MODELING OF COMPLEX LARGE-SCALE FLOW PHENOMENA

Abhinav Golas

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2015

Approved by:

Ming C. Lin

Dinesh Manocha

David Adalsteinsson

Anselmo Lastra

Jason Sewall

ABSTRACT

Abhinav Golas: MODELING OF COMPLEX LARGE-SCALE FLOW PHENOMENA.
(Under the direction of Ming C. Lin)

Flows at large scales are capable of unmatched complexity. At large spatial scales, they can exhibit phenomena like waves, tornadoes, and a screaming concert audience; at high densities, they can create shockwaves, and can cause stampedes. Though strides have been made in simulating flows like fluids and crowds, extending these algorithms with scale poses challenges in ensuring accuracy while maintaining computational efficiency. In this dissertation, I present novel techniques to simulate large-scale flows using coupled *Eulerian-Lagrangian* models that employ a combination of discretized grids and dynamic particle-based representations. I demonstrate how such models can efficiently simulate flows at large-scales, while maintaining fine-scale features.

In fluid simulation, a long-standing problem has been the simulation of large-scale scenes without compromising fine-scale features. Though approximate multi-scale models exist, accurate simulation of large-scale fluid flow has remained constrained by memory and computational limits of current generation PCs. I propose a hybrid domain-decomposition model that, by coupling Lagrangian vortex-based methods with Eulerian velocity-based methods, reduces memory requirements and improves performance on parallel architectures. The resulting technique can efficiently simulate scenes significantly larger than those possible with either model alone.

The motion of crowds is another class of flows that exhibits novel complexities with increasing scale. Navigation of crowds in virtual worlds is traditionally guided by a static global planner, combined with dynamic local collision avoidance. However, such models cannot cap-

ture long-range crowd interactions commonly observed in pedestrians. This discrepancy can cause sharp changes in agent trajectories, and sub-optimal navigation. I present a technique to add long-range vision to virtual crowds by performing collision avoidance at multiple spatial and temporal scales for both Eulerian and Lagrangian crowd navigation models, and a novel technique to blend both approaches in order to obtain collision-free velocities efficiently. The resulting simulated crowds show better correspondence with real-world pedestrians in both qualitative and quantitative metrics, while adding a minimal computational overhead.

Another aspect of real-world crowds missing from virtual agents is their behavior at high densities. Crowds at such scales can often exhibit chaotic behavior commonly known as *crowd turbulence*; this phenomenon has the potential to cause mishaps leading to loss of life. I propose modeling inter-personal stress in dense crowds using an Eulerian model, coupled with a physically-based Lagrangian agent-based model to simulate crowd turbulence. I demonstrate how such a hybrid model can create virtual crowds whose trajectories show visual and quantifiable similarities to turbulent crowds in the real world.

The techniques proposed in this thesis demonstrate that hybrid Eulerian-Lagrangian modeling presents a versatile approach for modeling large-scale flows, such as fluids and crowds, efficiently on current generation PCs.

To my family, none of this would have been possible without them.

ACKNOWLEDGMENTS

I would like to thank the many people who have made this dissertation possible. First and foremost, my advisor Ming C. Lin, who has supported my work – both sane and crazy – throughout my time at UNC, and provided valueable feedback in getting me past roadblocks, as well as improving my dismal writing skills. I also owe a great debt of gratitude to my committee members: Dinesh Manocha for opening my mind to new research problems and directions, Jason Sewall for teaching me how to think about parallel computing, and being an amazing help during deadlines and otherwise, Anselmo Lastra for showing me a new side of computer graphics I never knew existed, and David Adalsteinsson for helping me with math I could never wrap my head around. Without their help and support, this dissertation would not have been possible.

This dissertation is by no means an individual achievement, and I owe thanks to a lot of collaborators and coauthors. The biggest thanks of all to Rahul Narain, whose wizardry has given form to most of my ill-formed ideas. No less helpful have been my other collaborators: Sean Curtis, Jason Sewall, and Pavel Krajcevski.

I have been lucky to have friends who have served as sounding boards, and who I owe many thanks: Ravish Mehra, Shabbar Ranapurwala, Riha Vaidya, Gurkaran Buxi, Rachit Kumar, Stephen Guy, David Wilkie, and Anish Chandak. These are a few of the many people who have made my time at UNC memorable.

I must point out that this dissertation would not have existed had it not been the constant prodding of my father to pursue a Ph.D. especially when I had never considered one, my mother helping me every bit of the way, and my sister and brother for teaching me a good chunk of all I know.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

From rolling crowds, rising smoke, to water in fountains and rivers, our world is full of things that *flow*. These are very different from *rigid* objects like rocks, pieces of furniture etc. which maintain their shape. The key property of any flow is that it deforms continuously with no consistent rigid shape. Under this definition, a number of other commonly observed phenomena can be classified as flows. These include moving crowds of people, vehicular traffic, and even granular media like sand and other grains. Owing to their omnipresence nature, virtual environments often need to model flows for an immersive experience. This dissertation focuses on efficient modeling and simulation of such flows.

Given their variable nature, a single model cannot describe all flows – but most flow models do have significant similarities, particularly in their underlying representations and approaches used to simulate them. A multitude of models already exist for simulating different kinds of flows. For example, most Computer Generated Imagery (CGI) in movies uses fluid simulation models based on the Navier-Stokes equations [Bridson and Müller-Fischer, 2007]. These models trace their lineage back to the middle ages, when siege warfare motivated the study of flows to help understand the dynamics of moving large quantities of earth and water. Thus began research into the problem of *advection*, which describes the transport of any media by bulk motion of the flow. The advection problem for a conserved property $\mathbf{f}$ by a velocity field $\mathbf{u}$ can be described by the following differential equation:

$$\frac{\partial f}{\partial t} + \nabla \cdot (f\mathbf{u}) = 0 \tag{1.1}$$

This description covers everything from flowing water, to marching armies, and even to how air affects projectiles like bullets and arrows traveling through it. For flows consisting of discrete objects like people, this translates into simply moving all underlying objects

from one point in space to another. For these scenarios and others where the velocity is incompressible, the advection equation can be simplified to the more well-known form:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = 0 \tag{1.2}$$

For continuum flows like water solving the advection problem efficiently remains a key concern.

Their continuously deforming nature makes continuum flow advection a significantly more complex problem as compared to discrete rigid objects. Since the latter maintain shape and form, their motion can be completely described by defining the velocity of any two points on or within the object – or by two vectors at one point: velocity and angular velocity. Velocity at any other point within the extent of the object can then be derived analytically. This property does not hold for flows – velocity at any two points along a flow may not be correlated. This difference allows flows to exhibit the variety of behaviors they exhibit, but increases the complexity in simulating these very behaviors using a computer. Given initial values for velocities, a flow model defines their time evolution – usually using differential equations. A simulation then proceeds by updating the simulation state – velocity, position etc. – over short intervals of time, called *time steps*.

However, advection forms only one part of flow models. The flows discussed in this thesis also share one other component, one that maintains density. In fluids this is the "incompressibility" constraint, which mandates that a fluid must maintain a constant density. This constraint is traditionally modeled using a *pressure* force applied on the velocity field $\tilde{\mathbf{v}}$ resulting from advecting the velocity by itself. The nature of this pressure force is repulsive in high-density regions, and attractive in low-density regions. Mathematically it takes the form of a corrective term:

$$\mathbf{v}^{t+\Delta t} = \tilde{\mathbf{v}} - \Delta t \frac{1}{\rho} \nabla p \tag{1.3}$$

where $\mathbf{v}^{t+\Delta t}$ is the velocity at the next time step, ρ is the density of the fluid, and ∇p is the

gradient of pressure, which is computed as the solution to the equation:

$$\frac{\Delta t}{\rho}\nabla^2 p = -\nabla \cdot \tilde{\mathbf{v}} \tag{1.4}$$

This equation enforces the incompressibility constraint, and belongs to the class of *elliptic* differential equations. The key characteristic of such equations is that their solution at any point in space depends on the solution at every other point in space. This implies that information in an elliptic differential equation travels at infinite speed throughout the entire domain. An intuitive analogue to this is to consider a force applied on a rigid object, where the entire object moves while maintaining its rigid structure without deforming. The lack of deformation is due to the near-instantaneous propagation of the effects of the force through the object. Collision avoidance in crowds is also similar in nature; it ensures that virtual pedestrians do not collide with one another by maintaining a minimum distance, and thus local crowd density does not increase above a maximum threshold value.

As we increase the scale of the problem – be it in terms of spatial scale or increased detail – the complexity of the simulation increases, which translates into inaccurate results, significantly slower simulation, or both. There may be multiples sources of this complexity. The first source is the validity of the model itself. Motion of certain flows can exhibit patterns that cannot yet be captured using differential equations. Crowds of pedestrians for instance, can change their velocities discontinuously based on individual decisions of those very pedestrians. However, models do exist for simulating the navigation of virtual pedestrians in a medium-sized domain – like a small hall for instance – with an average pedestrian density of up to $3-4$ people per m^2. However, the same models are unable to recreate crowd behavior when domains are significantly larger (e.g. a large battlefield), or when crowd density is higher (e.g. in congested conditions when stampedes may occur). Existing crowd navigation models cannot model such scenarios since the criteria for pedestrian navigation are not yet modeled as a scale-invariant set of equations like the Navier-Stokes equations.

Thus, for flows such as crowds, any proposed model should be able to closely match observed real-world data.

The second aspect of this problem is the impact on performance. In order to make large-scale flow simulation viable, performance of proposed models must scale with increasing detail or size. In that context, the problem of solving elliptic differential equations like Eqn.(1.4) can be computationally more complex than advection. Advection relies purely on local information, like local velocity and local value of the property being advected. On the other hand, elliptic problems like incompressibility require incorporating information from the entire region in which the flow exists, i.e. the *domain* of the simulation. This global dependence is commonly modeled as a system of equations (like for equation (1.4)), or a global optimization problem as in the case of collision avoidance in crowds. In order to understand issues in scaling performance, it is important to understand the discretizations used by most models.

1.1 Modeling Flows

Any flow model must begin by defining certain properties: the domain, or the extent of the flow, and the velocity $\mathbf{v}$ and density ρ within the domain. For the flows discussed in this dissertation, the most common approach is to define sample points where these properties are explicitly defined, then use a set of basis functions to interpolate them at any other point in space. There are multiple options for choosing sample points, a common example being uniformly spaced samples along an orthogonal basis. The resulting discretizations are known as *finite difference* methods. These are popular since they are easy to analyze mathematically for correctness, and amenable to parallelization due to the regular nature of sample positions. Apart from how sample points are chosen for a discretization, one other important distinction is whether these sample points are static over time, or can move dynamically with the flow. The former leads to *Eulerian* models, while the latter defines *Lagrangian* models. For example, fluids can be simulated on Eulerian grids, or using particles

that move with the flow. In the latter case, particles also serve as markers on where the flow exists, removing the need for defining the property explicitly.

This dissertation looks into the problem of simulating flows at large scales using verifiably correct models that are efficient and scalable on desktop computers. However, to understand the complexity this poses, we must first understand these two representation classes and their impact on computational complexity.

1.1.1 Eulerian Representation

Eulerian representations discretize the flow *domain*, i.e. the space in which the flow exists, and may exist in the future. To do so, they define stationary points in the domain, where flow properties are sampled; values at other points in space are defined by interpolating sample values. Defining the behavior of a flow over time then requires describing the variation of flow properties at sample points over time. To illustrate this principle, consider a simple example of a stream with water flowing over rocks. Assuming the stream flow is in a stable state, the velocity of water at any particular point in the stream does not change with time. As a result, using an Eulerian discretization for this case, velocity values at any sample point will remain constant over time, even though water is moving. In addition, a simulation may focus on a small part of the stream – its domain – simulated with appropriate velocity boundary conditions. Simulation may then be performed by choosing sample points distributed uniformly throughout the domain.

The choice of the simulation domain and the granularity of discretization defines an Eulerian discretization. The chosen domain must be large enough to encompass a region where flow features of interest occur. However, due to limited computational resources, the placement and number of sample points is also a key decision. For most flow models, the domain of simulation can be made larger while keeping the number of sample points constant, without any computational penalty for one *time step*, i.e. the small duration of time by which the simulation advances in one step. However, the length of the time step must be modified

in accordance with stability and accuracy constraints. Though not required explicitly, most Eulerian models fix the domain, sampling positions, and desired time step at the beginning of the simulation for efficiency reasons. The static nature of these setups presents a problem for large scale flows, as covering the entire region where interesting flow features may occur, with sufficient resolution and a sufficiently small time step requires a prohibitive number of sample points.

The smallest scale resolved by Eulerian discretizations is defined by the minimal spacing between sample points. Thus, given certain computational restrictions, most Eulerian discretizations balance domain size with the separation between sample points. Thus, flows can either be sampled at fine granularity in small domains, or coarsely in larger domains [Fedkiw et al., 2001]. Adaptive discretizations are possible and have been proposed where sample points are added and removed based on the expected local details [Losasso et al., 2004, Adams et al., 2007]. Though computationally efficient as compared to naïve discretizations with uniformly spaced sample points, these methods have additional costs as well. Mathematical analysis of the method becomes more complex, and the irregular structure makes parallelization more challenging. In addition, choosing the appropriate sampling rate is a non-trivial undertaking, since predicting when and where detailed flow phenomena may appear is hard. This leads to constraints on the spatial variation of sampling rates. For example, in structures like octrees, sampling rates can vary only by a factor of two for adjacent samples for quality reasons, as heuristics maintaining sampling rates do not permit a larger variation.

This restriction on sampling rate variation has implications when simulating large scales. At sufficiently large scales, uniform sampling like the one chosen for the simulated stream or even adaptive sampling becomes untenable, as the spatial resolution of the flow becomes too low to resolve any features of interest, and choosing sufficient sample points becomes computationally intractable. For example, the chosen sampling rate may be insufficient for resolving a swirling motion in a container of liquid, or the sudden deviation of a few

pedestrian in a dense crowd.

1.1.2 Lagrangian Representation

An alternative approach to sampling flows is to follow a methodology similar to that used for rigid bodies. In such *Lagrangian* discretizations, sample points are advected with the flow itself. This leads to a very different definition of the domain, which now defines sample points not in the space which the flow *may* occupy, but only in the space the flow *does* occupy. For flows like water and crowds where spatial extent is restricted, Lagrangian discretizations can reduce the memory footprint of sampling. However, temporal progression of flow properties changes in such a discretization. To illustrate this, consider the previous example of the stream again. If the sample points where velocity is measured are Lagrangian and move with the stream flow, then velocity samples will show a temporal variation. At different time instants, the same Lagrangian sample may measure the velocity of water deep inside the stream, over a rock feature, or even falling off a cliff. To reconcile this viewpoint with the previously discussed Eulerian view, derivatives with respect to time are defined differently in the Lagrangian viewpoint, using *material* derivatives. These can be related to Eulerian derivatives by the following equation:

$$\frac{\mathrm{D}x}{\mathrm{D}t} = \frac{\mathrm{d}x}{\mathrm{d}t} + \mathbf{v} \cdot \nabla x \tag{1.5}$$

The above equation expresses the rate of change of a property as observed from a frame of reference moving at velocity $\mathbf{v}$ as the rate of change of the property w.r.t. a static frame of reference plus the effect of the motion – the spatial gradient of the property in the direction of motion. One complication that arises from the use of Lagrangian discretizations is ensuring consistent spacing of sample points. Over time, sample points can get clustered, leading to instability and accuracy issues. To remedy this, sample points must be redistanced or re-sampled periodically to ensure that the minimum and maximum separation between two

nearby sample points can be bounded.

In cases where the flow domain is a superset of the flow extent, Lagrangian discretizations may have a lower memory footprint than Eulerian methods. For example, a mass of water moving through a pipe always occupies a small finite region within the pipe. Over a period of time, though the flow may occupy a large segment of the pipe (the flow domain), at any instant of time the flow extent is always much smaller. Lagrangian discretizations for such a flow can discretize the smaller flow extent, while a static Eulerian discretization must work with the flow domain. In spite of these savings however, the growth rate of this memory footprint with increasing scale or detail remains high for both classes – growing at least linearly with the area (in 2D) or volume (in 3D) of the flow, and at quadratic or cubic rates with detail respectively.

Increasing memory footprint of the discretization used to resolve finer detail or increased spatial scale poses a computational issue for flow simulations, particularly due to the elliptic problems discussed earlier, which need to access a significant percentage of this memory footprint multiple times. Thus, to maintain reasonable throughput in the simulation, memory bandwidth must be sufficiently large.

1.2 Utilizing Parallel Desktop Architectures

To understand why larger memory footprints are poorly suited for shared memory architectures like desktops, it is necessary to look at the problem from an architecture perspective. Recent developments in processor architectures have attempted to increase computational performance by adding additional cores to processors, allowing more tasks to be run in parallel. This trend is evident in all compute architectures, including CPUs, as well as co-processors like the GPUs and the Intel® Xeon Phi. This push is motivated by the fact that a lot of workloads relevant for real-world applications are *compute-bound*, i.e. their performance is limited by the number of computing cores. However, with these parallel architectures, flow simulations can quickly become *memory-bound*, or more accurately memory

bandwidth bound, where computing cores spend most of the time waiting for data to be delivered by the memory subsystem, thus adding more cores does not help speed up the overall execution time of the simulation. This situation happens most often in elliptic problems noted previously, where systems of equations may need to access and update large regions of memory repeatedly in a global step.

Though this can be addressed by improving memory subsystem throughput, it is a complex problem, bounded by the cost of adding high-speed memory units close to computing cores. The solution commonly used is a hierarchy of caches, with slow DRAM chips at the bottom. In this hierarchy, latency and throughput limitations of DRAM are hidden by multiple levels of caches, each higher level of cache having lower latency of access, higher throughput, but smaller size. Owing to the cost constraint of adding bigger caches, in order to efficiently utilize these increasing computational resources, algorithms for flow simulation need to be architecture-aware in two key aspects.

First, any proposed algorithms need to be amenable to parallelization, preferably with minimal interaction or dependencies between parallel threads. The latter desire stems from the fact that highly parallel architectures like GPUs work optimally when each thread can run independently. Secondly, the working set of the algorithm and its overall memory bandwidth requirements need to be small. Ensuring this constraint helps ensure that at larger scales, computation does not become memory-bound, and the simulation can fully exploit all available compute cores. A smaller working set has a higher likelihood of remaining resident in high-speed caches, reducing latency of access, and improving overall throughput.

To incorporate these performance aspects, this dissertation proposes novel algorithms for scenarios where naïvely scaling existing models is untenable – either since the model cannot accurately capture phenomena at these scales like in the case of crowds, or since it leads to memory-bound computation as in the case of fluids. In case of fluids, this dissertation addresses computational scaling at large spatial scales. For crowds, it looks at modeling crowd behavior at large spatial scales, and at high densities.

1.3 Coupled Hybrid Simulation

In this dissertation I propose the use of coupled Eulerian-Lagrangian models for efficiently simulating phenomena at multiple spatial and temporal scales. These models are capable of simulating previously infeasible flow scenarios, as well as replicating certain flow behaviors not captured by existing models. An additional feature of these models is a lower memory footprint at large scales resulting in improved performance on parallel architectures, allowing the simulation of flows at larger scales than existing models under a limited computational budget.

These hybrid models can be used to simulate a number of real-world scenarios of interest in computer graphics and other fields, notably:

- **Simulating fluids at large scales**: Existing Eulerian models for simulating fluids scale poorly, with memory requirements growing with the 3^{rd} power of scale, and computational requirements growing with the 4^{th} power. These make large-scale simulations highly inefficient, to the extent that more computational power does little to increase performance.

- **Flow of pedestrians in large environments**: Most crowd models use a Lagrangian discretization, where each pedestrian is considered a discrete entity. These models include a trade-off between performance and accuracy w.r.t. distance – performing collision avoidance up to a nearby threshold distance. This is suitable for most real-world scenarios, but provides inaccurate results when the scene domain is large, while considering a larger threshold distance makes the algorithms scale quadratically with number of pedestrians. As a result, simulations are unable to maintain the desired real-time simulation rate.

- **Turbulent crowd flow in congested environments**: Congested crowds have conditions where the first assumption of simulated crowds fails; these simulations often

assume that pedestrians can avoid contact with others. However, in extremely congested environments where crowd density is very high ($5-6$ people per m^2 and higher), pedestrians collide almost continuously with others, giving rise to chaotic emergent behavior like stampedes, which cannot be replicated by existing models.

This dissertation discusses these scenarios in detail and proposes coupled Eulerian-Lagrangian models for efficient simulation.

1.4 Thesis Statement

My thesis statement is as follows:

Complex large-scale flows such as crowds and fluids can be simulated efficiently and scalably on current desktop hardware using novel algorithms that simulate multiple spatial and temporal scales using coupled Eulerian and Lagrangian discretizations.

To support this thesis, I present three methods; one method to efficiently simulate fluids at scale scales, and two methods to accurately simulate crowds at larger spatial scales, and at high densities.

1.5 Main Results

Large-scale Fluid Simulation

Simulating fluids in large scale scenes with high visual fidelity using state-of-the-art methods can lead to high memory and compute requirements. Since memory requirements are proportional to the product of domain dimensions, simulation performance is limited by memory access, as solvers for elliptic problems are not compute-bound on modern desktop systems. This is a significant concern for large-scale scenes. To reduce the memory footprint and memory/compute ratio, a different representation, called *vorticity* can be used, where vorticity represents rotational motion in fluid flow. When represented using a Lagrangian singularity representation, vorticity can represent rotational fluid motion very compactly.

In Chapter 2, I propose a hybrid model exploiting this insight; Using Lagrangian vortex singularity elements in the fluid interior, coupled with an Eulerian velocity-based simulation near all interfaces (with air or solid objects). The key benefit in using this model is that the memory footprint of the simulation is reduced significantly, as instead of discretizing the fluid volume, we discretize its surface, with additional cost proportional to the amount of vortex features.

The main contributions of my work are:

- Demonstration of a hybrid simulation algorithm that utilizes coupled Eulerian and Lagrangian discretizations at different spatial scales to simulate fluids
- A novel coupling algorithm that conserves vortex features in the fluid as well as mass and momentum of the flow
- An algorithm that can simulate large gaseous flows many orders of magnitude faster than existing methods, with significant gains for liquid flows

These results were published in [Golas et al., 2012].

Long-range collision avoidance in crowds

Local collision avoidance algorithms in crowd simulation often ignore virtual pedestrians (agents) beyond a neighborhood of a certain size. This cutoff can result in sharp changes in trajectory when large groups of agents enter or exit these neighborhoods. In this work, I exploit the insight that exact collision avoidance is not necessary between agents at such large distances, and propose modeling the avoidance problem at multiple spatial and temporal scales to perform approximate, long-range collision avoidance. This model can be used with both Eulerian and Lagrangian discretizations, as well as coupled Eulerian-Lagrangian discretizations for maximum efficiency.

This work is discussed in detail in Chapter 3. The primary results in my work are:

- A model that performs collision avoidance approximately at multiple spatial and temporal scales for both Eulerian and Lagrangian discretizations
- A hybrid model that utilizes both Eulerian and Lagrangian discretizations with coupled avoidance models to simulate collision avoidance in crowds at a wide range of crowd densities
- Simulated behavior generated by this hybrid model shows close correspondence to observed real-world crowd behavior
- Demonstration that efficient implementations of these models can simulate thousands of virtual pedestrians at interactive rates on current CPUs

This work was published as [Golas et al., 2013b, Golas et al., 2013a]

Crowd Turbulence in High-density crowds

With the growth in world population, the density of crowds in public places has been increasing steadily, leading to a higher incidence of crowd disasters at high densities. Recent research suggests that emergent chaotic behavior at high densities – known collectively as crowd turbulence – is responsible. Thus, a deeper understanding of crowd turbulence is needed to facilitate efforts to prevent and plan for chaotic conditions in high-density crowds. However, it has been noted that existing algorithms modeling collision avoidance cannot faithfully simulate crowd turbulence.

I propose a hybrid model for simulating crowd turbulence, that couples a Lagrangian agent model with an coarse Eulerian model for frictional forces arising from pedestrian interactions. The main results of this work are:

- A novel model that couples a fast-varying Lagrangian agent model with a slow varying Eulerian model for *inter-personal stress* – normal and tangential forces between nearby pedestrians

- Simulated behavior generated by this model shows close correspondence to observed real-world crowd behavior and known metrics
- Scalar implementation capable of interactive performance on current desktop CPUs

This work was published as [Golas et al., 2014].

1.6 Organization

The remainder of this dissertation is organized as follows. The discussion of hybrid models begins with the simulation of fluids at large scales in Chapter 2. This is followed by my work on collision avoidance behavior of crowds at large scales in Chapter 3. Finally, I discuss a model for simulating chaotic behavior in crowds at high-densities in Chapter 4. Chapter 5 concludes this dissertation by presenting a summary of this work, its contributions, and a discussion of future work.

Chapter 2: Large-scale Fluid Simulation using Velocity-Vorticiy Domain Decomposition

2.1 Introduction

State-of-the-art methods for fluid simulation, including velocity-based Eulerian methods and smoothed particle hydrodynamics, model the entire *spatial* extent of the fluid. Discretization of this space is often chosen to be able to sample sufficiently fine details under the restriction of limited computational resources. As a result, scenes with large spatial scales can only be simulated to coarse detail on PCs, relying on procedural methods to infuse detail. The simple computational kernels of these methods are largely *memory-bandwidth bound*, since domains of interest cannot reside in caches of current generation CPUs, and computational complexity cannot mask the cost of memory accesses. An alternate approach to modeling fluids is to model fluid *detail*, represented by the *vorticity* of the fluid, *i.e.* the curl of the velocity field. For incompressible flows, vorticity can be compactly represented by Lagrangian singularity elements. They are thus free of numerical dissipation, which can be a significant issue with Eulerian methods, and do not need to explicitly model the pres-

(a) **(b)** **(c)**

Figure 2.1: Examples of fluids simulated with my technique: (a) a city block hit by a tsunami (vortex domain in yellow) (b) seagulls flying through smoke (c) smoke flow around a sphere. I achieve up to three orders of magnitude of performance over standard grid-only techniques.

sure of the fluid. Though this leads to computational savings for scenes with unbounded fluid, robust and efficient modeling of obstacles or free-surfaces with two-way coupling using vorticity methods is challenging. Vortex singularity elements also serve as intuitive models for visual fluid detail, e.g. a smoke ring can be modeled as a vortex curve or filament.

These aspects make *detail* modeling of fluids with vortex singularity elements attractive, especially for large-scale scenes. To extend the applicability of these methods to free-surface fluids and scenes with deformable elements, I propose a hybrid domain-decomposition approach. Since Eulerian methods are adept at modeling such surfaces, coupling them with vortex methods can provide a robust, flexible, and efficient approach to fluid simulation. Particularly for scenes with large spatial scales but concentrated regions of detail, my approach can provide substantial computational and memory savings with reduced numerical dissipation, allowing simulations with more detail than previously possible.

It also poses significant challenges, the biggest of which is coupling heterogeneous methods in the same simulation while ensuring a consistent velocity field that matches the actual fluid velocity. I address this problem by separately coupling fluid flux and vorticity across simulation boundaries. I propose an iterative coupling algorithm that matches fluid flux across boundaries using appropriate boundary conditions for grid simulations and by creating vortex singularities on the boundary for vortex simulations. To accurately transfer vorticity across boundaries, I use a novel vortex particle creation algorithm and velocity boundary conditions for advection. It is important to note that maintaining vortex surface elements is computationally expensive and numerically ill-conditioned if surface mesh quality is poor. I develop an approach that addresses both of these issues by constructing meshes using grid faces, and efficiently computing fluxes induced by these face elements, even allowing pre-computation. I extend hierarchical approach of [Lindsay and Krasny, 2001] for faster computation of flux induced by vortex particles and surface elements.

Main Results: The key contributions of this work are:

- a hybrid fluid simulation algorithm that preserves vortex fluid features by using a com-

pact vortex basis with Lagrangian elements in the interior of the fluid, while enforcing arbitrary boundary conditions such as nonrigid obstacles and free surfaces using an Eulerian grid representation near boundaries;

- a novel two-way coupling between Eulerian Navier-Stokes simulations and Lagrangian vorticity simulations, which conserves vorticity over time and ensures continuity in the velocity and vorticity fields;

- a sampling algorithm to create a vortex particle representation of a given velocity field by minimizing residual in the L^2 norm; and

- an efficient and well-conditioned approach to computing strength of vortex surface elements, and an $O(m \log m)$ algorithm to evaluate flux using hierarchical methods.

The coupling techniques introduced in this chapter are quite general and can be used to connect different kinds of fluid simulation techniques in a heterogeneous domain decomposition framework. When applied to grid-based and vorticity-based methods, this method enables efficient simulation of large fluid volumes using a vorticity representation while supporting rich and complex interaction at boundaries. I demonstrate the benefits of this approach on several large-scale scenes (see Fig. 2.1) which would have a prohibitive computational cost using existing techniques.

These results were published in [Golas et al., 2012].

2.2 Related Work

Physically-based simulation of fluids has been a major focus of computer graphics research over the past decade. In this section, I briefly review the work in this area that is most relevant to this work. These may can be classified into two broad categories: *velocity-based* methods, which discretize the velocity field of the fluid directly, and *vorticity-based* methods, which describe it indirectly by its curl (vorticity) instead. These methods may

further classified in orthogonal categories based on the type of discretization, i.e. Eulerian or Lagrangian methods.

2.2.1 Velocity-based methods

Most of the *de facto* standard techniques for fluid simulation in computer graphics use a velocity-based representation. In such methods, one solves a discretization of the Navier-Stokes equations, which describe the evolution of the velocity field of a fluid over time.

A popular approach to solving these equations is by using finite difference methods on Eulerian grids — this was introduced to computer graphics by the pioneering work of Foster and Metaxas [Foster and Metaxas, 1996]. Stam [Stam, 1999] proposed semi-Lagrangian advection schemes to allow for unconditionally stable fluids, and Foster and Fedkiw [Foster and Fedkiw, 2001] developed methods for producing realistic, robust liquid surfaces in simulations. These methods have the advantage of being simple to implement and producing visually compelling results, including interactions with rigid and deformable objects [Chentanez et al., 2006, Batty et al., 2007, Robinson-Mosher et al., 2008]. Similar techniques have also been proposed using tetrahedral meshes [Wendt et al., 2007, Klingner et al., 2006, Chentanez et al., 2007] instead of rectilinear grids. However, Eulerian simulators have traditionally faced two challenges. First, they suffer from significant numerical dissipation, typically exceeding the desired viscosity in the fluid, causing flow detail to be undesirably damped out. Recent developments in improved advection schemes offer benefits in this aspect [Kim et al., 2007, Selle et al., 2008, Mullen et al., 2009, Lentine et al., 2011, Zhu and Bridson, 2005], while techniques for introducing additional detail have also been proposed [Fedkiw et al., 2001, Selle et al., 2005, Kim et al., 2008, Narain et al., 2008, Schechter and Bridson, 2008, Pfaff et al., 2010]. However, a second, more fundamental challenge is that Eulerian methods require voxelization of the entire simulation domain. The elliptical pressure projection operator needed to solve these equations demands that all pressure values be strongly coupled with each other, leading to

large computational and memory requirements for expansive scenes. Some recent work has attempted to address this issue, with level-of-detail representations [Losasso et al., 2004], coarse grid projections [Lentine et al., 2010], and model reduction [Treuille et al., 2006b, Wicke et al., 2009], but these are often incompatible with techniques for reducing dissipation.

An alternative approach is that of smoothed particle hydrodynamics, which models the fluid volume as a system of particles with pairwise forces between them. This approach has been employed for interactive simulation of liquids [Müller et al., 2003]. Similar to Eulerian grids, these methods can lead to a large number of particle primitives to sample fluid extent. In addition, enforcing incompressibility correctly can be expensive, owing to irregular computational elements. These concerns have been addressed partly with adaptive sampling [Adams et al., 2007] and predictive-corrective schemes for incompressiblility projection [Solenthaler and Pajarola, 2009]. Sin et al. [Sin et al., 2009] proposed a point-based approach that combines features from particle-based and grid-based methods. However, for the large scales under consideration, computational costs remain substantial for all such techniques.

2.2.2 Vorticity-based methods

Vortex simulations are a class of methods which were originally devised for aircraft wing design, and have recently begun to receive attention in computer graphics as well. These methods model the evolution of fluid *vorticity*, which is the curl of the velocity field, instead of velocity itself. With the exception of Elcott et al. [Elcott et al., 2007]'s Eulerian approach representing vorticity on a tetrahedral mesh, most of the methods in this category are Lagrangian, using singularities with the Green's function of the Laplace operator.

In this formulation, singularity methods can be applied, representing the vorticity distribution as a superposition of singularities such as particles [Chorin, 1973, Park and Kim, 2005], curves/filaments [Angelidis and Neyret, 2005, Weißmann and Pinkall, 2009], or surfaces/sheets in 3D space. Vortex singularities are

excellent basis functions for fluid velocity for a number of reasons. They offer a compact and exact representation of fluid velocity in unbounded domains, automatically ensure incompressibility, and are immune to numerical dissipation. Due to their Lagrangian nature, vortex singularity methods also allow easier user control of the simulation as compared to grid methods. These features have made these methods popular for interactive simulations [Angelidis et al., 2006, Weißmann and Pinkall, 2010]. Curve or filament representations also ensure incompressibility of vorticity, but can only model inviscid fluids. This is not the case with vortex particles, which allow viscous fluids to be modeled, at the cost of a slightly compressible vorticity field. The divergence-free constraint can be enforced iteratively through particle strength exchange methods [Cottet and Koumoutsakos, 1998], which can also model viscosity.

Enforcing boundary conditions requires the solution of a dense linear system. Though capable of arbitrary accuracy — conditions are enforced at mesh resolution — for nonrigid obstacles, precomputation of the linear system is impossible, which makes these the limiting factor in vortex simulations. Also, due to the singular nature of vortex singularities, proximity of elements has a major impact on the conditioning of this linear system, to the extent of rendering the system unsolvable due to poor conditioning. Large variations in the size of elements have a similar effect on conditioning. These issues make efficient and robust modeling of nonrigid obstacles non-trivial. It is also difficult to model the dynamics of free surfaces in this framework.

While there have been a number of hybrid simulation techniques combining, for example, rectilinear grids with tetrahedral meshes [Feldman et al., 2005], or grids with particle-based methods [Losasso et al., 2008], all these work solely with velocities, and I am aware of no work in computer graphics that allows combining velocity- and vorticity-based methods in the same simulation.

The remainder of the chapter is organized as follows. The hybrid domain decomposition algorithm is described in Section 2.3. I propose some tools for improving the efficiency of

Figure 2.2: Several seagulls flying through clouds of smoke to demonstrate airflow around their wings. On this scenes, my simulation achieved speedups of >1,000x

vortex simulations in Section 2.4. Finally, results and analysis of my implementation are described in Section 2.5.

2.3 Hybrid Fluid Simulation

Given velocity-based and vorticity-based methods have complementary advantages, I propose a hybrid approach that combines the respective strengths of both techniques. In particular, many different techniques have been proposed to support different types of boundary conditions for velocity-based methods, including free surfaces, deformable and thin objects, and two-way coupling. On the other hand, vorticity-based methods can compactly represent effectively infinite volumes of fluid where detail in the fluid motion is spatially limited. Therefore, I propose to combine both methods through a domain decomposition approach (Section 2.3.2), representing the fluid using velocity-based methods near boundaries such as obstacles and free surfaces, and employing vorticity-based methods in the large interior

region of the fluid.

However, the disparate nature of velocity and vorticity methods makes it challenging to combine them into a single heterogeneous simulation. An essential problem is that of coupling together the two representations at the interface between them, so that they represent a consistent velocity field for the entire fluid. I present a novel two-step coupling algorithm to address this problem: first matching normal velocities at the interface using an alternating scheme (Section 2.3.3), and then transferring vorticity information across subdomains through particle seeding (Section 2.3.4). As I show below, matching both velocity and vorticity across the interface is necessary to obtain a consistent and convergent simulation in this framework.

2.3.1 Subdomains

My simulation consists of Eulerian ($\mathbf{G}_i$) and Vortex ($\mathbf{V}_i$) subdomains. For notational simplicity, I use $\mathbf{G}$ and $\mathbf{V}$ to refer to the union of all Eulerian subdomains and Vortex subdomains respectively. Eulerian subdomains model the Navier Stokes equations using uniform grids, with velocity $\mathbf{u}$ sampled on a staggered grid. Operator splitting is used to integrate each term one by one, with a BFECC Semi-Lagrangian scheme for advection, explicit integration for external forces, and a sparse Poisson solve for enforcing incompressibility using pressure. Advection and Incompressibility solve steps take volume flux boundary conditions from the hybrid simulator, instead of zero flux enforced in traditional solvers. For more details I refer the reader to [Bridson and Müller-Fischer, 2007, Carlson, 2004].

Vortex subdomains ($\mathbf{V}_i$) model the evolution of vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, using the vorticity form of the Navier Stokes equations:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega}, \tag{2.1}$$

$$\nabla \cdot \boldsymbol{\omega} = 0, \tag{2.2}$$

under the assumption of no density gradients. Under this formulation, velocity can be expressed using the Green's function of the Laplace operator, giving rise to the Biot-Savart formula:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_{R^3} \boldsymbol{\omega}(\mathbf{z}) \times \frac{\mathbf{x} - \mathbf{z}}{\|\mathbf{x} - \mathbf{z}\|^3} d\mathbf{z}. \tag{2.3}$$

for a vorticity distribution in an infinite domain. Note the absence of a pressure term, as the velocity so defined is incompressible by definition. As mentioned before, this distribution can be represented as a superposition of discrete primitives such as points (particles), curves (filaments), or surfaces/meshes (sheets) giving rise three types of vortex singularity methods. I use a particle representation owing to its ease of use, and the possibility of modeling viscosity, which is not possible using filaments. Also, due to poor long term stability of ideal singularities, regularized singularities or "vortex blobs" are typically preferred. A popular choice is the Rosenhead-Moore kernel, a regularized form of the Biot-Savart kernel with a constant smoothing radius a to give

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_{R^3} \boldsymbol{\omega}(\mathbf{z}) \times \frac{\mathbf{x} - \mathbf{z}}{(\|\mathbf{x} - \mathbf{z}\|^2 + a^2)^{3/2}} d\mathbf{z}. \tag{2.4}$$

The smoothing radius governs the concentration of vorticity represented by a vortex blob, which affects the scale of vorticity features that can be represented by it.

The vortex particle algorithm proceeds by advecting vortex particles, perturbing particle strengths to model vortex stretching and viscosity, and creating vortex sheets to model obstacles. Particles whose strength falls below a minimum threshold are culled. For more details about advection, stretching and viscosity, I refer the reader to [Cottet and Koumoutsakos, 1998]. Obstacles are modeled by creating vortex sheets on their surface. [Weißmann and Pinkall, 2010] propose creating filaments along the edges of a polygonal mesh for this purpose, where the strength (Γ_i) of each filament (f_i) is determined to enforce zero flux through each face of the mesh, i.e. $\sum_i \Gamma_i flux_i(f_j) + flux_{\mathbf{V}}(f_j) = 0$. I utilize a similar algorithm relying on vortex particles instead of filaments to enforce flux.

In the hybrid case, I create sheets to match a non-zero flux, resulting in the equation for the strength of each filament: $\sum_i \Gamma_i flux_i(f_j) + flux_{\mathbf{V}}(f_j) = flux_{\mathbf{G}}(f_j)$. In case of multiple vortex domains, either all vortex sheets can be computed using one solve, or by iteratively solving for the strength of each vortex sheet separately. This choice usually depends on whether linear systems for each component can be precomputed or not.

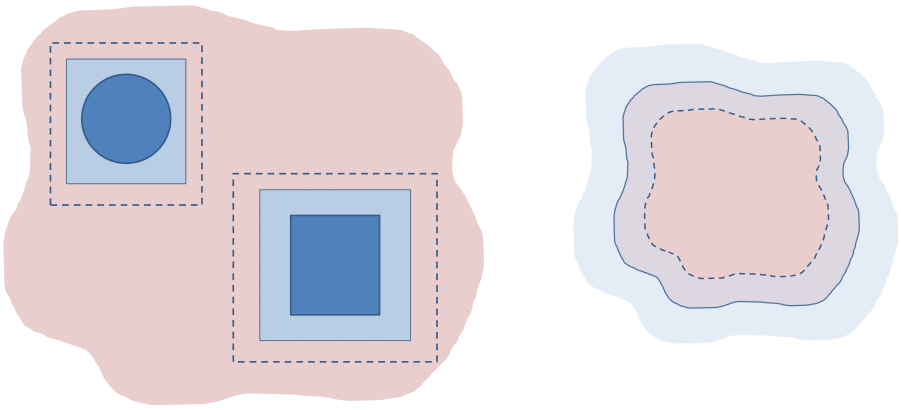
2.3.2 Hybrid Domain Decomposition

In my hybrid approach, I divide the simulation domain into a number of non-degenerate, overlapping subdomains, each of which is simulated using either the vortex method or Eulerian Navier-Stokes simulation.

It is natural to define one large region $\mathbf{V}$, consisting of the interior of the fluid at least a distance d away from boundaries, on which the vortex method is applied. This region consists of one or more disjoint subdomains $\mathbf{V}_i$. The other subdomains, labeled $\mathbf{G}_i$, use Eulerian Navier-Stokes simulations, and may contain boundaries such as static or moving obstacles and free surfaces. I assume that all Eulerian subdomains $\mathbf{G}_i$ are disjoint from each other (if not, I may merge any grids that overlap), and each of them overlaps with the vortex subdomain $\mathbf{V}$, in order to apply my coupling algorithm. Thus, the burden of supporting various boundary conditions is lifted from the vortex method, and placed on grid-based methods for which numerous techniques are available.

This decomposition is illustrated in Figure 2.3. For single-phase simulations like smoke, the domain consists of grids immersed in a vortex simulation, with the vortex domain extending to a user-specified distance d inside every grid. For free-surface simulations like water, the vortex simulation is embedded inside an Eulerian grid, with the boundary of the vortex domain being a distance d inside the fluid surface. As boundaries move, so do their corresponding subdomains, so that at all times the boundaries are contained entirely within grids and remain at least a distance d from the extent of the vortex subdomain.

Fluid velocities in the vortex subdomain are determined using the grid velocities as



(a) (b)

Figure 2.3: Decomposition of domain into vortex (red) and Eulerian grid (blue) subdomains for (a) single-phase flows, and (b) free-surface flows. Dotted lines denote grid region boundaries, while solid lines denote the vortex coupling sheet

boundary conditions. That is, the boundary of the vortex subdomain, which is a surface that lies entirely within the grid, is treated as an obstacle whose normal velocities are given by the grid velocity field, and a vortex sheet is computed on this surface to match the corresponding fluxes. Thus, the velocity field can be evaluated at any point in the interior using the Rosenhead-Moore kernel (2.4).

I define the timestepping scheme of the hybrid simulation as follows. Each subdomain is advanced independently over one time step Δt while assuming that velocities in the others are constant. Assuming constant velocities in other subdomains introduces an error on the order of $O(\Delta t)$ in both vortex and grid regions: in the vortex region because boundary conditions are determined by the grid; and in the grid because advection carries velocities in from the vortex region. In general, this magnitude of error is acceptable because the rest of the simulation (like most techniques for fluid simulation in graphics) is also first-order accurate. At the end of the time step, I couple the simulations back together by matching normal velocities at the boundaries, thus enforcing incompressibility, and matching vorticity in the interior of the overlap region. It can be shown that when the overlap region is simply connected, if both grid and vortex velocity fields have equal flux at the boundary and equal

vorticity in the interior, they must be identical [Cantarella et al., 2002]. For overlap regions with complex topologies, additional circulation constraints are needed. Thus, I can ensure that the combined simulation is consistent: where the grid and the vortex domains overlap, they agree on the fluid velocity.

In the following two subsections, I describe my proposed coupling techniques in more detail.

2.3.3 Velocity coupling

Each Eulerian solver stores a velocity field which determines the velocity $\mathbf{u}_{\mathbf{G}_i}$ at any point within its grid $\mathbf{G}_i$. Similarly, the vortex method determines the velocity $\mathbf{u}_{\mathbf{V}}$ at any point in its subdomain $\mathbf{V}$. By allowing domains $\mathbf{G}_i$ and $\mathbf{V}$ to overlap, I can enforce consistency between the corresponding simulations by ensuring that $\mathbf{u}_{\mathbf{G}_i}$ and $\mathbf{u}_{\mathbf{V}}$ are equal in the overlap region $\mathbf{O}_i = \mathbf{G}_i \cap \mathbf{V}$.

To enforce this constraint, I recall that the velocity field in any region is uniquely determined by the distribution of vorticity within it, and the flux at the boundaries of the region. Therefore, discrepancies between the velocities seen by different subdomains can only occur from having incorrect vorticity information in the interior or incorrect flux at the boundary. Assuming consistent vorticity in the overlap region, I need to ensure that flux across the boundary of the overlap region is consistent, i.e. velocity induced by both simulations is the same $\mathbf{u}_{\mathbf{G}_i} = \mathbf{u}_{\mathbf{V}}$ at the boundary .

Velocity can be matched at $\partial\mathbf{G}_i$ by enforcing the desired velocity as boundary conditions for the incompressibility projection step. To do the same with vortex simulation requires the creation of a vortex sheet $\mathbf{S}$ to match normal flux through $\partial\mathbf{V}$, i.e.

$$(\mathbf{u}_{\mathbf{S}} + \mathbf{u}_{\mathbf{V}}) \cdot \mathbf{n} = \mathbf{u}_{\mathbf{G}_i} \cdot \mathbf{n} \tag{2.5}$$

on the sheet, where $\mathbf{u}_{\mathbf{S}}$ is the velocity induced by the vortex sheet $\mathbf{S}$.

Thus the velocity coupling can be formulated as a fixed point iteration, one iteration of which can be expressed as follows:

1. Determine the velocity $u_{\mathbf{V}} + u_{\mathbf{S}}$ at the boundary $\partial\mathbf{G}_i$ of the Eulerian subdomain
2. Using $u_{\mathbf{V}} + u_{\mathbf{S}}$ as the boundary condition, perform the incompressibility projection on the grid $\mathbf{G}_i$
3. Determine the velocity $u_{\mathbf{G}_i}$ at the boundary $\partial\mathbf{V}$ of the subdomain $\mathbf{V}$
4. Compute strength of the vortex sheet $\mathbf{S}$ to match vortex velocity $u_{\mathbf{V}}$ to $u_{\mathbf{G}_i}$

Coupling iterations are performed till $u_{\mathbf{G}}$ and $u_{\mathbf{V}} + u_{\mathbf{S}}$ converge. For coupling multiple Eulerian subdomains with the vortex subdomain, this iteration can be performed in lockstep for every pair of overlapping subdomains.

This algorithm belongs to the class of methods known as the Schwarz alternating methods [Toselli and Widlund, 2004], which are commonly used in domain decomposition methods. Schwarz alternating methods are guaranteed to converge to a unique solution for second order PDEs, and thus this iteration ensures that the velocity field is consistent at the boundaries of all subdomains, and consequently in the entire domain.

2.3.4 Vorticity exchange

Velocity coupling on the subdomain boundaries will yield consistent velocities over the entire overlap region only if the vorticities seen by both representations are equal. However, even if the vorticities are equal in the initial conditions, they will gradually go out of sync over time, as vorticity is transported into the overlap region through advection. Therefore, to ensure consistency at all times, I need to account for this by exchanging vorticity between the grid and the particle domains. I derive this procedure by considering the two cases where vorticity is brought into the overlap region from the vortex particles and from the grid domain respectively.

From the vortex particle domain, vorticity enters the overlap region when a vortex particle flows in and crosses the boundary of the grid. In this case, transferring vorticity into the grid's velocity field can be done with appropriate boundary conditions. When I perform velocity advection on the grid using, say, a semi-Lagrangian scheme, it typically requires velocity information at locations outside the grid; I fill this in using the velocities determined by the vortex particle representation. Thus, as a vortex particle enters the overlap region, advection on the grid automatically pulls in its corresponding vorticity. Further, when the particle moves into the grid-only region, it may be deleted as its vorticity remains represented on the grid, or preserved to drive vorticity confinement.

Handling the transfer from the grid to vortex particles is somewhat more involved. As advection on the grid moves velocities around, the vorticity present in the grid-only region may be transported into the overlap region. This vorticity is unaccounted for by existing vortex particles in the overlap region, creating error in the representation. Therefore, I must insert new vortex particles in the overlap region to make the vorticities match.

I do this in a greedy fashion, at each iteration inserting the particle which best reduces the difference between the vorticity due to the particles and the vorticity present in the grid, denoted $\Delta\omega$. This is the particle at position x_p with strength α_p which minimizes the L_2-norm of the vorticity difference,

$$\varepsilon = \int_{G_i \cap V} \big(\Delta\omega_p(x) + k(x - x_p, \alpha_p)\big)^2 dV, \tag{2.6}$$

where k is the Rosenhead-Moore kernel. I found that the smoothing kernel used to obtain the Rosenhead-Moore kernel from the Biot-Savart kernel is well approximated by a Gaussian of standard deviation $a/2$. The choice of Gaussian smoothing is motivated by its smoothing properties in scale space, due to which features smaller than a are smoothed away, and computational efficiency afforded due to its linear separability property. Therefore, I smooth $\Delta\omega$ with the Gaussian and choose x_p at which the smoothed field attains its maximum

magnitude in the overlap region. Once x_p is fixed, it is straightforward to find the particle strength α_p which minimizes ε, as k is linear in α_p. I add this particle into the simulation and then repeat the process, until ε or $\|\alpha_p\|$ fall below chosen thresholds.

With this process, I maintain consistency between the grid and the vortex particles in the overlap region. To reduce the particle count, I also merge particles which are within a certain fraction of the smoothing radius a of each other. At every time step, I consider the $O(dn^2)$ cells in the overlap region. Though in the worst case, all these cells may result in new vortex particles, temporal coherency results in the creation of $O(dn)$ particles per step. I note that even though the Rosenhead-Moore kernel has infinite support, vortex particles can be created with finite information offered by the grid, since the distribution of vorticity around a particle decays rapidly with distance.

The outline of my resulting algorithm is shown in Figure 2.5.

2.4 Efficient vortex particle simulation

Vortex particle simulation is one of the major underlying components of my method. In this section, I discuss my implementation, including optimizations — some derived from theoretical concerns, others from practical ones. Efficient algorithms for purely Eulerian fluid simulation already exist, thus I do not delve into them here. As discussed in Section 2.2, the three main steps of vortex particle simulation are advection, stretching, and obstacle handling. Advection requires the computation of velocity at every particle position, and naïve summation using the Biot-Savart law leads to an $O(p^2)$ algorithm for p particles. [Lindsay and Krasny, 2001] propose a hierarchical summation approach to compute velocities, that reduces the cost of this step to $O(p \log p)$. However, in the presence of obstacles, the most expensive step is evaluating flux through the obstacle surface, and the subsequent linear system solve. In addition, mesh quality plays a pivotal role in the conditioning of this system.

2.4.1 Well-conditioned vortex sheet computation

The computation of vortex sheets that match normal velocities at their surface is a major step in vortex singularity simulation and my velocity coupling algorithm. Like any other vortex singularity element, the sheet induces a velocity field determined by the Biot-Savart law. In practice, these sheets are discretized as polygonal meshes, where each mesh face is a singularity with a distribution of vorticity. I use the approach proposed by [Weißmann and Pinkall, 2010], which assumes a constant strength for each face and represents each with a filament geometry defined by the edges of the face. To determine the strength of each face, a linear system is constructed that matches normal velocities or total flux at the face in accordance with equation (2.5). Due to reciprocity between filaments, the resulting matrix is symmetric. I first evaluate the net flux on each boundary polygon due to all vortex particles in the scene, giving $\mathbf{u_V} \cdot \mathbf{n}$, and then solve a linear system to compute vortex sheet strengths that will produce the desired change in flux, $(\mathbf{u_{G_i}} - \mathbf{u_S}) \cdot \mathbf{n}$.

In the presence of topological holes, fluxes are insufficient to uniquely determine the velocity field, and a circulation constraint must be specified for each hole. For example, if the sheet is a torus, normal fluxes alone cannot capture a purely tangential flow through the torus. For a formal discussion, I refer the reader to [Cantarella et al., 2002]. The condition number of this problem depends heavily on the quality of the underlying mesh. When matching flux through each face, a high variance in face areas can result in a poorly conditioned system. Similarly, due to the singular nature of these vortex elements, the minimum separation between faces also affects conditioning. Very low separation among a few faces can skew the eigenvalues of the matrix, making it poorly conditioned to the extent of losing rank.

To remedy this, I propose constructing vortex sheets using grid faces. Since I construct sheets at a distance d from any surface, this is equivalent to measuring distance using the infinity norm. The resulting mesh has two important properties, firstly that faces have zero variance of area, and that minimum separation between two faces is at least the grid cell

width Δx. Using these meshes results in linear systems with much better conditioning than those obtained using the traditional marching cubes algorithm, which relies on the 2-norm. In addition, the entries of any possible matrix constructed can be precomputed, since the set of all possible mesh faces is a finite set.

2.4.2 Hierarchical methods for flux computation

As noted earlier, flux computation is one of the major computational kernels of my algorithm. Since the velocity induced by a vortex element is the curl of its corresponding vector potential $\mathbf{\Phi}$, Stokes' theorem lets us express its flux through a polygon η as

$$flux(\boldsymbol{\eta}) = \iint_{\eta} \mathbf{u}(x) \cdot d\mathbf{A} = \sum_{i} \int_{\partial\eta_i} \mathbf{\Phi}(x) \cdot d\mathbf{l} \tag{2.7}$$

where

$$\Phi_p(x) = \frac{1}{4\pi} \int_{R^3} \frac{\boldsymbol{\omega}(\mathbf{z})}{\sqrt{\|\mathbf{x}-\mathbf{z}\|^2 + a^2}}. \tag{2.8}$$

To create the linear system I need to determine flux induced by a vortex particle or filament though a polygon. Though closed forms can be obtained for both, a better approach is to extend the hierarchical structure created to compute velocities. This structure can be easily extended to compute the vector potential induced by a set of vortex particles at any point in space. Then, a quadrature rule can be used to evaluate the integral in equation (2.7). I found that a 5th order Gauss-Legendre quadrature achieves a relative error < 0.01 when compared to the closed form.

Creating a hierarchical approach for filaments is non-trivial. However, using the same idea of evaluating integrals with Gaussian quadrature, a filament edge can be discretized into a number of vortex particles placed according to the quadrature sample positions, with their strengths being the filament strength multiplied by quadrature weights, strength vectors being aligned to the edge. This is especially simple for the mesh I create, since edge lengths are equal, but can be applied to any polygonal mesh.

Example	dt	Grid Resolution (Eulerian)	Simulation Time Eulerian (s)	Grid Resolution (Hybrid)	Simulation Time Hybrid (s)	Max. Vortex Particles	Average coupling sheet size	Speedup
Dam Break	0.0167	128x192x128	93.375	-	83.122	15860	490x490	1.123
		256x384x256	1313.94	-	514.41	32089	2316x2316	2.55
Smoke around	0.04	72x118x72	3.66	32x34x32	1.033	11516	1232x1232	3.54
sphere	0.04	144x236x144	47.712	64x68x64	5.959	23976	1232x1232	8.006
	0.04	144x236x144	47.712	64x68x64	29.7295	20105	4928x4928	1.604
	0.04	288x474x288	1277.37	128x136x128	177.044	34298	5640x5640	7.214
Wave	0.04	600x100x120	403.89	-	118.897	1567	2742x2742	3.4
City	0.04	312x60x210	87.51	-	66.801	7208	1722x1722	1.31
Seagull	0.04	520x256x520	NA	84x60x84	15.31	25674	1296x1296	>1000

Table 2.1: Single thread performance for my examples (All time values for one simulation step)

Thus the same hierarchical approach used for vortex particles can be used for vortex sheet velocity and flux computation, creating a unified representation for the vortex simulation. This suggests an iterative approach to solving the linear system to determine vortex sheet strength, since for a mesh containing m faces, instead of using $O(m^2)$ operations for matrix vector multiplication, I can do the same computation hierarchically in $O(m \log m)$ operations. This is especially helpful since for a grid of size $O(n \times n \times n)$, m can be $O(n^2)$, thus a matrix-vector multiply of complexity $O(n^4)$ can be performed with $O(n^2 \log n)$ operations. By using Fast Multipole Methods, this cost can be brought down further to $O(n^2)$.

As to the choice of iterative methods, I use the GMRES algorithm with restarts [Barrett et al., 1994], since the matrix is indefinite. I observe convergence in $O(n)$ iterations on average, $O(n^2)$ iterations in the worst case. Thus, the strength of a vortex sheet can be determined in $O(n^4 \log n)$ time in the worst case, and $O(n^3 \log n)$ on average.

It is important to note that the underlying bounding volume hierarchy constructed for this method is also used to accelerate neighborhood queries needed for the vortex stretching step, bringing its complexity down from $O(p^2)$ to $O(kp)$ where k is the number of neighbors considered.

2.5 Results

My method was used to simulate a number of challenging examples. Vorticity confinement was not used in any scenario.

Coupled simulation in unbounded space: Seagulls in flight (Figure 2.2) are used

to demonstrate how my method can be used for a scenario which would be challenging to simulate with either vortex or velocity methods alone. In this scene, three seagulls fly through four plumes of laminar smoke flow. To best demonstrate the effect of obstacle interaction, buoyancy is not modeled — the flow is kept laminar and does not become turbulent on its own. Vortex methods cannot be coupled robustly with deformable objects, while a domain of this size cannot be voxelized in desktop or mobile PC memory. The deforming wings of the seagulls induce vorticial details into the flow, which is carried by vortex particles even after the birds have moved on to different parts of the scene. Although I have applied a bound to the domain by determining the extents of all elements of the scene, the domain is unbounded in principle; the memory savings noted are lower bounds of what can be obtained with even larger scenes.

Preservation of vortex features: Figures 2.4a and 2.4b show fluid flow around simple boundaries where vortex features are preserved. It is important to note that speedup obtained for the sphere scene is not as significant as the seagull scene, since it has high-detail vortices in nearly its entire domain of interest, and thus cannot benefit as greatly from a compact representation of vorticity.

Tsunami striking a city (Figure 2.1a). A tsunami breaks over a city block, demonstrating fluid detail created by a high number of rigid objects; this scenario involves interaction with a large area of obstacle surfaces. The complexity of my approach scales with the total surface area of all elements in the scene — this scene is not expected to enhance performance as well as other cases. However, my approach helps in preserving flow details even in the absence of vorticity confinement.

Velocity coupling/comparison: In this scene, I show wave forming, cresting, and breaking. Visual fidelity and detail are maintained, resulting in a qualitatively similar simulation as previous methods, while affording performance benefits.

Scenes are modeled and rendered using Blender® and Maya®. Water is rendered as a mesh with an appropriate water shader, while smoke is rendered as a density field. Smoke

is advected passively through the flow, but can also be used to create density estimates for buoyancy forces.

2.5.1 Performance

For each of these examples, the vortex smoothing radius a is chosen to be close to the grid cell width; in all the examples shown here, this was in the range $[\Delta x, 4\Delta x]$. Correspondingly, the size of the overlap region d is kept between $[a, 2a]$ to allow sufficient information to create vortex particles, since the decay of the vorticity kernel is faster than linear. In practice, I have observed that an average of 2-3 coupling iterations are sufficient to allow the simulation regions to converge with an error of 10^{-4} in the L2 norm of the residual.

I measured the performance of my method on an Intel® Xeon® X5560 processor running at 2.8GHz on a system with 48GB of RAM and 8MB lowest-level cache. My method is implemented in C/C++, and some components use SIMD (4-wide single-precision SSE) instructions to improve performance and resource utilization.

The timing and performance results for all of my examples are shown in Table 3.1; here I show a performance comparison between purely Eulerian and hybrid simulation for single phase and free-surface simulations demonstrating the speedups obtained by using my method. Overall my approach gives varying speedups that are most pronounced for single-phase simulations. This is expected; free-surface scenes require regeneration of the coupling vortex sheet every step, which represents additional overhead in my technique not present in standard fluid solvers and proportionally mitigates (but does not eliminate) the overall advantage of my technique in efficiency.

Additional scope for optimization of these results exists, by using appropriate preconditioners for GMRES, and more optimized solvers. In addition, for free-surfaces, the same coupling sheet can potentially be used for multiple frames and linear solvers can be warm started with values from the previous time steps leading to more performance benefits.

2.5.2 Controlling Dissipation

As some of my examples may appear diffusive, I offer some insight into controlling diffusion. First of all, it is important to note that smoke sources in my scenes introduce purely laminar flow, with no model for buoyancy. This is done in order to highlight the preservation of vorticity across subdomain boundaries, since such an observation would be difficult in turbulent flow. Because my coupling algorithm matches flux and vorticity, any error in the latter would be observed as spurious vorticity at subdomain boundaries.

Three main sources of possible diffusion while using my algorithm are the choice of smoothing radius a, grid advection, and vortex stretching algorithms chosen. In figure 2.6 I show the difference in using PIC v.s. FLIP advection algorithms [Zhu and Bridson, 2005]. The choice of advection algorithms has an equally large impact in single-phase simulations, and some of my examples exhibit such dissipation as well. However, as shown by the example, this can be readily addressed by the use of less dissipative advection algorithms. The impact of advection algorithms on grid vorticity is analogous to the effect of vorticity stretching in vortex domains along with the choice of smoothing kernel. Accurate and stable stretching of vortex particles requires careful creation of new particles as needed, and the enforcement of the divergence-free vorticity constraint. Deviation from the constraint results in a tradeoff between numerical stability and diffusive behavior. Though a higher order kernel can improve accuracy, third order and higher kernels induce negative vorticity which is visually undesirable. The choice of smoothing radius is also important since it controls the highest frequency of vorticity detail that can be modeled.

2.6 Summary

I have presented a hybrid simulation algorithm for simulating fluids in large-scale scenes with a reduced memory and computational footprint that is proportional to the total area of all surfaces. It provides memory and execution time improvements from 2x–1000x, de-

pending on the scene. This performance gain is achieved via a novel algorithm that couples vortex singularity methods and Eulerian velocity simulations. I also propose a vortex particle creation algorithm, which creates particles to compactly represent a velocity field by minimizing the L_2 norm of the difference. My generalized approach also offers a flexibility to choose different regimes and numerical methods for distinct regions of a scene.

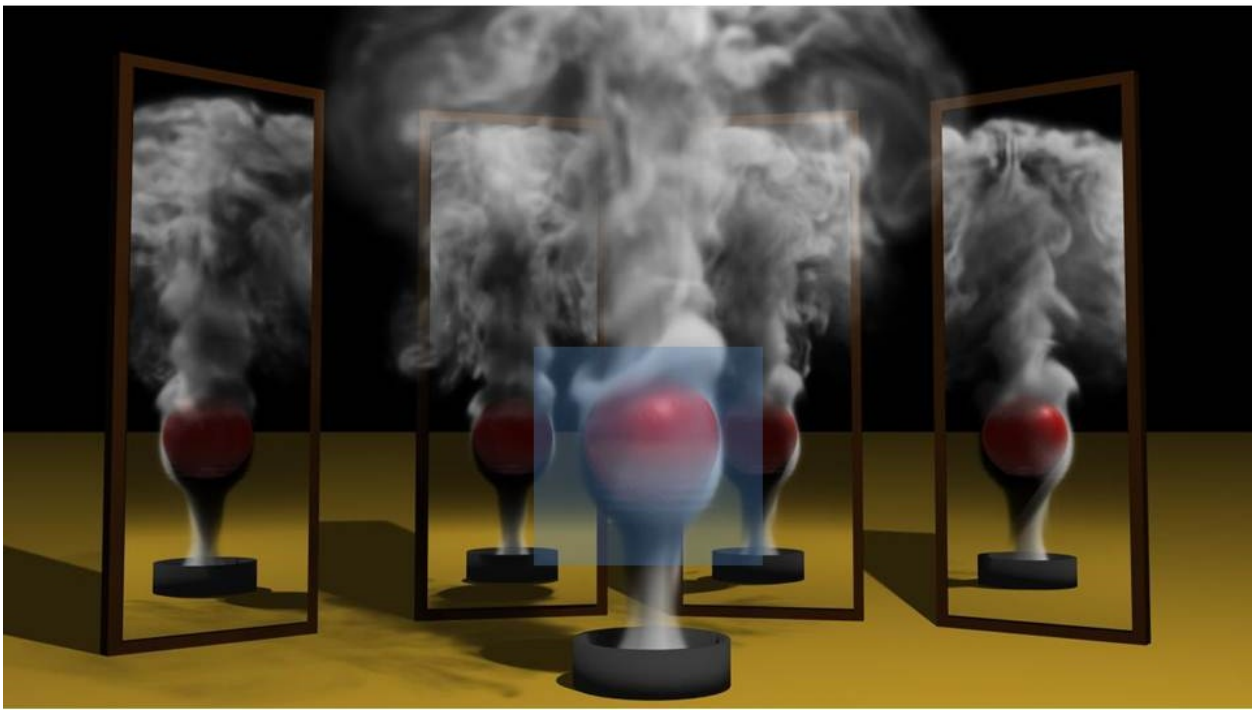
2.6.1 Limitations and future work

My method discretizes the vorticity space – rather than spatial extent – using one set of basis functions, defined by the smoothing radius a. To achieve high performance, my method relies on sparsity in this space. For scenes that contain dense vortex detail, my method's computational advantage diminishes, bringing its performance closer to other techniques. For specific static scenes, more compact bases could be derived, but such an approach would not scale to dynamic scenes without substantial pre-computation.

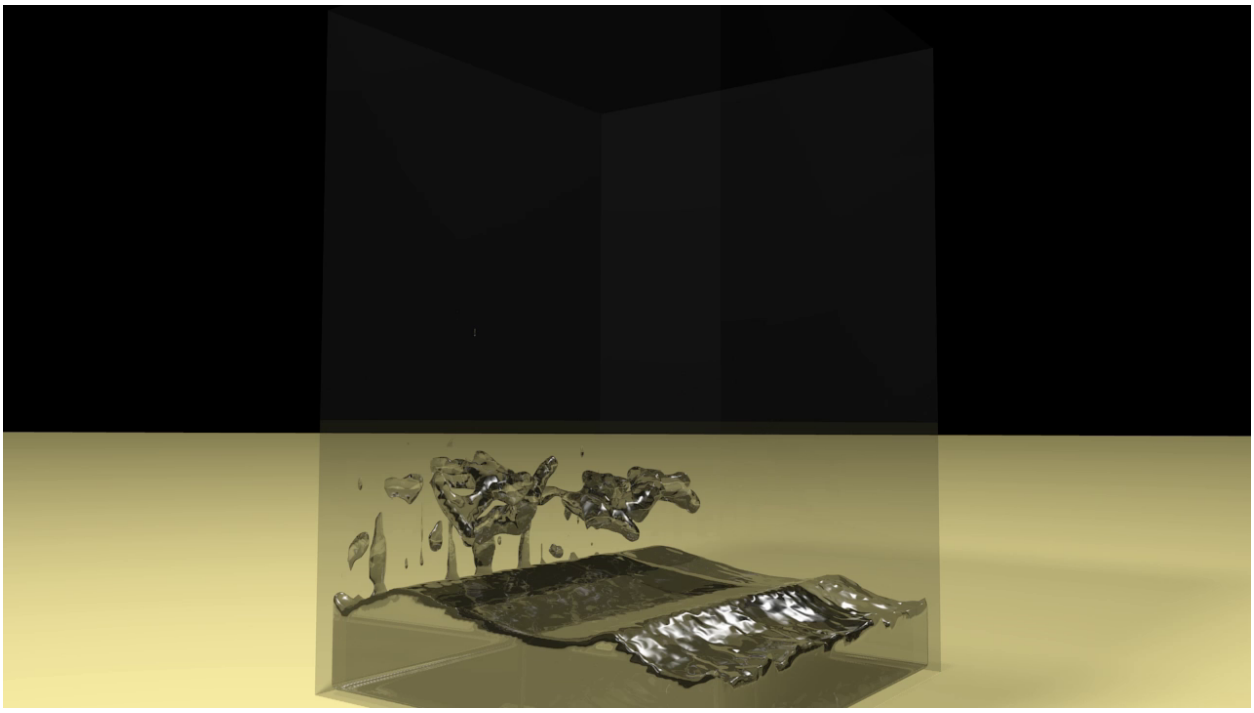
I also note that my particle seeding algorithm chooses a subset of possible vortex bases: those centered at grid cell centers. Though this does not reduce the applicability of the method, the possibility of expanding the set of bases to different smoothing radii and generalized particle placement would add to the efficiency and compactness of the vortex representation.

My formulation uses overlapping subdomains for coupling since it simplifies the coupling algorithm. A non-overlapping coupling, if possible, could allow the formulation of a vortex boundary condition for free-surfaces and two-way coupled obstacles. This would reduce the need for a full-fledged grid simulation, and afford greater flexibility in the choice for simulation methods.

Representing velocity fields as vortex particles opens the possibility of compact storage and manipulation of velocity fields. This can be used for artistic control of fluid velocity and even for accurate modeling of fluid turbulence. Further investigation of these avenues would be valuable for increasing fidelity and artistic control of existing algorithms as well.



(a)



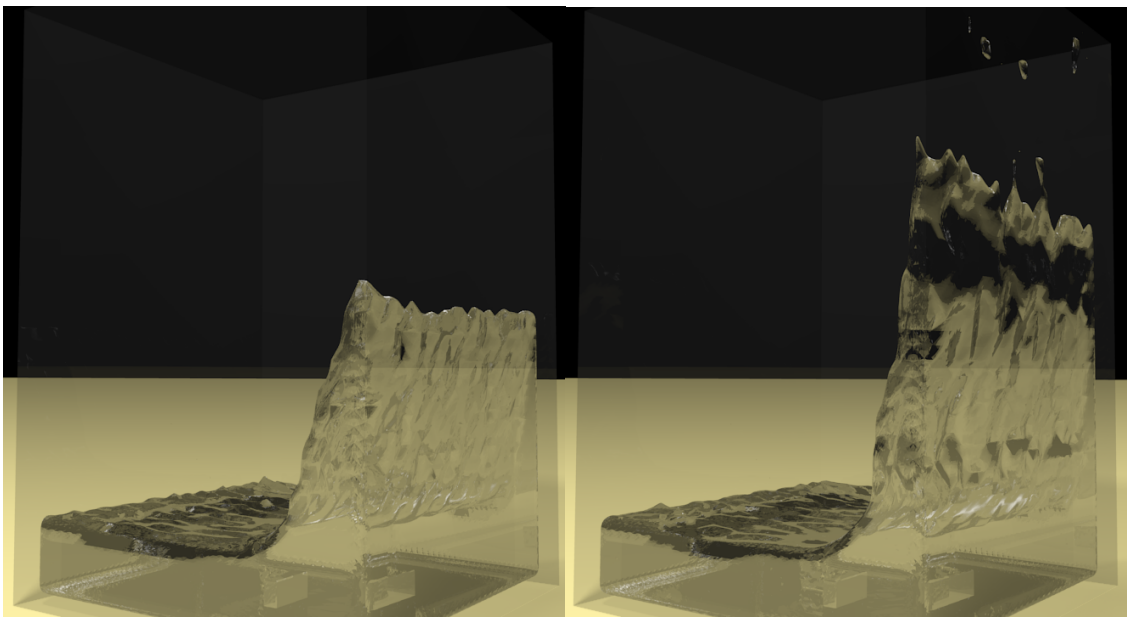
(b)

Figure 2.4: Examples of fluid flow and domain decomposition (a) static, for single-phase flows, (b) dynamic, for free-surface flows changing with the topology of water extent

For each time step:

1. Advance level-set surfaces, if any
2. Advect velocity fields $u_{\mathbf{G}_i}$ and apply any external forces
3. Convect vortex particles in $\mathbf{V}$
4. Repeat until convergence or maximum iterations:
   (a) Perform incompressibility projection on all grids $\mathbf{G}_i$ using boundary conditions $u_{\mathbf{V}} + u_{\mathbf{S}}$ from vortex particle simulation
   (b) Rebuild vortex sheet(s) $\mathbf{S}$ in the middle of grid-vortex overlap
   (c) Determine strength of vortex sheet(s) $\mathbf{S}$ to match vortex and grid velocities as per Equation (2.5)
5. Vorticity exchange
   (a) Create vortex particles to minimize vorticity residual
   (b) Perform vortex stretching to model fluid viscosity

Figure 2.5: The main steps of my method.



(a) **(b)**

Figure 2.6: Dam Break example at t=0.48s using PIC(left) and FLIP(right) showing maximum height reached by water

Chapter 3: Hybrid Long-Range Collision Avoidance for Crowd Simulation

3.1 Introduction

Long-range vision is critical to human navigation; in addition to avoiding nearby obstacles, the human visual system looks ahead to perform dynamic global planning and local navigation. By considering the distribution of other pedestrians and obstacles over large distances, people can anticipate overcrowded regions and navigate around them, thereby finding efficient, uncongested paths to their goals. Thus long-range vision greatly improves crowd flow and progress. Most existing work addresses either global navigation around static obstacles or local avoidance of collisions with nearby pedestrians, but often neglects the importance of long-range collision avoidance. Modeling long-range collision avoidance holds tremendous potential, to improve the flow of simulated crowds and help them reach their goals faster. To maximize utility, such a model should improve crowd flow without disrupting existing crowd simulation pipelines. Thus I consider this as the primary goal of this work.

The state of the art for this topic is a synthetic-vision based steering algorithm proposed by Ondřej et al. [Ondřej et al., 2010]. This method explores a vision-based approach for collision avoidance among walkers. It offers global efficiency among the agents in terms of overall walking time. Achieving reasonable performance is perhaps the key challenge of using this approach for large-scale, interactive applications. Even a parallel, GPU-based implementation cannot handle more than 200 agents at interactive rates. Complementing this approach, my work addresses this problem by offering a simple and efficient alternative that naturally extends existing local collision avoidance algorithms to provide long-range collision avoidance. My avoidance algorithm works based on the concept of *lookahead*, i.e. future agent states are approximated using past and present information; and these states

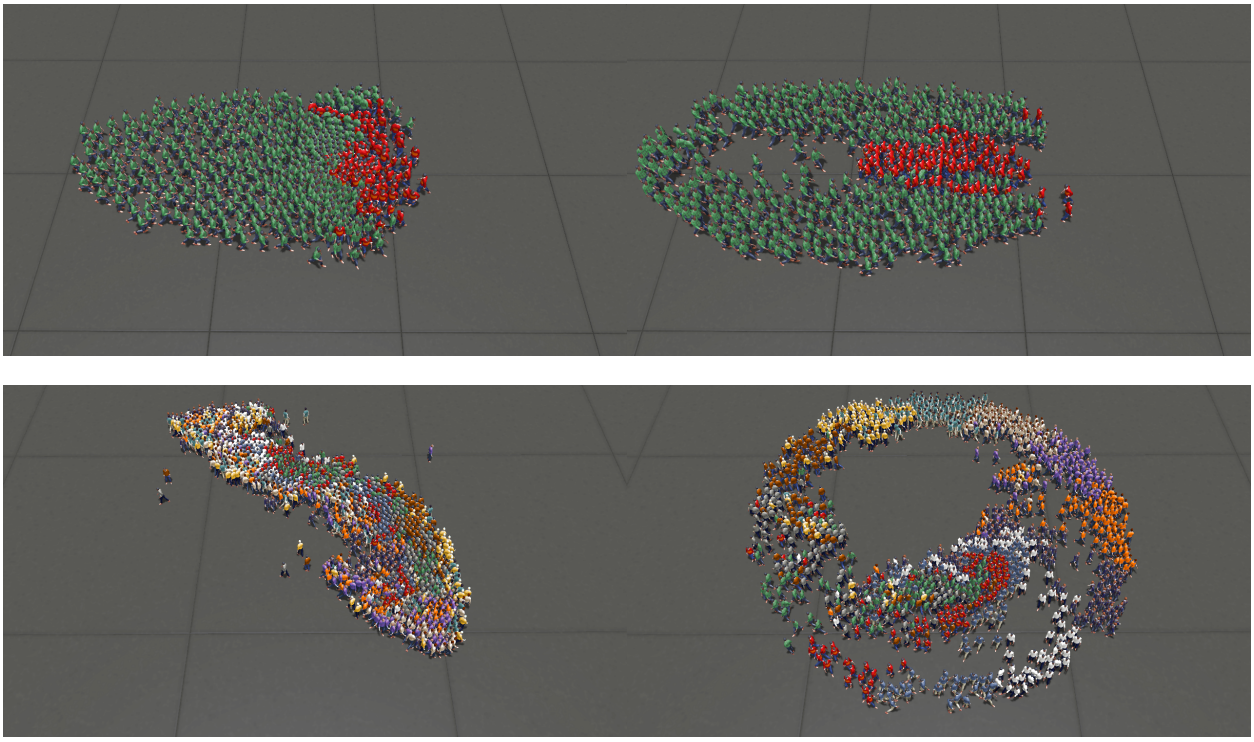


Figure 3.1: Results without lookahead (left) and with lookahead (right) for 2 demo scenarios. **Crossing**: (Top) shows two groups of agents seeking to exchange positions at simulation time $t = 10\,s$. Note how, with lookahead, the bigger group parts to allow smaller group through. **Circle**: (Bottom) shows agents on the edge of a circle heading to diametrically opposite points at simulation time $t = 40\,s$. Note significantly improved progress with lookahead.

are used to model possible collisions with agents not considered by local collision avoidance. My method is robust even in presence of obstacles and chaotic crowd motion, and provides improved correspondence to real-world behavior.

Collision avoidance algorithms can be broadly classified into two categories: discrete and continuum – based on the underlying representation of crowds. I formulate and demonstrate my lookahead approach for both classes of algorithms, as the problem is not restricted to either class. Though my demonstration in this chapter uses specific examples of continuum and discrete algorithms, my technique can be easily applied and generalized to other collision avoidance algorithms.

The use of continuum and discrete algorithms for collision avoidance also brings up a common issue with either class, namely their applicability to different ranges of agent density. Continuum algorithms (e.g. [Narain et al., 2009]) are ideally suited for medium to high densities, since the continuum assumption holds when pedestrian paths are tightly constrained by the nearby neighbors found at these levels of density. On the other hand, though discrete algorithms can be applied at any density, their computational costs escalate at high densities, along with numerical issues in some cases. Since crowds can exhibit an entire spectrum of densities even in any particular scene, these cases are not exceptions, but common occurrences. This insight suggests the need for an inexpensive hybrid scheme that locally blends both approaches for efficient collision avoidance over the entire spectrum of crowd densities. Such a scheme should choose the most suitable underlying algorithm for a particular simulation sub-domain, especially for problem cases where the wrong choice can lead to instabilities or other computational issues such as in scenarios of high or low densities, high variance of agent velocities, etc.

To address these problems, this chapter introduces the following main results:

- A novel approach for approximate long-range collision avoidance that can be used with discrete or continuum algorithms with minimal increase in computational costs.

- An inconsistency metric to measure oscillations in agent trajectories that can be used to

detect chaotic crowd behavior and curtail lookahead or to serve as a basis for comparing crowds, real or simulated.

- A hybrid algorithm that combines existing continuum and discrete collision avoidance algorithms to efficiently compute smooth local collision avoidance responses in any sub-domain.

- Comparison to real-world data that demonstrates improved speed sensitivity to density in simulated crowds using my algorithm, similar to human crowds as measured using the *fundamental diagram*.

My results show significant improvements in crowd progress with minor increases in computational costs In Fig. 3.1, I demonstrate my approach on two scenarios, where improvements in crowd behavior and progress are seen with less than 3x computational overhead. My approach is able to perform interactive long-range steering for both large, dense crowds and sparsely populated scenes, but also achieve interactive rates on a commodity laptop.

3.2 Background

I model a crowd as a set of agents, each of which has a specified goal position that it attempts to reach while avoiding collisions with other agents and with static obstacles in the environment. The standard crowd simulation loop that I and others often use is as follows:

1. For each agent, perform *global planning* to find a path to the goal that avoids collisions with static obstacles while ignoring other agents. Set the preferred velocity v_p along the direction of the initial segment of the path.

2. For each agent, perform *local collision avoidance* (LCA) to steer the preferred velocity v_p away from collisions with other agents, yielding the actual velocity v that the agent moves with.

Below, I briefly discuss some of the prior work relating to these two steps and discuss some of the data and techniques used for validating crowd simulation.

Most algorithms for global planning represent the connectivity of free space in the environment as a graph, and perform search queries for each agent to determine a collision-free path [Shao and Terzopoulos, 2007, Bayazit et al., 2002, Kamphuis and Overmars, 2004, Lamarche and Donikian, 2004, Pettré et al., 2005, Sung et al., 2004, Sud et al., 2007]. I do not diverge from previous work in this aspect.

A variety of models have been proposed for local collision avoidance among agents. These may use either discrete or continuum representations of the crowd. In discrete models, each agent considers other agents as individual obstacles, and attempts to avoid all of them simultaneously. Collision avoidance in this context can be formulated in terms of repulsion forces between agents [Reynolds, 1987, Helbing and Molnár, 1995, Reynolds, 1999, Heigeas et al., 2003, Lakoba et al., 2005, Sugiyama et al., 2001, Sud et al., 2007, Pelechano et al., 2007], or geometrically based algorithms [Fiorini and Shiller, 1998, Feurtey, 2000, Paris et al., 2007, Sud et al., 2008, Kapadia et al., 2009]; the state of the art involves treating possible collisions as obstacles in velocity space [van den Berg et al., 2008a, van den Berg et al., 2008b, Guy et al., 2009, van den Berg et al., 2009]. As considering the interaction of all pairs of agents becomes expensive in large crowds, such methods typically only take into account neighboring agents that lie within a specified radius, limiting the amount of lookahead possible. Guy et al. [Guy et al., 2010] propose a method to mitigate the computational cost of large neighborhoods by approximately clustering agents.

In a continuum-based approach, one first obtains from the set of agents a density field and a velocity field by accumulating the agents' positions and velocities on a background grid. This smoothed representation can then be used to compute the ideal motion of agents while avoiding regions of high density. The method of Treuille et al. [Treuille et al., 2006a] performs a global solve over the obtained density and velocity fields, giving compelling results

including long-range congestion avoidance effects. However, its computational cost increases steeply with the number of distinct goals in the simulation, making this approach unsuitable for interactive crowd simulation where agents may have many diverse goals. Narain et al. [Narain et al., 2009] propose a technique that prevents overcrowding in highly dense crowds, but it relies on purely local information and thus cannot plan around congestion at a large distance.

Validating crowd simulation has always been challenging. Historically, the presence of so-called "emergent phenomena" has been considered evidence which suggested correctness. Steerbench is a suite of tests designed to allow comparison of models [Singh et al., 2009]. While it suggests some basis for comparing models, it does not present (or use) a ground truth; there is no data of human pedestrians used in performing the benchmarks.

In the pedestrian dynamics community, the most common quantitative metrics for crowd behavior deal with aggregate crowd properties: flow and density. The relationship between flow and density has been referred to as the "fundamental diagram" [Weidmann, 1993]. In addition to this aggregate analysis, Guy et al. propose a new statistical metric for measuring how likely a particular pedestrian model is to match a given set of data [Guy et al., 2012]. In recent years, experiments have been performed with human subjects in various scenarios and several data sets have been made publicly available: "one-dimensional" pedestrian movement, uni-directional movement [Seyfried et al., 2005], uni- and bi-directional flow in a corridor [Zhang et al., 2011, Zhang et al., 2012], and flow through a bottleneck [Seyfried et al., 2009]. The value of lookahead is greatest in the case of conflict. As such, I do not perform validation against the uni-directional corridor or bottleneck flow. Instead, I perform analysis on an experimental setup similar to [Zhang et al., 2012]. Ideally, I would prefer data of pedestrians moving over a large space, however, even in this limited scenario, I can show that lookahead improves the behavior of the simulated crowds (see Fig. 3.12).

My approach aims to extend some of the existing work in LCA algorithms to support long-range collision avoidance queries. I accomplish this through the simple yet effective approach

of extrapolating agents' motion into the future. My algorithm is described in section 3.3, and I demonstrate its application to continuum and discrete algorithms in section 3.3.1 and section 3.3.2. In some cases lookahead may not be possible, particularly in presence of obstacles and turbulent flow. These are detailed in section 3.4 in addition to a novel metric for measuring oscillation and chaotic behavior in crowds. Furthermore, using discrete models alone can be extremely expensive in dense crowds, while continuum models are poorly suited to representing the motion of sparse crowds. In section 3.5, I propose a hybrid algorithm that blends results from continuum and discrete algorithms, producing consistently realistic results for both low and high densities under various velocity conditions. I demonstrate the advantages of my proposed techniques with examples in section 3.6, and compare my proposed lookahead based long-range collision avoidance algorithm with real-world data in section 3.7. Finally, I conclude with the limitations of my method, and discuss avenues for future work in section 3.8.

3.3 Lookahead for Long-Range Collision Avoidance

In this section, I describe my approach for efficiently computing long-range collision avoidance for both continuum-based and discrete agent-based crowd models. The problem can be summarized as follows: For each agent with a given preferred velocity v_p (as computed by the global planning stage), I wish to find an updated velocity v close to the preferred velocity v_p that avoids congestion in front of the agent at a range of distances from far to near, and also avoids collisions with neighboring agents. Influences from nearer agents should receive priority; that is, the agent should not divert itself to collide with a nearby agent in trying to avoid congestion farther away.

Given the extensive amount of already existing work on *local* collision avoidance algorithms, my aim is to take advantage of these existing techniques to solve the problem of *long-range* collision avoidance. In this chapter, I propose a general approach for decomposing long-range collision avoidance into a sequence of simple LCA queries. Thus, my algorithm

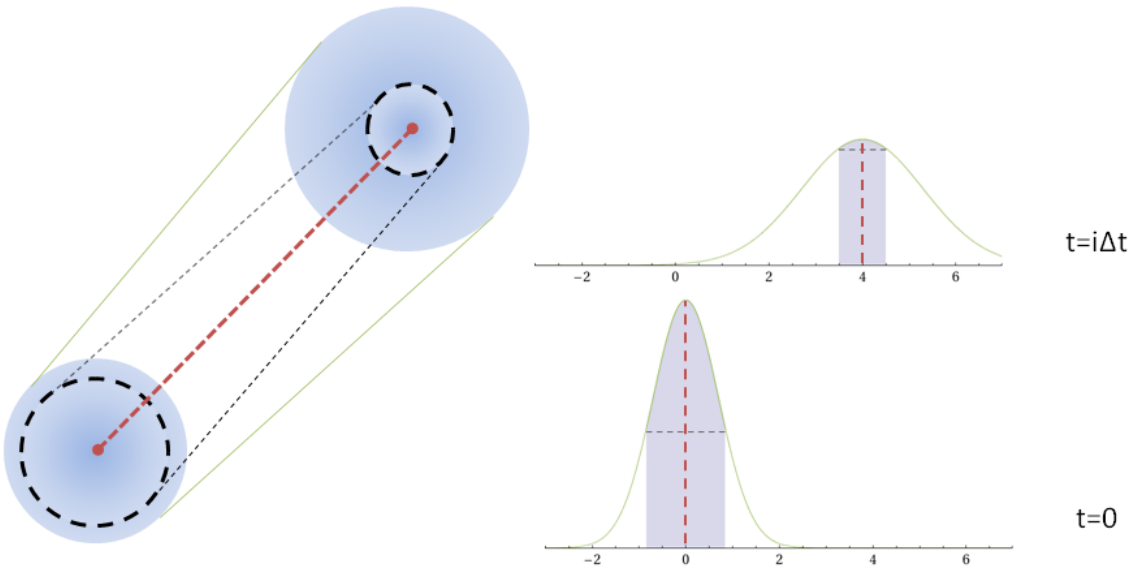


Figure 3.2: Effect of extrapolation in time from $x = 0, t = 0$ to $x = 4, t = i\Delta t$. Dotted line indicates effective radius ($P \geq 0.4$) of agent for collision avoidance, while spread of gaussian curve indicates splatting area for density field creation.

can re-use existing LCA algorithms with minimal increase in computation and coding effort. I show how to apply this approach to both the discrete and continuum settings, resulting in efficient algorithms for long-range collision avoidance in both cases. The crux of the idea lies in evaluating LCA queries not only on the current state of the crowd, but on its future state, estimated at a series of future times, enabling greater lookahead while using only local operations.

When an agent plans its long-term motion, it needs to estimate the motion of other agents over a large time interval into the future. While the future motions of other agents are of course unknown, they can be estimated with some degree of confidence using the agents' current velocities. To reflect the uncertainty in this estimation, I treat an agent's predicted location in the future not directly as a point, but as a probability distribution representing the expected probability of finding the agent at a given position. Intuitively, one can think of this as a traveling "blob" of probability, whose center $x(t)$ is linearly extrapolated from the agent's current position and velocity, and whose spread $\sigma(t)$ gradually expands over time, reflecting the increasing uncertainty as one looks further in the future.

In the continuum representation of the crowd, this has the effect of smoothing out the influence of any agent on the crowd density field, which enlarges the distance over which it influences the motion of other agents while simultaneously attenuating the magnitude of its effect. Thus, when an agent performs a short lookahead, only its nearby agents are significantly influential, while over a large lookahead, it only sees a smoothed-out density field that averages over many agents across a large area. In the discrete model, an agent is treated as a rigid, impenetrable "blocker" of fixed radius. When the agent position is uncertain, I consider a point to be blocked by the agent if the probability that the agent covers that point is at least a certain threshold p. As can be seen in Fig. 3.2, as the uncertainty increases, the effective size of the blocker decreases. This has the desirable effect that agents planning far into the future are not excessively hindered by the estimated motion of other agents, given that the latter is unreliable.

With this model for uncertainty, I can formulate the basic lookahead algorithm for long-range collision avoidance. The algorithm starts with the preferred velocities v_p obtained from the global planning stage, and performs a number of iterations $i = i_{\max}, i_{\max-1}, \ldots, 0$ with decreasing time horizons $\Delta t_i = 2^{i-1}\Delta t, i > 0$, and $\Delta t_0 = 0$. In each iteration, I extrapolate the state of the crowd by a time interval Δt_i into the future, perform an LCA query (with uncertainty) using the preferred velocity, and then replace the preferred velocity with the result of the LCA, as illustrated in Fig 3.3. In the last iteration, I set the lookahead Δt_0 to zero, so that I perform the standard LCA with no uncertainty, and thus maintain the collision avoidance guarantees of the underlying LCA.

With this scheme, agents are sampled in a larger radius than in the standard LCA query, and extrapolated queries are biased towards the direction of motion, providing lookahead. My approach smoothly merges the effects of distant and nearby avoidance considerations. Congestion avoidance with a long time horizon takes place in earlier levels, influencing the final result by updating the preferred velocity; nevertheless, this can still be overridden if needed to avoid imminent collisions with nearby agents, which are considered later in the

process.

The algorithm is defined formally in Fig. 3.4, where I denote by $v = A(v_p, v_c, x, \rho)$ an LCA query performed for an agent at position x with current velocity v_c and preferred velocity v_p in a region of local density ρ (people per unit area), producing a collision-free velocity v. In the following subsections, I apply my generic long-range collision avoidance algorithm to two examples of LCA algorithms, one continuum-based and one discrete, showing the broad applicability of my technique. I also describe some optimizations that are possible in the specific cases.

3.3.1 Continuum Lookahead

Continuum representations treat the crowd as a continuous distribution of density ρ and velocity v over space, through which any given agent must navigate. The knowledge of the density distribution enables us to determine congestion directly as regions of high density. It is well known that pedestrians walk slower in regions of high density [Fruin, 1971], a fact that can be formalized into a graph known as the *fundamental diagram* relating density, ρ, to a natural walking speed: $v_{\max} = f(\rho)$ Therefore, agents should navigate around overcrowded regions to avoid lowering their speed and maximize their rate of progress towards their goals. In this section, I first describe a simple algorithm that uses this idea to avoid congestion over a short time horizon, then extend it to look much further in time using my long-range approach.

Consider an agent that has a preferred velocity v_p pointing towards of the goal. Suppose over the planning time horizon Δt, the agent maintains a constant heading along a chosen direction $\hat{v}$ and walks at the maximum speed allowed by the fundamental diagram f. Then to first order, the density it passes through will change at a rate of $f\hat{v} \cdot \nabla\rho$, and so its displacement after time Δt will be

$$d(\Delta t) = f\hat{v}\Delta t + \frac{1}{2}(f\hat{v} \cdot \nabla\rho)f'\hat{v}\Delta t^2, \tag{3.1}$$

where f and f' are evaluated at the density at the current position. I choose $\hat{v}$ to maximize the progress towards the goal, $v_p \cdot d(\Delta t)$. This formulation reduces to the following optimization problem,

$$arg\max_{\hat{v}} \left(v_p \cdot \hat{v} + \frac{f'\Delta t}{2}(\hat{v} \cdot \nabla\rho)(v_p \cdot \hat{v}) \right) \quad \|\hat{v}\| \leq 1. \tag{3.2}$$

I can solve this problem using projected gradient descent, with the direction of the current velocity as the initial guess; this converges in less than ten iterations on average. This simple approach produces excellent avoidance results with maximal progress while still being computationally inexpensive. Though similar in spirit to [Narain et al., 2009], it avoids the need to calculate a global pressure to exert forces.

Before formulating the lookahead algorithm for continuum crowds, I first need to estimate the future densities of the crowd. In accordance with the uncertainty model, extrapolation further into the future requires that each agent's contribution to the density field be spread out over larger and larger areas, which can become inefficient with the traditional "splatting" approach. Instead, it is more efficient to represent future states on coarser grids, which will automatically have the effect of increasing the agents' effective footprint. Each successive grid is coarsened by a user-defined factor c, which represents the increase in uncertainty $\sigma(\Delta t)$ from one level of lookahead to the next. Thus, a pyramid of grids is constructed, where each level is coarser than the one below it by a factor c. Level i of the pyramid contains the future state of the crowd at time $t + \Delta t_i$.

With this representation, the lookahead algorithm as defined in Fig. 3.4 can be directly applied. For each cell at the bottom of the pyramid, I solve the LCA problem separately at multiple levels of the pyramid, starting from the top and cascading the solution at level i as the preferred velocity at level $i - 1$. Though this involves redundant computation being a depth-first approach, I prefer this approach as opposed to the breadth-first model where all cells at level i are solved, and the solution is cascaded down to level $i - 1$. This allows improved parallelism since there is no interdependence of solutions of neighboring cells, a reduced memory footprint since intermediate solutions do not need to be stored, and reduced

interpolation and smoothing artifacts.

3.3.2 Discrete Lookahead

I now extend the algorithm to discrete collision avoidance. Here, I use an extended reciprocal velocity obstacle (RVO) algorithm [van den Berg et al., 2009], which is implemented in the RVO2 library [van den Berg et al.,]. The RVO algorithm performs collision avoidance in velocity space, that is, the space of possible velocities that an agent may choose. In this space, I create a "velocity obstacle" for each neighboring agent, which represents the set of velocities that would lead to a collision with that agent. Then, choosing a velocity outside the union of all these obstacles ensures collision avoidance. Each obstacle is modeled as a constraint in a linear optimization problem to determine a collision-free velocity closest to the preferred velocity.

The RVO library requires the choice of a neighborhood radius R and time horizon τ, and only guarantees collision-free behavior within time τ with nearby agents no further than R distance away. This technique is limited to local planning in a small neighborhood, as increasing τ and R to large values degrades the performance of the method. To support long-range collision avoidance, I apply my algorithm to RVO-based LCA with minor modifications.

Instead of constructing trees for each future instant, I approximate future neighbor searches from the current state. Recall that my generic algorithm has multiple levels, and at the ith level, I consider a lookahead of Δt_i time into the future. In the discrete setting, I search for agents which may collide with the current agent within time Δt_i. Since distance between two agents can change at most by $2v_{\max}\Delta t_i$ in this time, where $v_{\max}$ is the maximum agent speed, the agents relevant at level i are those that lie at distances between $R + 2v_{\max}\Delta t_{i-1}$ and $R + 2v_{\max}\Delta t_i$ from the current agent at the present time. Once these neighbors are determined, I create velocity constraints using the agents' extrapolated future positions, with their effective radii r_f reduced by a ratio c for every step into the future. (This constant is the same as that grid coarsening factor for the continuum formulation, since

both density and effective agent radius are inversely proportional to the standard deviation of the probability distribution assumed for the agent.)

Now, all the long-range interactions considered at different levels are represented simply as constraints on the final velocity of the agent. Instead of solving the levels one after another, I may apply all the constraints simultaneously in a single RVO optimization. This means that only one optimization solve needs to be performed per agent, but at the expense of an increase in the number of constraints. I therefore adopt a level-of-detail approach to reduce the number of constraints by adaptively grouping distant agents into clusters.

As I extrapolate agents further in time, their effective radius reduces further, and thus have a decreased effect on agent velocity. Thus, it is prudent to cluster these agents both to improve efficiency, and to increase the probability of avoiding a future collision. I use a spatial hierarchy, such as a $k-d$ tree, over the agent positions to choose the clusters. Such a hierarchy already exists in the RVO implementation to support nearest neighbor queries, and so does not require additional computational effort. The nodes of the tree can provide suitable candidates for agent clusters.

When considering lookahead at level i, that is, until future time Δt_i, I only consider agents at a distance between $R + 2v_{\max}\Delta t_{i-1}$ and $R + 2v_{\max}\Delta t_i$. These agents should be grouped into clusters of size $\frac{\Delta t_i}{\Delta t}$ as shown in Fig. 3.5 . Instead of performing multiple searches to collect nodes at each level, I perform one tree traversal where the level of the node can be determined based on its distance from the agent. Thus, I perform a tree traversal where at any node C, I can determine its level i by checking which distance band it lies in, i.e. $d_C \in [R + 2v_{\max}\Delta t_{i-1}, R + 2v_{\max}\Delta t_i]$. However, every node may not form a good candidate since the distribution of agents in the subtree of this node may be sparse. Therefore I use a maximal separation as a quality measure of each node, i.e. $\max_{i \in subtree(C)} dist_i$, where $dist_i = \min_{k \in subtree(C)}(\|x_i - x_k\|)$. Though exact computation of this value is expensive, I compute an approximate value during tree construction by choosing the maximum of child values, and the separation between the nodes themselves. Thus, if a node satisfies this

quality constraint, it can be added as a velocity constraint.

Once I have a chosen a set of agents to form a cluster, I set its position x_C and velocity v_C as the mean of the positions and current velocities of its member agents. I choose the effective radius r_C of the cluster so that it covers all the expected agent positions, and is padded by the effective agent radius r_f for time $t + \Delta t_i$. In other words, for a cluster of m agents $\{x_1, x_2, \ldots, x_m\}$, I define

$$x_C = \frac{1}{m} \sum_{j=1}^{m} x_j, \tag{3.3}$$

$$v_C = \frac{1}{m} \sum_{j=1}^{m} v_j, \tag{3.4}$$

$$r_C = r_f + \max_j \|x_j - x_C\|. \tag{3.5}$$

With this definition, I define my discrete lookahead algorithm in Fig 3.6.

3.4 Curtailing Lookahead

The discussion thus far has been based on the assumption that agent states can be extrapolated to any future time. However in certain cases, the information available to long-range collision avoidance may be insufficient to create a reliable future state. In such cases, I curtail lookahead to the last reliable future time.

If the extrapolated path intersects an obstacle, I can infer that the agent has a plan to prevent this. However, determining this plan would require a query to the local planner, which in turn would require positions of all neighbors. This can recursively trigger local planning queries for an increasing number of agents, thus increasing computational cost. Thus choosing the simpler alternative of curtailing lookahead is a computationally efficient choice. This also has parallels among human crowds, which tend to plan within their visual range. Since obstacles restrict the visual range, it is natural to allow obstacles to curtail the planning region in a simulation as well.

The other scenario where I propose curtailing lookahead is when an agent has a chaotic trajectory - as measured over a small window of previous time steps. Extrapolating such a trajectory using any low-order polynomial function is likely to result in large approximation errors. This may even be indicative of a chaotic crowd, or an artifact in the underlying algorithm. In either case, the local crowd state is not amenable for long-range collision avoidance, as its future state is hard to predict.

I now detail the exact method utilized to address these two cases.

3.4.1 Obstacles

I allow extrapolation step i if and only if the path from the agent's current position to its extrapolated position does not intersect any obstacles, which can be modeled as a visibility query. Correct extrapolation of the agent's position would necessitate a model for agent response to an obstacle. As previously explained, this can result in a significant increase in computation. In addition, this may require information from the global planner. This information is typically not available. While it could be made available, this would require significant changes to the simulation pipeline, undercutting one of the goals of this approach: to extend current systems with minimum modification.

These queries can be efficiently implemented for both continuum and discrete simulations. Continuum simulations traditionally model obstacles as distance fields. These can be used to efficiently perform the needed intersection tests. The extrapolation curve representing the expected trajectory can be checked for collisions by finding the minimum distance to any obstacle in the scene. Such algorithms involve sampling the distance field at multiple points - either uniformly or adaptively - along the curve, with sampling controlled by the cell width of the distance field and the curvature of the curve itself. This approach can also be used for discrete simulations. However, in case the underlying simulation represents obstacles as discrete objects, an alternate approach can be used. In such a scenario, the problem can be modeled as a ray shooting problem. Using a hierarchical structure for static obstacles in the

scene, such visibility queries can be performed efficiently at run-time.

3.4.2 Chaotic Crowds

Extrapolation assumes that agent velocity is temporally consistent, i.e. the agent is expected to follow a predictable path. However, if the underlying crowd flow presents chaotic disturbances, then future agent states cannot be determined reliably with low-order extrapolation. Determining whether a crowd exhibits chaotic behavior locally or globally requires the knowledge of agent velocities for a time window. Given this information, I propose a new 'inconsistency metric' to detect chaotic behavior, and measure the suitability of the agent's state to extrapolation.

Inconsistency Metric

In case of an oscillating agent trajectory, extrapolation of an agent's position is not feasible due to the uncertainty in the agent's velocity. This oscillation can be measured by the acceleration of the agent, determined using its first-order approximation as

$$\delta v = \frac{v^i - v^{i-1}}{\Delta t}, \tag{3.6}$$

where v^i is the agent velocity at step i, and Δt is the time step between the two steps. Thus for each agent, a relative deviation can be computed at every time step i.

The inconsistency metric is computed using a set of n deviation vectors. These 2-dimensional deviation vectors are concatenated into a $n \times 2$ matrix M. I use Principal Component Analysis (PCA) to analyze the deviation vectors by computing the eigenvalues of the matrix:

$$C = \frac{1}{n} M^T M. \tag{3.7}$$

The eigen-decomposition of this matrix is of the form:

$$X^T \Lambda^2 X, \tag{3.8}$$

where Λ is a diagonal matrix with entries λ_1 and λ_2, as shown in Fig. 3.7 . The metric is then defined as the sum of the magnitudes of the eigenvalues:

$$\sigma = |\lambda_1| + |\lambda_2| \tag{3.9}$$

Low values of the metric indicate a smooth low-order trajectory of the agent, while higher values indicate chaotic paths. The rows of X give the corresponding eigenvectors x_1 and x_2, which can be used to further analyze the nature of the deviation.

To determine whether an agent can be accurately extrapolated, the inconsistency metric can be computed on a local neighborhood of the agent. In this case the matrix M is constructed from the deviation vectors of the agent and its neighbors over a time window of τ seconds, i.e. for k agents, data for f time steps, $f = \frac{\tau}{\Delta t}$ is concatenated into a $kf \times 2$ matrix M. The contribution of older samples can be also weighted to facilitate graceful degradation. For analysis of the entire crowd, the metric can be computed for all agents using a single matrix containing all deviation vectors for the given time window. Such analysis can be useful for comparing collision avoidance algorithms, preferring those which result in smoother paths as indicated by slowly varying velocity.

Curtailing

The inconsistency metric value as computed for the local neighborhood of an agent can be used to curtail the extent of lookahead. Variation of velocity in the vicinity of the agent indicates that the agent velocity is likely to change in the near future as well. For inconsistency values near zero, the agent can perform the allowed maximum $i_{\max}$ lookahead steps. As the metric value rises to a maximum $\sigma_{\max}$ and beyond, the maximum allowed

lookahead steps decrease to zero, at which point only local collision avoidance is performed. I use linearly spaced transition points, performing at most i lookahead steps if the metric value σ lies in the range:

$$\sigma \in \left[\frac{(i-1)\sigma_{\max}}{i_{\max}} \quad \frac{i\sigma_{\max}}{i_{\max}} \right] \tag{3.10}$$

for simplicity. The agent's choice of how far to look ahead is assumed to be based on local crowd conditions. Thus, only neighbors as defined by the local collision avoidance algorithm are used for computing the inconsistency metric for an agent. This metric value controls the agent's lookahead steps, as well as the lookahead of any agents considering this agent for collision avoidance. This implies that if a distant agent cluster has a metric value higher than that allowed for the lookahead step, then that cluster will not be considered for collision avoidance at that step, since that cluster cannot be expected to behave coherently as a single entity. Also, though the metric can be computed for every agent, its value is expected to vary smoothly, hence it can be sampled at certain points and interpolated for remaining agents, improving the efficiency of this computation.

3.5 Hybrid Crowd Simulation

Collision avoidance guarantees, provided by algorithms discussed thus far, are conditional on certain assumptions. In situations where these assumptions are violated, collision avoidance guarantees do not hold, and this can produce incorrect or at least visually unappealing results. For example, continuum algorithms work on the assumption that a crowd can be represented accurately as a density field. In low density regions, where this assumption does not hold, agents routinely collide with each other, or have to be pushed apart creating oscillatory behavior. In addition, grid representations of these fields can suffer from aliasing issues, resulting in damped or smoothed velocities. Discrete algorithms suffer from numerical issues at high densities, due to low inter-agent separation. For example, force based methods use repulsive forces that are inversely proportional to

distance. As a result, strict limits need to be enforced to prevent agents from colliding, either on the time step, or on the repulsive forces themselves. In geometric methods like RVO [van den Berg et al., 2008a, van den Berg et al., 2009], this is reflected in increased constraints on the solution, meaning that the algorithm needs to spend more computational time to converge, or risk failing to compute a collision-free velocity. In addition, computational costs of discrete algorithms are proportional to the number of agent neighbors. Since this cost rises sharply for high density regions, discrete algorithms can lose their performance edge for such scenarios.

My lookahead formulation performs successive local collision avoidance queries, thus such errors are likely to accumulate and cause significant issues. To address this issue, I propose a simple and efficient hybrid algorithm that blends discrete and continuum collision avoidance results. This is possible since problem cases for either class of algorithms do not overlap. The choice of algorithm is based on both density and variance in velocity. For varying density, there are three possible cases:

- $[0, \rho_{cmin}]$: Discrete collision avoidance
- $[\rho_{cmin}, \rho_{dmax}]$: Blend Discrete and Continuum collision avoidance
- $[\rho_{dmax}, \rho_{max}]$: Continuum collision avoidance

where ρ_{dmax} is the maximum density at which discrete collision avoidance can be applied, ρ_{cmin} is the minimum density for continuum collision avoidance, and $\rho_{cmin} \leq \rho_{dmax}$. By using linear blending, this can be expressed as:

$$v = v_{disc}(1 - w) + v_{cont}w \tag{3.11}$$

$$w = w_\rho = clamp\left(\frac{\rho - \rho_{cmin}}{\rho_{dmax} - \rho_{cmin}}, 0, 1\right) \tag{3.12}$$

where $clamp(x, min, max)$ clamps the value of x to the range $[min, max]$, and v_{disc} and v_{cont} are collision-free velocities generated by discrete and continuum algorithms respectively. In

my examples I use $\rho_{cmin} = 2$, $\rho_{dmax} = 4, \rho_{max} = 5.5$. The values of these parameters are guided by the densities observed in human crowds, being less than 5.5 people per m^2 in most cases. For choosing ρ_{cmin}, I rely on guidance from [Hughes, 2003], which states that the crowds can be represented as a continuum "*provided the characteristic distance scale between pedestrians is much less than the characteristic distance scale of the region in which the pedestrians move*". In medium-scale examples like those shown in this chapter, densities above $\rho_{cmin} = 2$ people per m^2 satisfy this constraint in the average sense. At higher density ranges, I observe qualitatively similar results using continuum and hybrid algorithms at densities higher than 4 people per m^2, which guides the choice of ρ_{dmax}.

To address the case of high velocity variance, I can define similar linear blending weights. In this case, blending weights are controlled by the standard deviation σ_v of the local velocity. I blend discrete and continuum velocities in a user-defined range $[c_1 v, c_2 v]$, where v is the local velocity, and c_1, c_2 are constants. In regions of low velocity variance, continuum avoidance is preferred, with discrete avoidance preferred in regions of low variance, where variance is σ_v^2. Then, in a manner similar to equation (3.12), I can define blending weights for this case as well:

$$w_{\sigma_v} = clamp\left(\frac{c_2 v - \sigma_v}{(c_2 - c_1)\, v}, 0, 1\right) \tag{3.13}$$

Note that this weight has a slightly different form to maintain the convention in (3.11).

I now need to combine these two weights to produce a single interpolation weight. Though a number of possible combinations exist, I choose $w = w_\rho w_{\sigma_v}$. This is biased towards a discrete solution, which ensures that the likelihood of regions with high velocity variance being simulated with continuum methods remains low. This weighing can be tuned by appropriately choosing c_1 and c_2. I find best results by choosing $c_1 = 1$, $c_2 = 2$. It is important to note that these two weights are not independent. As has been noted by [Narain et al., 2009] and others, velocity variance decreases at high densities. Thus adding a weight for velocity variance does not affect high density behavior significantly.

Scene	#Agents	Simulation			Time per Step (ms)
		Method	Duration (s)	Improv -ement	
Crossing	500	Disc	135.2	2.6x	0.98
		Disc LA	52.5		3.68
Circle	1000	Disc	347.2	2.2x	2.6
		Hyb LA	156.0		5.2
4 way	2000	Disc	-	-	5
		Disc LA			47.3
		Cont LA			6.8
		Hyb LA			16.29
4 groups	2000	Cont	107.1	1.5x	6.6
		Cont LA	72.28		6.89

Table 3.1: Single thread performance for my examples ($dt = 0.01s$). **Legend**: **LA** - with lookahead, **Disc** - Discrete, **Cont** - Continuum, **Hyb** - Hybrid. **Note**: $>$ 20 fps performance in all cases ($>$ 60 fps w Hyb) and 1.5x - 2.6x reduction in simulation duration with LA

3.6 Results

My algorithms were implemented in C/C++ using scalar code. In Table 3.1, I provide running times on a quad-core Intel Core i7 965 at 3.2GHz. Note that these times can be significantly improved by using appropriate vector instructions. I modified the RVO2 library to remove the restriction of maximum neighbors so as to provide collision avoidance guarantees for the neighborhood threshold supplied.

My lookahead and hybrid algorithms were tested on a number of cases. The first example scenes demonstrate the lookahead algorithm for the continuum and discrete cases. In the discrete case, two groups of agents – one bigger than the other – head towards each other on a collision course. Using my lookahead algorithm, the larger group of agents parts to allow the smaller group to go through, which is not observed in the traditional RVO algorithm. Though I encounter a 3.5x slowdown, the progress seen by agents is more than double, thus over the time of the simulation, the overall cost is less than 2x. In the continuum case 3.10, where 4 groups of agents attempt to reach diametrically opposite regions, more distant agents avoid the high density region in the center. In contrast to a simulation without lookahead, agents are able to reach their destination sooner, demonstrating improved crowd flow and

progress. An advantage of the continuum case is that lookahead is extremely inexpensive, as is seen with this example, where significant improvements in behavior can be seen at almost no cost

The second example scene demonstrates a 4-way crossing. With traditional collision avoidance schemes, a bottleneck quickly develops in the middle of the scene hindering progress and causes spurious behavior. Such behavior is significantly reduced with lookahead, both in the discrete and continuum cases. Using hybrid algorithm in this case provides two benefits. Oscillation of agents in low density regions is reduced as compared to the continuum algorithm, while significant performance benefits are obtained vs. the discrete algorithm. Though discrete lookahead slows down significantly due to the number of neighbors to be considered, the hybrid algorithm shows a performance benefit of 3x while retaining the same behavior. A visual comparison in Fig. 3.8 shows the difference in the 4 kinds of simulation.

I replicated the well-known circle demo with 2000 agents. In this scenario, agents are seeded on a circle and attempt to reach the diamterically opposite point as shown in Fig. 3.1 . At a 2x extra computational cost, I observe significant improvement in behavior and progress. Agents are able to reach their goal in less than half the time, which balances the computational cost.

To quantify the benefit of curtailing lookahead I analyze the circle (Fig. 3.1) and 4-groups (Fig. 3.10) demos. I examine how the choice of $\sigma_{\max}$ and $i_{\max}$ affects the simulation duration, i.e. the total time taken by all agents to reach their goals. Fig. 3.9 shows the results of this experiment. In each graph, dotted lines represent simulation duration of constant lookahead, while solid lines represent curtailed lookahead. In both scenarios, as $\sigma_{\max}$ increases, overall performance improves beyond constant lookahead, as demonstrated by lower simulation durations. As $\sigma_{\max}$ continues to increase, simulation performance asymptotically approaches constant lookahead. It is clear that constant lookahead in scenarios with inconsistent behavior causes agents to become overly conservative, reducing the efficiency of crowd

flow. By curtailing lookahead with appropriate parameters, agents reach their goals more efficiently, providing improvements of $10-100\%$. In addition to improvements in simulation performance, curtailing lookahead also results in reduced computation.

Evaluation of the inconsistency metric for curtailing lookahead increases computation by less than 5%, and this extra computation is balanced by the reduction in number of lookahead steps. Performance can be improved further by coarser evaluation, as noted in Section 3.4.2.

Most demos in this video use a maximum of 8 lookahead steps. As demonstrated by Fig. 3.9 , different scenarios and collision avoidance algorithms can demonstrate best performance for different values of $i_{\max}$ and $\sigma_{\max}$. My experiments indicate that $i_{\max}$ and $\sigma_{\max}$ can be optimized one at a time in order. This task if simplified by the fact that these optimizations do not exhibit local minima.

3.7 Comparison to Real-World Data

I compare the simulation result of my proposed algorithm against available open-source data. I use the data provided by [Zhang et al., 2012], specifically a bi-directional flow in a constrained environment, whose scene setup is shown in Fig. 3.11 . In this experiment, two groups of pedestrians travel down a narrow corridor 3.6 m wide in opposing directions. Though this presents a restricted environment, agents attempt to move in a constant direction due to which they can lookahead, and their future position can be determined with some reasonable accuracy. I compare the behavior of real pedestrians with that of simulated crowds first by examining the relationship between speed and density (commonly known as the fundamental diagram), and second, by using the metric defined in section 3.4.2. I provide comparison results using the discrete collision avoidance algorithm, RVO2 [van den Berg et al., 2009, van den Berg et al.,]. Fig. 3.12 demonstrates how adding lookahead causes the simulated agents to exhibit a density-dependent behavior similar to that of real pedestrians. Real pedestrians demonstrate a significant sensitivity to density. Speeds

decrease by a factor of 2–3 as crowd density increases for 1 person per m^2 to 3 people per m^2. However, a significant number of agents using RVO2 move with a maximum speed towards the goal as denoted by the number of sample points at $v = 1.6$ m/s in Fig. 3.12 (Top). In Fig. 3.12 (Bottom), I note that agents show the same downward trend of speed with increasing density as real pedestrians. Though the observed patterns do not match human behavior exactly, the graphs demonstrate that adding lookahead improves the behavior of simulated agents. Exact matching cannot be expected unless the underlying local collision avoidance algorithm models human behavior exactly. One caveat is that agents exhibit a decreased aversion to being in dense scenarios, as is indicated by significant clustering in density range of 3–3.5 people per m^2. This is partly due to lower speeds resulting in slower dissipation of congestion.

Simulated crowds in the experimental setup get to their goals in 98.4 seconds with lookahead, while taking approximately 169.88 seconds with purely local collision avoidance. For the same example, real crowds are able to navigate the same scene in 76.75 seconds. Thus, in addition to improved crowd behavior, lookahead results in a 1.73x improvement in crowd progress bringing the progress of the simulated crowd within 28.2% of observed crowds. Using a window of 10 time steps, i.e. 0.4 seconds, I computed my inconsistency metric for simulated crowds. Local collision avoidance produced metric values in the range $[0, 3.192]$ with a mean of 0.635, while crowds with lookahead resulted in values in the range $[0, 2.776]$ with a mean of 0.505. The inconsistency metric measures path smoothness as indicated by low-acceleration. By this measure, lookahead improves the mean path smoothness by 20.55%. Real-world observations in this experimental setup are made using head-trackers, which introduce a minor oscillation into agent positions. To remove this oscillation, I compute metric values for real-world data after smoothing the trajectories over a window of 2 time steps. The resulting data produces metric values in the range $[0, 2.213]$ with a mean of 1.45.

3.8 Summary and Future Work

I have presented a new, generic algorithm that can extend both existing discrete and continuum methods to provide a simple yet effective lookahead to achieve long-range collision avoidance for crowd simulations. This approach results in smoother crowd movement and exhibits an agent's tendency to avoid congestion that is often observed in real crowds. To quantify the smoothness of agent trajectories, I have proposed a novel metric. This metric also serves as the means to curtail the extend of lookahead in presence of chaotic crowd behavior. In addition, I have proposed the use of this metric for comparing crowd simulation algorithms based on smoothness, as well as comparing simulated crowds to real-world data. I have further introduced a hybrid technique that enables the simulation system to seamlessly transition between discrete and continuum formulations by locally blending the results and by optimizing for performance and quality of resulting simulations based on the local crowd density.

My stated goal is to improve crowd flow using long-range collision avoidance; measured as crowd progress. With the improvements proposed in this chapter, simulated crowds reach their desired goals 1.5x-2.6x faster (shown in Table 3.1) while using speeds similar to those used by real crowds in similar scenarios (shown in Fig. 3.12). As shown by these measures, lookahead brings the behavior of simulated crowds closer to real humans. However, in some cases I observe outliers. After detailed analysis, I believe these cases arise due to the underlying collision avoidance model. In case of Fig. 3.12 , solitary agents sometimes turn around to avoid collisions with oncoming groups of agents. This occurs when the space behind said agent is empty. Thus the geometrically optimal solution for collision avoidance as defined by [van den Berg et al., 2009] led the agent to turn around. This behavior is independent of lookahead and inherent in the underlying collision avoidance algorithm. However, lookahead allows agents behind such agents to preemptively avoid the oncoming group. Though this improves overall flow, it makes this artifact visually more

prominent.

In case of Fig. 3.1 , the larger group parts cleanly for the other. The underlying [van den Berg et al.,] algorithm optimizes with respect to hard constraints. Thus, constraints from near agents and far groups are equally important and must be respected. In this case, it leads to the clean separation of the larger group. To remedy this artifact, the formulation of constraints in [van den Berg et al., 2009] would need to be revisited taking into account of observed human behaviors.

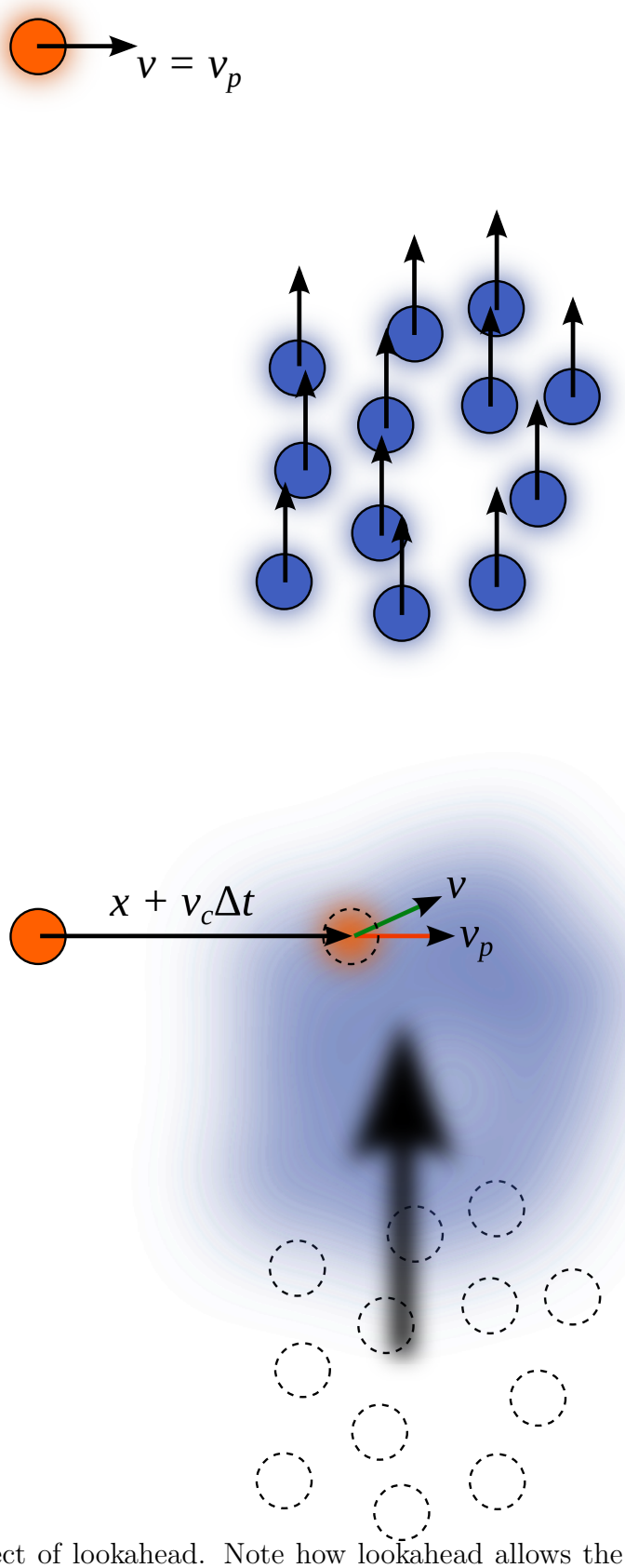


Figure 3.3: Effect of lookahead. Note how lookahead allows the orange agent to see the approaching crowd and adjust its velocity from preferred velocity v_p to v by incorporating information from the future crowd state at time $t + \Delta t$.

For each leaf node p

- Foreach level i in range $i_{\max}$ to 0 DO
  1. Determine future state of crowd $x_i = x + v_c \Delta t_i$
  2. Solve local collision avoidance problem $v = A(v_p, v_c, x_i, \rho_i)$
  3. Update preferred velocity $v_p = v$
- END Foreach

Figure 3.4: Lookahead Algorithm using LCA algorithm A.

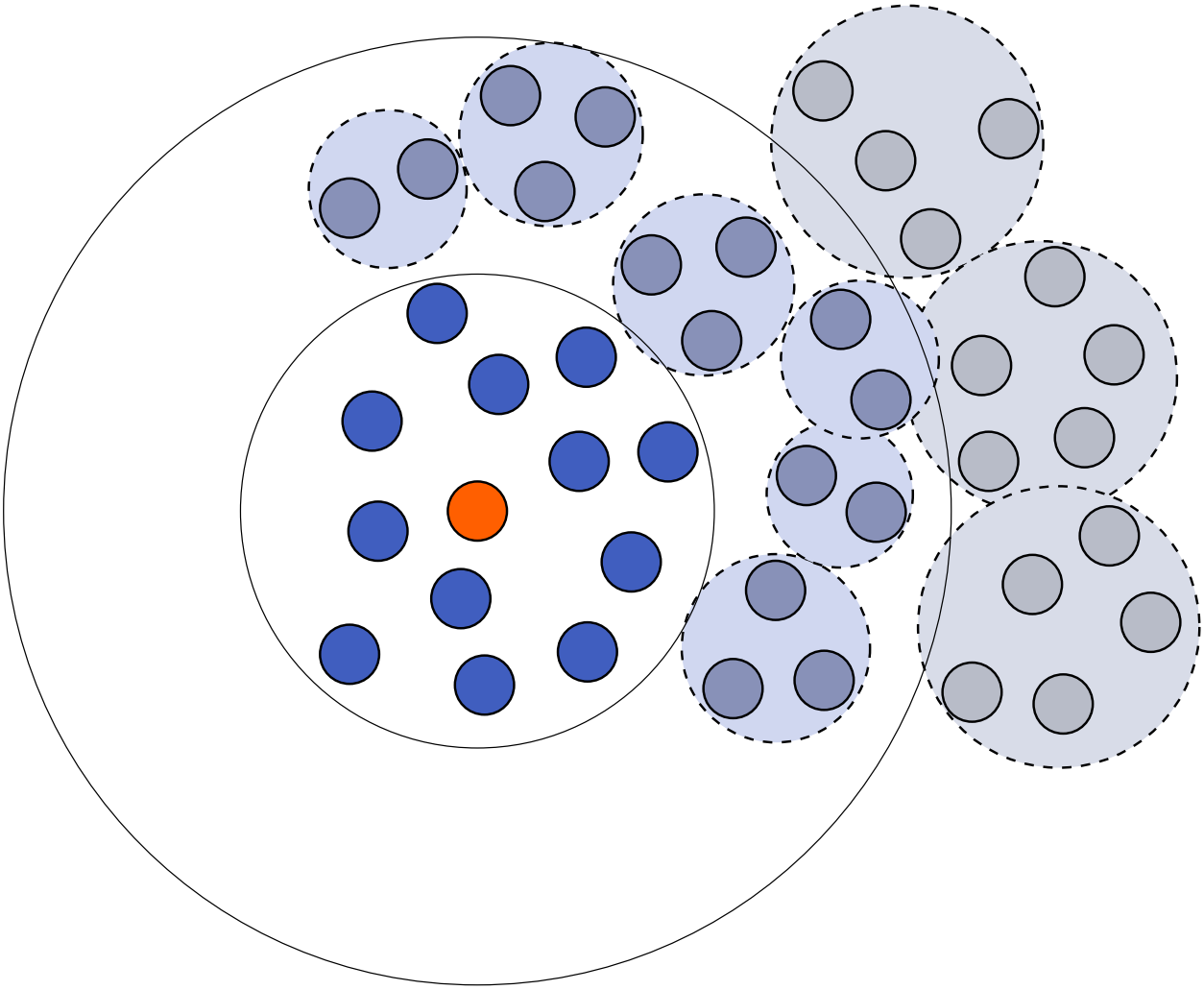


Figure 3.5: Distant agents can be clustered for collision avoidance, cluster size being proportional to distance. Since possible collisions with distant agents lie in future timesteps, extrapolated future agent states have high uncertainty, and hence small effective radii, making individual avoidance inefficient.

For each agent j, traverse the tree T starting from the root node, at each node C:

- If node C does not satisfy maximal separation constraint recurse on its children
- If constraint is satisfied and its level $i \leq i_{max}$, formulate velocity obstacle constraint for node C

where i_{max} is the highest tree level considered.

Figure 3.6: Lookahead Algorithm using RVO.

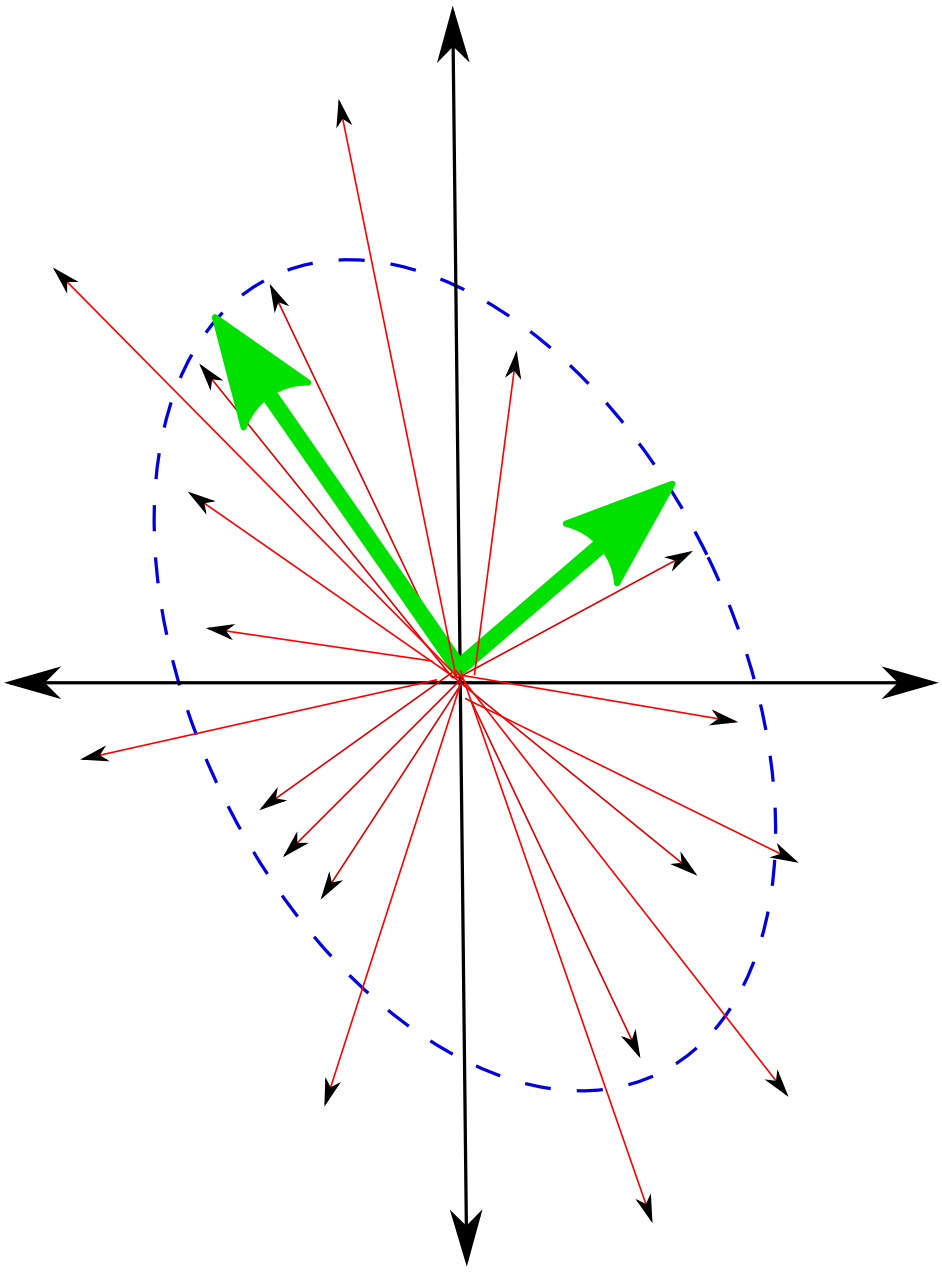
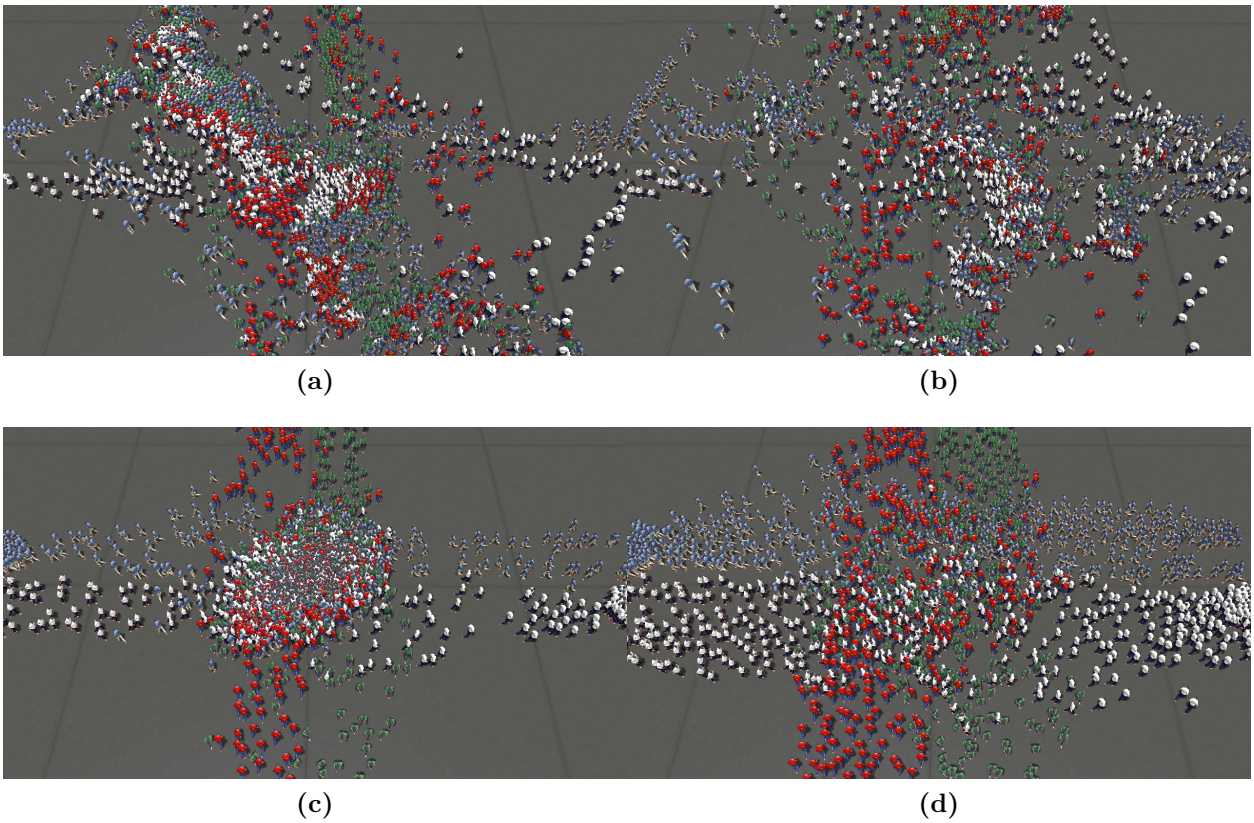


Figure 3.7: My proposed inconsistency metric σ is computed as the sum of the eigenvalues λ_1, λ_2 of the given deviation vectors δv (in red), which represents the variance of these deviations. The eigenvectors x_1, x_2 (in green) corresponding to these eigenvalues represent the principal components of the space of deviations.



(a) **(b)**

(c) **(d)**

Figure 3.8: 4 way crossing of agents with 2000 agents. (a) Discrete (b) Discrete with lookahead (c) Continuum (d) Continuum with lookahead. Note lack of agent buildup in cases with lookahead.

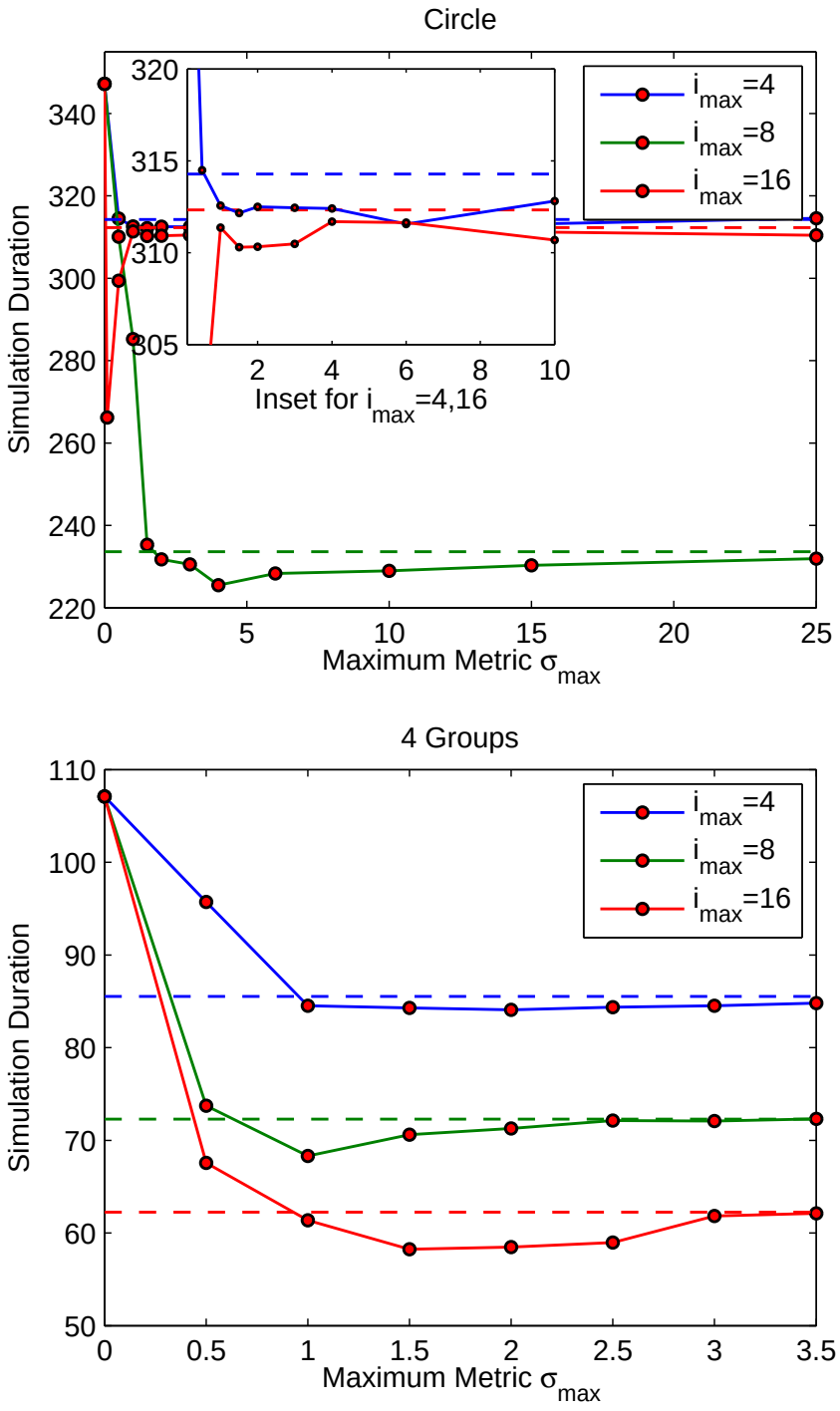
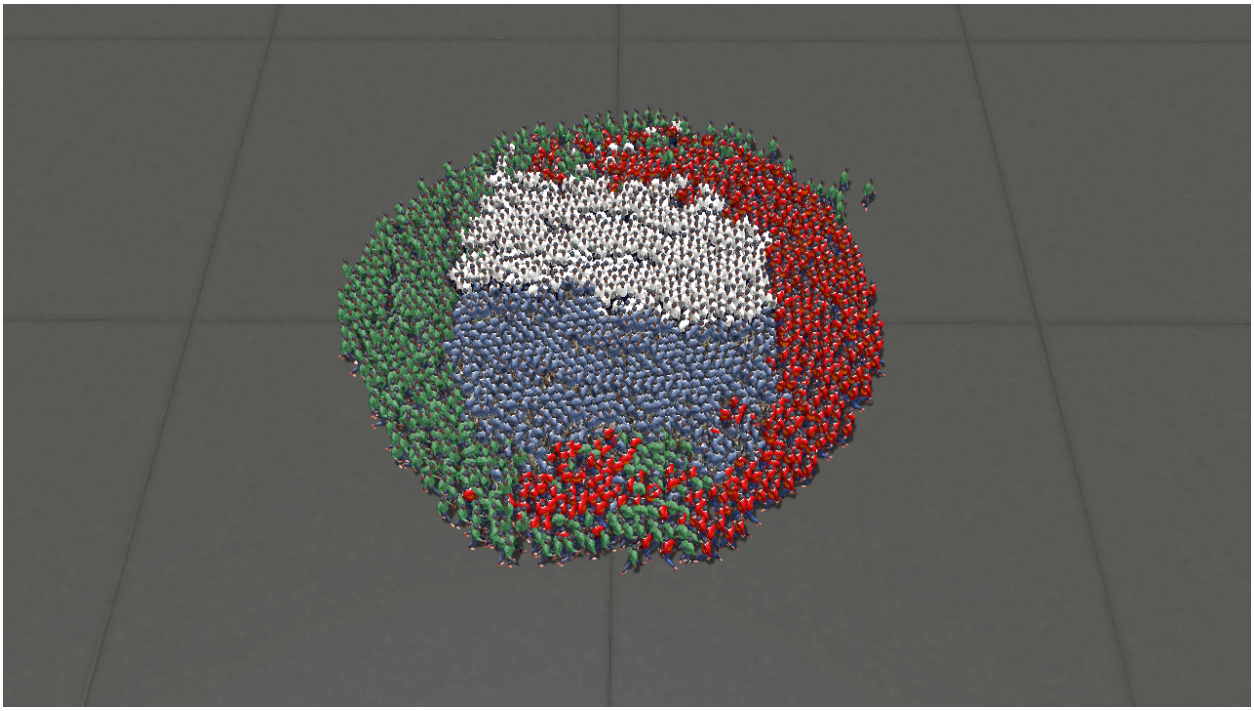
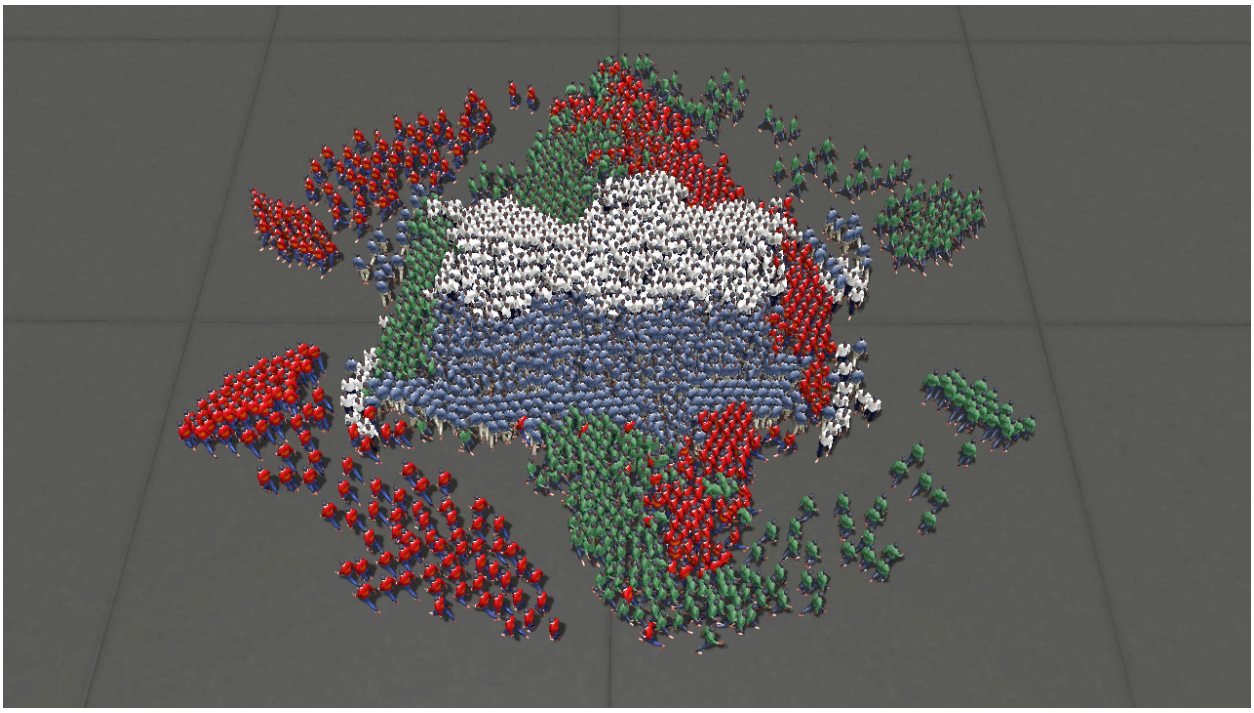


Figure 3.9: Effect of Curtailing Lookahead in the circle (Top) and 4 groups (Bottom) demo. Solid lines show curtailed lookahead, while dotted lines show constant lookahead. Note how curtailing lookahead with a maximum metric value $\sigma_{\max}$ allows agents to reach their goals faster, as demonstrated by the reduced duration. In some cases, curtailing can double the improvement shown by lookahead (circle, $i_{\max} = 16$). Even with optimal choice of $m_{\max}$, I see benefits of 10%.



(a)



(b)

Figure 3.10: 4 groups of agents in circular formation exchange their positions. Notice how lookahead (b) shows red and green agents moving around the built up region in the center and avoid getting stuck as is the case in (a).

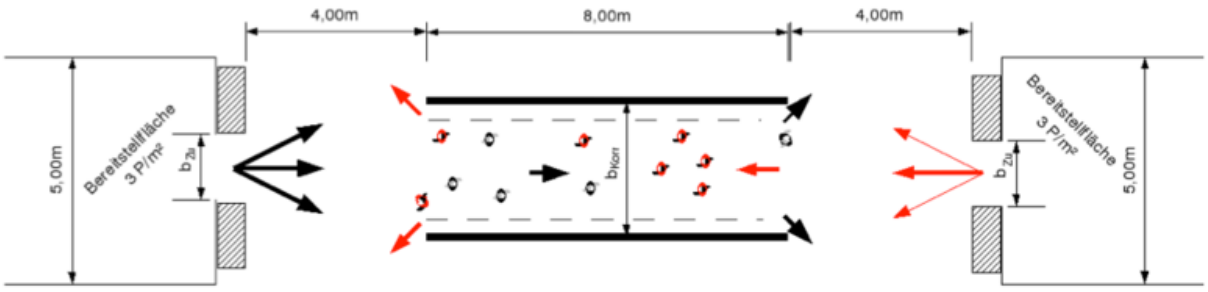


Figure 3.11: Experimental setup for bi-directional crowd flow (Image courtesy [Zhang et al., 2012]).

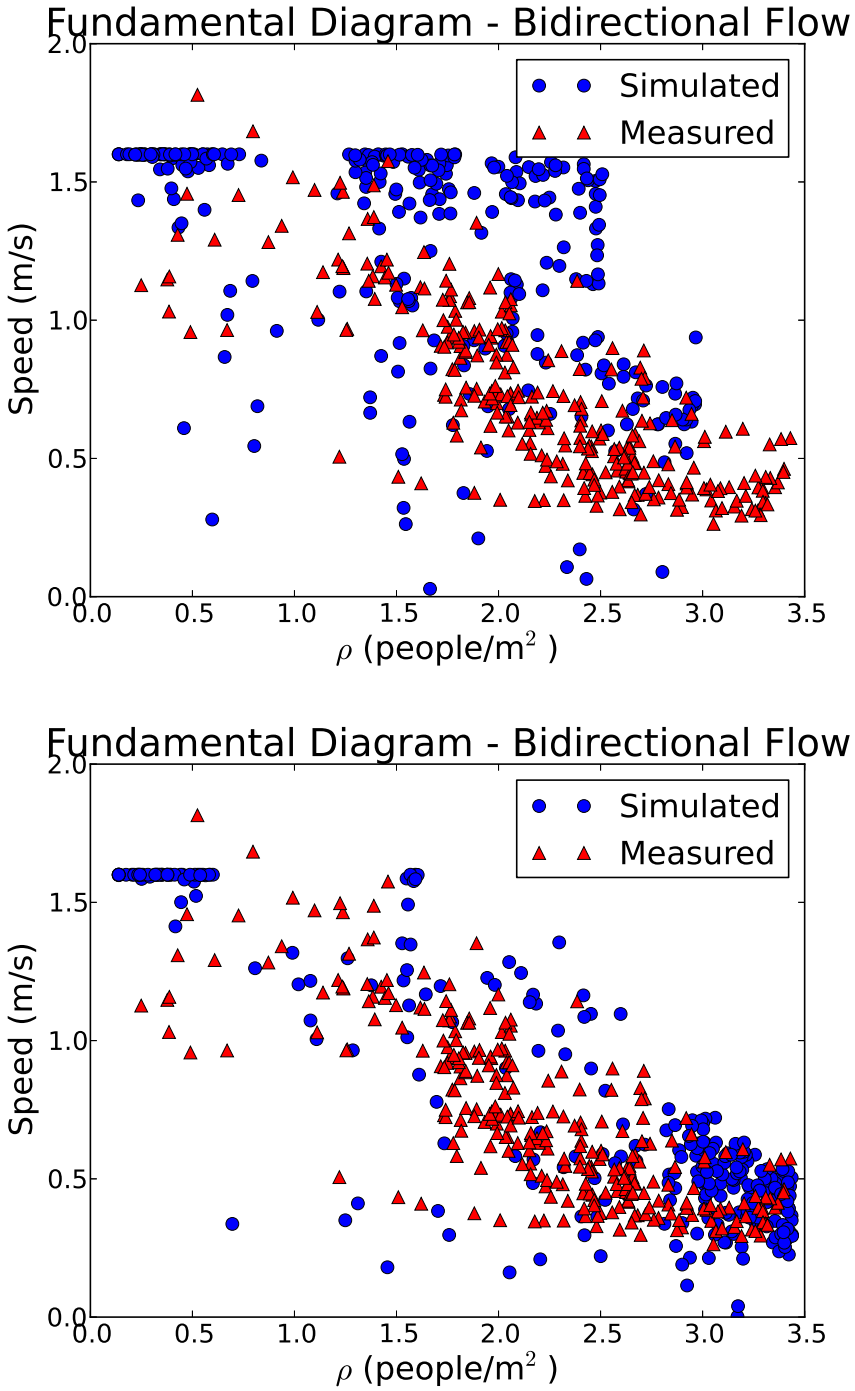


Figure 3.12: Speed v.s. density plots for simulated crowds using (Top) traditional local collision avoidance, and (Bottom) long-range collision avoidance. Note how simulation with lookahead improves correspondence to speeds observed in the real-world data as compared to density, and how collision avoidance with lookahead demonstrates the same downward trend in speed with increasing density.

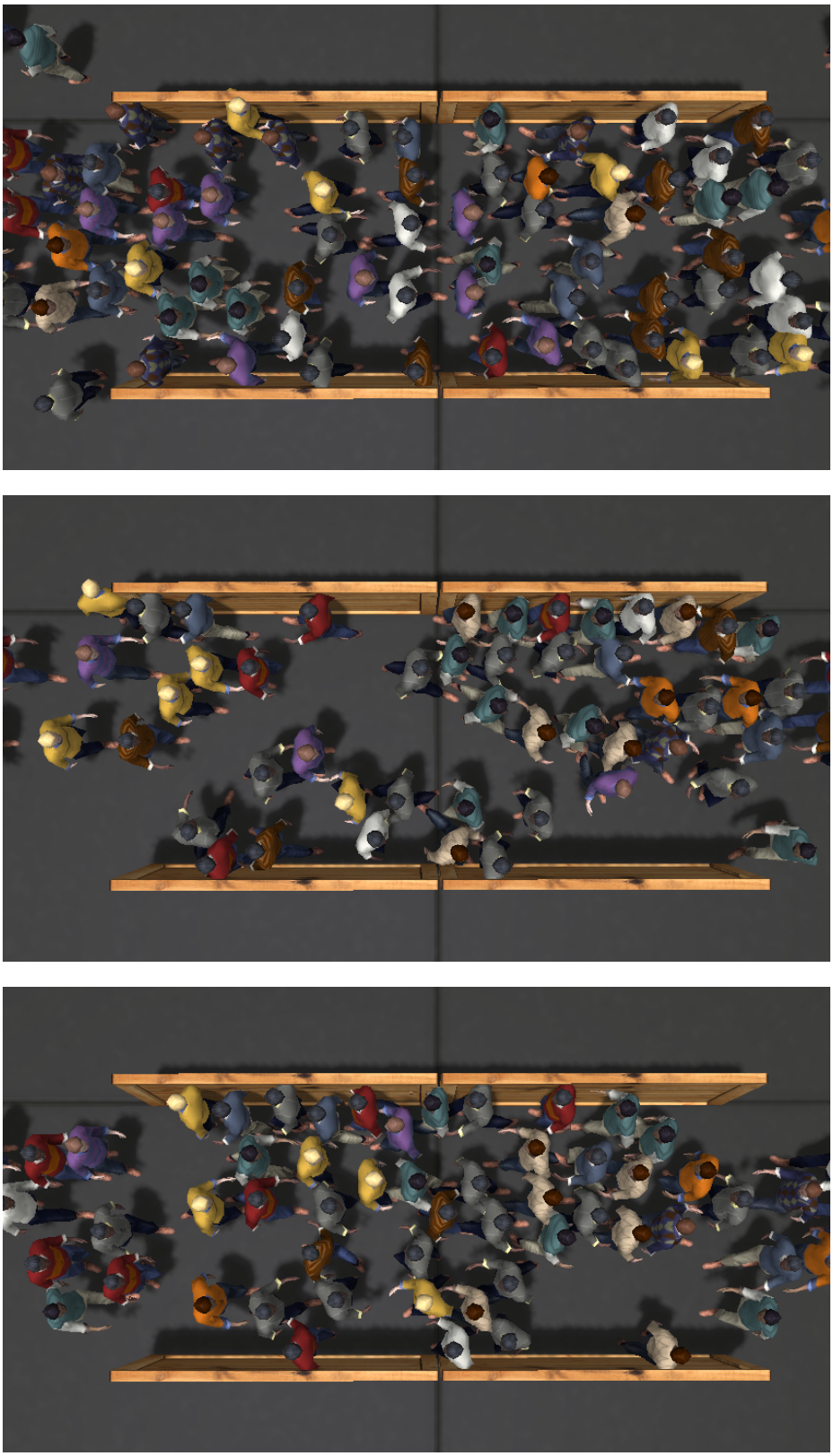


Figure 3.13: Crowd motion using (Top) real-world data, (Middle) Local collision avoidance, (Bottom) Long-range collision avoidance.

Chapter 4: Simulation of Turbulent Behavior in Human Crowds

4.1 Introduction

Crowd disasters, like those at the Hajj in Mecca and the Love Parade disaster in 2010, have shown that current crowd control methods can prove untenable at high densities. As population increases, instances of high-density crowds can only grow more frequent. This poses a significant safety risk that can only be addressed by a deeper understanding of crowd behavior in such conditions. Recent research by Helbing et al. [Helbing et al., 2007, Helbing and Mukerji, 2012] has started to shed light on the unique characteristics of crowd motion in these high-density disasters.

Under normal conditions, crowds of pedestrians typically move smoothly, with slowdowns and fluctuations only near obstacles. When the density and urgency of pedestrians increases, however, the behavior of crowds changes dramatically. First, the crowd begins to exhibit stop-and-go waves: the motion of pedestrians alternates between stopped at high density and moving forward at low density, with these density waves traveling backwards through the crowd. Second, at even higher densities, the crowd shows a chaotic, oscillatory behavior reminiscent of turbulence in fluids, with pedestrians being pushed in all directions, even backwards against their desired direction of motion. The term *crowd turbulence* has been introduced to describe such conditions [Helbing et al., 2007].

Even under such extreme conditions, pedestrian motion shows a coherent structure. Individual motion is replaced by mass motion, and the scale of turbulent oscillations spans several times the pedestrian separation. Crowds also exhibit a stick-slip instability [Helbing et al., 2007] which leads to a rupture when stress in the crowd becomes too large. As a result, the crowd splits up into clusters of different sizes, with strong velocity correlations within each cluster, and distance-dependent correlations between clusters.

Simulation can be an invaluable tool to understand the processes behind this behavior, and to formulate strategies to prevent their occurrence. However, existing crowd simulation techniques have been found to be inadequate for modeling crowd turbulence by [Helbing et al., 2007]. I hypothesize that at high densities, physical forces between pedestrians become relevant as they are brought into close contact. Accordingly, I propose that a model for inter-personal forces is essential for simulating crowd behavior at high-densities. I use a continuum approach where inter-personal forces are represented as stress fields caused by discomfort and friction of the underlying simulated pedestrians.

Numerical simulations of real-world scenarios using my model demonstrate close qualitative and quantitative correspondence to observed data. I believe that models like mine incorporating inter-personal stress can be used to obtain a better understanding of crowd behavior in dense, panicked situations for both theoretical analysis and practical situations.

4.2 Background

Behavior of pedestrians in public spaces has been the focus of much research over the past few decades. This includes insights obtained by observing real-world crowds [Fruin, 1971], as well as observations made in controlled laboratory conditions [Zhang et al., 2011, Zhang et al., 2012] to help improve crowd flow in public spaces. A well-known result of such research is that pedestrian speed decreases with crowd density [Pipes, 1953, Polus et al., 1983, Fruin, 1993, Weidmann, 1993, Seyfried et al., 2005]. This relationship is known as the *fundamental diagram*. Another major focus of research is to understand crowd behavior in order to help reduce the occurence of disasters [Fruin, 1993]. Recent research has started looking into disasters at large scales and high crowd densities. Insights offered by [Helbing et al., 2007] into the crowd disasters at the Hajj, and by [Helbing and Mukerji, 2012] into the Love Parade disaster in particular, form the basis for my proposed model.

For large-scale or high-density crowds, simulations offer the one of very few avenues for

study, as such crowds cannot be recreated safely under controlled conditions. The field of crowd simulation is quite broad and includes modeling of navigation, psychological factors, social interactions, and so on; navigation, however, is the most important factor for reproducing the dynamics of large-scale crowd motion. Accordingly, in this section I review the related work in this area relevant to the problem of simulating crowd turbulence, restricting attention to models that adhere to the fundamental diagram.

Virtual crowds are traditionally modeled as collections of individual virtual agents. In this model, the navigation problem is decomposed into two sub-problems, *global planning* and *local navigation*. Global planning plots a path for each agent to its goal, while only considering static obstacles in the scene. Local navigation then attempts to guide the agent along this path while avoiding collisions with other agents in the scene. As this component is responsible for reacting to the motion of other agents, it determines how well the model can reproduce stop-and-go waves and crowd turbulence.

Local navigation algorithms can be broadly classified into *discrete* or *continuum* algorithms. Discrete algorithms [Guy et al., 2010, Moussaïd et al., 2011, Portz and Seyfried, 2010, Curtis and Manocha, 2012, Reynolds, 1987] solve the navigation problem for each agent individually, considering other agents as separate entities. For efficiency reasons, only a limited set of nearby agents is considered for avoidance. The alternate approach is to consider the crowd as a continuum [Hughes, 2003, Treuille et al., 2006b], with the avoidance problem being recast as navigating around regions of congestion. This is ideally suited for medium to high densities. Narain et al. [Narain et al., 2009] propose a macroscale collision avoidance model for calm dense crowds, though densities at which turbulence is not observed. The lookahead approach described in Chapter 3 is also appropriate for this purpose, as it allows local navigation methods to conform to the fundamental diagram without explicit modeling.

One of the key insights offered by recent studies of crowd disasters [Helbing et al., 2007, Helbing and Mukerji, 2012], has been the similarity of crowds to granular materials be-

yond the level of analogy established in previous work [Hoogendoorn et al., 2007]. Therefore, I draw on ideas from granular flow simulation in developing my model. Granular materials have long been the focus of scientific study, and accurate numerical simulation of their behavior is challenging both for discrete and continuous approaches [Behringer et al., 1999]. Nevertheless, several approximate continuum models have been proposed [Quecedo et al., 2004, Aranson and Tsimring, 2001, Josserand et al., 2004]. These models reproduce much of the qualitative properties of granular materials, and their relative computational efficiency guides my decision to use a continuum formulation for my approach.

Existing work has sought to model crowd turbulence using discrete force-based models. Yu et al. [Yu and Johansson, 2007] propose that existing force-based models like social forces [Helbing and Molnár, 1995] can be used to recreate situations where variance of agent velocity reaches levels similar to those seen in turbulent crowds. However, their results do not demonstrate the emergent clusters of motion that is characteristic of turbulent flows. Similarly, the model proposed by Moussaïd et al. [Moussaïd et al., 2011] demonstrates fluctuations in agent positions similar to turbulent crowds, but does not match any other crowd turbulence metrics like the "pressure" defined by [Helbing et al., 2007], or the clustering of agents demonstrated by turbulent crowds.

I postulate that similar to granular media, an essential component in any model for turbulent crowds is inter-personal friction, as described in Section 4.4. This, combined with a physically-based model for agent acceleration completes my model for crowd turbulence, detailed in Section 4.3. I have validated my model against two observed instances of real-world crowd turbulence, the Hajj, and the Love Parade disaster of 2010 in Section 4.5. I conclude with the limitations of my model and scope for future work in Section 4.6.

4.3 Continuum Model for Crowd Turbulence

4.3.1 Crowd as a Continuum

Real-world crowds consist of individuals, and most exisiting crowd simulation models emulate this using discrete virtual pedestrians or *agents*. Doing so is essential for resolving possible pedestrian motion at low-to-medium densities. Higer density models subsume this discrete model into a continuum navigation model, where discrete agents represent pedestrians, but their motion is modeled using a continuum representation of the crowd. Since this model is primarily intended for high-density crowds, I begin with this continuum representation, similar to the Fluid Implicit Particle (FLIP) approach [Brackbill and Ruppel, 1986].

In the continuum formulation, the crowd state is defined in terms of two properties, *density* ρ (people per m^2) and *current velocity* $\mathbf{v}$. These properties are defined in a continuous 2-dimensional space tending to zero outside the bounds of the crowd. To couple this represention with discrete agents, I define a per-agent state using position $\mathbf{x}_i$ and current velocity $\mathbf{v}_i$ defined in two dimensions.

In particular, the density field ρ can be defined as

$$\rho(x) = \sum_i w(x - x_i), \tag{4.1}$$

where the sum is computed over all agents, and w is a smoothing kernel. In my simulations, I use the piecewise bilinear hat function with a support of 1m. Continuum representations of any agent property, like velocity, can then be computed as a weighted average,

$$\mathbf{v}(x) = \frac{\sum_i w(x - x_i)\mathbf{v}_i}{\rho(x)}. \tag{4.2}$$

The time evolution of the crowd in this FLIP formulation is as follows. The position of each agent changes according to its current velocity. However, as agents are performing active locomotion rather than being passive physical objects, their velocities must be updated

through a computational procedure taking into account both their planning decisions and external forces. This is performed in two steps: first, a global planning step computes a preferred velocity $\mathbf{v}_p$ for each agent to direct it towards its goal. Local navigation then computes a collision-free velocity $\mathbf{v}_{cf}$ accounting for influences from other nearby agents. The collision-free velocity field is then constructed using equation (4.2).

In order to obtain valid behavior, these fields must be defined appropriately at boundaries, particularly near solid obstacles. Continuity of the density field, in particular, is essential to obtain correct behavior and avoid collisions with obstacles. In grid cells which are partially covered by obstacles, I compute a coverage fraction $\phi \in [0, 1]$ – the fraction of the cell covered by obstacles. In such cells Equations (4.1) and (4.2) can be corrected by dividing by the fraction $(1-\phi)$. In cells completely covered by obstacles, density is defined via extrapolation. That is, the density of the cell is chosen to be that of nearest cell not completely covered by obstacles, ensuring that the gradient of density is purely tangential at obstacle boundaries. This choice allows the local collision avoidance model described in Section 4.3.3, and the stress model in Section 4.4 to generate correct behavior around solid obstacles.

4.3.2 Continuum Turbulence Model

My model for crowd turbulence consists of a local navigation algorithm that defines appropriate behavior over the full range of low to high crowd densities. Even though my primary focus is a model for crowd behavior at high densities, a model for low-medium densities is necessary to impose realistic initial and boundary conditions for high-density regions.

I assume the existence of global planner that provides a preferred velocity $\mathbf{v}_p$ for each agent at every time step, defining the optimal direction for the agent to reach its goal. I do not make any assumptions about its temporal continuity or smoothness, thus the preferred velocity may change sharply between time steps.

I begin by choosing an existing local navigation algorithm that satisfies the fundamental

diagram. Given the preferred velocity $\mathbf{v}_p$, the algorithm attempts to compute a "collision-free" velocity $\mathbf{v}_{cf}$, that is, a velocity close to $\mathbf{v}_p$ which will not bring the agent into collision with its neighbors. Adherence to the fundamental diagram implies that as crowd density ρ increases, the collision-free speed of the agent $|\mathbf{v}_{cf}|$ decreases toward zero. Agents tend towards this velocity over a characteristic time τ, via an acceleration

$$\mathbf{a} = \frac{\mathbf{v}_{cf} - \mathbf{v}_t}{\tau}. \tag{4.3}$$

Such a model is appropriate for densities up to those observed in most real-world crowds, i.e. up to a density of 5.5 to 6 people per m^2 [Pipes, 1953, Weidmann, 1993]. At higher densities, collision avoidance may be impossible, as agents may be in near-constant contact with other agents. Thus, existing collision avoidance models cannot be used directly. Therefore, I introduce a high-density model which describes the effects of these contact interactions, and whose contribution is zero below a threshold density.

In keeping with observations of [Helbing et al., 2007], I postulate that crowd behavior at high densities is controlled predominantly by inter-personal stress similar to the stresses observed in granular media. Accordingly, I define the stress σ as a symmetric 2×2 tensor field representing the internal forces in the crowd. This tensor σ can be decomposed into its normal and deviatoric components,

$$\sigma = p\mathbf{I} + \mathbf{s}, \tag{4.4}$$

where the normal component p represents normal or repulsion forces, while the deviatoric stress $\mathbf{s}$ is trace-free and represents frictional forces. I describe how to compute p and $\mathbf{s}$ in section 4.4. The force due to stress is

$$\nabla \cdot \sigma = \nabla p + \nabla \cdot \mathbf{s}. \tag{4.5}$$

Therefore, the combined contribution of stress forces (4.4) and local navigation (4.3) can

then be expressed by modifying the definition of agent acceleration as:

$$\mathbf{a} = \frac{\mathbf{v}_{cf} - \mathbf{v}_t}{\tau} - \frac{1}{\rho}\left(\nabla p + \nabla \cdot \mathbf{s}\right) \tag{4.6}$$

The acceleration implied by this equation may be greater than what can be achieved by pedestrians in the real world. The spontaneous oscillations seen in crowd turbulence can also arise from limits on pedestrians' response to variations in the crowd, namely, their bounded accelerations. Thus, I impose a constraint that the magnitude of the actual agent acceleration must be smaller than a constant $a_{\max}$. Thus, the agent velocity $\mathbf{v}$ changes as

$$\frac{d\mathbf{v}}{dt} = \hat{\mathbf{a}} \min\left(a_{\max}, \|\mathbf{a}\|\right). \tag{4.7}$$

where $\hat{\mathbf{a}}$ is the direction of the acceleration vector. The agent position can then be updated through time integration.

The resulting algorithm is shown in Fig. 4.1 .

- Define the crowd density field using Equation (4.1)
- For each agent a, compute optimal direction $\hat{\mathbf{v}}$ using Equation (4.9)
- Use Equation (4.2) to determine the continuum velocity field
- Compute frictional stress using this velocity field
- Advect agents and update states using Equation (4.7)

Figure 4.1: Continuum Simulation Algorithm for Crowd Turbulence

4.3.3 Local Collision Avoidance

I use the lookahead-based collision avoidance model defined in chapter 3 for computing the collision-free velocity $\mathbf{v}_{cf}$. My choice of this model is guided by the fact that it has been validated against real-world data for low-medium density crowds. In my model, this optimal direction is computed independently for every agent.

To recap, in this model agents move at their maximum natural speed in a direction that maximizes progress towards goal. The agent velocity can thus be expressed as $\mathbf{v}_{cf} = \hat{\mathbf{v}} f(\rho)$, where $f(\rho)$ is the average speed at the given density ρ. I use the same linear form for $f(\rho)$ proposed in chapter 3, with the exception of having the minimum value of $f(\rho)$ to be 0.01m/s instead of zero. Assuming that the agent maintains a constant heading along a chosen direction $\hat{\mathbf{v}}$ for a planning horizon Δt, the displacement after time Δt will be:

$$\mathbf{d}(\Delta t) = f\hat{\mathbf{v}}\Delta t + \frac{1}{2}(f\hat{\mathbf{v}} \cdot \nabla\rho) f'\hat{\mathbf{v}}\Delta t^2 \tag{4.8}$$

where f and f' are evaluated at the density at the current position. The direction $\hat{\mathbf{v}}$ is chosen to maximize progress towards goal $\mathbf{v}_p \cdot \mathbf{d}(\Delta t)$, giving the optimization problem

$$\arg\max_{\hat{\mathbf{v}}} \left(\mathbf{v}_p \cdot \hat{\mathbf{v}} + \frac{f'\Delta t}{2}(\hat{\mathbf{v}} \cdot \nabla\rho)(\mathbf{v}_p \cdot \hat{\mathbf{v}}) \right) \tag{4.9}$$

subject to $\|\hat{\mathbf{v}}\| \leq 1$. Solving this problem using their lookahead approach yields a velocity that satisfies the fundamental diagram.

4.4 Inter-personal Stress

I now complete my model by defining the method to compute inter-personal stress. As noted previously, under low-to-medium density conditions, local collision avoidance computes a path that avoids collisions with other agents. At higher densities, agents may be unable to avoid collisions. In such constrained conditions, I postulate that agent behavior is governed predominantly by the desire to avoid or minimize the likelihood of contact and collisions with other agents and static elements in the scene. This implies that when possible, agents attempt to maximize inter-agent separation, as that would reduce the likelihood of collisions. At a macroscopic scale, this is equivalent to the crowd as a whole trying to attain a state with lower mean density. This is analogous to the role of pressure in fluids and granular materials.

Considering the crowd as a compressible fluid, I define a pressure-like field p as a monotonically increasing function of density ρ. Its effect is to allow density fluctuations to propagate through the crowd, similar to the waves observed in turbulent crowds. In order to avoid confusion with the "pressure" metric [Helbing et al., 2007] used later in the chapter, I call this functional form *discomfort*. The behavior described previously can thus be expressed as a desire by the crowd to lower discomfort caused by high densities and cramped spaces by moving along its gradient. By ensuring that the gradient of density is tangential to obstacles in Section 4.3.1, I ensure that pressure gradient is tangential to obstacles as well, ensuring that the component of agent velocity normal to obstacles is zero.

To define a functional form for discomfort, I make certain observations:

1. Discomfort must be a constant below a critical density of 3.5 people per m^2, i.e. its gradient is zero. Pedestrians at such densities can move freely towards their goal, and thus I expect virtual agents to do the same.

2. At higher densities, discomfort rises steeply. This is expected as pedestrians still have available space to navigate and reach lower density regions.

3. Beyond a certain yield density of 5.5 to 6 people per m^2 the gradient of discomfort reduces and plateaus. This is a reflection of the fact that at such densities, pedestrians would lose the available space to navigate and thus have a more limited set of responses.

Note that a number of functions can follow these considerations and yield similar results. I choose a form similar to one proposed by [Hughes, 2002], where a similar concept of discomfort is proposed:

$$g(\rho) = \begin{cases} 0, & \text{if } \rho < \rho_{crit} \\ c_1 \frac{\rho}{\sqrt{\rho_{yield} - \rho + \epsilon}} - c_2, & \rho_{crit} \leq \rho < \rho_{yield} \\ c_3 \rho - c_2, & \rho \geq \rho_{yield} \end{cases} \tag{4.10}$$

where ϵ is a small positive constant, c_1 is a parameter, and c_2 and c_3 are chosen to ensure

continuity of the function at ρ_{crit} and ρ_{yield}. The function grows steeply to the yield point ρ_{yield} after which it grows linearly. Specific values used for these constants as used in my simulations are defined in Section 4.5.

The other essential component of inter-personal stress is friction. Without modeling friction, stick-slip instabilities observed in turbulent crowds cannot be obtained. Friction modeled here represents the combination of two possible responses. The first is the friction between agents in contact, which in turbulent crowds can cause physical damage and even rip clothes off of people [Helbing et al., 2007]. The second follows from the original intent of pedestrians to avoid collisions. At very high densities, relative motion between agents is likely to cause contact. In such situations, maintaining the relative configuration amongst agent neighbors can be a viable strategy to avoid collisions. At a macroscopic scale, this corresponds to clusters of agents moving coherently.

Though the use of friction as part of a collision avoidance algorithm has been suggested in different contexts, these approaches have modeled only one kind of friction [Helbing et al.,]. In order to obtain stick-slip instabilities, it is essential to model both static and kinetic friction, along with an appropriate yield condition that controls the transition between the two. Static friction prevents relative motion between two bodies in contact, while kinetic friction acts to oppose relative velocity between two bodies already having relative motion. A number of models exist in the domain of granular materials to model such phenomena. I choose a computationally efficient continuum model [Narain et al., 2010], which I find is sufficient to replicate stick-slip instabilities.

Based on the maximum dissipation principle, this formulation computes the frictional stress which minimizes the kinetic energy of the crowd flow, as defined using velocity from Equation (4.7). However, the frictional stress is constrained by a *yield criterion*, which relates the magnitude of frictional stress to the amount of normal stress, which in my case

is discomfort. In keeping with [Narain et al., 2010], I use the Drucker-Prager yield criterion:

$$\|\mathbf{s}\|_F \leq \sqrt{3}\alpha g(\rho) \tag{4.11}$$

where $\|\mathbf{s}\|_F = \sqrt{\sum s_{ij}^2}$ is the Frobenius norm, and α is the coefficient of friction.

The kinetic energy after a time step Δt can be approximated [Narain et al., 2010] as

$$E = \frac{1}{2\rho_{yield}} \int \|\rho\left(\mathbf{v}_t - \nabla p\right) + \Delta t \nabla \cdot \mathbf{s}\|^2 dV, \tag{4.12}$$

which is a quadratic functional of the frictional stress field $\mathbf{s}$. The value of $\mathbf{s}$ is determined by minimizing E subject to the yield criterion (4.11). Upon discretizing $\mathbf{s}$ on the simulation grid and approximating the yield criterion by an intersection of half-spaces, I obtain a constrained quadratic programming problem, which can be solved using standard methods. I obtain efficient convergence using the solver proposed by [Dostál and Schöberl, 2005].

4.5 Validation

I ran simulations using my model to recreate the behavior observed in two crowd disasters. For all simulations, the critical density ρ_{crit} was 3.5 people per m^2, yield density $\rho_{yield} = 5.5$ people per m^2, constants $c_1 = 13.33$, $c_2 = 1.5$, with $\epsilon = 0.01$. I use a friction coefficient $\alpha = 0$ at densities less than ρ_{crit} , $\alpha = 0.75$ at densities greater than ρ_{yield}, with linearly interpolated values for $\rho \in [\rho_{crit}, \rho_{yield}]$. I use a second-order midpoint method for time integration.

Animated visualizations of the simulations generated by my crowd turbulence model in the following two scenarios can be found online [1]

[1] http://gamma.cs.unc.edu/CrowdTurbulence

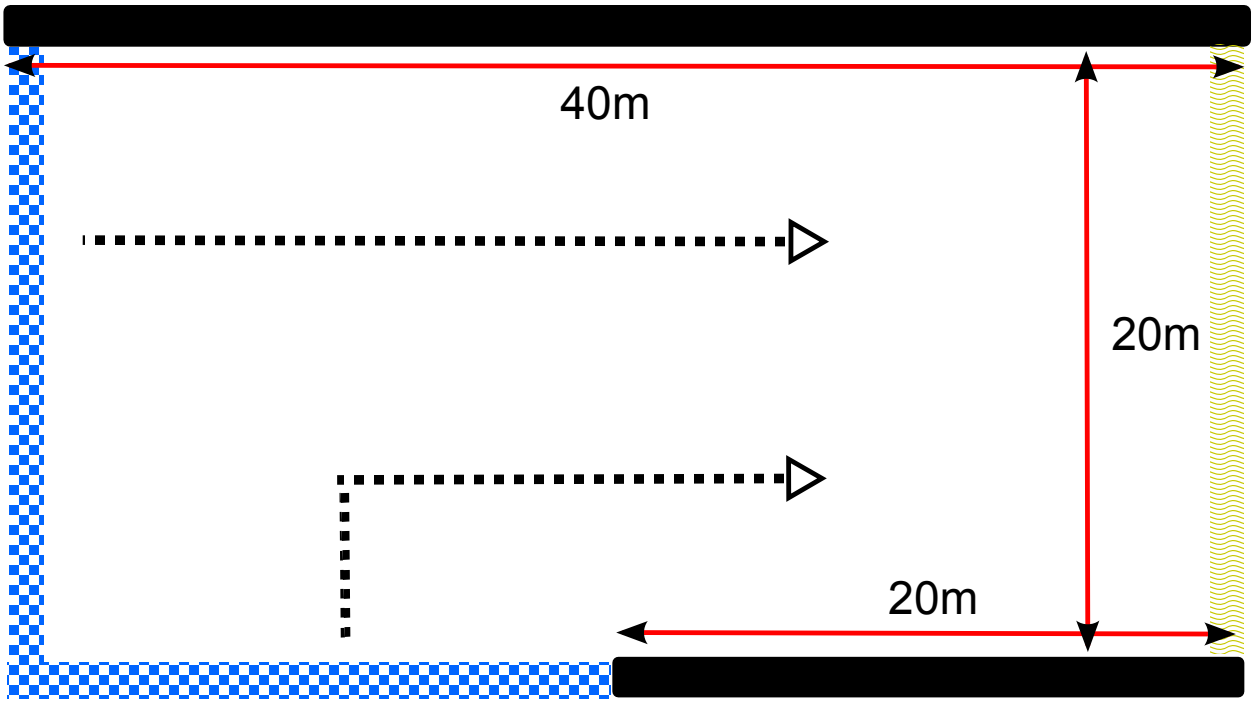


Figure 4.2: Scene setup for Hajj simulations. The scene consists of two merging sets of agents emerging from the inlets shown in the blue checkerboard pattern, following the path highlighted by dotted arrows and exiting via the wavy yellow outlet on the right. Corridor dimensions are shown by red arrows.

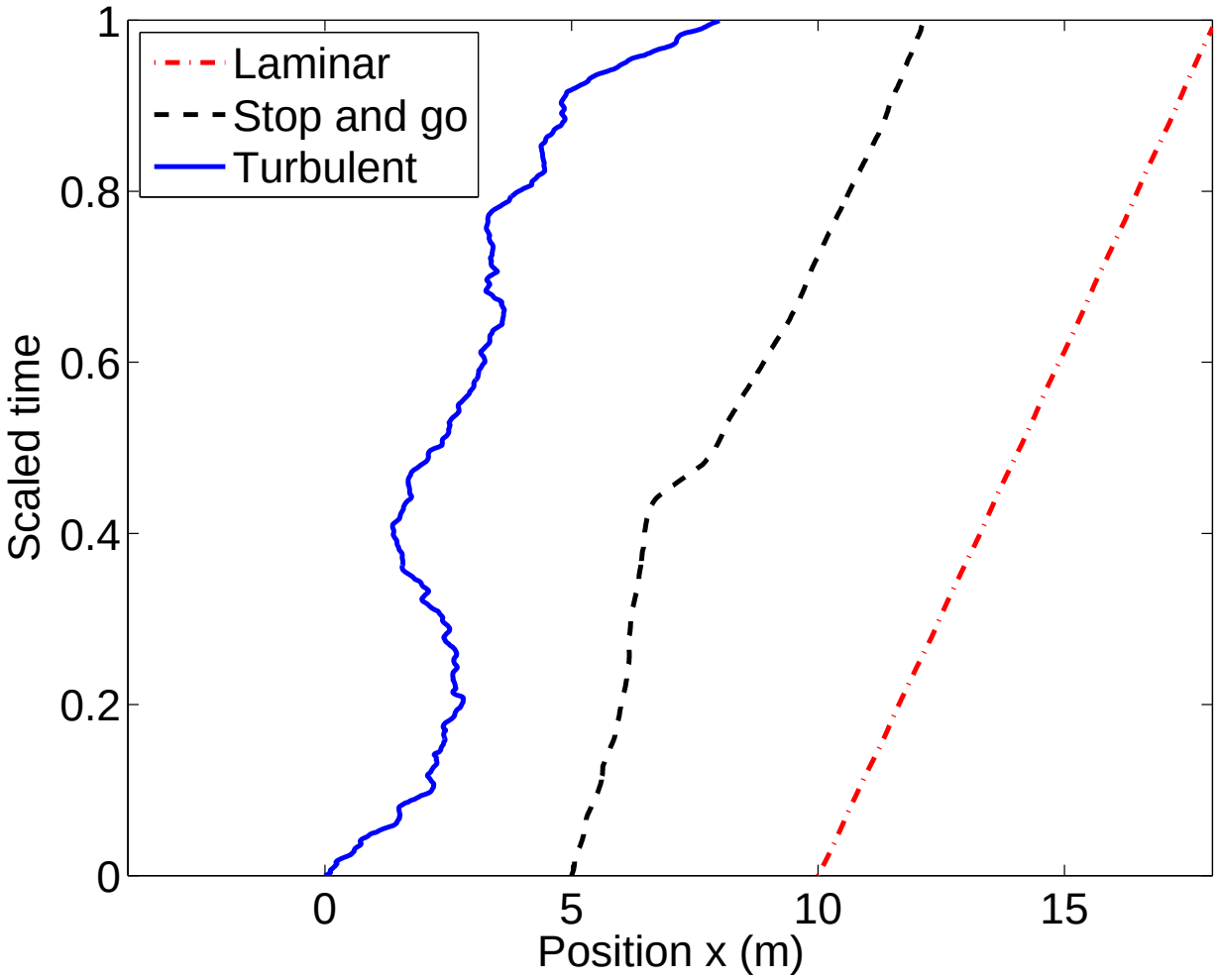


Figure 4.3: Representative trajectories taken by agents in the Hajj simulation to travel a distance of 8m in laminar (red, right), stop-and-go (black, middle) and turbulent flow (blue, left), with time rescaled to unity.

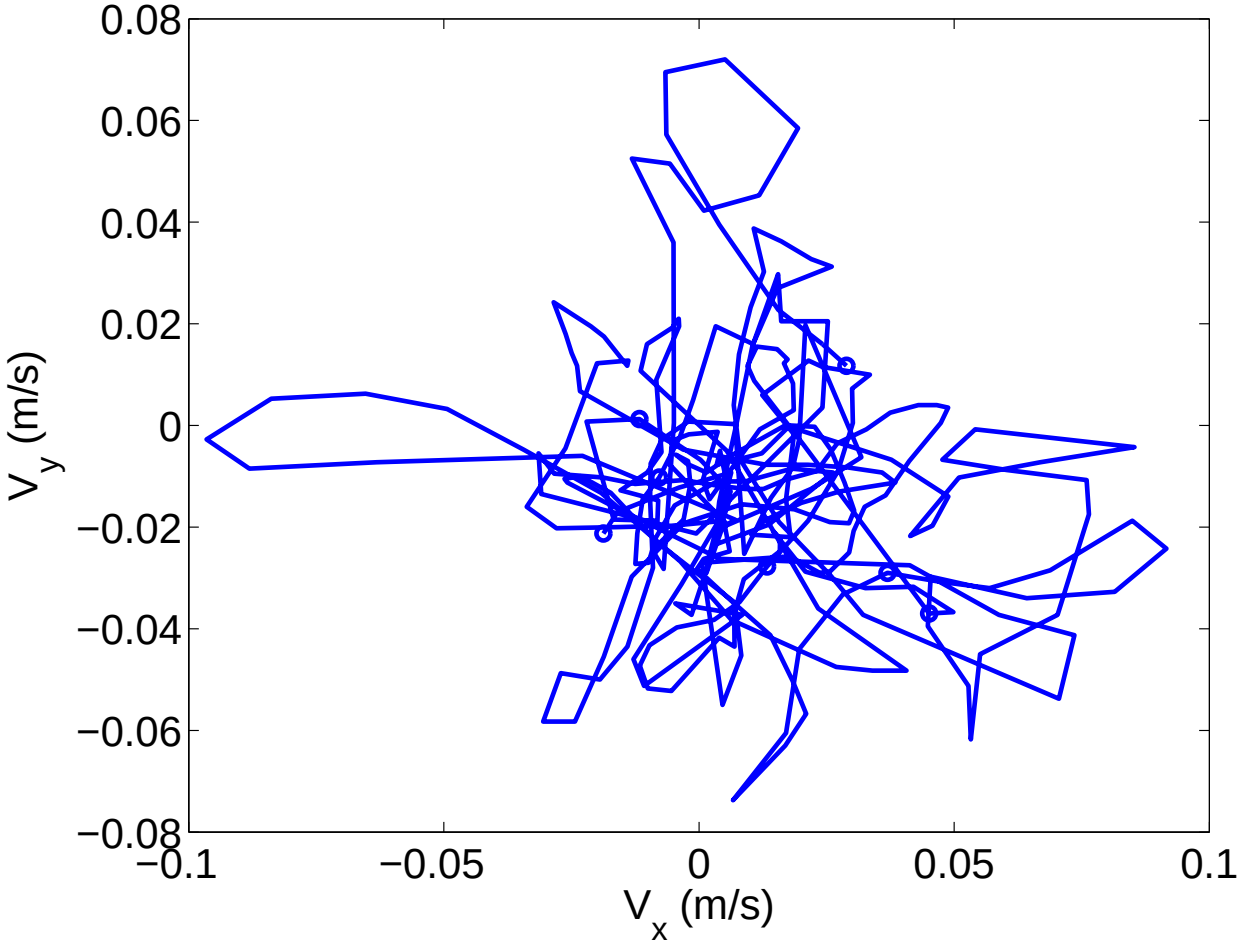
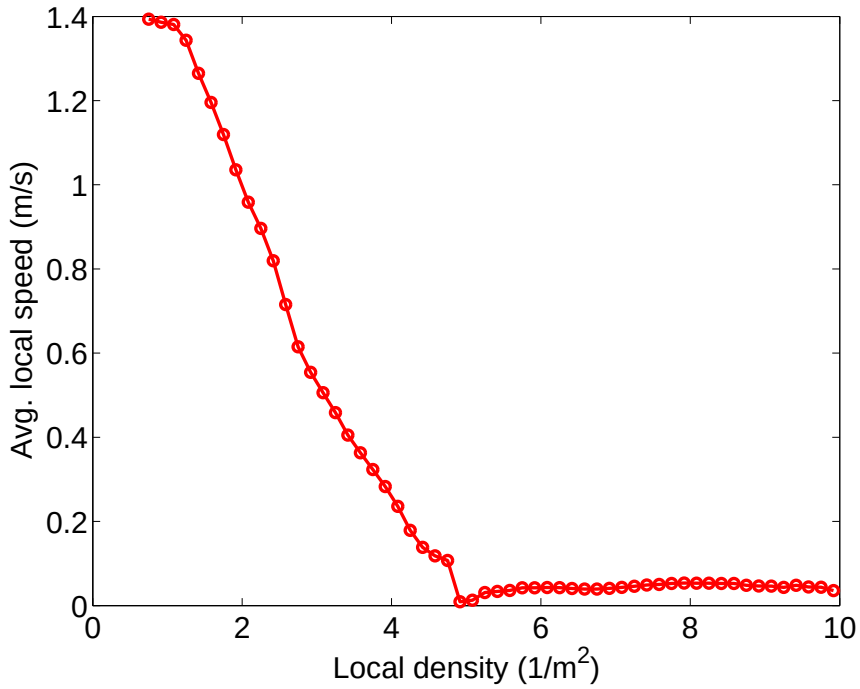
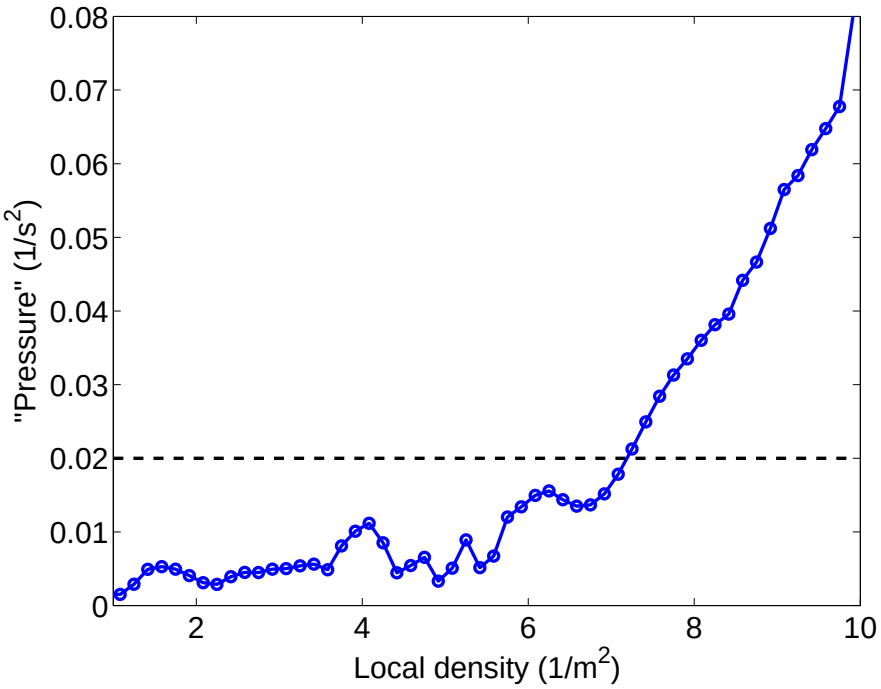


Figure 4.4: Temporal evolution of velocity components v_x and v_y. Under laminar flow v_y will remain close to zero. However, under turbulent flow, such as in this case, I observe motion that is orthogonal and even against the desired direction of motion (along the $+x$ direction).

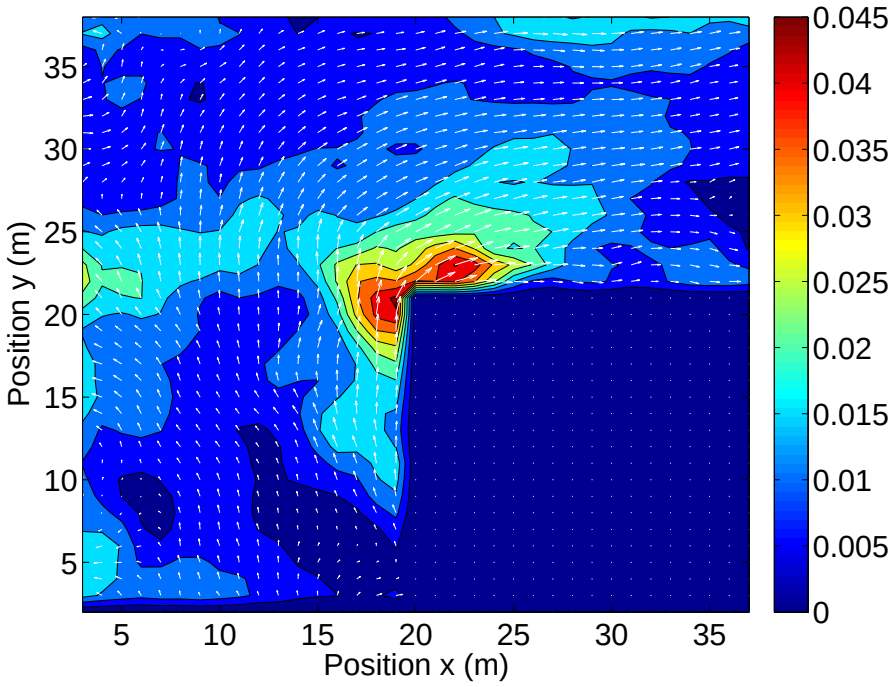


(a)

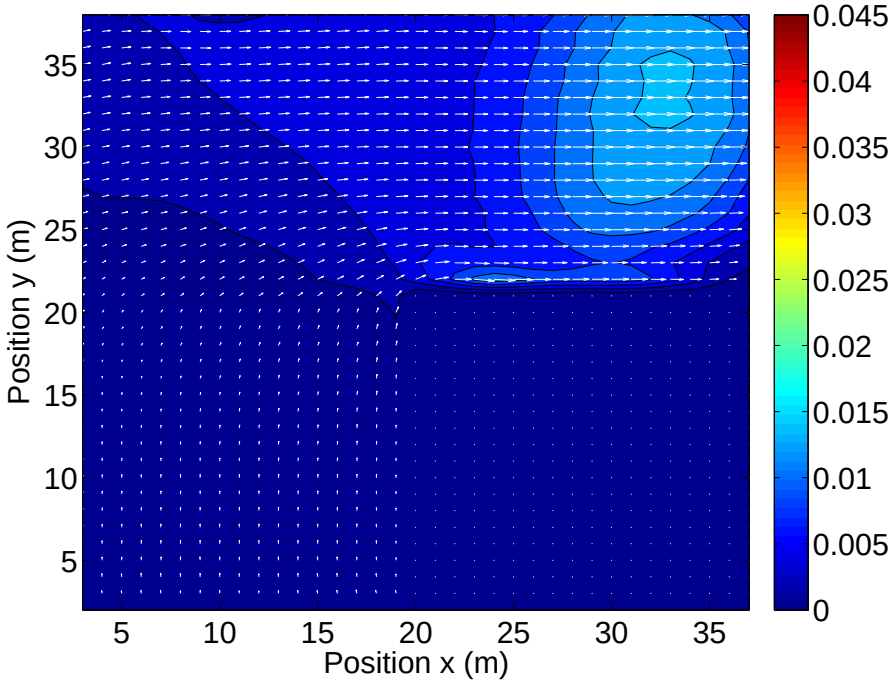


(b)

Figure 4.5: (a) Plot of average speed vs. density. Note that even though average speeds may be low at high densities, high variance implies that agent speeds may be significantly higher than the mean (b) Plot of crowd "pressure" $P(\mathbf{x}) = \rho(\mathbf{x})\,\mathrm{Var}_{\mathbf{x}}(\mathbf{v})$ as defined by [Helbing et al., 2007]. I observe values greater than 0.02 at local densities of 7 people per m^2 and higher, which are indicative of crowd turbulence.



(a)



(b)

Figure 4.6: Plot of crowd "pressure" overlaid with mean flow velocities for a time period of 3 seconds for (a) stop and go flow, and (b) turbulent flow. Note formation of irregular clusters as denoted by isocontour lines. Color bars show range of "pressures" observed; note that values greater than 0.02 do not arise in the occurrence of stop-and-go waves.

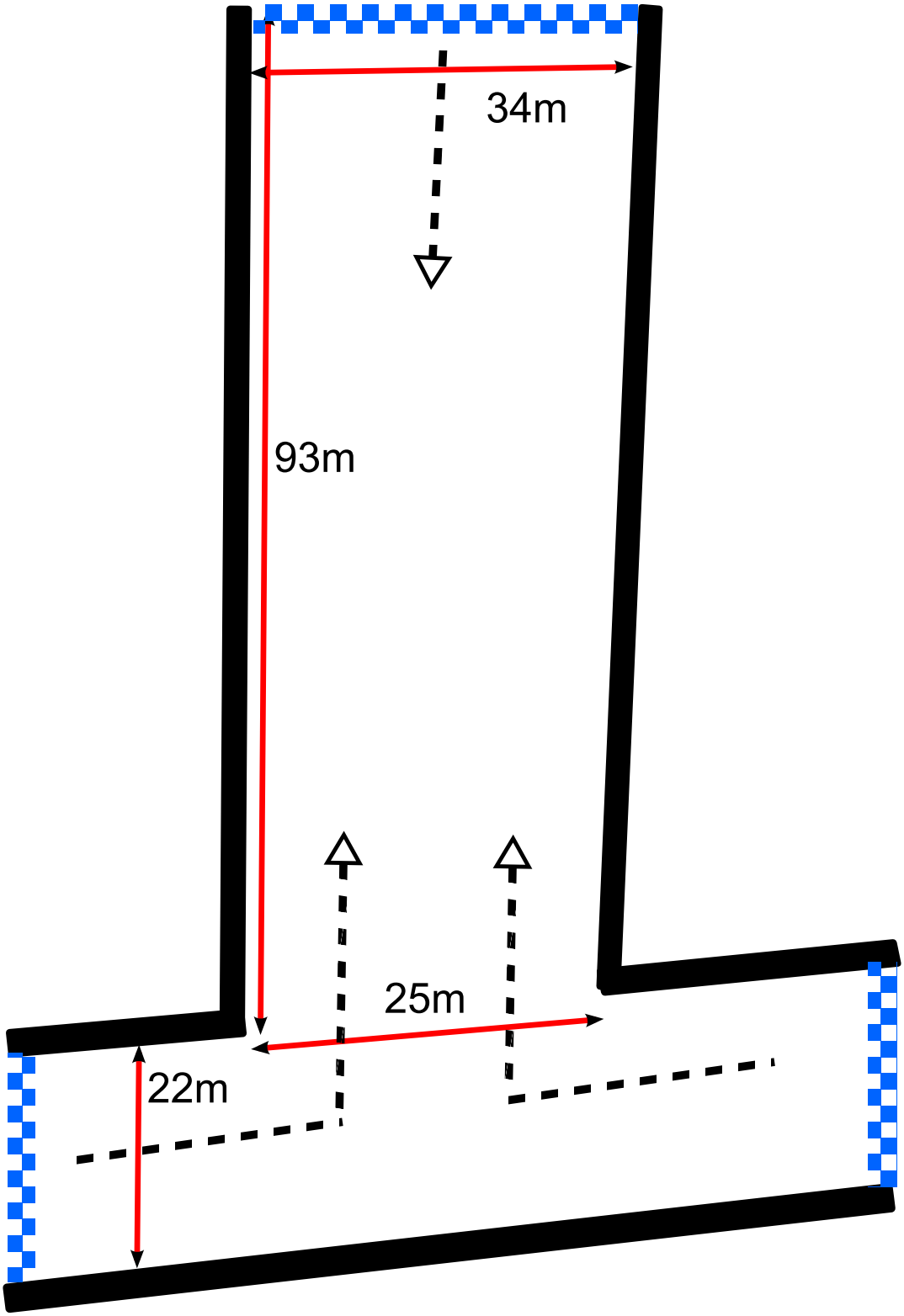


Figure 4.7: Scene setup for Love Parade simulations. The scene consists of three merging sets of agents, emerging from the inlets shown in the blue checkerboard pattern. Agents from the bottom two inlets proceed towards the third inlet at the top and vice versa, following the paths highlighted by the dotted arrows. Corridor dimensions are shown by red arrows.

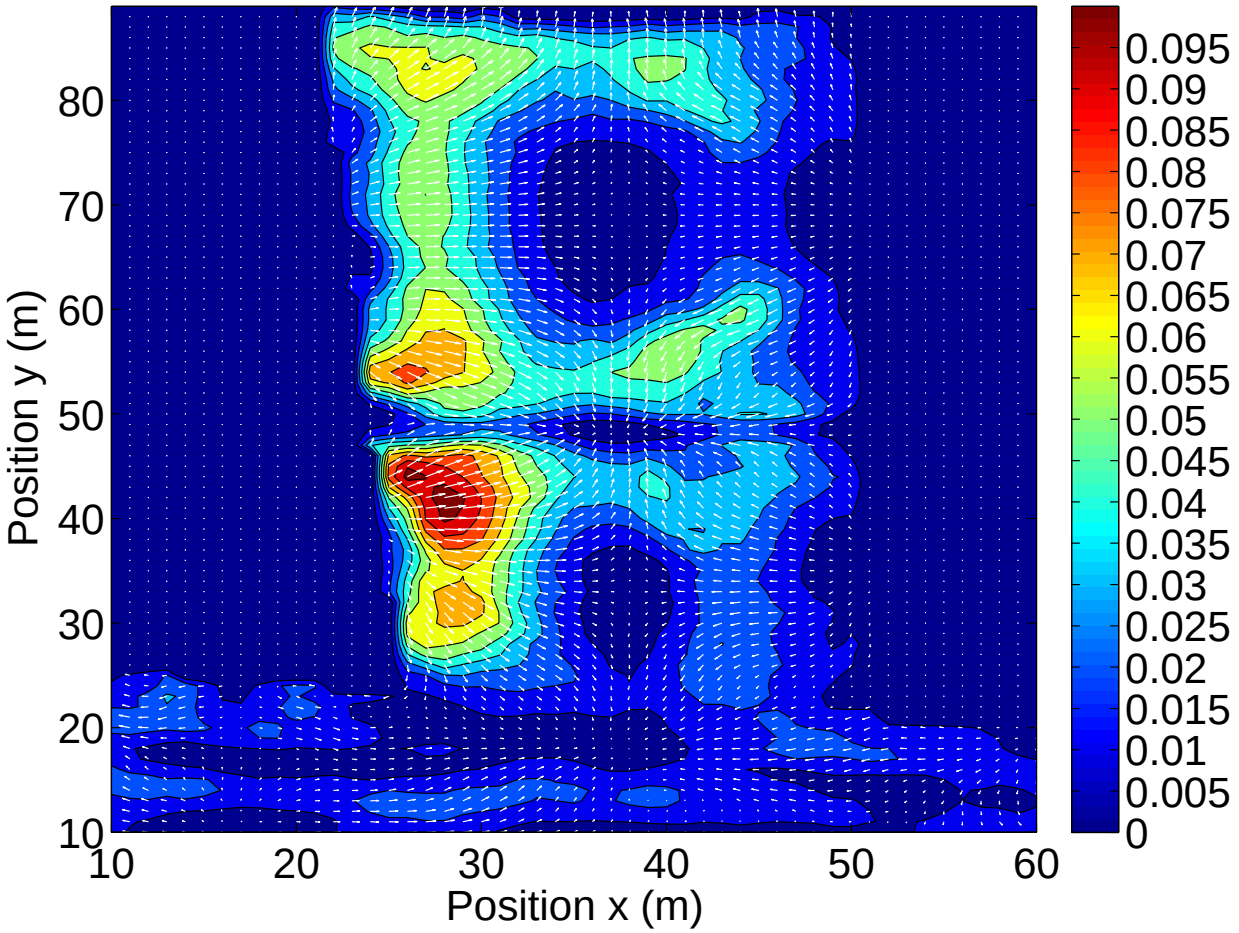
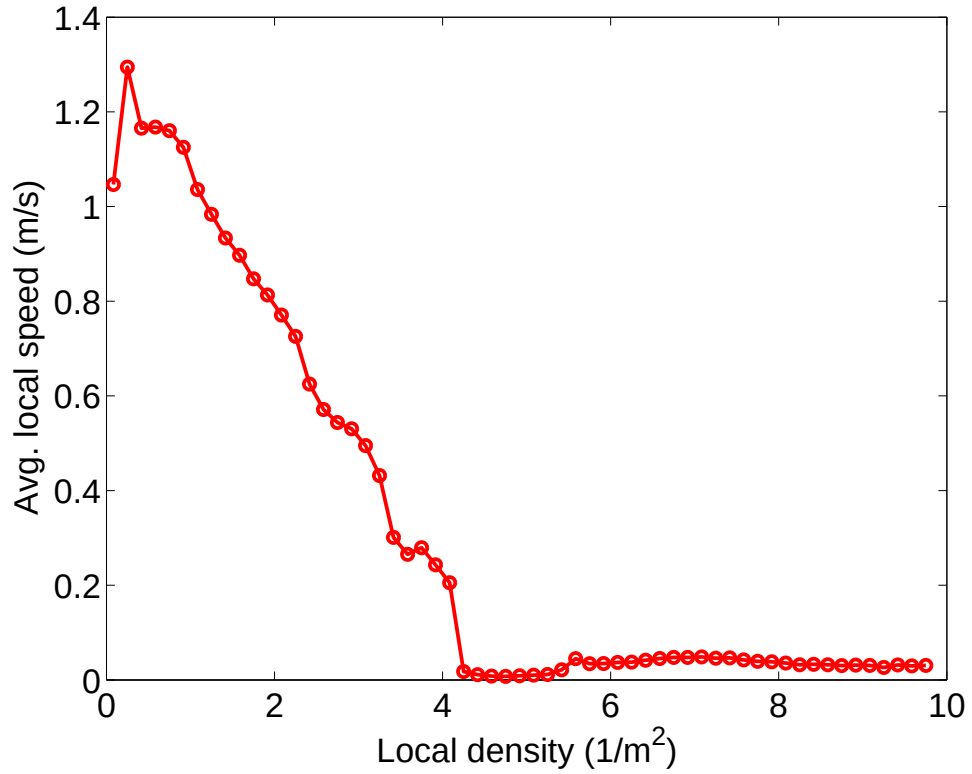
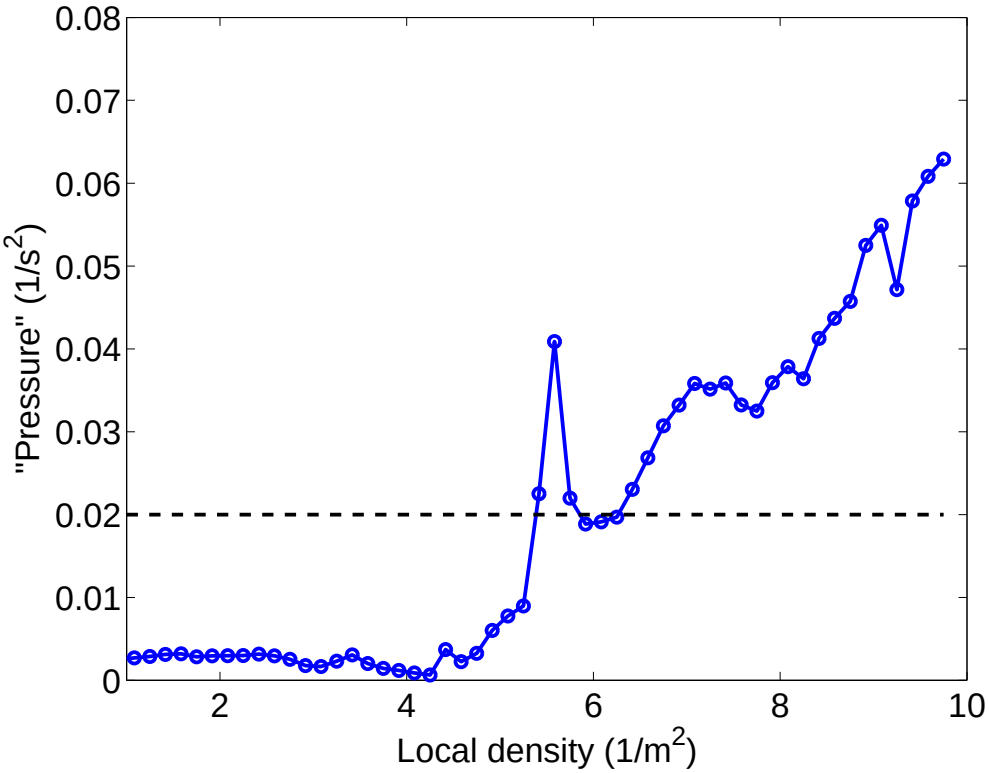


Figure 4.8: Plot of crowd "pressure" overlaid with mean flow velocities for a time period of 3 seconds for turbulent flow. Note formation of irregular clusters as denoted by isocontour lines. Color bars show range of "pressures" observed.



(a)



(b)

Figure 4.9: (a) Plot of average speed v.s. density. (b) Plot of crowd "pressure". I observe values greater than 0.02 at local densities of 5.5 people per m^2 and higher, which are indicative of crowd turbulence.

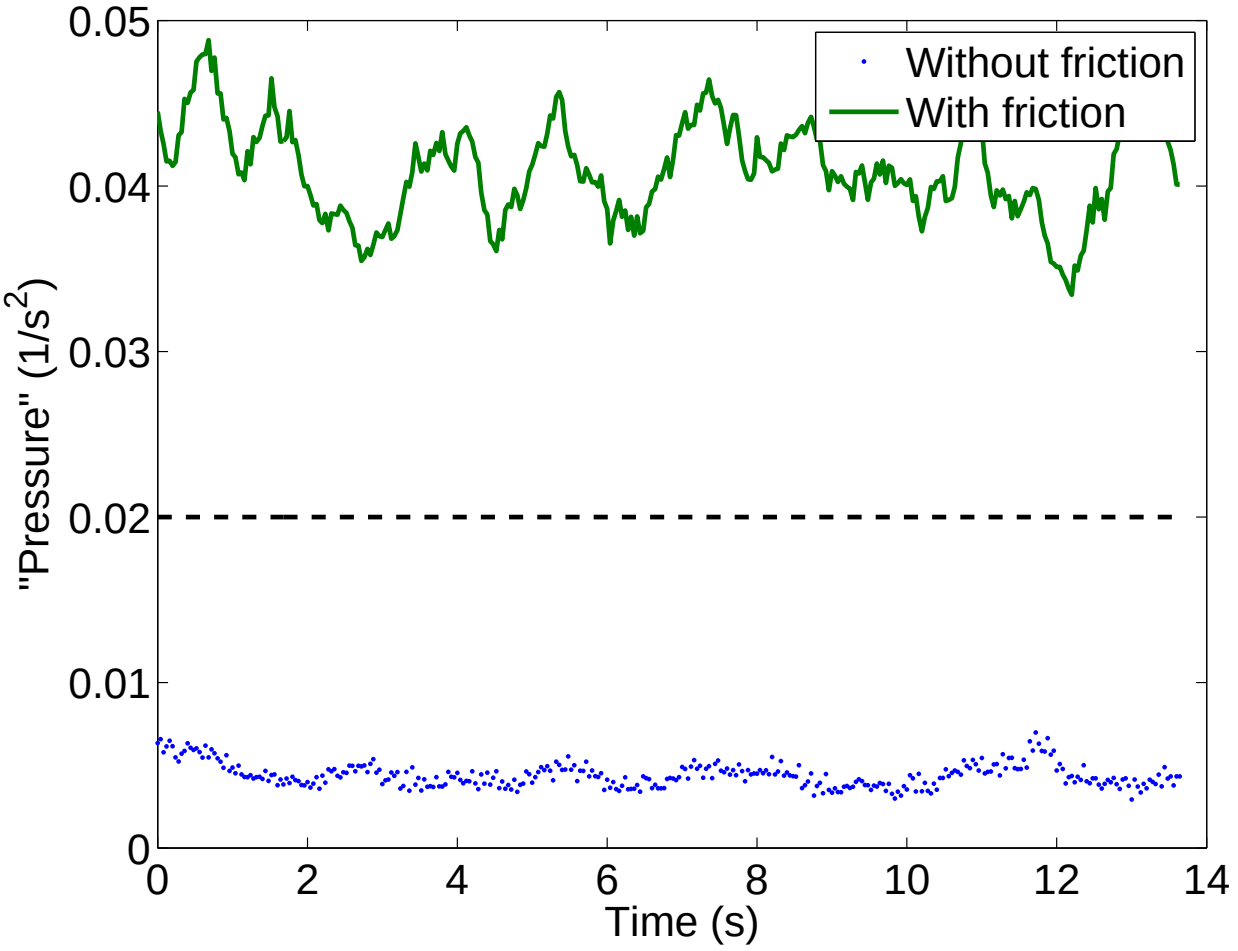


Figure 4.10: Plot of crowd "pressure" $P(t) = \langle \rho(t) \operatorname{Var}_t(\mathbf{v}) \rangle_{\mathbf{x}}$ in the Hajj scenario with both discomfort and friction (top), and with discomfort alone (bottom). $\langle \cdot \rangle_{\mathbf{x}}$ denotes the mean computed over the entire scene. Note how discomfort alone is unable to recreate turbulent flow conditions.

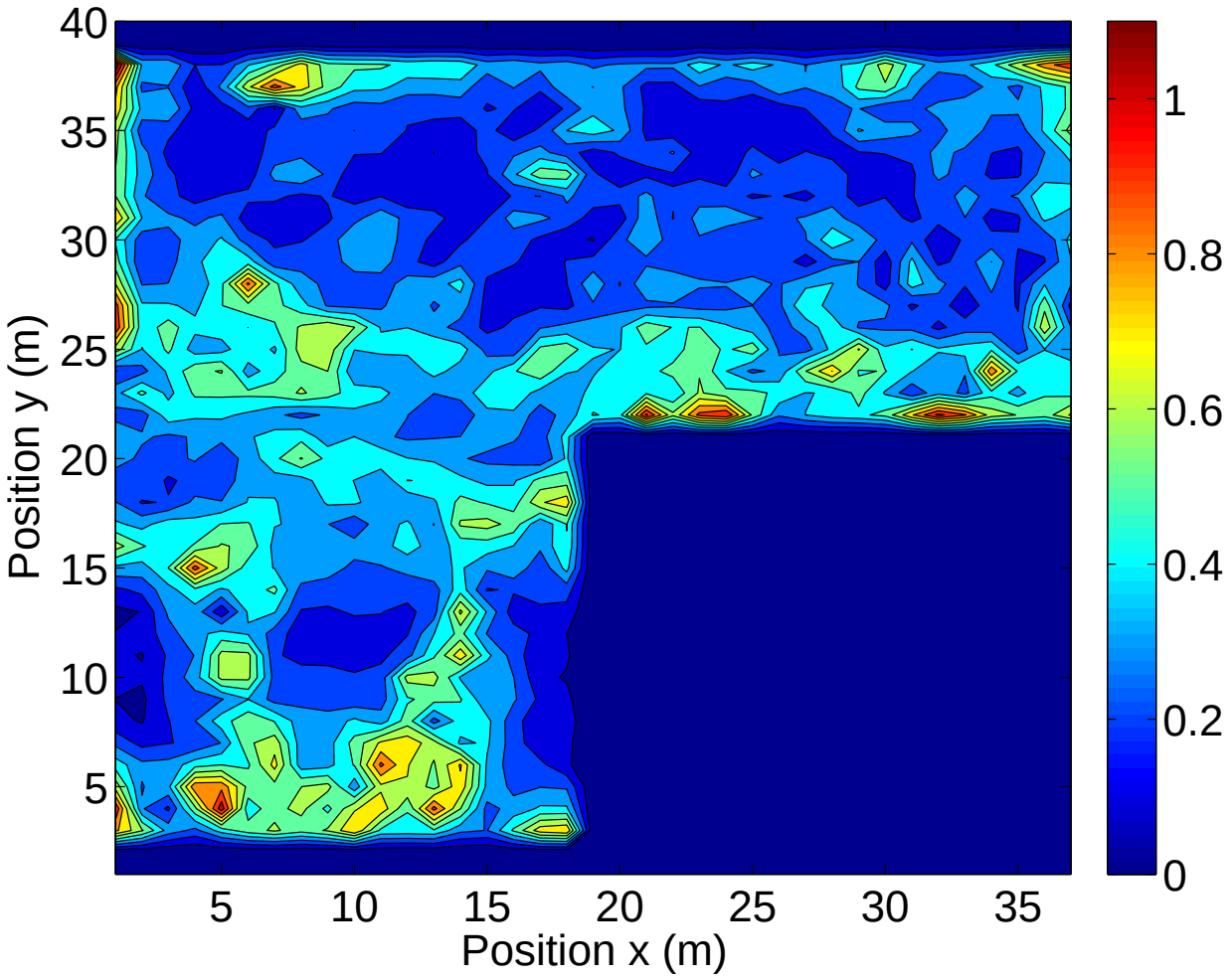


Figure 4.11: Plot of the magnitude of the strain rate tensor $\|\nabla v + (\nabla v)^T\|_F$ at $t = 3s$ for turbulence flow in Hajj scenario. Note cluster formation as shown by isocontours.

4.5.1 Hajj

The Hajj is an Islamic pilgrimage to the city of Mecca, where millions of people undertake every year. Over the past two decades, it has been the location of multiple crowd disasters resulting in the loss of over a thousand lives. As a result, it has also been the focus of research in crowd analysis and simulation. Simulations have been performed on a representative scene proposed by [Helbing and Mukerji, 2012, Yu and Johansson, 2007]. The scene setup is shown in Fig. 4.2 .

I ran simulations for a wide variety of average densities, ranging from 3.5 people per m^2 (4400 agents) to 6 people per m^2 (6800 agents). I observe the emergence of stop-and-go waves even at densities at lower end of my range, while at the higher end of the spectrum I observe the emergence of crowd turbulence. As seen in Fig. 4.5a , the variation of speed vs. density follows a similar decreasing trend as observed for other models and real-world data. At densities below 5 people per m^2, behavior is primarily dictated by the existing local navigation model from chapter 3. Since my model utilizes a linear representation of the fundamental diagram, slope of the curve at these densities follows a linear trend. At densities higher than 5 people per m^2, my model exhibits a close correspondence with the speed trend noted by [Helbing et al., 2007], as well as similar magnitudes.

I use the"pressure" metric P defined by Helbing et al. [Helbing et al., 2007] as an indicator of crowd turbulence:

$$P(\mathbf{x}, t) = \rho(\mathbf{x}, t)\, \mathrm{Var}_{\mathbf{x},t}(\mathbf{v}) \tag{4.13}$$

where $\mathrm{Var}_{\mathbf{x},t}(\mathbf{v})$ is the variance of velocity. In practice, I only compute this variance with respect to either space or time. Previous work shows that real crowds become turbulent at values of P greater than 0.02. As shown in Fig. 4.5b , my simulation replicates turbulent crowd conditions at densities greater than 7 people per m^2.

Fig. 4.3 shows representative trajectories for agents moving in the Hajj scene. Under laminar flow conditions, agents move unhindered along their desired direction of motion,

forward along the x-axis. At the onset of stop-and-go flow, I observe that agents sometimes come to a standstill, as denoted by the trajectory becoming vertical for a short period of time. However, at the onset of turbulent flow, I see that agents can even be pushed backwards against their will. Fig. 4.4 shows that in such cases, agents may move in nearly any direction, including directions orthogonal and opposite to their intended direction of motion, as shown by the variation of the velocity components v_x and v_y.

Fig. 4.6b and Fig. 4.6a demonstrate the cohesive flow that arises at the onset of stop-and-go waves and turbulence respectively. At the onset of stop-and-go waves, a concentration of agents develops behind and in front of the rarefaction wave. As these disturbances intensify into crowd turbulence, I see creation of agent clusters as illustrated by irregular isocontours in Fig. 4.6a . The stick-slip instability manifests at the boundaries of these clusters. As demonstrated by these plots, my model is able to replicate known observations of crowd turbulence at the Hajj.

4.5.2 Love Parade 2010

The Love Parade was an annual electronic dance music festival held in Germany. At the event in 2010, held at a railway yard in Duisburg, Germany, a crowd control failure led to 21 casualties from suffocation, with more than 510 people injured. Though the venue was intended for 250,000 people, up to 500,000 people reportedly attended, with about 3200 police present for crowd control. Due to miscommunication between the various police groups, two inlets were opened allowing a large influx of pedestrians to enter a corridor blocked by an outflux of pedestrians wanting to leave the concert. The resulting bottlenecked situation caused turbulent conditions to develop, leading to multiple casualties.

Based on known data from the venue, I set up a representative scene to model a portion of the railway yard where the mishap occurred. This scene setup is shown in Fig. 4.7 . Recently, the disaster was analyzed in [Helbing and Mukerji, 2012], and an official documentary [2]

[2] "Official Documentary of the Love Parade 2010 Disaster", http://www.youtube.com/watch?v=

reinforced the view that crowd turbulence was responsible for the casualties.

I note that in my simulations, I observe turbulence at densities of 5.5 people per m^2 and higher, as seen in Fig. 4.9b , and observe turbulent oscillations similar to those seen in the official documentary (5:42–5:52). As seen in Fig. 4.8 , most intense turbulence is observed in the region where incoming and outgoing agents meet. In addition, I see formulation of clusters with stick-slip instabilities similar to those observed in the Hajj scenario as demonstrated by variation in "pressure" throughout the crowd.

4.5.3 Importance of Friction in Simulating Crowd Turbulence

In order to validate the hypothesis that inter-personal friction is essential for simulating crowd turbulence, I simulated the Hajj scenario with discomfort alone. As is evident from the "pressure" plot in Fig. 4.10 , without the added contribution of friction, turbulence is not observed.

As noted earlier, cluster formation is one of the hallmarks of crowd turbulence. A plot of the magnitude of the strain rate tensor $\|\nabla v + (\nabla v)^T\|_F$ in Fig. 4.11 shows isocontours delineating variation in velocity. These variations indicate shear regions where agents slip against each other, thereby marking cluster boundaries. Such a transition cannot occur without friction, underlining its importance in a crowd turbulence model.

4.6 Summary

In this chapter, I have postulated that modeling inter-personal forces is necessary for simulating crowd turbulence. To validate this hypothesis I propose a novel model for turbulent crowds, using which I am able to simulate stop-and-go waves as well as chaotic behavior symptomatic of crowd turbulence. These reproduce the features of well-known crowd disasters at the Hajj and the Love Parade disaster of 2010. My model also shows good correspondence with quantitative metrics proposed for detecting crowd turbulence and

`8y73-7lFBNE` (2010).

establishes the importance of modeling friction for simulating crowd turbulence.

My model can simulate such crowds with globally computed stress at interactive rates on current generation PCs. I believe this finding opens up possibilities of real-time response to crowd conditions that may pose threats of developing into crowd turbulence. It should also prove to be a valuable aid for safety planning and disaster prevention, by allowing planners to test designs for the possibility of crowd turbulence.

The numerical validation shown in this chapter has focused on the limited set of metrics reported in previous work. My model is consistent with real-world data as measured by these metrics. The lack of detailed real-world datasets prevents more extensive validation of the model, my hypothesis, and the underlying forces.

In order to ensure efficient simulation, I compute stress using the solver used by [Narain et al., 2010] which approximates the Frobenius norm using the infinity norm. Though the error of such an approximation is bounded, it may cause minor deviations from expected behavior. A solver that can determine stress under Frobenius norm constraints could improve accuracy of computed results.

Though my model closely matches data from real instances of crowd turbulence, I believe that results can be further improved with automatic parameter tuning [Guy et al., 2012] and validation of my model with more real-world data. Expanding the limited set of publicly available data from crowd disasters with fine-grained information about pedestrian trajectories and velocities would allow the development of significantly more accurate models.

Chapter 5: Conclusion

In this dissertation, I have proposed the use of coupled Eulerian-Lagrangian models for simulating complex large-scale flows. These models are capable of simulating previously infeasible scenarios efficiently and scalably on current parallel architectures. I have demonstrated their applicability for incompressible and weakly compressible flows like fluids and crowds. As demonstrated by the examples in this dissertation, these methods can recreate phenomena not just for visualization in computer graphics, but also those of interest in the domain of engineering.

5.1 Summary of Results

I have presented a technique of simulating fluids at large-scales by using Eulerian velocity-based models coupled with Lagrangian vorticity-based models. Lagrangian models for discretizing vorticity offer a compact representation for fluid velocity in domains where the fluid is homogeneous, but can be expensive to the point of being intractable at surfaces. Conversely, Eulerian velocity-based models excel at modeling velocity transitions, including discontinuous transitions at fluid-air and fluid-solid boundaries. By coupling these two models together – using Lagrangian vortex models in the fluid interior, coupled with Eulerian velocity models at surfaces – I demonstrate that large-scale scenes which were previously too expensive to simulate on PCs, can be simulated efficiently using such a coupled model.

To enable interaction between these disparate models for simulation, I propose a novel coupling algorithm that conserves momentum and vorticity across subdomains. I also propose an algorithm to represent incompressible vector fields like velocity using Lagrangian vortex singularity elements. This can be leveraged to compactly store and modify incompressible vector fields, particularly for controllable fluid simulations. In addition, my pro-

posed method can be used with dynamically changing domain boundaries, which allows flexible simulation with memory footprints up to 1000x smaller than comparable Eulerian simulations, along with similar performance improvements.

Secondly, I have presented a method for simulating collision avoidance behavior of pedestrians in large-scale environments. I propose solving this problem of long-range collision avoidance using a recursive formulation using short-range collision avoidance at multiple spatial and temporal scales. The proposed algorithm is agnostic to the underlying short-range algorithm, and as demonstrated using two examples, it can be applied to many such algorithms to significantly improve collision avoidance behavior without compromising performance.

To quantify the benefits of this approach, I propose a metric to measure path smoothness, by quantifying the deviation of velocity from a desired or mean velocity. Real-world crowds consistently exhibit very small deviations from this mean path, while crowds simulated with short-range avoidance demonstrate significantly higher deviations. Simulating the same scenarios with long-range avoidance using my proposed algorithm generates behavior closer to real-world behavior as noted qualitatively with visual inspection, and quantitatively with the proposed metric. In addition, completion times, i.e. the time taken for all agents to reach their goal, for crowds simulated using long-range avoidance are closer to real-world pedestrians than for short-range simulation. Finally, I also propose a simple scheme for hybrid simulations to ensure efficient collision avoidance at a wide range of densities. By using linear blending of Eulerian and Lagrangian collision avoidance algorithms, the most efficient algorithm can be used at all densities while maintaining smooth transitions with correct behavior.

In addition to simulating crowd behavior at large spatial scales, I also propose a method to simulate crowds at high densities. In such conditions, crowd behavior deviates significantly from that which we observe in everyday conditions. Collision avoidance becomes untenable as pedestrians reach states of near-constant contact and collision with others. Thus, a

model for crowd behavior dictated by collision response is needed to accurately simulate such conditions. I postulate that a model for the resulting inter-personal forces is essential for simulating crowds in such conditions. To compute inter-personal stress in such crowds, I propose a coupled Eulerian-Lagrangian model, where discrete Lagrangian agents are driven by forces defined using an Eulerian stress model.

My proposed model can replicate known crowd behaviors in such conditions, including stop-and-go waves, and more chaotic behaviors known collectively as crowd turbulence. In qualitative visual comparisons, and quantitative comparisons using known metrics, pedestrians simulated using my proposed algorithm show close correspondence to available data from two recent recorded instances of crowd turbulence, at the Hajj in Mecca, and the Love Parade disaster in Duisberg, Germany, in 2010. My proposed formulation can simulate crowds at interactive rates on current generation PCs. This is made possible by the use of a coarse large-scale Eulerian model for stress coupled with a fine grained Lagrangian model for pedestrian motion.

5.2 Limitations

I discuss specific limitations of each of my proposed methods in their corresponding chapters. In this chapter I discuss the limitations of my general approach. While coupled Eulerian-Lagrangian methods present a promising approach for modeling physical phenomena at varying scales, it is important to understand their limitations for correct application.

The utility of such coupled methods lies in utilizing two methods whose complexity scales in a complementary fashion with scale. For example, the complexity of vortex methods scales with vortex detail, and is unrelated to size, while that for Eulerian velocity-based methods scales with the size of the domain, but avoids the complexity explosion with detail by clamping the amount of velocity detail it maintains. Such a complementarity may not exist in all physical phenomena of interest. Even though such a complementarity exists, coupling two or more models presents a challenge unique to the phenomena under consideration, i.e.

the coupling algorithms presented in this dissertation may not be applicable directly to the phenomena of interest. The efficiency of such a method cannot be guaranteed, but is highly likely for cases where the underlying models are complementary for the scales in question.

It is also important to note that the approaches presented in this dissertation choose to discretize certain scales of the underlying phenomena, which existing models are not able model accurately while maintaining efficiency. Though the proposed approach is capable of modeling any scales of interest, the choice of underlying models and their parameters is governed by these particular scales, and may need to be modified appropriately for different phenomena. Particularly, for phenomena which are not completely understood, significant experimentation may be needed to converge to scales of interest. However, as long as the underlying models are capable to simulating particular scales, the interaction and interplay of phenomena at multiple scales can be modeled using this approach.

5.3 Future Work

Coupled Eulerian-Lagrangian models are a promising approach for various multi-scale physical phenomena, and open up avenues for modeling various physical phenomena, as well as improve the computational efficiency of existing models. Here, I list certain such examples, and other avenues of research made possible with the models proposed in this dissertation.

One aspect highlighted by the hybrid fluid simulation proposed in Chapter 2 is that it is possible to simulate any free-surface fluid using vorticity-based Lagrangian methods, using an Eulerian velocity-based model for vorticity creation and exchange at the surface. Thus, a promising avenue for future research would be to explore the creation of an accurate Lagrangian model for vorticity exchange at a free-surface that does not rely on an intermediate model. Doing so would reduce a 3-dimensional simulation to a 2-dimensional simulation using a Boundary Element Model, significantly reducing memory requirements, and opening the door for the use of Fast Multipole Methods for efficient fluid simulation using Lagrangian vortex methods for fluids in multiple states.

This dissertation focuses on enabling the possibility of simulations working at multiple spatial and temporal scales. However, it does not explore all possible avenues for improving simulation performance using this model. One key aspect is that components of the simulation at multiple spatial and temporal scales do not need to execute with the same time step—components modeling larger spatial and temporal scales can be executed on a coarser scale. In addition, the use of varied scales and specific computational kernels also lends an opportunity of exploiting vector and parallel architectures including multi-core CPUs, GPUs, as well as other many core architectures. The use of these methods on such data-parallel architectures opens up the possibility of simulating virtual environments impossible to recreate previously.

Though implementations of the proposed algorithms in this dissertation focus on shared memory architectures commonly seen in desktop computers, the same concepts can also be applied to message passing approaches used in multi-node clusters. For such architectures, the cost of accessing a layer in the memory hierarchy gets replaced by the cost of sending messages across nodes—farther nodes mimicking lower layers of the hierarchy. Similarly, a small working set implies a limited number of messages sent across nodes, improving performance. An interesting avenue for further work would be to map coupled Eulerian-Lagrangian models to MPI (Message Passing Interface) based architectures. Though the motivation to use these models remains the same, challenges faced in obtaining optimal performance on such architectures are likely to be very different, and may even provide insights to further improve the core models themselves.

5.4 Acknowledgments

I would like to thank the anonymous reviewers for their valuable suggestions; Adam Lake and Nico Galoppo at Intel for their support during the early stage of concept exploration. This research was supported in part by ARO Contract W911NF-04-1-0088,NSF awards 0917040, 0904990, 100057 and 1117127, and Intel.

REFERENCES

[Adams et al., 2007] Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. (2007). Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26. 6, 19

[Angelidis and Neyret, 2005] Angelidis, A. and Neyret, F. (2005). Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 87–96. ACM. 19

[Angelidis et al., 2006] Angelidis, A., Neyret, F., Singh, K., and Nowrouzezahrai, D. (2006). A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 25–32. Eurographics Association. 20

[Aranson and Tsimring, 2001] Aranson, I. S. and Tsimring, L. S. (2001). Continuum description of avalanches in granular media. *Phys. Rev. E*, 64:020301. 78

[Barrett et al., 1994] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA. 32

[Batty et al., 2007] Batty, C., Bertails, F., and Bridson, R. (2007). A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3):100. 18

[Bayazit et al., 2002] Bayazit, O. B., Lien, J.-M., and Amato, N. M. (2002). Better group behaviors in complex environments with global roadmaps. *Proc. 8th Intl. Conf. Artificial Life*, pages 362–370. 43

[Behringer et al., 1999] Behringer, R., Jaeger, H., and Nagel, S. (1999). Introduction to the focus issue on granular materials. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(3):509–510. 78

[Brackbill and Ruppel, 1986] Brackbill, J. U. and Ruppel, H. M. (1986). FLIP - A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65:314–343. 79

[Bridson and Müller-Fischer, 2007] Bridson, R. and Müller-Fischer, M. (2007). Fluid simulation: Siggraph 2007 course notes. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 1–81. ACM. 1, 22

[Cantarella et al., 2002] Cantarella, J., DeTurck, D., and Gluck, H. (2002). Vector calculus and the topology of domains in 3-space. *Amer. Math. Monthly*, 109(5):409–442. 26, 30

[Carlson, 2004] Carlson, M. T. (2004). Rigid, melting, and flowing fluid. Ph.D. Dissertation. 22

[Chentanez et al., 2007] Chentanez, N., Feldman, B. E., Labelle, F., O'Brien, J. F., and Shewchuk, J. R. (2007). Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 219–228. Eurographics Association. 18

[Chentanez et al., 2006] Chentanez, N., Goktekin, T. G., Feldman, B. E., and O'Brien, J. F. (2006). Simultaneous coupling of fluids and deformable bodies. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 83–89. 18

[Chorin, 1973] Chorin, A. J. (1973). Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57(04):785–796. 19

[Cottet and Koumoutsakos, 1998] Cottet, G. H. and Koumoutsakos, P. D. (1998). *Vortex Methods: Theory and Practice*. Cambridge University Press. 20, 23

[Curtis and Manocha, 2012] Curtis, S. and Manocha, D. (2012). Pedestrian simulation using geometric reasoning in velocity space. *Pedestrian and Evacuation Dynamics*. 77

[Dostál and Schöberl, 2005] Dostál, Z. and Schöberl, J. (2005). Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Comput. Optim. Appl.*, 30(1):23–43. 86

[Elcott et al., 2007] Elcott, S., Tong, Y., Kanso, E., Schröder, P., and Desbrun, M. (2007). Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26. 19

[Fedkiw et al., 2001] Fedkiw, R., Stam, J., and Jensen, H. W. (2001). Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 15–22. ACM. 6, 18

[Feldman et al., 2005] Feldman, B. E., O'Brien, J. F., and Klingner, B. M. (2005). Animating gases with hybrid meshes. In *Proceedings of ACM SIGGRAPH 2005*. 20

[Feurtey, 2000] Feurtey, F. (2000). Simulating the collision avoidance behavior of pedestrians. Master's thesis, University of Tokyo, School of Engineering. 43

[Fiorini and Shiller, 1998] Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *Intl. J. on Robotics Research*, 17(7):760–772. 43

[Foster and Fedkiw, 2001] Foster, N. and Fedkiw, R. (2001). Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 23–30. ACM. 18

[Foster and Metaxas, 1996] Foster, N. and Metaxas, D. (1996). Realistic animation of liquids. *Graph. Models Image Process.*, 58. 18

[Fruin, 1971] Fruin, J. (1971). *Pedestrian planning and design*. Metropolitan Association of Urban Designers and Environmental Planners. 48, 76

[Fruin, 1993] Fruin, J. (1993). The Causes and Prevention of Crowd Disasters. In Smith, R. A. and Dickie, J. F., editors, *Engineering for Crowd Safety*. Elsevier. 76

[Golas et al., 2013a] Golas, A., Narain, R., Curtis, S., and Lin, M. C. (2013a). Hybrid long-range collision avoidance for crowd simulation. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1. 13

[Golas et al., 2013b] Golas, A., Narain, R., and Lin, M. (2013b). Hybrid long-range collision avoidance for crowd simulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 29–36. ACM. 13

[Golas et al., 2014] Golas, A., Narain, R., and Lin, M. C. (2014). Continuum modeling of crowd turbulence. *Phys. Rev. E*, 90:042816. 14

[Golas et al., 2012] Golas, A., Narain, R., Sewall, J., Krajcevski, P., Dubey, P., and Lin, M. (2012). Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph.*, 31(6):148:1–148:9. 12, 17

[Guy et al., 2009] Guy, S., Chhugani, J., Kim, C., Satish, N., Lin, M. C., Manocha, D., and Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 43

[Guy et al., 2010] Guy, S. J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. (2010). Pledestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 119–128, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 43, 77

[Guy et al., 2012] Guy, S. J., van den Berg, J., Liu, W., Lau, R., Lin, M. C., and Manocha, D. (2012). A statistical similarity measure for aggregate crowd dynamics. *ACM Trans. Graph.*, 31(6):190:1–190:11. 44, 100

[Heigeas et al., 2003] Heigeas, L., Luciani, A., Thollot, J., and Castagné, N. (2003). A physically-based particle model of emergent crowd behaviors. *Proc. Graphikon '03*, 2. 43

[Helbing et al.,] Helbing, D., Farkas, I., and Vicsek, T. Simulating dynamical features of escape panic. *Nature*, page 2000. 85

[Helbing et al., 2007] Helbing, D., Johansson, A., and Al-Abideen, H. Z. (2007). Dynamics of crowd disasters: An empirical study. *Phys. Rev. E*, 75:046109. xiii, 75, 76, 77, 78, 81, 84, 85, 90, 97

[Helbing and Molnár, 1995] Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51:4282. 43, 78

[Helbing and Mukerji, 2012] Helbing, D. and Mukerji, P. (2012). Crowd disasters as systemic failures: analysis of the love parade disaster. *EPJ Data Science*, 1(1):7. 75, 76, 77, 97, 98

[Hoogendoorn et al., 2007] Hoogendoorn, S., Luding, S., Bovy, P., Schreckenberg, M., and Wolf, D. (2007). *Traffic and Granular Flow ' 03*. Springer. 78

[Hughes, 2002] Hughes, R. L. (2002). A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507 – 535. 84

[Hughes, 2003] Hughes, R. L. (2003). The flow of human crowds. *Annual Review of Fluid Mechanics*, 35(1):169–182. 58, 77

[Josserand et al., 2004] Josserand, C., Lagrée, P.-Y., and Lhuillier, D. (2004). Stationary shear flows of dense granular materials: a tentative continuum modelling. *The European Physical Journal E: Soft Matter and Biological Physics*, 14:127–135. 78

[Kamphuis and Overmars, 2004] Kamphuis, A. and Overmars, M. (2004). Finding paths for coherent groups using clearance. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 19–28. 43

[Kapadia et al., 2009] Kapadia, M., Singh, S., Hewlett, W., and Faloutsos, P. (2009). Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 215–223, New York, NY, USA. ACM. 43

[Kim et al., 2007] Kim, B., Liu, Y., Llamas, I., and Rossignac, J. (2007). Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics*, 13:135–144. 18

[Kim et al., 2008] Kim, T., Thürey, N., James, D., and Gross, M. (2008). Wavelet turbulence for fluid simulation. *ACM Trans. Graph.*, 27:50:1–50:6. 18

[Klingner et al., 2006] Klingner, B. M., Feldman, B. E., Chentanez, N., and O'Brien, J. F. (2006). Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH 2006*, pages 820–825. 18

[Lakoba et al., 2005] Lakoba, T. I., Kaup, D. J., and Finkelstein, N. M. (2005). Modifications of the Helbing-Molnar-Farkas-Vicsek social force model for pedestrian evolution. *SIMULATION*, 81:339. 43

[Lamarche and Donikian, 2004] Lamarche, F. and Donikian, S. (2004). Crowd of virtual humans: a new approach for real-time navigation in complex and structured environments. *Computer Graphics Forum*, 23(3):509–518. 43

[Lentine et al., 2011] Lentine, M., Aanjaneya, M., and Fedkiw, R. (2011). Mass and momentum conservation for fluid simulation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 91–100. ACM. 18

[Lentine et al., 2010] Lentine, M., Zheng, W., and Fedkiw, R. (2010). A novel algorithm for incompressible flow using only a coarse grid projection. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 114:1–114:9. ACM. 19

[Lindsay and Krasny, 2001] Lindsay, K. and Krasny, R. (2001). A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow. *J. Comput. Phys.*, 172:879–907. 16, 29

[Losasso et al., 2004] Losasso, F., Gibou, F., and Fedkiw, R. (2004). Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 457–462. ACM. 6, 19

[Losasso et al., 2008] Losasso, F., Talton, J., Kwatra, N., and Fedkiw, R. (2008). Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14:797–804. 20

[Moussaïd et al., 2011] Moussaïd, M., Helbing, D., and Theraulaz, G. (2011). How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*. 77, 78

[Mullen et al., 2009] Mullen, P., Crane, K., Pavlov, D., Tong, Y., and Desbrun, M. (2009). Energy-preserving integrators for fluid animation. *ACM Trans. Graph.*, 28:38:1–38:8. 18

[Müller et al., 2003] Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 154–159. Eurographics Association. 19

[Narain et al., 2009] Narain, R., Golas, A., Curtis, S., and Lin, M. C. (2009). Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.* 41, 44, 49, 58, 77

[Narain et al., 2010] Narain, R., Golas, A., and Lin, M. C. (2010). Free-flowing granular materials with two-way solid coupling. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 173:1–173:10, New York, NY, USA. ACM. 85, 86, 100

[Narain et al., 2008] Narain, R., Sewall, J., Carlson, M., and Lin, M. C. (2008). Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph.*, 27:166:1–166:8. 18

[Ondřej et al., 2010] Ondřej, J., Pettré, J., Olivier, A.-H., and Donikian, S. (2010). A synthetic-vision based steering approach for crowd simulation. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 123:1–123:9, New York, NY, USA. ACM. 39

[Paris et al., 2007] Paris, S., Pettre, J., and Donikian, S. (2007). Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, 26(3):665–674. 43

[Park and Kim, 2005] Park, S. I. and Kim, M. J. (2005). Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 261–270. ACM. 19

[Pelechano et al., 2007] Pelechano, N., Allbeck, J. M., and Badler, N. I. (2007). Controlling individual agents in high-density crowd simulation. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108. 43

[Pettré et al., 2005] Pettré, J., Laumond, J.-P., and Thalmann, D. (2005). A navigation graph for real-time crowd animation on multilayered and uneven terrain. *First Intl. Workshop on Crowd Simulation*, pages 81–90. 43

[Pfaff et al., 2010] Pfaff, T., Thuerey, N., Cohen, J., Tariq, S., and Gross, M. (2010). Scalable fluid simulation using anisotropic turbulence particles. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10, pages 174:1–174:8, New York, NY, USA. ACM. 18

[Pipes, 1953] Pipes, L. A. (1953). An operational analysis of traffic dynamics. *J. Appl. Phys.*, 24(3):274 – 281. 76, 81

[Polus et al., 1983] Polus, A., Schofer, J., and Ushpiz, A. (1983). Pedestrian flow and level of service. *Journal of Transportation Engineering*, 109(1):46–56. 76

[Portz and Seyfried, 2010] Portz, A. and Seyfried, A. (2010). Modeling Stop-and-Go Waves in Pedestrian Dynamics. *ArXiv e-prints*. 77

[Quecedo et al., 2004] Quecedo, M., Pastor, M., Herreros, M. I., and Merodo, J. A. F. (2004). Numerical modelling of the propagation of fast landslides using the finite element method. *International Journal for Numerical Methods in Engineering*, 59(6):755–794. 78

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH*, 21:25–34. 43, 77

[Reynolds, 1999] Reynolds, C. W. (1999). Steering behaviors for autonomous characters. *Game Developers Conference*, 1999. 43

[Robinson-Mosher et al., 2008] Robinson-Mosher, A., Shinar, T., Gretarsson, J., Su, J., and Fedkiw, R. (2008). Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, 27:46:1–46:9. 18

[Schechter and Bridson, 2008] Schechter, H. and Bridson, R. (2008). Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*. 18

[Selle et al., 2008] Selle, A., Fedkiw, R., Kim, B., Liu, Y., and Rossignac, J. (2008). An unconditionally stable maccormack method. *J. Sci. Comput.*, 35:350–371. 18

[Selle et al., 2005] Selle, A., Rasmussen, N., and Fedkiw, R. (2005). A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 910–914. ACM. 18

[Seyfried et al., 2009] Seyfried, A., Passon, O., Steffen, B., Boltes, M., Rupprecht, T., and Klingsch, W. (2009). New insights into pedestrian flow through bottlenecks. *Transportation Science*, pages 395–406. 44

[Seyfried et al., 2005] Seyfried, A., Steffen, B., Klingsch, W., and Boltes, M. (2005). The fundamental diagram of pedestrian movement revisited. *J. Stat. Mech.*, (10). 44, 76

[Shao and Terzopoulos, 2007] Shao, W. and Terzopoulos, D. (2007). Autonomous pedestrians. *Graph. Models*, 69(5-6):246–274. 43

[Sin et al., 2009] Sin, F., Bargteil, A. W., and Hodgins, J. K. (2009). A point-based method for animating incompressible flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 19

[Singh et al., 2009] Singh, S., Kapadia, M., Faloutsos, P., and Reinman, G. (2009). Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds*, 20(5-6):533–548. 44

[Solenthaler and Pajarola, 2009] Solenthaler, B. and Pajarola, R. (2009). Predictive-corrective incompressible sph. *ACM Trans. Graph.*, 28:40:1–40:6. 19

[Stam, 1999] Stam, J. (1999). Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128. 18

[Sud et al., 2008] Sud, A., Andersen, E., Curtis, S., Lin, M., and Manocha, D. (2008). Real-time path planning in dynamic virtual environments using multi-agent navigation graphs. *IEEE Trans. Visualization and Computer Graphics*, 14(3):526–538. 43

[Sud et al., 2007] Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., and Manocha, D. (2007). Real-time navigation of independent agents using adaptive roadmaps. In *Proc. ACM Symp. Virtual Reality Software and Technology*, pages 99–106. 43

[Sugiyama et al., 2001] Sugiyama, Y., Nakayama, A., and Hasebe, K. (2001). 2-dimensional optimal velocity models for granular flows. In *Pedestrian and Evacuation Dynamics*, pages 155–160. 43

[Sung et al., 2004] Sung, M., Gleicher, M., and Chenney, S. (2004). Scalable behaviors for crowd simulation. *Computer Graphics Forum*, 23(3 (Sept)):519–528. 43

[Toselli and Widlund, 2004] Toselli, A. and Widlund, O. (2004). *Domain Decomposition Methods - Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer. 27

[Treuille et al., 2006a] Treuille, A., Cooper, S., and Popovic, Z. (2006a). Continuum crowds. *ACM Trans. Graph.*, 25(3):1160–1168. 43

[Treuille et al., 2006b] Treuille, A., Lewis, A., and Popović, Z. (2006b). Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 826–834. ACM. 19, 77

[van den Berg et al., 2009] van den Berg, J., Guy, S. J., Lin, M. C., and Manocha, D. (2009). Reciprocal n-body collision avoidance. *Proc. Intl. Symposium on Robotics Research*. 43, 50, 57, 61, 63, 64

[van den Berg et al.,] van den Berg, J., Guy, S. J., Snape, J., Lin, M. C., and Manocha, D. RVO2 Library: Reciprocal collision avoidance for real-time multi-agent simulation. 50, 61, 64

[van den Berg et al., 2008a] van den Berg, J., Lin, M. C., and Manocha, D. (2008a). Reciprocal velocity obstacles for realtime multi-agent navigation. *Proc. IEEE Conf. Robotics and Automation*, pages 1928–1935. 43, 57

[van den Berg et al., 2008b] van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. C. (2008b). Interactive navigation of individual agents in crowded environments. *Proc. ACM Symposium on Interactive 3D Graphics and Games*, pages 139–147. 43

[Weidmann, 1993] Weidmann, U. (1993). *Transporttechnik der Fussgaenger*. Number 90. 2 edition. 44, 76, 81

[Weißmann and Pinkall, 2009] Weißmann, S. and Pinkall, U. (2009). Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *Workshop in VRIPS 2009*. Eurographics Association. 19

[Weißmann and Pinkall, 2010] Weißmann, S. and Pinkall, U. (2010). Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 115:1–115:12. ACM. 20, 23, 30

[Wendt et al., 2007] Wendt, J. D., Baxter, W., Oguz, I., and Lin, M. C. (2007). Finite volume flow simulations on arbitrary domains. *Graph. Models*, 69:19–32. 18

[Wicke et al., 2009] Wicke, M., Stanton, M., and Treuille, A. (2009). Modular bases for fluid dynamics. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 39:1–39:8. ACM. 19

[Yu and Johansson, 2007] Yu, W. and Johansson, A. (2007). Modeling crowd turbulence by many-particle simulations. *Phys. Rev. E*, 76:046105. 78, 97

[Zhang et al., 2011] Zhang, J., Klingsch, W., Schadschneider, A., and Seyfried, A. (2011). Transitions in pedestrian fundamental diagrams of straight corridors and t-junctions. *J. Stat. Mech.*, 2011(06):P06004. 44, 76

[Zhang et al., 2012] Zhang, J., Klingsch, W., Schadschneider, A., and Seyfried, A. (2012). Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. *Journal of Statistical Mechanics: Theory and Experiment*, 2:2. xii, 44, 61, 72, 76

[Zhu and Bridson, 2005] Zhu, Y. and Bridson, R. (2005). Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 965–972, New York, NY, USA. ACM. 18, 35