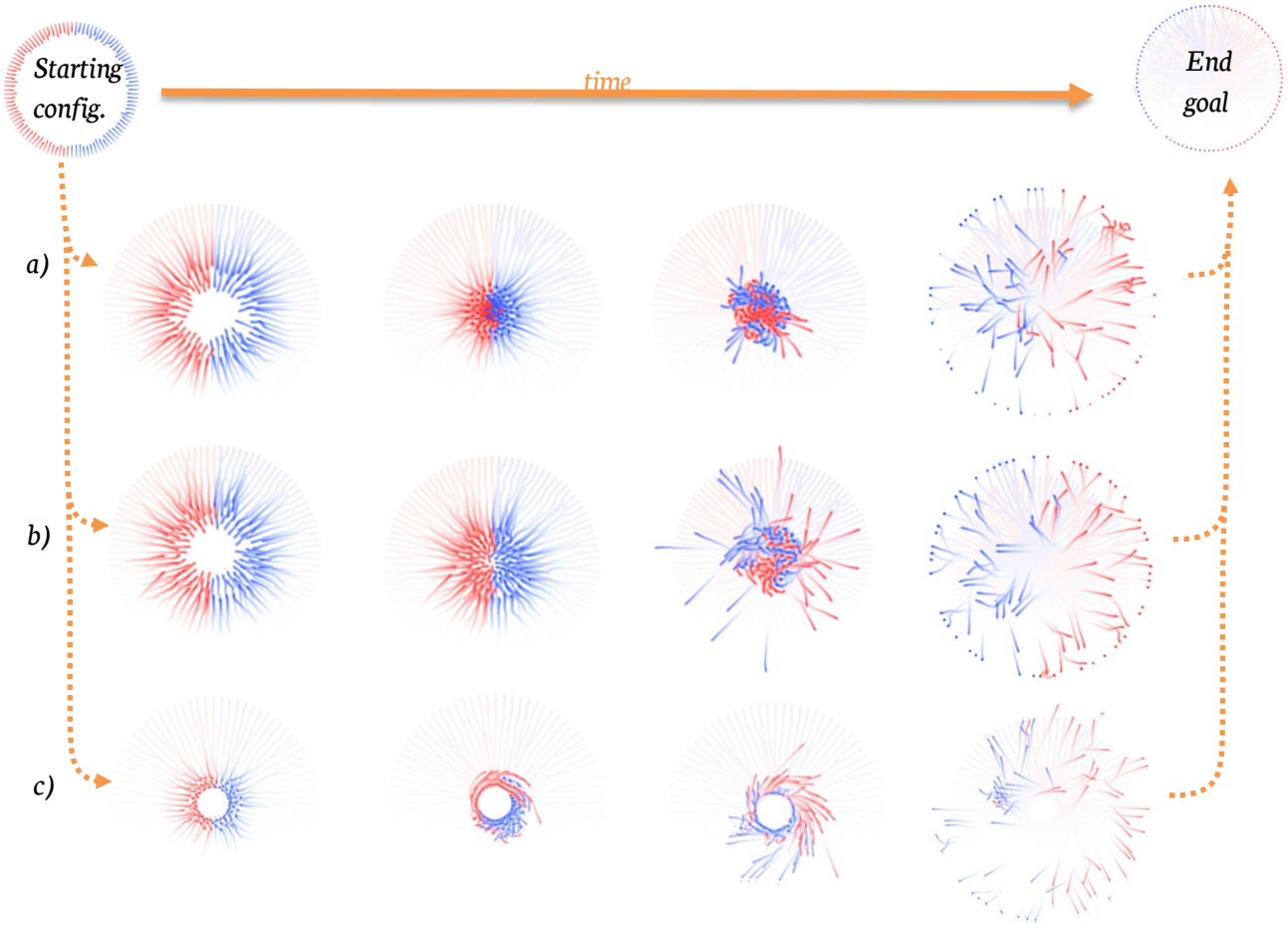


Figure 4.15 Concentric crowd motion under different parameter values: a) aggressive crowd; b) low aggression crowd; and c) round architectural artifact at the center of the ring with a low aggression crowd.

In-lab experiments of real pedestrians confirm the possibility of crowd motion shaping solely through passive obstacle design. For instance, it turns out that forcing queue lines to form near a congested gate will emergently reduce the peak experienced density [81]; also, counter-intuitively, a column placed near (but not blocking) a gate will improve its flow rate and reduce experienced density [82]). The simulated CPD scenario in Figure 4.15's could represent a high-traffic high-density zone in a busy mall, and the obstacle could be a seating area or some architectural feature that passively guides the flow of pedestrians to achieve favourable density outcomes. Given an objective function that encompasses the

desired experienced densities, optimization (or machine learning) techniques could automatically optimize the obstacle's polygonal shape. Such an approach does not restrict the creative input of the architect/designer, but it is rather viewed as a computer-assisted design exploration tool, allowing them to select from multiple local minima (i.e. select from various shape solutions of high-quality objective function scores) and further sculpt their designs from the nearest match to their desired aesthetic. This subsection has demonstrated how CPD can compliment and integrate with the recent trend of Generative Design (i.e. design-by-optimization) in the field of architecture & urban design [83].

4.7.3 Uncooperative Behaviour – Passage Blocking

The ideal pedestrian would pay attention all the time to their surroundings. By pre-emptively and carefully retaining their personal space, they should be able to avoid most collisions and disruptions to their intended motion. We see such efficiencies in busy crossings, such as Shibuya, Japan, where hundreds of pedestrians with competing trajectories are able to cross smoothly. However, we also simulated behaviour that would be deemed uncooperative to the collective pedestrian motion, potentially disrupting it.

Unlike clearly visible and static obstacles in the scene (e.g. wall, vegetation, park benches, etc.), stationary subgroups in the crowd can be more difficult detect in a dense scenario until very close to that group. Figure 4.16 shows a bidirectional scenario in a 3.5m hallway. We assign a few pedestrians to stand still, effectively blocking passage. As expected, the pedestrian flowrate across the hallway is decreased. We observed a reduction by 40% on average and reaching 0% (i.e. complete blockage) at worst for 36% of the entities in the hallway.

Clogging the path to a complete but temporary stop is an emergent effect we criticized other methods for in the discussion about unconstrained Bidirectional flow in section 4.3. The key difference in this scenario, is that the clogging behaviour matches observed behaviour in real narrow hallways [82]

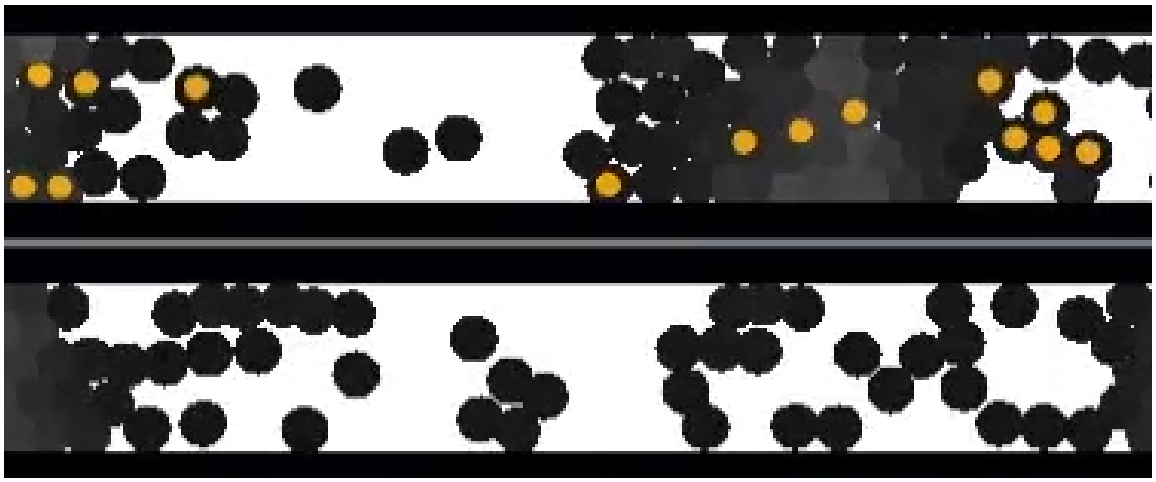


Figure 4.16 In narrow hallways, a few pedestrians standing still (e.g. chatting, etc.) could cause significant congestion. Left: entities standing still (yellow); Right: same simulation time instance with no blockage.

involving uncooperative agents. It appears that our 2D compressible PS kernel permits smooth passage for many entities, but over time, the density pileup overwhelms the compressibility threshold of the PS kernels (a threshold that varies stochastically across entities) and thus creating a congestion that slows the flow rate significantly until completely stopping for the previously mentioned 36%. An intuitive way to visualize this phenomenon is to imagine that eventually, the congestions leaves a single lane that's one entity-wide, and now both directions of flow need to alternate in sharing that single path. Thus, many entities will wait until that lane frees up, or the overall density dissipates to the point that compressibility will allow for other lanes again.

When a much wider 40m corridor was simulated, as shown in Figure 4.17 (think a busy path on a campus, or access to cafeteria area), an interesting observation arose. As expected, the area surrounding the uncooperative bunch (bright green) experienced congestion. However, later in the simulation, we observed multiple pockets of congestion forming away from the initial bundle. These secondary masses of congestion result from the diverted traffic concentrating on the new limited space. The insight here is that even if the width of passage is much wider than the uncooperating group, a high pedestrian flow

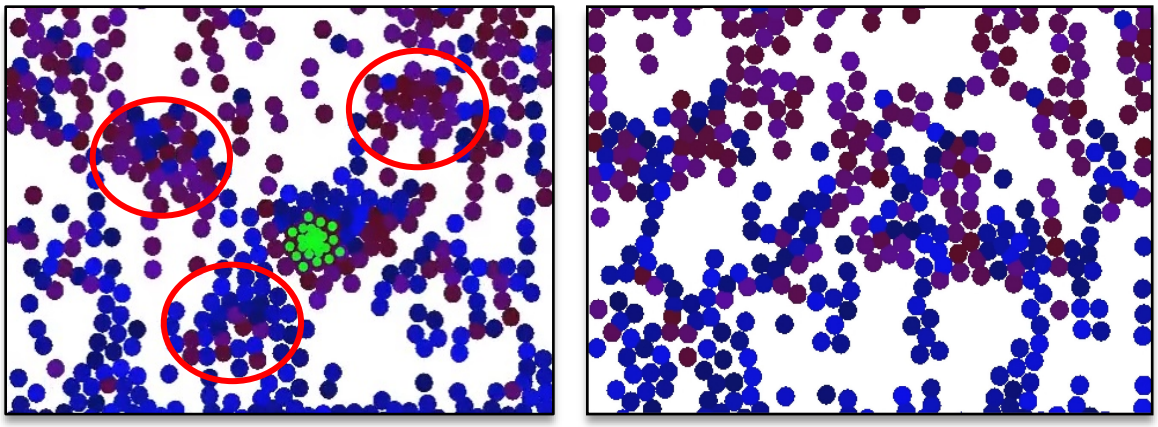


Figure 4.17 A north-south bound (red-blue) bidirectional flow. Left: entities disrupting the flow by standing still (shown in green) lead to pockets of congestion across the entire corridor within a few of minutes. Right: an unimpeded corridor captured at the same simulation time as the left scenario; here, lane formation across the full width (~40m) resulted in a more spread-out distribution of density, while having already let more people pass through at that point in simulation time.

rate will cause pockets of congestion to inevitably form across the width of the corridor. This observation agrees with intuition that a vehicular accident in one lane can cause traffic to slow down and potentially congest across the width of the road, due to the decreased traffic bandwidth (i.e. available lanes).

4.7.4 Uncooperative Behaviour – Distracted Pedestrians

Distracted pedestrians can cause injury to themselves and others. Large events, as depicted in Figure 4.17, are tricky to navigate as it is and the possibility for slight collision (shoulder rubbing) is reasonable to expect. So, we wanted to simulate how distracted pedestrians might make navigating such events even harder. One of the causes of such distracted behaviour is pedestrians texting/browsing/gaming on their phone [84], [85]. It is a serious enough issue that the experiments from the literature recommend that cities with high pedestrian traffic should consider and implement a ban on cell-phone usage at street crossings [86].

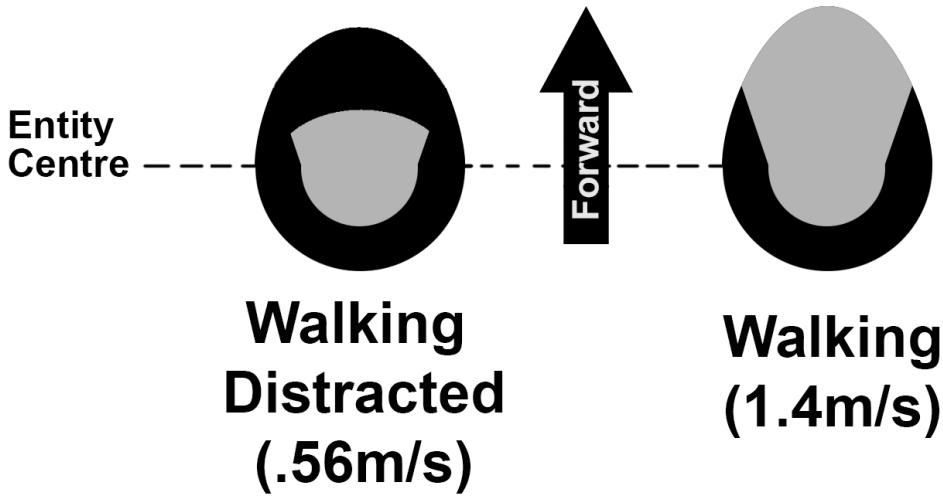


Figure 4.18 The personal space (PS) weight map for a pedestrian distracted on their phone (left) is culled from the front due to lack of visibility, in contrast to a normally walking CPD kernel from section 3.4.1.

The asymmetric PS kernels (from section 3.4.1) have been modified to represent the personal spaces of distracted pedestrians as follows:

- Distraction period: 5 seconds every 15 seconds (~third of their time distracted on their phone).
- Speed slows down to 40% [84]; and the PS weight map is culled to match the reduced visibility ahead of the distracted entity, as shown in Figure 4.18.

Figure 4.19 illustrates the scene setup while Figure 4.20 charts a sample of collision counts recorded. In the absence of any distracted pedestrians, only a handful of instances of high collision likelihood have been observed. The count increases exponentially as the ratio of distracted entities increases within the dense crowd. These collision counts were also inversely proportional to corridor width; not due to increased bidirectional flow density, but rather due to the lack of additional space for undistracted pedestrians to perform their avoidance maneuvers. Collisions counts were much less pronounced in unidirectional flow, where the biggest effect was instead the slowdown of surrounding entity motion.

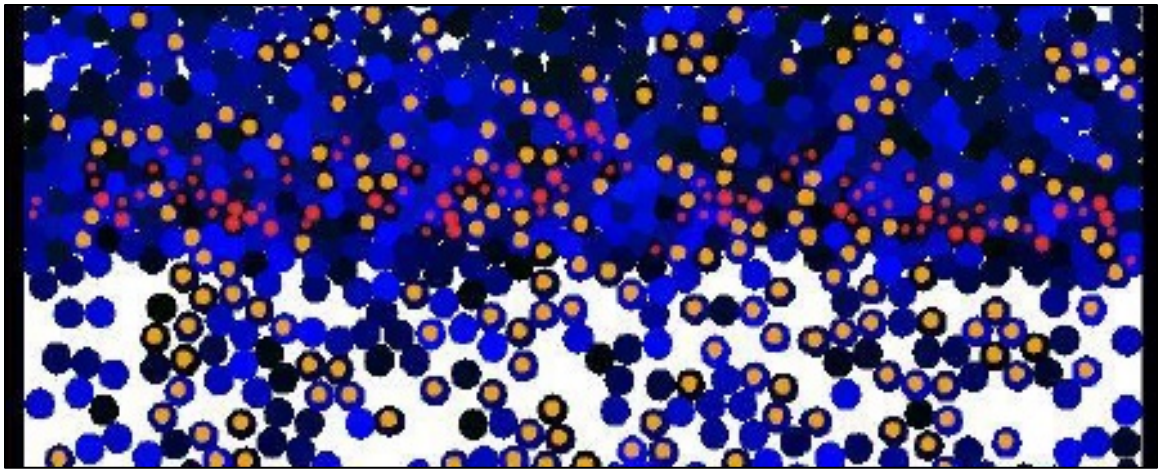


Figure 4.19 An example of 30% distracted pedestrians in north-south bidirectional flow. Red indicates detected instance of high likelihood of collision. Orange indicates all distracted pedestrians.

That can be explained by observing that relative velocities between the entities are on-average less than the relative velocities in bidirectional flow, which gives fully-aware entities a larger amount of time to react and manoeuvre around the distracted crowd when needed.

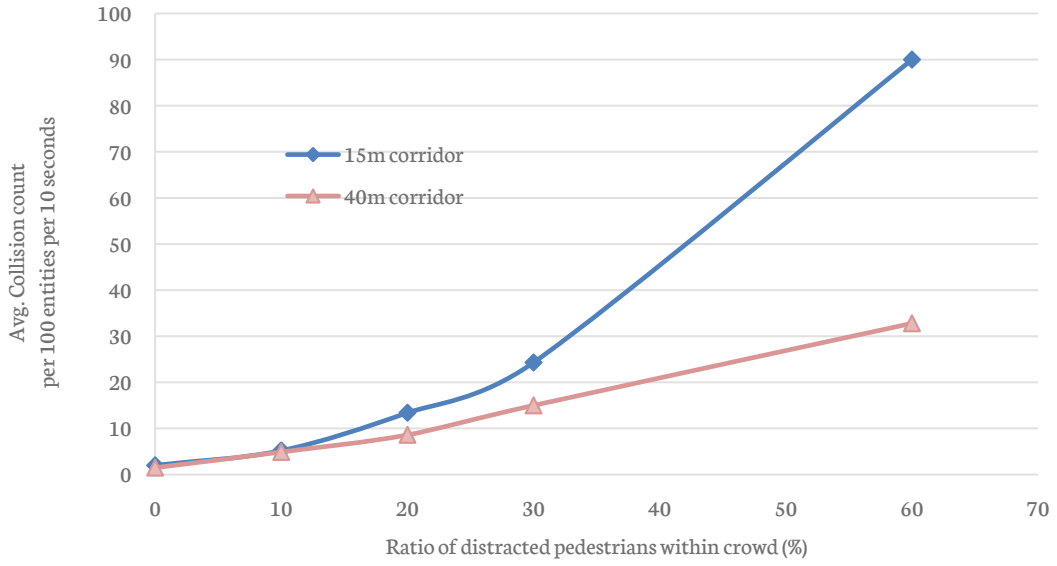


Figure 4.20 Collision counts recorded from scenario in Figure 4.19.

Chapter 5

Conclusions and Future Work

Here, we outline a few limitations and opportunities for future research emanating from the proposed method in its current state.

5.1 Validation

The application target of our dissertation and its proposed CPD method so far has been the film, gaming, and serious education industries. But safety-critical applications such as civil planning, crowd control, and large-scale event threat assessment stand to benefit the most from dense-crowd simulation research. Although our method uses empirically-driven parameters to produce visually convincing emergent behaviour (as presented in Chapter 4), we could not yet endorse its use for safety-critical applications. We echo our earlier assertion that crowd path modeling is essentially an exercise in abstraction with no precise microscopic "ground truth" to converge on. However, there are aggregate statistical measures that academic literature and commercial tools have relied on to help validate their simulated crowds, and thus increase confidence in a crowd simulation model to the point where it is acceptable for such critical applications. In this section, we outline the metrics that, when rigorously measured, could further improve the confidence in CPD's simulated results.

- Fundamental diagram – given a static sampling area, the fundamental diagram [87], [88] is a flow-density graph that plots the crowd's flow rate through the area against the area's density. This macroscopic measure has been extensively studied in real crowds and in-lab scenarios [27], [59], [73], [89], [90]; and a general plot profile for human crowd flow has been documented. In essence, it's an inverse relationship, where the flow rate or overall speed reduces as density increases. This

is the most common data-validation approach in crowd simulation literature; and it is the relationship profile that many agent-based crowd simulation methods (Lagrangian or Eulerian-based) calibrate their agent-based parameters to try and emergently best fit. But it is, again, an aggregate macroscopic measure, so the curve plot abstracts away all the microscopic details of the entities' motion paths (e.g. lane formation, lane switching, arching, petalling, etc.).

- Rate of egress (evacuation) – there are studies that document the duration and rate at which occupants, under various psychological (i.e. panic) conditions, would evacuate a building or a stadium [8], [91], [92]. Given precise floor plans, it would possible to replicate those real-life experiments in virtual crowds and obtain statistically similar rates and durations of egress. This is a macroscopic measure that is primarily concerned with testing evacuation protocols in buildings and stadiums; and can help architects and safety engineers make decisions on gate closures, barriers, and other flow-shaping measures [82] to achieve optimal evacuation routes that let the occupants evacuate in the shortest time possible while maintaining a safe overall density to avoid crowd crushes or stampedes. The rate of egress does not measure localized density statistics and hence is not a suitable metric for validating microscopic phenomenon. Our experiments on competitive pathing (presented in section 4.7.2) and the resulting aggregated local measures could complement the rate of egress metric; so that, given real data, those experiments could become part of the data-validation effort of the CPD model, and not just a design exploration tool.
- Governing distributions – recent studies [23] have demonstrated that humans move in anticipation of future events, particularly potential collisions events. When crowd motion is analysed in the velocity-space, a statistically strong power-law relationship between an entity's current velocity and its anticipated-time-to-next-collision emerges [56].
- Local Statistical Similarity Measure – a relatively recent development in crowd data-validation literature has been the pursuit of statistical similarity metrics [93] that compare the local density

patterns and distributions surrounding each entity. These are inspired by the Structural Similarity Index (SSIM), a widely adopted metric to assess the quality and error of pixel intensities in image compression algorithms when compared to the source uncompressed image [94]. Unlike the previously discussed crowd metrics, this one is localised and does not require a static sampling area. The sampling can (and should) move with the entity allowing for a more agent-oriented microscopic metric of crowd densities and observed trajectories. This similarity metric has been seeing increased acceptance (academically, at least) for use as the primary objective function in automated model parameter calibration, and model output validation [23], [95].

According to the established wisdom in crowd M&S literature (extensively surveyed in [61]), the aforementioned metrics, when pursued and measured properly, would improve the confidence in using our proposed CPD algorithm for short-range collision-avoidance. However, having said that, the very concept of data-validation in the context of crowd simulation is not without its criticism.

The first critique comes from fire safety literature [79], which brings attention to the absence of an international standard for the verification and validation of evacuation models, and that the definition of "validity" is itself still ambiguous and can carry different meanings and differing levels of acceptable rigour to different experts in different fields. 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 being used to validate ship or stadium evacuation models).

Another critique comes from the field of neurocomputing [96] which presented crowd-trajectory data captured in lab settings; and it argues that collision avoidance methods in general should take empirically captured macroscopic statistical "truths" into account when deriving their microscopic model. That is, instead of calibrating some 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, and 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 knowledge about the surrounding aggregate dynamics than the real pedestrian entity it supposedly models.

State-of-the-art methods that share our target application of real-time crowd dynamics for film, gaming and education (methods such as WarpDriver [28], ORCA[50], RVO [24], [25], [50], Social Forces [18], and Position-based [54] crowds) typically use one or more of the aforementioned macroscopic metrics to indicate the validity of their simulated model. However, as shown in Chapter 4, those methods also struggle to reproduce microscopic effects, particularly in dense Bidirectional flow scenarios, due to the emergent lane rigidity or artificial congestion.

One of the causes for their artificial congestion was the rigid 1D separation distance between entities. In contrast, CPD models a compressible 2D area. 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, where we focused our attention on addressing the close-range avoidance dynamics first, in a localised 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).

Another possibility is to use CPD's low-cost density estimator (section 3.7.4) as a congestion heatmap that feeds *back* to global path planning models that optimize for shortest path *and* congestion avoidance (e.g. [97], [98]). This is one way to extend CPD's PS model utility beyond short-range avoidance.

5.2 Fully Decentralized & Distributed CPD Implementation

It is fair to wonder if the GPU-accelerated global PSM computation (i.e. centralized cone-rendering implementation presented in section 3.7) betrays the spirit of locally-encapsulated agent-based simulation. Perhaps so, but it is a well-intentioned computational betrayal/optimization based on the reasonable assumption that the *primary* users of our algorithm would be simulation modelers and practitioners running simulation scenarios on workstation desktop computing hardware (or equivalent server tech for cloud-computing; or powerful consoles for gaming). Thus, it made sense to utilize the available parallel computing resources (namely, the GPU's rendering pipeline) even though it led to the synchronized computing of the PSM for all agents at once in a centralized -yet highly efficient- manner. But out of completeness, let us briefly document that the PSM (as described in section 3.3) can in fact be constructed in a completely *decentralized* agent-oriented manner. Such an implementation would also become more applicable and practical to use in distributed collision-avoidance environments (e.g. self-driving cars, robots, or drones) where centralized computing and message passing costs could become a bottleneck in what is an otherwise decentralized and distributed autonomous system.

In the centralized PSM computation, the GPU renders a global PSM map that all entities can query. To decentralize this, we still utilize a Truncated Voronoi tessellation, but now every entity computes only the Voronoi cell it belongs to, and not the whole PSM diagram. This way, every entity ends up with a polygonal cell representing their unviolated PS kernel, and the centroidal force computation continues in the same data-parallel fashion as before (section 3.7). Figure 5.1 sketches the cell construction process.

This is a zero-communication implementation that doesn't require message passing between the agents, does not query a centralized cone renderer, but instead relies purely on the sensory abilities

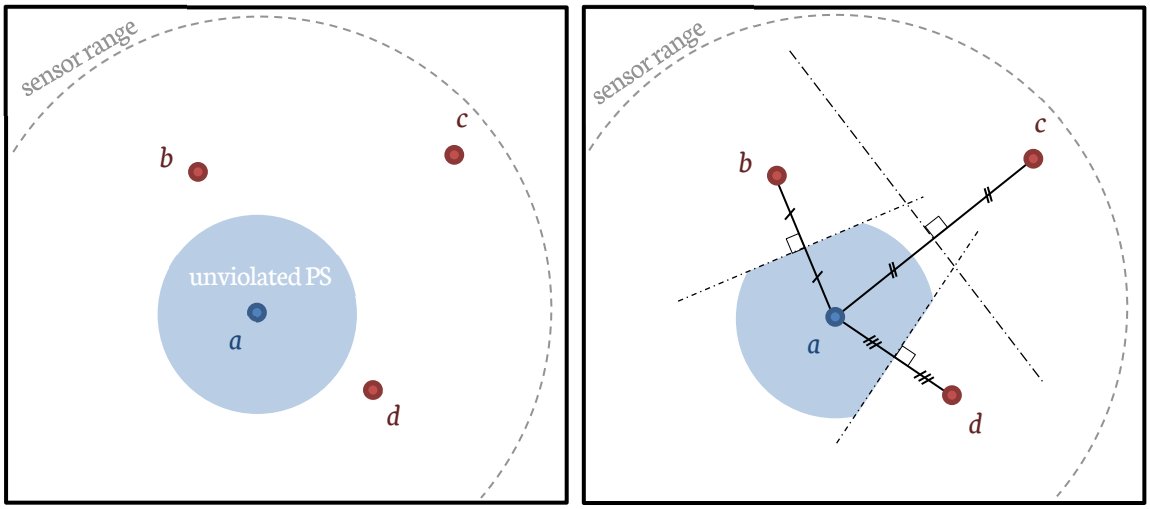


Figure 5.1 Given an entity *a* surrounded by neighbours *b*, *c*, and *d* within *a*'s proximity sensor range, the unviolated PS cell can be updated using the perpendicular bisector of the vector to each neighbour. Notice that even though *c* is within *a*'s detection range, it is not close enough to affect the PS cell update.

available to each agent. If the agents lack perfect detection and proximity sensing of their surroundings, then it will be reflected in the fidelity of the PS kernel they can locally generate and update.

5.3 Heterogeneous Crowd and Multi-layered PSM

The centroidal force computed a locally preferred bearing and direction of motion for the entity to restore its personal space. However, acting on that centroidal "preference" can be 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 non-human and heterogeneous entities sharing the same simulation space, we can still compute the centroidal forces as we did with humans, but the mechanics of following that centroidal "preference" to compute the relevant forces might differ for each kind of entity (e.g. strollers, shopping carts, bikes, vehicles, etc.).

Those other kinds of entities will consume the same rules about personal space update but execute those maneuvers under their own physical constraints (e.g. a bike or a car will 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 possibly spine segment. In this case, the jump flooding technique [99] might be a suitable alternative to Voronoi cones.

Another feasible improvement on CPD involves separating the gaze vector from body orientation for situations involving, for example, looking both ways before crossing the road. The gaze kernel could be constructed in its own PSM layer, and its influence would become another parameter in the weighting of centroidal force calculation.

5.4 Additional Social Rules

By design, CPD deals primarily with short-range collision avoidance and does not attempt to simulate grouping or flocking behaviour as entities are assumed individualistic with their own target destination in mind. This was intentional, to study the effects of centroidal forces in isolation, and to serve as a sanity check before incorporating additional complexity into the model.

With confidence in CPD's ability to reproduce emergent phenomenon in dense crowds, we presented a few micro-grouping experiments presented in section 4.7.1. It is possible to further manipulate the global pathing vectors being input to CPD model, allowing it to be augmented by psychosocial behaviours such as the ones studied for evacuation scenarios (e.g. SAFEgress [91]), to account for each entity's familiarity with the environment, social attitudes, and herd dynamics (leader-follower, social order, etc.) and panic levels.

5.5 Discrete-event CPD Model and Simulation

To recap from section 2.2, discrete-event simulation triggers state updates in response to scheduled (internal) or received (external) events, with event-dependent time stepping. Additionally, modeling using a formalism such as PDEVS (Parallel Discrete Event System Specification) can aid in the model's

design and management of its complexity [5]. Such formalisms typically provide highly optimized universal simulation engines, allowing the modeler to focus on abstracting the behaviour being studied, without concern for engine implementation details. Pursuing such a formal model would present several advantages compared to home-brew event-driven solutions, namely: composability with the existing library of DEVS models; interfacing with modeling standards such as Building Information Models (BIM) [64], [100]; direct simulation on existing cloud computing infrastructures [101], [102]; and the ability to submit the model to formal verification, validation, and static analysis techniques (e.g. deadlock detection, unreachable states, etc.) [6], [103].

5.6 Machine Learning

We proposed an analytical model in this thesis that produced convincing visual realism for dense crowd trajectories. However, we'd like to highlight an opportunity for future research: applying AI (machine learning) techniques to automatically model a person's reaction to violations of personal space.

At the heart of our CPD method is the thesis that local avoidance dynamics can be modeled as an optimization problem over a 2D personal space area; which finds the best response vector to minimize personal space violations; and we presented force functions in section 3.4 to compute the optimization.

According to the universal approximation theorem [104], there exists an artificial neural network (ANN) with a finite number of nodes (neurons) that could be trained to approximate any continuous function. We wonder, then, could an ANN approximate the centroidal force functions, as shown in Figure 5.2? And could it represent the model parameters α, β, γ as dynamic neural edge weights in the ANN rather than being static polynomial coefficients?

Given a PSM computed from real crowd footage/trajectory, could an ANN (or the 2D-friendly convolutional neural network [105]) ingest the 2D personal space violations in the PSM and produce the

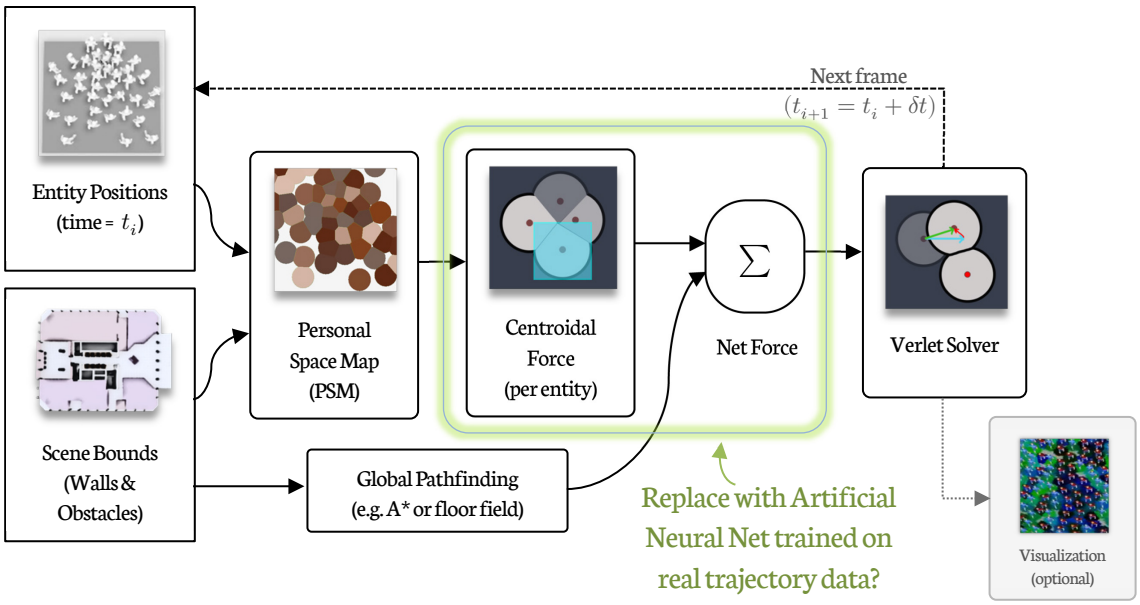


Figure 5.2 A machine learning algorithm could replace the currently fixed centroidal force and net force calculations; and potentially providing an automated data-driven calibration of the model's parameters.

optimal response force based on what it learned from the responses observed in real crowd trajectories? The ANN could, theoretically, automate the adjustment of the centroidal and net force parameters for cultural and contextual differences (e.g. theme park vs stadium) by training on footage/trajectory data sourced from those contexts. Such a data-driven model would adapt the force equations to achieve a more faithful representation of the crowd, without additional intervention from a modeling expert, and the AI model could additionally provide robustness in unforeseen scenarios.

5.7 Teachability and Student Work

The simulation engine(s) proposed in Chapter 3 and benchmarked in Chapter 4 was found to be easily understood by visual learners. With documentation and inline commenting designed to guide the student through their code exploration, we built a beginner-friendly framework to facilitate the teachability of the CPD model and to allow its expandability by future researchers. At Carleton University's department of Systems and Computer Engineering, graduate students from the Master's level engaged us in learning about crowd simulation research. Those students had no prior crowd

modeling experience (and for some, no prior simulation experience at all). Appendix B. shows part of a larger documentation package made available to graduate students and researchers, enabling them to participate in crowd simulation, experiment design, and behavioural modeling. We have supervised and co-authored research findings with these students in [80], where they assisted with experiment setup and behavioural modeling for micro-grouped, distracted, and uncooperative pedestrians. CPD's 2D area-based approach facilitates reasoning about behavioural modifications in a visual way (e.g. via computational geometry).

5.8 Conclusion

We presented centroidal particle dynamics (CPD), an agent-based short-range collision avoidance model for pedestrians in dense crowds. We've shown our model's ability to reproduce emergent dense crowd phenomenon that show high congruence to real pedestrian trajectory data; especially in cases where state-of-the-art struggles (Table 4.1); and have explained our performant implementations suitable for simulating high density crowds in film, gaming, and educational applications.

To further the trust in CPD for use in more critical and safety-oriented application like event and evacuation planning, the model's parameters will require further calibration, and we've presented how future research could incorporate data-validation methods to enhance trust in the CPD model, and the potential for automating its data-driven calibration via AI.

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 favourable 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 the PS as a rigid 1D separation distance leading to artificial congestion and unnecessary clogging of pathways.

References

[1] D. C. Duives, W. Daamen, and S. P. Hoogendoorn, "State-of-the-art crowd motion simulation models," *Transp. Res. Part C Emerg. Technol.*, vol. 37, no. 0, pp. 193–209, 2014.

[2] S. A. Turris, A. Lund, and R. R. Bowles, "An analysis of mass casualty incidents in the setting of mass gatherings and special events," *Disaster Med. Public Health Prep.*, vol. 8, no. 2, pp. 143–149, 2014.

[3] J. Fruin, "The Causes and Prevention of Crowd Disasters," *First Int. Conf. Eng. Crowd Saf.*, vol. 1, pp. 1–10, 1993.

[4] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM SIGGRAPH 2006 Pap. - SIGGRAPH '06*, vol. 25, no. 3, p. 1160, 2006.

[5] A. C. H. A. C. H. Chow and B. P. B. P. Zeigler, "Parallel DEVS: a parallel, hierarchical, modular, modeling formalism," in *Proceedings of Winter Simulation Conference*, 1994, pp. 716–722.

[6] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. Academic Press, 2007.

[7] S. Huerre, J. Lee, M. C. Lin, and C. O'Sullivan, "Simulating believable crowd and group behaviors," *Acm Siggraph Asia 2010 ...*. ACM, p. 92, 2010.

[8] I. A. S. Z. Preschl, "Evacuation Capacity of Door Openings in Panic Situations," *Bouw.*, vol. 26. pp. 62–67, 1971.

[9] R. A. Smith, "Volume flow rates of densely packed crowds," *Eng. crowd Saf.*, pp. 313–319, 1993.

[10] R. L. Hughes, "The Flow of Human Crowds," *Annu. Rev. Fluid Mech.*, vol. 35, no. 1, pp. 169–182, 2003.

[11] T. M. Kisko, R. L. Francis, and C. R. Nobel, "Evacnet4 user's guide," *Univ. Florida*, 1998.

[12] S. Ashraf Tashrifullahi and M. A. Hassanain, "A simulation model for emergency evacuation time of a library facility using EVACNET4," *Struct. Surv.*, vol. 31, no. 2, pp. 75–92, 2013.

[13] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma, and N. Nosovic, "Dijkstra's shortest path algorithm serial and parallel execution performance analysis," in *MIPRO, 2012 Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2012, pp. 1811–1815.

[14] J. W. Chinneck, *Practical optimization: A gentle introduction*. 2004.

[15] D. Thalmann, S. Musse, and A. Braun, *Crowd simulation*. 2007.

[16] S. Bandini, S. Manzoni, and G. Vizzari, "Crowd Modeling and Simulation," in *Recent Advances in Design and Decision Support Systems in Architecture and Urban Planning*, Springer, Dordrecht, 2004, pp. 161–175.

[17] A. Morin, J. B. Caussin, C. Eloy, and D. Bartolo, "Collective motion with anticipation: Flocking, spinning, and swarming," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 91, no. 1. pp. 763–782, 2015.

[18] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, May 2000.

[19] N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling individual agents in high-density crowd simulation," *Proceedings of the 2007 ACM SIGGRAPHEurographics symposium on Computer animation*, vol. 1. Eurographics Association, p. 108, 2007.

[20] D. C. Brogan, R. A. Metoyer, and J. K. Hodgins, "Dynamically simulated characters in virtual environments," *IEEE Comput. Graph. Appl.*, vol. 18, no. 5, pp. 58–69, 1998.

[21] O. De Gyves, L. Toledo, and I. Rudomín, "Proximity Queries for Crowd Simulation Using Truncated Voronoi Diagrams," *Proceedings of Motion on Games - MIG '13*. ACM, pp. 87–92, 2013.

[22] O. Hesham and G. Wainer, "Centroidal Particles for Interactive Crowd Simulation," in *Proceedings of the Summer Computer Simulation Conference*, 2016, pp. 7:1-7:8.

[23] J. Pettré, J. Ondřej, A.-H. Olivier, A. Cretual, and S. Donikian, "Experiment-based modeling, simulation and validation of interactions between virtual walkers," *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA '09*. ACM, p. 189, 2009.

[24] J. Den Van Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," *Proceedings - IEEE International Conference on Robotics and Automation*. pp. 1928–1935, 2008.

[25] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Springer Tracts in Advanced Robotics*, vol. 70, no. STAR, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Springer, Berlin, Heidelberg, 2011, pp. 3–19.

[26] A. Best, S. Narang, S. Curtis, and D. Manocha, "DenseSense: Interactive Crowd Simulation using Density-Dependent Filters," in *Vladlen Koltun and Eftychios Sifakis*, 2014.

[27] S. Narang, A. Best, S. Curtis, and D. Manocha, "Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors," *PLoS One*, vol. 10, no. 4, 2015.

[28] D. Wolinski, M. C. Lin, and J. Pettré, "WarpDriver: context-aware probabilistic motion prediction for crowd simulation," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–11, 2016.

[29] M. Moussaid, D. Helbing, and G. Theraulaz, "How simple rules determine pedestrian behavior and crowd disasters," *Proc. Natl. Acad. Sci.*, vol. 108, no. 17, pp. 6884–6888, 2011.

[30] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," *ACM Trans. Graph.*, vol. 29, no. 4, p. 1, 2010.

[31] S. Yiquan *et al.*, "A grid-based spatial data model for the simulation and analysis of individual behaviours in micro-spatial environments," *Simul. Model. Pract. Theory*, vol. 38, no. 0, pp. 58–68, 2013.

[32] A. Al-Habashna and G. Wainer, "Modeling pedestrian behavior with Cell-DEVS: Theory and applications," *Simulation*, vol. 92, no. 2, pp. 117–139, 2016.

[33] S. Sarmady, F. Haron, and A. Z. Talib, "A cellular automata model for circular movements of pedestrians during Tawaf," *Simul. Model. Pract. Theory*, vol. 19, no. 3, pp. 969–985, 2011.

[34] M. Van Schyndel, O. Hesham, G. Wainer, B. Malleck, and B. Kennedy, "Crowd Modeling in the Sun Life Building," in *Proceedings of SimAUD 2016*, 2016.

[35] G. W. Rhys Goldstein, "Simulation of Deformable Biological Structures with a Tethered Particle System Model," in *The Canadian Medical and Biological Engineering Society (CMBEC)*, 2009.

[36] Y. Tang, K. S. Perumalla, R. M. Fujimoto, H. Karimabadi, J. Driscoll, and Y. Omelchenko, "Optimistic Parallel Discrete Event Simulations of Physical Systems Using Reverse Computation," *Work. Princ. Adv. Distrib. Simul.*, pp. 26–35, 2005.

[37] J. Model and M. C. Herbordt, "Discrete event simulation of molecular dynamics with configurable logic," *Proc. - 2007 Int. Conf. F. Program. Log. Appl. FPL*, pp. 151–158, 2007.

[38] N. Santoro, *Design and Analysis of Distributed Algorithms*. 2006.

[39] K. M. Chandy, J. Misra, and L. M. Haas, "Distributed deadlock detection," *ACM Trans. Comput. Syst.*, vol. 1, no. 2, pp. 144–156, May 1983.

[40] J. Matocha and T. Camp, "A taxonomy of distributed termination detection algorithms," *J. Syst. Softw.*, vol. 43, no. 3, pp. 207–221, 1998.

[41] J. Nutaro, *Building Software for Simulation Theory & Algorithms*. 2011.

[42] G. A. Wainer, *Discrete Event Simulation and Modeling: Theory and Applications - Model-Based Design*. 2009.

[43] L. Yilmaz, S. J. E. Taylor, R. Fujimoto, and F. Darema, "Panel: The future of research in modeling & simulation.," *Proceedings of the Winter Simulation Conference 2014*. IEEE Press, p. 2797, 2014.

[44] S. Bandini, M. Mondini, and G. Vizzari, "Modelling negative interactions among pedestrians in high density situations," *Transp. Res. Part C Emerg. Technol.*, vol. 40, pp. 251–270, 2014.

[45] J. A. Manual, "Special Events Contingency Planning," United States. Federal Emergency Management Agency, Apr. 2010.

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

[47] J. Gillian, "Inaugural crowd sizes ranked | PolitiFact," *Politifact*, 20-Jan-2017.

[48] A. Seyfried, B. Steffen, W. Klingsch, T. Lippert, and M. Boltes, "Steps Toward the Fundamental Diagram – Empirical Results and Modelling," in *Pedestrian and Evacuation Dynamics 2005 SE - 36*, Springer, Berlin, Heidelberg, 2007, pp. 377–390.

[49] J. Zhang and A. Seyfried, "Empirical characteristics of different types of pedestrian streams," *Procedia Eng.*, vol. 62, pp. 655–662, 2013.

[50] J. Snape, S. J. Guy, D. Vembar, A. Lake, and M. C. Lin, "Reciprocal Collision Avoidance and Navigation for Video Games," in *Game Developers Conference*, 2012, vol. 27599, pp. 1–14.

[51] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," in *ACM SIGGRAPH Asia 2009*, 2009, p. 1.

[52] S. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha, "Pledestrians: a least-effort approach to crowd simulation," *Proc. 2010 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.*, p. 119, 2010.

[53] B. J. Mohler, W. B. Thompson, S. H. Creem-Regehr, H. L. Pick, and W. H. Warren, "Visual flow influences gait transition speed and preferred walking speed," *Exp. Brain Res.*, vol. 181, no. 2, pp. 221–228, 2007.

[54] T. Weiss, C. Jiang, A. Litteneker, and D. Terzopoulos, "Position-based multi-agent dynamics for real-time crowd simulation," in *Proceedings of the Tenth International Conference on Motion in Games - MIG '17*, 2017, pp. 1–8.

[55] S. J. Guy, N. Sohre, R. Narain, and S. J. Guy, "Implicit Crowds: Optimization Integrator for Robust Crowd Simulation," *ACM Trans. Graph. Artic.*, vol. 36, no. 136, 2017.

[56] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Phys. Rev. Lett.*, vol. 113, no. 23, p. 238701, 2014.

[57] E. T. Hall, "Distances in Man," *hidden Dimens.*, pp. 113–129, 1990.

[58] D. P. Kennedy, J. Gläscher, J. M. Tyszka, and R. Adolphs, "Personal Space Regulation by the Human Amygdala," *Nat. Neurosci.*, vol. 12, no. 10, pp. 1226–1227, 2010.

[59] U. Chattaraj, A. Seyfried, and P. Chakroborty, "Comparison of Pedestrian Fundamental Diagram Across Cultures," *Adv. Complex Syst.*, vol. 12, no. 03, pp. 393–405, 2009.

[60] G. Van Den Bergen and D. Gregorius, *Game Physics Pearls*. 2010.

[61] X. Shi, Z. Ye, N. Shiwakoti, and Z. Li, "A Review of Experimental Studies on Complex Pedestrian Movement Behaviors," in *Cictp 2015*, ascelibrary.org, 2015, pp. 1081–1096.

[62] K. E. Hoff, J. Keyser, M. Lin, D. Manocha, and T. Culver, "Fast computation of generalized Voronoi diagrams using graphics hardware," *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*. ACM Press/Addison-Wesley Publishing Co., pp. 277–286, 1999.

[63] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[64] C. Eastman *et al.*, "Simulating Human Behavior in not-yet Built Environments by means of Event-based Narratives," *Automation in Construction*, vol. 38, no. 1. Society for Computer Simulation International, pp. 109–127, 28-Apr-2015.

[65] W. Zeng and R. L. Church, "Finding shortest paths on real road networks: The case for A*," *Int. J. Geogr. Inf. Sci.*, vol. 23, no. 4, pp. 531–543, Apr. 2009.

[66] I. Rudomin, B. Hernández, O. De Gyves, L. Toledo, I. Rivalcoba, and S. Ruiz, "GPU generation of large varied animated crowds," *Comput. y Sist.*, vol. 17, no. 3, pp. 365–380, 2013.

[67] J. Pettré, H. Grillon, and D. Thalmann, "Crowds of moving objects: Navigation planning and simulation," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2007, pp. 3062–3067.

[68] Y. Xiao, Z. Gao, Y. Qu, and X. Li, "A pedestrian flow model considering the impact of local density: Voronoi diagram based heuristics approach," *Transp. Res. Part C Emerg. Technol.*, vol. 68, 2016.

[69] Y. Qu, Y. Xiao, J. Wu, T. Tang, and Z. Gao, "Modeling detour behavior of pedestrian dynamics under different conditions," *Phys. A Stat. Mech. its Appl.*, vol. 492, pp. 1153–1167, 2018.

[70] O. Hesham and G. Wainer, "Context-sensitive Personal Space for Dense Crowd Simulation," in *Proceedings of the Symposium for Architecture and Urban Design*, 2017, pp. 173–180.

[71] J. A. Shiraef, "An Exploratory Study of High Performance Graphics Application Programming Interfaces," University of Tennessee at Chattanooga, 2016.

[72] F. I. Pelupessy, W. E. Schaap, and R. van de Weygaert, "Density estimators in particle hydrodynamics," *Astron. Astrophys.*, vol. 403, no. 2, pp. 389–398, May 2003.

[73] J. Zhang and A. Seyfried, "Empirical characteristics of different types of pedestrian streams," *Procedia Eng.*, vol. 62, pp. 655–662, 2013.

[74] M. Saberi, K. Aghabayk, and A. Sobhani, "Spatial fluctuations of pedestrian velocities in bidirectional streams: Exploring the effects of self-organization," *Phys. A Stat. Mech. its Appl.*, vol. 434, pp. 120–128, 2015.

[75] S. Chandra and A. K. Bharti, "Speed Distribution Curves for Pedestrians During Walking and Crossing," *Procedia - Soc. Behav. Sci.*, vol. 104, pp. 660–667, 2013.

[76] O. Hesham and G. Wainer, "Centroidal Particles for Interactive Crowd Simulation," in *Proceedings of the Summer Computer Simulation Conference*, 2016, vol. 48, no. 9, pp. 7:1-7:8.

[77] ISO/IEC JTC1/SC22/WG21, "ISO International Standard ISO/IEC 14882:2011(E) – Programming Language C++," 2011.

[78] J. Kessenich, D. Baldwin, R. Rost, and LunarG, "The OpenGL Shading Language Version 4.30," 2013.

[79] E. Ronchi, E. D. Kuligowski, D. Nilsson, R. D. Peacock, and P. A. Reneke, "Assessing the Verification and Validation of Building Fire Evacuation Models," *Fire Technol.*, vol. 52, no. 1, pp. 197–219, 2016.

[80] O. Hesham, W. Aburime, Z. Rabeh, S. Bhushan, and G. Wainer, "Observed Behaviours in Simulated Close-range Pedestrian Dynamics," *Proc. Symp. Simul. Archit. Urban Des.*, 2018.

[81] A. Sieben, J. Schumann, and A. Seyfried, "Collective phenomena in crowds - where pedestrian dynamics need social psychology," *PLoS One*, vol. 12, no. 6, pp. 1–19, Feb. 2017.

[82] X. Shi, Z. Ye, N. Shiwakoti, D. Tang, and J. Lin, "Examining effect of architectural adjustment on pedestrian crowd flow at bottleneck," 2018.

[83] D. Nagy *et al.*, "Project Discover : An application of generative design for architectural space planning," *Symp. Simul. Archit. Urban Des.*, no. June, pp. 59–66, 2017.

[84] L. L. Thompson, F. P. Rivara, R. C. Ayyagari, and B. E. Ebel, "Impact of social and technological distraction on pedestrian crossing behaviour: An observational study," *Inj. Prev.*, vol. 19, no. 4, pp. 232–237, 2013.

[85] J. L. Nasar and D. Troyer, "Pedestrian injuries due to mobile phone use in public places," *Accid. Anal. Prev.*, vol. 57, pp. 91–95, Aug. 2013.

[86] D. C. Schwebel, D. Stavrinos, K. W. Byington, T. Davis, E. E. O'Neal, and D. de Jong, "Distraction and pedestrian safety: How talking on the phone, texting, and listening to music impact crossing the street," *Accid. Anal. Prev.*, vol. 45, no. 2, pp. 266–271, Mar. 2012.

[87] A. Seyfried, B. Steffen, W. Klingsch, T. Lippert, and M. Boltes, "Steps Toward the Fundamental Diagram – Empirical Results and Modelling," *Pedestr. Evacuation Dyn. 2005*, pp. 377–390.

[88] A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes, "The fundamental diagram of pedestrian movement revisited," *J. Stat. Mech. Theory Exp.*, vol. 2005, no. 10, pp. P10002–P10002, Oct. 2005.

[89] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram," *J. Stat. Mech. Theory Exp.*, vol. 2012, no. 2, p. P02002, 2012.

[90] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, "Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions," *J. Stat. Mech. Theory Exp.*, vol. 2011, no. 6, p. P06004, Apr. 2011.

[91] M. L. Chu, P. Parigi, J.-C. Latombe, and K. Law, "SAFEgress: A flexible platform to study the effect of human and social behaviors on egress performance," *Proceedings of the Symposium on Simulation for Architecture & Urban Design*. Society for Computer Simulation International, pp. 4:1-4:8, 2014.

[92] H. Kl, "Models for Crowd Movement and Egress."

[93] S. J. Guy, J. van den Berg, W. Liu, R. Lau, M. C. Lin, and D. Manocha, "A statistical similarity measure for aggregate crowd dynamics," *ACM Trans. Graph.*, vol. 31, no. 6, p. 1, 2012.

[94] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[95] D. Wolinski, S. J. Guy, A. H. Olivier, M. Lin, D. Manocha, and J. Pettré, "Parameter estimation and comparative evaluation of crowd simulations," in *Computer Graphics Forum*, 2014, vol. 33, no. 2, pp. 303–312.

[96] W. Lu, X. Wei, W. Xing, and W. Liu, "Trajectory-based motion pattern analysis of crowds," *Neurocomputing*, vol. 247, pp. 213–223, 2017.

[97] W. G. Van Toll, A. F. Cook IV, and R. Geraerts, "Real-time density-based crowd simulation," *Comput. Animat. Virtual Worlds*, vol. 23, no. 1, pp. 59–69, 2012.

[98] A. Barnett, H. P. H. Shum, and T. Komura, "Coordinated Crowd Simulation With Topological Scene Analysis," *Comput. Graph. Forum*, vol. 35, no. 6, pp. 120–132, 2016.

[99] G. Rong and T. Tan, "Jump flooding in GPU with applications to Voronoi diagram and distance transform," *in Studies in Logical Theory, American Philosophical Quarterly Monograph 2*. ACM, pp. 109–116, 12-Dec-2006.

[100] S. Wang, M. Van Schyndel, G. Wainer, V. S. Rajus, and R. Woodbury, "DEVS-based Building Information Modeling and simulation for emergency evacuation," *Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC)*. pp. 1–12, 2012.

[101] S. Wang and G. Wainer, "A simulation as a service methodology with application for crowd modeling, simulation and visualization," *Simulation*, vol. 91, no. 1, pp. 71–95, Apr. 2015.

[102] D. Zehe, A. Knoll, W. Cai, and H. Aydt, "SEMSim Cloud Service: Large-scale urban systems simulation in the cloud," *Simul. Model. Pract. Theory*, vol. 58, pp. 157–171, 2015.

[103] G. A. Wainer, *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press, 2009.

[104] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control. Signals, Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[105] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2014.

Appendices

Appendix A. Source Code Fragments

A.1 Host (CPU) Programming

Sample code for CPU color-coding and the computation of centroidal forces, as presented in 3.7.

colorID_hash.pde

```
// One-to-One hash to encode pedestrian ID into unique RGB24 colors (alpha ignored)
color ID_TO_COLOR(int id){
  return id>=0? 0xFF00000A+(id*COLOR_ID_STEP): color(0,0,id+10);
}

// One-to-One Hash to recover pedestrian ID from an RGB24 colors (alpha ignored)
int COLOR_TO_ID(color colorCode)
{
  return (colorCode<color(0,0,11)) ?
      //reserved obstacle color : regular entity color
        (int)blue(colorCode)-10 : (colorCode-0xFF00000A)/COLOR_ID_STEP;
}
```

computeCentroids.pde

```
Canvas calcCentroids(EntitySystem sys) {
    Entity[] entities = sys.getEntities();
    PGraphics ds = this.drawingSurface; //shorthand
    PGraphics cs = this.centroidSurface; //shorthand
    int w = ds.width, h = ds.height;
    float coneArea = PI*CONE_RADIUS*CONE_RADIUS; // full PS area
    int[] hits = new int[CROWDCOUNT]; // visible portion of the softicle

      // Either: a) Aggregate per pixel (more suitable for turning into a MapReduce)
      if(CROWDCOUNT*coneArea > ds.width*ds.height){
        for (int y=0; y<ds.height; y++) {
          for (int x=0; x<ds.width; x++) {
            int id = COLOR_TO_ID(ds.get(x, y));
            if (id < 0 || id >= CROWDCOUNT) continue;
            entities[id].getCentroidPosition().add(x, y, 0);
            hits[id]++;
          }
        }
      }

      // or b) Aggregate per entity (PS overlap, but faster in sparse sims)
```

```
        else {
          for (int i = CROWDCOUNT; i-->0; ) { //[TIGHT_LOOP]
            float s = CONE_RADIUS+1;
            float x = entities[i].getPosition().x, y=entities[i].getPosition().y;
            // top-left PS corner
            int x1 = (int)clamp(x-s, 0, w), y1 = (int)clamp(y-s, 0, h);
            // bot-right PS corner
            int x2 = (int)clamp(x+s, 0, w), y2 = (int)clamp(y+s, 0, h);
            for (int xx=x1; xx<=x2; xx++) { // [TIGHT_LOOP]
              for (int yy=y1; yy<=y2; yy++) {
                if (ds.get(xx, yy)==entities[i].getColorCode()) {  // [old] get int()
                  entities[i].getCentroidPosition().add(xx, yy, 0);
                  hits[i]++;
                }
              }
            }
          }
        }


        // Finally, after either (a) or (b), compute the centroids
        for (int i = CROWDCOUNT; i-->0; ){
          if (hits[i]==0) continue; // skip softicles that never appeared on screen
          entities[i].getCentroidPosition().div(hits[i]);
          float portion = hits[i]/(coneArea+10);
          entities[i].setPressure(1-portion);
        }
      }
    }
```

A.2 GPU Shaders (GLSL2.0)

Sample source code for the data-parallel geometry shader computing the PSM from a point cloud of pedestrian positions.

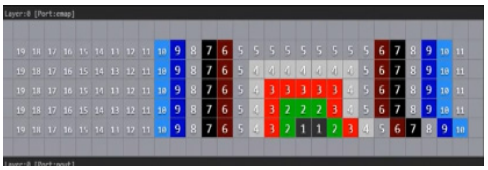
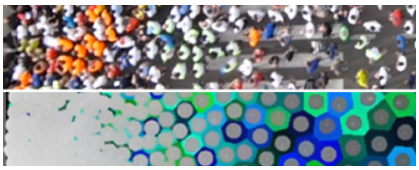
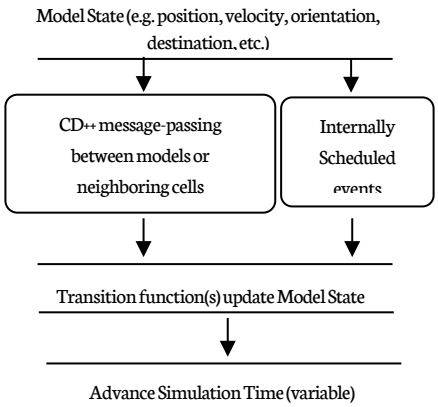
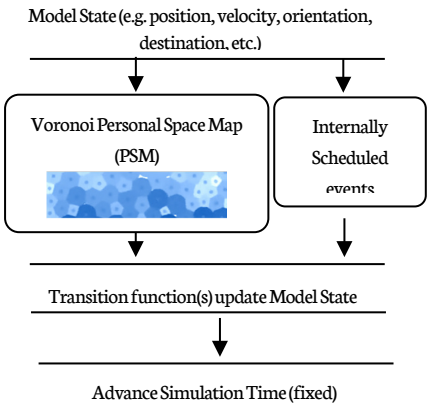
voroGeom.glsl

```
#version 150
// ----------- Geom I/O Setup ------------------//
layout(triangles) in;                // pedestrian positions point cloud
layout(triangle_strip, max_vertices = 256) out; // output cone per pedestrian

// ----------- User I/O Setup -----------------//
uniform int CONIC_RES;        // cone resolution
uniform int CONE_RADIUS;      // cone radius
```

```
uniform float ASPECT_RATIO; // drawing surface w:h ratio

in VertexData{
        vec4 color;
        int vertID;
    } VertexIn[];

out FragData {
  vec4 color;
} FragOut;

// ---------- Helper Functions -----------------//
void makeVert(vec4 pos){
        gl_Position = pos;
        EmitVertex();
}

void makeTriangle(vec4 v0, vec4 v1, vec4 v2){
        makeVert(v0);
        makeVert(v1);
        makeVert(v2);
        EndPrimitive();
}

// ---------- Shader Main ---------------------//
void main()
{
    int coneResMax = 40;              // max cone resolution
    // Cone Per-slice Angle
    float slice = radians(360.0/CONIC_RES);

    // Cone Color
    FragOut.color = VertexIn[0].color;
    vec4 halfcol = FragOut.color / 2;

    // Cone Head
    vec4 coneHead = gl_in[0].gl_Position;

    // Cone Side
    float factor = CONE_RADIUS*0.0016666;
    for(int i=0; i<coneResMax; i++){
       makeTriangle(coneHead,
       coneHead + factor * 2 * vec4(cos(i*slice), sin(i*slice) * ASPECT_RATIO,1,0),
       coneHead + factor * 2 * vec4(cos((i+1)*slice), sin((i+1)*slice) *
                                                          ASPECT_RATIO,1,0));
        if (i > CONIC_RES) break;
    }
}
```

Appendix B. Proposed CPD Engine Characteristics

The following is a brief outline of the differences between the simulation engine developed in Chapter 4 and an established formalism-based discrete-event simulation engine, CD++ [103].

	CD++	**Centroidal Particles**
		
Language	C++ (Atomic models) & MA (Coupled/Cell models)	Processing (Java)
Uses	General Purpose	Solely for modeling Local Crowd Dynamics
Dev Status	Mature, with hundreds of publications, books, and sample projects.	Early Prototype Stage
Modeling Method	Clean separation between *Model* and *Simulation Engine*	Separation not fully developed yet. Changing model behaviour often requires changing engine source code as well
Simulation Formalism	Cell DEVS (Discrete-event)	Agent-based Particles (Discrete-time)
Simulation Space	Discretized Cells (Eulerian Evaluation) *[suitable for accurate modeling of macroscopic/aggregate crowd behaviour]*	Free-moving Particles (Lagrangian Evaluation) *[suitable for accurate modeling of microscopic/individual behaviour]*
Engine Execution Overview **(Similar yet different)**		
Data Encapsulation & Neighbourhood Access	Once CD++ has exchanged all messages, each cell has access to its neighouring cell data (which were received through those messages)	Once the global PSM is computed, each entity looks only at its own pixels to evaluate personal space violations (doesn't access neighbouring/nearby particle values)

Appendix C. DISCLAIMERS, COPYRIGHT, AND MEDIA LICENSES

This work benefited from several contributions from the art and design community.

C.1 Fonts and Formatting

The thesis layout and final formatting was done in Microsoft Office 365 and customized to suit the author's needs, starting from a template graciously developed by the staff at the Carleton University Library. More about Carleton's thesis policies here:

- https://gradstudents.carleton.ca/thesis-requirements/

This thesis has been typeset using the Neuton font, designed by Brian Zick and made available under the SIL Open Font License.

- More about the font and its designer:
  https://fonts.google.com/specimen/Neuton
- The SIL Open Font License can be found at:
  https://scripts.sil.org/cms/scripts/page.php
- Equations and inline mathematical expressions were typeset in GUST's LM Math:
  http://www.gust.org.pl/projects/e-foundry/lm-math

C.2 Copyright

Unless otherwise stated, images and figures in this document either belong to the Author, or the Author has obtained a content publishing license.

FAIR DEALING (Canada) / FAIR USE (U.S.) NOTICE: material labeled "permission-pending" is used for non-commercial academic purposes only; with citations to the original source when possible; in accordance with the Canadian Copyright Act, and U.S. Copyright Law.

Appendix D. About the Author

D.1 Short Bio

Omar Hesham (BIT 2009, MSc 2012) is the Graphics and High-Performance Computing (HPC) Lead at Zetane Systems, an AI startup in Montreal, Canada. He pursued a PhD in Electrical and Computer Engineering at Carleton University's Advanced Real-time Simulation Lab, where he researched topics in agent-based simulation and visualization; and gained four years of teaching experience at various levels (TA, course instructor, and private tutor). In his spare time, Omar enjoys operating and developing content for Koldora, a Creative Commons-licensed educational video channel that illustrates concepts in computer graphics research and computational geometry while catering to the visual learner in all of us.

D.2 Contact

Homepage:	https://www.omarhesham.com
Academic CV:	https://www.omarhesham.com/cv
Email:	omar.hesham@carleton.ca
Twitter:	@omaraitch
LinkedIn:	linkedin.com/in/omar3D