Geometric Collision Avoidance for Heterogeneous Crowd Simulation

Stephen J. Guy

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
2012

ABSTRACT

STEPHEN J. GUY: Geometric Collision Avoidance for Heterogeneous Crowd Simulation
(Under the direction of Ming C. Lin and Dinesh Manocha)

Simulation of human crowds can create plausible human trajectories, predict likely flows of pedestrians, and has application in areas such as games, movies, safety planning, and virtual environments. This dissertation presents new crowd simulation methods based on geometric techniques. I will show how geometric optimization techniques can be used to efficiently compute collision-avoidance constraints, and use these constraints to generate human-like trajectories in simulated environments. This process of reacting to the nearby environment is known as *local navigation* and it forms the basis for many crowd simulation techniques, including those described in this dissertation.

Given the importance of local navigation computations, I devote much of this dissertation to the derivation, analysis, and implementation of new local navigation techniques. I discuss how to efficiently exploit parallelization features available on modern processors, and show how an efficient parallel implementation allows simulations of hundreds of thousands of agents in real time on many-core processors and tens of thousands of agents on multi-core CPUs. I analyze the macroscopic flows which arise from these geometric collision avoidance techniques and compare them to flows seen in real human crowds, both qualitatively (in terms of flow patterns) and quantitatively (in terms of flow rates).

Building on the basis of these strong local navigation models, I further develop many important extensions to the simulation framework. Firstly, I develop a model for global navigation which allows for more complex scenarios by accounting for long-term planning around large obstacles or emergent congestion. Secondly, I demonstrate methods for using data-driven approaches to improve crowd simulations. These include using real-world data to automatically tune parameters, and using perceptual user study data to introduce behavioral variation.

Finally, looking beyond geometric avoidance based crowd simulation methods, I discuss methods for objectively evaluating different crowd simulation strategies using statistical measures. Specifically, I focus on the problem of quantifying how closely a simulation approach matches real-world data. I propose a similarity metric that can be applied to a wide variety of simulation approaches and datasets.

Taken together, the methods presented in this dissertation enable simulations of large, complex humans crowds with a level of realism and efficiency not previously possible.

To Laura and QBasic; one never forgets their first true loves…

ACKNOWLEDGEMENTS

There are many people who, through support, kindness, and constructive conflict, have contributed a great deal to my research in graduate school and eventually this thesis. First and foremost, I want to thank my advisors Ming Lin and Dinesh Manocha who have both been a constant source of guidance throughout my entire six years here. Ming and Dinesh were some of the first faculty I met at UNC and have, at every point, pushed me to accomplish as much as possible during my time in the department. I have also benefited greatly from my committee members: Lana Lazebnik for teaching me virtually everything I know about computer vision and machine learning, Marc Niethammer for great discussions on optimization theory, and Jatin Chhugani whose world class expertise on computer architecture and practical approach to conducting research has left a positive impact on every project I have worked on since meeting him.

I will also be forever indebted to two amazing mentors: Leonid Zhigilei for introducing me to the world of university research, and Pradeep Dubey for showing me the best of what is possible at an industrial lab. My work at their respective labs inspired my desire for research and contributed directly to the material presented here.

I would additionally like to thank Jur P. van den Berg, both for his cheerful mentorship and for introducing me to RVO – which would forever change my research career. I've also been blessed with a large supply of collaborators in my research: Sujeong Kim, Sahin Patil, Sean Curtis, Jamie Snape, Rahul Narain, and Wenxi Liu, any use of the word 'I' in this document likely reflects a great deal of contribution from these people.

I would also like to thank my parents who have always done everything possible to support my education and nurture my development. Finally, I would like the thank my finacée, and life partner, Laura; without her constant love, support, encouragement, picnics in the lab, and paper editing I would likely not have made it through a single deadline, let alone a dissertation's worth of them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CA	Cellular Automata
FVO	Finite-time horizon Velocity Obstacle
GPS	Global Position Systems
LiDAR	Light Detection And Ranging
ORCA	Optimal Reciprocal Collision Avoidance
PLE	Principle of Least Effort
RVO	Reciprocal Velocity Obstacle
VO	Velocity Obstacle
VR	Virtual Reality

CHAPTER 1

Introduction

Building good models of human crowds is important to many fields such as social science, statistical physics, fire safety planning, and other areas of science and engineering. These models help us gain an understanding of crowds and can facilitate growth, develop more effective societies and cities, and can lead to safer cultural activities and events. By using these models to virtually test design possibilities it is possible to save time, money, and other resources. Devising such models requires us to draw upon developments from various fields which study crowds ranging from sociology to psychology to biomechanics.

This dissertation proposes new computational models to simulate trajectories and movement of human crowds. These models seek to predict plausible or likely motion of humans in crowds, both in terms of paths and other aspects such as densities, speeds, and overall behavioral patterns. While this is a broad and challenging goal, realistic computer models of how human crowds move around obstacles, respond to stress, and avoid others can have many applications. For example, crowd simulations can help city planners determine what potential designs will improve flow through the city or expedite emergency evacuations. Likewise, computer game designers could use such models to create crowds of computer controlled characters which respond plausibly to the actions of the main character.

However, creating these realistic models is a challenging and multi-faceted problem. The range of motion humans perform and the variety of situations which people encounter is almost endless. To make progress, it becomes necessary to limit the scope of the problem to modeling some subset of human motion. In this dissertation, I focus on heterogenous, agent-based crowd simulations; that is, crowds of individual entities where each individual has been given its own intent, characteristics and personality. I further explore a dichotomy of various simulation approaches in the next section.

1.1 Crowd Simulation Overview

The term "crowd simulation" admits a broad range of interpretations that can vary based on the target application and domain of study. For the purpose of this dissertation, I define crowd simulation as the process of creating visual simulations of motion of one or more individuals interacting with each other and their environment. As such, crowd simulation is a complex task that has several components. At a high level, creating a crowd simulation involves at least three important, distinct tasks: firstly, specifying the scenario (see Section 1.1.1), secondly, computing the movement of the simulated individuals (see Section 1.1.2), and thirdly, rendering animated imagery consistent with the computed motion of each agent (see Section 1.1.3). This partitioning follows the broad description of multi-agent simulation offered by Reynolds (Reynolds, 1999).

1.1.1 Defining the Scenario

At an abstract level, crowds can be viewed as a set of entities (i.e., individuals in the crowds), with a collection of goals, and a set of obstacles which form the environment. Defining each of these sets specifies a unique instance of a crowd simulation scenario. However, the method of specifying an individual's state, their goals, and the environment will vary based on the simulation methods being used. In this dissertation, I adopt broad interpretations for these terms to allow for a wide variety of different simulation approaches; each term is described in more detail below.

Crowd State: I define the state of a crowd as a collection of properties associated with each individual in the crowd that can vary between individuals. In all the work presented in this dissertation the state includes the position and velocity of each individual in the crowd. At times, the crowd state will need to be expanded to include other attributes such as an individual's orientation, personal space requirements, or personality to allow for more rich and complex simulations.

Goal Selection: Each individual in the crowd is assumed to have a physically embodied goal. The representation of this goal can vary based on the simulation scenario. For example, a goal of a desired position can be represented by a static location (point in 2D or 3D space). A goal of entering a room or moving to a specific area can be represented by a target region of 2D space. Goals can also be dynamic, such as moving to follow a particular individual, and a complex high-level goal can be broken down into a sequence of subgoals (e.g., go here, then there).

More complex desired behavior can be specified using navigation fields (Patil et al., 2011). These are spatially varying flow fields that define a desired velocity for an individual that depends on that individual's position. Navigation fields can be used to specify desired overall flow patterns. For example, Figure 1.1 shows a still from the simulation of the Tawaf ritual, a part of the Islamic pilgrimage to Mecca for the Hajj. A navigation field is used to specify the desired counterclockwise motion pattern around the Kabah (central black structure) that is part of the ritual (for example see the simulations in (Schneider et al., 2011; Curtis et al., 2011)).

Environment Specification: Beyond specifying the goal points and regions, an environment for a crowd simulation may include walls, obstacles, and other regions not accessible to individuals in the crowd. In this dissertation, I assume these obstacles are represented by (potentially non-convex) polygons that bound the obstacle area. It is the responsiblity of the crowd simulation to ensure that agents do not enter the obstacle regions. In the simulation shown in Figure 1.1, the central Kabah structure is represented by a bounding rectangle which defines a region the agents may not enter during their motion.

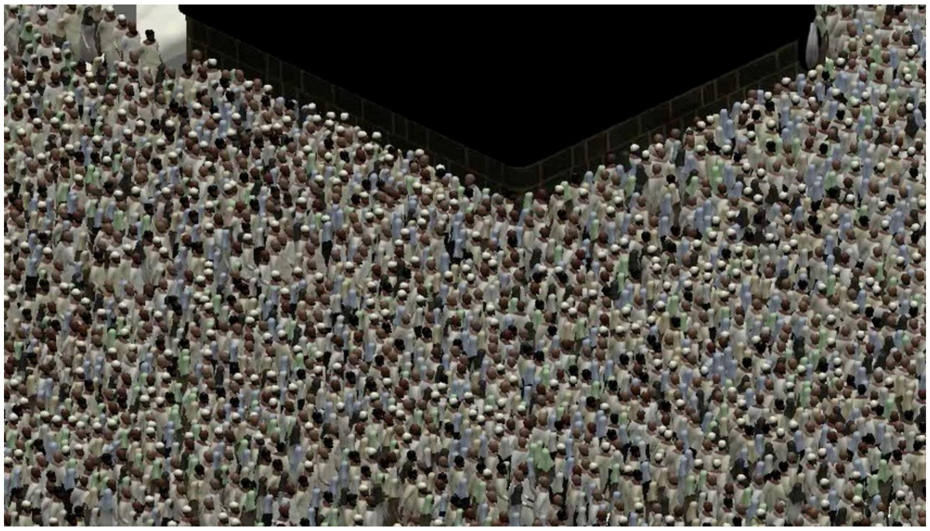


Figure 1.1: Still from a visualization of a simulation of the Tawaf ritual. Agents receive a desired velocity from a flow field directing them in a counterclockwise motion around the Kabah (central structure).

1.1.2 Computing Crowd Motion

Given the current state of the individuals in the crowd and the set of goals to reach, the crowd motion problem is to compute short-term paths which are responsive to the local conditions but still lead to an entity's ultimate goals. Reynolds refers to the process of intermediate-level planning as *steering behaviors*. Steering behaviors in crowds are responsible for collision avoidance and local navigation, with the goal of making individuals in a crowd walk around each other in a natural manner (Reynolds, 1999).

Simply relying on local collision avoidance can result in agents getting stuck behind obstacles like walls and other local minima in complex environments. This problem can be alleviated by using global path planning, a process which computes long-term paths that reach a target destination without getting stuck at local minima. Standard techniques from the fields of robotics and motion planning, such as probabilistic roadmaps and visibility graphs, can be used to compute paths that are free from these local minima (LaValle, 2006).

This dissertation focuses on the problem of motion computation and steering behaviors for crowd simulation. More details on the approaches for motion planning in crowds are given in Section 1.2.

1.1.3 Animating and Rendering of Crowds

If the crowd simulation is to be used in a graphical environment, it is necessary to visually render the simulation results. Creating a natural visual rendering of the computed motion of human characters is a challenging but well-studied problem in computer graphics. The most standard approach for animating a human's motion is known as a 'walk cycle', which involves translating the display of the human character's center of mass along the motion vector while moving the character's arms and legs in a simple, periodic cycle. The cyclic animation of limb motion can be either created by an artist or recorded from the motion of real humans using motion capture equipment.

The position of joint and limbs needed to produce natural walk cycles can be computed using Inverse Kinematics (IK) (Peiper, 1968). IK can be used to devise algorithms to dynamically adapt limb motion in order to create a variety of different types of walks, for example (Calvert et al., 1993) and (Bruderlin and Calvert, 1996). Recorded data of human motion can also be used to drive the motion of animated characters. For example, (Ko and Badler, 1993) provides a method of converting 2D rotoscoped data of human motion into walking animation appropriate for people moving at different speeds and for people of different genders, weights, and body types. Data-driven methods for animated characters have been extended into 3D as well (Witkin and Popovic, 1995), and can be used to synthesize a large variety of different recorded motions (Kovar et al., 2002).

When displaying very large crowds, rendering can become especially challenging. Recent papers have addressed this rendering challenge in different ways. One option is to use differing levels of details, which switches from complex visual models to simpler ones when characters get far away from the camera (Luebke, 2003). This process can be used in an aggressive fashion by using very simple geometry such as a single quad with a detailed human texture as described in (Tecchia et al., 2002), a technique known as impostors. Impostors can be combined with a geometric level of detail system to allow real-time rendering of thousands of agents as in (Dobbyn et al., 2005). Impostors can be improved by using more detailed, animated 2D geometry as is done in (Kavan et al., 2008). Using such methods can significantly reduce rendering costs, which is critical when creating real-time animation of crowds for interactive applications like games, training, and VR.

1.2 Motion in Crowds

Several methods can be employed to compute the motion of individuals in crowds. The choice of the most appropriate method depends on the type of crowds being simulated, the complexity of the environment, and the densities of the agents. The following subsections explain some of the high-level considerations that affect crowd simulation algorithms.

1.2.1 Heterogeneous Crowd Simulations

An important consideration in crowd simulation is whether the crowds being simulated are homogeneous or heterogeneous. The study of heterogeneous crowds date back to at least the work of Le Bon, who analyzed how members of a crowd can have different races, genders, intents, and backgrounds (Bon, 1895). This dissertation focuses on these types of heterogeneous crowds, where each individual maintains a distinct, observable identity. This identity may be seen in a person's goals, desired speeds, politeness, cooperation, and many other factors affecting their motion.

In contrast, homogenous crowds occur when a clear unity of action leads to a "disappearance of conscious personality" leading to a homogeneity of motion (Bon, 1895). When simulating homogenous crowds, it is possible to exploit the coherence in individual motion to accelerate the simulation. For example the work of (Hughes, 2002) and (Hughes, 2003) presents a differential equation that uniformly dictates the flow of crowds across space. The homogeneity assumption found in this work has been relaxed by more recent approaches, such as (Treuille et al., 2006) which allows for a small, fixed number of goals, and (Narain et al., 2009) which allows for a unique goal for each individual in the crowd. However, these methods still enforce a continuum of motion, which results in nearby individuals being grouped together to share common characteristics and states such as similar speeds and motion, thereby reducing the heterogeneity of the simulation.

Agent-based Crowd Simulations:

In contrast to continuum methods, agent-based simulation methods allow for true heterogeneity in an individual's motions. In these simulations, each person in the crowd is represented as a simulated agent (typically in a 1-to-1 correspondence). Since the motion of each agent can be computed independently, it is possible to simulate crowds with vastly differing motion for different agents, even those moving right next to each other.

An agent-based approach was taken in the Boids algorithm, the pioneering work of Reynolds (Reynolds, 1987). The Boids algorithm produced steering behavior that resembled flocking, herding, and school behaviors commonly observed in animal motion. Similar to Boids, this dissertation presents an agent-based model, but designed for simulation of human crowd motion.

1.2.2 Global Navigation

Computing an agent's motion breaks down into two distinct tasks: local navigation and global navigation. Global navigation is the process of determining a long-term path that is free of local minima. To perform global navigation, it is common to use a spatial navigation data-structure, such as roadmaps, to guide agents around large, static obstacles in order to avoid local minima.

A roadmap is a graph consisting of a set of vertices positioned in freespace (*i.e.* not inside an obstacle), and a set of edges connecting these vertices. Two vertices are connected by a link if and only if the direct path between the two nodes is traversable (*i.e.* no obstacles block the direct path). Such roadmaps can be constructed by an artist, or automatically generated using visibility graphs (Lozano-Pérez and Wesley, 1979), probabilistic roadmaps (Kavraki et al., 1996), or other techniques (Latombe, 1991; LaValle, 2006).

Agents can compute global paths using such roadmaps. An agent chooses the closest node to its current position as a start vertex, and the closest vertex to its goal position as an end vertex. Paths can then be represented by a set of edges between vertices. To find the shortest path between the start and end vertex, standard graph search techniques, such as Dijkstra's algorithm (Dijkstra, 1959) or A* (Hart et al., 1968), can be used. The direction along the current link towards the next vertex in the path forms a basis for an agent's *goal velocity*. It is the responsibility of the local navigation algorithm to move towards the goal velocity without colliding into other nearby agents.

1.2.3 Velocity Space Planning for Local Navigation

There are a variety of ways to perform local navigation and collision avoidance. A common technique is to plan based on the current positions of other nearby agents, with closer agents being given a greater weight when planning than farther away agents (for example by using repulsive forces which are inversely proportional to a neighbors distance). This technique is commonly used in simulations, including work by (Reynolds, 1987), (Helbing and Molnar, 1995), and (Pelechano et al., 2007).

These techniques implicitly assume that all of an agent's neighbors will remain in the same position for the foreseeable future. As such, it is common for such methods to result in collisions between agents when traveling at high speeds and in complex cross-flows. Additionally, as agents come towards each other, forces typically grow exponentially, which can lead to issues with simulation stability and result in unnatural and oscillatory motion.

In contrast, this dissertation presents methods which plan motion for each agent in *velocity space*. Velocity space is a vector space of relative velocities for each agent. Each neighboring agent will constrain an agent's potential velocities based on its relative positions and velocities. Planning in velocity space corresponds to making a short-term constant velocity assumption. This assumption allows for better avoidance behavior in complex, congested scenarios with lots of cross flows. See (Fiorini and Shiller, 1998) for a further discussion of velocity space planning.

Chapter 2 discusses how to adapt velocity space planning to scenarios where multiple agents simultaneously avoid each other. It also presents an efficient implementation of a velocity space planning technique designed to exploit the hardware trends discussed in Section 1.3.

1.3 Modern Processor Architecture Trends

There are various applications of crowd simulations where strong computational performance is important. For example, in computer games and related areas such as training and VR, crowds often must respond interactively to the actions of a user. This requirement means that the actions of all the agents in the crowd must be computed in a few milliseconds or less to reach the common simulation update target of 30Hz. Another example is architectural crowd flow analysis, where crowd simulations are used to predict crowd flows in new architectural or urban domains and to analyze which changes improve these flows. Architectural analysis benefits from exploring as many options as possible, as quickly as possible. When designing new simulation algorithms where computational performance is a major issue, it is important to look at the current trends in modern processors and understand how to exploit them.

The most important trend in processors is Moore's law, which predicted an exponential growth in transistor count and computational power. While this trend has continued over the past 50 years, the growth in single-core performance has not kept pace with the growth in the number of transistors. Despite increasing clock speeds and advances in instruction level parallelism (ILP) such as out-of-order execution, deep pipelines, and multiple ALUs, the increase in performance of single-core processing has slowed considerably in recent years due to power requirements (Borkar and Chien, 2011). Instead, growth in performance comes from larger vector processors (data-level parallelism) and more cores per chip (thread-level parallelism). Unlike ILP, exploiting these types of parallelism requires careful thought from an algorithm designer. In this dissertation, I develop algorithms that can utilize these increasing levels of parallelism to accelerate crowd simulation.

Another important trend in modern processors is the fact that the growth in compute power is outpacing the growth in cache speeds and memory access times. The result of this trend is to effectively reduce the cost of compute relative to each memory access. These architectural developments have a two-fold impact on the simulation algorithm. Firstly, because memory access is more likely to be a computation bottleneck, more compute-intensive approaches to crowd simulation can run at very similar speeds as other simpler, memory-access-bound methods. Secondly, it creates a greater importance for developing data-structures that can efficiently exploit the cache and other features of the memory subsystem in order to reduce the effect of memory access times. Together, these trends guide the development of the algorithms I present in this dissertation.

1.4 Thesis statement

My thesis is as follows:

Geometric optimization approaches to collision avoidance can be combined with global navigation techniques to provide an efficient means of computing the trajectories of individuals in crowd simulations. Such methods can exploit parallelism in modern processors to run in real time, produce heterogeneous virtual crowds, match human locomotion data, and reproduce emergent phenomena and flows seen in real crowds.

1.5 Main Results

In support of my thesis statement, I present several results relating to generating efficient, large-scale, realistic crowd simulations. By using geometric optimization approaches I develop simulation techniques that are scalable, with applications from modeling the motion of just two people crossing paths to crowds of hundreds of thousands of people. To this end there are three main contributions. First, I demonstrate how to exploit the architecture of modern computer processors to efficiently compute these geometric optimization problems (Section 1.5.1). Second, I discuss how to incorporate important biomechanically inspired considerations into the optimization framework to produce more realistic simulations (Section 1.5.2). Third, I provide a method for simulating the effect of various personalities on a person's path (Section 1.5.3). Additionally, I validate the motion predicted by this model against data of real humans paths to analyze the correctness of the proposed models (Section 1.5.4).

1.5.1 Parallel, Optimization-Based Collision Avoidance

One important result from this dissertation is a new technique for distributed collision avoidance. I present the ClearPath algorithm, an adaptation of velocity-based collision avoidance methods from robotics to the domain of distributed collision avoidance for crowd simulations.

Additionally, I discuss how the implementation of ClearPath exploits recent hardware trends to maximize performance. This performance improvement comes from increasing parallel processing, exploiting SIMD/vector processing, and using spatial data structures to reduce memory access costs.

The resulting algorithm is capable of simulating crowds in a scalable manner. ClearPath can compute smooth, collision-free trajectories for small crowds, in dense congestion, and

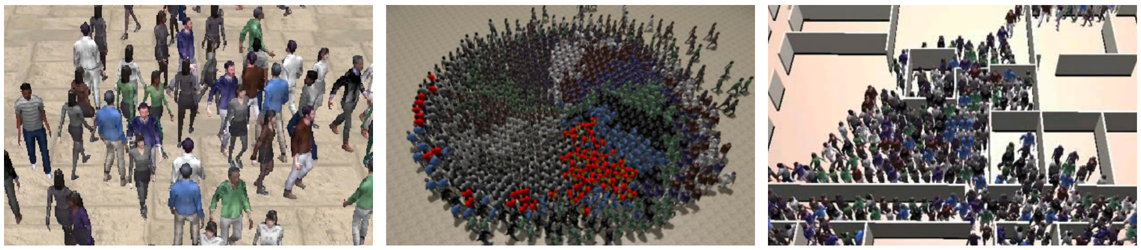

(a) Bi-directional Flow (b) Dense Congestion (c) Evacuation

Figure 1.2: ClearPath performing collision avoidance on several large-scale scenarios with hundreds of agents at varying densities.

in the presence of obstacles, see Figure 1.2 for examples. These results were published in (Guy et al., 2009) and are discussed in Chapter 2.

1.5.2 Crowd Simulation based on the Principle of Least Effort

Additionally, in this dissertation I present a simulation algorithm that builds on distributed collision techniques, like ClearPath, to create biomechanically inspired crowd trajectories based on the Principle of Least Effort (PLE). The main result of this work is a unified framework for simulating crowds that reproduces several emergent phenomena commonly found in real crowds, and that matches real-world trajectory data from human crowds.

To understand the accuracy of these PLE-based simulations, I present several forms of analysis and comparison between the simulated trajectories and the known motion from actual human paths. I demonstrate that well-known emergent behaviors such as lane-formation and arching arise in both real human crowds and in these simulations, in both simple targeted set-ups and in simulations of complex real-world scenarios as shown in Figure 1.3. These results were published in (Guy et al., 2010a) and are discussed in Chapter 3.

I also present the results of quantitative analysis of the accuracy of this simulation method. This involves both a direct comparison between real and simulated paths in scenarios consisting of one to two isolated individuals, as well as a comparison of real and

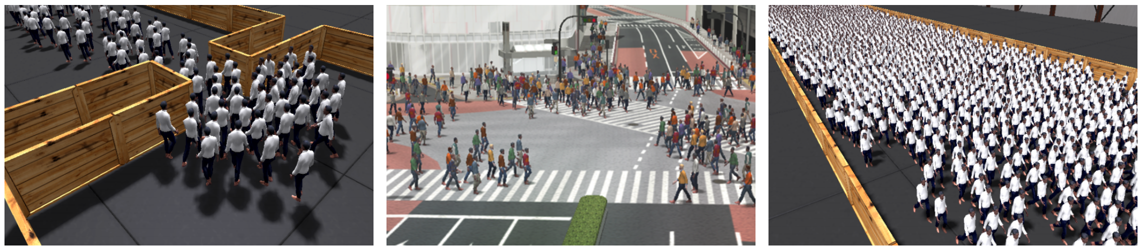

(a) Narrow Passage (b) Shibuya Crossing (c) Long Corridor

Figure 1.3: Images of various crowd simulations using ideas inspired by biomechanics research. The simulations show a variety of emergent phenomena such as lane formation.

simulated aggregate flow rates in larger scenarios. These results were published in (Guy et al., 2012) and are discussed in Chapter 4.

1.5.3 Data Use in Crowd Modeling

I also present results related to incorporating various data into crowd simulations. Due to the complexity of human behavior, no model that simply tries to create rules from first principles can easily capture all aspects and variety of human motion as part of a crowd. Because of this variation, it can be necessary to use data of real crowds to guide simulations. I present a method for matching simulations closely to real world data. These results were published in (Guy et al., 2010b) and are discussed in Chapter 5.

I also present a method based on data from user-studies to simulate the diversity of behaviors that comes from different personalities. The result is a method that is able to simulate individuals with different apparent personalities such as “aggressive” or “shy”. These results were published in (Guy et al., 2011) and are discussed in Chapter 6.

1.5.4 Simulation Validation form Real-World Data

Real-world crowd data can also be used to validate crowd simulations. I present techniques to compute a similarity measure of how close a given simulation technique matches a real-world dataset. This similarity measure, called the *Entropy Metric*, uses statistical inference techniques to estimate the amount of error a simulation has with respect

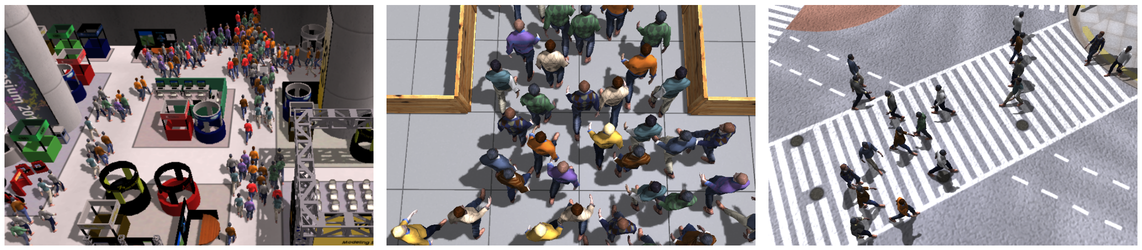

(a) Varying Personalities (b) Narrow Hallway (c) Street Crossing

Figure 1.4: (a) Data-driven crowd simulations (b-c) Data-driven evaluation scenarios

to validation data. Such a metric can ultimately be used to measure the accuracy of different simulation techniques, including those presented in this dissertation, at capturing different recorded crowd behaviors such as those seen in Figure 1.4b-c. These results are discussed in Chapter 7.

1.6 Organization

The rest of this dissertation is organized as follows: Chapter 2 presents the ClearPath algorithm for local navigation and provides a parallel implementation of the approach. This chapter also briefly discusses the Optimal Reciprocal Collision Avoidance (ORCA) collision-avoidance approach used later in the subsequent work.

Chapter 3 describes an algorithm for incorporating a biomechanically motivated version of the Principle of Least Effort into a distributed collision avoidance technique in order to create simulations of human crowds. Chapter 4 compares the motion computed by Least-Effort crowd simulations to the motion and flow of human crowds based on standard techniques used by pedestrian analysis and fire-safety researchers.

Chapter 5 demonstrates how to use optimization techniques to improve the accuracy of crowd simulations. Chapter 6 introduces a data-driven method to model variations due to personality differences. Chapter 7 discusses how to robustly use real-world data to validate simulation methods. Finally, Chapter 8 presents a summary of my contributions and a discussion of future work.

CHAPTER 2

ClearPath: A Method for Decentralized Collision Avoidance

2.1 Introduction

From record-setting crowds at rallies and protests to futuristic swarms of robots, our world is currently experiencing a continuing rise of complex, distributed collections of independently acting entities. All of these domains can be effective model as multi-agent system, that is as a set of interacting dynamic entities. With potential applications such as predicting crowd disasters, improving robot cooperation, and enabling the next generation of air travel, developing models to reproduce, control, predict and understand these types of systems is becoming critically important.

One of the key challenges in the area of multi-agent systems is to provide a means of producing and controlling the behaviors of the agents in the system, while maintaing important safety constraints. The work presented in this chapter focuses on maintaining collision-free motion while allowing each agent to reach it's own (externally specified) goal. This approach will provide a decentralized solution where each agent can make it's own navigation decisions without explicit communications with it's neighbors nor a central authority.

While there are many potential uses for distributed collision avoidance in different types of multi-agent systems, I will focus on uses in crowd simulation as that is the driving domain of this dissertation. In subsequent chapters, I will explain how I build on top of this distributed collision avoidance framework to make a full crowd simulation algorithm. In different domains (*e.g.* robotic) other considerations might be needed (sensor integration, uncertainty, etc.).

One of the goals of this work is to study the computational issues involved in enabling real-time agent-based simulation to exploit current architectural trends. Recent and future commodity processors are becoming increasingly parallel. Specifically, current GPUs and upcoming x86-based many-core processors such as the experimental Larrabee processors consist of tens or hundreds of cores, with each core capable of executing multiple threads and vector instructions to achieve higher parallel-code performance. Therefore, it is important to design collision avoidance and simulation algorithms such that they can exploit substantial amounts of fine-grained parallelism.

2.1.1 Main Results

This chapter describes an algorithm for distributed, multi-agent collision avoidance called ClearPath. The approach is based on the robotics concept of *velocity obstacles* (VO) that was introduced by Fiorini and Shiller (Fiorini and Shiller, 1998) for motion planning among dynamic obstacles. By using an efficient velocity-obstacle based formulation that can be combined with any underlying multi-agent simulation, the local collision avoidance computations can be reduced to solving a quadratic optimization problem that minimizes the change in underlying velocity of each agent subject to non-collision constraints

In describing ClearPath I will present several main results:

1. A scaleable, decentralized approach for multi-agent collision avoidance
2. A multi-threaded implementation with dynamic load balancing
3. A method for exploiting SIMD/vector processing units for increased computation

The result is a polynomial-time algorithm for agents to compute collision-free, 2D motion in a distributed manner. When published, ClearPath was more than one order of magnitude faster than prior velocity-obstacle based methods. Since then, recent work such as (Snape et al., 2009) and (van den Berg et al., 2011) have sense directly built on the techniques proposed in this chapter and produce similar performance results.

2.1.2 Organization

The result of this chapter is organized as follows. Section 2.2 provides a brief review of closely related work. Section 2.3 describes in detail the ClearPath approach for local collision avoidance. Section 2.4 details a method for parallel implementation exploiting both thread-level and data-level parallelism. Sections 2.5 and 2.6 describes a basic method for incorporating this local collision avoidance method to simulate crowds and analyzes the performance of the resulting system. In Section 2.7, I discuss the limitations of this method and some scope for future work.

2.2 Previous Work

Different techniques have been proposed to model behaviors of individual agents, groups and heterogeneous crowds. Excellent surveys have been recently published including (Thalmann et al., 2006) and (Pelechano et al., 2008a). These include methods to model local dynamics and generate emergent behaviors such as (Reynolds, 1987) and (Reynolds, 1999); psychological effects and cognitive models like the work of (Shao and Terzopoulos, 2005) and (Yu and Terzopoulos, 2007); and cellular automata models and hierarchical approaches (Schadschneider et al., 2011). In this section, I give a brief overview of related work in collision detection and avoidance.

2.2.1 Collision detection and path planning

There is a rich literature on detecting collisions between two or multiple objects. Many fast algorithms have been proposed for checking whether objects overlap at a given time instance (discrete collision detection) or over a one dimensional continuous interval (continuous collision detection), a more though discussion of collision detection can be found in (Ericson, 2004).

The problem of computing collision-free motion for one or multiple robots among static or dynamic obstacles has been extensively studied in robot motion planning (LaValle, 2006). These include global algorithms based on randomized sampling, local planning techniques based on potential field methods, centralized and decentralized methods for multi-robot coordination, etc. In general, these methods are either too slow for interactive applications or may suffer from paths that get stuck at local minima.

2.2.2 Collision avoidance

Collision avoidance problems have been been studied in control theory, traffic simulation, robotics and crowd simulation. Optimization techniques for local collision detection have been proposed for a pair of objects, including separation or penetration distance computation between convex polytopes (Cameron, 1997; Lin, 1993) and local collision detection between convex or non-strictly convex polyhedra with continuous velocities (Faverjon and Tournassoud, 1987; Kanehiro et al., 2008).

Different techniques have been proposed for collision avoidance in group andcrowd simulations for example (Reynolds, 1999), (Foudil and Noureddine, 2027),(Sugiyama et al., 2001), (Musse and Thalmann, 1997), (Thalmann et al., 2006), (Helbing et al., 2005),(Lamarche and Donikian, 2004), and (Sud et al., 2007b). These techniques can be based on local coordination schemes, velocity models, prioritization rules, force-based techniques, or adaptive roadmaps. Other techniques have used LOD techniques to trade-off fidelity for speed such as (Yersin et al., 2008).

The notion of velocity obstacles (VO) was proposed for motion planning in dynamic environments and has been extended to deal with uncertainty in sensor data (Fiorini and Shiller, 1998; Fulgenzi et al., 2007; Kluge and Prassler, 2007). van den Berg et al. extended the VO formulation to compute collision avoid paths between agents (van den Berg et al., 2008a,b). Other extensions such as (Shiller et al., 2001) and (Paris et al., 2007a) have also been proposed. However, these techniques provide higher-order path-planning with implementations that are not yet fast enough for very large simulations.

2.2.3 Parallel algorithms

Many parallel algorithms have been designed for collision detection, Voronoi diagrams and related geometric computations that can exploit multiple cores or vector capabilities of GPUs (Owens et al., 2007). Sud et al. (Sud et al., 2007a) use GPU-based discretized Voronoi diagrams for collision detection among scenes with few hundred agents. There is extensive literature on developing parallel and distributed algorithms for multi-agent simulations (Tolwinski, 2002) including work on specialized hardware such as the cell processor (Reynolds, 2006).

2.3 Local Collision Avoidance

This sections presents the mathematical details of the ClearPath collision avoidance algorithm. The resulting approach is general and can be combined with different crowd and multi-agent simulation techniques.

Assumptions and Notation: ClearPath assumes that the scene consists of heterogeneous agents with static and dynamic obstacles. The behavior of each agent is governed by some extrinsic and intrinsic parameters and computed in a distributed manner for each agent independently. The overall simulation proceeds in discrete time steps and the state of each agent, including its position and velocity, is updated each time step. Given the position and velocities of all the agents at a particular time instant T, and a discrete time interval of ΔT, the goal is to compute a velocity (and therefore a linear path) for each agent that results in no collisions during the interval $[T, T + \Delta T]$. This velocity will be recomputed for every agent, every time step.

The algorithm, as presented in this chapter, also assume that the agents are moving on a 2D plane, though the approach can be extended to handle agents moving in 3D space.

At any time instance, each agent has the information about the position and velocity of nearby agents. This can be achieved efficiently, but storing the position of every agent in a kD-tree, neighbors searches can then be preformed in O(log n) times. Each agent is represented using a circle or convex polygon in the plane. If the actual shape of the agent is non-convex, its convex hull can be used. The resulting collision-avoidance algorithm becomes conservative in such cases. In the rest of the chapter, a circular agent will be assumed, though the algorithm can be easily extended to other convex shapes.

Given an agent A, $\mathbf{p}_A$, $\mathbf{r}_A$, and $\mathbf{v}_A$ is used to denote the agent's position, radius and velocity, respectively. The algorithm assumes that the underlying simulation algorithm uses intrinsic and extrinsic characteristics of the agent or some high level model to compute desired velocity for each agent ($\mathbf{v}_A^{des}$) during the timestep. The symbols $\mathbf{v}_{max}$ and $\mathbf{a}_{max}$ will represent the maximum velocity and acceleration, respectively, of the agent during a timestep. Furthermore, $q^{\perp}$ denotes a line perpendicular to line q.

2.3.1 Velocity Obstacles

ClearPath builds on the concept of velocity obstacles (Fiorini and Shiller, 1998). Let $A \oplus B$ represent the Minkowski sum of two objects A and B and let $-A$ denote the object A reflected in its reference point. Furthermore, let $\lambda(\mathbf{p}, \mathbf{v})$ denote the a ray starting at $\mathbf{p}$ and heading in the direction of $\mathbf{v}$: $\lambda(\mathbf{p}, \mathbf{v}) = \{\mathbf{p} + t\mathbf{v} \,|\, t \geq 0\}$.

Let A be an agent moving in the plane and B be a (moving) obstacle on the same plane. The velocity obstacle $VO_B^A(\mathbf{v}_B)$ of obstacle B to agent A is defined as the set consisting of all those velocities $\mathbf{v}_A$ for A that will result in a collision at some moment in time $(t \geq T)$ with obstacle B moving at velocity $\mathbf{v}_B$. This can be expressed as:

$$VO_B^A(\mathbf{v}_B) = \mathbf{v}_A \,|\, \lambda(\mathbf{p}_A, \mathbf{v}_A - \mathbf{v}_B) \cap B \oplus -A \neq \emptyset$$

This region has the geometric shape of a cone. Let $\phi(\mathbf{v}, \mathbf{p}, \mu)$ denote the distance of point $\mathbf{v}$ from $\mathbf{p}$ along μ:

$$\phi(\mathbf{v}, \mathbf{p}, \mu) = \{(\mathbf{v} - \mathbf{p}) \cdot \mu\}$$

Henceforth, the region inside the cone is represented as

$$VO_B^A(\mathbf{v}) = (\phi(\mathbf{v}, \mathbf{v}_B, \mathbf{p}_{ABleft}^{\perp}) \geq 0) \wedge (\phi(\mathbf{v}, \mathbf{v}_B, \mathbf{p}_{ABright}^{\perp}) \geq 0),$$

where $\mathbf{p}_{ABleft}^{\perp}$ and $\mathbf{p}_{ABright}^{\perp}$ are the inwards directed rays perpendicular to the *left* and *right* edges of the cone, respectively. The VO is a cone with its apex at $\mathbf{v}_B$ (Fig. 2.1).

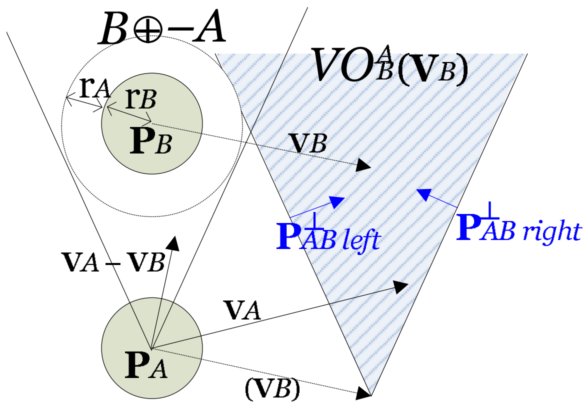


Figure 2.1: The Velocity Obstacle $VO_B^A(\mathbf{v}_B)$ of a disc-shaped obstacle B to a disc-shaped agent A. This VO represents the velocities that could result in a collision of A with B (i.e. potentially colliding region).

Van den Berg et al. (van den Berg et al., 2008a) presented an extension to VO called RVO. The resulting velocity computation algorithm guarantees an oscillation-free behavior for two agents. An RVO is formulated by moving the apex of the VO cone from $\mathbf{v}_B$ to $\frac{(\mathbf{v}_A + \mathbf{v}_B)}{2}$. If A has N nearby neighbors, then N avoidance cones will be created, and each agent needs to ensure that its desired velocity for the next frame, $\mathbf{v}_A^{des}$, is *outside* the union of all N velocity obstacle cones to avoid collisions. RVO algorithm performs random sampling of the 2D space in the vicinity of $\mathbf{v}_A^{des}$, and heuristically attempts to find a solution that satisfies the constraints. However, the RVO algorithm has the following limitations: RVO performs random sampling, and may not find a collision-free velocity even if there is a feasible solution. Since RVO uses infinite cones, the extent of the feasible region decreases as N increases. In practice, RVO formulation can become overly conservative for tightly packed scenarios.

2.3.2 Optimization Formulation for Collision Avoidance

The local collision avoidance problem for N agents as a combinatorial optimization problem. ClearPath extends the VO formulation by imposing additional constraints that can guarantee collision avoidance for each agent during a given interval. It accounts for this time interval by defining a truncated cone to represent the collision-free region during the time interval corresponding to ΔT. This finite time horizon velocity obstacle is denoted as FVO in this chapter. This truncation is a similar ideas presented in (Fiorini and Shiller, 1998), however here it is extend from one agent moving through unresponsive dynamic obstacles, to handle several responsive agents moving around each other. The original VO or RVO is defined using only two constraints (left and right), as shown in Fig. 2.1. The FVO is defined using a total of three conditions. The *two* boundary cone constraints of the FVO are same as that of RVO:

$\text{FVO}_{L}{}^{A}_{B}(\mathbf{v}) = \phi(\mathbf{v}, (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{p}^{\perp}_{ABleft}) \geq 0$

$\text{FVO}_{R}{}^{A}_{B}(\mathbf{v}) = \phi(\mathbf{v}, (\mathbf{v}_A + \mathbf{v}_B)/2, \mathbf{p}^{\perp}_{ABright}) \geq 0$

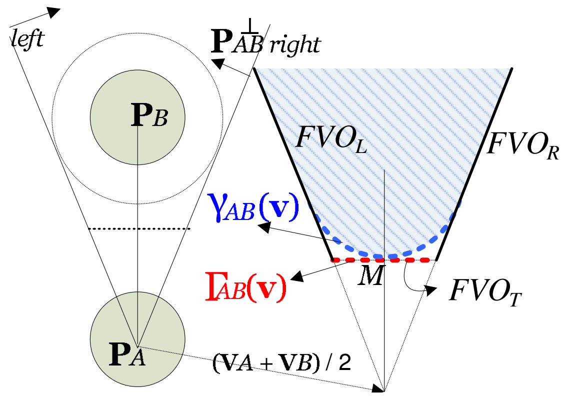

left
$\mathbf{P}^{\perp}_{AB}$ right
$\mathbf{P}_B$
FVO_L
FVO_R
$\gamma_{AB}(\mathbf{v})$
$\Gamma_{AB}(\mathbf{v})$
M
FVO_T
$\mathbf{P}_A$
$(\mathbf{V}A + \mathbf{V}B) / 2$

Figure 2.2: The Finite-time horizon Velocity Obstacle $\text{FVO}^{A}_{B}(\mathbf{v}_B)$ of a disc-shaped obstacle B to a disc-shaped agent A. This formulation takes into account the time interval for local collision avoidance.

Ann additionally boundary is used to form an FVO:
Collision avoidance in only guaranteed for the duration ΔT. This leads to a finite subset of the RVO cone that corresponds to the forbidden velocities that could lead to collisions in ΔT. The truncated cone (expressed as shaded region in Figure 2.2) is bounded by a curve $\gamma_{AB}(\mathbf{v})$ (derivation is given in Appendix A). Due to efficiency reasons, $\gamma_{AB}(\mathbf{v})$ is replaced with a conservative linear approximation $\Gamma_{AB}(\mathbf{v})$. Note that this approximation is conservative in nature, and in practice, this increases the area of the truncated cone by a small amount. This constitutes the additional constraint, defined as: FVO$_T{}^A_B(\mathbf{v}) =$

$$\begin{aligned}\Gamma_{AB}(\mathbf{v}) &= \lambda\left(M - \widehat{\mathbf{p}_{AB}^{\perp}} \times \eta,\ \mathbf{p}_{AB}^{\perp}\right), \text{where}\\ \eta &= \tan\left(\sin^{-1}\frac{\mathbf{r}_A + \mathbf{r}_B}{|\mathbf{p}_{AB}|}\right) \times (|\mathbf{p}_{AB}| - (\mathbf{r}_A + \mathbf{r}_B)), \text{and}\\ M &= (|\mathbf{p}_{AB}| - (\mathbf{r}_A + \mathbf{r}_B)) \times \widehat{\mathbf{p}_{AB}} + \frac{\mathbf{v}_A + \mathbf{v}_B}{2}\end{aligned}$$

Any feasible solution to all of constraints, which are separately formulated for each agent, will guarantee collision avoidance. A new, constraint satisfying velocity for each agent (in a distributed manner) which minimizes the deviation from $\mathbf{v}_A^{des}$–the velocity desired by the underlying simulation algorithm.

Let B_1,.., B_N represent the N nearest neighbors of an agent A. The computation of a new velocity ($\mathbf{v}_A^{new}$) can be posed as the following optimization problem:
Minimize $\|(\mathbf{v}_A^{new} - \mathbf{v}_A^{des})\|_2$ such that:

$$\begin{gathered}((\sim FVO_{LB1}^{A}(v_A^{new}) \cup \sim FVO_{RB1}^{A}(v_A^{new}) \cup \sim FVO_{TB1}^{A}(v_A^{new})) \cap \sim FVO_{CB1}^{A}(v_A^{new})) \cap\\ \vdots\\ ((\sim FVO_{LBN}^{A}(v_A^{new}) \cup \sim FVO_{RBN}^{A}(v_A^{new}) \cup \sim FVO_{TBN}^{A}(v_A^{new})) \cap \sim FVO_{CBN}^{A}(v_A^{new}))\end{gathered}$$

This is a quadratic optimization function with non-convex linear constraints for each agent. There is polynomial time solution when the dimensionality if the constrains is constant – in this case two.

Let the union of all the FVO from an agent's neighbors be referred to its potentially colliding region ($\mathcal{PCR}$), and the boundary segments of each neighbor's FVO as collectively the Boundary Edges ($\mathcal{BE}$). $\mathcal{BE}$ consists of $3N$ boundary segments – 3 from each neighbor of A. By exploiting the geometric nature of the problem it is to possible to derive the following lemma (proof in Appendix A):

Lemma 1: *If $\mathbf{v}_A^{des}$ is inside PCR, $\mathbf{v}_A^{new}$ must lie on the boundary segment of the FVO of one of the neighbors.*

In many simulations, there are other constraints on the velocity of an agent. For example, kineodynamic constraints (LaValle, 2006), which impose certain bounds on the motion (e.g. maximum velocity or maximum acceleration). In case the optimal solution to the quadratic optimization problem doesn't satisfy these bounds, the constraints can be relaxed by removing the furthest agent from A, and recomputing the optimal solution by considering only $N - 1$ agents. This relaxation step is carried on until an optimal solution satisfying all the constraints is obtained.

2.3.3 ClearPath Implementation

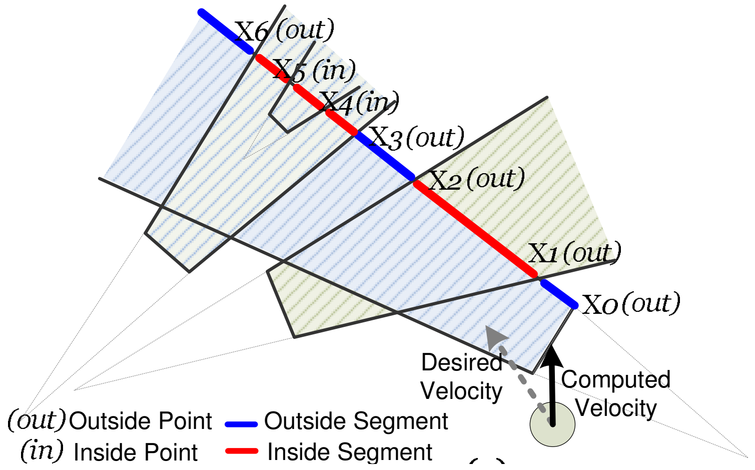


Figure 2.3: Classifying FVO boundary subsegments as Inside/Outside of the remaining FVO regions for multi-agent simulation. The optimization algorithm performs these classification tests to compute a non-colliding velocity for each agent.

This mathematical optimization formulation can be used to design a fast algorithm to compute a collision-free velocity for each agent independently. Specifically, Lemma 1 can be exploited to compute all possible intersections of the boundary segments of $\mathcal{BE}$ with each other. Consider segment X in Figure 2.3. The k intersection points of the FVO region labeled as X_1, .. , X_k. Note that these points are stored in a sorted order of increasing distance from the corresponding end point (X_0) of the segment. Each intersection point can be further classified as being *Inside* or *Outside* of $\mathcal{PCR}$. After performing this classification, the subsegments between these points can also as being *Inside* or *Outside* of the $\mathcal{PCR}$, based on the following lemma (proof in Appendix A):

Lemma 2: *The first subsegment along a segment is classified as Outside iff both its end points are tagged as Outside. Any other subsegment is classified as Outside iff both its end points are Outside, and the subsegment before it is Inside the* $\mathcal{PCR}$.

For example, both X_0 and X_1 are tagged as *Outside*, and hence the subsegment X_0X_1 is tagged as *Outside*. However, the subsegment X_1X_2 is tagged as *Inside* since X_0X_1 is *Outside* the *PCR*.

After classifying the subsegments of $\mathcal{BE}$, *Outside* subsegments are considered, the one closest to $\mathbf{v}_A^{des}$ is returned as the new velocity. The ClearPath algorithm performs the following steps for each agent:

Step 0. Given an agent, query the kD-tree for the N nearest agents, and compute the FVO constraints to arrive at $\mathcal{BE}$.

Step 1. Compute the normals of the each segment in $\mathcal{BE}$.

Step 2. Compute the intersection points along each segment of $\mathcal{BE}$ with the remaining segments of $\mathcal{BE}$.

Step 3. Classify the intersection points as *Inside* or *Outside* of the *PCR*.

Step 4. Sort the intersection points for each segment with increasing distance from its corresponding end point.

Step 5. Classify the subsegments along each segment as *Inside* or *Outside*, and compute/maintain the closest point for the *Outside* subsegments.

Step 6. In case the resulting solution does not satisfy the kineodynamic or velocity constraints, relaxing the constraints by removing the FVO corresponding to the furthest neighbor, and repeat the algorithm with fewer agents.

For M number of total intersections segments in $\mathcal{BE}$, the runtime of the algorithm for a single agent is O$(N(N+M))$.

Collision Resolution: Due to initialization conditions, numerical precision errors, and approximations in implementation is still it possible to have collision between agents. In cases when agents are colliding, they should choose a new velocity that resolves the collision as quickly as possible. This results in an additional set of constraints in velocity space this can be conservatively approximated as a cone (Figure 2.4). In this case, the Potentially Colliding Region is the intersection between two circles. The first circle, P, is the set of maximal velocities reachable by an agent in a single time step (a circle of radius $a_{max} \times \Delta T$). The second circle, R, is the Minkowski difference of the two agents: $B \oplus -A$. The region which lies in the 1^{st} circle, but not the second is the set of velocities which escapes the collision in 1 time step. The region which lies in both circles are reachable velocities which *fail* to resolve the collision next time step. This area is conservatively approximated with a cone, which create an additional constraint for any agents that are colliding. These constraints are combined with the existing PCR from the neighboring agents. A colliding agent's preferred velocity is set to **0** so that the smallest possible velocity which resolves the collision is chosen. This will minimize oscillations due to collision resolution.

2.4 P-ClearPath: Parallel Collision Avoidance

The current trend is for processors to deliver high-performance through multithreading and wider SIMD instructions. This section describes a data-parallel extension of ClearPath that can exploit the capabilities of current multi-core CPUs and many-core accelerators.

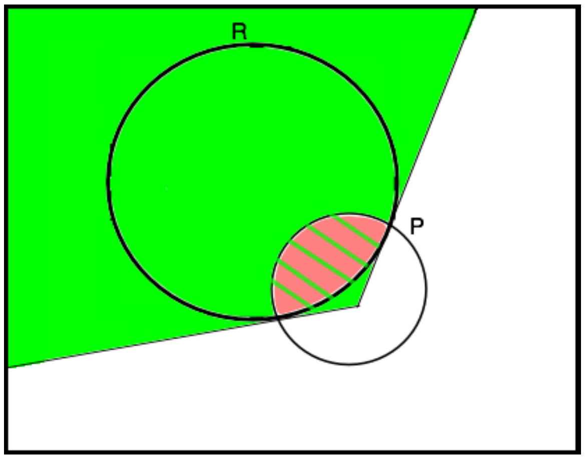


Figure 2.4: Red lens area: Velocities which will not remove agent from collision this time-step. Green cone: Conservative conical approximation of red area. Circle **R**: Minkowski sum of colliding agents. Circle **P**: Positions current agent can reach next time step.

ClearPath operates on a per-agent basis in a distributed manner, finding each agent's nearest neighbors and computes a velocity that is collision-free with respect to those neighbors.

There are two fundamental ways of exploring Data-Level parallelism (henceforth referred to as DLP).

Intra-Agent: Consider Figure 2.5(a). For each agent, DLP can be used within the ClearPath computation. Since the agents operate in 2D, they can perform their X and Y coordinate updates in a SIMD fashion. This approach does not scale to wider SIMD widths.

Inter-Agent: Operate on multiple agents at a time, with each agent occupying a slot in the SIMD computation. This approach is scalable to larger SIMD widths, but needs to handle the following two issues:

1. Non-contiguous data access: In order to operate on multiple agents, ClearPath requires *gathering* their obstacle data structure into a contiguous location in memory. After computing the collision-free velocity, the results need to be *scattered* back to their respective non-contiguous locations. Such data accesses become a performance bottleneck without efficient *gather/scatter* operations.

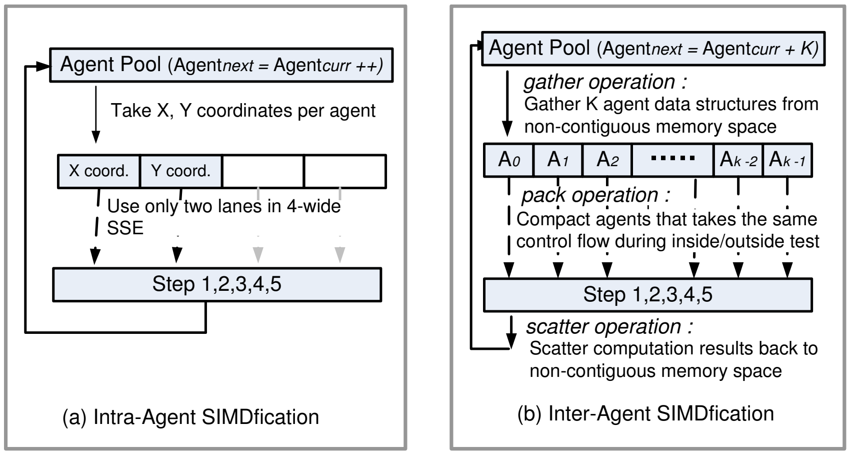

Agent Pool (Agentnext = Agentcurr ++)
Take X, Y coordinates per agent
X coord.
Y coord.
Use only two lanes in 4-wide SSE
Step 1,2,3,4,5
(a) Intra-Agent SIMDfication
Agent Pool (Agentnext = Agentcurr + K)
gather operation :
Gather K agent data structures from non-contiguous memory space
A0
A1
A2
Ak-2
Ak-1
pack operation :
Compact agents that takes the same control flow during inside/outside test
Step 1,2,3,4,5
scatter operation :
Scatter computation results back to non-contiguous memory space
(b) Inter-Agent SIMDfication

Figure 2.5: Data-Parallel Computations: (a) Intra-Agent SIMDfication for SSE (b) Inter-Agent SIMDfication for wide SIMD

2. Incoherent branching: Multiple agents within a SIMD register may take divergent paths. This degrades SIMD performance, and is a big performance limiter because of intersection computations and inside/outside tests. One or more of the agents may terminate early, while the remaining ones may still be performing their comparison operations.

The current SSE architectures on commodity CPUs does not have efficient instructions to resolve the above two problems. Hence, it is necessary to use the intra-agent SIMDfication approach, but this obtains only moderate speedups (see Section 2.5). For the remainder of this section, I will focus on exploiting wider SIMD, with the SIMD width being $\mathcal{K}$-wide.

P-ClearPath adopts the Inter-agent approach, and performs computation on $\mathcal{K}$ agents together. Figure 2.5(b) shows a detailed mapping of the various steps in ClearPath algorithm. For collision-free velocity computation, each agent A_i is given as input its neighboring velocity obstacles (truncated cones) and the desired velocity. The steps performed by each agent are described in Section 2.3.3.

First the obstacle data structure *gathers* $\mathcal{K}$ agents into contiguous chunks of memory and then loads various fields as needed by the SIMD operation. Although each of the steps listed there map themselves well to SIMD, there are a few *important issues* that need to be addressed.

1. Varying number of neighbors for each agent: This affects each of the steps, decreasing the SIMD utilization. For example, if one of the agents in the SIMD computation has N neighbors, and the second one has $N/2$, the second agent is masked out for half of the execution and does not perform any computation. Reordering the agents based on their neighbor count, and executing agents that have the same number of neighbors together in a SIMD fashion helps address this issue. Since the number of agents is a relatively small number, this reordering runs in linear time, and takes up insignificant portion of runtime.

2. Constraint Relaxation by some agents: After computing the collision-free velocity, it is indeed possible for some agents to not satisfy their kineodynamic or other constraints – thereby going back to the start and computing the solution with one less constraint in an iterative manner. A *pack instruction* (Seiler et al., 2008) can be exploited to improve the efficiency of this step. After all the agents have completed one iteration, the slots that need to be recomputed are identified with a mask, and are *packed* their data structure (i.e. the cone information) together in a separate memory location. This step is performed for all the agents. After the first iteration, the result is a *contiguous* list of agents that have failed the constraints, and need to be operated upon. The are operated on again – loading $\mathcal{K}$ agents each time and operating on them in a SIMD fashion as before. Note that after termination, the computed results need to be *scattered* to the appropriate memory location.

3. Classifying points as inside/outside: This is the most important part of the algorithm. While classifying points as being *inside* or *outside* of the truncated cones, their orientation is tested with respect to each truncated cone, and as soon as it is detected being *inside* any of the cones, it does not need to be tested against the remaining cones. However, it is often the case that some other point within the SIMD register is still being tested and the computation for other lanes is wasted. In the worst case, the SIMD utilization may be as low as $1/\mathcal{K}$.

The ClearPath's algorithm efficiency can be improved as follows. After testing the orientation with respect to the *first* cone, the points contiguously as described above are *packed*. In the subsequent iteration, the points are loaded and tested against the next cone, and the process repeated. Note that in comparison to the scalar version of the code, each point is tested against the same number of cones in the SIMD code. However, to improve SIMD efficiency, it is *packed*, and then retrieved for each cone it is checked against. This increases the overhead, but improves the SIMD efficiency to around $\mathcal{K}/3$-$\mathcal{K}/2$. With the above discussed modifications, and appropriate support for *gather/scatter* and *pack* instructions, P-ClearPath should achieve around $\mathcal{K}/2$ SIMD speedup as compared to the scalar version. A detailed analysis with SIMD scaling for each step in CleatPath algorithm is discussed in Sec. 2.5.3.

2.5 Implementation and Results

This sections describes the implementation as part of a simple crowd simulation and demonstrate the performance of both the serial and parallel version of the algorithms on various benchmarks.

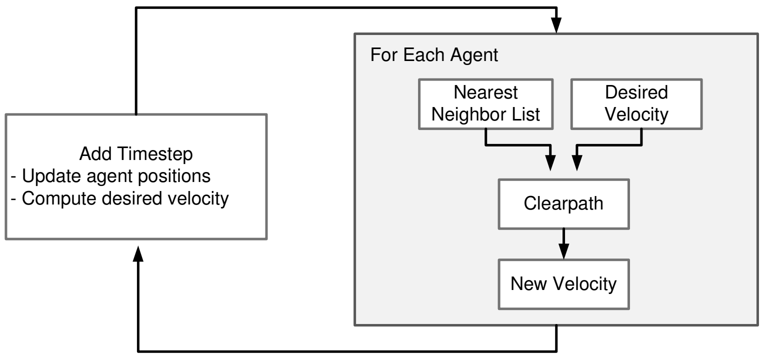


Figure 2.6: For each agent, each time step, ClearPath takes a desired velocity from a multi-agent simulation, and a list of nearby neighbors and computes a new minimally perturbed velocity with avoids collisions. This velocity is then used as the agent's new velocity.

To test the ClearPath, it was incorporated into two open-source crowd simulation systems: the RVO-Library (van den Berg et al., 2008a) and OpenSteer (Reynolds, 1999). These simulations provided the desired velocities, which ClearPath minimally modified to provide collision-free motion among the agents. A diagram of this process is shown in Figure 2.6. *New Velocities* were applied using simple Euler integration. The *Nearby Neighbor List* was obtained efficiently using a kD-tree. *ClearPath* was implemented as described in Section 2.3.3. The algorithm for the *Desired Velocity* for each agent depends on the simulation and is described briefly below.

RVO-Library: This library provides a global goal for each agent and can perform collision avoidance. The built-in RVO based collision avoidance was removed and replaced with ClearPath. The global navigation is based on a graph-based roadmap that is pre-computed for a given environment. At runtime, A* is used to search the graph and compute a desired path for each agent towards its goal position. The direction along the path provides the desired velocity, $\mathbf{v}^{des}$, for each agent.

OpenSteer: OpenSteer uses steering forces to guide agents towards their goals and away from each other. Two configurations where tested in connection with OpenSteer. For one, it output was used directly as an agent's desired velocity. In the second, the OpenSteer's collision avoidance components were removed from computing it's desired velocity and only ClearPath's collision avoidance was used.

Through the similar procedures the ClearPath collision-avoidance algorithm could be incorporated into other multi-agent simulation systems.

Different types of benchmarks were used to evaluate the performance of the algorithms. These varied from simple game-like scenes with a few dozen agents to complex urban scenes with tens to hundreds of thousands of agents. In these tests, the collision avoidance part of local navigation was found to take $50\% - 80\%$ of the total runtime and is a major bottleneck in the the performance of the overall multi-agent system. The algorithms were evaluated on different scenarios with varying the number of agents from 500 to 250K. Each agent is modeled as a heterogeneous agent and separate collision avoidance on each agent was performed in a distributed manner.

Performance Comparison of Serial Algorithm: The performance of the serial implementation of ClearPath with roadmaps was compared versus the original RVO-Library and open source implementation of collision-avoidance in OpenSteer. All of them were running on a single Xeon core (running at 3.14 GHz) with no data parallelism. Note that all these algorithms result in different agent behaviors with varying velocities and motions, though each of them is a goal-directed simulation where each agent has a desired goal. Hence, it is hard to make direct comparisons, especially with OpenSteer whose behavior differed significantly from the other two.

In general, there was $8 - 12$X improvement when using ClearPath over original RVO-Library (Fig. 2.7), and the absolute running time of ClearPath is comparable to the collision-avoidance routine in OpenSteer. However, OpenSteer can results in many collisions among the agents. Fig. 2.7 shows the absolute frame rates of simulation systems that use RVO-Library, ClearPath and P-ClearPath for collision avoidance.

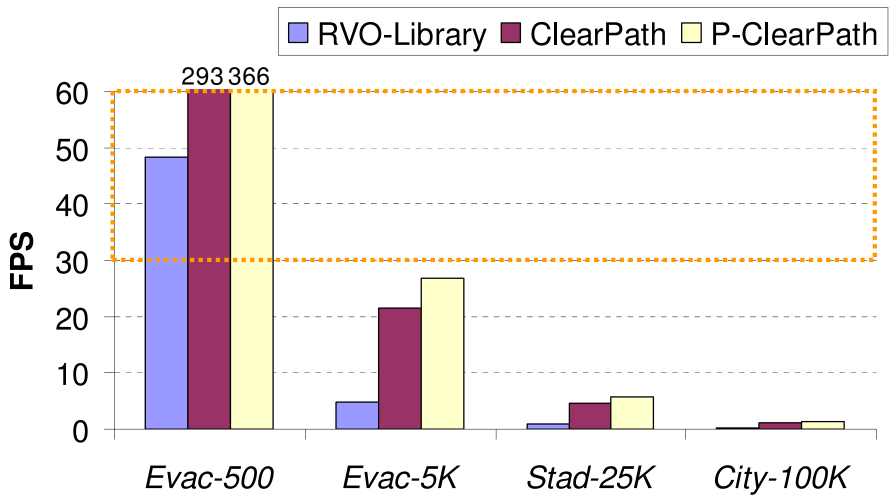


Figure 2.7: Performance of RVO-Library, ClearPath and SSE implementation of P-ClearPath measured on a single core Xeon. The ClearPath implementation is about 5X faster than RVO-Library. The SSE capability offers additional $25 - 50\%$ speedup.

2.5.1 Behavior Evaluation

Artificial scenarios were set up to evaluate local navigation and some emergent behaviors of ClearPath. The following emergent behaviors were observed from ClearPath in the benchmarks: lane-formation, vortices, slow down in congestion, respond to collisions, avoiding each other, swirl to resolve congestion, jamming at exits, and arching at narrow passages.

Circle-1K: $1,000$ agents start arranged uniformly around a circle and try to move directly through the circle to the their antipodal position on the other side (see Fig. 2.8). The scenario becomes very crowded when all the agents meet in the middle and swirling behavior occurs.

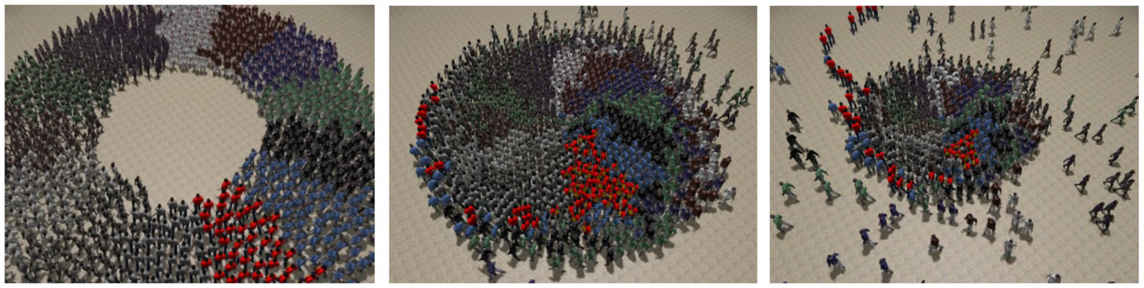


Figure 2.8: **Dense Circle Scenario**: $1,000$ agents are arranged uniformly around a circle and move towards their antipodal position. This simulation runs at over 1,000 FPS on an Intel 3.14 GHz quad core, and over 8,000 FPS on 32 Larrabee cores.

4-Streams: $2,000$ agents are organized as four streams that walk along the diagonals of the square. This is similar to the benchmark in Continuum Crowds (Treuille et al., 2006), though ClearPath results in different behaviors, including smooth motion, lane formation and some swirls.

Back&Forth: Between 10 and 100 agents are asked to move back and forth along a line. This is test is run side-by-side with OpenSteer to compare the number collision with unmodified OpenSteer to the number of collisions with OpenSteer combined with ClearPath. With both techniques agents generally smoothly avoid each other. However, when the ClearPath local collision avoidance algorithm is added, there are no longer any penetrations between the agents. With OpenSteer alone there are a significant number of collisions.

2.5.2 Complex Scenarios

Larger, more realistic scenarios were used to evaluate the performance of the overall system. All of these demos have a large, complex set of obstacles, with roadmaps to guide agents to their goal.

Building Evacuation: Agents were given initial positions of different rooms in an office building. The scene has 218 obstacles and the roadmap consists of 429 nodes and 7.2K edges. The agents move towards the goal positions corresponding to the exit signs. The hallway quickly fills up with the agents and there is congestion at the exits, which allow for only $1-2$ agents to leave at a time. Three versions of this scenario are used with 500, 1K and 5K agents and they are denoted as Evac-500, Evac-$1K$, and Evac-$5K$, respectively.

Stadium Scene: The scenario was a simulation of 25K agents as they exited from their seats out of a stadium. The scene has approximately 1.4K obstacles and the roadmap consists of almost 2K nodes and 3.2K edges. The agents moves towards the corridors and produce congestion and highly-packed scenarios. This benchmark is denoted as Stad-$25K$.

City Simulation: This scenarios uses a city model with buildings and streets with 1.5K obstacles. The roadmap has 480 nodes and 916 edges. Different agents are simulated as they walk around the city streets. The agents move at different speeds and overtake each other and avoid collisions with oncoming agents. Three versions with 10K, 100K and 250K agents were performed and denote as City-$10K$, City-$100K$ and City-$250K$, respectively.

2.5.3 Parallel Implementation

ClearPath can be easily parallelized across multiple agents since the computation performed by each agent is local and independent of the other agents. Specifically, performance on two kind of parallel processors and systems with different characteristics were tested. They are:

1. Multi-core Xeon processors: The algorithms were tested on a PC workstation with a Intel quad-core Xeon processor (X5460) running at 3.16 GHz with 32KB L1 and 12MB L2 cache. Each core runs a single thread. There is no support for gather/scatter operations.

2. Many-core Larrabee simulator: Larrabee (Seiler et al., 2008) is an x86-based many-core processor architecture based on small in-order cores that uniquely combines full programmability of today's general-purpose CPU architecture with compute-throughput and memory bandwidth capabilities similar to modern GPU architectures. Larrabee processor core consists of a vector unit (VPU) together with a scalar unit augmented with 4-way multi-threading, and the VPU supports 16 32-bit float or integer operations per clock. Each core has 32KB L1 data cache, and 256KB L2 cache. Hardware support for masking, gather/scatter, and pack instructions allows a substantial amount of fine-grained parallelism to be exploited by P-ClearPath.

2.5.3.1 Data-Parallelism

Figure 2.7 shows the improvement due to SSE instructions for P-ClearPath on Xeon processors. There is only about $25 - 50\%$ speedup with SSE instructions as Xeon processors do not support scatter and gather instructions, have limited support for incoherent branches.

Figure 2.9 shows the performance of P-ClearPath on SSE and Larrabee architectures. For Larrabee, the performance is measured in terms of Larrabee units. A Larrabee unit is defined to be one Larrabee core running at 1 GHz. The reported performance data is derived from performance simulation of complete multithreaded execution that takes into account various execution stalls arising from instruction dependency, resource contention, and cache/memory misses.

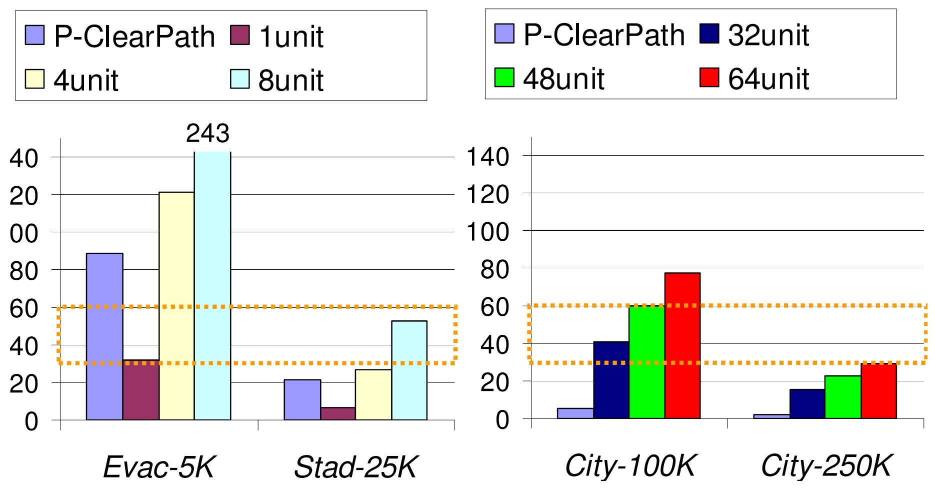


Figure 2.9: Performance (FPS) of P-ClearPath on SSE (left most column) and Larrabee (with different units) architectures. Even on complex scenes, P-ClearPath achieves $30 - 60$ FPS on many-core processors.

2.5.3.2 Thread-level Parallelism (TLP)

One of the issues that affects scaling is the load balancing amongst different threads. Some of the agents in a dense scenarios may perform more computations than the ones in sparse regions, as they consider more neighbors within the optimization computation. Hence, a static partitioning of agents amongst the threads may suffer from severe load balance problems, especially in simulations with few number of agents for large number of threads. The main reason is that the agents assigned to some specific thread(s) may finish their computation early, while the remaining ones are still performing the collision avoidance and other computations.

ClearPath reduces the load imbalance by using a scheme based on dynamic partitioning of agents. Specifically, Task Queueing (Mohr et al., 1990) is used to decompose the execution into parallel tasks consisting of a small number of agents. This allows the runtime system to schedule the tasks on different cores. In practice, the scaling improves by more than 2X as compared to static partitioning for 16 threads. This speedup occurs in small game like scenarios with tens or hundreds of agents. By exploiting TLP, P-ClearPath achieves around 3.8X parallel speedup on the quad-core X5460 over all benchmarks.

Additionally, the combined benefits of TLP and data-level parallelism were evaluated. Running 4 threads per core, the working set fits in L1 data cache and the implementation is not sensitive to memory latency issues. 16-wide SIMD provides around 4X scaling over scalar code. Hardware gather/scatter provides 50% additional speedup, resulting in 4.5X scaling. The pack instructions provide additional 2X scaling to achieve the overall 6.4X scaling.

2.5.3.3 SIMD Scalability

ClearPath Steps	% Time Breakdown	SIMD Scaling
Step 1	14%	7.9X
Step 2	39%	5.6X
Step 3	21%	5.9X
Step 4	13%	9.3X
Step 5	13%	5.6X
Total	100%	6.4X

Table 2.1: Average time (%) spent in various steps (Section 3.3) of Clearpath and the corresponding SIMD scalability on Larrabee simulator of execution cycles as compared to the scalar version.

Table 2.1 shows the SIMD scaling achieved by each of the first five steps of ClearPath described in Section 3.3. The table provides a breakdown of the time spent in that part and the overall speedup. By performing repeated packing and memory gather operations during these steps to negative performance effects of path divergence and early termination of the agent cone intersection and inside/outside computation can be reduced. This improves the SIMD utilization but the performance improvement is limited by the overhead of theseoperations.

2.6 Analysis and Comparisons

2.6.1 Performance Analysis

A key issue for many interactive applications is the fraction of processor cycles that are actually spent in collision avoidance and multi-agent simulation. Note that collision avoidance can take a high fraction of frame time, especially when dealing with dense scenarios with a high number of agents. The left-hand side graph in Fig. 2.9 highlights the performance on simulations with 5K and 25K agents. P-ClearPath achieves real-time simulation rates of $30 - 60$FPS with only one Larrabee unit of computation. Using 64 Larrabee units, this would take up less than 2$\%$ of the total computation time and the rest of the remaining 98$\%$ time could be used for AI, Physics, behavior, rendering and related computations. Even on a quad-core Xeon CPU, P-ClearPath takes up only 20$\%$ of the available computation time for 5K agents. As a result, P-ClearPath running on a commodity many-core processor may be fast enough for game-like scenes. The right-hand side graph in Fig. 2.9 highlights the scalability of ClearPath on extremely large models. For an urban simulation with 100K to 250K agents, real-time rates are achieved with 32-64 Larrabee units.

2.6.2 Behavioral Analysis

Evaluating behavior of the simulation can be difficult, as there is no clear standard for the "right way" to avoid collisions, especially among a high number of agents in a dense scenario. However qualitatively ClearPath results in smooth collision-free motion. Quantitatively, the number of times agents are colliding can be directly measured. In the 80 agent version of the **Back&Forth** benchmark, OpenSteer agents were colliding with each other nearly 16 times per frame. In contrast, when ClearPath was added there were no collisions among the agents over the entire simulation (Fig. 2.10).

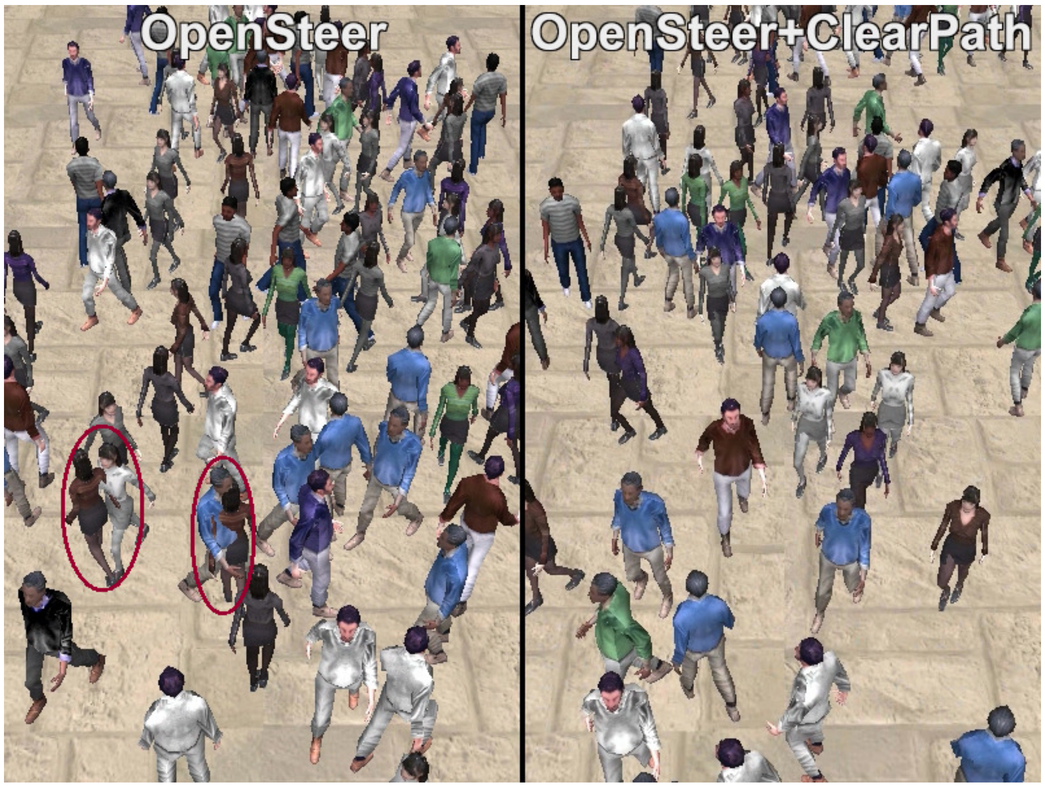


Figure 2.10: A snapshot of the 80 agent Back&Forth demo. Opensteer (on the left) averages 16 collisions per frame. When combined with ClearPath (on the right) the simulation is collision-free. Two particularly egregious collision are circled in red.

As the density increases OpenSteer may result in more collisions, whereas simply using ClearPath or adding it to OpenSteer results in virtually no collisions regardless of the density of the agents (Fig. 2.11).

2.6.3 Comparison and Limitations

Previously, the performance of ClearPath's serial implementation was compared other approaches. Here, crowd systems that use the parallel capabilities of GPUs or multiple CPUs are compared.

The parallel implementation P-ClearPath can handle heterogeneous agents, global navigation and support collision response between the agents. Some earlier algorithms also offered similar capabilities. These include Parallel-SFM (Quinn et al., 2003), which is an

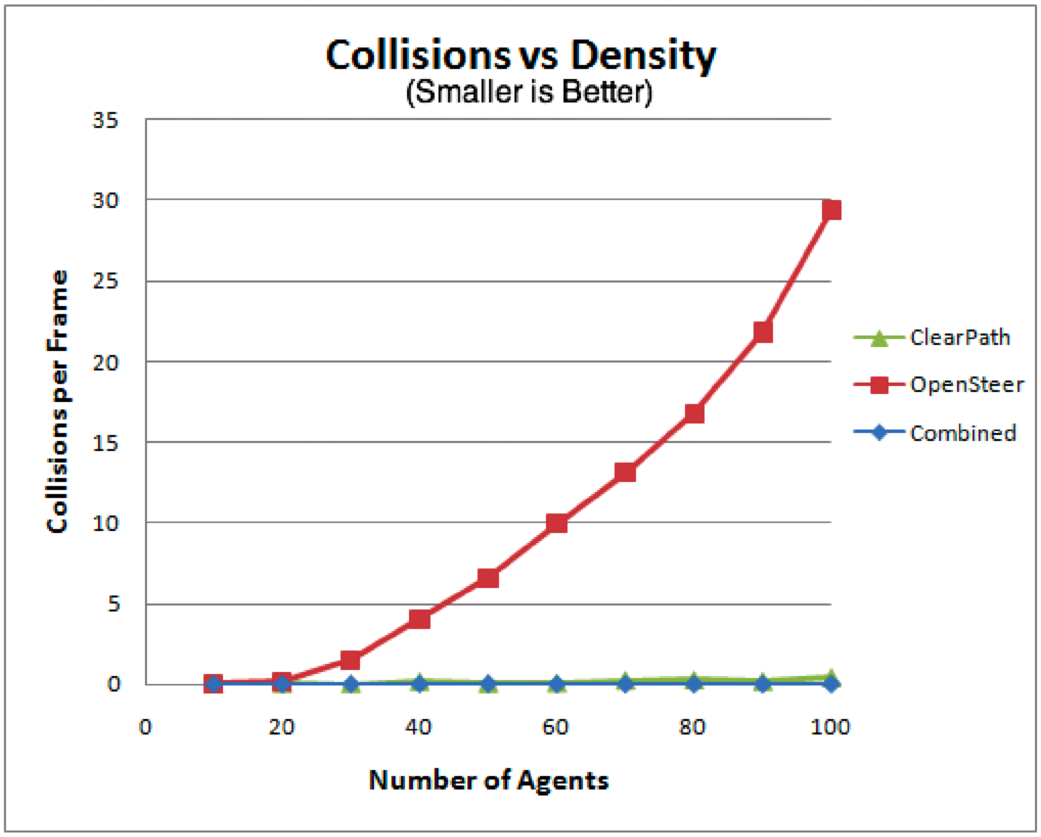


Figure 2.11: As density increases, OpenSteer's collision rate increase super-linearly. Simply using ClearPath on it's own, and combining OpenSteer with ClearPath results in practically no collisions.

implementation based on a social force model that parallelizes the simulation process over 11 PCs and used for simulations with thousands of agents. A multi-core implementation of RVO-Library on a 16 Core system is described in (van den Berg et al., 2008a). Sud et al. (Sud et al., 2007a) used GPU-based discretized Voronoi diagrams for multi-agent navigation (MANG), but this approach doesn't scale to large number of agents. It is hard to make direct comparisons with Parallel-SFM and MANG, as they have very separate behavior than ClearPath. However, these papers report significantly slower performance numbers than ClearPath.

There are other approaches that can handle some complex scenarios. But it is hard to make direct comparison with them because some of the underlying features of these approaches are different. FastCrowd (Courty and Musse, 2004) is an implementation of a similar social force model on a single GPU, but it doesn't include collision response. PSCrowd (Reynolds, 2006) implements a simple flocking model on a Cell Processor with 6 SPUs, but provides only limited collision avoidance. Continuum Crowds (Treuille et al., 2006) is designed to handle homogeneous groups of large agents. It has impressive performance numbers on a single CPU for these large homogeneous groups, but is inappropriate for heterogeneous crowds.

2.7 Conclusions and Future Work

This chapter has presented a robust algorithm for collision avoidance among multiple agents. The approach is general and works well on many different complex multi-agent simulations, including those with tightly-packed and dense scenarios. The algorithm is almost one order of magnitude faster than prior VO-based approaches. Moreover, the algorithm can be combined with other crowd simulation systems and used to generate collision-free motion for each agent (e.g. OpenSteer). I described a parallel extension using data-parallelism and thread-level parallelism and use that for real-time collision avoidance in complex scenarios with hundreds of thousands of agents.

2.7.1 Limitations

ClearPath has some limitations. The FVO constraints highlighted in Section 3.2 are conservative. It is possible that there is a collision free path for the agents, but the algorithm may not be able to compute it. The data parallel algorithm has room for improvement, and may be able to obtain up to 50% improvement as a function of SIMD width and the performance varies based on cache size and memory bandwidth. Lastly, ClearPath has a very simplified model of an agent: a single disk with no holomonic motion restrictions. This model is overly simplistic for many applications.

2.7.2 Future Work

There are many avenues for future work. One important thing to try would be porting P-ClearPath to a many-core GPUs and evaluate its runtime performance. Another interesting extension would be to incorporate kinematic motion constraints related with ClearPath, which can be important in robotic systems and human character animation.

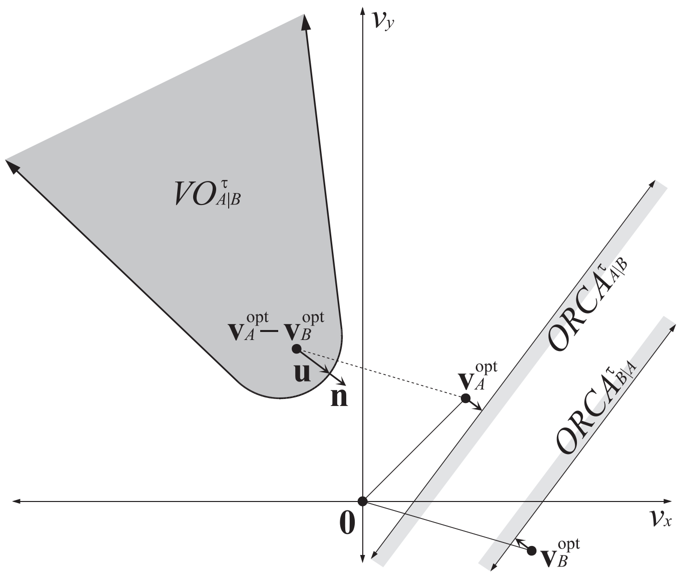
Important improvements such as incorporating human dynamics, personal differences, and evaluating simulation accuracy with respect to real-world data will be discussed later in this dissertation.

2.8 ORCA

As mentioned in the introduction I worked with others to produce a system which runs faster than ClearPath with very similar behavior (van den Berg et al., 2011). The resulting system, known as ORCA, approximates the ClearPath FVO with a linear constraint in velocity-space (see Figure 2.12). By using only linear constraints, the convex optimization described in Section 2.3.3 can be replaced by simple linear programing. The resulting algorithm has a 2-5X speed up over ClearPath depending on the simulation.

ORCA is well suited for large scale simulations such as the Tawaf simulation pictured below (Figure 2.13). Despite involving 25,000 agents ORCA is still able to run the simulation in real-time.

Subsequent results in this dissertation will build of the faster ORCA formulation rather than ClearPath.



v_y
$VO^{\tau}_{A|B}$
$\mathbf{v}_A^{\text{opt}} - \mathbf{v}_B^{\text{opt}}$
$\mathbf{u}$
$\mathbf{n}$
$\mathbf{v}_A^{\text{opt}}$
$ORCA^{\tau}_{A|B}$
$ORCA^{\tau}_{B|A}$
$\mathbf{0}$
v_x
$\mathbf{v}_B^{\text{opt}}$

Figure 2.12: Constructing the set of ORCA allowed velocities. $\mathbf{v}^{\text{opt}}$ is the agent's current velocity. ORCA forces agents to choose new velocities which avoid at least half the collision $\mathbf{u}$. In RCAP, only agents whose $T^{B|A}_{sight} < T - T_{obs}$ generate $ORCA$ constraints.

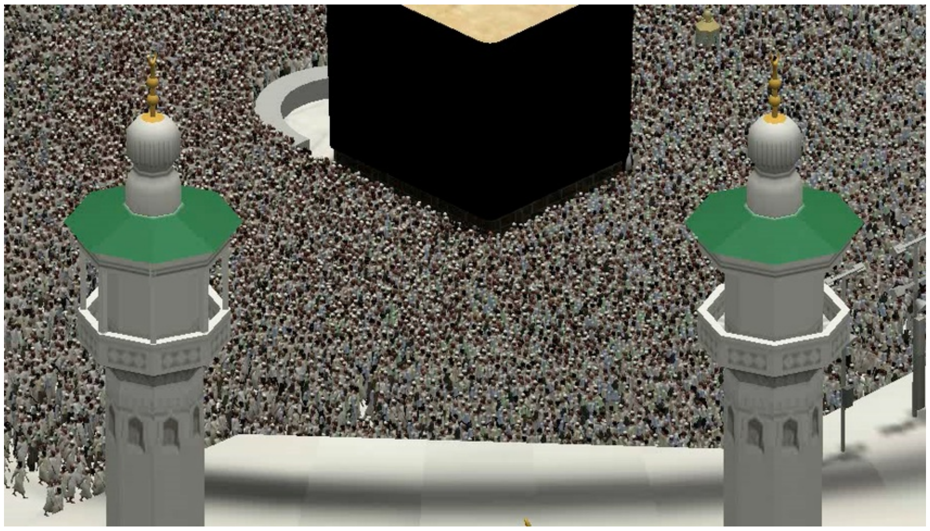


Figure 2.13: Still from a real-time, ORCA simulation of Tawaf ritual involving 25,000 agents.

CHAPTER 3

PLEdestrians: A Caloric-Minimization Approach to Crowd Simulation

3.1 Introduction

The main role of collision avoidance techniques such as ClearPath or ORCA is to tell agents where they cannot go. An interesting question still is where to go. Specifically, ORCA and ClearPath will generally leave a large range of allowed velocities but only one will be taken at any timestep. Which velocity is chosen can sometimes be largely a matter of personal preference, but often there are simple, general guidelines which can naturally describe typical motion. This chapter explores one such guideline known as the Principle of Least Effort (PLE) for it's analogous role to the principle of least action from the field of physics.

As a guiding principle for understanding human motion PLE dates back to at least the 1940s with Zipf's classic book on human behavior (Zipf, 1965). The principle suggests that people will naturally achieve their goals in a manner which uses the least amount of effort possible. The least-effort principle has influenced the design of recent crowd modeling systems which tend to find paths which minimize an agent's travel time, travel distance, congestion, acceleration or other factor which can be indirectly related to the concept of effort.

In this chapter, I propose to directly measure an agent's effort as it's expected caloric expenditure. Minimizing caloric expenditure precedent from the field of biomechanics, an area which extensively studies human motion. For example, (Inman et al., 1981) uses the minimum calorie principle to successfully explain human locomotion patterns. Likewise, minimum calorie analysis can correctly predict the approximate speed which individuals generally travel (Whittle, 2002) and is valid across a wide range of body types (Browning and Kram, 2005).

3.1.1 Main Result

This chapter describes a unified system for simulating human trajectories based on a biomechancially inspired implementation of the principle of least effort. The heuristic of minimizing the total amount of calories consumed, combined with ORCA for collision avoidance, can completely specify a crowd simulation. I will show how to use PLE to inform both low-level collision avoidance, and high level path selection. The resulting system is shown to generate more efficient trajectories than other recent crowd simulation work.

3.1.2 Organization

The rest of this chapter is organized as follows. Section 3.2 gives an overview of some related work. Section 3.3 describes the mathematical foundation of the PLE model. Section 3.4 details how to implement PLE as part of a full crowd simulation. Sections 3.5 and 3.6 analyze the resulting simulations in terms of fulfilling the principle of least effort and qualitative behaviors. Section 3.7 proves a summary of the work and outlines some future directions.

Chapter 4 continues the analysis of the PLE approach to crowd simulation and focuses on measuring its accuracy at capturing actual human motion.

3.2 Previous Work

There is extensive literature on simulating crowd dynamics and distributed motion planning, the surveys of (Pelechano et al., 2008b) and (LaValle, 2006) cover these areas respectively. Recent work has included attempts to generate group behaviors (Bayazit et al., 2002; Kamphuis and Overmars, 2004) and and support real-time navigation of large numbers of agents (Gayle et al., 2009).

3.2.1 Least Effort in Crowd Simulation

Crowd simulation has often been formulated as a problem of minimizing some metric a for group of independent agents. Techniques such as distance based roadmaps, help agents find minimum distance paths to reach a goal. Recent methods have focused on minimizing various effort functions directly inspired by PLE by using cellular automata (Still, 2000; Sarmady et al., 2010) or phenomenological forces (Kagarlis, 2002). These methods aim to capture the large scale patterns of movement, and are not well suited for animations, which require high quality, smooth, collision-free motion.

Several techniques have been proposed specifically for animating large crowds. There is extensive work in this area and, at a broad level, many of them can be classified into five main categories: *potential-based* which focus on modeling agents as particles with potentials and forces(Helbing and Molnar, 1995; Karamouzas et al., 2009), *boid-like* approaches based on the seminal work of Reynolds which create simple rules for velocities (Reynolds, 1987, 1999), *geometric* which compute collision-free paths using sampling (van den Berg et al., 2008a) or optimization (Guy et al., 2009), and *field based* which either compute fields for agents to follow (Yersin et al., 2005; Pettré et al., 2009; Chenney, 2004; Jin et al., 2008), or by generate fields based on continuum theories of flows (Treuille et al., 2006) or fluid models (Narain et al., 2009).

Additionally, other approaches for crowd simulation are based on cognitive modeling and behavioral (Shao and Terzopoulos, 2005; Yu and Terzopoulos, 2007) or sociological or psychological factors (Pelechano et al., 2007).

3.2.2 Motion Synthesis

Related to the Principle of Least Effort, the Principle of Minimum Energy governs the behaviors of many dynamic systems. In fact, energy minimization techniques have been extensively used for character animation and to synthesize motions like walking, running etc. (Kang et al., 1998; Juang, 1999). Gait generation algorithms have also been proposed to minimize energy consumption (Channon et al., 1992; Roussel et al., 1998).

3.3 PLE Model

The Principle of Least Efforts (PLE) was summarized by Zipf with the following observation. "an organism will expend the *least average rate probable of work* as estimated by itself." (Zipf, 1965) The basic idea is that living beings will naturally choose the path to their goal which they expect will require the least amount of "effort". This concept has been used in many domains, such as analyzing traffic patterns (Masucci et al., 2009) and has been observed directly in human walking, which occurs in a manner that minimizes metabolic energy (Inman et al., 1981). More recently, Still (Still, 2000) illustrated numerous cases and data that exhibit crowd dynamics and behaviors which appear to follow the PLE model. Inspired by these findings, this chapter presents a simple yet effective mathematical model for PLE to compute energy-efficient trajectories for each agent in a virtual crowd.

3.3.1 Notation and Overview

Let the simulated environment consist of N heterogeneous agents and optionally contain static and dynamic obstacles. Each agent (A) has a current position ($\mathbf{p}_A$), and a goal position ($\mathbf{G}_A$), both viewed as input. Each agent and obstacle is represented using a circle or polygon in the plane. Each agent has an independent radius (r_A) and velocity ($\mathbf{v}_A$). The goal position may change dynamically during the course of the simulation. While the PLE approach can extend to agents moving in 3D space, the chapter assume agents are moving on a 2D plane. For any vector $\mathbf{n}$, let $\hat{\mathbf{n}}$ denotes a unit vector along $\mathbf{n}$, and $|\mathbf{n}|$ denotes the magnitude of the vector $\mathbf{n}$.

The overall simulation proceeds in discrete time steps, and the position and velocity of each agent are updated at every step. At each time step, the agent uses its current position, goal position, and information about it's neighbors to computes a new velocity for the time step. The PLE algorithm uses a local collision avoidance method (van den Berg et al., 2011) that computes a range of permissible velocities (denoted PV_A) for each agent at each time step. The PV_A is computed by taking into account the position and velocity of other nearby agents and obstacles. The algorithm chooses a velocity from among those allowed by PV_A which will minimize the expected effort to reach to goal.

3.3.2 Least Effort Function

As noted by Still, key aspects of human behavior arise form the principle of least effort (Still, 2000). Specifically, individuals or agents should:

1. Take the shortest available routes to their destinations.
2. Attempt to move at their preferred speed.

By choosing an appropriate function to represent "effort", the underlying mathematical formulation for PLE should be able to model these behaviors. A simple effort function that minimizes the distance to reach the goal does not address the influence of speed. Similarly, a metric that only minimizes the time to reach the goal will result in the agents walking at their maximum speed rather than at their preferred speed, expending more energy than necessary. This chapter present a novel metric to model PLE based on biomechanical principle: minimize the *total biomechanical energy* expended by an individual during locomotion, measured in Joules (J).

The measurement of biomechanical energy expended by an agent is derived from prior experiments of subjects walking on treadmills at various speeds (Whittle, 2002). By measuring the oxygen consumed, the instantaneous power (P) spent by the subjects walking can be modeled as a function of the underlying speed:

$$P = e_s + e_w|\mathbf{v}|^2, \tag{3.1}$$

where $\mathbf{v}$ is the instantaneous velocity, and e_s (measured in J/Kg/s) and e_w (measured in Js/Kg/m^2) are per-agent constants[1]. This formulation is adapted to model the total effort for each person as the *total metabolic energy* expended while walking along a path, that is:

$$E = m \int (e_s + e_w |\mathbf{v}|^2) dt, \tag{3.2}$$

where m is the mass of the person. The PLE trajectory computation algorithm aims to minimize this function for each agent.

I now present an important lemma regarding the trajectories which will minimize this proposed effort function. In the absence of dynamic obstacles, it can be shown that (proof given in Appendix):

Lemma 1: *The total effort (Eqn. 3.2) spent while walking to a goal is minimized by an agent moving at a constant speed of* $\sqrt{(e_s/e_w)}$ *along the shortest path to the goal.*

It is important to note that $\sqrt{e_s/e_w}$ = $\sqrt{2.23/1.26}$ = 1.33 m/s, which is the average walking speed for humans in an unconstrained environment (Whittle, 2002). Because of this, Lemma 1 highlights the strong connection between PLEdestrians and real human motion. As a Corollary, it is possible to compute the minimum possible effort to travel a given distance.

Corollary 1: *For a path of length* L*, the minimum amount of effort expended by a person of mass* m *to traverse it is* $2mL\sqrt{e_s e_w}$.

Lemma 1 and its corollary highlight that the proposed effort function matches the above two criteria: the metric is minimized by agents taking the shortest path at natural human walking speed.

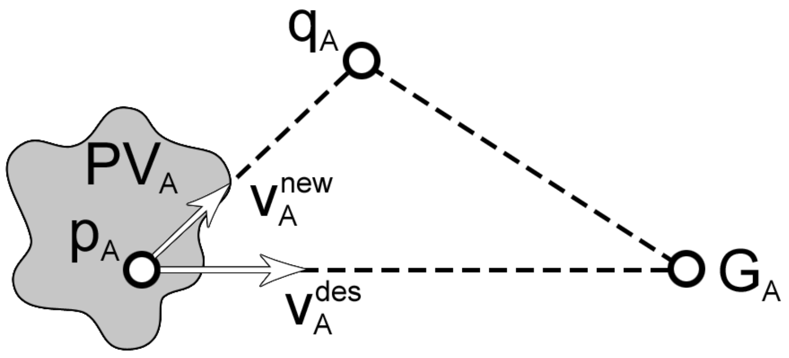


Figure 3.1: The algorithm computes the new velocity ($\mathbf{v}_A^{new}$) for moving from $\mathbf{p}_A$ to $\mathbf{G}_A$ in accordance with PLE. The effort function is analytically minimized over all possible intermediate positions $\mathbf{q}_A$, such that the the agent expends least amount of effort to reach its goal.

3.3.3 Mathematical Model for Effort Minimization

Given the effort function, it is possible to reduce the problem of crowd simulation governed by the PLE to an optimization problem. For any agent A, find the trajectory which minimizes the total biomechanical energy expended while moving from its current position to its goal, and move the agent by the corresponding velocity at the start of that trajectory.

Consider Fig. 3.1, showing an agent at $\mathbf{p}_A$ with a goal of $\mathbf{G}_A$. Given a set of velocities PV_A, which will not cause any near-term collisions, a new velocity must be chosen from this set which will minimize the expected biomechanical effort. To evaluate a potential new velocity, $\mathbf{v}_A^{new}$, for this time step it is necessary to estimate how much effort a path starting with $\mathbf{v}_A^{new}$ would take. Any potential $\mathbf{v}_A^{new}$ is assumed to remain approximately constant for the next τ seconds, and the resulting position of following that velocity is denoted as $\mathbf{q}_A$. The effort expended for moving from $\mathbf{p}_A$ to $\mathbf{G}_A$ can be decomposed into the sum of energy expended for traversing from $\mathbf{p}_A$ to $\mathbf{q}_A$ and energy for going from $\mathbf{q}_A$ to $\mathbf{G}_A$. Minimizing the total effort E reduces to solving:

$$\textbf{Minimize} \;\; m\tau(e_s + e_w|\mathbf{v}_A^{new}|^2) + E_{\mathbf{q}_A\mathbf{G}_A}, s.t. \mathbf{v}_A^{new} \in PV_A. \tag{3.3}$$

[1] $e_s = 2.23\frac{\mathrm{J}}{\mathrm{Kg\ s}}$ and $e_w = 1.26\frac{\mathrm{Js}}{\mathrm{Kg\ m^2}}$ for an average human (Whittle, 2002)

Complexity of optimal solution: Finding the optimum value of $E_{\mathbf{q}_A \mathbf{G}_A}$ (*e.g.* through recessive approximation refinement) would be computationally prohibitive. In fact, the goal of computing a globally optimal solution to the effort function, can be reduced to the problem of computing an optimal path for multiple robots in the plane. The complexity of such motion planning problems tend to increase as an exponential function of the number of robots or the total number of degrees of freedom (LaValle, 2006). As a result, the complexity of computing a globally optimal path for each agent that minimizes the effort function (Eqn. 3.3) would have exponential complexity in the number of agents.

Instead, the following greedy local approach is used to compute an approximately optimal velocity individually for each agent.

Greedy Approach: Referring back to Eqn. 3.3, it is necessary to evaluate $E_{\mathbf{q_A G_A}}$ to compute the total expected energy consumption. Rather than evaluating it exactly a greedy heuristic is used, replacing $E_{\mathbf{q}_A \mathbf{G}_A}$ with the minimum possible amount of effort required to traverse from $\mathbf{q}_A$ to $\mathbf{G}_A$ as provided by Corollary 1. As a result, assuming L as the distance from $\mathbf{q}_A$ to the goal, the effort function can be given as: $E = m\tau(e_s + e_w|\mathbf{v}_A^{new}|^2) + 2mL\sqrt{e_s e_w}$, with the resulting optimization formulation being:

$$\textbf{Minimize}\, \tau(e_s + e_w|\mathbf{v}_A^{new}|^2) + 2|\mathbf{G}_A - \mathbf{p}_A - \tau\mathbf{v}_A^{new}|\sqrt{e_s e_w},\ s.t.\ \mathbf{v}_A^{new} \in PV_A. \quad (3.4)$$

This objective function is convex, with one global minimum (proof in Appendix). Optimizing over this functions returns a new velocity, $\mathbf{v}_A^{new}$, to be undertaken by agent A for this time step.

3.3.4 Properties of the PLE Metric

When minimizing energy using a local, greedy approach (Eqn. 3.4), agents will move along smooth paths, and expend energy within a small bound of the minima. This arises

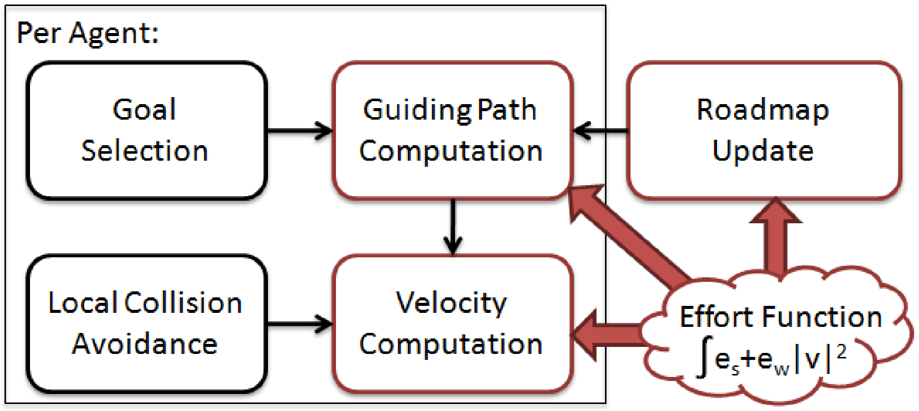


Figure 3.2: **Multi-agent navigation**: An overview of the proposed approach for computing the trajectory for each agent. Each agent performs these computations at each time step. The roadmap used for navigation is also updated. The effort function shown in Eq. 3.4 is used by the optimization algorithm for velocity computation.

from key properties of the metric relating to smoothness and accuracy. The most important smoothness property is captured in the following lemma, which holds for a fixed goal.

Lemma 2: *The trajectories traversed by the agents using the PLEdestrians are C1 continuous (if allowed by the PV.)*

A detailed proof is given in Appendix B. Additionally, assuming a bounded period of congestion, bounds for the accuracy of the heuristic can be derived, more details are given in the appendix.

3.4 Trajectory Computation

The section presents a trajectory computation algorithm that uses the PLE function presented in Section 3.3. Given the goal position, this algorithm computes a biomechanically energy-efficient trajectory that avoids collisions with the other agents and obstacles.

3.4.1 Algorithm Overview

Figure 3.2 highlights the various components of the algorithm. First, a global roadmap is precomputed for and used for collision-free navigation around static obstacles. This roadmap is represented as a graph used by each agent to compute a path to its goal position.

During *Goal Selection*, the desired goal for each agent is computed by some high-level crowd simulation algorithm during each time step.

This goal position is used by the *Guiding Path Computation* module to compute a path from each agent's current position to its goal position along the precomputed roadmap. With each edge of the roadmap, a weight is dynamically assigned that is a measure of the biomechanical effort needed to traverse the edge. The edges with slow moving agents indicate congestion and the algorithm will assign them large weights, while edges with little or no congestion will have lower weights. The A* graph search algorithm is used to compute a minimum-energy path to the goal along the roadmap. For efficiency, if the expected energy consumption along the path has not worsened since the last timestep and the goal position has not changed, the path computed during the previous time step can be used again.

The *Local Collision Avoidance* module returns a set of permissible velocities (PV_A) that will be free from collision with all nearby obstacles and agents. This information is used in the *Velocity Computation* step, which computes the velocity which results in the minimal estimated energy to reach the next intermediary node along the guiding path form the roadmap. The geometric algorithm of (van den Berg et al., 2011) is used to compute the permissible region of non-colliding velocities. In this case, PV_A is a convex region and this property is exploited to design an efficient algorithm for solving the optimization problem of computing the new velocity for each agent.

3.4.2 Dynamic Energy Roadmap

After the algorithm has computed a new velocity for each agent, the edges in the roadmap are updated in the *Roadmap Update* module. First the average agent speed along the edge ($|\mathbf{v}^{avg}|$) is computed. This velocity is then used along with Eqn. 3.2 to estimate the total biomechanical energy (per unit mass) that will be spent while crossing the edge, resulting in the following equation (assuming edge length l):

$$E_{link} = (\frac{e_s}{|\mathbf{v}^{avg}|} + e_w|\mathbf{v}^{avg}|)l \tag{3.5}$$

This equation must be updated as $|\mathbf{v}^{avg}|$ changes throughout the simulation. After Eqn. 3.5 is evaluated for each edge, the roadmap weights correspond to the total energy needed to navigate though environment without colliding at the current time step.

3.4.3 Velocity Computation

I will now present an optimization algorithm to compute a velocity in PV_A that minimizes the energy function Eqn. 3.4 (see Figure 3.3).

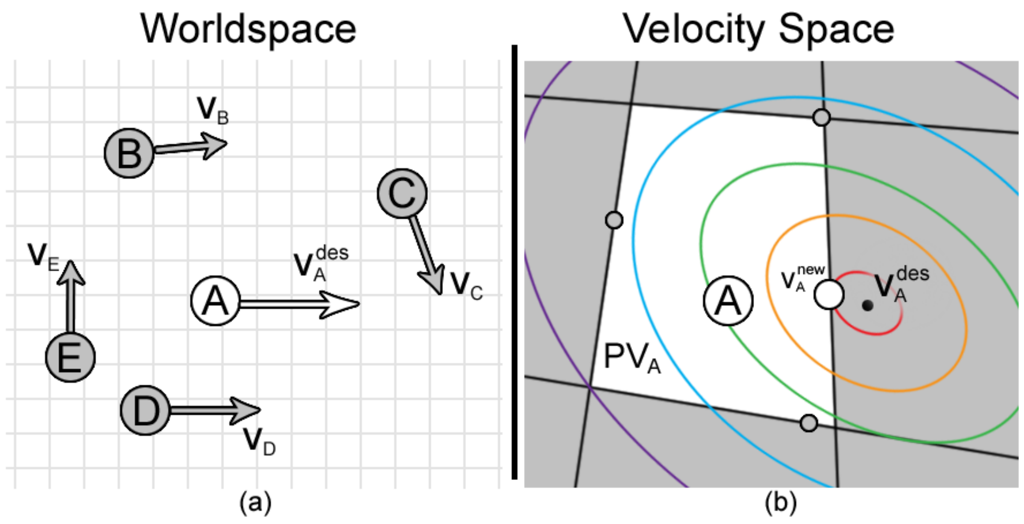


Figure 3.3: Velocity selection - (a) Agent A avoids 4 neighboring agents. (b) The permissible velocities PV_A of agent A is shown in white. Radiating ellipses correspond to iso-contours of the energy function. The circles show the local minima along each line segment, the enlarged white circle being the global minima of the energy function and the new velocity computed for this agent for the next time step, $\mathbf{v}_A^{new}$

Because of the convex shape of PV and the convexity of the objective function (Eqn. 3.4) basic optimization theory suggests the function must be minimized at either:

1. The velocity oriented straight towards to goal at magnitude of $\sqrt{e_s/e_w}$ (denoted $\mathbf{v}_A^{des}$)

   OR

2. A point along the boundary of PV_A

Case 1 can be easily tested for. Case 2 requires finding the optimal point along each line segment of the boundary of PV_A and taking the minimum of the points which optimize energy for a given line segment constraint. Since boundary of PV_A consists of linear segments, I will first describe the algorithm to minimize the energy function on a given line, followed by the minimization over the convex region PV_A.

Energy Minimization along a Line: Let a line be represented using the y-intercept form: $y = mx + e$. The velocity along this line that minimizes eqn. 3.4 can be computed using the following formulation. Let $\mathbf{v}_A^{new}$ be defined as an offset from a vector towards the goal:

$$\mathbf{v}_A^{new} = (\mathbf{G}_A - \mathbf{p}_A)/\tau + \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}. \tag{3.6}$$

Let $(\mathbf{G}_A - \mathbf{p}_A)/\tau$ = $(d_x$, $d_y)$. The magnitude r can be computed by solving the following quartic equation:

$$r^4 + Ar^3 + Br^2 + Cr + D = 0 \tag{3.7}$$

where:

$$A = \sqrt{\frac{4e_s}{e_w}},\ \ B = \frac{e_s}{e_w} + \frac{\tau^{-2}(d_x + md_y)^2(d_y - md_x)^2 - e^2}{(1+m^2)},$$
$$C = \frac{-2\sqrt{e_s}(\frac{d_y}{\tau} - m\frac{d_x}{\tau} - e)^2}{\sqrt{e_w}(1+m^2)},\ \ D = \frac{e_s(\frac{d_y}{\tau} - m\frac{d_x}{\tau} - e)^2}{e_w(1+m^2)}$$

The orientation θ can be computed as:

$$\theta = \arcsin\left(\frac{-m(d_x + md_y)\tau^{-1}}{(r + \sqrt{e_s/e_w})(1+m^2)} - \frac{\frac{d_y}{\tau} - m\frac{d_x}{\tau} - e}{r(1+m^2)}\right) \tag{3.8}$$

Substituting the appropriate root from Eqn. 3.7 and θ from Eqn. 3.8 into Eqn. 3.6 computes optimal velocity along the line.

Energy Minimization for PV_A: The energy function needs to be minimized along all boundary line segments of PV_A. The (expected) linear-time linear programming algorithm from (de Berg et al., 2008) is used which consists of the following steps:

Step 1: Decompose the set of PV_A into line segments ($\mathcal{L}$). The line segments are obtained by intersecting a randomized permutation of the boundary lines with each other. Since the lines form a convex region, the boundary line segments can be obtained in an expected time linear in the number of lines (de Berg et al., 2008).

Step 2: For each $l \in \mathcal{L}$, compute the point along the line segment that minimizes the energy metric, as defined by Eqn. 3.6 to Eqn. 3.8. Note that for any line segment, the minimum point may lie on one of its end points. At the end of Step 2, there will be a set of $|\mathcal{L}|$ points.

Step 3: Return the point computed in Step 2 that evaluates to the minimum total energy in Eq. 3.4. This algorithm runs in $O(n)$ time per agent, where n is the number of neighboring agents and obstacles used to compute the non-colliding constraints.

3.4.4 Clustering-based Optimizations

Recalling that N is the total number of agents in the simulation, our total runtime is $O(Nn)$. The value of n is bounded by the total number of agents N, providing a total runtime of $O(N^2)$. It practice, it possible to only consider the closest neighboring agents during the computation of PV_A. This is sufficient to avoid collisions, but since this approach ignores agents beyond a certain distance, it can potentially lead to increased agent density in certain regions – leading to congestion.

To prevent this effect, and account for all agents in a computationally efficient way, distant agents are *clustered* using kD-trees and use these clusters to compute constraints in the velocity space to prevent the agent from walking towards dense groups of other agents. By computing clusters to be the leaves of the kD-tree at a fixed depth, the total number of such constraints is limited to $logN$, reducing the total run time to $O(NlogN)$.

3.5 Results

The PLE algorithm was implemented in C++ using OpenGL for visualization on a Windows Vista x64 operating system with an Intel i7 965 quad core system with a 3.2GHz processor and 6GB of memory. Each core supports simultaneous multi-threading (SMT) with two hardware threads per core.

3.5.1 Benchmarks

Two kind of benchmarks were used to test the algorithm's performance. The first set of benchmarks were used to test the emergent behaviors and crowd effects. These include:

n-Agent Circle - A small number of agents pass each other walking to antipodal positions of a 10m radius circle.

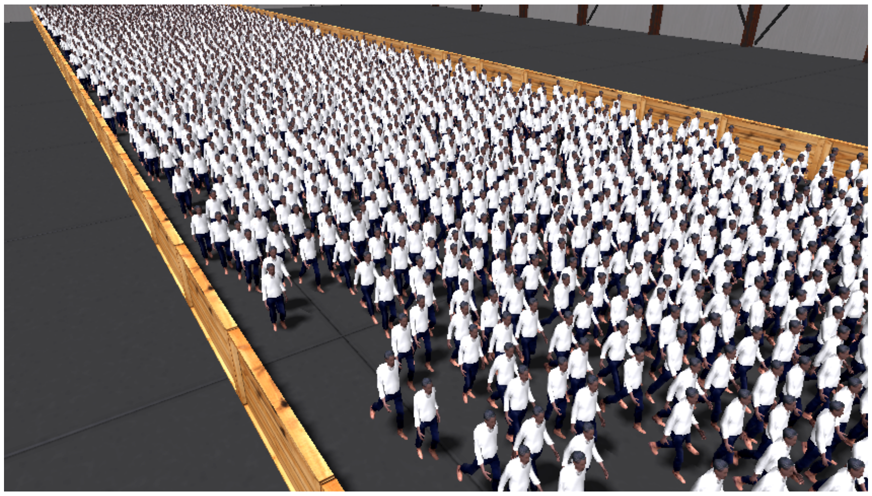


Figure 3.4: Long Corridor Scenario

Long Corridor - Ten thousand agents that fill a corridor that is 300m long. The agents all have a random goal that is located 100m or more south of their initial position. (Fig. 3.4)

Narrow Passage - 100 agents must pass through a narrow passage to reach their goals (Fig. 3.5).

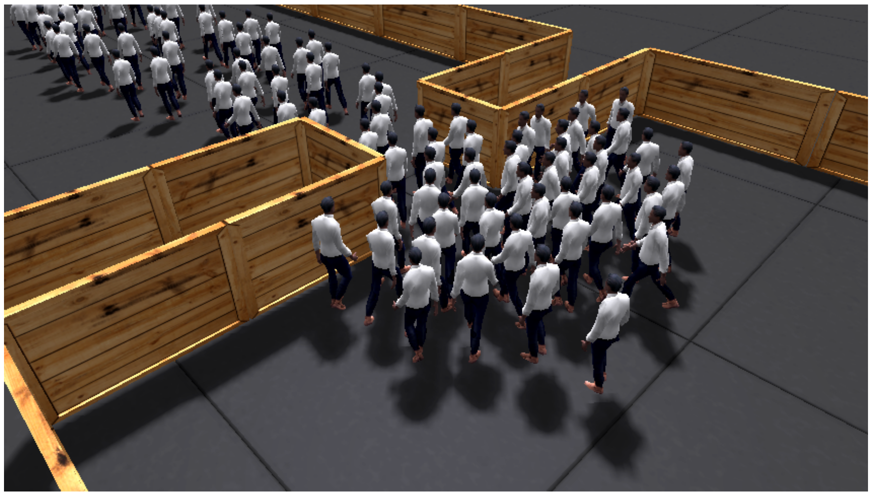


Figure 3.5: Narrow Passage Scenario

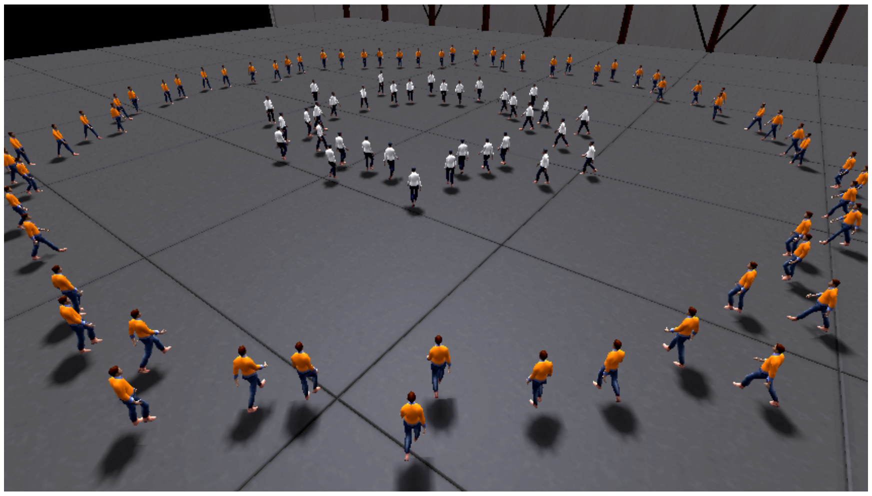


Figure 3.6: Narrow Passage Scenario

Concentric Circles - 100 agents are placed along two concentric circles, 34 in the inner circle and 66 in the outer one. Their goal position is given by the antipodal position on the corresponding circle (Fig. 3.6).

The second set of benchmarks are designed to test realistic scenarios and the overall performance of the algorithm. These include:

Trade-show Floor - A recreation of a typical exhibition floor found at a trade show. All 1000 agents are asked to leave the floor, but given a goal positions corresponding to the exit *furthest away* from their starting position causing them traverse almost the full length of the floor and encouraging potential congestion (Fig. 3.7).

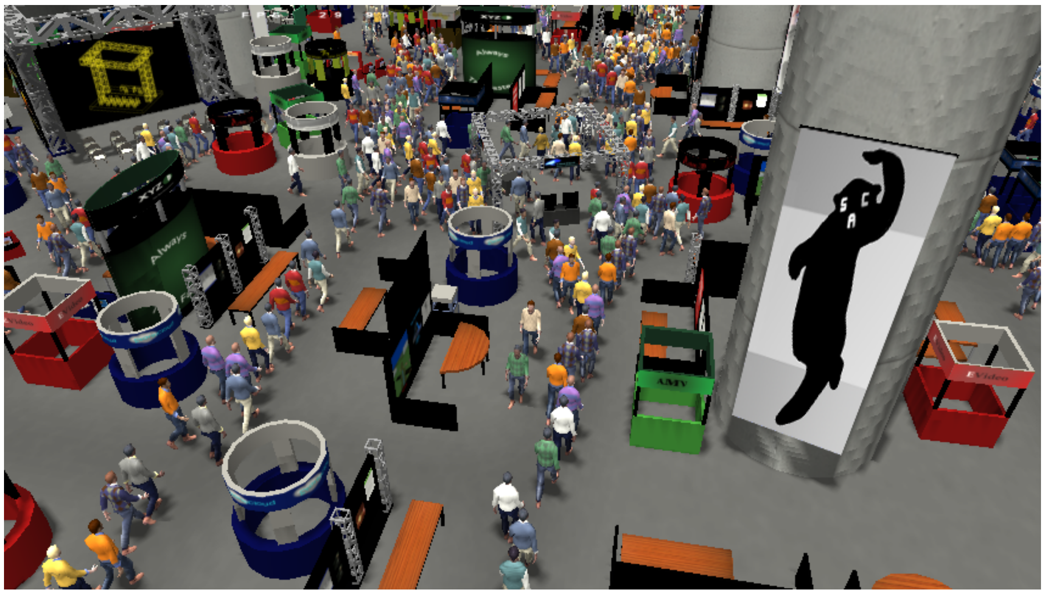


Figure 3.7: A simulation frame from the Trade Show Floor consisting of $1,000$ agents. PLEdestrians computes collision-free trajectories with many emergent behaviors at 31 fps.

Shibuya Crossing - A recreation of the 5-way scramble crossing in front of the Shibuya train station in Tokyo, Japan. Here, 1000 agents cross along the various crossings provided (Fig. 3.8 & 4.10a).

3.5.2 Comparison to Other Methods

Here, the trajectories computed by the PLE algorithm are compared to those generated by other crowd or multi-agent simulation methods. First, for simple scenarios, the results of various crowd simulations are compared against the analytical minimum energy possible. Secondly, various methods to each other numerically in terms of the biomechanical energy consumed.

Four popular simulation techniques which have been proposed for simulating large crowds with hundreds or thousands of agents are chosen for comparison. These methods are: Helbing social force with the tuned parameters suggested in (Helbing et al., 2000); OpenSteer

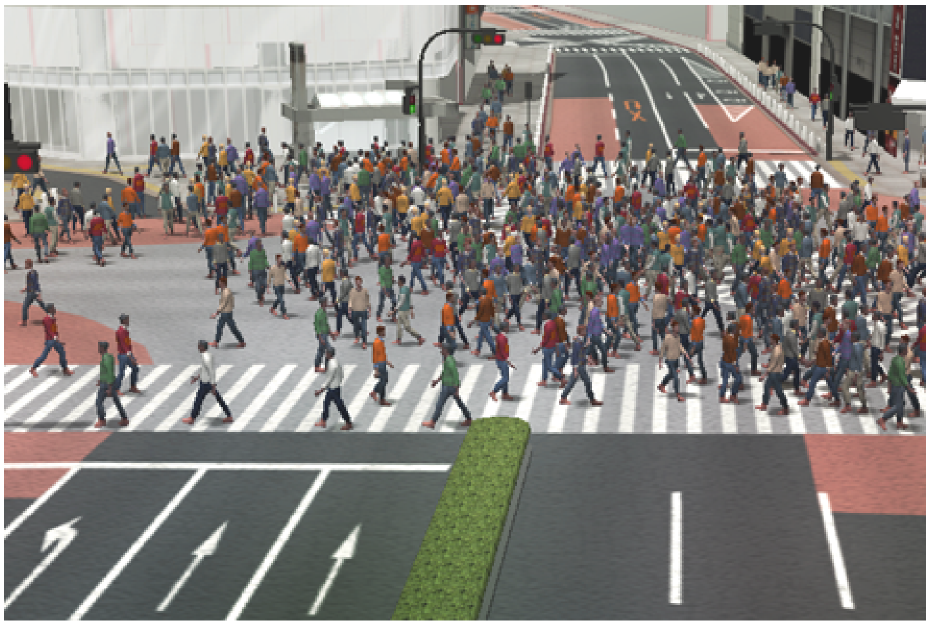


Figure 3.8: Simulation of Shibuya Metro Station

steering based model, which is an extension of Reynold's flocking model (Reynolds, 1999); RVO collision avoidance method that uses sampling (van den Berg et al., 2008a); and the ClearPath collision avoidance method (Guy et al., 2009).

Analytical Comparisons: In order to analyze how well the approximate greedy formulation comes to the true global minimum, the energy required by the simulated paths can be compared to the actually minimum energy required to reach the goal. Least energy, non-colliding trajectories can be found analytically for simple scenarios with few agents. Here I compute the biomechanical energy of two agents swapping position using the various methods, and compare to the analytical minimum. For a small number of agents PLEdestrians performs similarly to RVO and ClearPath in using close to the theoretical minimum amount of energy. Helbing and OpenSteer however use significantly more energy in this scenario, this is to be expected as they are not explicitly optimizing for energy spent. The effect of these inefficiencies is the planned motion is visibly less natural paths as can be seen in Fig. 3.9.

Numerical Comparisons: For more complex scenarios, a true analytical minimum of least effort paths are unknown. However, I can compare the total biomechanical energy

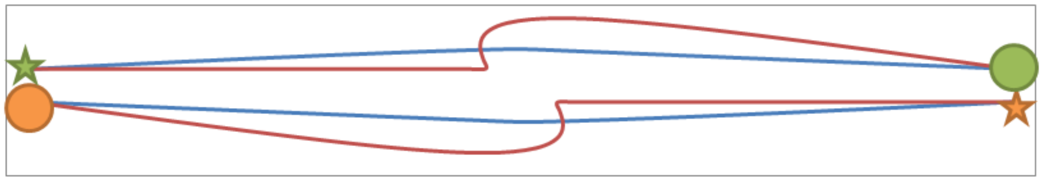


Figure 3.9: **Comparisons of path traced for 2-Agent Crossing:** The figure shows the initial position (star) and final position (circle) for each agent, along with a comparison of the paths traced by PLEdestrians (blue) and Helbing social force algorithm (red). PLEdestrians paths have less deviation and consume less total effort for the agents.

used by PLE to other standard crowd simulation methods. Table 3.1 shows a comparison of energy used in two complex scenarios where the analytical solution is not known: the *10-Agents Circle* demo, and the *Concentric Circles* demo. PLEdestrians produces trajectories which use the least energy in all scenarios, providing support for the acceptability of the local, greedy heuristic proposed in Sec 3.3.3.

	Avg. Energy (J/kg)			**Avg. Time (s)**		
Method	**#1**	**#2**	**#3**	**#1**	**#2**	**#3**
ClearPath	33.4	39.8	315	**7.5**	11.3	124.5
OpenSteer	36.1	43.7	251	8.1	10.5	54*
Helbing	39.3	45.6	211	10.0	14.2	70.5
RVO	33.9	42.1	195	7.6	13.3	64.0
PLE	**33.3**	**35.7**	**183**	**7.5**	**10.4**	**61.7**

Table 3.1: Energy expended and simulated time to complete the benchmark for 2-Agent swapping (#1), 10-Agent Circle (#2) and Concentric Circles (#3). *OpenSteer fails to avoid collisions in the dense regions of this scenario.

3.5.3 Comparison to Data from Crowd Studies

The trajectories computed by PLEdestrians can be compared with prior studies on human and crowd motion.

Quantitative Comparisons: Data has been collected by social scientists on the paths traversed by humans, as they move in crowds. One important analysis is how the humans respond to congestion: as local density increases, the speed decreases. Fruin (Fruin, 1971) collected data of commuters at bus terminals and transit stations in various cities, and produced a numerical curve showing the empirical response. Later studies have examined more data in a variety of circumstances and have suggested the equation $S = k(1 - \alpha\rho)$, where S is the speed of an individual ($\frac{m}{s}$), ρ is the density ($\frac{ppl}{m^2}$) and k and α are constants which vary based on the situation as described by Nelson and Maclennah (Nelson and Maclennan, 1995).

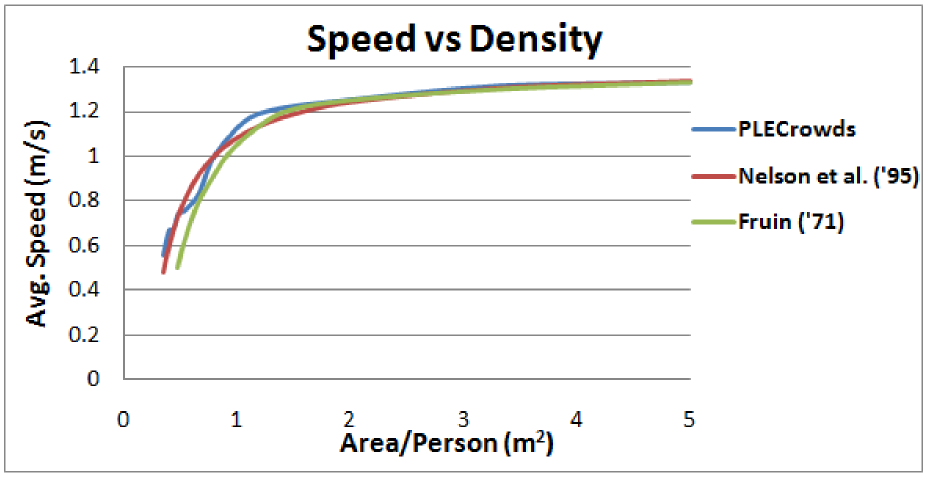


Figure 3.10: **Effect of density on the speed.** This graph compares the results of PLEdestrians with the prior data collected on real humans. The PLE model matches real-world data closely.

Figure 3.10 shows Fruin's original commuter data, as well as Nelson and Maclennah's empirical equation (with k= 1.4 for corridors and α=0.266 for the level floors). The figure also shows data collected from several runs of the PLE simulations at various densities. The simulated results match closely to both Fruin's and Nelson's data.

Emergent behaviors: A different way to evaluate how human-like the motion generated by the PLEdestrian algorithm is by investigating it's ability to generate well-known emergent crowd phenomena which have been reported by social scientists. Below is a list of such phenomena with brief descriptions, all of these occur in both real humans and in the PLEdestrian simulations. In humans, these behaviors have been noted by several researchers such as Still (Still, 2000) and Helbing et al. (Helbing and Molnar, 1995), and in several field studies such as Fruin (Fruin, 1971). Examples from the simulations are shown in the supplemental videos, and are highlighted below.

- Jams/Bottlenecks - congestion form at narrow passages
- Arching - semi-circular arches form at exits
- Lane formation - opposing flows pass through each other
- Swirling - vortices can form in cross flows
- Wake effect - empty space persists behind obstacles
- Uneven densities - regions form with more or less people than the surrounding areas
- Edge effects - agents move faster near the edges of crowds
- Overtaking - fast individuals move past slower neighbors
- Congestion avoidance - individuals tend to avoid overly dense regions if possible

The *Long Corridor* scenario provides a clear demonstration of the *edge effect*. Agents at either edge of the crowd move noticeably faster than other agents in the center. Figure 3.11 shows the average velocities along a cross section of the agents. Agents at the left (0m) and right (25m) are moving 33% faster than those in the center (12.5m). This benchmark also shows agents on the sides *overtaking* those in the centers, and demonstrates the *uneven densities* that can form in crowds.

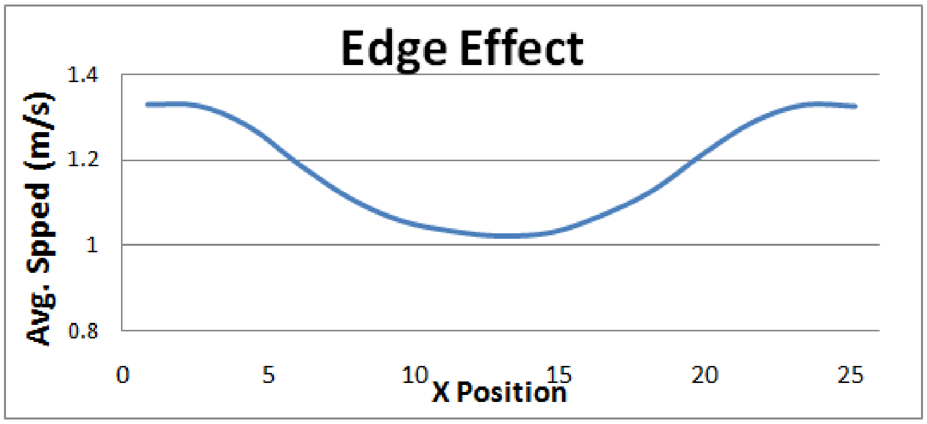


Figure 3.11: **Edge-Effect Phenomena.** A graph of speed vs. a cross section of the PLEdestrian simulation. Agents near the edge of the crowd move faster than those in the center.

The *Narrow Passage* benchmark demonstrates how *jamming and bottlenecks* form at narrow passages. Additionally, the well known *arching effect* is visible as the agents form an arch around the entrance of the corridor. Lastly, the *wake effect* can be seen as people slowly spread out after the narrow passage rather than fill the available space immediately.

The *Concentric Circles*, *Trade-show Floor* and *Shibuya Crossing* benchmarks all demonstrate *congestion avoidance* in various ways. In the *Concentric Circles* scenario, the agents move around the congestion that starts to form in the center. In the *Trade-show Floor* agents plan new paths around the congestion that starts to form in the central passages. In *Shibuya Crossing*, the agents spread out on the crosswalks to avoid congestion. The effects of congestion avoidance can be quantified by examining the energy consumption. Fig. 3.12 shows the biomechanical energy consumed in the *Trade-show Floor* with varying number of agents. The congestion avoidance that arises from the PLEdestrians simulation drastically reduces the average amount of energy consumed per agent.

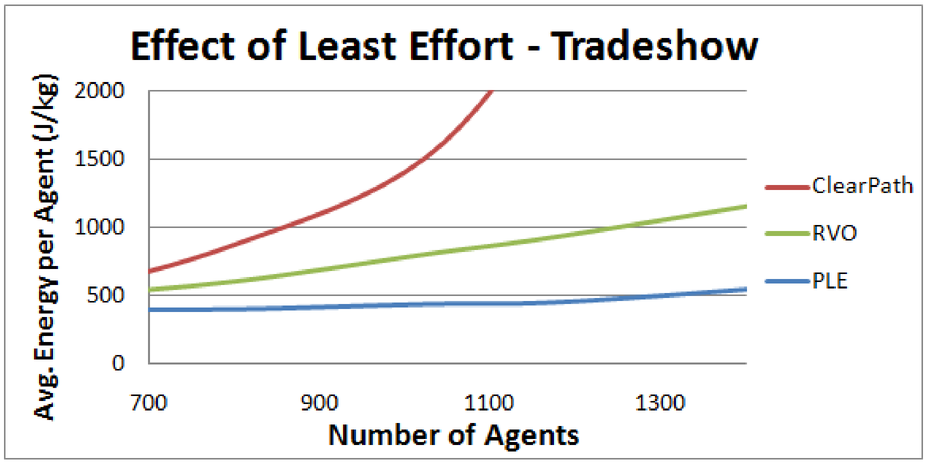


Figure 3.12: **Effect of PLE on Congestion:** A comparison of the effort of each agent in PLEdestrians vs. ClearPath and RVO in the trade-show benchmark. The PLEdestrian approach avoids congestion and there is only a slow increase in the average effort that is needed to reach the goals as the number of agents increases.

3.5.4 Performance Results

I report performance results on the three most complex benchmarks. The *Long Corridor* benchmark has $10,000$ agents and 2 obstacles. The *Trade-show Floor* demo with $1,000$ agents, 500 obstacle segments, and a roadmap with 300 edges. The *Shibuya Crossing* benchmark has $1,000$ agents, 200 obstacle segments, and a roadmap with 70 links. The results are shown in Table 3.2, both for single thread performance, and at full parallel utilization. In all cases the simulations ran at interactive rates.

Benchmark	**Agents**	**FPS - 1 Core**	**FPS - 4 Core**
Long Corridor	10K	15.1	58.9
Shibuya	1K	29.9	113.1
Trade-show	1K	31.0	114.7

Table 3.2: Performance Results for different benchmarks. The algorithm scales almost linearly with the number of cores.

By using clustering techniques, there is as much as a 60x speed up at run-time for a simulation of 1,000 agents. For a given scenario, the total energy used differs by less than 5$\%$.

3.6 Analysis

There are two important criteria to evaluate the PLE work on. First, what the accuracy of our trajectory computation algorithm based on the least effort model proposed in Section 3.3.2 (*i.e.* minimization of the total effort spent per agent). Second, to what extent in the algorithm capable of generating natural behaviors.

In terms of minimizing the effort, the PLEdestrian algorithm performs quantitatively better than other widely used agent-based crowd simulation models. As Table 3.1 and Figure 3.12 show, the total energy expended per agent was much less for PLEdestrians than other widely used approaches. Furthermore, in simple cases with a known theoretical minimum, PLEdestrians comes within 99% of the theoretical minimum energy, validating the greedy optimization heuristic.

The smoothness of the generated trajectories proven by Lemma 2, can be clearly seen in the the paths agents take walking around their neighbors. Paths such as those show in Figure 3.9 and in the demos in the accompanying video highlight the smoothness of the paths generated by PLEdestrians vs. other methods such as social forces.

In terms of matching the behavior of real humans, the an important evaluation is to check how well the algorithm reproduces well-known emergent phenomena seen in real-world crowds. As discussed in Section 3.5 and can be seen in the supplemental videos, the PLEdestrians algorithm reproduces many of these effects. The jamming, arching, lane formation, wake effects, uneven densities, edge effects, overtaking, and vortices were consistently generated by the PLEdestrian algorithm. While other approaches can also reproduce some of these phenomena, seeing them in PLEdestrian simulations suggests that these phenomena are consistent with the PLE hypothesis.

The validation of the PLE model is further strengthened by the match between aggregate data collected on real people, and the same data collected on the simulated agents in similar scenarios. As shown in Figure 3.10, the PLE simulations match the prior data very well in terms of how quickly individuals move at various levels of congestion. Because agents are represented as a hard disk with radius of at least $0.3m$(an area of $0.28m^2$), the system cannot generate accurate simulations with densities greater than 4 agents$/m^2$ without creating overlaps between the agents. This limits the accuracy of the results in scenarios where more than 4 people are packed per m^2. (However, it is important to note this is beyond what is normally considered a safe density.)

3.7 Summary and Conclusions

This chapter has presented a novel mathematical formulation for generating energy-efficient trajectories based on a biomechanical principle for guiding agents in crowd simulations. I have presented a simple optimization algorithm to compute paths based on the well known Principle of Least Effort. I have also validated the results by comparing them to prior studies on crowd simulation and other real world data. The overall approach can be used for interactive crowd simulation with thousands of agents and automatically generates many emerging behaviors.

3.7.1 Limitations

The method has some limitations. Most importantly, the measurement of the biomechanical energy of locomotion is based only on studies of humans walking in straight lines at normal speeds. While this approach is sufficient to produce a wide array of emergent crowd behaviors and match real data, the generated motion could be more accurate with a more complex energy function accounting for various rates of turning and different styles of gaits. For example, this approach is unable to model motions such as running, panic situations and other atypical behaviors. Also, there are many behaviors of real-world crowds that are reproducable by this approach such as aggressive behavior.

Additionally, humans are modeled as a hard disk of fixed radius. While this can be a sufficient approximation for many scenarios, it ignores the fact that sometimes people may “squeeze” themselves to fit into very narrow passages or may come very close to other agents in a highly dense setting. This assumption also artificially slows down the rate at which the agents move through narrow passages.

Finally, given the complexity of computing the global optimum of the energy, a heuristic approach was used for minimization and may not be able to compute the most optimal trajectory in terms of total effort.

3.7.2 Future Work

There are several avenues for future work. This work has only examined the scenarios in which the agents move in either open space or around solid obstacles. These is an interesting range of other environments such as uneven or sandy terrain which would significantly change effort assumptions. Additionally, relaxing the fixed disk representation of humans would likely lead to better simulations in certain constrained scenarios. This could be in the form of an disk of ellipse whose size changes based on speed of local density.

The next chapter will explore the validating of this model in more detail, specifically focusing on it's ability to reproduce real-world data.

CHAPTER 4

PLE Simulation Approach: Analysis and Validation

4.1 Introduction

The work presented in Chapters 2 and 3 have primarily focused on the process of developing local collision avoidance methods. In this chapter, I will analyze the accuracy of this model. Previous chapters have provided evidences that the approximation of the PLE principle present here produces collision-free paths which are calorically efficient. The specific question addressed in this chapter is to what extent does modeling the PLE assumption reproduce typical human movement patterns and flow rates.

In this chapter, I show that this biomechanically-inspired model of pedestrian dynamics model can meaningfully predict the trajectories of humans in crowds and thereby generate natural crowd movement. Specifically, I show that individuals in a crowd minimizing their expected caloric expenditure results in a number of common emergent behaviors and phenomena. I also show that the model's predictions match the paths and flow rates of real humans well. While previous methods have demonstrated *some* of these aspects, the approach from this dissertation can demonstrate all of these aspects and is based on a simplified interpretation of human energy consumption presented in the biomechanics literature.

4.1.1 Main Result

This chapter will provide both qualitative and quantitative evidence that modeling the PLE assumptions leads to expected crowd motion. That is, that "least-effort" trajectories lead to emergent crowd behaviors.

Results of PLE simulations are compared against known human motion patterns and emergent phenomena. Overall flow rates of simulated agents and real humans are also compared. Finally, the specific paths taken by the PLE agents are compared to real human paths.

The analysis shows that the results of the computer simulations compare well with empirical data on human trajectories. The analysis also shows that the model can correctly reproduce many emergent crowd behaviors and numerically predict crowd flows in different settings.

4.1.2 Organization

This chapter is organized as follows. Section 4.2 briefly discusses related work. Section 4.3 provides an summary of the PLE method as implemented for these validation experiments. Section 4.4 covers the various simulation results and comparisons.

4.2 Previous Work

Many researchers, including Hoogendoorn and Bovy (Hoogendoorn and Bovy, 2004), have shown that crowd motion can be modeled as independent agents that try to optimize some utility function. Different models have been proposed to simulate human motion in a crowd. While some methods try to model the macroscopic or overall motion of the crowd (Hughes, 2003, 2002), they do not accurately model trajectories of individual pedestrians. In contrast microscopic, or agent-based, model of human motion specifically tries to model the position and velocity of each individual over time. Previously proposed models of this type include many-particle force-based models (Helbing et al., 2000, 2005) and their important extensions (Yu et al., 2005; Chraibi et al., 2010), methods based on simple rules (Reynolds, 1987), flow-field based methods (Kretz, 2009), and Cellular Automata models (Muramatsu and Nagatani, 2000; Hartmann, 2010). Methods have also been proposed which fit models to recorded data (Johansson et al., 2008) or extended existing models to capture newly recognized phenomena (Yu and Johansson, 2007). An overview of this general area of the physics of complex systems and transport can be found in Schadschneider et al. (Schadschneider et al., 2011).

Recently, researchers have proposed geometric optimization based, multi-agent simulation methods where agents attempt to directly compute collision-free velocities in a predictive manner, based on anticipated motion (Paris et al., 2007b), data collected on humans (Pettré et al., 2009), and cognitive theories of perception (Moussaïd et al., 2011). The PLE model likewise uses an optimization framework, which in contrast to these works, is based on biomechanically-inspired principles of individual motion.

A key issue in the design of an optimization-based approach is determining the correct metric to optimize. Previous approaches have suggested minimizing the total distance traveled (such as by using differential geometry (Maury and Venel, 2008), planning on geodesics (Hartmann, 2010), or using flow-based techniques (Hughes, 2002)). However, path length minimization is not a complete metric as its value is independent of an agent's speed. Additionally, it fails to model important real-world phenomena such as why humans tend to avoid congestion. In contrast, biomechanics research suggests the natural metric of calories expended over a path.

When walking in an unconstrained environment, people are known to move at velocities that minimize their caloric expenditure per unit distance (Browning and Kram, 2005; Whittle, 2002). However, when in crowd-like settings, other people create constraints on the possible motion an individual can take. My PLE work posits that it is this interaction between different individuals, each of whom is independently minimizing his or her expected effort, that gives rise to the emergent behaviors and motion patterns exhibited by crowds. Such behaviors can be numerically approximated by performing a constrained minimization over all the paths each individual can take, as described below. The PLE model is to compute all the pedestrian trajectories in a crowd based on this formulation.

4.3 Least-Effort Model

This section summarizes the PLE method. More detail can be found in Chapter 3.

A person's caloric expenditure rate, R, can be well approximated by a quadratic function of their instantaneous speed (Browning and Kram, 2005; Whittle, 2002). My work uses the approximation provided by Whittle (Whittle, 2002): $R = e_w|\mathbf{v}|^2 + e_s$, where the parameters of e_w and e_s can vary based on gender, age, and fitness level. This function was derived empirically by fitting a curve to data extracted from oxygen consumption of participants walking on a treadmill at various speeds.

For any given trajectory, integrating this caloric rate function over the trajectory will result in an estimate of the total calories a human would expend by traversing the trajectory. This leads to the following equation for the energy expended by a person moving along a path Π:

$$E(\Pi) = m \int_{\Pi} (e_w|\mathbf{v}|^2 + e_s)dt, \tag{4.1}$$

where m is the person's mass, e_w captures how efficiently calories are used, and e_s is a person's rate of energy consumption when standing still.

Based on this model, a person will be walking most efficiently when Eq. (4.1) is optimized per unit distance (Fig. 4.1). This happens with a path of a constant speed of $\sqrt{e_s/e_w}$. For the average adult male $e_w = 1.26$ and $e_s = 2.23$ (Whittle, 2002), which corresponds to a speed of 1.33 m s^{-1}. This matches the measured average walking speed for humans in low density environments (Klüpfel et al., 2005). In other words, a least effort analysis correctly predicts that people in an unconstrained environment will take the shortest path to their goal at their optimal speed.

In a crowded environment, however, nearby humans and obstacles result in additional constraints on an individual's motion. The strongest of these constraints is that two people cannot share the same physical space. Additionally, humans tend to avoid collisions in an anticipatory manner, reacting to the collisions before they occur (Pettré et al., 2009). This

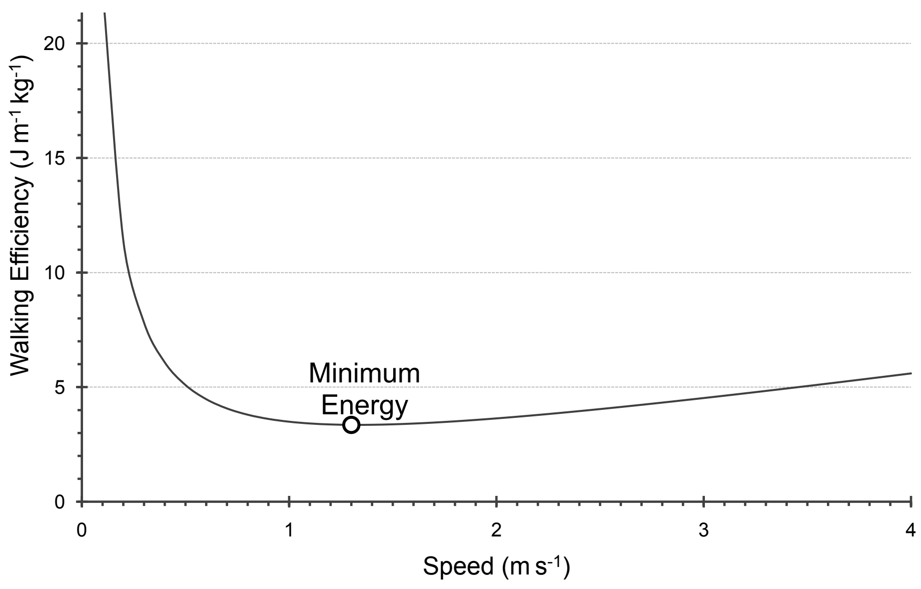


Figure 4.1: Graph of the empirical relationship between velocity and caloric efficiency for adult males (Whittle, 2002) (Eq. 4.1). The minimum energy corresponds to the velocity 1.3 m/s, the average walking velocity for adult males. (Klüpfel et al., 2005)

can result in a need to constantly adjust paths to avoid collisions well ahead of time to account for the potential actions of others. In other words, humans tend to choose velocities that will result in collision-free motion with respect to other nearby people and obstacles.

To capture these two aspects of human navigation, the PLE model represents each individual in the crowd as a virtual agent that attempts to find the minimum energy path to its goal while avoiding collisions. It models the tendency to anticipate collisions as a restriction on the set of permissible velocities an agent can take to include only those which result in collision-free paths for the near future. Here I denote these velocities as PV (Fig. 4.2 - white region).

The PLE formulation assumes each individual chooses the velocity from this set that is expected to minimize the energy described by Eq. (4.1). To achieve this minimization, the set of potential velocities PV is computed using a set of linear constraints on the velocities of each agent (Fig. 4.2). This set can be computed efficiently using geometric optimization techniques. In the remainder of this section I summarize the method by which this set PV can be generated and the optimal velocity computed for each agent.

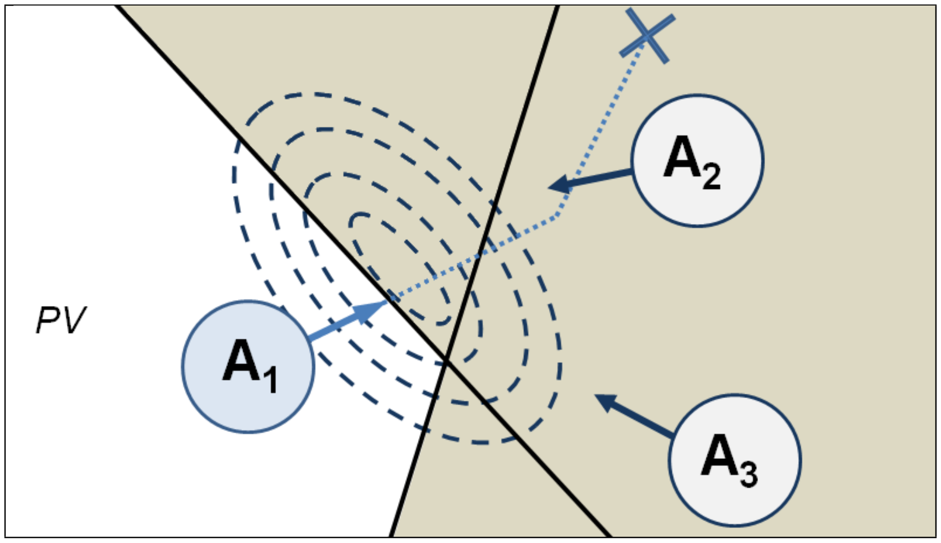


Figure 4.2: Computation Overview. The current agent, A_1, has a goal marked X, but needs to avoid two approaching agents, A_2 and A_3, each with some velocity (arrows). Each neighbor creates a restriction on the velocity the current agent can take (boundary line and shaded regions show forbidden endpoints of the velocity vector of A_1), leaving the set of collision-free permissible velocities (PV). Each velocity results in some expected energy to reach the goal (dashed ellipses mark the iso-contours of this function). The computed new velocity (light arrow) is the one which leads to the collision-free path to the goal (dotted line) using the least expected energy. This model is used to compute a new velocity for each agent at each simulation time-step.

4.3.1 Optimization Formulation

Agents are represented as a hard disk with a fixed radius r. Following the methods proposed in Berg et al. (van den Berg et al., 2011), PV is defined as a intersection of several linear constraints on an agent's velocity, one constraint for each neighboring agent. Given an agent A, for each neighboring agent B, B's constraint on A's velocity in computed by first finding the minimum change in the relative velocity between A and B needed to avoid collision for at least τ seconds. This change in velocity is denoted as the vector $\mathbf{u}$. A's velocity is the constrained to change by at least $\frac{1}{2}\mathbf{u}$ (with the assumption B will likewise avoid the other half of the collision). Therefore, given A has a current velocity of $\mathbf{v}_A^{\text{cur}}$, the permitted velocities given B are:

$$PV_{A|B} = \{\mathbf{v} : (\mathbf{v} - (\mathbf{v}_A^{\text{cur}} + \frac{1}{2}\mathbf{u})) \cdot \mathbf{u} \geq 0\}. \tag{4.2}$$

The boundary of this set is a line which goes through the point $(\mathbf{v} + \frac{1}{2}\mathbf{u})$ with the slope $\mathbf{u}^{\perp} = (\mathbf{u}.y, -\mathbf{u}.x)$. A similar formulation can handle avoidance of obstacles with the exception that obstacles can not be expected to reciprocate in avoiding collisions and therefore the entire vector $\mathbf{u}$ must be accounted for. Therefore, for an obstacle O:

$$PV_{A|O} = \{\mathbf{v} : (\mathbf{v} - (\mathbf{v}_A^{\text{cur}} + \mathbf{u})) \cdot \mathbf{u} \geq 0\}. \tag{4.3}$$

The union of these linear velocities constraints across all agents forms PV (Fig. 4.2). Formally:

$$PV = \bigcap_{B \neq A} PV_{A|B} \cap \bigcap_{O} PV_{A|O}. \tag{4.4}$$

The PLE notion of the collision-free path taking the least caloric energy can now be defined as:

$$\textbf{Minimize}\; E(\Pi)\; s.t.\; \mathbf{v}^{init} \in PV, \tag{4.5}$$

where agents are limited to paths whose initial velocity, $\mathbf{v}^{init}$, lies with the set of non-colliding velocities PV. Solving this equation produces a model for crowd motion which can be summarized as: *each agent finds the path,* Π, *with an initially velocity* $\mathbf{v}^{init}$ *from the permitted velocities* PV, *which minimizes the expected biomechanical effort to reach the goal.*

In practice people can only avoid other agents and obstacles that they are aware of. For the results in this chapter the region of awareness for an agent is represented as a circle of radius 10m, centered around their current position. Other, anisotropic, sensing models are also possible as the method is independent of the underlying sensing model.

4.3.2 Geometric Solution

Agent trajectories can be computed by solving Eq (4.5) for each agent. As a simplifying assumption, only paths Π which can be represented by two linear segments are considered. The first segment corresponds to a motion that avoids collisions with nearby obstacles and other individuals, and the second segment leads the agent directly to its goal position (Fig. 4.2 dotted line). By assuming that the avoidance segment takes τ seconds at an initial velocity of $\mathbf{v}$, the exact effort along the path can be computed as follows: Given an individual's current position, $\mathbf{p}$, and goal position, $\mathbf{G}$, Eq. (4.1) is used to compute the expected energy along a path as:

$$E(\mathbf{v}) = \tau(e_w|\mathbf{v}|^2 + e_s)m + 2|\mathbf{G} - \mathbf{p} - \tau\mathbf{v}|\sqrt{e_s e_w}m. \tag{4.6}$$

By combining Eq. (4.5) and Eq. (4.6), the Least Effort model for trajectory computation and motion in crowds can be defined as:

$$\textbf{Minimize}\, E(\mathbf{v}^{new})\ s.t.\ \mathbf{v}^{new} \in PV. \tag{4.7}$$

Equation (4.7) can be solved efficiently by exploiting the convexity of the energy function, E, and the convexity of the set of potential velocities, PV as described in Chapter 3, by using a linear programming type solution, where each linear segment on boundary of PV is optimized for sequentially. Specifically, the velocity that minimizes Eq. (4.6) along a chosen line segment which we denote as $\mathbf{v}^{opt}$. For Eq. (4.6), this point can be found analytically through differentiation. For each linear boundary segment of PV, it is checked if $\mathbf{v}^{opt}$ is a permitted velocity. If it is not a permitted then a new optimal velocity is found along the linear boundry. This new point now serves as $\mathbf{v}^{opt}$, and this process is repeated for each remaining segment of PV. The final value of $\mathbf{v}^{opt}$ will be the point that minimizes Eq. (4.7).

This process is repeated for each agent to compute the optimal velocity, $\mathbf{v}^{opt}$, for that agent. All agent positions are then updated using Eulerian integration of their velocity over discretized time-steps (0.1s in the results below). This process is repeated until each agent reaches its goal position. As discussed in (van den Berg et al., 2011) and (Guy et al., 2010a), this general method for collision avoidance will lead to provably smooth and collision-free paths (provided there is sufficient free space for agents to maneuver). This adds to the generality of the model by alleviating the need to tune specific parameters to find smooth or collision-free paths; rather free parameters can be used to capture the naturally occurring variation in human motion. Additionally, because the implicit cooperation between agents (the results of avoiding only $\frac{1}{2}\mathbf{u}$) large timesteps can be used while still maintaining collision-free motion between agents (van den Berg et al., 2011).

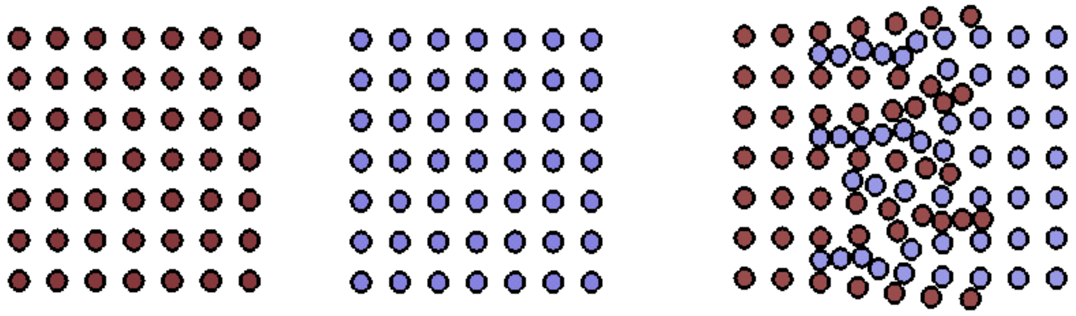
The resulting model has three free parameters to describe each agent: the agent's radius, r, and the parameters e_s and e_w in their energy function which define their preferred velocity.

4.3.3 Global Navigation

In cases when an agent's goal is not immediately visible, we use a roadmap (a graph of connected, mutually visible, intermediate goals) to select a path of intermediate goals for an agent. The agent then navigates via these intermediate goals along the way to its ultimate destination (Guy et al., 2010a; van den Berg et al., 2011). Here, the roadmap is formed by randomly sampling potential, collision-free positions to select intermediate goals and then connecting mutually visible intermediate goals (whose direct between them does not pass through walls) to create a graph with the intermediate goals as nodes and the path between these goals as edges. Each edge is weighted by the expected caloric energy needed to traverse the link. Standard graph search techniques are used to find the series of intermediate goals which forms the path of least expect effort to reach an agent's final goal (LaValle, 2006). The next visible of these intermediate goals serves as an agent's goal, $\mathbf{G}$, in Eq. (4.6).

4.4 Results

The validity of the model described in Sec. II can be analyzed in several respects. First, I examine the emergent phenomena generated by the model. These are effects which are not explicitly accounted for in the formulation, but reliably occur in simulations due to the interaction between agents. Secondly, I present a quantitative analysis of how closely the simulated results match data collected about real-word crowd flows and paths taken by humans in controlled studies. Finally, I analyze results from simulations of complex scenarios consisting of thousands of independent agents and hundreds of obstacles.



(a) Initial Conditions (b) Lane Formation

Figure 4.3: Lane Formation. (a) Two opposing groups of agents, (dark) red and (light) blue, have opposing initial conditions with goals past each other. (b) As the groups approach the agents naturally form into small coherent lanes reducing the overall effort of each individual.

4.4.1 Emergent Phenomena

As discussed briefly in Chapter 3, several different crowd movement patterns and other emerging behaviors arise from the Least-Effort model which can be compared them to observed phenomena in real crowds. For example, simulated individuals tend to dynamically form emergent lanes when they are moving in bi-directional flows as demonstrated in Fig. 4.3. In this scenario two groups of agents are given goals corresponding to a horizontal movement in opposite directions along the x-axis. In the process of reaching their goals, agents naturally self-organize into lanes. This is a result of the fact that agents spend fewer

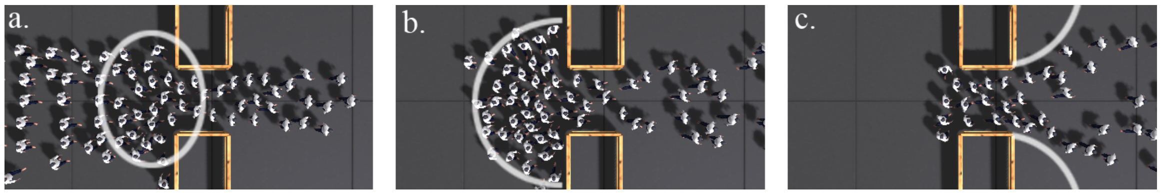


Figure 4.4: Stills from a simulation of humans walking through a narrow passage, taken at 15 second intervals. There is initially jamming at the passage (a), followed by a semi-circular arch forming around the exit (b). Once through the passage, individuals do not immediately spread out, but leave an empty space or "wake" behind the obstacles (c).

calories by joining existing lanes of people moving with a similar direction and speed. This allows individuals to move at their most energy efficient speed without having to slow down to avoid collisions and thereby minimize the total individual effort. This emergent lane formation has been commonly reported in observations of real-world crowds (Still, 2000; Helbing et al., 2005; Klüpfel et al., 2005).

The PLE model is also able to reproduce observed human behavior at narrow passages. The scenario shown in Fig. 4.4 highlights many of these behaviors. Here, each agent is given a goal horizontally along the x-axis beyond the narrow passage. The simulated agents tend to jam in the congestion that forms at a narrow passage as they attempt to avoid colliding with other individuals who are nearby. This also leads to semi-circular arching around the passage as the individuals try to come as close as possible to the exit in order to minimize time spent in congestion. These phenomena of jamming, congestion, and arching around exits have all been reported in studies of real human crowds (Klüpfel et al., 2005; Predtetschenski and Milinski, 1971).

The process of each individual minimizing his or her caloric energy also allows the model to capture several other common crowd phenomena. For example, obstacles in a crowd's path create an open space behind them which people do not immediately fill (Fig. 4.4, right panel). This is known as the wake effect (Still, 2000) as it is reminiscent of flow separation regions in fluids. In such regions the agents tend to choose a direct path towards the goal as it is more efficient overall than filling in the free space behind an obstacle.

Energy minimization also explains overtaking behavior seen in crowds. As shown in Fig. 4.1, moving slower than the optimal speed is inefficient. Individuals with higher optimal velocity will therefore overtake the slower ones, minimizing their overall effort. Another related phenomenon is congestion avoidance. Taking paths which avoid regions of high density, slow moving individuals often results in using less caloric energy than slowing down and moving through the congestion. Simpler simulation methods such as finding the shortest or quickest path fails to reproduce such effects.

4.4.2 Flow Analysis

Further validation of this biomechanically-inspired model of crowd motion can be performed by comparing predictions from the simulation to actual flow data. For example, several recent studies have analyzed human exit times through doorways of various widths (Müller, 1981; Nagai et al., 2006; Kretz et al., 2006; Seyfried et al., 2009).

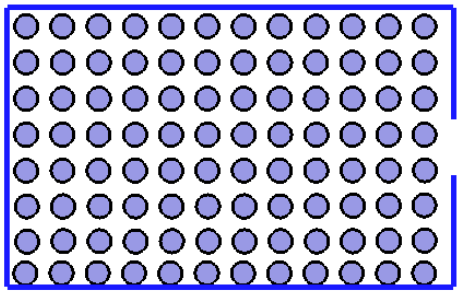


Figure 4.5: Flow Analysis Scenario. 96 agents are placed in a room of dimension 5m x 8m. Agents are given a goal outside the room which requires them to pass through the exit on the right wall. The experiment is repeated for various exit widths varying from 0.8m to 1.4m.

To compare the PLE simulation results to these studies I created a scenario similar to those of the above studies. Several simulations were run where approximately 100 simulated agents are given a goal of a point 10m outside the room centered along the exit's midpoint. For each run the width of the room's exit was varied from 0.8m to 1.4m. Fig. 4.5 summarizes this simulation set-up.

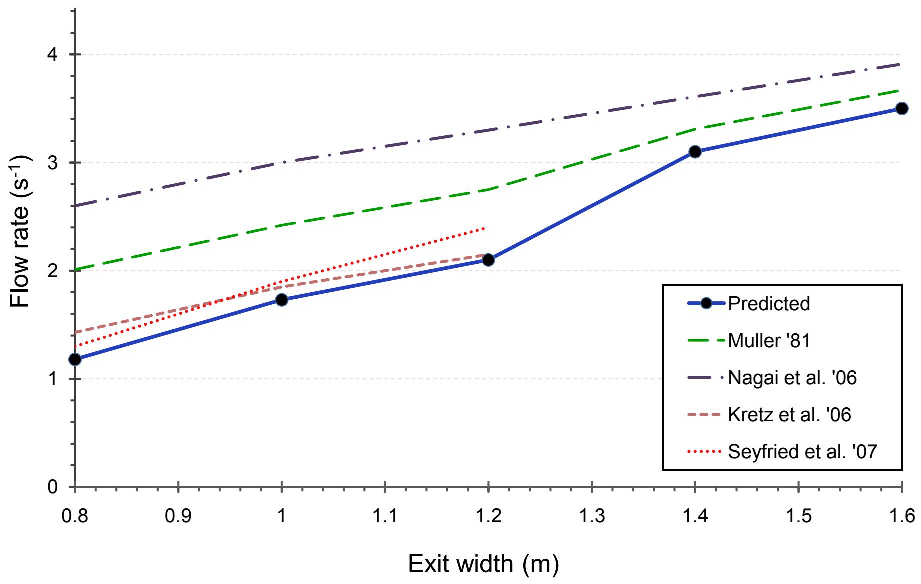


Figure 4.6: Real and Simulated Flow Rates. A comparison of the effect of exit width on the flow for real (dashed lines) and simulated (solid line) humans. Agents simulated with the PLE model exhibit similar flow as real humans.

Fig. 4.6 compares the flow rate predicted by the least-effort simulations and flow rates measured on real humans. The predicted flow rates lie within the range of flows reported for humans for a range of exit widths.

Another aspect of human flow is the well established correlation between increased density and slower speeds known as the fundamental diagram (Seyfried et al., 2008; Weidmann, 1992). The PLE model shows a similar trend. As an example, I initialize agents with the positions and velocities reported in several timesteps of the Bottleneck benchmark in (Seyfried et al., 2008) and plot the predicted speed of each agent as a function of density (Fig. 4.7). While there is variation in the agents' speeds at any density, in general, agents in high density regions move significantly slower than those in low density regions. The blue line in Fig. 4.7 shows a quadratic fit of the data, this fit closely matches the fundamental diagram established by Weidmann (Weidmann, 1992).

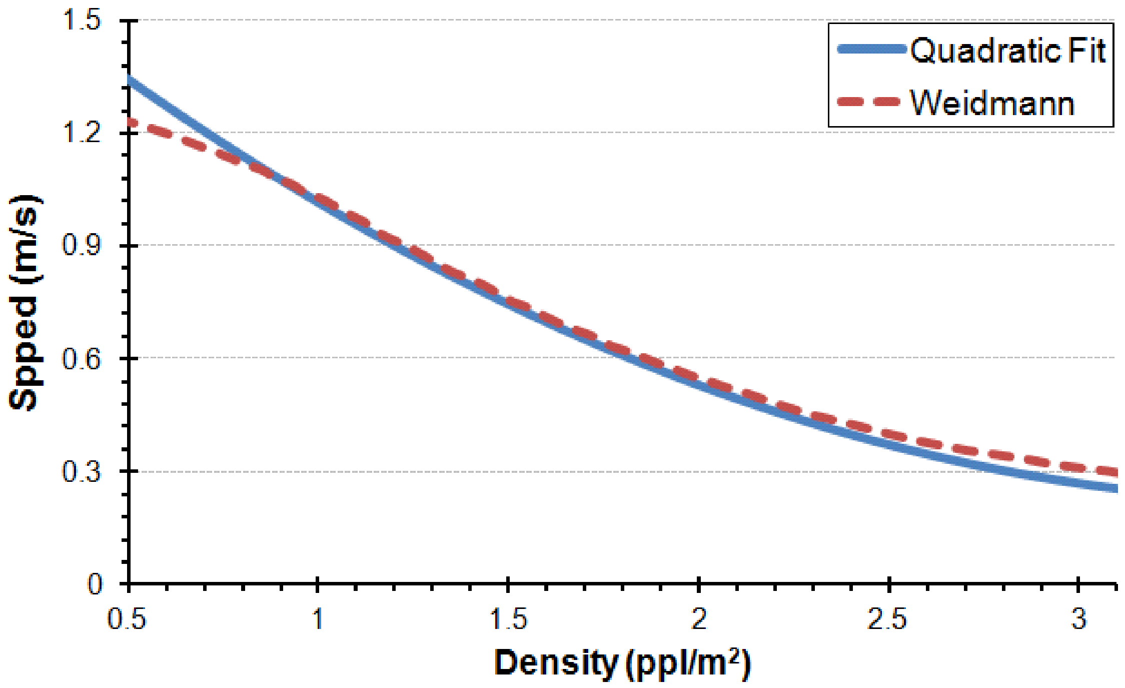


Figure 4.7: The fundamental diagram comparing agent speeds vs their local density (solid line) matches the relationship established by Weidmann (dashed line) (Weidmann, 1992).

4.4.3 Path Comparison

In addition to comparing real and predicted flows, I can also compare the paths predicted by the PLE method to those of real people walking in similar condition. Here the data comes from two scenarios gathered at the Centre de Recherches sur la Cognition Animale, CNRS, Toulouse, France and presented in (Moussaïd et al., 2011).

The first scenario involved two people standing about 6m apart and being instructed to exchange places. Fig. 4.8 shows the mean and standard deviation of the participants' paths. In this figure all the paths have been normalized so that "forward" corresponds to the positive x-axis. Overlaid on the actual human paths is the path predicted by the PLE method (r=.28m). The path predicted by the PLE method falls within the variation of the human paths and closely matches the mean of the human paths.

In the second scenario (Fig. 4.9) paths from the model are validated against those of real humans when walking around a static obstacle. Again, the path predicted by the PLE

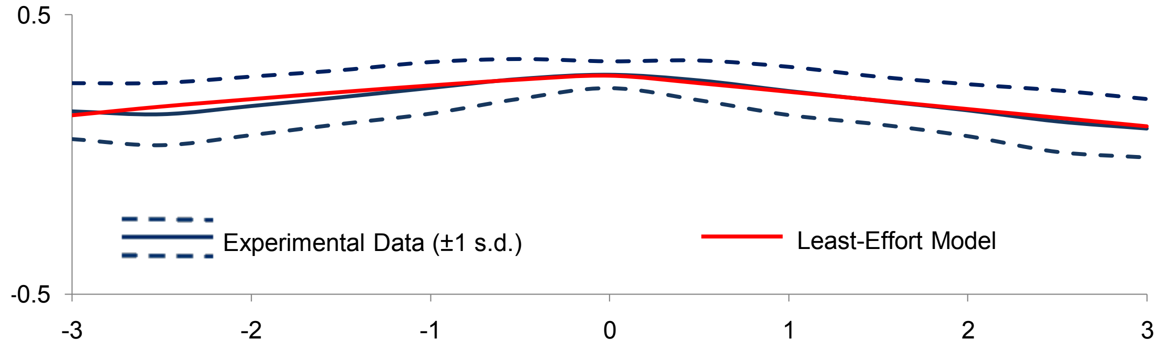


Figure 4.8: Paths of two humans passing each other. The model's path (light solid line) matches very closely with the the mean of the human paths (dark solid line), and within one standard deviation of human paths (dashed lines).

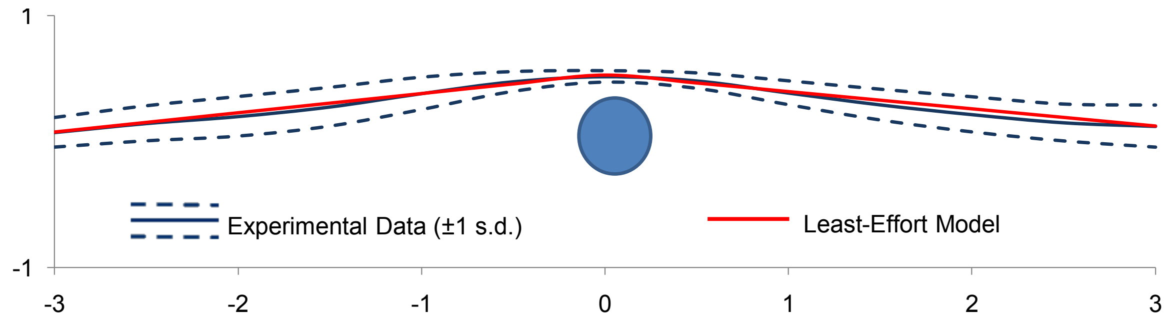


Figure 4.9: Paths taken past a static obstacle (circle). The model's path (light solid line) matches very closely with the the mean of the human paths (dark solid line), and within one standard deviation of human paths (dashed lines).

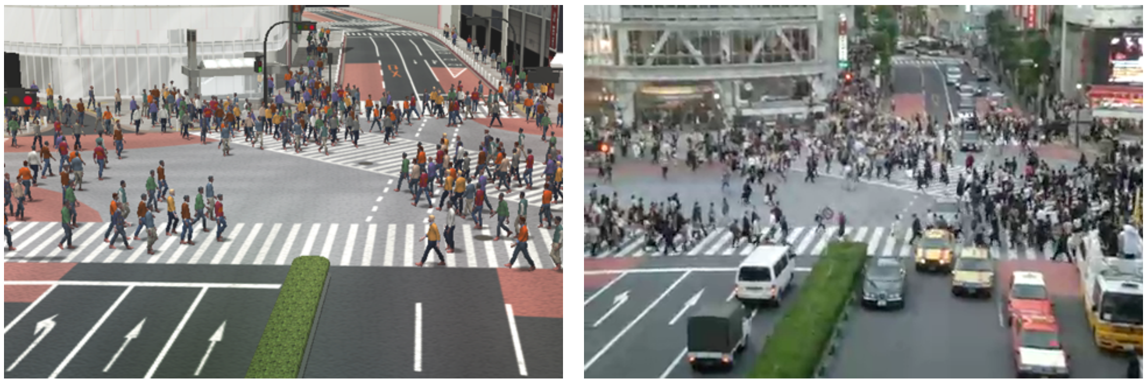
model lies within the variation seen in human paths and is close to the mean of the human paths.

4.4.4 Complex Scenarios

The simplicity of the PLE algorithm allows for computationally efficient implementations capable of simulating large scale crowd behavior in real time. Because of the underlying efficiency of the approach, the model can be used to generate realistic crowd behaviors for complex, real-world scenarios.

One of the key challenges in simulating complex environments is to avoid collisions of the agents with each other and between the agents and the obstacles in the environment. The PLE approach is scalable and can handle complex scenarios involving thousands of virtual agents. In these complex scenarios, the constrained optimization framework successfully avoids collisions between agents and with obstacles while still navigating agents to their goal.

In these complex scenarios, the algorithm is capable of producing motion which is similar to real human paths. Figure 4.10 shows a comparison between a simulation and real footage of the five-way scramble crossing outside Shibuya Metro Station in Tokyo. The agents perform the crossing at similar speed to the real people and with similar over all flow patterns and lane formation. Importantly, this scenario also demonstrates phenomena such as lane formation in a natural setting.



(a) A still from crossing simulation (b) A still from a video of the crossing

Figure 4.10: The PLE approach automatically generates many emergent crowd behaviors at interactive rates in this simulation of Shibuya Crossing (left) that models a busy crossing at the Shibuya Station in Tokyo, Japan. (right). The trajectory for each agent is computed based on minimizing an biomechanically-inspired effort function.

4.5 Summary and Conclusions

In summary, I have introduced a new computational model to simulate crowds that display collective behaviors formed by individual trajectories. By combining fundamental biomechanical measurements and the Principle of Least Effort I was able to develop a crowd simulation system based on constrained energy minimization. The chapter has validated the model by comparing the predicted results with data from real-world crowds and have shown that it can accurately model humans paths, crowd flows, and emergent behaviors.

4.5.1 Limitations

While I have shown that constrained caloric energy minimization can successfully reproduce typical crowd behaviors, there are still scenarios which are not currently well modeled by this approach. For example, there are social and psychological factors, such as running when panicked, that can not be captured simply in terms of minimizing the biomechanical energy of locomotion. Additionally, different people do not always take the exact same path, but rather exhibit variations which come from differences in personality and style.

4.5.2 Future Work

Looking forward, I conjecture that this approach can be extended to eventually model several of these sociological and psychological factors, as well capture some of the variations seen in humans. Accounting for factors such as discomfort in dark areas or close to walls could further enhance the approach. Such a model could be used to analyze crowd flows in various environments and assist in predicting and controlling crowds in large assemblies.

Additionally, the least-effort model should be compared to other optimization-based or predictive approaches, ideally on a qualitatively bases. Such a study would ideally focus on highly discriminative scenarios such as two pedestrians approaching at various angles. Chapter 7 presents a possible approach for performing such an evaluation.

CHAPTER 5

Data-driven Simulation of Variations in Human Trajectories

5.1 Introduction

The work presented in Chapters 2 through 4 focus on how to simulate *typical* people in expected environments. However, most people deviate from the norm in interesting ways: they have differences in comfort zones, differences in reaction times and even differences in physical abilities. This chapter proposes a data-driven way to capture these important differences in human motion within the ORCA simulation framework.

Here I propose several changes to ORCA to form a new algorithm which focuses on Reciprocal Collision Avoidance for Pedestrians or RCAP. The RCAP algorithm explicitly models a human's reaction and observation time, as well as kinodynamic constraints on their motion. The model is compared to and validated against data of real humans avoiding collisions with each other that was collected from the Locanthrope project by (Pettré et al., 2009). Additionally, statistical distributions of key model parameters are derived from the human motion data. Finally, the performance of the model is demonstrated on simulations of both small and large groups of agents, with and without obstacles.

5.1.1 Main Result

This chapter describes RCAP an extension of ORCA which:

- Accounts for reaction time
- Models variations in personal space
- Models kineodynamic movement limitations

- Is derived from real motion data

- Provides for a natural variation in motion generated from recorded data

5.1.2 Organization

This chapter is organized into the following sections. Section 5.2 gives an overview of prior work on collision-free navigation and modeling of human motion. Section 5.3 provides a brief summary of ORCA and other multi-robot navigation techniques used in the RCAP approach. Section 5.4 introduces the new algorithm for pedestrian navigation called Reciprocal Collision Avoidance for Pedestrians (RCAP). Section 5.5 describes the experimental setup and methods used to train the model and Section 5.6 highlights the validation of the model a real-world dataset. Section 5.7 presents the simulation results and I analyze key aspects of the algorithm in Section 5.8.

5.2 Background and Previous Work

5.2.1 Multi-robot Collision Avoidance

Multi-robot collision avoidance is a form of robot motion planning which involves moving a robot directly towards its immediate goal, while only avoiding collisions with obstacles and other robots as needed. A well established techniques for local collision avoidance comes from the concept of Velocity Obstacles (Fiorini and Shiller, 1998) which computes a forbidden set of potentially colliding velocities which provides an appropriate means to develop such a strategy. However, this method fails when the obstacle being avoided is also actively reacting to avoid that robot as well, e.g. in the case of humans avoiding other humans. Reciprocal Velocity Obstacles (RVO) (van den Berg et al., 2008a) provides an extension to the Velocity Obstacles concept that works for agents who are actively avoiding each other, and ORCA (van den Berg et al., 2011) extends this concept further to work for any number of robots.

5.2.2 Modeling Human Motion

There have been many attempts to characterize how humans move using mathematical or scientific models. At a physiological level, researchers study topics such as human foot placement and the characterization of various gaits (Whittle, 2002). Other low-level aspects of human motion have been extensively studied and a survey of this topic can be found in (Elgammal et al., 2007).

5.2.2.1 Cognitive and Behavioral Models

Sever researches have attempted to explicitly model the cognitive aspects of avoiding and interacting with others. (Funge et al., 1999) used cognitive modeling methods to allow agents to plan and perform high level tasks. (Shao and Terzopoulos, 2005) proposed an artificial life model where agents make decisions at different levels of abstraction. (Yu and Terzopoulos, 2007) introduced a method to simulate behaviorally animated agents using bayesian decision networks, which also allowed for variations in agent behaviors.

When modeling the behavior of humans interacting with each other, it can be important to capture the social and psychological aspects of these interactions. For example, (Pedica and Vilhjalmsson, 2008) proposed a method for modeling social interaction by incorporating a set of prioritized behavior models which are executed at different frequencies. Additional models have been proposed to incorporate the results of *Proxemics studies* on how humans perceive and use the space around them such as (Rehm et al., 2005).

An important issue with any simulation method is to ensure that the model used accurately captures the motion and behavior seen in real people. Data-driven approaches explicitly address this issue by deriving a model directly from some real-world crowd data. For example, (Arechavaleta et al., 2008) derived a numerical optimization based model from motion capture data of people walking to a target spot on the floor. Data-driven methods have also been used to produce simulated crowds which behave with a certain trait or "style" matching some source data (typically a video of a crowd). One example is (Lee et al., 2007), which used data-driven methods to match recorded motion from videos by training a behavior model. The work of (Ju et al., 2010) also proposed a data-driven method which attempts to match the style of simulated crowds to those in a reference video.

5.3 Distributed Collision Avoidance

The RCAP approach to modeling human collision avoidance builds off recent work in collision avoidance for mobile robots, specifically the ORCA collision avoidance algorithm (van den Berg et al., 2011). This section provides an overview of the ORCA algorithm and the optimization framework used to compute agent paths.

There are several reasons why RCAP is based on ORCA. First, ORCA provides strong collision avoidance guarantees, which is appropriate as humans seldom collide, and rarely overlap each other. Secondly, it incorporates a notion of reciprocity or implicit cooperation when avoiding collisions, which is important in modeling how humans generally avoid collision without any explicit coordination. Finally, ORCA is built on the notion of respecting time-to-collision constraints, that is forbidding velocities which might cause collisions in the near future. Recent studies such as (Schrater et al., 2000) and (Hopkins et al., 2004) have shown that human brains have special cells dedicated to processing time-to-collision, i.e. how long until they would run into another person or object. This suggests that exploiting time-to-collision calculations (as ORCA does) might provide an accurate model of human collision avoidance.

Problem Definition: ORCA provides an efficient solution to the n-body collision avoidance problem in a distributed manner. The n-body collision avoidance problem involves n virtual agents sharing the same environment, each with their own current position, physical extent, and velocity (which are all known to other nearby agents), and an internal desired goal velocity which could change as the simulation progresses. For simplicity, each agent is represented as a 2D disk, with a current position, a current velocity and a desired velocity. The goal is to compute a new velocity for each agent at every step of the simulation such that none of the resulting trajectories will collide. ORCA solves this problem in O(n) running time for each agent, where n is the number of nearby agents.

Table 5.1 gives a description of the variables used by ORCA to represent an agent. These variables taken together, completely determine the unique state of any agent in the simulation.

Symbol	**State**	**Description**
r_a	External	Agent A's radius
$\mathbf{p}_a$	External	Agent A's position
$\mathbf{v}_A$	External	A's Current velocity
$\mathbf{v}_A^{\max}$	Internal	A's Maximum velocity
$\mathbf{v}_A^{\mathrm{pref}}$	Internal	A's Desired (goal) velocity

Table 5.1: Variables of state for each agent

5.3.1 Velocity Obstacles

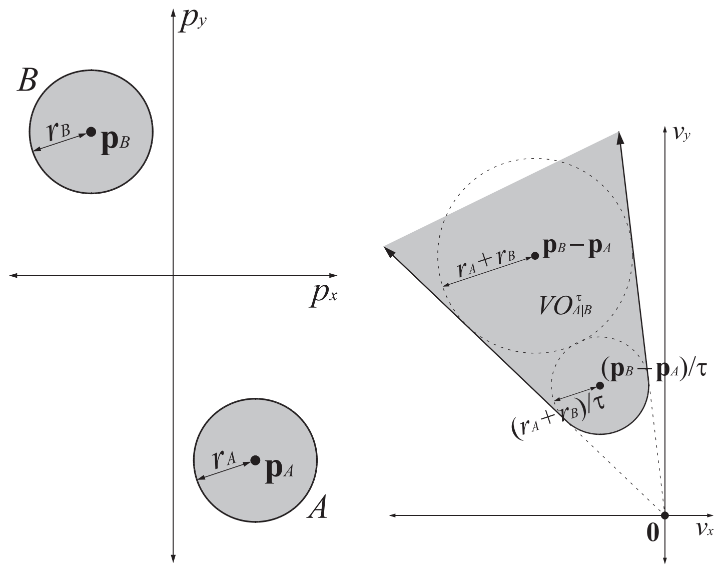
ORCA is built on the concept of velocity obstacles (VO) (Fiorini and Shiller, 1998). A VO is a set of forbidden velocities which would lead to a collision between an agent and an obstacle in the next τ seconds. Formally, a VO is defined as follows. Let $D(\mathbf{p}, r)$ denote an open disc of radius r centered at $\mathbf{p}$:

$$D(\mathbf{p}, r) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{p}\| < r\}, \tag{5.1}$$

then, given a time horizon τ for which to avoid colliding with any obstacles:

$$VO^{\tau}_{A|B} = \{\mathbf{v} \mid \exists t \in [0, \tau] :: t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, r_A + r_B)\} \tag{5.2}$$

The geometric interpretation of a VO is shown in Figure 5.1. This figure is drawn in velocity space where the graph's origin corresponds to a velocity of $\mathbf{0}$ (standing still), the x-axis corresponds to the x-component of the velocity, and the y-axis to the y-component of the velocity. In this space, a VO^{τ} has the shape of a truncated cone. Intuitively, this can be thought of as the set of all the velocities which move an agent A towards obstacle B too quickly to be able to avoid collisions with it.



(a) 2 Agents $A\&B$ (world space) (b) $VO^{\tau}_{A|B}$ (velocity space)

Figure 5.1: (a) Shows the positions of two agents A and B, with zero relative velocity. (b) Shows the VO in A's velocity space induced by B. This is the set of all of A's velocities which would collide with B within τ seconds.

VOs are also appropriate for dynamic obstacles. If the obstacle B was moving with respect to A, the apex of the VO would be shifted to lie at the relative velocity $\mathbf{v}_B - \mathbf{v}_A$.

Whenever A chooses a velocity that is outside of $VO^{\tau}_{A|B}$, agent A is *guaranteed* not to collide with the obstacle B for at least τ seconds. This guarantee only holds *assuming B does not change its velocity* over the course of those τ seconds. If B is another intelligent agent and not an obstacle following a predefined trajectory, this assumption does not hold. If A and B are on a colliding path, B will change its velocity (in an attempt to avoid colliding with A). If two agents use a strictly VO based means to avoid each other, they will constantly oscillate between over-correcting for the collision and under-correcting for it (van den Berg et al., 2011).

5.3.1.1 Reciprocal Collision Avoidance

The work of (van den Berg et al., 2011) introduced the notion of reciprocity into multi-agent planning. Instead of taking full responsibility for avoiding a collision, if A knows B is a responsive agent, A will perform *only half* the work of avoiding collisions with the expectation that B will similarly perform the other half of the collision avoidance work. Assuming that both agents involved are following the same basic strategy, this method is *provably* oscillation-free and collision-free (van den Berg et al., 2011).

5.3.2 Optimization Formulation

Building on the concept of reciprocity, the ORCA algorithm provides Optimal Reciprocal Collision Avoidance between multiple agents. The method works by creating linear constraints that ensure every agent's *new* velocity will be outside of the VO^{τ} of every other agent's new velocity (van den Berg et al., 2008a). This is in contrast to previous techniques such as (Fiorini and Shiller, 1998), which assigns agents new velocities that are outside the VOs generated by other agents' *old* velocities.

The ORCA algorithm generates linear constraints on an agent's velocity, which will guarantee reciprocal collision avoidance. The first step in constructing these constraints is to find the smallest change required in the relative velocity of A and B to avert the collision between them and denote this vector as $\mathbf{u}$. Assuming agents A and B are traveling at $\mathbf{v}_A^{\text{opt}}$ and $\mathbf{v}_B^{\text{opt}}$, respectively, $\mathbf{u}$ can be geometrically interpreted as the vector going from the current relative velocity $(\mathbf{v}_B^{\text{opt}} - \mathbf{v}_A^{\text{opt}})$ to the closest point on the VO^{τ} boundary (see Figure 5.2). Specifically,

$$\mathbf{u} = \big(\underset{\mathbf{v} \in \partial VO^{\tau}_{A|B}}{\arg\min} \, \|\mathbf{v} - (\mathbf{v}_A^{\text{opt}} - \mathbf{v}_B^{\text{opt}})\| \big) - (\mathbf{v}_A^{\text{opt}} - \mathbf{v}_B^{\text{opt}}). \tag{5.3}$$

Because the agents are implicitly "sharing responsibility" of collision avoidance, each agent needs to change its velocity by (at least) $\frac{1}{2}\mathbf{u}$ with the expectation that the other agent

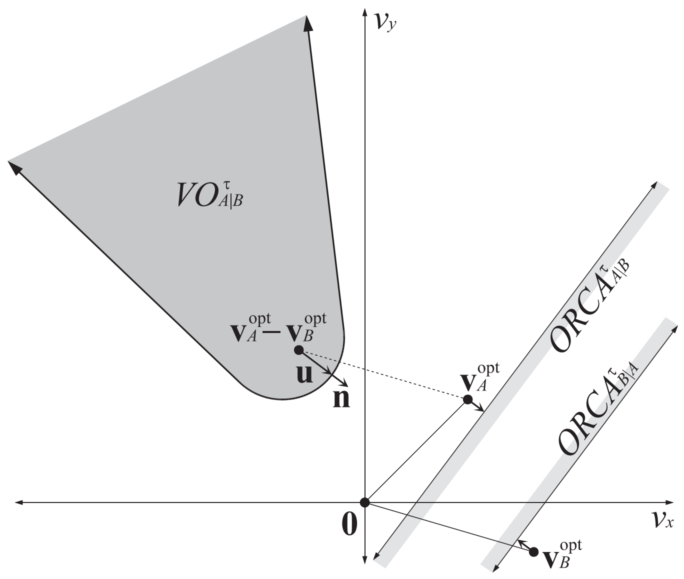

v_y
$VO^{\tau}_{A|B}$
$\mathbf{v}^{\mathrm{opt}}_A - \mathbf{v}^{\mathrm{opt}}_B$
$\mathbf{u}$
$\mathbf{n}$
$\mathbf{v}^{\mathrm{opt}}_A$
$ORCA^{\tau}_{A|B}$
$ORCA^{\tau}_{B|A}$
$\mathbf{0}$
v_x
$\mathbf{v}^{\mathrm{opt}}_B$

Figure 5.2: Constructing the set of ORCA allowed velocities. $\mathbf{v}^{\mathrm{opt}}$ is the agent's current velocity. ORCA forces agents to choose new velocities which avoid at least half the collision $\mathbf{u}$. In RCAP, only agents whose $T^{B|A}_{sight} < T - T_{obs}$ generate $ORCA$ constraints.

will take care of the other half. Therefore, the set of velocities permitted by ORCA for agent A is the half-plane starting at point $\mathbf{v}^{\mathrm{opt}}_A$ and facing away from $VO^{\tau}_{A|B}$. The normal of the half-plane, $\mathbf{n}$, is chosen to be the normal of the closest point on $VO^{\tau}_{A|B}$ in order to maximize allowed velocities near $\mathbf{v}^{\mathrm{opt}}_A$. Therefore, the set of ORCA allowed velocities for A is:

$$ORCA^{\tau}_{A|B} = \{\mathbf{v} \,|\, (\mathbf{v} - (\mathbf{v}^{\mathrm{opt}}_A + \frac{1}{2}\mathbf{u})) \cdot \mathbf{n} \geq 0\}. \tag{5.4}$$

$ORCA^{\tau}_{B|A}$ for B is defined symmetrically (see Figure 5.2).

5.3.2.1 Multi-agent Collision Avoidance

If agent A is avoiding collisions with multiple agents, its allowed velocities are simply the intersection of the $ORCA^{\tau}_{A|B}$s generated by each other agent B. If this set is empty, a "least bad" velocity can be chosen as discussed in (van den Berg et al., 2011). The entire set of allowed velocities for an agent A, referred to as $ORCA_A^\tau$, can be formally defined as:

$$ORCA_A^\tau = D(\mathbf{0}, v_A^{\max}) \cap \bigcap_{B \neq A} ORCA_{A|B}^\tau. \tag{5.5}$$

Note that this definition also includes the maximum speed constraint on the agent A of $\mathbf{v}_A^{\max}$.

Each $ORCA_{A|B}^\tau$ corresponds to a linear constraint on A's velocity. The task of selecting a new velocity closest to A's desired velocity $\mathbf{v}_A^{pref}$ subject to the linear $ORCA$ constraints can be solved efficiently using linear programming.

By always choosing a new $ORCA$ allowed velocity as close as possible to $\mathbf{v}_A^{pref}$ for each agent, all of the agents will move in a theoretically sound and efficient manner.

5.4 Reciprocal Collision Avoidance for Pedestrians

Unlike ORCA, real humans do not move in a perfectly efficient manner. This section presents a novel extension of the ORCA collision avoidance algorithm by taking into account some of the characteristics of human motion.

5.4.1 Modeling Human Motion

As described in Section 5.3, the ORCA algorithm solves the n-body collision avoidance problem in an efficient and robust manner. While it is theoretically sound, as a model for humans navigating around each other, ORCA misses two key aspects. First, humans take time to react to any collisions, whereas the ORCA model responds to collisions instantaneously. Second, even when a human decides how to avoid a collision, he or she is subject to physical constraints in terms of how quickly they can adopt their new velocity (Arechavaleta et al., 2008). This decision making process happens all while the humans are still walking towards their goals.

5.4.1.1 Response and Observation Time

When two humans first see each other, it typically takes time for them to understand and evaluate what is going on in terms of their relative motion. This time includes visually processing the appearance of the other individual, recognizing that this individual is walking towards them, deciding that their trajectories may pass too close to each other, and calculating a new velocity that will avoid a collision with the other individual.

This aspect of human motion is modeled by introducing a new parameter T_{obs} that corresponds to the time required for observation and reaction for each agent. Given a simulation currently at time T, if agent A spots a new neighbor at time T_{sight}, this new neighbor would not be considered as an obstacle (nor contribute to A's motion) until T_{obs} time has passed, that is until:

$$T \geq T_{sight} + T_{obs}. \tag{5.6}$$

The agent A will see each new neighboring agent B at a unique time which is denoted as $T_{sight}^{B|A}$. This allows us to modify the original $ORCA$ equation (Equation 5.5) to include the observation time effect:

$$ORCA_A^\tau = \bigcap_{B \neq A} \{ORCA_{A|B}^\tau : T_{sight}^{B|A} < T - T_{obs}\}. \tag{5.7}$$

While Equation 5.6 assumes a universal T_{obs} for all individuals, one can use a different observation time parameter for each different virtual human as discussed in Section 5.7.1.

5.4.1.2 Kinodynamic Constraints

In addition to response and observation time, humans are subject to kinodynamic constraints on their motion. These are constraints on the possible velocities and accelerations an agent may take (LaValle, 2006). While T_{obs} provides a way to model human mental constraints, it does not capture the effect of these physical constraints on human motion well.

The original ORCA formulation includes a term $\mathbf{v}^{\max}$, which models the fact that agents have a maximum possible speed. Beyond this constraint, humans also have other significant limitations on their motion. For example, humans cannot simply choose any new velocity instantaneously. There are physical limits in terms of how fast a person can come to a stop or accelerate from a resting position to a desired velocity, or switch from heading left to heading right, etc.

In order to model these physical constraints, I introduce a second parameter $a^{\max}$. This parameter captures the maximum rate at which an agent can change its velocity. If the acceleration required to reach the newly computed velocity, $\mathbf{v}_{computed}$, is larger than that allowed by $a^{\max}$, the new velocity will be clamped to be within the allowable range using the following equation, where ΔT is the amount of time which has passed since the last timestep, and $\Delta\mathbf{v} = \mathbf{v}_{computed} - \mathbf{v}_{old}$:

$$\mathbf{v}_{new} = \mathbf{v}_{old} + a^{\max}\Delta T \frac{\Delta\mathbf{v}}{\|\Delta\mathbf{v}\|}. \tag{5.8}$$

In addition to physical constraints, psychological factors such as distraction or anxiety can also effect the rate at which people respond to collisions and thereby effect $a^{\max}$.

5.4.1.3 Personal Space

When humans are represented as disks which tightly bound of their physical extent, a precise solutions to collision avoidance would produce paths were they would brush shoulders. However, when real humans pass each other they typically do not brush shoulders, but pass with some space between them. In practice, as *Proxemics* (the study of distances between interacting people) tells us, humans instead give each other a wider *affordance* (Hall, 1963), a concept commonly referred to as "personal space" (see Figure 5.3). This personal space provides room to safely swing limbs and serves as a buffer of comfort between people. Rather than planning around an agent's physical extent, I use instead the agent's personal space to define the planning radius, r. Like T_{obs} and $a^{\max}$, personal space r may vary from person to person, and change between different cultures or environmental settings (Hall, 1963).

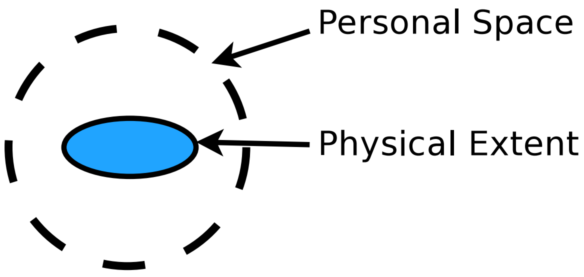


Figure 5.3: Comparison of a tight oval bounds on the physical space (blue oval) to the larger personal space which is used for planning (dashed circle).

Table 5.2 summarizes the three parameters introduced by RCAP.

5.4.2 Static Obstacles and Global Navigation

The RCAP formulation can account for static obstacles in the same collision avoidance framework. Small obstacles such as trees, tables, and poles can be accounted for by treating them much as an agent who remains still. The one important difference is that an obstacle will not share the responsibility in collision avoidance, forcing the agent to account for the entire avoidance vector $\mathbf{u}$ (the minimal change in velocity required to resolve the collision). Formally, with respect to an obstacle O, linear ORCA constraints can be computed as:

$$ORCA^{\tau}_{A|O} = \{\mathbf{v} \mid (\mathbf{v} - (\mathbf{v}^{\text{opt}}_{A} + \mathbf{u})) \cdot \mathbf{n} \geq 0\} \tag{5.9}$$

where $\mathbf{u}$, as above, is smallest vector moving the velocity out of $VO^{\tau}_{A|O}$ and $\mathbf{n}$ is the normal of the VO at this closest point.

When navigating around obstacles, it is assumed that agents are not subjected to the mental processing constraints of T_{obs}. This is because the obstacles don't move and are unlikely to surprise the agents. However, the kinodynamic constraints on motion are still enforced, as is the concept of a small personal space buffer around an agents physical extent.

If agents need to navigate around very large obstacles (such as walls, and buildings), treating them the same way as small obstacles can lead agents to get stuck in local minimums. In these cases, a roadmap is used for global navigation as in (van den Berg et al., 2008b). Agents are given an immediate goal of the closest node on the roadmap which leads to their goal.

5.4.3 Algorithm Overview

Algorithms 1 and 2 provide an overview of the entire RCAP algorithm in pseudocode. In both these algorithms T is the current time in the simulation. Two helper functions are also used. The first is `ClampVelocity()` which implements Equation 5.8. The second is `LinearProgramming(`$goal$`, `$constraints$`)` which computes the velocity that is closest to the $goal$ velocity and does not violate the $constraints$.

Algorithm 1: RCAP Algorithm

```
Input: Agent $A$, List of neighbors $\mathcal{B}$
Output: $v_{new}$ - The new velocity for $A$
1 $ORCA^{\tau}_{A} \leftarrow \emptyset$;
2 foreach $B \in \mathcal{B}$ do
3     if $T > T^{B|A}_{sight} + T_{obs}$ then
4         $ORCA^{\tau}_{A} \leftarrow ORCA^{\tau}_{A} \cap ORCA^{\tau}_{A|B}$
5 $\mathbf{v}_{computed}$ ←LinearProgramming($\mathbf{v}_{pref}$,$ORCA^{\tau}_{A}$);
6 $\mathbf{v}_{new}$ ← ClampVelocity($\mathbf{v}_{computed}$, $\mathbf{v}_{old}$,$a^{\max}$)
```

The RCAP algorithm is part of the general loop that updates the simulation, as shown in Algorithm 2. Here, `Neighbors(`A`)` returns all the agents nearby to A. It is also responsible for updating $T^{B|A}_{sight}$ if it is the first time a new agent B was sighted.

5.5 Model Optimization & Training

In order to compute appropriate values for the RCAP parameters T_{obs}, $a^{\max}$, and r data was used from several sessions of two tracked humans interacting with each other in

Algorithm 2: Simulation Update

```
Input: $AgentList$ a list of agents to simulate, $\Delta T$ simulation timestep
$T \leftarrow 0$;
while Simulation is running do
    foreach $A \in AgentList$ do
        $\mathcal{B} \leftarrow$Neighbors($A$);
        $\mathbf{v}^A_{new} \leftarrow$RCAP($A$,$\mathcal{B}$);
        $\mathbf{v}_a \leftarrow \mathbf{v}_{new}$;
        $\mathbf{p}_a \leftarrow \mathbf{p}_a + \Delta T \mathbf{v}_a$;
        $T \leftarrow T + \Delta T$;
```

a motion capture studio provided by (Pettré et al., 2009). This data contains high-quality motion capture paths of two humans as they pass each other in a large lab space. In the experiment, two people start at two randomly chosen corners of a 15m x 15m room. The two participants can initially not see each other, due to the presence of 5m long occluding walls, which serve as obstacles. The participants are simultaneously directed to walk to the opposite corner of the room. After a few meters, the participants will have moved passed the occluding barriers and be able to see each other. At this point the participants have reached the *interaction area* and will need to respond to each other to avoid collisions. The participants continue walking until they reach the opposite corner. This experimental setup is shown in Figure 5.4.

Data was used from 474 different runs of this experiment collected across 5 different days. The data provides a wide variety in the initial conditions of the simulations. Specifically, each time the participants entered the interaction area they do so at different positions, with different velocities, have different average speeds, and walk towards different spots in the opposite corners. All these differences provide a variety of scenarios to compare the trajectories of virtual humans with their real counterparts.

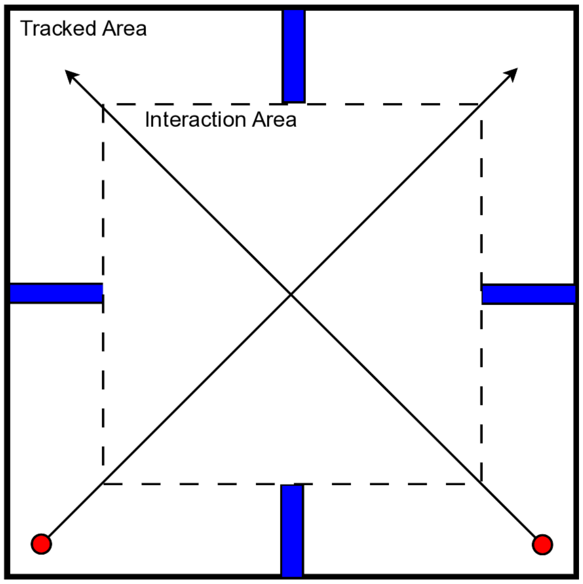


Figure 5.4: **Experimental set-up** Two people (red circles) are placed in a $15m$ x $15m$ motion-tracked lab. Participants start at randomly selected corners, initially unable to see each other due to $5m$ long barriers (blue rectangles). They were simultaneously asked to walk to the opposite corner. A few meters into their path they enter the *Interaction Area* and can see the other participant and react to avoid colliding. *(Drawn to Scale)*

5.5.1 Training Approach

In order to learn the appropriate parameters for the RCAP model, the experimental data was split into training data and validation data. Finding the values of RCAP parameters which best fit the real human data can be viewed as a multi-dimensional optimization problem. A standard simulated annealing method is used to find the optimal values of the RCAP parameters at which the simulations most closely matched the real data.

5.5.1.1 Optimization Metric

In order to optimize the RCAP parameters some metric must be specified for measuring how close any given run of the simulation is to the actual human paths. A poor choice of the optimization metric can lead to undesirable and unrealistic behavior. For example, the simple metric of minimizing the absolute difference in the predicted and actual paths leads to two issues. First, this metric does not penalize collisions between the simulated agents (while the real participants never collided). Secondly, it allows for no natural variations in paths. The choice to avoid someone on the left or right is often symmetric and does an equal job of avoiding collision, however an agent could be penalized heavily by this simple metric. For example, the predicted response might be an exactly mirrored reflection of the actual paths the participants took, but the positions will not be close to each other so the response might be heavily penalized.

To solve this problem, the RCAP approach compares the trajectories of simulated agents to the real participants based on *collision response curves*. The metric is defined as the change in the projected distance of the closest point the two agents will approach. For example, when two people are on a head-on, center-to-center collision path, their projected minimum distance will be at or near zero. This is commonly the case in the experiments when two people first see each other. As people move along and respond to the impending collision, the minimum projected distance will grow as they move away from a collision course. When the minimum projected distance is greater than a person's width, the individuals are on safe trajectories but they still may continue to adjust their velocities because of personal space considerations or other proxemic reasons.

The change of the minimum projected distance as the agents interact is what defines the collision response curve for any run. A graph of a typical response curve is shown in Figure 5.5, which compares the response curves predicted by RCAP to the actual ones generated by the human subjects in a particular trial. The collision response curves were used to train the RCAP parameters using two different procedures. The first time, the parameters were trained by optimizing once over all runs to find values of these parameters for average humans. Secondly, each run was optimized individually in order to train parameters specifically for that run. This produces a distribution of the RCAP parameters across all the participants.

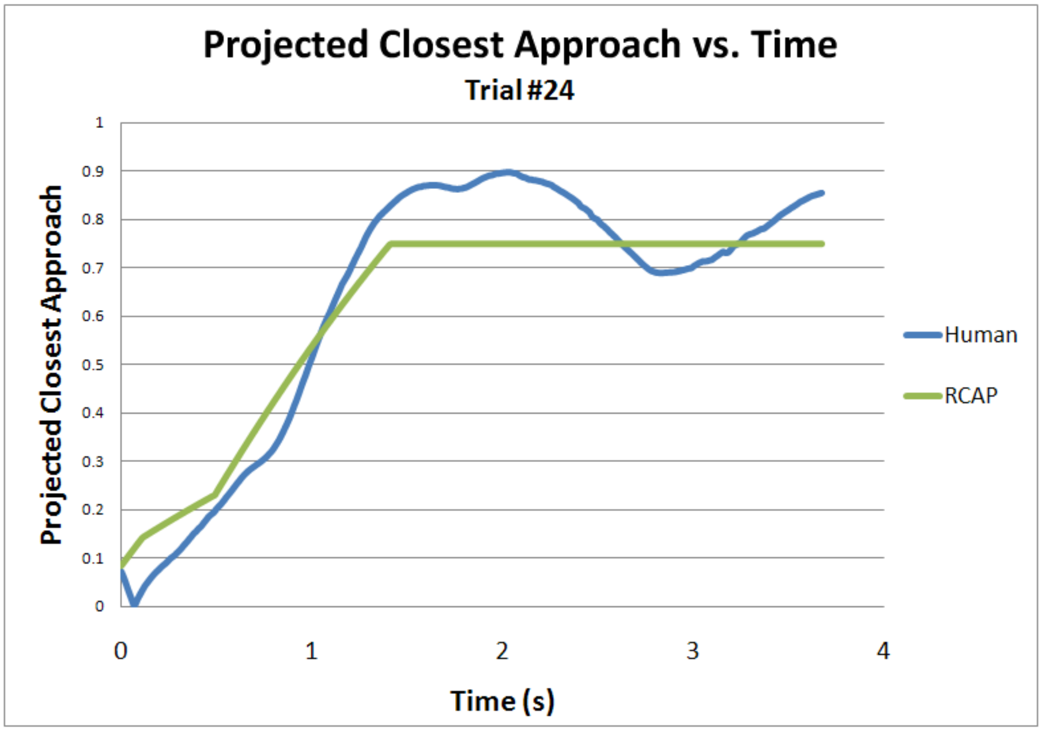


Figure 5.5: Graph of how the distance between the agents at projected closest point between two agents' trajectories changes over time during a single trial. *Blue Line*: Two real people initially start on a colliding path (if unchanged, their centers would be only $.1m$ apart). As the experiment progresses, the people eventually sort out the collision and adopt velocities which will have their centers pass $.7m$ apart, more than far enough to avoid a collision (at least $.5m$). *Green Line*: An RCAP simulation initialized with the above conditions.

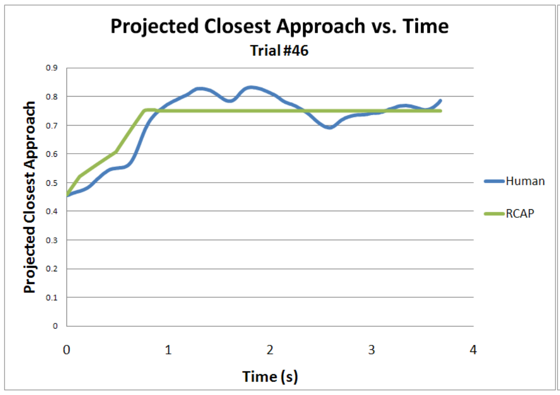
5.5.2 Optimal RCAP Values

As discussed above, one optimization was run over the entire training dataset to compute appropriate average value of the RCAP parameters. This provides a baseline for a typical human (at least among the age ranges present in the experimental data). The values obtained from optimizing over the entire training data set are shown in Table 5.2.

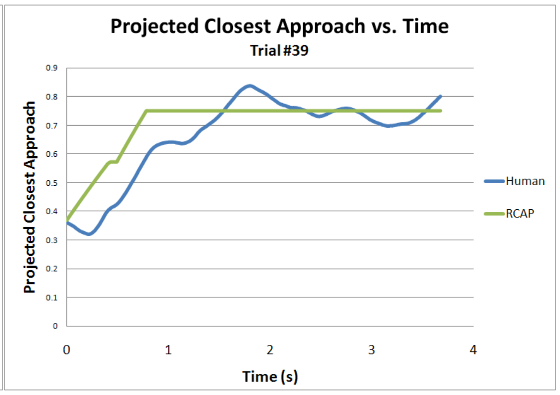
Symbol	Value	Description
r	$0.375m$ (15")	Personal Space (radius)
$a^{\max}$	$0.098\frac{m}{s^2}$	Maximum Acceleration
T_{obs}	$0.486s$	Observation Time

Table 5.2: Table of RCAP constants.

For any given run, the predicted collision response curve can be compared to the actual human response. One such comparison is shown in Figure 5.5, where the blue line charts the response from actual participants and the green line shows RCAP's prediction from the corresponding simulation. Further comparisons of real and predicted collision response curves are shown in Figure 5.6 and 5.7. Two runs are shown where the participants initially are on a collision course (Figures 5.6a and 5.6b), and one run where the participants will pass closely but do not collide (Figure 5.7).



(a)



(b)

Figure 5.6: Closest Approach Graphs for 2 different runs.

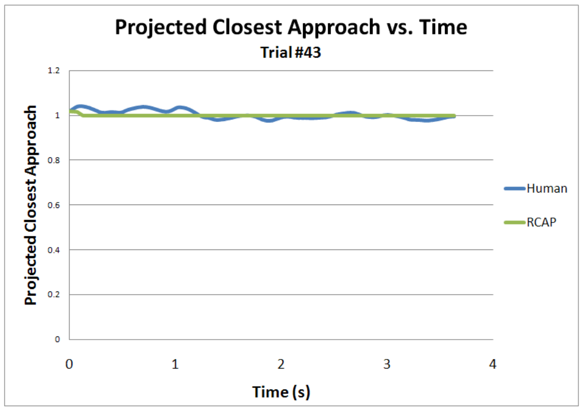


Figure 5.7: The participants' desired velocities do not lead to a collision course, the real and virtual agents both chose to maintain their desired velocities, instead of changing them.

5.5.3 RCAP Parameter Distribution

In contrast to above approach which optimized parameters once over the entire training dataset, it is possible to train individually optimized parameters for each run of users in the study. These per-run results are denoted as “RCAP Tuned” as this process provides an estimate of the RCAP parameters tuned to each pair of individuals. By collecting this data over multiple runs, the variance of these parameters between different participants can be analyzed and used to build a distribution over the various runs. Table 5.3 shows the mean value and standard deviation of each of the parameters.

Symbol	**Value**	**Std. Dev**
r	$0.397m$ (16”)	$0.120m$ (4.2”)
$a^{\max}$	$0.289\frac{m}{s^2}$	$0.445\frac{m}{s^2}$
T_{obs}	$0.504s$	$0.470s$

Table 5.3: Variance of RCAP parameters between participants.

Figure 5.8 shows a graphical depiction of the distribution of the learned personal space r. The parameters follow an approximately bell-shaped distribution. The range of values correlates well with the area defined as *intimate distance* in the proxemics literature (Hall, 1963). This is defined as the distance which is viewed as entirely personal and any entrance into it is viewed as an intrusion. This is similar to the RCAP definition of personal space. Approximately 80% of the tuned values for r fall into the zone of between $0.14m$ - $0.46m$, as suggested by Hall. This suggests the results of the RCAP method are in agreement with the proxemics literature.

Figure 5.9 shows the distribution of the maximum acceleration and reaction time parameters. The distributions of both parameters are strictly positive but have a mean within one or two standard deviations of zero. This prevents the distributions from being a bell-curved and results in both distributions having a positive skew.

To understand the effect of tuning the RCAP parameters per person, Figure 5.10 shows the improvement from using the the per-run, tuned parameters over the generic ones for two

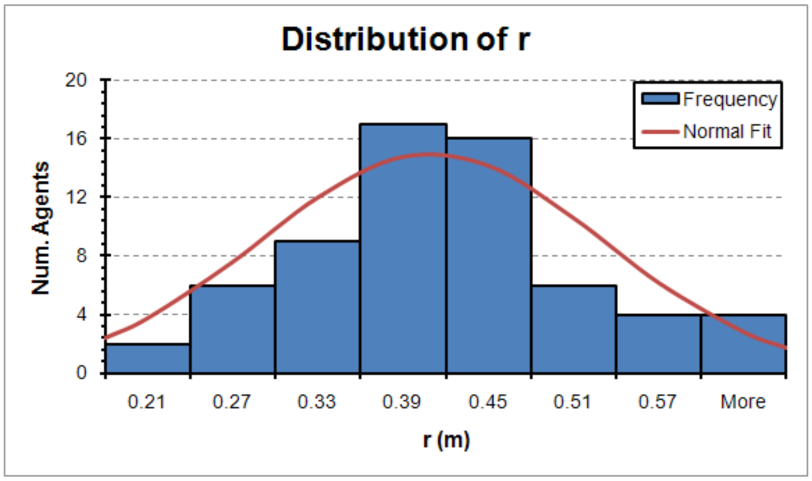
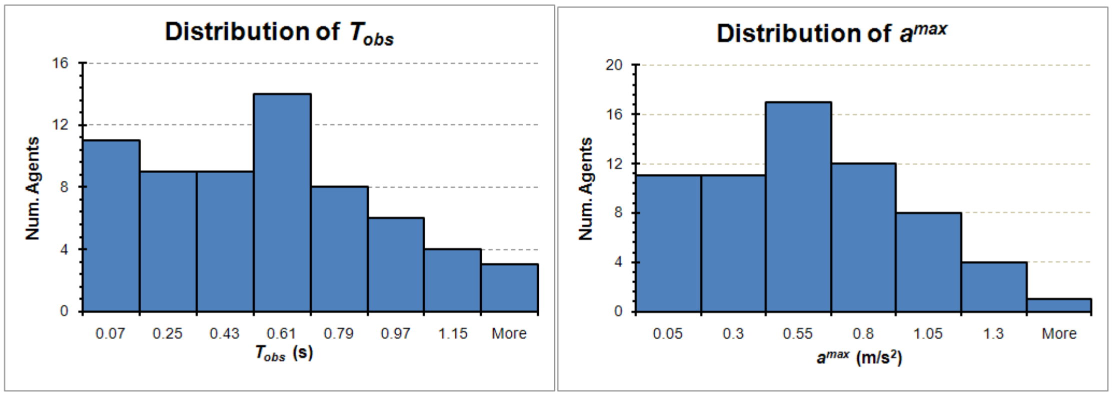


Figure 5.8: Distribution of T_{obs} across all runs.



(a) (b)

Figure 5.9: Distribution of $a^{\max}$, and r across all runs.

sample runs. In Figure 5.10(a), the generic parameters overestimate the amount of personal space desired by the participants. In Figure 5.10(b), the generic parameters underestimate this amount. In both cases, the scenario specific tuned parameters provide higher accuracy in terms of estimating the personal space, the reaction and observation time, and the rate of velocity response in order to more closely match the collision response curves.

5.6 Model Validation

To analyze how close the RCAP model of human motion matches the data collected from real-world experiments, the paths in the validation dataset where compared to those

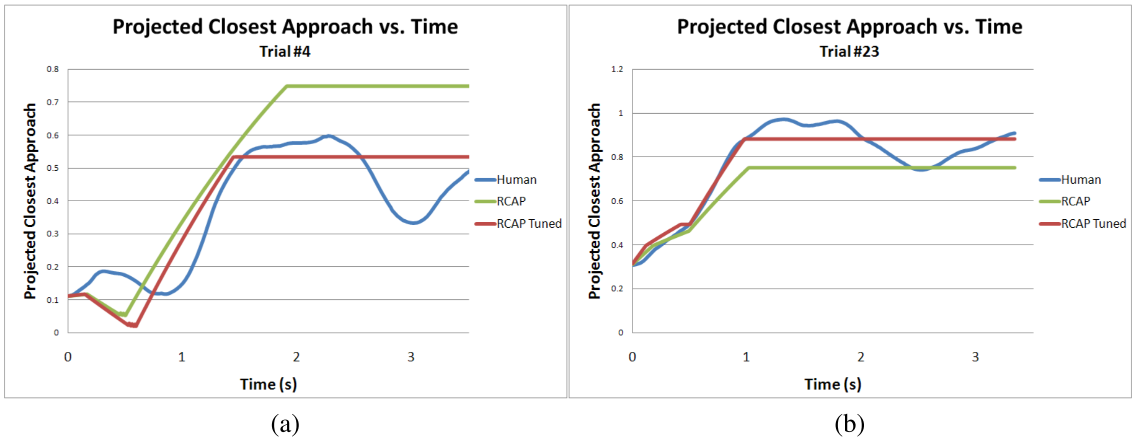


Figure 5.10: Comparison of the true collision response curve (*Blue Line*) to the one predicted by RCAP with generic constants (*Green Line*) and the prediction with constants tuned to the specific runs (*Red Line*).

predicted by the RCAP algorithm. For this comparison the mean values for T_{obs}, $a^{\max}$, and r were used as given in Table 5.2. The results are analyzed qualitatively and quantitatively along three dimensions: the collision response timing, a biomechanical energy consumption based analysis of the trajectories, and on the paths themselves. A five-fold cross validation was also performed to analyze error in the model while reducing over-fitting.

5.6.1 Collision Response Phases

In analyzing the results qualitatively, three distinct phases of RCAP agent motion are apparent. These phases can be clearly seen in the collision response curves such as those in Figures 5.5, 5.6(a) and 5.6(b). The first phase is the *observation phase*, which lasts a little under a second. Here the agents move along at their preferred velocities, without any changes. Secondly, is the *reaction phase*, where the agents have computed an appropriate velocity and take a second or two to achieve it (depending on how far it is from the observation phase velocity). Finally, there is the *maintenance phase* where the agents maintain their collision-free velocities.

Participants in the experiment often show a similar means of response to a collision as the three phase response predicted by RCAP. People would at first not react to avoid the collision, then slowly adopt a correct velocity, and finally maintain velocities which generally result to collision-free trajectories. This is in stark contrast to ORCA agents as they instantaneously adopt and maintain a collision-free velocity.

In all cases, there is little change in velocity when there is no imminent collision to avoid, such as in the scenario reported in Figure 5.7.

5.6.2 Biomechanical Energy Consumption Analysis

A more quantitative way to measure how realistic the simulated paths are is to analyze the biomechanical effort implied by the trajectory. As humans move in the environment, they expend energy and turn chemical potential energy stored in their body into the physical kinetic energy of motion. Humans have been shown to walk at speeds which minimize the amount of energy spent walking (Inman et al., 1981). Given an agent's weight, velocities, and path taken, it is possible to calculate how much much energy the agent must have spent walking along that path (Whittle, 2002). This calculation can be performed for both the real and virtual humans, which gives us a means to determine if the virtual agents choose similarly efficient paths as compared to real humans.

Assuming a weight of 70 Kg, over the course of all the runs, the average real human consumed 1,778 joules (J) walking to his or her goal (standard deviation 306 J). During the same runs, the virtual agents consumed 1,770 J (s.d. 307 J). On any given run, the average difference between the energy consumed by real and virtual humans was only 20 J.

5.6.3 Path Similarity

In addition to comparing manner and pacing of collision response as above, the absolute paths taken by virtual agents can be compared to those of the real humans. In general the paths taken are very similar. Figure 5.11 shows a run of the simulation (shown in red) overlaid with the paths that the actual participants took. On average, the simulated and real humans were only 0.168m apart at any time.

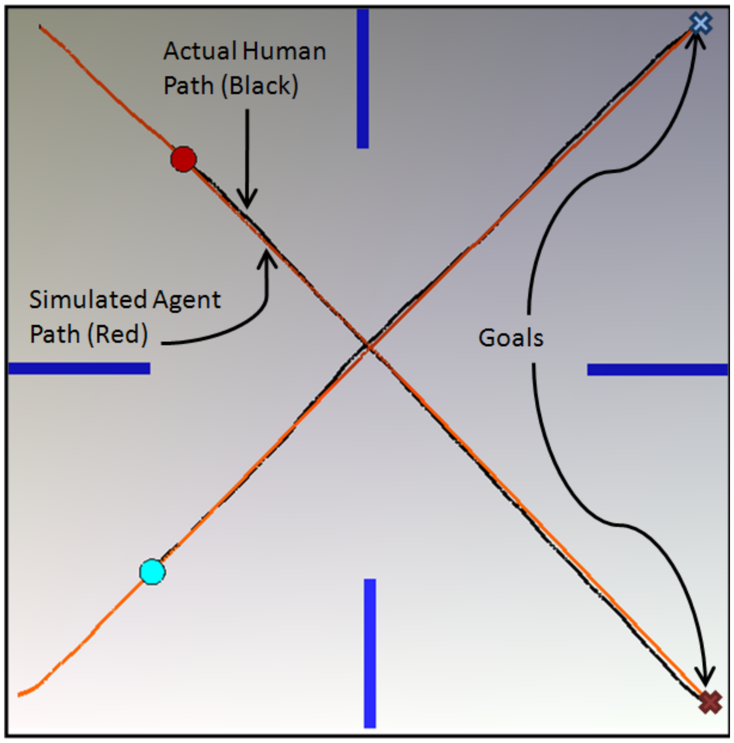


Figure 5.11: A comparison of paths Real Humans vs Virtual Humans. Agents are displayed as circles with their goal for this simulation run show in Xs. The redline shows the path that the simulated humans took, and the black line shows the path of the humans.

Figure 5.12 shows more paths from different initial conditions for the real humans and virtual agents. Even in just these three runs, the six participants invoke a variety of different techniques to avoid collisions with each other including slowing down, speeding up, veering left or right, and keeping the same path while the other person adjusts. Despite this variety, the virtual agents are still able to follow the trajectories of the real humans very closely.

5.6.4 Cross Validation

In order to reduce the effect of overfitting, the above experiments war re-ran using a five-fold cross validation. That is, the model was trained five separate times with a different fifth of the training data removed each time. For each fold, error statistics were computed with respect to the removed data set. In all cases the resulting models show a strong agreement between the predicted paths and the validation data. Averaged across the five folds, the model showed an average root-mean square error (RMSE) of 0.217m (8.5”).

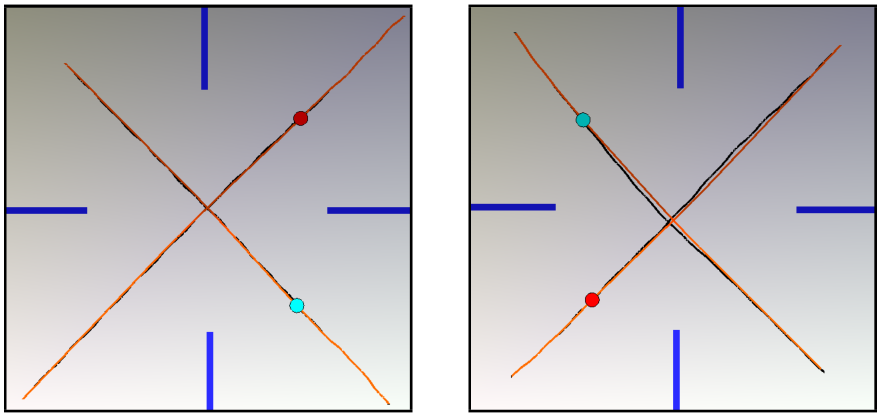


Figure 5.12: Comparisons of paths Real Humans vs Virtual Humans from two additional runs. *Black lines*: Real humans' paths, *Red lines*: Virtual Agents' paths

Additionally, the positions of the modeled agents and actual humans physically overlapped 95.1% of the time.

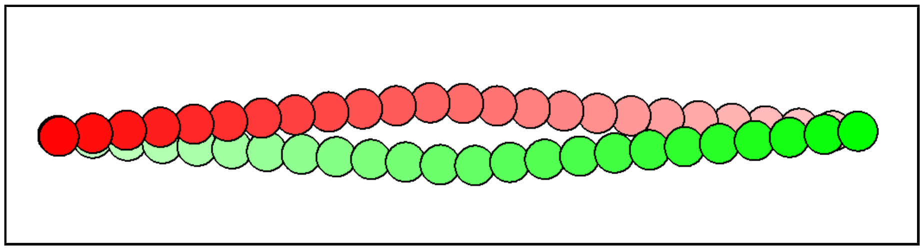
5.7 Simulation Results

This section presents results of various simulations created with RCAP agents. I discuss both how to handle simulations with more than two agents, and how to generate data driven crowd simulations using RCAP.

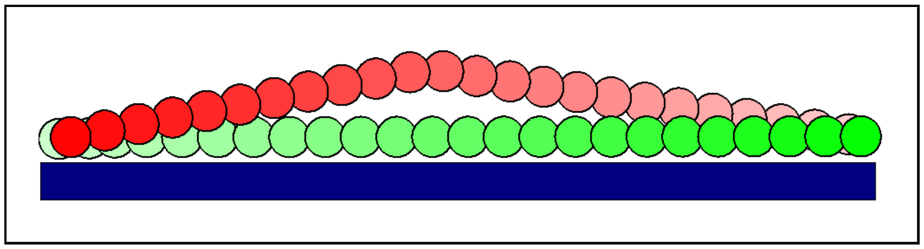
5.7.1 Multi-person Simulations

While tuned on two-person interactions, RCAP scales without modification to handle complex simulations with any number of individuals. It can be further adopted to handle more complex scenarios involving obstacles by simply adding a new linear constraint which forbids moving towards or into an obstacle fast enough to reach it in time τ, as described in Section 5.4.2.

When RCAP virtual agents pass each other without a wall nearby, they chose reciprocating paths, sharing the burden of collision avoidance (Figure 5.13(a)). However, when an agent is too close to a wall, they may not be able to reciprocate. Here, the agent with enough free space to respond to the collision automatically adapts and eventually takes the entire responsibility for avoiding the collision (Figure 5.13(b)). I consider this a reasonable behavior as it efficiently avoids collisions between the two agents.



(a) No Wall



(b) Wall

Figure 5.13: Time-lapse diagram of agent positions. (a) Two RCAP agents exchange positions. The agents reciprocate, each taking half the responsibility. (b) A wall prevents the green agent from turning away from the collision. The red agent automatically accommodates, eventually taking full responsibility for avoiding the collision.

5.7.1.1 Data-driven Crowd Simulations

When simulating crowds or large groups of people it is often desirable to have a natural diversity in how the individuals in the crowd behave. This task is known as heterogeneous crowd simulation. This diversity in the crowds can be modeled in a natural manner by drawing values from the distributions given in Figures 5.8 and 5.9 to set the RCAP parameters for the various individuals in the simulated crowd. Repeating the same process with a different sample population would results in a different simulation which reflects the new population

Figure 5.14 shows a trace of the paths from a simulation with 6 agents with RCAP parameters drawn from the distributions shown in Figures 5.8 and 5.9. Each agent is trying to get across the circle to the other side from where it started (the antipodal position). Agents

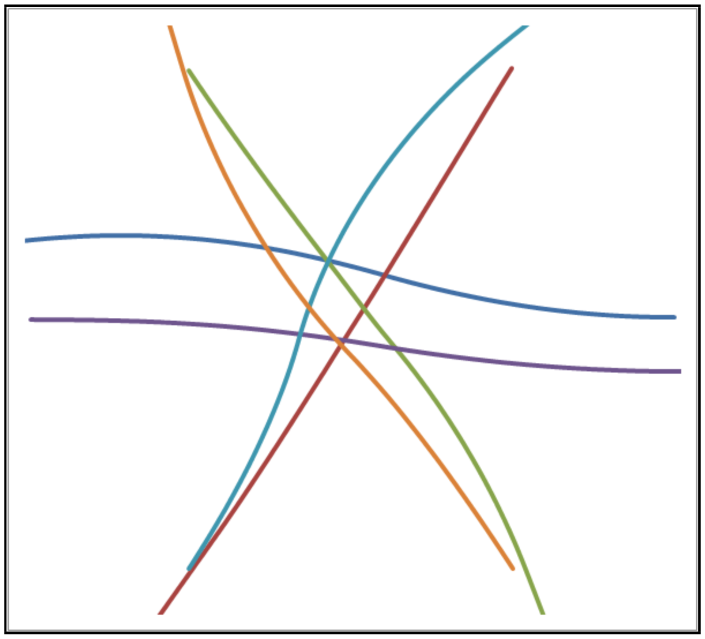


Figure 5.14: Trace of a 6 agent simulation. Agents follow smooth, simple, curved paths similar to those from the trials with humans. No agents collide.

are able to negotiate around each other without collisions while maintaining smooth, gentle curving paths.

RCAP can be extended to perform simulations with a large number (hundreds or thousands) of agents. To test the performance in these larger scenarios, a simulation of thousands of agents exiting an office environment was ran. Again the parameters for these agents are drawn from the distributions shown in Figure 5.8 and 5.9. A snapshot from this simulation is shown in Figure 5.15. This large simulation ran at real-time rates (about 80 ms/frame).

5.8 Comparison & Analysis

This section analyzes RCAPs performance and provides a qualitative and quantitative comparison with other collision avoidance techniques.

Figure 5.15: Snapshot from a simulation of 1,000 people evacuating an office environment.

5.8.1 Path Analysis

RCAP performs extremely well in its ability to match human-like paths. In the majority of cases, the difference between trajectories traveled by the real and virtual agents is rather low. In fact, the trajectories of the virtual agent and the real human physically overlap 94.7% of the time, *i.e.* the distance between paths is less than the person's physical radius.

The biomechanical comparison discussed in Section 5.6.2 can help in "quantifying" the naturalness of the paths. The amount of energy consumed during walking is an indirect measurement of how efficient the simulated walking is in terms of calories consumed. By this measurement, RCAP agents tend to choose paths that were nearly as efficient as those of real humans. The maximum difference in energy consumed is about 1% of actual energy consumed and 0.1% when averaged out over all the runs. The efficiency of human motion in avoiding collisions is well captured by the RCAP model for these benchmarks.

The collision response graphs from Section 5.5 can also be used to analyze how human-like the motion produced by the RCAP model is. Figures 5.5 and 5.6 show the clear improvement that RCAP has over ORCA in terms of modeling how humans respond to collisions. The response of ORCA agents was sudden and immediate, thus making it a poor method for modeling human motion. The three-stage response of RCAP agents provides a fairly good match for human response. The most significant deviation from real humans comes in the *maintenance phase* where real humans continue to have slight changes in their velocity during the maintenance phase rather than keeping it constant.

Beyond reacting correctly when there is a collision, correctly choosing not to react when there is no collision is also important. The RCAP model can also handle such a case. One example is shown in Figure 5.7. Here the two people start with collision-free velocities. Real humans realize that they were not in danger of colliding and maintain about the same velocity. RCAP agents correctly reproduce this behavior.

5.8.2 Simulation Analysis

5.8.2.1 Behavioral Analysis

The RCAP simulations produced smooth, natural motion for each agent. Multiple agents were able to navigate around each other successfully and could cope with the presence of obstacles while avoiding each other. The RCAP model to drive crowd simulations with thousands of agents and still produce smooth, collision-free motion.

Additionally, across several different validation metrics (both qualitative and quantitative) that paths resulting from RCAP have shown to match those taken by humans. Compared to real humans, RCAP agents take similar absolute paths that would expend similar amounts of energy and have similar collision response curves. This evidence helps support both the validity of RCAP, and its generality as a model of human collision avoidance.

5.8.2.2 Performance Analysis

Computationally, RCAP adds almost no additional overhead over ORCA. The simulations were able to run at the same high speeds of the original ORCA implementation. A $5,000$-person variant of the office evacuation demo took only 80 ms per timestep to compute the new paths. If higher performance or larger simulations were desired, an implementation that exploits data-level parallelism and vector processing units would be possible, similar to the method proposed in (Guy et al., 2009) and discussed in Chapter 2.

5.8.3 Comparison to Other Methods

RCAP can be compared against several of the previous models for human motion discussed in Section 5.2. As compared to these models, RCAP does a better job in terms of modeling how humans respond to each other while navigating around each other. By explicitly accounting for human traits and limitations in their motions, RCAP can more closely model how humans react to various collision conditions.

A detailed comparison for a specific run from the Locanthrope data is shown in Figure 5.16. In this run agents were initially on a collision course and need to respond to avoid the collision. The human participants show the typical pattern of little initial reaction, followed by an avoidance phase, then a maintenance phase (Figure 5.16 - blue line) The graph comparing the collision response curves of RCAP, ORCA, and the Helbing Social Force Model all initialized with the same start and goal positions and velocities.

In contrast to RCAP, the Helbing Social Force Model works by applying repulsive forces when agents get too close (Helbing et al., 2005). This approach leads to collision responses which poorly match that of real humans. This poor match is because in Helbing's model, agents do not react to collisions until they are very close to each other, this causes the reactions to come too late and requires the agents to turn too quickly, as compared to a natural human response. This lack of anticipation can be seen in Figure 5.16 (light blue line) where the Helbing agents have almost no collision response for over 3 seconds at which point they drastically swerve to avoid collisions.

ORCA (Figure 5.16 - dashed purple line) and its predecessor, (van den Berg et al., 2008a) RVO model (not graphed) perform nearly identically for simulations of two agents. While they perform collision avoidance well, both suffer from the same fundamental limita-

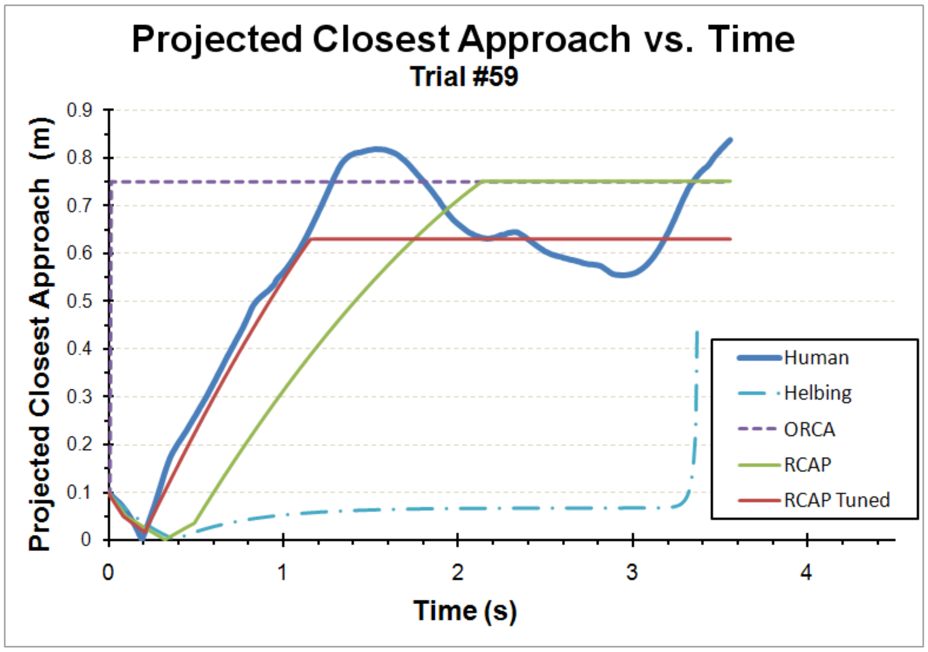


Figure 5.16: A Comparison of RCAP, ORCA, and Helbing Social Force Model to real human data during a single trial.

tions of not explicitly modeling human characteristics and physical capacities. As can be seen in Figure 5.16, the reaction to potential collisions happens too suddenly and at too fast a rate.

While ORCA agents respond to collisions too early, and Helbing agents respond too late, RCAP agents lie in-between and model human collision response more closely. Other approaches have different issues. For example, the Reynold's steering model works by checking each agent to see if they are heading towards a collision, and if so, turns them away a small amount and checks again (Reynolds, 1999). This model fits the real-human data better than Helbing's but is still limited due to the lack of implicit cooperation between the agents. This allows the agents to alternate between wide swings of overreacting and underreacting to potential collisions. A further analysis of these approaches applied to the Locanthrope data can be found in (Pettré et al., 2009).

In the same study that collected the Locanthrope data, (Pettré et al., 2009) proposed a model for human motion using geometric and numeric techniques. This proposed model matches the human data well and shows a three-phase collision response similar to that displayed by the RCAP model. However, this technique is more complex than RCAP, requiring more free parameters and significant computational effort to model inter-agent collision avoidance. As a result, (Pettré et al., 2009) method can be up to three orders of magnitude slower than RCAP, taking 16ms to compute a new velocity for just two agents. RCAP is more appropriate in terms of performance and simplicity for simulations involving a large number of agents and in resource-limited situations such as video games and virtual reality environments.

5.9 Summary and Conclusions

This Chapter presented a new technique to extend ORCA into a simple yet effective model for human collision avoidance. The resulting RCAP model successfully captures key features of human collision avoidance. Agents maintained course when appropriate and moved to avoid collisions when necessary. The virtual humans chose paths which were equally efficient as the paths walked by real humans, given the same initial conditions. The computed paths followed the trajectories traveled by the real humans very closely. Most importantly, the virtual humans responded to collisions at the same rate and in the same manner as real humans did in a variety of experiments across multiple data sets. The method can be used to both model two-person interactions and generate data-driven simulations.

5.9.1 Limitations

There are of course, several aspects of human motion that this model does not capture. For example, there are some important secondary motion effects when walking, such as when humans move on a straight path their center of mass does not move directly forward but rather shifts slightly from side-to-side as their weight shifts from foot to foot while walking. This phenomena is completely missed by the RCAP model but can be seen in the paths traced by the real humans.

There are also some more subtle aspects of the human collision response curves that are not well captured by RCAP. For example, Figures 5.6, 5.10(a) and 5.16 show a noticeable dip during the *maintenance phase* of the human's collision response curve, which is not present in the RCAP model. However, the source of this dip is not clear, and it is not seen across all trials (it appears very weakly in or absent from Figures 5.6(a), 5.6(b) and 5.10(b)). I suspect this might be due to mismatches in personal space between the participants, but more study is needed on this issue.

Additionally, this work only tries to model typical, non-panic scenarios often captured in the lab experiments. Human emotions and personality have a significant effect on how people respond to each other. Capturing this effect could be important in trying to accurately reproduce scenarios such as panicked evacuation. More discussion of modeling personality will be presented in Chapter 6.

Lastly, each human is represented with a very simplified state (e.g. a 2D circle representation). Other issues related to modeling the high degrees of freedom of the full human skeleton are ignored. A fully accurate model would need to take into account factors such as low-level gait analysis, position in stride, and foot placement.

5.9.2 Future Work

Beyond addressing the above limitations, there are many other avenues for future work. Further studies are needed to determine the best shape and positioning to represent the personal planning space. For example, some proxemics research suggests anisotropic shapes such as ellipses might perform better than circles. Finding efficient ways to handle the resulting orientation dependencies is an interesting and challenging problem.

The validation present here has focused only on scenarios with only a small number of people and obstacles. In general validation should be extended to large crowds. An important issue for validating simulations of large crowds is that due to the large number of complex interactions, the stochastic nature of human motion can cause extreme variations in crowd motion. This will cause many of the validation metrics used in this chapter (such as path similarity) to be inappropriate for large crowds and devising new metrics appropriate for crowds will be an important area. Discussion about ways to validate against large scale crowd data is presented in Chapter 7

CHAPTER 6
Data-driven Simulation of Variations in Personality

6.1 Introduction

When simulating heterogeneous crowds one of the most important aspects is the variation between individuals in the crowds. In fact, according to *Convergence Theory*, crowd behavior is not a product of the crowd itself, rather it is carried into the crowd by the individuals (Turner and Killian, 1987). As a result, it is important to accurately model the behavior and interactions among the individuals to generate realistic, heterogeneous crowd behaviors.

In terms of modeling the behavior of individuals within a crowd, even simple tasks, such as walking toward a given destination, involve several complex decisions such as what route to take and the various ways to avoid collisions with obstacles and other individuals. As a result, different people will achieve the same goal in different manners. While there are many factors that govern people's overall behaviors, such as biological and developmental variations, I focus on capturing the portion of these variations that are due to differences in underlying personality.

In general, categorizing the variety of personalities that humans exhibit is a difficult and multifaceted task. While many psychologists have proposed different models to organize this variation in personality, there are limitations in their ability to capture all types of human personality using a single classifying model (Harvey et al., 1995; Reise et al., 2000). In fact, personality can be defined as the interplay between maintaining goal-directness while responding to the demands of the current situation (Pervin, 2003). Rather than trying to directly encode this complex interplay by hand, these personalities are characterized based on data from a user study which asked participants to describe the perceived behaviors of individual agents in computer-generated crowds. sThis chapter focuses on the problem of generating heterogeneous crowd behaviors by adjusting the simulation parameters to emulate personality traits of individuals within a crowd and evaluate the effects of individual personalities on the overall crowd simulation. The approach is based on Personality Trait Theory, which proposes that complex variations in behavior are primarily the result of a small number of underlying traits. It draws on established models from Trait Theory to specify these variations for each individual. The well-known Eysenck 3-Factor personality model (Eysenck and Eysenck, 1985) is used to establish the range of personality variation. This is a biologically-based model of three independent factors of personality: Psychoticism, Extraversion, and Neuroticism. This so-called PEN model has inspired other similar personality models, most famously the Big-5 or OCEAN personality model (Costa and McCrae, 1992), which proposes five independent axes of personality based on a factor analysis of user responses. The OCEAN model has been previously used as framework for exploring variations in crowd simulations (Durupinar et al., 2008).

6.1.1 Main Result

This main result of this work is an efficient approach to create and control the perceived personalities of agents in a crowd simulation. I present a mapping between the low-level simulation parameters and high level behavior descriptors. This mapping is used to control the extent that agents exhibit various degrees of aggressive, shy, tense, assertive, active, and impulsive behaviors. These parameters are also placed in the context of the PEN personality model. Additionally, I propose a novel two-dimensional factorization of personality traits derived from the empirical study results on perceived personalities in computer-generated crowds. These mappings are used to generate heterogeneous crowd simulations with different, predictable perceived agent personalities.

6.1.2 Organization

This chapter is organized as follows. Section 6.2 highlights related work in crowd simulation and behavior modeling. Section 6.3 gives a brief overview of established personality models and Trait Theory. The user study on perceived personalities in Sec. 6.4, and Sec. 6.5 uses the results to compute the mappings. Section 6.6 demonstrates the resulting behavior of agents simulated with various personalities.

6.2 Previous Work

6.2.1 Human Behavior Modeling

Many researchers have proposed approaches to simulate crowds that can closely model human behavior. Funge et al. (Funge et al., 1999) proposed using Cognitive Modeling to allow agents to plan and perform high level tasks. Shao and Terzopoulos (Shao and Terzopoulos, 2005) proposed an artificial life model with several components, that enabled agents to make decisions at both the reactive/behavioral and proactive/cognition levels of abstraction. Yu and Terzopoulos (Yu and Terzopoulos, 2007) introduced a decision network framework for behaviorally animated agents that was capable of simulating interactions between multiple agents and modeling the effect of different personalities.

Other approaches have directly incorporated personality models into crowd simulations. Durupinar et al. (Durupinar et al., 2008) suggested a method to vary the parameters of the HiDAC simulation model based on the OCEAN personality model by choosing a plausible mapping between OCEAN personality factors. Salvit and Sklar (Salvit and Sklar, 2011) created a testbed world based on termites collecting food where they demonstrated a variety of food-gathering patterns based on varying parameters of the MBTI personality model.

Perceptual or user studies have been used to improve crowd behaviors and rendering. McDonnell et al. (McDonnell et al., 2009) utilized perceptual saliency to identify important features that need to be varied to add visual variety to the appearance of avatars. McHugh et al. (McHugh et al., 2010a) investigated the effect of an agent's body posture on their perceived emotional state. Durupinar et al. (Durupinar et al., 2011) evaluated their method to model the OCEAN personality with a user study.

6.2.2 Modeling Crowd Styles

Previous approaches have used data-driven methods to produce simulated crowds which behaved with a certain trait or "style". These methods commonly train models for crowds based on input video data. For example, Lee et al. (Lee et al., 2007) used data-driven methods to match recorded motion from videos by training a group behavior model. Ju et al. also proposed a data-driven method which attempts to match the style of simulated crowds to those in a reference video (Ju et al., 2010) .

6.3 Personality Models and Trait Theory

Psychologists have proposed various ways of characterizing the spectrum of personalities exhibited by humans. Several theories focus on aspects of personality that show cross-situational consistency, i.e. behavior aspects that are relatively consistent over time and across various situations. While there are many sources of variety in behavior, psychologists have proposed methods to categorize and organize these variations. This work builds on Trait Theories of personality, a broad class of theories which categorizes people's behavior based on a small number of personality traits (Pervin, 2003).

6.3.1 Trait Theory

A personality trait is an habitual pattern of behavior, thought or emotion. While humans display a vast number of different traits, a small number of these traits are believed to be central to an individual's basic personality. Trait theories identify these primary traits, which can be used to describe variations in personality; an individual's personality is described based on a score of how strongly or weakly they exhibit each of these primary traits.

One of the most well established trait theories is the Eysenck 3-factor model (Eysenck and Eysenck, 1985). This model identifies three major factors which categorize personality: Psychoticism, Extraversion, and Neuroticism (commonly referred to as PEN). An individual's personality is identified according to what extent they exhibit each of these three traits. The Psychoticism factor is a measure of a person's aggression and egocentricity. The Extraversion factor is a measure of social interest and higher levels of extroversion are associated with more active, assertive and daring behaviors. Finally, the Neuroticism factor is a measure of emotional instability which can correspond to shyness and anxiety (Eysenck and Eysenck, 1977). Each of Eysenck's three PEN traits have been linked to biological basis, such as the levels of testosterone, serotonin and dopamine present in one's body.

6.3.2 Factor Analysis

The Eysenck 3-factor model is one of several different trait theories. Other theories have used different methods for classifying the fundamental dimensions of human personality. A particularly successful method of identifying basic personality traits comes from applying *factor analysis* to various user studies where participants use common personality adjectives to describe the behaviors of themselves or others in various situations (Cattell and Eber, 1972). Factor analysis is the process for determining which small number of unobserved latent variables can describe the behavior of a large number of observed variables. In the context of personality trait theory, the observed variables are the many different adjectives that people use to describe personalities, while the latent variables are a smaller number of axes which explain the correlation in the way people use these personality describing adjectives. An example latent variable might be extraversion, which is associated with the uses of the adjectives outgoing, active, and assertive.

Costa & McCrae (Costa and McCrae, 1992) applied factor analysis to data collected from various personality studies and suggested five primary factors of personality which they dubbed: “Openness to experience”, “Conscientiousness”, “Extraversion”, “Agreeableness”, and “Neuroticism” (commonly referred to as OCEAN). While the OCEAN model is very popular, other researches have applied factor analysis to similar user studies and found different factors or different numbers of factor (e.g. the 16 Personality Factor model (Cattell and Eber, 1972)). Additionally, many studies have shown that the five OCEAN factors are not fully orthogonal (i.e. not independent from each other) (Draycott and Kline, 1995). Furthermore, OCEAN, along with other models such as PEN, deals with personality in the context of general human behaviors. The work presented in this chapter seeks to study personality specifically within the context of crowd simulations. To that end, a similar factor analysis technique is applied to user responses about personalities perceived in computer-generated crowd simulations.

6.4 Behavior Perception User Study

The goal of this work is to understand how varying parameters in a crowd simulation affects the perceived behavior of agents in the crowd. To this end, several low-level parameters commonly used in crowd simulations were investigated in the study: preferred speed, effective radius (how far away an agent stays from other agents), maximum number of neighbors affecting the local behavior of an agent, maximum distance of neighbors affecting the agent, and planning horizon (how far ahead the agent plans). Many agent-based crowd simulation methods use these or similar parameters to compute the mutual interaction between agents.

A data-driven approach is used to derive a mapping between simulation parameters and perceived agent behaviors based on the results of this perceptual study. This approach has at least two advantages over trying to hand-tune a plausible mapping. First, it ensures that the perceived personality results are based on the input of a wide range of study participants.

Second, it allows for richer, more complex mappings than would otherwise be possible with hand-tuning plausible parameters.

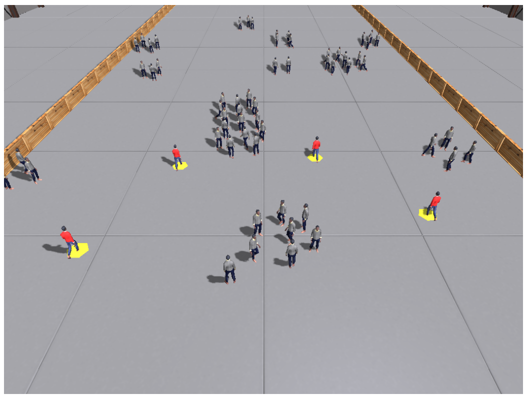
In designing the study, there were multiple goals the system needed to satisfy. First, the ability to produce mappings to several common adjectives used to describe individuals in crowds, such as “shy”, “assertive” and “aggressive”. Second, the ability to produce a mapping from simulation parameters to an established psychological theory, such as the Eysenck’s PEN model. Finally, the gathered data should be sufficiently rich enough to support a factor analysis that enables us to extract underlying latent variables describing the space of personality seen in crowd simulations.

6.4.1 Method

To achieve the above stated goals, I designed a user study, which allowed participants to describe behavior in crowd simulations using several adjectives. The study involved 40 participants (40% female) between 24 and 64 years old, with an average age of 33 years (std. dev. of 12 years). In this study, participants were asked to view three different scenarios of computer generated crowds. In each video, several agents were highlighted to be the focus of user questions. Animations of these scenarios can be seen in the supplementary video. All simulations were created using the publicly available RVO2 Library for multi-agent simulation (van den Berg et al., 2011).

Fig. 6.1 shows a still from each of the scenarios used in the study. The first scenario was the Pass-Through scenario, where four highlighted agents move through a cross-flow of 400 agents. Second was the Hallway scenario where four highlighted agents move through a hallway past 66 other agents, who are in several small groups. Lastly, was the Narrowing Passage scenario where 40 highlighted agents walk alongside 160 other agents towards a narrowing exit. In all cases, the non-highlighted agents were given the default parameters from the simulation library, which mostly results in homogeneous behaviors of the agents in

(a) Pass-Through Scenario


(b) Hallway Scenario

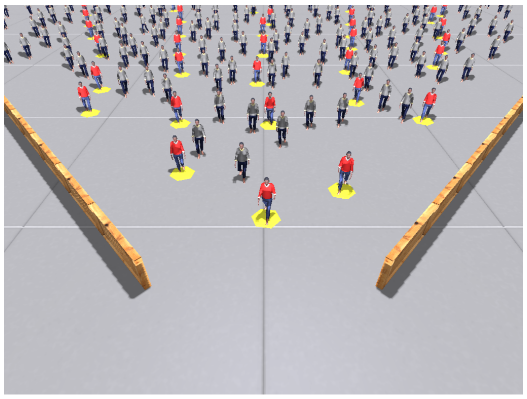

(c) Narrowing Passage Scenario

Figure 6.1: Three crowd simulation scenarios. (a) Four highlighted agents move through crowd. (b) Four highlighted individuals move through groups of still agents. (c) 20 highlighted individuals compete with others to exit through a narrowing passage.

the simulation. The highlighted agents all share the same simulation parameters, that are randomly chosen for each question given to the participants.

In all scenarios, the highlighted agents are displayed wearing a red shirt with a yellow disc beneath them to allow them stand out in the crowd. Each participant was shown several videos for each scenario with randomly chosen simulation parameters for the highlighted agents. Each video was shown side-by-side with a reference video in which all the agents were simulated using the default set of parameters of the library. This “reference video” was the same for each question involving the same scenario to provide a consistent baseline for comparison.

The participants were asked to rate how the highlighted agents behaved in comparison to those in the reference video. Participants were asked to describe the differences in behavior as being more or less "Aggressive", "Shy", "Assertive", "Tense", "Impulsive" and "Active". These particular six adjectives were chosen both because they are useful in describing behaviors of individuals in crowds, and can span the space covered by the PEN model, with at least two adjectives for each PEN trait (Pervin, 2003). Participants then rated each crowd video in terms of all six personality adjectives on a scale from 1-9, with 9 meaning, for example, "much more assertive" than the references video, 5 meaning "about as assertive" and 1 meaning "much less assertive". The participants were allowed to re-watch the videos as many times as they felt necessary, and could go back and forth between questions within a section and revise their answers if desired.

To generate the highlighted agents in the question video the following simulation parameters were randomly chosen: maximum distance to avoid neighbors, maximum number of neighbors to avoid, planning horizon, agent radius, and preferred speed. The random parameter values were shared by all the highlighted agents in each video. The range of the sampled values is shown in Table 6.1.

Parameter	Min	Max	Unit
Max. neighbors dist.	3	30	m
Max. num. neighbors	1	100	(n/a)
Planning horizon	1	30	s
Agent radius	0.3	2.0	m
Preferred speed	1.2	2.2	m/s

Table 6.1: Range of simulation parameters.

For this study, approximately 100 videos were pre-generated for the 3 different scenarios with random values for each of the 5 simulation parameters. Each subject was asked to rate behaviors in several videos randomly chosen from this pool. To keep subjects engaged, the number of videos shown to each participant was limited to 6 randomly chosen clips from each of the 3 different scenarios (18 videos total); users were given the option to skip videos and watched an average of 15 video each. Each video was accompanied with 6 questions, which resulted in a total of approximately 3,600 data points mapping each set of input parameters to perceived levels of various personality traits.

6.5 Data Analysis

Given the large number of data points from the study, it is possible to derive a mapping of the relationship between crowd simulation parameters and the perceived personality of the agents. A linear model is used for the mapping, though other forms of regression are possible.

6.5.1 Mapping Perceived Behaviors

Using a QR decomposition with column pivoting, a linear regression between simulation parameters and perceived behaviors is found. The difference between the given agents' parameters and those of the agents in the reference video serves as an input to the regression model. This removes the need to compute an offset as part of the regression. The input is normalized by dividing each parameter by half of its min-to-max range to increase the numerical stability of the linear regression.

The mapping then takes the following form:

$$\begin{pmatrix} Aggressive \\ Assertive \\ Shy \\ Active \\ Tense \\ Impulsive \end{pmatrix} = A_{adj} \begin{pmatrix} \frac{1}{13.5}(Neighbor\ Dist - 15) \\ \frac{1}{49.5}(Max.\ Neighbors - 10) \\ \frac{1}{14.5}(Planning\ Horiz. - 30) \\ \frac{1}{0.85}(Radius - 0.8) \\ \frac{1}{0.5}(Pref.\ Speed - 1.4) \end{pmatrix}$$

Using a linear least-squares approach on the user study data produces the following 6-by-5 matrix A_{adj}:

$$A_{adj} = \begin{pmatrix} -0.02 & 0.32 & 0.13 & -0.41 & 1.02 \\ 0.03 & 0.22 & 0.11 & -0.28 & 1.05 \\ -0.04 & -0.08 & 0.02 & 0.58 & -0.88 \\ -0.06 & 0.04 & 0.04 & -0.16 & 1.07 \\ 0.10 & 0.07 & -0.08 & 0.19 & 0.15 \\ 0.03 & -0.15 & 0.03 & -0.23 & 0.23 \end{pmatrix}$$

Though A_{adj} is a not a square matrix, it it still possible to compute a mapping from high-level behaviors specified by the adjectives to simulation parameters by taking its pseudoinverse A_{adj}^{+}. In this way, the perceived change in behavior of an agent can be predicted as the simulation parameters are adjusted to achieve the desired behavior for each agent.

6.5.2 Mapping Parameters for the PEN Model

In addition to building a mapping for each of the six personality adjectives individually from the study, a similar procedure to build a mapping for the 3-factor PEN model. The adjectives from the user study can be mapped to the three PEN factors. The correspondence of adjective to PEN factors found in Pervin (Pervin, 2003) was used, and summarized in Table 6.2.

Trait	Adjectives
Psychoticism	Aggressive, Impulsive
Extraversion	Assertive, Active
Neuroticism	Shy, Tense

Table 6.2: Excerpt from the mapping between adjectives and PEN factors given in (Pervin, 2003) and used create A_{pen}.

Like the personality adjectives, a linear mapping for the PEN model can be determined, where:

$$\begin{pmatrix} Psychoticism \\ Extraversion \\ Neuroticism \end{pmatrix} = A_{pen} \begin{pmatrix} \frac{1}{13.5}(Neighbor\ Dist - 15) \\ \frac{1}{49.5}(Max.\ Neighbors - 10) \\ \frac{1}{14.5}(Planning\ Horiz. - 30) \\ \frac{1}{0.85}(Radius - 0.8) \\ \frac{1}{0.5}(Pref.\ Speed - 1.4) \end{pmatrix}$$

Based on a linear regression of the study data, A_{pen} was found to be

$$A_{pen} = \begin{pmatrix} 0.00 & 0.08 & 0.08 & -0.32 & 0.63 \\ -0.02 & 0.13 & 0.08 & -0.22 & 1.06 \\ 0.03 & -0.01 & -0.03 & 0.39 & -0.37 \end{pmatrix}$$

Again, this mapping can be used to predict the expected PEN values from any given simulation parameters.

6.5.3 Factor Analysis

Analyzing the various features of the A_{pen} matrix, reveals a strong correlation between the different PEN factors. Psychoticism and Extraversion show a strong positive correlation with each other and both are negatively correlated with Neuroticism. Likewise in the A_{adj} matrix shows a correlation between several factors such as Aggressive and Assertive, which have a Pearson r-squared value of 0.45 in the data collected from the user study. These correlations suggest that just a few underlying latent factors might be able to explain the perceived behaviors in the simulations.

Similar to the original OCEAN studies (Costa and McCrae, 1992), these few primary factors can be found using factor analysis methods. A Principal Component Analysis (PCA) on A_{adj}, revealed that two factors that can explain over 95% of the linear relationship between the simulation parameters and behaviors. This result suggests that low-dimension models such as the PEN model offers sufficiently rich dimensions to characterize personality traits in crowd navigation. The two Principal Component found through factor analysis on the user study data are:

$$\begin{pmatrix} PC1 \\ PC2 \end{pmatrix} = \begin{pmatrix} 0 & -0.04 & 0.04 & 0.75 & 0.66 \\ 0.14 & 0.5 & 0.8 & 0.15 & -0.19 \end{pmatrix}$$

The factor PC1 primarily has the effect of increasing an agent's radius and speed. The factor PC2 primarily makes agents plan further ahead and consider more agents for local avoidance. For these reasons, I suggestively refer to PC1 as "Extraversion" and PC2 as "Carefulness". Figure 6.2 shows which personality adjective is most affected, as PC1 and PC2 are jointly varied. The chart indicates that as "Extraverted" agents become more "Careful", they move from appearing Aggressive to Assertive to Active. Likewise, agents who are not "Extraverted" appear Shy, as long as they are "Careful" enough to avoid looking impulsive. Furthermore, agents who are too "Careful" appear to be Tense. I believe these two principal components cover the personality space in an interesting and intuitive fashion.

6.6 Simulation Results and Validation Study

Using the above mappings of A_{pen} and A_{adj}, its possible to perform crowd simulations in which certain agents appear to exhibit high levels of the different PEN traits, or appear to display high levels of one or more of the studied personality adjectives. This sections show the resulting trajectory of agents displaying various personalities in several different scenarios. I also present the results of a second user study, designed to validate the ability of the approach to generate agents with a given personality using the derived mappings from the user study (see Sec. 6.5).

For the purpose of this validation study, the agents' preferred velocities were clamped to the range [1.35,1.55] m/s. This range was chosen for two reasons. First, this is the range

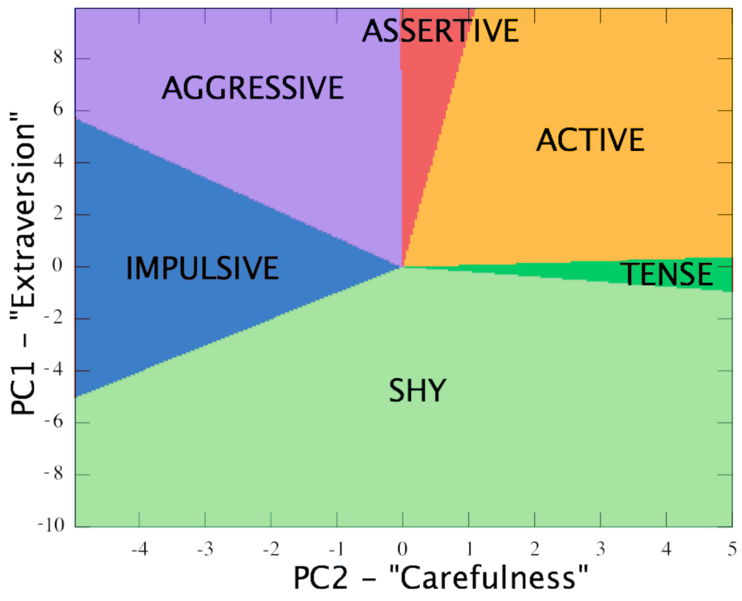


Figure 6.2: This chart shows which behavior adjective has the largest change as the two principal components are varied.

of normal walking velocities observed in crowds (Still, 2000), which focuses the study on normal behaviors rather than extreme ones. Second, inspecting the columns of A_{adj} and A_{pen} suggests that perceived personalities are most dependent on preferred velocities, by limiting this range the effect of other simulation parameters is better highlighted. Given these constraints on preferred velocity, the trained mappings was used to find simulation parameters for various adjectives and traits covered in the user study. Again, to limit unnatural or extreme behaviors, parameters were chosen that change behavior by only one "unit" (on the 1-9 scale described in Sec 6.4.1). The parameters used are summarized in Table 6.3.

6.6.1 Simulation Results

I now show the results of agents with various personalities in different scenarios. Figure 6.3 shows paths taken by the highlighted agents in the Pass-Through scenario. The Aggressive agents can be seen to be taking fairly direct paths. The Impulsive agents still

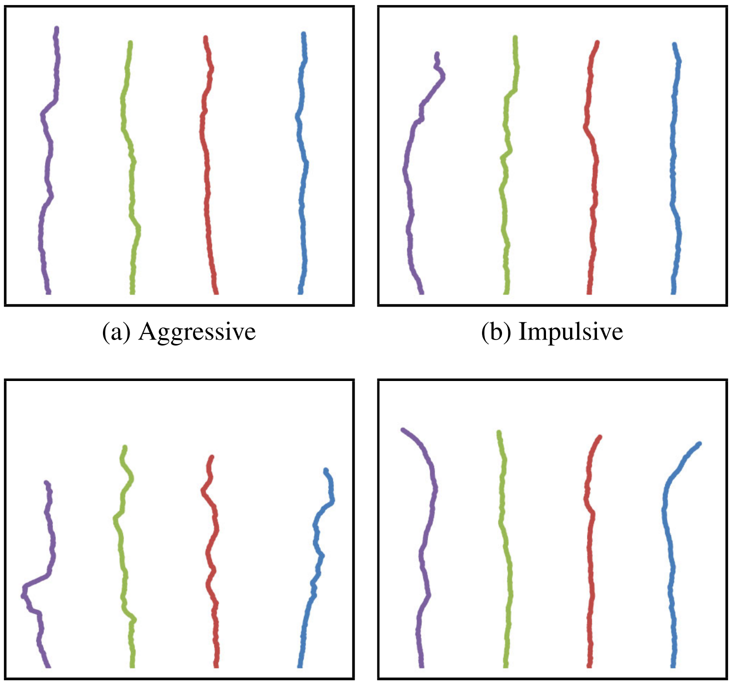
Trait	Neigh. Dist	Num. Neigh.	Plan. Horiz.	Radius	Speed
Psych.	15	40	38	0.4	1.55
Extrav.	15	23	32	0.4	1.55
Neuro.	15	9	29	1.6	1.25
Aggres.	15	20	31	0.6	1.55
Assert.	15	23	32	0.5	1.55
Shy	15	7	30	1.1	1.25
Active	13	17	40	0.4	1.55
Tense	29	63	12	1.6	1.55
Impul.	30	2	90	0.4	1.55

Table 6.3: Simulation parameters for various personality traits.

move quickly, but tend to take less direct routes. Shy agents avoid others more often, so progress more slowly. Tense agents take the least jittery paths, but are deflected by the crowds more than aggressive agents.

Agent behaviors can also be chosen based on the Eysnek 3-factor personality model by using A_{pen}. Figure 6.4 shows the Hallway scenario with agents that have a high level of "Psychoticism" (P-factor), agents with a high level of "Extraversion" (E-factor), and agents with a high level of "Neuroticism" (N-factor). The agents with a high level of Eysnek's P-factor take fast and direct paths coming close to other agents. The agents with a high level of Eynsek's E-factor also move quickly, but take more daring paths, sometimes attempting to weave through the other agents in the crowd. The agents with a high level of Eysnek's N-factor take slower less direct paths and move farther away to avoid the static gray agents.

In the Narrowing Passage scenario, agents also show a variety of behaviors for different personalities. Figure 6.5 shows the same time-step from two different simulations. In the left simulation, the light red agents are assigned a personality of Aggressive. In the right simulation, the light red agents are Shy. At this point, a few seconds into the simulation, many more Aggressive agents have moved through the exit than the Shy agents. Furthermore, several of the Shy agents can be seen to be holding back away from the exit causing less congestion.

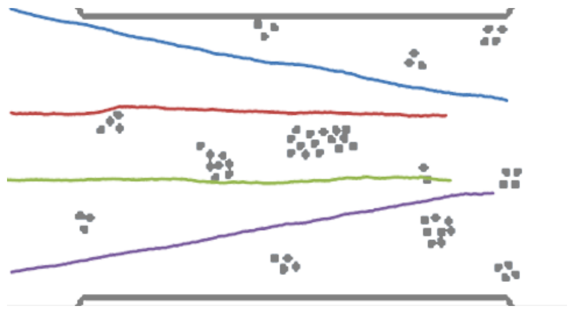

(a) Aggressive (b) Impulsive

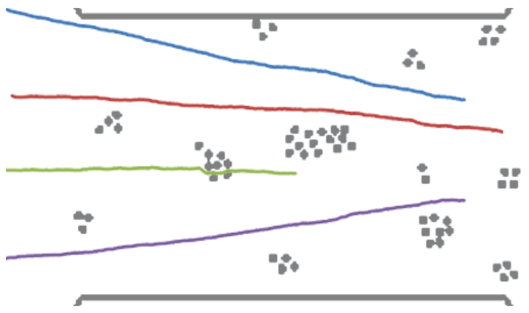
(c) Shy (d) Tense

Figure 6.3: **Pass-through Scenario.** Paths of agents trying to push through a crowd in various simulations. The agent's parameters correspond to various personalities. All paths are displayed for an equal length of time. (a) Aggressive agents make the most progress with the straightest paths. (b) Impulsive agents move quickly but take less direct routes. (c) Shy agents are diverted more easily in attempts to avoid others (d) Tense agents take less jittery paths, but are easily deflected by the motion of others.

A comparison of the rate at which the agents of various personalities passed through the exit is shown in Fig. 6.6. Shy and Tense agents were the slowest to pass through the exit, as they moved less quickly and packed in less tightly than the Aggressive and Assertive agents who made it out fastest.

The evacuation results change when too many of the agents are acting aggressively. Figure 6.7 shows how the average speed of the Aggressive agents in the scenario varies as the percent of Aggressive agents increases. As the graph shows, the Aggressive agents exhibit the well known "faster-is-slower" behavior associated with panic in crowds (Helbing et al., 2000). Once a critical threshold of too many aggressive agents is reached, the aggressive agents actually exit the room slower than a non-aggressive agents would.


(a) High Psychoticism (P)


(b) High Extraversion (E)

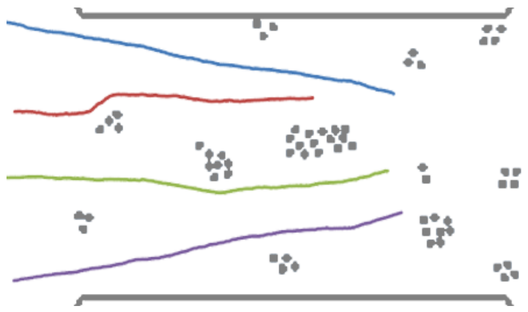
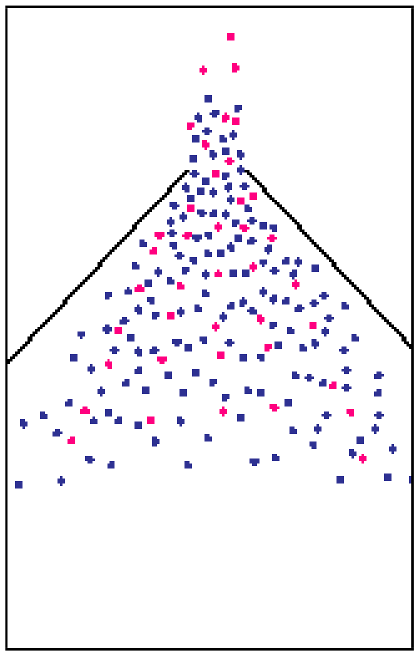

(c) High Neuroticism (N)

Figure 6.4: **Hallway Scenario.** A comparison between (a) agents with high levels of "Psychoticism", (b) "Extraversion" and (c) "Neuroticism". Each of the four agents' paths is colored uniquely. The high P-factor agents repeatedly cut close to others taking the most direct paths. The high E-factor agents take faster and occasionally "daring" paths, the high N-factor agents take more indirect paths and keep their distance from others.

6.6.2 Heterogeneous Crowds

Using the mappings derived from experimental study, it is easy to generate different simulations that map to different high-level personality specifications. This can be used to create interesting variations in complex, heterogeneous crowd simulations. For example, in the following an evacuation scenario 215 agents simultaneously compete for space as they leave a room through the same exit. Using the personality-to-parameters mapping, the work presented hear can easily create a wide variety of specific behaviors during the evacuation, as shown in Fig. 6.8. The agents' shirts are color coded by their personalities, for example agents with red shirts are aggressive and those with brown shirts are shy. The agents behave as expected with aggressive ones exiting first, active agents darting around slow agents in front of them and shy agents hanging back.


(a) Aggressive

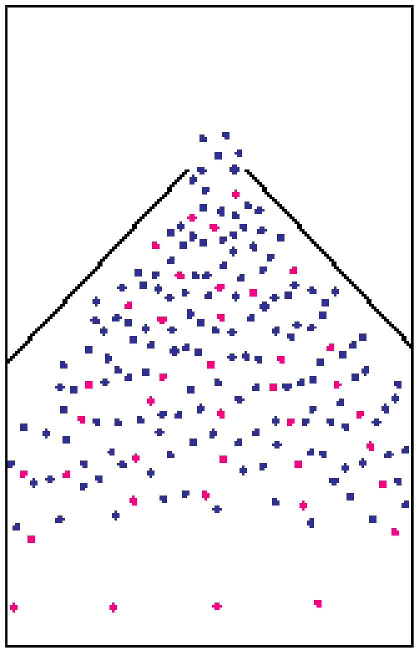

(b) Shy

Figure 6.5: **Narrowing Passage Scenario.** A comparison between dark-blue default agents and light-red Aggressive agents (a) and light-red Shy agents (b). The Aggressive agents exited more quickly, while several Shy agents stay back from the exit causing less congestion.

6.6.3 Timing Results

Because the behavior mapping can be computed as a pre-processing step, the method presented here adds no overhead to the overall simulation runtime. Table 6.4 shows the execution time for simulating agents in several different scenarios, the timings were computed on a 3.2 GHz Intel i7 processor. In all cases, the simulation ran at interactive rates.

Scenario	Agents	Obstacles	Time (msec)
Hallway	70	2	0.4
Narrowing Passage	200	2	1.9
Pass Through	404	0	1.4
Evacuation	215	125	4.5

Table 6.4: Performance timings per frame.

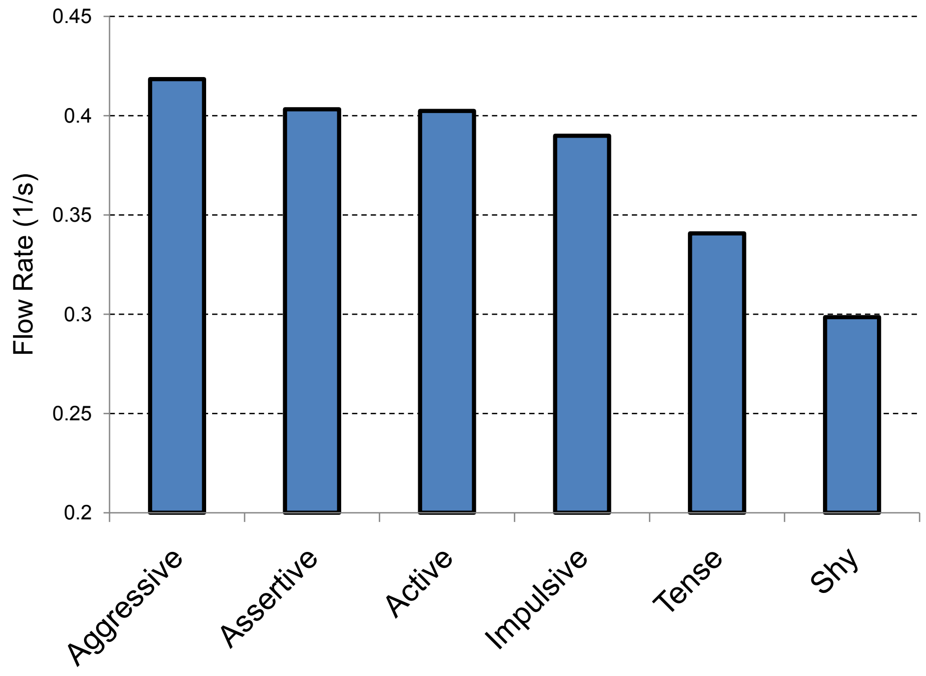


Figure 6.6: **Exit Rate.** Rate at which agents of various personalities exit in the Narrowing Passage scenario.

6.6.4 Validation Study

To validate these personality mappings, a follow-up user study was performed where users were asked questions targeted at evaluating how well the model performed at producing simulations with the expected behavior. The study was taken by 19 participants (39% female, average age 37±16), 72% of whom had participated in the original study. This follow-up study consisted of three sections. This validation study used entirely new videos to reduce participant bias. In the first two sections, a personality trait was selected at random, and a pair of videos were generated: one showing a simulation of that trait using the values in Table 6.3, and one chosen to contrast the selected trait. Participants were asked to choose which of the two videos better showed the personality trait in question. The first section of the study evaluated the six personality adjectives (aggressive, assertive, shy, active, impulse, and tense). The second section evaluated the PEN traits after a brief explanation of each

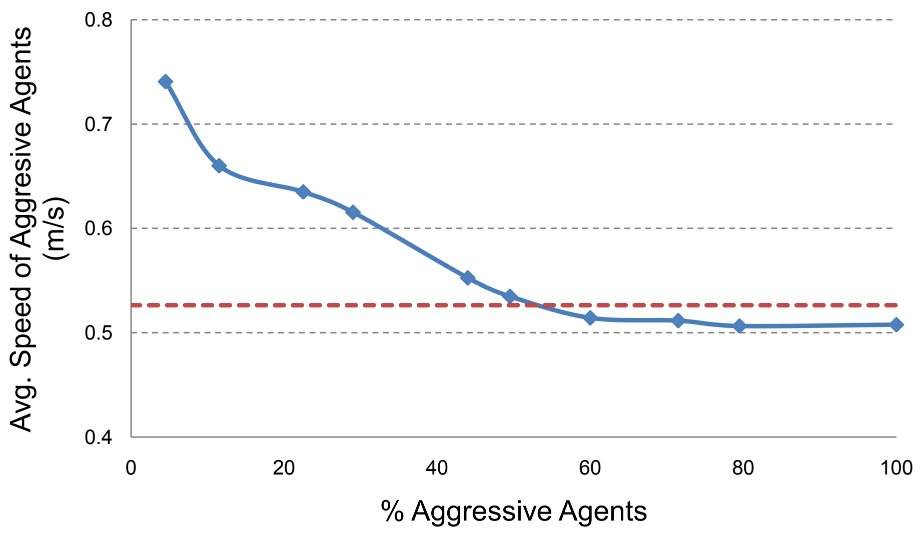


Figure 6.7: **Faster-is-slower behavior.** This graph shows the speed of Aggressive agents exiting in the Narrowing Passage scenario (solid blue line). As a larger percentage of agents become aggressive, their ability to exit quickly is reduced to the point where they exit more slowly than less Aggressive agents with the same preferred speed (dashed red line). This result is consistent with the well-known "faster-is-slower behavior" (Helbing et al., 2000).

of their meanings to the participants. These sections were intended to measure how well a given personality attribute could be reproduced.

In a third section, participants were shown a video where agents were chosen to display a high level of one adjective while maintaing no increase in another one (e.g. Active, but not Aggressive). Participants were then asked to choose which of the two adjectives better described the video. This task was intentionally chosen to be challenging, as it explores to what degree the mapping can model each adjective independent of the others. Some combinations (such as "Impulsive, but not Active") were not used in the study as the mapping suggested the adjectives were too strongly correlated to be independently varied within the domain of allowed velocities.

The results of the three sections are summarized in Table 7.2. For all three sections the model predicted the perceived personalities correctly at a statistically significant rate ($p<.05$). For all results, the statistical p-values were calculated using a a one-tailed test with an exact binomial calculation of probability. The low p-values provide strong evidence these results are due capturing a mapping of traits to parameters and not just statistical noise.

Sec.	Description	Accuracy	p-value
1	Chose video from adjective	87%	1e-7
2	Chose video from PEN trait	96%	1e-11
3	Chose adjective from video	72%	1e-7

Table 6.5: Performance on validation study

The results of the study can be further broken down by analyzing the results for each adjective separately. In the first section, users perform with a 100% success rate at identifying which videos corresponded to Assertive, Shy, and Active. Aggressive and Impulsive were also identified at a high, statistically significant, rate of 80% and 85% respectively.

When combined with the more difficult task of separating two simultaneous personalities constraints (such as Shy, but not Impulsive) the overall success rate drops. However, participants were still able to correctly identify most adjectives at a statistically significant rate. Figure 6.9 shows a graph of the breakdown of the overall success rate for all questions involving each of the six adjectives. An asterisk next to the adjective indicates a statistically significant result ($p<.05$).

This data suggests the traits of “Aggressive” and “Impulsive” were hard to vary independently without affecting the perceived levels of other traits, such as Assertiveness and Shyness. This result is consistent with the high correlations seen between these adjectives in the initial user study.

The method presented heard also performed well at generating the specific PEN personality traits. Figure 6.10 shows the success rate for questions involving the PEN values. The high success rate indicates participants were easily able to apply the high level concepts behind the PEN model to evaluating various behaviors in the simulations.

6.7 Summary and Conclusions

This chapter presented a perceptually driven approach to model the personality of different agents in a crowd simulations. This approach can successfully generate crowd simulations in which agents appear to depict specific, user-specified personalities, such as assertive, shy, and impulsive. Furthermore, I have shown that this approach can successfully generate simulations where agents appear to have various levels of the established PEN personality traits. Finally, I proposed two novel factors (PC1 and PC2) which are highly orthogonal, and are able to capture more than 95% of the linear correlation captured in the experimental data. To the best of my knowledge, this is the first factor-model specifically targeted at analyzing various perceived personalities in crowd simulations.

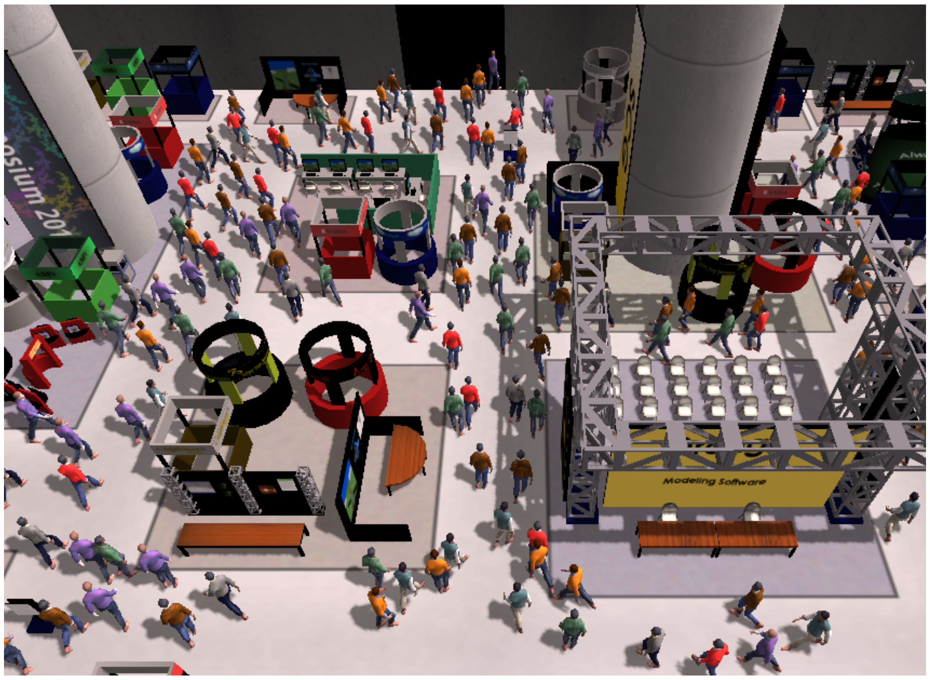
6.7.1 Limitations

The approach presented here has some limitations. The current implementation only explores variation allowed by the RVO2 library. I would like to use this approach with other collision avoidance and simulation methods to see if more drastic variation in behavior is possible. Moreover, this work focused on local behaviors and interactions between agents. However, the completed decision-making process includes global navigation and path-planning which are not modeled adequately by the simulation parameters used in this work. Given the large difference in approach between local and global planning, it is possible other personality models such as the Myers-Briggs Type Indicator (Myers et al., 1999) might be more appropriate to capture such behaviors.

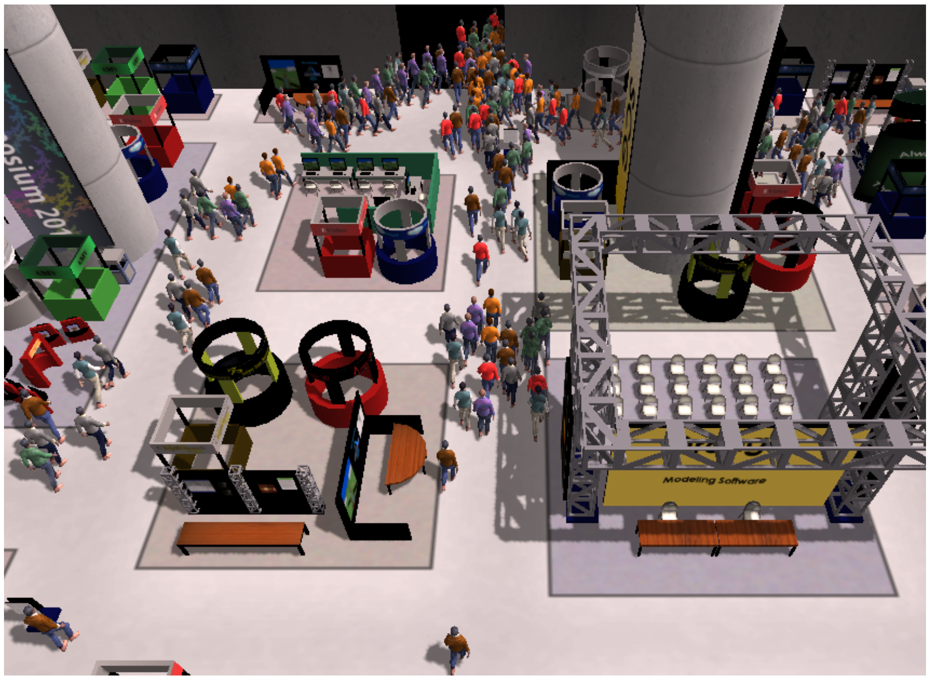
Additionally, a generic mapping is computed between simulation parameters and personality traits which is intended to hold across a wide variety of scenarios. By focusing on more specific scenarios, it may be possible to find more precise mappings for those particular scenarios.

6.7.2 Future Work

This overall approach could be applied to other crowd simulation and collision avoidance techniques, including cellular automata and social-force models. The same data-driven techniques could be adopted to build mappings from simulation parameters to other personality trait theories, such as the OCEAN model. I would also like to investigate the extent that my proposed two-factor model is appropriate for human behaviors in real-world crowds (perhaps based on video footage). Additionally, this work have focused only on computing the trajectory of the agents. Other aspects of virtual agents such as posture, facial expression, and walking style can provide clues to an agent's personality and should be taken into account when modeling various personalities.



(a) Initial Conditions



(b) Mid Simulation

Figure 6.8: **Evacuation Scenario.** 200 agents evacuating a building. Shy agents (brown shirts) hold back while Aggressive agents (red shirts) dart forward. The other personalities also display a variety of behaviors such as quick maneuvers, overtaking and pushing through.

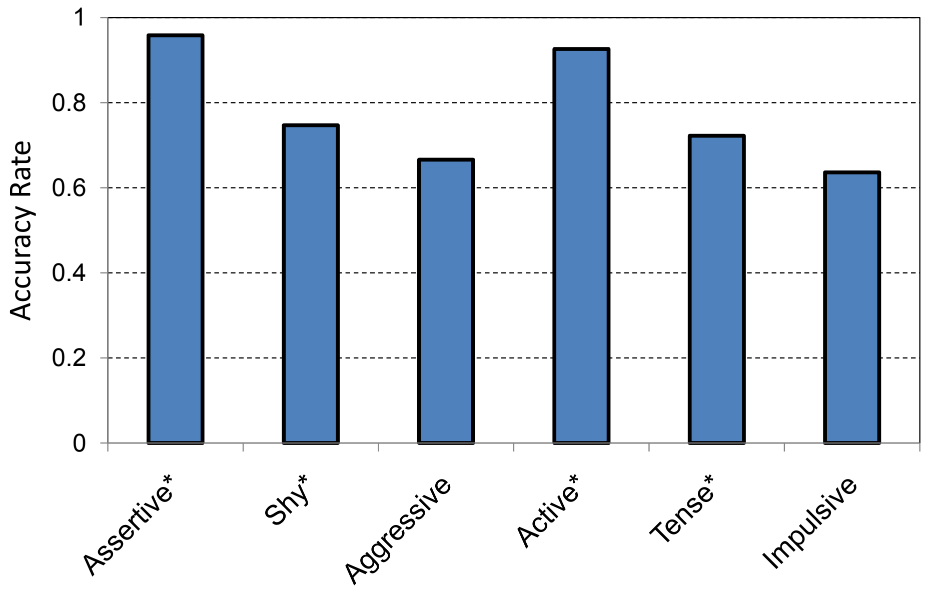


Figure 6.9: **Adjective Success Rate.** Rate at which user responses matched the indented adjective for all questions involving the six personality adjectives studied. *indicates statistically significant (p<.05).

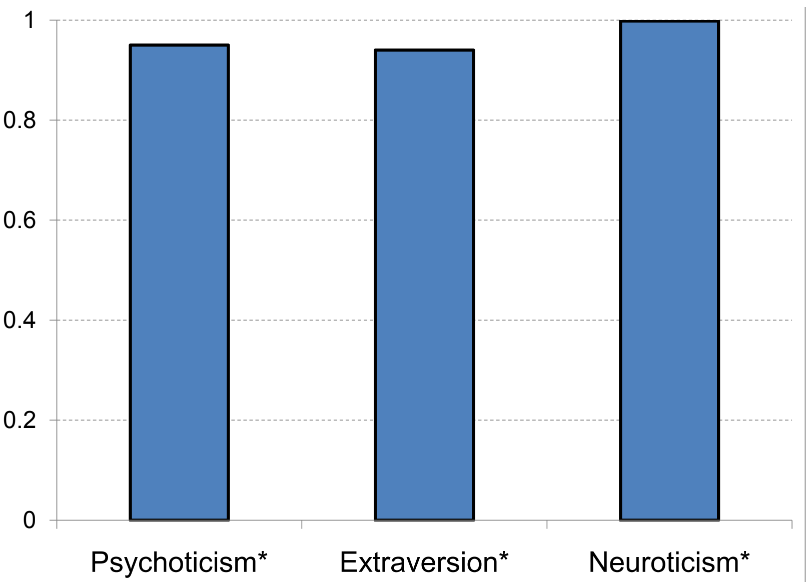


Figure 6.10: **PEN Success Rate.** Rate at which user responses matched the intended personality trait for questions involving the PEN traits. *indicates statistical significant (p<.05).

CHAPTER 7

An Entropy Metric for Evaluating Crowd Simulations

7.1 Introduction

At a broad level, there are two types of tasks where crowd simulations are used. One type is the visual simulation commonly used in games, VR, and animation. Here, what is important is the ability to create *plausible* motion. That is, motion which is realistic enough that is doesn't look wrong to a carefully engaged end-user. A second type of application, often referred to as Pedestrian Dynamics, focuses an generating crowds simulations for the purpose of predicting flows, densities, and general movement patterns. Understanding Pedestrian Dynamics is important especially in areas like building design, event planning, and architectural analysis where the goal is often to predict human flows in places which do not yet exist as an aide in planning. Simulations for these purposes require a much higher bar in terms of validation. The important question becomes not how plausible the simulated motion is, but how *accurate* it is. If crowd simulations are to be used to inform decision making it is important to know how closely the simulation captures real human behavior.

While there are multiple ways to qualify the accuracy of a simulation, this chapter focuses on the quantitative comparison of crowd simulations to trajectory data captured on real human motion. The intuition is that the closer a given simulation method can come to reproducing known examples of real human trajectories the more accurately it is capturing real human motion. The end goal is to develop an objective and quantitative formal approach that is general in terms of evaluating the accuracy of any crowd simulation technique with respect to any real-world crowd data.

As a result of recent advances in sensing technologies, including high-resolution cameras, Light Detection And Ranging (LiDAR) and Global Position Systems (GPS), combined with computer vision and sensor processing algorithms there has been a dramatic rise in the amount of data available for validation. While this data is a valuable resource, many issues arise in terms of using them for evaluation. One issue is that the trajectories may not be accurate, due to occlusion, sensor noise, and other limitations of crowd tracking algorithms.

The most serious hurdle to developing reliable techniques for analyzing crowd simulations is to the non-linear, chaotic, non-deterministic nature of human motion and group dynamics. For example, a small perturbation in the initial conditions of a simulation can potentially lead to large differences in the computed trajectories. As a result, the sensitivity of a simulation algorithm can greatly magnify the unwanted effects of sensor noise or the tracking algorithm. Beyond the uncertainty that arises from the sensor noise and simulation dynamics, paths traveled by humans have an inherent uncertainty that comes from individual differences (e.g. personality), emotional state (e.g. happy, stressed), and other subtle factors. Thus, when two humans are seemingly presented with the same choice in paths, they may choose different ways of moving around their neighbors. The ideal technique for evaluating the accuracy of crowd simulation must take into account these issues.

7.1.1 Main Result

This chapter describes an “Entropy Metric” to evaluate the accuracy of crowd simulations with respect to real-world crowd data. The metric is quantitative, general, and applicable to a variety of simulation techniques. The approach takes into account noise in the measured data, as well as non-determinism and unmodeled effects in the crowd simulation method. Such errors are modeled using a probability distribution and the most likely distribution is estimated from the observed trajectory data using Bayesian smoothing.

The method presented here simultaneously computes and optimizes the error distribution, using the expectation-maximization (EM) algorithm formulated as a combination of Ensemble Kalman smoothing and maximum likelihood estimation. Moreover, I show that the entropy metric is rankable, predictable, discriminative, and robust with respect to sensor noise. I demonstrate its application to evaluate the accuracy of three different crowd simulation algorithms against different real-world crowd trajectories corresponding to both indoor and outdoor scenes.

7.1.2 Organization

This chapter is organized as follows: Section 7.2 gives a broad characterization of crowd simulation algorithms and introduces the notation used in this chapter. Section 7.3 describes the theoretical basis of the entropy metric and present an efficient algorithm to compute the metric. The application of the metric to three crowd simulation algorithms is demonstrated in Section 7.4. Section 7.5 highlights many properties of the metric and compare it with prior crowd evaluation techniques.

7.2 Background and Notation

This section gives a brief background on crowd simulation algorithms and capturing crowd data and introduce the notation used in the rest of the chapter.

7.2.1 Definitions

The following notational conventions throughout this chapter: Variables a printed in italics denote scalars or functions, variables $\mathbf{a}$ printed in boldface denote vectors and variables $\mathbb{A}$ printed in blackboard bold denote vector spaces. Variables A printed in capitals denote (covariance) matrices, and variables $\mathcal{A}$ printed in calligraphic typefaces denote probability distributions.

Crowd State In the context of this chapter, the term crowd is used to refer to a collection of entities (e.g. people) whose behaviors and dynamics evolve over time. The notion of a crowd state is defined as follows: for a given (discrete) point in time k, the state $\mathbf{x}_k$ of a crowd contains all non-constant information of a crowd that is relevant to its evolution over time. Moreover, the space of all crowd states is denoted as $\mathbb{X}$. For instance, for a crowd consisting of n agents, the state $\mathbf{x}_k \in \mathbb{X} = \mathbb{R}^{5n}$ may correspond to a vector containing the two dimensional position, two diminutional velocity, and one demential orientation of all n agents moving on a 2D plane. Other time-varying aspects, such as mental state of the agents or dynamic behavior parameters may also be part of the state. I make no specific assumptions about the representation of a crowd state. In addition to the crowd state, the computational algorithms also use *constant* information, such as obstacles (with pre-defined motion paths) in the world, or fixed attributes of the agents.

Crowd Evolution Crowds are not static but evolve over time. That is, if the state of the crowd at time k is $\mathbf{x}_k \in \mathbb{X}$, the crowd evolves into a state $\mathbf{x}_{k+1} \in \mathbb{X}$ one unit of time later. The crowd dynamics, the mathematical formulation driving this evolution, is not well understood and can not be analytically modeled or derived from first principles. In order to characterize this unknown crowd dynamics, an abstract function $f : \mathbb{X} \to \mathbb{X}$ is defined, such that:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k). \tag{7.1}$$

It should be stressed that f is abstract and unknown, and is only used as a formulation to describe crowd evolution.

7.2.2 Crowd Simulation and Trajectory Generation

Crowd simulation refers to the process of simulating the movement of a large number of human agents, taking into account the interactions between them as well the environment. At a broad level, these simulations take into account group dynamics and human behaviors. While the field of crowd simulation covers many facets of generating human motions and behaviors (such as full-body biomechanics, facial expressions and gestures, and motion dynamics based on the laws of physics), modeling and analyzing all aspects of a crowd is highly challenging and can lead to combinatorial explosion of potential variations. Therefore, the work here only considers the output of these crowd simulation algorithms that corresponds to existing recorded data which is commonly positions or trajectories.

Simulating Crowds: There has been extensive work on crowd simulation for more than three decades, leading to a wide variety of strategies for simulating crowds. These include force-based methods (Helbing and Molnar, 1995; Pelechano et al., 2007; Karamouzas et al., 2009), boids and steering models (Reynolds, 1987, 1999), techniques based on velocity obstacles and geometric optimization (Guy et al., 2010b; van den Berg et al., 2011), field based methods (Patil et al., 2011; Pettré et al., 2009; Ondrej et al., 2010; Sung et al., 2004), continuum models (Treuille et al., 2006; Narain et al., 2009), cognitive models and decision networks (Funge et al., 1999; Yu and Terzopoulos, 2007), and more. These methods model different aspects of crowds including collision avoidance between agents, emergent behaviors, path navigation, high-level behaviors. However, all of them compute continuous trajectories for each agent as a representation of crowd movement as a function of characteristics of agents in the crowds and their environment.

In this sense, all such crowd simulation algorithms can be abstracted as a function $\hat{f} : \mathbb{X} \to \mathbb{X}$ which attempts to approximates the function f:

$$\hat{f}(\mathbf{x}_k) \approx f(\mathbf{x}_k). \tag{7.2}$$

That is, a crowd simulation $\hat{f}$ takes in a state $\mathbf{x}_k$ of the crowd at time k and produces an estimate of the state $\mathbf{x}_{k+1}$ of the crowd at one unit of time later. This formulation assumes that the crowd simulation algorithm works in a continuous space and this approach may not be applicable to discrete approaches (e.g. techniques based on cellular automata). Section 7.4.1, describes the detailed representation of function $\hat{f}$ for some of the commonly used crowd simulation algorithms.

7.2.3 Crowd Data Sources

Empirical datasets of human crowd motion from videos, LiDAR, and GPS sensors are becoming increasingly available, aided by research in computer vision, robotics and pedestrian dynamics on extracting crowd trajectories from sensors and cameras (Seyfried et al., 2010; Lee et al., 2007; Rodriguez et al., 2009; Kratz and Nishino, 2011; Pettré et al., 2009). In fact, a recent trend has been to combine tracking algorithms with crowd dynamics models to extract more accurate trajectories or detect crowd behaviors (Pellegrini et al., 2009; Mehran et al., 2009).

Most of these tracking algorithms represent the position data or the trajectory as time-stamped vectors $\mathbf{z}_k, \mathbf{z}_{k+1}, \ldots$ that provide a partial (and potentially noisy) projection of the true crowd state $\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots$ at the corresponding moment in time. It is assumed that the relation between the crowd state $\mathbf{x}_k$ and the data $\mathbf{z}_k$ available of the crowd at time k is given by a known function h:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{q}_k, \qquad \mathbf{q}_k \sim \mathcal{Q}, \tag{7.3}$$

where $\mathbf{q}_k$ represents the noise or uncertainty in the real world data, drawn from a constant distribution $\mathcal{Q}$. The rest of this chapter assumes that $\mathcal{Q}$ (the sensor uncertainty) is known.

7.3 Entropy Metric

This section describes the proposed entropy metric and presents an efficient algorithm to compute it. The entropy metric begins with the assumption that any crowd simulation technique could generate real-world crowd trajectories, if some, non-deterministic, correction is added to the simulation at each time step (Figure 7.1 illustrates this notion). These correction vectors are denoted as $\mathbf{m}_k$, which together form a probability distribution $\mathcal{M}$:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) = \hat{f}(\mathbf{x}_k) + \mathbf{m}_k, \qquad\qquad \mathbf{m}_k \sim \mathcal{M}. \tag{7.4}$$

The distribution $\mathcal{M}$ depends on the underlying crowd simulation algorithm $\hat{f}$ and encompasses all error and unmodeled effects in $\hat{f}$, as well as potential non-determinism in the function f. The larger this distribution $\mathcal{M}$ is, the less accurately the simulation is capturing the real-world data.

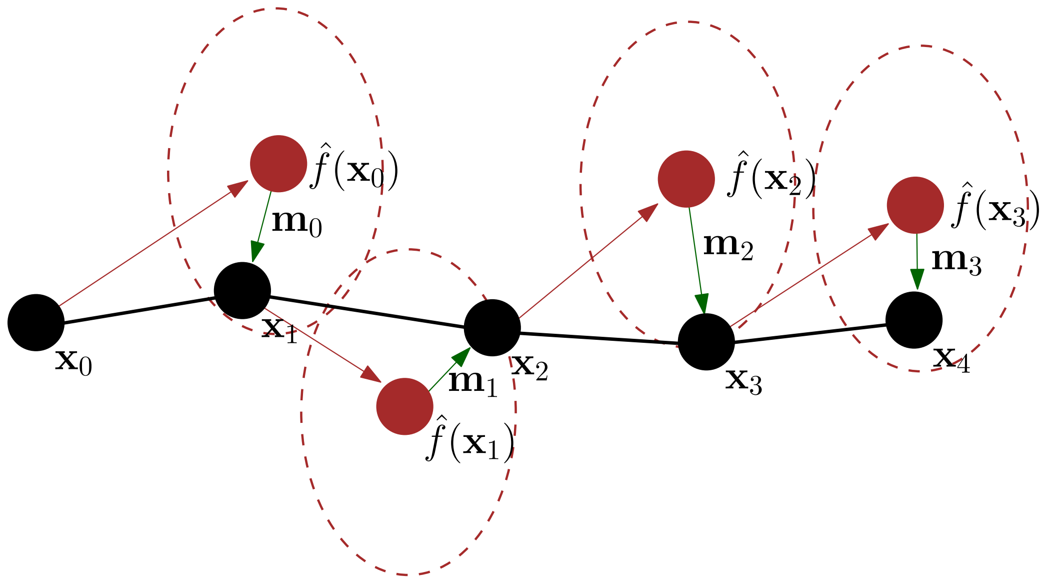


Figure 7.1: Adding a corrective vector $\mathbf{m}_k$ (green arrow) to each simulation prediction (red dot) can correctly produce true states of the crowd $\mathbf{x}_k$ (black dots). The true evolution of the state is shown in black as agents evolve from state $\mathbf{x}_0$ to $\mathbf{x}_4$. At each timestep, the crowd simulation $\hat{f}$ is used to predict the next state (red dots). In order to reproduce the true state, it is necessary that some correction vector, $\mathbf{m}_k$, is applied at each timestep (green arrow). The distribution of these $\mathbf{m}_k$'s is denoted as $\mathcal{M}$ (dashed ellipse) and contains the true state.

In order to quantify the "size" of this distribution, the notion of *entropy* from information theory is used as a measure of unpredictability. The *entropy* of the distribution $\mathcal{M}$ is the amount of bits of information that is missing from the crowd simulation algorithm $\hat{f}$ to completely model the function f. As a result, given two crowd simulation or trajectory computation algorithms, $\hat{f}_1$ and $\hat{f}_2$, the algorithm for which the entropy of $\mathcal{M}$ is lower is regarded as a "better" crowd simulator for that scenario. Therefore, the evaluation metric is defined as the entropy of the distribution $\mathcal{M}$ of the correction vectors. The higher this entropy, the larger the error distribution is, and the less accurate the simulation is.

Entropy metric: *The entropy of the distribution $\mathcal{M}$, of error between the evolution of a crowd predicted by a simulator $\hat{f}$ and by the function f.*

Given the entropy metric, the underlying problem of evaluating the accuracy of a crowd simulation algorithm can be defined as follows: given a crowd trajectory computation algorithm $\hat{f}$, real-world trajectory data $\mathbf{z}_0, \ldots, \mathbf{z}_t$ of a crowd over a period of time $0, \ldots, t$, a function h that defines the data measurement process, and distribution $\mathcal{Q}$ that defines the noisiness of the data, estimate the distribution $\mathcal{M}$ of the deviation between the simulator and reality, and compute its entropy.

7.3.1 Computing the Entropy Metric

The main issue in computation of the entropy metric is that the distribution $\mathcal{M}$ is unknown for a given crowd simulator $\hat{f}$. Moreover, the function f is unknown. Therefore, the real world data, $\mathbf{z}_0, \ldots, \mathbf{z}_t$, must be used to estimate $\mathcal{M}$ and compute its entropy.

Likewise, the true crowd states $\mathbf{x}_0, \ldots, \mathbf{x}_t$ that correspond to the given data are unknown. If the true crowd states were known, or even only their probability distributions $\mathcal{X}_0, \ldots, \mathcal{X}_t$ (such that $\mathbf{x}_k \sim \mathcal{X}_k$), the distribution $\mathcal{M}$ could be easily estimated by measuring for each crowd state $\mathbf{x}_k$ the expected error between the next state $\hat{f}(\mathbf{x}_k)$ predicted by the simulator and the true next state $\mathbf{x}_{k+1}$. Fortunately, it is possible to use *Bayesian inference* (McLachian and Krishnan, 1996) to infer the probability distributions $\mathcal{X}_0, \ldots, \mathcal{X}_t$ of the true crowd states from the given crowd data, but only if $\mathcal{M}$ is known. This results in a circular dependency, $\mathcal{M}$ can be computed if $\mathcal{X}_0, \ldots, \mathcal{X}_t$ is known, and $\mathcal{X}_0, \ldots, \mathcal{X}_t$ can be computed if $\mathcal{M}$ is known.

This problem can be solved by iteratively computing the distributions $\mathcal{X}_k$ given the current estimate of distribution $\mathcal{M}$ (using Bayesian smoothing), and computing $\mathcal{M}$ given the current estimate of the distributions $\mathcal{X}_k$ (using maximum-likelihood estimation), bootstrapped by a rough initial estimate of $\mathcal{M}$ and $\mathcal{X}_0$. This process is known as the *EM-algorithm* (McLachian and Krishnan, 1996), and is guaranteed to converge in a coordinate-ascent manner to a locally optimal estimate of the distributions $\mathcal{X}_k$ and $\mathcal{M}$ (in terms of their *likelihood*) given the observed data $\mathbf{z}_0, \ldots, \mathbf{z}_t$. This process is summarized in Figure 7.2 and discussed in detail bellow. The entropy of the estimated distribution $\mathcal{M}$ is then computed and used as the evaluation metric for the crowd simulator $\hat{f}$.

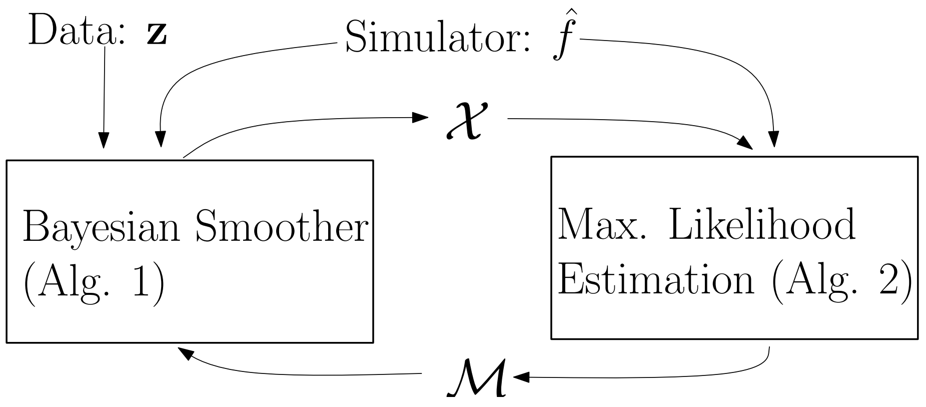


Figure 7.2: The error distribution $\mathcal{M}$ for crowd simulation algorithm is estimated via an iterative process, based on the EM-algorithm. A Bayesian Smoother is used to estimate the true crowd states $\mathcal{X}$ given data $\mathbf{z}$, a simulator $\hat{f}$, and an error distribution $\mathcal{M}$. Next, the maximum likelihood estimate of $\mathcal{M}$ given the simulator is computed and used to estimate the state distributions $\mathcal{X}$. This process is repeated until convergence.

7.3.2 Simplifying Assumptions

There are many difficulties in computing the Entropy metric exactly, arising from both theoretical and practical issues related to the underlying complexity and non-linear aspects of crowd dynamics, combined with the general non-parametric nature of the distribution $\mathcal{M}$. Therefore, appropriate approximations need to be made in order to compute an approximated value of the Entropy metric.

The most important assumption is that all relevant distributions for computing the metric can be modeled as Gaussians. This choice is motivated in part by the central limit theorem (CLT) and in part by the maximum-entropy principle (MEP). In order to make the computations tractable, the distributions $\mathcal{M}$ and $\mathcal{X}_k$ need to represented in a parametric form, and it is natural to choose the first two moments (mean and variance) of the distribution as the relevant parameters. The CLT states that uncertainty that arrises as the combination of many independent sources can be modeled as a Gaussian. The MEP states that in the absence of knowledge about the true nature of a distribution one should choose a distribution with the largest entropy so as to minimize the amount of information introduced. Given only the mean and variance this distribution is also the Gaussian distribution.

Therefore the distribution $\mathcal{X}_k$ of the state at time k and the error distribution $\mathcal{M}$ are represented as Gaussians. Further, the crowd state $\mathbf{x}_k$ is assumed to be composed of the states of each of the n individual agents within the crowd. Hence, if the state of a single agent has dimension d, then the dimension of the composite crowd state is nd. Three further assumptions regarding the error distribution $\mathcal{M}$: (1.) The crowd simulator has no systemic bias in the error of its predictions; (2.) The crowd simulator is not systemically more accurate for some agents within a crowd than for others; (3.) There is no systemic covariance between the prediction errors of different agents within the crowd. The result is that the distribution $\mathcal{M}$ has a zero mean, and that its covariance matrix is block-diagonal;

$$\mathcal{M} = \mathcal{N}(\mathbf{0}, \begin{bmatrix} M & 0 & \cdots & 0 \\ 0 & M & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & M \end{bmatrix}), \tag{7.5}$$

where M is a $d \times d$ covariance matrix which appears n times along the diagonal. In this case, M models the per-agent error variance of the crowd simulator, and distribution $\mathcal{M}$

is fully defined by variance M. This representation also allows for the use of datasets of crowds of different sizes in evaluation.

7.3.3 Computing Crowd State Distributions

In order to compute the maximum-likelihood estimate of distribution $\mathcal{M}$ given the validation data $\mathbf{z}_0, \ldots, \mathbf{z}_t$, the EM-algorithm is used as outlined above. The first step of the EM-algorithm consists of estimating the distributions $\mathcal{X}_k$ of the true crowd states $\mathbf{x}_k$ given an estimate for $\mathcal{M}$ using a Bayesian smoother. Since these distributions are assumed to be Gaussian, it is natural to use a non-linear variant of the Kalman smoother. Because the state spaces are (very) high dimension, nd, these distributions $\mathcal{X}_k$ are not represented in an explicit parametric form with each of its t variance matrices consists of n^2d^2 entries. Instead, an ensemble representation is used, where each distribution $\mathcal{X}_k$ is represented by a number of m samples (with $m << nd$), and use an Ensemble Kalman Smoother (EnKS) (Evensen, 2003) to estimate $\mathcal{X}_k$.

The EnKS algorithm tends to work particularly well for high-dimensional state spaces and non-linear dynamics (Evensen, 2003). In this case, each distribution $\mathcal{X}_k$ is represented by an ensemble of m samples: $\mathcal{X}_k = \{\hat{\mathbf{x}}_k^{(1)}, \ldots, \hat{\mathbf{x}}_k^{(m)}\}$, and I assume an initial ensemble $\mathcal{X}_0$ is given. The smoother then proceeds as shown in Algorithm 3. This computes a representation for $\mathcal{X}_0, \ldots, \mathcal{X}_t$, given a current estimate of $\mathcal{M}$ and the trajectory data $\mathbf{z}_0, \ldots, \mathbf{z}_t$.

7.3.4 Computing the Variance M

The second step of the EM-algorithm consists of computing the maximum-likelihood estimate of the distribution $\mathcal{M}$, given the current estimates of the distributions $\mathcal{X}_0, \ldots, \mathcal{X}_t$ of the crowd states as computed in the first step. Since only the estimated distribution are known, and not the states themselves, the distribution $\mathcal{M}$ of which the *expected* likelihood is maximal is computed. Since $\mathcal{M}$ is fully defined by variance M, this is equivalent to maximizing the expected likelihood of M. Further, it is mathematically convenient to

Algorithm 3: Ensemble Kalman Smoothing to estimate crowd states

```
Input: Measured crowd data $\mathbf{z}_1 ... \mathbf{z}_k$, Crowd Simulator $\hat{f}$, Estimated error variance $M$
Output: Estimated crowd state distributions $\mathcal{X}_1 ... \mathcal{X}_k$
foreach $k \in 1 \dots t$ do
    // Predict
    foreach $i \in 1 \dots m$ do
        Draw $\mathbf{m}_{k-1}^{(i)}$ from $\mathcal{M}$
        $\hat{\mathbf{x}}_k^{(i)} = \hat{f}(\hat{\mathbf{x}}_{k-1}^{(i)}) + \mathbf{m}_{k-1}^{(i)}$
        Draw $\mathbf{q}_k^{(i)}$ from $\mathcal{Q}$
        $\hat{\mathbf{z}}_k^{(i)} = h(\hat{\mathbf{x}}_k^{(i)}) + \mathbf{q}_k^{(i)}$
    $\bar{\mathbf{z}}_k = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{z}}_k^{(i)}$
    $Z_k = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)(\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)^T$
    // Correct
    foreach $j \in 1 \dots k$ do
        $\bar{\mathbf{x}}_j = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{x}}_j^{(i)}$;
        $\Sigma_j = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}_j)(\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)^T$
        foreach $i \in 1 \dots m$ do
            $\hat{\mathbf{x}}_j^{(i)} = \hat{\mathbf{x}}_j^{(i)} + \Sigma_j Z_k^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^{(i)})$
```

maximize the expected *log-likelihood* $\ell\ell(M)$ (the logarithm cancels against the exponent in the probability density function of a Gaussian distribution), which is equivalent since the logarithm is a monotonic function.

The part of the nd-dimensional crowd state $\mathbf{x}_k$ that contains the state of agent $j \in 1 \dots n$ is denoted as $\mathbf{x}_k[j]$. Now, the expected log-likelihood of variance matrix M under the assumption of a Gaussian distributions is given by:

$$\begin{aligned} \mathrm{E}(\ell\ell(M)) = -\sum_{k=0}^{t-1} \sum_{j=1}^{n} \mathrm{E}\big(& (\mathbf{x}_{k+1}[j] - \hat{f}(\mathbf{x}_k)[j])^T M^{-1} \cdot \\ & (\mathbf{x}_{k+1}[j] - \hat{f}(\mathbf{x}_k)[j])\big), \qquad \mathbf{x}_k \sim \mathcal{X}_k. \end{aligned} \tag{7.6}$$

Given the ensemble representations available of the distributions $\mathcal{X}_k$, the variance M that maximizes this expected log-likelihood is given by Algorithm 4.

Algorithm 4: Maximum Likelihood Estimation

```
  Input: Estimated crowd state distributions $\mathcal{X}_1 ... \mathcal{X}_k$, Crowd simulator $\hat{f}$
  Output: Estimated error variance $M$
1 $M = 0$;
2 foreach $k \in 0 \dots t-1$ do
3     foreach $i \in 1 \dots m$ do
4         foreach $j \in 1 \dots n$ do
5             $M += (\hat{\mathbf{x}}_{k+1}^{(i)}[j] - \hat{f}(\hat{\mathbf{x}}_k^{(i)})[j])(\hat{\mathbf{x}}_{k+1}^{(i)}[j] - \hat{f}(\hat{\mathbf{x}}_k^{(i)})[j])^T$
6 $M /= tmn$
```

The EM-algorithm is initialized with an initial guess for M and $\mathcal{X}_0$, and both steps are repeatedly performed until convergence. The resulting M is a (local-)maximum-likelihood estimate of the error variance of crowd simulator $\hat{f}$.

7.3.5 Computing the Entropy of $\mathcal{M}$

Given the per-agent variance M as computed above, it still remains to compute the entropy of the Gaussian distribution $\mathcal{M}$ of Eqn. (7.5). The entropy is given by:

$$e(\mathcal{M}) = \frac{1}{2} n \log((2\pi e)^d \det(M)), \tag{7.7}$$

where n is the number of agents in the crowd, and d is the dimension of the state of a single agent. In order to make the metric independent of the number of agents in the crowd, I normalize the above equation by dividing by n. This in fact gives the entropy of the normal distribution $\mathcal{N}(\mathbf{0}, M)$ that models the per-agent error of the crowd simulator $\hat{f}$. It can be seen that the metric for evaluating crowd simulators is proportional to the determinant of the per-agent variance M. Note that while for presentation purposes the algorithm is presented for a validation data-set of a single crowd for a single duration of time, the algorithm can be used on multiple dataset of different crowds of different sizes.

7.4 Implementation and Experimental Set-up

This sections demonstrate the application of the entropy metric on three crowd simulation algorithms. Furthermore, their accuracy in three different settings by computing the corresponding entropy metric. Three widely used crowd simulation algorithms are used: steering behaviors, force-based, and geometric collision avoidance. In each case, some of the parameters are varied to generate a family of methods and evaluated their accuracy on crowd data.

7.4.1 Simulation Models

In terms of a steering-behavior algorithm, the classical steering method proposed by Reynolds is used (Reynolds, 1999). A combinations of steering behaviors are used to achieve higher level goals, such as walking to a point, or steering away from a close neighbor. The simulation can be parameterized to generate different agent behaviors or trajectories by changing the collision radius of an agent and the preferred speed. Three different sets of parameters for the simulator are tested, which are referred to as: Steer-1, Steer-2, and Steer-3.

The second simulation algorithm is based on the Social Force Model (SFM) (Helbing and Molnar, 1995). Many variants and extensions of this model have been proposed and widely used in different applications. SFM computes the trajectory of each agent by applying a series of forces to each agent that depends on the relative positions and velocities of nearby agents. An agent A receives a repulsive force pushing away from each neighbor B, denoted as f_{AB}. Moreover, each agent is applied a force pushing perpendicularly away from the walls or obstacles, denoted as f_W. The magnitude of these forces decrease exponentially with the distance. Each agent also has a goal velocity, $\mathbf{v}_A^{pref}$, which is used to compute the desired speed and direction. The simulation function $\hat{f}$ can be expressed as:

$$\mathbf{f}_A^{new} = \frac{\mathbf{v}_A^{pref} - \mathbf{v}_A}{\tau} + \sum_{A \neq B} f_{AB} + \sum_{W} f_W \tag{7.8}$$

where τ controls the rate of acceleration. The simulation can be parameterized to generate different behaviors by changing the collision radius of an agent and the preferred speed. Three different sets of parameters for this simulator are tested, referred to as: SFM-1, SFM-2, and SFM-3.

For geometric collision avoidance, I use a velocity-based formulation called reciprocal velocity obstacle (van den Berg et al., 2011) and its implementation as the RVO2 library. This algorithm represents the crowd state $\mathbf{x}_k$ using the position and velocity for each agent. Each agent navigates by computing nearby neighbors and obstacles and computes a new velocity that would avoid a collision with those neighbors for at least τ seconds. Each such neighbor's information is represented using a velocity constraint, and the union off all these velocity constraints is denoted as RVO_A^{τ}. An agent A is also assumed to have a desired or goal velocity, $\mathbf{v}_A^{pref}$. The resulting simulation function $\hat{f}$ can be expressed as:

$$\mathbf{v}_A^{new} = \underset{\mathbf{v} \in RVO_A^{\tau}}{\arg\min} \|\mathbf{v} - \mathbf{v}_A^{pref}\|. \tag{7.9}$$

The new velocity for each agent is computed by optimization using linear programming. The simulation can be parameterized to generate different behaviors by changing parameters such as τ, the collision radius of an agent, and the preferred speed. Three different sets of parameters for the simulator were tested, referred to as: RVO-1, RVO-2, and RVO-3.

7.4.2 Real-World Crowd Data

In order to evaluate the algorithm, several sources of data are used. They correspond to different real-world scenarios (e.g. indoor or outdoor) and have varying number of agents. Furthermore, each data was captured using different sensing hardware and noise characterization (see Table 7.1).

The first real-world data corresponds to a study performed in a motion capture lab, where two humans were placed about 6m apart and asked to swap their positions (Moussaïd et al., 2011) (see Fig. 7.3a). This benchmark is labeled as *Lab*.

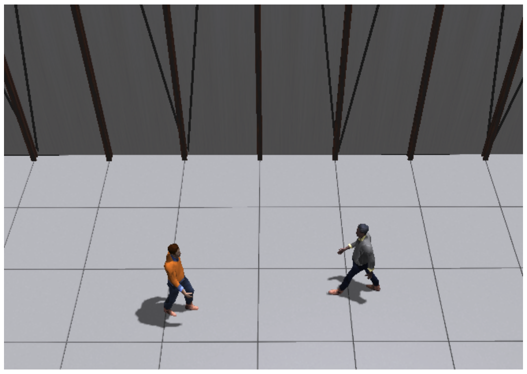
The second source of crowd data corresponds to pedestrians walking on a street and was captured using an overhead camera. The trajectories of each agent are extracted using multi-object tracking (Pellegrini et al., 2009) (see Fig. 7.3b). Two different trajectories are chosen from this crowd data, and each of them is about 10 sec and corresponds to different group motions. These medium-sized trajectory datasets are labeled as *Street-1* and *Street-2*.

The third real-world data source comes from a large indoor experiment used to analyze exiting behavior through passages of varying sizes (Seyfried et al., 2010). The experiment involved use of markers and optical tracking equipment to gather high-quality data corresponding to subjects' positions near the entrance of the passageway. While the average number of subjects tracked during a single frame was around 50, hundreds of subjects moved through the tracked area (see Fig. 7.3c). Two sources of data were used from this dataset, the first with a passage of width 1m and the second with a passage of width 2.5m, these are denoted as *Passage-1* and *Passage-2* respectively.

Scenario	Average	Density	Capturing technique
Passage-1	40	2.76	optical tracking+camera
Passage-2	59	2.38	optical tracking+camera
Street-1	18	0.42	overhead camera
Street-2	11	0.37	overhead camera
Lab	2	-	motion capture h/w

Table 7.1: Summary of real-world crowd datasets used in this work.

For all these scenarios, it is assumed that the agent's goal position is the last tracked position in the dataset and compute $\mathbf{v}^{pref}$ accordingly. However, in some scenarios (such as people moving in a maze) this assumption may not hold. In such cases $\mathbf{v}^{pref}$ can be considered as part of the state and inferred along with other parameters using Bayesian


(a) Lab (small)

(b) Street (medium)

(c) Passage (large)

Figure 7.3: Rendering of real-world crowd trajectories used for evaluation

inferencing. Table 7.2 shows the result of the entropy metric for each simulation method on each dataset.

7.5 Analysis and Comparisons

The entropy metric is a measure of the amount of information missing from a crowd simulation algorithm, in terms of non-deterministic correction with respect to a given dataset.

Scenario	RVO-1	RVO-2	RVO-3	SFM-1	SFM-2	SFM-3	Steer-1	Steer-2	Steer-3
Passage-1	3.048	2.329	3.400	6.576	6.581	6.579	6.403	6.490	6.435
Passage-2	1.991	0.690	1.990	5.430	5.458	5.451	4.713	4.748	4.764
Street-1	2.744	3.156	2.800	4.500	4.707	4.665	2.979	3.569	3.838
Street-2	2.709	2.564	2.520	3.793	3.885	3.780	2.660	2.744	3.060
Lab	1.920	1.610	1.230	2.538	2.523	2.509	1.871	1.847	2.305

Table 7.2: Entropy metric for different simulation algorithms on various datasets.

This section highlights some key properties of the metric and compare this approach with prior techniques used to evaluate crowd simulation algorithms.

7.5.1 Metric Properties

The entropy metric has several useful properties, in terms of generating discriminative, ranakable results that are applicable to a variety of algorithms and data sources. Furthermore, it does not make any assumptions in terms of number of agents, their density, velocity, or the time-interval of the simulation.

Rankable results: The Entropy metric provides rankable results because it computes a single number in $\mathcal{R}$ and has a single dimension. The result can be ranked uniquely when there are no ties. If the entropy metric for $\hat{f}_1$ is lower than the entropy metric for $\hat{f}_2$, this implies that $\hat{f}_1$ is a better crowd simulator for that given real-world crowd dataset.

Discriminative: The data presented in Table 7.2 highlights the discriminative nature of the entropy metric. The metric creates a clear quantitative ranking with different simulation algorithms having different scores.

Generality: The metric makes very few assumptions about the simulation algorithm and the data. The underlying Bayesian smoothing framework can automatically translate between the representation of the validation data and the representation of crowd state in $\hat{f}$. As a result, the underlying crowd data used does not impose any restrictions on the simulation algorithm.

Many simulation algorithms represent the crowd state in terms the position and velocity of each agent (e.g. steering algorithm). Often the crowd trajectory data only has the positional information for the agents. One possibility is to approximate the velocity using finite differences. However, this computation can be sensitive to the noise in the data. Instead, the Bayesian smoothing framework infers a distribution of the most likely velocities using the simulator itself, and thereby has an estimate of the complete simulation

Scenario	Correlation
Passage-1 & Passage-2	.975
Street-1 & Street-2	.917
Street-1 & Passage-2	.585
Passage-1 & Street-2	.414

Table 7.3: The entropy metric results on similar datasets such as (Passage-1,Passage-2) or (Street-1,Street-2) are highly correlated. The metric shows lower correlation for different dataset pairs.

state for the algorithm. The Bayesian smoothing is not limited to just positions, velocities and accelerations, but can be applied to any other component of the simulation state.

7.5.2 Consistency

In order to maximizes usefulness, it is important that a metric provides consistency in its results. It is important that the results generated from one dataset should be similar to results generated from a similar dataset. The empirical results in Table 7.1 suggest such a property. Specifically, the results on similar benchmarks are well correlated with each other. In particular, the ordering from best to worse simulation algorithms for the benchmarks Passage-1 and Passage-2 does not differ significantly. As summarized in Table 7.3, the metric values computed for each simulation algorithm for (Passage-1, Passage-2) benchmarks have a high Pearson's correlation. There is a similar correlation for the (Street-1, Street-2) benchmarks.

The results from Table 7.3 suggest that the entropy metric produces (relatively) consistent results given similar inputs. However, if two datasets are very different, such as passage vs. street, different simulations can perform very differently and new values of the metric should be computed.

Another form of consistency is the robustness to minor variations, such as sensor noise, in the validation data. The EnKS framework from Algorithm 3 naturally handles sensor noise via the filtering process. This ability is demonstrated in Fig 7.4, where uniformly distributed noise is added to the validation data, while keeping Q constant and the Entropy metric score for RVO-1, SFM-2, and Steer-3 simulations are graphed for the Street-1 benchmark. The gradual increased entropy comes from the additional noise being attributed to the crowd simulations. However, the metric is robust to this noise and the relative ranking between the three simulators remains unchanged.

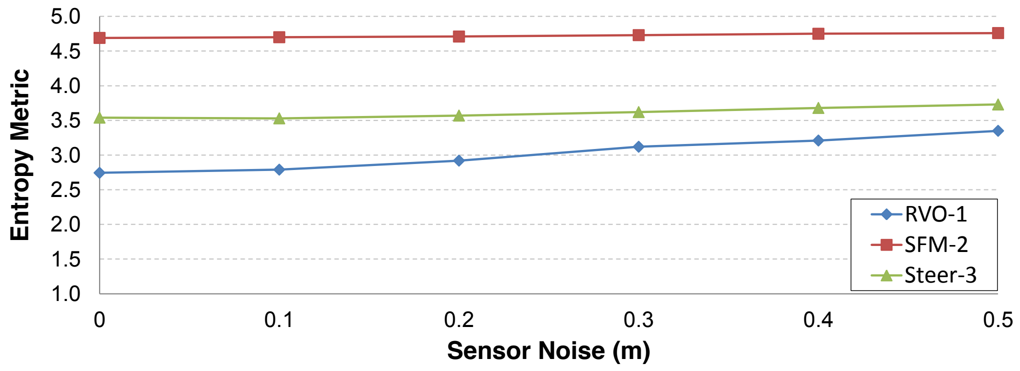


Figure 7.4: The impact of artificial error (uniformly distributed) on the Street-1 benchmark for different algorithms. The entropy metric is stable to this error and the relative ranking of different algorithms does not change.

7.5.2.1 Predictiveness

A metric is classified as predictive, if a good score indicates a good correlation between the simulation and the source of the data. In order to test the predictiveness of the entropy metric, synthetic trajectory data was created using a specific crowd simulation algorithm ($\hat{f}_1$). Later this trajectory data was used as an input to the evaluation algorithm, i.e. the output of $\hat{f}_1$ corresponds to $\mathbf{z}_0, \ldots, \mathbf{z}_k$ and used to compute the entropy metric for $\hat{f}_1$ as well as other crowd simulation algorithms (*e.g.* $\hat{f}_2$, $\hat{f}_3$). Ideally, each simulation technique would be the best predictor of it's own synthetic data (i.e. the entropy metric would be the minimum). Three different algorithms where used, RVO-1, SFM-2 and Steer-3, and synthetic trajectory data were generated for each of them for 100 timesteps, and used to the entropy metric for each simulation. The results are shown in Table 7.4. As the results show, this similarity matrix is optimized along the diagonal, demonstrating the predictiveness of the metric.

Simulator	RVO-1	SFM-2	Steer-3
RVO-1	**0.19**	3.17	3.16
SFM-2	4.34	**1.38**	1.58
Steer-3	2.26	1.66	**0.90**

Table 7.4: Entropy metric evaluation for a synthetic data. Each row corresponds to a different simulation algorithm. The algorithm listed in the column is used to generated the synthetic trajectory data. As expected, the entropy metric is minimized when the accuracy of an algorithm is evaluated on the synthetic trajectory data generated by the same algorithm.

Simulator	Metric	Improvement Method
RVO-A	6.07	-
RVO-B	2.09	Match radius from data
RVO-C	1.36	Match speed from data
RVO-D	0.75	Increase planning horizon

Table 7.5: As the accuracy of the RVO algorithm is improved based on the data from Lab benchmark, the entropy metric decreases. RVO-A is the worst algorithm for this benchmark and RVO-D is the best algorithm. There is a direct correlation between the entropy metric and accuracy of the simulation algorithm.

The predictiveness of the entropy metric can also be evaluated by measuring the changes on the metric which are aimed at improving the accuracy of an underlying crowd simulation algorithm. As a simple example, the two-agent Lab benchmark is used to evaluate the performance of a series of RVO-based algorithms on this benchmark.

I start with a fairly poor set of parameters for the RVO simulator (τ=0.1s, radius=0.1m, $|\mathbf{v}^{pref}|$ = 0.9 m/s), which is denoted as RVO-A. Next, different parameters are changed, one at a time, to improve its accuracy. First, the agent radius is reduced to be half of the closest approaching distance in the real-world data (RVO-B algorithm). Next, the average speed of each agent is increased to correspond to the average speed of the two participants in the real-world data (RVO-C algorithm). Finally, the planning horizon τ is increased, so the agents can better anticipate the collisions and thereby avoid it (RVO-D algorithm). By design, the RVO-D parameters resultin the best crowd simulation for this scenario and RVO-A is the worst. The entropy metric is computed for each of these simulations algorithms and there is a direct correlation between the entropy metric and the accuracy of the resulting simulation, as shown in Table 7.5.

7.5.3 Comparison with Prior Approaches

There is considerable work on evaluating and validating crowd simulations. These include SteerBench (Singh et al., 2009), which is a benchmark framework to evaluate steering behavior for virtual agents and includes a diverse set of virtual scenarios, metrics and a scoring system. Recently, Kapadia et al. (Kapadia et al., 2011) proposed a method to generate representative scenarios with statistical properties and statistical metrics to evaluate steering algorithms. Some of the crowd validation methods are based on study of presence in virtual environments (Pelechano et al., 2008b) or measuring the perceptual effects of illumination, position, orientation, and camera viewpoints on the plausibility of pedestrian formations in crowds (Ennis et al., 2011; Jarabo et al., 2012). The entropy metric is complimentary to these techniques, as it tries to evaluate the accuracy of a simulation algorithm with respect to given real-world crowd data.

Many crowd evaluation methods use real-world data or data-driven metrics to evaluate their accuracy. This includes comparing individual trajectories extracted from videos with the synthetic trajectories generated using a crowd simulation algorithm (Lerner et al., 2009). Other techniques try to compare the measurements of flow parameters such as crowd densities and velocity with those of a simulation (Seyfried et al., 2010). The work of (Pettré et al., 2009) used experimental data to model interactions between virtual walkers and validate their model by directly comparing the trajectories.

In contrast to these approaches, the entropy metric directly compares the underlying simulation algorithm with the real-world crowd data (e.g. trajectories) and automatically takes into account the noise in the data and uncertainty in the simulation algorithm. As a result, this evaluation approach is more general, and is applicable to a wide variety of simulation algorithms and real-world datasets from many different types of group motion and collective behaviors (such as fleets of vehicles, schools of fishes, flocks of birds, etc.).

7.6 Summary and Conclusions

This chapter introduced an entropy metric for evaluating crowd simulations against real-world crowd data. This metric provides a single real-valued number with a meaningful quantification that can be used to rank various crowd simulation algorithms in a predictive and consistent manner. To the best of my knowledge, this is the first attempt at designing an objective, quantitative evaluation metric that measures the accuracy of a crowd simulation algorithm with respect to real-world data and explicitly accounts for sensor noise, model uncertainty, and non-determinism. The metric was used it to evaluate the performance of steering behaviors, force-based models, and velocity-based crowd simulation algorithms on different real-world crowd trajectories.

7.6.1 Limitations

This approach does have some limitations. Most importantly, it assumes simulations work on continuous spaces and may not be able to evaluate discrete approaches (e.g. cellular automata). While the metric is simple and provides a consistent, rankable measure as a single real number, it may not be able to analyze some important aspects of crowd simulation algorithms (e.g. emergent behaviors) and which may require a separate, multi-dimensional metric.

The entropy metric is only defined for a given set of real-world crowd data and cannot directly compare different simulation algorithms in the absence of such data. Moreover, it is not invariant to properties of the data and may change with reparameterizations or changes in timestep. For example, a “null” simulation which assume each person never moves might score well at extremely small timesteps, but will score poorly when the data is collected at a higher speeds and the motion of human agents is apparent.

7.6.2 Future Work

Looking forward, beyond address the above limitations, I would also like to analyze the accuracy of other widely used crowd simulation techniques such as continuum techniques, vision-based steering, and cognitive models that can be described using continuous space formulation. Given other forms of real-world crowd data, such as behavior or personality (Lee et al., 2007), velocity or density information (Seyfried et al., 2010), the approach could be extend to compare corresponding characteristics of crowd simulation algorithms.

CHAPTER 8
Conclusion

Throughout this dissertation, I have addressed a variety of aspects of heterogeneous crowd simulation: from low-level collision avoidance to long-term path planning, from biomechanical variations to personality differences, from direct path comparisons to statistical validation methods. Simulating and analyzing crowds is an incredibly rich and rewarding area of study. As research in crowds continues, I expect further progress in all these areas. While there is always more to do, the work presented in this dissertation has made notable advances on many of the important issues in this field.

To summarize the main results presented in this dissertation:

Distributed Collision Avoidance I presented ClearPath, a method for decentralized, distributed collision avoidance based on geometric optimization methods.

Massively Parallel Crowd Simulation I presented a parallel implementation of ClearPath that utilizes both thread-level parallelism (multi-core) and data-level parallelism (SIMD/vector units) to exploit the recent hardware trends of increasing levels of parallel processing capability.

Biomechanically-Inspired Crowd Simulation Formulation: I presented a unified crowd simulation approach, based on a biomechanically-inspired interpretation of the well known *Principle of Least Effort*; it addresses both local collision avoidance and path planning based on minimizing caloric energy expenditure. I also demonstrated an efficient approach to implementing this method based on a local, greedy approximation.

Analysis of Least Effort Hypothesis I also demonstrated that by following the least *caloric* effort hypothesis for choosing an agent's paths, the resulting simulations produced trajectories which matched human data very closely – both in terms of qualitative features (*e.g.* lane formation, jamming, congestion avoidance) and quantitative metrics (*e.g.* path differences and flow rates).

Data-driven Approach to Parameter Variation I have presented the Reciprocal Collision Avoidance for Pedestrians (RCAP) algorithm which can simulate variation between agents based on various motion parameters. I also demonstrated how to learn these parameters from motion data gathered on real humans, and how to use the learned data to create data-driven crowd simulations.

Data-driven Approach to Personality Variation I presented a method to simulate the variation found in human crowds due to personality characteristics based on data collected from a user study, which was targeted at understanding the perception of personality in simulated crowds. I also propose a factor-model specifically targeted at analyzing various perceived personalities resulting from the proposed geometric collision avoidance methods.

Data-driven Metric for Evaluating Crowd Simulations I presented a metric to measure the extent to which different simulation methods were able to reproduce data collected on real crowds. This metric is robust to the presence of simulation uncertainty, sensor error, and the non-deterministic nature of human motion. I applied this metric to a variety of different simulations and different scenarios, to analyze existing crowd simulation methods.

8.1 Limitations

In a problem as multi-faceted as simulating crowds, there will also be important limitations to any proposed techniques, including those presented in this dissertation. Most importantly, the work presented here has only focused on trajectory computation. However, there are several other factors that are important to capture all aspects of people in crowds. For example, the body posture of agents is known to affect how emotion is perceived in simulated agents in a crowd (McHugh et al., 2010b), and could be integrated with the personality-based trajectory planning presented in Chapter 6. Similarly, facial expressions, body language, and even clothing choices should likely be reflective of one's personality, and should ultimately be reflected in a simulation.

In general, separating the crowd simulation problem from the crowd rendering problem, as was done is this dissertation, limits the amount the simulation state can effect the rendered state and vice-versa. Because this methods presented in this dissertation assumes that agents are hard 2D disks, while visualizing them as walking 3D beings, they are unable to capture effects directly relating to walk cycles (e.g., the unnaturalness of turning when both feet are on the ground), can not account for people's orientation changes and deformations (e.g., turning and squeezing to get past someone), and does not allow true 3D motion, such as jumping and climbing over people that might happen in a panic situation. Overcoming these limitations will likely require a more sophisticated model of an agent and a tighter coupling between the model used for simulation and the one used for rendering.

Additionally, this dissertation only looked at how behavioral variations between people effect their local navigation. However, differences in mood, opinions, and personality should cause differences in more long-term decision making. For example, people tend to group with their friends more than strangers, some people like to chose more direct paths, some people like more secluded paths, and most people tend to change these preferences over time or based on the current situation. Capturing these types of phenomena will likely require agents to change their final goals or intermediate milestones based on behavioral parameters. I believe the methods presented in this dissertation can be extended to capture several of these behaviors and should be an interesting avenue for future work.

8.2 Future Work

There are many exciting directions for future work, both in crowd simulation and in related domains such as multi-robot navigation. When teams or swarms of robots cooperate to accomplish a task they face many of the same problems that confront humans in crowds. Therefore, I believe many of the ideas proposed here can be applicable to robot navigation. Some of the key challenges in applying these techniques to robots will involve handling sensor noise, actuator uncertainty, and kinematic constraints. This direction has been pursued recently in the work of Snape et al. (Snape et al., 2011) and other researches in the field of biological inspired robotics. I believe further innovations are likely in this domain.

Due to the parallelizable nature of the algorithms presented in this dissertation, simulation methods presented here are well poised to be implemented on GPUs to exploit an increasingly popular source of many-core parallelism. More complex agent representations should also be explored. For example, all the work I presented here assumes agents (or their personal space) can be well represented by a disk. This prohibits the simulations from capturing any anisotropic behaviors displayed by people in crowds (for example a tendency for people to walk more closely side-by-side than front-to-back). Expanding models to account for an agent's orientation would be more computationally expensive, but has the potential to improve simulations. Additionally, more psychological factors can be incorporated into the crowd formulation including discomfort, panic, stress, groups, friends, family and other important factors that affect human crowds. Recent work has started to address some of these issues including psychologically-inspired models of stress (Kim et al., 2012) and mood (Durupınar, 2010), and I believe more can be done in this area.

From a planning perspective, all the methods proposed in this dissertation for collision avoidance involve the simple motion model of assuming that every agent will maintain a constant velocity for the foreseeable future. This assumption is a result of working in velocity space, which can only represent fixed-velocity trajectories. By moving to a higher dimensional space that can represent a larger diversity of trajectories, it may be possible to relax this assumption. Recent work with motion primitives (Hauser et al., 2008) and path sampling (Branicky et al., 2008) are particularly promising avenues for better understanding how to efficiently represent the space of all possible trajectories.

More broadly in crowd simulation, I believe one of the most important tasks is a deeper understanding of what level of accuracy different models provide. More progress in this area will likely come from close cooperation with architects, civil engineers, safety engineers, and others who are interested in using models of human motions to make real-world decisions. As simulations start playing an increasing role in decision making it is important to have a thorough understanding of the conditions under which these simulations are correct, and determine the level of accuracy needed to make good design decisions.

Simulating crowds present a truly limitless scope for investigation and improvement. Because crowds consist of humans interacting with each other, there are always new levels or more depths that can be added to models, both in terms of more human-like decision making and more realistic interactions. This complexity makes the simulation of crowds one of the most fascinating areas for research.

APPENDIX A

ClearPath Derivations and Proofs

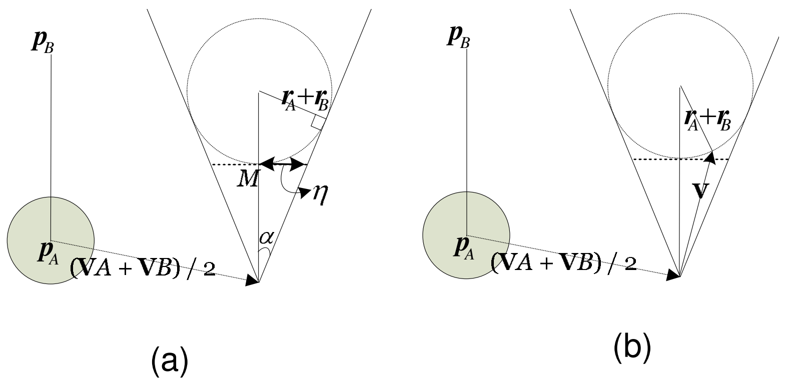


Figure A.1: Derivation of (a) $\Gamma_{AB}(\mathbf{v})$ and (b) $\gamma_{AB}(\mathbf{v})$

A.1 Derivation of $\Gamma_{AB}(\mathbf{v})$

For symbols, refer to Section 3.2

In Figure A.1(a), $\sin\alpha = \frac{(\mathbf{r}_A+\mathbf{r}_B)}{|\mathbf{p}_{AB}|}$. Therefore,

$$\eta = \tan\left(\sin^{-1}\frac{\mathbf{r}_A+\mathbf{r}_B}{|\mathbf{p}_{AB}|}\right) \times (|\mathbf{p}_{AB}| - (\mathbf{r}_A+\mathbf{r}_B))$$

M is computed as follows.

$$M = (|\mathbf{p}_{AB}| - (\mathbf{r}_A+\mathbf{r}_B)) \times \widehat{\mathbf{p}_{AB}} + \frac{\mathbf{v}_A+\mathbf{v}_B}{2}$$

As we define λ in Section 3.1,

$$\Gamma_{AB}(\mathbf{v}) = \lambda\left(M - \widehat{\mathbf{p}_{AB}^{\perp}} \times \eta,\ \mathbf{p}_{AB}^{\perp}\right)$$

A.2 Derivation of $\gamma_{AB}(\mathbf{v})$

For symbols, refer to Section 3.2

We first define $|\mathbf{v}| = \delta$ (refer to Figure A.1(b)).

$$t_{collision} = \frac{\mathbf{v}_A + \mathbf{v}_B}{2} + \frac{\delta}{2\Delta T}$$

Using the triangle cosine rule,

$$2\delta(\mathbf{p}_{AB}) \cdot \hat{v} = \delta^2 + |\mathbf{p}_{AB}|^2 - (\mathbf{r}_A + \mathbf{r}_B)^2$$
$$\delta = (\mathbf{p}_{AB}) \cdot \hat{v} - \sqrt{((\mathbf{p}_{AB}) \cdot \hat{v})^2 - |\mathbf{p}_{AB}|^2 + (\mathbf{r}_A + \mathbf{r}_B)^2}$$

Therefore,

$$\gamma_{AB}(\mathbf{v}) = \frac{\mathbf{v}_A + \mathbf{v}_B}{2} + \frac{(\mathbf{p}_{AB}) \cdot \hat{v} - \sqrt{((\mathbf{p}_{AB}) \cdot \hat{v})^2 - |\mathbf{p}_{AB}|^2 + (\mathbf{r}_A + \mathbf{r}_B)^2}}{2 \cdot \Delta T}$$

A.3 Proof of Lemma 1

We prove by contradiction. Let $\mathbf{v}_A^{new}$ not be on the edge of one of the segments of PCR and say it minimizes the distance from $\mathbf{v}_A^{des}$. Consider a circle of radius $\epsilon|\mathbf{v}_A^{new} - \mathbf{v}_A^{des}|$ centered at $\mathbf{v}_A^{new}$ ($\epsilon > 0$). The circle is completely outside the PCR, and we have a point $\mathbf{v}_A^{new} + \epsilon\,(\mathbf{v}_A^{des} - \mathbf{v}_A^{new})$ that is outside and at a distance of $(1-\epsilon)|\mathbf{v}_A^{new} - \mathbf{v}_A^{des}|$ from $\mathbf{v}_A^{des}$. Thus we arrive at a contradiction. Therefore, $\mathbf{v}_A^{new}$ must lie on the boundary segment of one of the neighbors.

A.4 Proof of Lemma 2

This follows from the fact that the truncated cones are convex regions. Hence once we identify a point that is outside the remaining truncated cones, the subsegment before it would be inside or outside. We use this fact to compute the inside/outside regions on the cone boundaries of the constraints.

A.5 Proof of Theorem 1

All the agents are in a colllision-free state if the resultant velocity is outside the VO of its neighbors. $FVO_{L}{}^{A}_{Bi}(\mathbf{v})$ and $FVO_{R}{}^{A}_{Bi}(\mathbf{v})$ ensure that the resultant velocity is outside the RVO of the neighboring agents. Furthermore, for every pair of agents, $FVO_{C}{}^{A}_{Bi}(\mathbf{v})$ ensures that we are on the same side of the line joining the centers of the agents. Hence it follows from RVO pair-wise collision-free property by Berg et al. (van den Berg et al., 2008a) that all pairs of agents are in a collision-free state w.r.t each other, and hence the system is in a collision-free state.

APPENDIX B

PLEdesrian Derivations and Proofs

B.1 Proof of Lemma 1 (Nature of Min. Energy Path)

Proof: Consider a small segment of length dx along the path. Assuming a speed of $\mathbf{v_x}$ along that segment, the total energy expended to traverse the distance dx is equal to: $E_x = m\int(e_s + e_w|\mathbf{v_x}|^2)dt = m(e_s + e_w|\mathbf{v_x}|^2)(dx/|\mathbf{v_x}|) = m(e_s/|\mathbf{v_x}| + e_w|\mathbf{v_x}|)dx$. In order to minimize the energy, $\frac{\partial E_x}{\partial|\mathbf{v_x}|}$ = 0 implies $m(-e_s/|\mathbf{v_x}|^2 + e_w)dx$ = 0. Therefore, $|\mathbf{v_x}|$ = $\sqrt{(e_s/e_w)}$. In order to minimize the total energy expended, the agent needs to traverse each segment of length dx (and hence the whole path) at a speed of $\sqrt{(e_s/e_w)}$. For a total path length of L, the total energy expended evaluates to $m(\sqrt{(e_s e_w)} + \sqrt{(e_s e_w)})L = 2mL\sqrt{(e_s e_w)}$. The above statement also implies that the agent needs to take the shortest path available from its source to destination in order to reduce the total distance traversed, and correspondingly the total effort (or energy) expended.

B.2 Objective Function of Eqn. 3.4 is a convex function

Refer to Section 3.3 for notations and figures. $E(\mathbf{v}_A^{new}) = m\tau(e_s + e_w|\mathbf{v}_A^{new}|^2) + 2m|\mathbf{G}_A - \mathbf{p}_A - \tau\mathbf{v}_A^{new}|\sqrt{(e_s e_w)}$
= $m\tau e_s + m\tau e_w|\mathbf{v}_A^{new}|^2 + 2m\tau\sqrt{(e_s e_w)}|\mathbf{v}_A^{new} - (\mathbf{G}_A - \mathbf{p}_A)/\tau|$ It follows from first principles of convex functions (Boyd and Vandenberghe, 2004) that $|\mathbf{v}_A^{new}|^2$ and $|\mathbf{v}_A^{new} - (\mathbf{G}_A - \mathbf{p}_A)/\tau|$ are individually convex functions. A weighted sum (with all positive weights) of convex functions is also a **convex function**. Since both $m\tau e_w$ and $m\tau\sqrt{(e_s e_w)}$ are greater than *zero*, $E(\mathbf{v}_A^{new})$ is convex.

B.3 Minima of Eqn. 3.4 lies on the boundary of PV_A

It follows from Lemma 1 that $\mathbf{v}_A^{des} = \sqrt{(e_s/e_w)}(\widehat{\mathbf{G}_A - \mathbf{p}_A})$. Let $\mathbf{v}_A^{new} = (x,y)$. To find the minima of the objective function, we set $\frac{\partial E(\mathbf{v}_A^{new})}{\partial x} = 0$ and $\frac{\partial E(\mathbf{v}_A^{new})}{\partial y} = 0$, which implies $x/y = (\mathbf{G}_A - \mathbf{p}_A)_x/(\mathbf{G}_A - \mathbf{p}_A)_y$. Also, $x^2 + y^2 = e_s/e_w$. Hence, $\mathbf{v}_A^{new} = \mathbf{v}_A^{des}$. In case $\mathbf{v}_A^{des} \notin PV_A$, we need to compute the optimal point within the region of permissible velocities (PV_A). We now prove $\mathbf{v}_A^{new}$ lies on a linear boundary segment by contradiction. Assume the optimal velocity $\mathbf{v}'$ (= $\mathbf{v}_A^{new}$) lies strictly inside the PV_A region. Consider the segment joining $\mathbf{v}'$ to $\mathbf{v}_A^{des}$. Since $E(\mathbf{v}_A^{new})$ is convex, its projection function along any line would also be convex (Boyd and Vandenberghe, 2004). Since $\mathbf{v}_A^{des}$ is the *global minimum*, $E(\mathbf{v}_A^{new})$ is *strictly increasing* along the line segment from $\mathbf{v}_A^{des}$ to $\mathbf{v}'$. Since $\mathbf{v}'$ is inside PV_A, the segment intersects the PV_A at a point for which the objective function evaluates to a smaller value. Hence $\mathbf{v}'$ is not the optimal value, and we have arrived at a contradiction.

B.4 Proofs of Lemma 2 and Lemma 3

Proof of smoothness (**Lemma 2**). **Proof:** To show that the trajectories generated are C1-continuous, we need to prove that the paths are C0-continuous, and their derivative (i.e. velocity) is also C0-continuous. We first assume that discrete time steps of the underlying simulation approach zero in the limit. Our simulation displaces the agent by the product of the instantaneous agent velocity and the time change (Euler integration). Since time varies C0-continuously, the agent traverses C0-continuous trajectory. In order to prove that the velocity of the agent is C0-continuous, we need to prove that our energy minimization formulation (Eqn. 4) computes velocities that vary in a C0-continuous fashion.

Consider the agent A. We assume that the region of permissible velocities changes in a C0-fashion (i.e. for any boundary curve of PV_A, and a point on that curve, the path traced out by that point, with change in time, is C0-continuous). Furthermore, the boundary curves themselves are continuous, with at least C0 continuity at their end points

Consider the boundary segment along which the energy function is minimized. Note that all the coefficients in Eqn. 4 are either constant or vary with the positions of the agent and its neighbors. To minimize Eqn. 4, we set the partial derivative of the objective function of Eqn. 4 to be equal to zero. This results in finding the roots of a polynomial equation, whose coefficients trace a C0-continuous path. A polynomial equation with C0-continuous coefficients also has C0-continuous roots (Coolidge, 1908). Hence as long as the minimum lies on a specific boundary curve, the path traced by the velocity is C0-continuous. Furthermore, as the minima changes from one boundary curve to another curve, the partial derivative at their common end point should also evaluate to zero (follows from the C0-continuity of the PV_A boundary curves at their end points).

BIBLIOGRAPHY

Arechavaleta, G., Laumond, J. P., Hicheur, H., and Berthoz, A. (2008). An optimality principle governing human walking. *IEEE Transactions on Robotics*, 24(1):5–14.

Bayazit, O. B., Lien, J.-M., and Amato, N. M. (2002). Better group behaviors in complex environments with global roadmaps. *Int. Conf. on the Sim. and Syn. of Living Sys. (Alife)*, pages 362–370.

Bon, G. L. (1895). *The Crowd: A Study of the Popular Mind.* Reprint available from Dover Publications.

Borkar, S. and Chien, A. (2011). The future of microprocessors. *Communications of the ACM*, 54(5):67–77.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Branicky, M., Knepper, R., and Kuffner, J. (2008). Path and trajectory diversity: Theory and algorithms. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1359–1364. IEEE.

Browning, R. and Kram, R. (2005). Energetic cost and preferred speed of walking in obese vs. normal weight women. *Obesity*, 13(5):891–899.

Bruderlin, A. and Calvert, T. (1996). Knowledge-driven, interactive animation of human running. In *Graphics Interface*, pages 213–221. Citeseer.

Calvert, T., Bruderlin, A., Dill, J., Schiphorst, T., and Weilman, C. (1993). Desktop animation of multiple human figures. *Computer Graphics and Applications, IEEE*, 13(3):18–26.

Cameron, S. (1997). Enhancing gjk: Computing minimum and penetration distances between convex polyhedra. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 3112–3117. IEEE.

Cattell, R. and Eber, H. (1972). The 16 personality factor questionnaire. *Institute for Personality and Ability Testing*.

Channon, P. H., Hopkins, S. H., and Phan, D. T. (1992). Derivation of optimal walking motions for a biped walking robot. In *Robotica*, volume 10, pages 165–172.

Chenney, S. (2004). Flow tiles. *Proc. 2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

Chraibi, M., Seyfried, A., and Schadschneider, A. (2010). Generalized centrifugal-force model for pedestrian dynamics. *Phys. Rev. E*, 82(4):046111.

Coolidge, J. L. (1908). The continuity of the roots of an algebraic equation. In *The Annals of Mathematics*, volume 9, pages 116–118.

Costa, P. and McCrae, R. (1992). *Revised NEO Personality Inventory (NEO PI-R) and Neo Five-Factor Inventory (NEO-FFI).* Psychological Assessment Resources.

Courty, N. and Musse, S. (2004). Fastcrowd: Real-time simulation and interaction with large crowds based on graphics hardware. Technical report, INRIA. http://home.tele2.fr/ncourty/fastCrowd.htm.

Curtis, S., Guy, S., Zafar, B., and Manocha, D. (2011). Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 128–135. IEEE.

de Berg, M., Cheong, O., van Breveld, M., and Overmars, M. (2008). *Computaional Geometry: Algorithms and Agglications.* Springer-Verlag.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dobbyn, S., Hamill, J., O'Conor, K., and O'Sullivan, C. (2005). Geopostors: a real-time geometry/impostor crowd rendering system. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 95–102. ACM.

Draycott, S. G. and Kline, P. (1995). The big three or the big five-the epq-r vs the neo-pi: a research note, replication and elaboration. *Personality and Individual Differences.*

Durupınar, F. (2010). *From audiences to mobs: Crowd simulation with psychological factors.* PhD thesis, BILKENT UNIVERSITY.

Durupinar, F., Allbeck, J., Pelechano, N., and Badler, N. (2008). Creating crowd variation with the OCEAN personality model. In *Autonomous agents and multiagent systems.*

Durupinar, F., Pelechano, N., Allbeck, J., Gudukbay, U., and Badler, N. (2011). How the ocean personality model affects the perception of crowds. *IEEE Computer Graphics and Applications*, 31(3):22–31.

Elgammal, A., Rosenhahn, B., and Klette, R., editors (2007). *Human Motion - Understanding, Modeling, Capture and Animation*. Springer.

Ennis, C., Peters, C., and O'Sullivan, C. (2011). Perceptual effects of scene context and viewpoint for virtual pedestrian crowds. *ACM Trans. Appl. Percept.*, 8:10:1–10:22.

Ericson, C. (2004). *Real-Time Collision Detection.* Morgan Kaufmann.

Evensen, G. (2003). The ensemble kalman filter: theoretical formulation. *Ocean Dynamics*, 55:343–367.

Eysenck, H. and Eysenck, M. (1985). *Personality and individual differences: A natural science approach.* Plenum Press New York.

Eysenck, S. and Eysenck, H. (1977). The place of impulsiveness in a dimensional system of personality description. *The British journal of social and clinical psychology*, 16(1):57.

Faverjon, B. and Tournassoud, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. *ICRA*, pages 1152–1159.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. In *IJRR*.

Foudil, C. and Noureddine, D. (2027). Collision avoidance in crowd simulation with priority rules. *European Journal of Scientific Research*, 15(1).

Fruin, J. (1971). Pedestrian planning and design. New York. Metropolitan Association of Urban Designers and Environmental Planners.

Fulgenzi, C., Spalanzani, A., and Laugier, C. (2007). Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In *ICRA*, pages 1610–1616.

Funge, J., Tu, X., and Terzopoulos, D. (1999). Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *ACM SIGGRAPH*, pages 29–38. ACM Press.

Gayle, R., Sud, A., Andersen, E., Guy, S., Lin, M., and Manocha, D. (2009). Real-time navigation of independent agents using adaptive roadmaps. *IEEE TVCG*, (Jan/Feb):34–38.

Guy, S. J., Chhugani, J., Curtis, S., Lin, M. C., and Manocha, D. (2010a). Pledestrians: A least-effort approach to crowd simulation. In *Symposium on Computer Animation*. ACM.

Guy, S. J., Chhugani, J., Kim, C., Satish, N., Lin, M. C., Manocha, D., and Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Symposium on Computer Animation*. ACM.

Guy, S. J., Curtis, S., Lin, M. C., and Manocha, D. (2012). Least-effort trajectories lead to emergent crowd behaviors. *Phys. Rev. E*, 85:016110.

Guy, S. J., Kim, S., Lin, M., and Manocha, D. (2011). Simulating heterogeneous crowd behaviors using personality trait theory. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 43–52. The Eurographics Association.

Guy, S. J., Lin, M. C., and Manocha, D. (2010b). Modeling collision avoidance behavior for virtual humans. In *International Conference on Autonomous Agents and Multiagent Systems*.

Hall, E. T. (1963). A system for the notation of proxemic behavoir. *American Anthropologist*, 65(5):1003–1026.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107.

Hartmann, D. (2010). Adaptive pedestrian dynamics based on geodesics. *New Journal of Physics*, 12:043032.

Harvey, R. J., Murry, W. D., and Stamoulis, D. T. (1995). Unresolved issues in the dimensionality of the myers-briggs type indicator. *Educational and Psych. Measurement*.

Hauser, K., Bretl, T., Harada, K., and Latombe, J. (2008). Using motion primitives in probabilistic sample-based planning for humanoid robots. *Algorithmic Foundation of Robotics VII*, pages 507–522.

Helbing, D., Buzna, L., Johansson, A., and Werner, T. (2005). Self-organized pedestrian crowd dynamics and design solutions: Experiments, simulations and design solutions. *Transportation Science*, 39(1):1–24.

Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407:487–490.

Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51:4282.

Hoogendoorn, S. and Bovy, P. (2004). Pedestrian route-choice and activity scheduling theory and models. *Transportation Research Part B: Methodological*, 38(2):169–190.

Hopkins, B., Churchill, A., Vogt, S., and Rönnqvist, L. (2004). Braking reaching movements: a test of the constant tau-dot strategy under different viewing conditions. *Journal of motor behavior*, 36(1):3–12.

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

Hughes, R. (2003). The flow of human crowds. *Annual review of fluid mechanics*, 35(1):169–182.

Inman, V., Ralston, H., Todd, F., and Lieberman, J. (1981). *Human walking*. Williams & Wilkins.

Jarabo, A., Eyck, T. V., Sundstedt, V., Bala, K., Gutierrez, D., and O'Sullivan, C. (2012). Crowd light: Evaluating the perceived fidelity of illuminated dynamic scenes. *Proc. of Eurographics*. to appear.

Jin, X., Xu, J., Wang, C. C. L., Huang, S., and Zhang, J. (2008). Interactive control of large crowd navigation in virtual environment using vector field. In *IEEE CG&A*.

Johansson, A., Helbing, D., and Shukla, P. B. (2008). Specification of a microscopic pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex System*, 10:271–288.

Ju, E., Choi, M. G., Park, M., Lee, J., Lee, K. H., and Takahashi, S. (2010). Morphable crowds. *ACM Transactions on Graphics*, 29(6):140.

Juang, J. (1999). Minimal energy control on trajectory generation. *International Conference on Information Intelligence and Systems*, page 204.

Kagarlis, M. (2002). Method and apparatus of simulating movement of an autonomous entity through an environment. United States Patent No. US 7,188,056.

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.

Kanehiro, F., Lamiraun, F., Kanoun, O., Yoshida, E., and Laumond, H. (2008). A local collision avoidance method for non-strictly convex polyhedral. In *Robotics: Science and Systems*.

Kang, Y., Park, S., and Lee, E. (1998). An efficient control over human running animation with extension of planar hopper model. In *Pacific Graphics*, pages 169–176.

Kapadia, M., Wang, M., Singh, S., Reinman, G., and Faloutsos, P. (2011). Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 53–62.

Karamouzas, I., Heil, P., Beek, P., and Overmars, M. (2009). A predictive collision avoidance model for pedestrian simulation. *Proc. of Motion in Games*, pages 41–52.

Kavan, L., Dobbyn, S., Collins, S., Žára, J., and O'Sullivan, C. (2008). Polypostors: 2d polygonal impostors for 3d crowds. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 149–155. ACM.

Kavraki, L., Svestka, P., Latombe, J., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580.

Kim, S., Guy, S., Manocha, D., and Lin, M. (2012). Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Symposium on Interactive 3D Graphics and Games*. ACM.

Kluge, B. and Prassler, E. (2007). Reflective navigation: Individual behaviors and group behaviors. In *ICRA*, pages 4172–4177.

Klüpfel, H., Schreckenberg, M., and Meyer-König, T. (2005). Models for crowd movement and egress simulation. *Traffic and Granular Flow*, pages 357–372.

Ko, H. and Badler, N. (1993). Straight line walking animation based on kinematic generalization that preserves the original characteristics. *Graphics Interface*.

Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 473–482. ACM.

Kratz, L. and Nishino, K. (2011). Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (99):1–1.

Kretz, T. (2009). Pedestrian traffic: on the quickest path. *Journal of Statistical Mechanics: Theory and Experiment*, 2009:P03012.

Kretz, T., Grünebohm, A., and Schreckenberg, M. (2006). Experimental study of pedestrian flow through a bottleneck. *Journal of Statistical Mechanics: Theory and Experiment*, page P10014.

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.

Latombe, J.-C. (1991). *Robot motion planning*. Springer.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/).

Lee, K. H., Choi, M. G., Hong, Q., and Lee, J. (2007). Group behavior from video: a data-driven approach to crowd simulation. In *Symposium on Computer Animation*, pages 109–118.

Lerner, A., Chrysanthou, Y., Shamir, A., and Cohen-Or, D. (2009). Data driven evaluation of crowds. In *Proceedings of the 2nd International Workshop on Motion in Games*, pages 75–83.

Lin, M. (1993). *E cient Collision Detection for Animation and Robotics*. PhD thesis, PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

Lozano-Pérez, T. and Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570.

Luebke, D. (2003). *Level of detail for 3D graphics*. Morgan Kaufmann Pub.

Masucci, A. P., Smith, D., Crooks, A., and Batty, M. (2009). Random planar graphs and the london street network. *The European Physical Journal B - Condensed Matter and Complex Systems*, 71(2):259–271.

Maury, B. and Venel, J. (2008). A mathematical framework for a crowd motion model. *Comptes Rendus Mathematique*, 346(23-24):1245–1250.

McDonnell, R., Larkin, M., Hernández, B., Rudomin, I., and O'Sullivan, C. (2009). Eye-catching crowds: saliency based selective variation. *ACM Transactions on Graphics (TOG)*.

McHugh, J., McDonnell, R., O'Sullivan, C., and Newell, F. (2010a). Perceiving emotion in crowds: the role of dynamic body postures on the perception of emotion in crowded scenes. *Experimental brain research*.

McHugh, J., McDonnell, R., O'Sullivan, C., and Newell, F. (2010b). Perceiving emotion in crowds: the role of dynamic body postures on the perception of emotion in crowded scenes. *Experimental brain research*, 204(3):361–372.

McLachian, G. and Krishnan, T. (1996). *The EM Algorithm and Extensions*. John Wiley and Sons.

Mehran, R., Oyama, A., and Shah, M. (2009). Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935–942.

Mohr, E., Kranz, D. A., and Halstead Jr., R. H. (1990). Lazy task creation: a technique for increasing the granularity of parallel programs. In *Proc. of the ACM conference on LISP and functional programming*.

Moussaïd, M., Helbing, D., and Theraulaz, G. (2011). How simple rules determine pedestrian behavior and crowd disasters. *Proceedings of the National Academy of Sciences*, 108(17):6884.

Müller, K. (1981). *Zur Gestaltung und Bemessung von Fluchtwegen für die Evakuierung von Personen aus Bauwerken auf der Grundlage von Modellversuchen*. PhD thesis, Technische Hochschule Otto von Guericke Magdeburg.

Muramatsu, M. and Nagatani, T. (2000). Jamming transition in two-dimensional pedestrian traffic. *Physica A: Statistical Mechanics and its Applications*, 275(1-2):281–291.

Musse, S. R. and Thalmann, D. (1997). A model of human crowd behavior: Group interrelationship and collision detection analysis. *Computer Animation and Simulation*, pages 39–51.

Myers, I., McCaulley, M., Quenk, N., and Hammer, A. (1999). *MBTI manual*. Consulting Psychologists Press.

Nagai, R., Fukamachi, M., and Nagatani, T. (2006). Evacuation of crawlers and walkers from corridor through an exit. *Physica A: Statistical Mechanics and its Applications*, 367:449–460.

Narain, R., Golas, A., Curtis, S., and Lin, M. C. (2009). Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.*, 28(5).

Nelson, H. E. and Maclennan, H. A. (1995). Emergency movement. In *SFPE handbook of fire protection engineering*.

Ondrej, J., Pettre, J., Olivier, A., and Donikan, S. (2010). A synthetic-vision based steering approach for crowd simulation. *ACM Trans. on Graphics*, 29(4):123:1–123:9.

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., and Purcell, T. (2007). A survey of general-purpose computation on graphics hardware. *Computer graphics forum*, 26(1):80–113.

Paris, S., Pettre, J., and Donikian, S. (2007a). Pedestrian reactive navigation for crowd simulation: a predictive approac. *Computer Graphics Forum*, 26:665–674.

Paris, S., Pettré, J., and Donikian, S. (2007b). Pedestrian reactive navigation for crowd simulation: a predictive approach. In *Computer Graphics Forum*, volume 26, pages 665–674.

Patil, S., van den Berg, J., Curtis, S., Lin, M. C., and Manocha, D. (2011). Directing crowd simulations using navigation fields. *IEEE Trans. on Vis. and Comp. Graphics*, 17(2):244–254.

Pedica, C. and Vilhjalmsson, H. (2008). Social perception and steering for online avatars. In *Proceedings of the 8th International Conference on Intelligent Virtual Agents*, pages 104–116.

Peiper, D. (1968). *The kinematics of manipulators under computer control*. PhD thesis, Stanford University.

Pelechano, N., Allbeck, J., and Badler, N. (2008a). *Virtual Crowds: Methods, Simulation, and Control*. Morgan and Claypool Publishers.

Pelechano, N., Allbeck, J. M., and Badler, N. I. (2007). Controlling individual agents in high-density crowd simulation. *Proc. Symposium on Computer Animation*, pages 99–108.

Pelechano, N., Allbeck, J. M., and Badler, N. I. (2008b). *Virtual Crowds: Methods, Simulation and Control*. Morgan and Claypool Publishers.

Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268.

Pervin, L. (2003). *The Science of Personality*. Oxford University Press, Oxford.

Pettré, J., Ondřej, J., Olivier, A., Cretual, A., and Donikian, S. (2009). Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Symposium on Computer Animation*. ACM.

Predtetschenski, V. and Milinski, A. (1971). *Personenströme in Gebäuden-Berechnungsmethoden für die Projektierung*. Staatsverlag der Deutschen.

Quinn, M. J., Metoyer, R. A., and Hunter-zaworski, K. (2003). Parallel implementation of the social forces model. In *in Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics*, pages 63–74.

Rehm, M., André, E., and Nischt, M. (2005). Let's come together—social navigation behaviors of virtual and real humans. *Intelligent Technologies for Interactive Entertainment*, pages 124–133.

Reise, S. P., Waller, N. G., and Comrey, A. L. (2000). Factor analysis and scale revision. *Pshycological Assesment*.

Reynolds, C. (2006). Big fast crowds on ps3. In *Sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 113–121. ACM.

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

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

Rodriguez, M., Ali, S., and Kanade, T. (2009). Tracking in unstructured crowded scenes. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1389–1396.

Roussel, L., de Wit, C. C., and Goswami, A. (1998). Generation of energy optimal complete gait cycles for biped robots. *IEEE Transactions on Robotics and Automation*, 16:2036–2041.

Salvit, J. and Sklar, E. (2011). Toward a Myers-Briggs Type Indicator Model of Agent Behavior in Multiagent Teams. *Multi-Agent-Based Simulation XI*, pages 28–43.

Sarmady, S., Haron, F., and Talib, A. (2010). Simulating Crowd Movements Using Fine Grid Cellular Automata. In *UKSim*.

Schadschneider, A., Chowdhury, D., and Nishinari, K. (2011). *Stochastic Transport in Complex Systems: From Molecules to Vehicles*. Elsevier.

Schneider, J., Garatly, D., Srinivasan, M., Guy, S., Curtis, S., Cutchin, S., Manocha, D., Lin, M., and Rockwood, A. (2011). Towards a digital makkah–using immersive 3d environments to train and prepare pilgrims. In *international Conference on Digital Media and its Applications in Cultural Heritage (DMACH)*.

Schrater, P., Knill, D., and Simoncelli, E. (2000). Mechanisms of visual motion detection. *Nature Neuroscience*.

Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., and Hanrahan, P. (2008). Larrabee: A Many-Core x86 Architecture for Visual Computing. *ACM Trans. Graph.*, 27(3):1–15.

Seyfried, A., Boltes, M., Kähler, J., Klingsch, W., Portz, A., Rupprecht, T., Schadschneider, A., Steffen, B., and Winkens, A. (2008). Enhanced empirical data for the fundamental diagram and the flow through bottlenecks,. *Pedestrian and Evacuation Dynamics*.

Seyfried, A., Boltes, M., Kähler, J., Klingsch, W., Portz, A., Rupprecht, T., Schadschneider, A., Steffen, B., and Winkens, A. (2010). Enhanced empirical data for the fundamental diagram and the flow through bottlenecks. pages 145–156. Springer.

Seyfried, A., Passon, O., Steffen, B., Boltes, M., Rupprecht, T., and Klingsch, W. (2009). New insights into pedestrian flow through bottlenecks. *Transportation Science*, 43(3):395–406.

Shao, W. and Terzopoulos, D. (2005). Autonomous pedestrians. In *Proc. of the Symposium on Computer Animation*, pages 19–28.

Shiller, Z., Large, F., and Sekhavat, S. (2001). Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3716–3721.

Singh, S., Kapadia, M., Reinmann, G., and Faloutsos, P. (2009). Steerbench: A benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds*, 20:533–548.

Snape, J., Berg, J., Guy, S., and Manocha, D. (2011). The hybrid reciprocal velocity obstacle. *Robotics, IEEE Transactions on*, (99):1–11.

Snape, J., Van Den Berg, J., Guy, S., and Manocha, D. (2009). Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5917–5922. Ieee.

Still, G. (2000). *Crowd Dynamics*. PhD thesis, University of Warwick, UK.

Sud, A., Andersen, E., Curtis, S., Lin, M., and Manocha, D. (2007a). Real-time path planning for virtual agents in dynamic environments. *Proc. of IEEE VR*, pages 91–98.

Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., and Manocha, D. (2007b). Real-time navigation of independent agents using adaptive roadmaps. *Proc. of ACM VRST*, pages 99–106.

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

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

Tecchia, F., Loscos, C., and Chrysanthou, Y. (2002). Image-based crowd rendering. *Computer Graphics and Applications, IEEE*, 22(2):36–43.

Thalmann, D., O'Sullivan, C., Ciechomski, P., and Dobbyn, S. (2006). *Populating Virtual Environments with Crowds*. Eurographics 2006 Tutorial Notes.

Tolwinski, R. (2002). Simulation of multiagent system in parallel and distributed programming. In *Proceedings of the 16th European Simulation Multiconference on Modelling and Simulation 2002*, pages 57–59. SCS Europe.

Treuille, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1160–1168, New York, NY, USA. ACM.

Turner, R. H. and Killian, L. M. (1987). *Collective Behavior*. Prentice Hall.

van den Berg, J., Guy, S., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. *Robotics Research*, pages 3–19.

van den Berg, J., Lin, M. C., and Manocha, D. (2008a). Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA*.

van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. (2008b). Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 139–147, Redwood City, California. ACM.

Weidmann, U. (1992). Transporttechnik der fußgänger. *Strasse und Verkehr*, 78(3):161–169.

Whittle, M. (2002). *Gait analysis: an introduction*. Butterworth-Heinemann Medical.

Witkin, A. and Popovic, Z. (1995). Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108. ACM.

Yersin, B., Maim, J., Ciechomski, P., Schertenleib, S., and Thalmann, D. (2005). Steering a virtual crowd based on a semantically augmented navigation graph. In *VCROWDS*.

Yersin, B., Maïm, J., Morini, F., and Thalmann, D. (2008). Real-time crowd motion planning. *The Visual Computer*, 24:859–870.

Yu, Q. and Terzopoulos, D. (2007). A decision network framework for the behavioral animation of virtual humans. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 119–128.

Yu, W. and Johansson, A. (2007). Modeling crowd turbulence by many-particle simulations. *Phys. Rev. E*, 76(4):046105.

Yu, W. J., Chen, R., Dong, L. Y., and Dai, S. Q. (2005). Centrifugal force model for pedestrian dynamics. *Phys. Rev. E*, 72:026112.

Zipf, G. (1965). *Human behavior and the principle of least effort: An introduction to human ecology*. Hafner, New York.