VELOCITY-SPACE REASONING FOR INTERACTIVE SIMULATION OF DYNAMIC CROWD BEHAVIORS

Sujeong Kim

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

Chapel Hill
2015

Approved by:

Dinesh Manocha

Ming C. Lin

Jan-Michael Frahm

Stephen J. Guy

Carol O'Sullivan

ABSTRACT

Sujeong Kim: Velocity-Space Reasoning for Interactive Simulation of Dynamic Crowd Behaviors
(Under the direction of Dinesh Manocha and Ming C. Lin)

The problem of simulating a large number of independent entities, interacting with each other and moving through a shared space, has received considerable attention in computer graphics, biomechanics, psychology, robotics, architectural design, and pedestrian dynamics. One of the major challenges is to simulate the dynamic nature, variety, and subtle aspects of real-world crowd motions. Furthermore, many applications require the capabilities to simulate these movements and behaviors at interactive rates.

In this thesis, we present interactive methods for computing trajectory-level behaviors that capture various aspects of human crowds. At a microscopic level, we address the problem of modeling the local interactions. First, we simulate dynamic patterns of crowd behaviors using Attribution theory and General Adaptation Syndrome theory from psychology. Our model accounts for permanent, stable disposition and the dynamic nature of human behaviors that change in response to the situation. Second, we model physics-based interactions in dense crowds by combining velocity-based collision avoidance algorithms with external forces. Our approach is capable of modeling both physical forces and interactions between agents and obstacles, while also allowing the agents to anticipate and avoid upcoming collisions during local navigation.

We also address the problem at macroscopic level by modeling high-level aspects of human crowd behaviors. We present an automated scheme for learning and predicting individual behaviors from real-world crowd trajectories. Our approach is based on Bayesian learning algorithms combined with a velocity-based local collision avoidance model. We further extend our method to learn time-varying trajectory behavior patterns from pedestrian trajectories. These behavior patterns can be combined with local navigation algorithms to generate crowd behaviors that are similar to those observed in real-world videos. We highlight their performance for pedestrian navigation, architectural design and generating dynamic behaviors for virtual environments.

ACKNOWLEDGEMENTS

I have been blessed to have the support of many people during my time at UNC. They have supported me during both the good and tough times and I would like to thank them for making this thesis a reality. First of all, I deeply thank my advisors, Dinesh Manocha and Ming C. Lin. They not only provided excellent vision and feedback about my goals and results, but also led me to extend my potential much further than what I thought would be possible.

I also cannot give enough thanks and credit to Stephen J. Guy who is my friend (thanks for the support during the tough times), collaborator (for the many papers we worked together) and committee member. I also am thankful for the support from the rest of my committee members: Carol O'Sullivan, who was also my mentor at Disney Research, helped me many times with great suggestions and feedback for my work. I appreciate Jan-Michael Frahm for introducing me to computer vision through his class which made me actually act on conducting a research in a new direction.

I cannot forget my friends and collaborators: Sean Curtis provided amazing tools and lots of advice both for research and school life. Aniket Bera, David Wilkie, and Wenxi Liu helped me a lot with my later work related to computer vision and robotics. To my other friends and GAMMA team members, too many to list here, I will never forget the great feedback and help in developing and polishing my work. I thank my friend Lenin Singaravelu, who patiently listened to all my complaints about anything and also helped proofreading the thesis.

Finally, I thank my parents so much. They were almost in the opposite side of the earth from Chapel Hill, but I could always feel their endless love and support for me.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF ABBREVIATIONS xviii

1 Introduction 1

1.1 Motivation 1

1.2 Crowd Simulation 2

1.2.1 Local Collision Avoidance in Velocity-Space Reasoning 3

1.2.2 Modeling Local Interactions 3

1.2.3 High-level Behavior Modeling 5

1.2.4 Crowd Simulation for Robotics 5

1.3 Thesis Statement 6

1.4 Main Results 7

1.4.1 Modeling Dynamic, Heterogeneous Behaviors 8

1.4.2 Modeling Physical Interactions 10

1.4.3 Adaptive Motion Model: Learning Individual Motion Parameters from Real World Crowd 11

1.4.4 Time-varying Crowd Behavior Learning for Crowd Simulation 12

2 Interactive Simulation of Dynamic Crowd Behaviors using General Adaptation Syndrome Theory 15

2.1 Introduction 15

2.2 Related Work 17

2.2.1 Stress 17

2.2.2 Crowd Simulation 18
2.2.2.1 Interactive Crowd Simulation 18
2.2.2.2 Behavior Modeling 18
2.3 Preliminaries and Overview 19
2.3.1 Psychological Models of Stress 20
2.3.2 General Adaptation Syndrome (GAS) 20
2.3.3 Approximation of the GAS Model 21
2.3.4 Overview of our Approach 22
2.4 Modeling Stress and Stressors 22
2.4.1 Perceived Stress 23
2.4.2 Stressor Prototypes 24
2.4.3 Stress Model 25
2.5 Behavior Mapping 26
2.5.1 Incorporation of Behavior Changes 26
2.5.2 Coupling with Personality Attributes 27
2.5.3 Modeling from Real-world Data 28
2.6 Results 28
2.6.1 Emergent Behaviors 29
2.6.2 Validation 30
2.6.3 Performance Results 32
2.7 Conclusion 33
3 Velocity-Based Modeling of Physical Interactions in Dense Crowds 34
3.1 Introduction 34
3.2 Related Work 37
3.2.1 Multi-Agent Motion Models 37
3.2.2 Dense Crowd Simulation 37
3.2.3 Force-Based Techniques for Character Animation 38

3.2.4 Crowd Simulation in Game Engines . . . 39
3.3 Velocity-based Modeling of Physical Interactions . . . 39
3.3.1 Overview . . . 40
3.3.2 Anticipatory Collision Avoidance . . . 41
3.3.2.1 Agent-agent collision avoidance . . . 41
3.3.2.2 Agent-dynamic obstacle collision avoidance . . . 41
3.3.3 Constraints from Physical Forces . . . 42
3.3.3.1 Force computation . . . 42
3.3.3.2 Force constraints . . . 44
3.3.4 Benefits of Force Constraints . . . 45
3.4 Higher-Level Behavior Modeling . . . 46
3.4.1 The Behavior Finite State Machine . . . 46
3.5 Results . . . 48
3.5.1 Agent-Agent Interaction . . . 49
3.5.2 Agent-object Interaction . . . 50
3.5.3 Dodge-ball scenario . . . 52
3.5.4 Large-Scale Simulation: Tawaf Scenario . . . 54
3.5.4.1 Effect of physical interactions . . . 56
3.6 Analysis . . . 59
3.7 Conclusions and Future Work . . . 62
4 BRVO: Predicting Pedestrian Trajectories using Velocity-Space Reasoning . . . 64
4.1 Introduction . . . 64
4.2 Related Work . . . 66
4.2.1 Motion Models . . . 66
4.2.2 People-Tracking with Motion Models . . . 67
4.2.3 Robot Navigation in Crowds . . . 68
4.3 Background . . . 68

4.3.1 Motion Prediction Methods using Agent-Based Motion Models 69
4.3.2 RVO Multi-Agent Simulation 70
4.4 Bayesian-RVO 72
4.4.1 Problem Definition 72
4.4.2 Model Overview 73
4.4.3 State Representation 76
4.4.4 State Estimation 76
4.4.5 Maximum Likelihood Estimation 77
4.4.6 Implementation 78
4.5 Results and Analysis 80
4.5.1 Method Analysis 82
4.5.1.1 EnKF for Online Learning 82
4.5.1.2 Online vs Offline Learning 82
4.5.1.3 EM based estimation refinement 83
4.5.2 Environmental Variations 85
4.5.2.1 Noisy Data 85
4.5.2.2 Density Dependance 87
4.5.2.3 Sampling Rate Variations 90
4.5.3 Model Performance Comparison 90
4.5.4 Analysis 92
4.6 Robot Navigation with BRVO 92
4.7 Conclusion and Future Work 97
5 Interactive Data-Driven Crowd Simulation using Time Varying Pedestrian Dynamics 98
5.1 Introduction 98
5.2 Related Work 100
5.2.1 Crowd Simulation 101
5.2.2 Data-driven Crowd Simulation 101

5.2.3 Video Based Crowd Analysis 102
5.3 Time Varying Pedestrian Dynamics 103
5.3.1 Pedestrian State 103
5.3.2 Pedestrian Dynamics 104
5.3.3 State Estimation 106
5.3.4 Dynamic Movement Flow Learning 107
5.3.5 Entry-Points Learning 109
5.4 Data-driven Crowd Simulation 110
5.4.1 Pedestrian Dynamics Retrieval 111
5.4.1.1 Adapting to Different Environments and Situations 112
5.5 Results 114
5.5.1 Scenarios 115
5.6 Analysis and Comparisons 119
5.6.1 Comparisons 119
5.6.2 User Study 120
5.7 Conclusions, Limitations and Future Work 121
6 Conclusion and Future Work 123
BIBLIOGRAPHY 126

LIST OF TABLES

2.1 Data-driven stress parameters. Values derived from fitting our stress model to street crossing data from Crompton [1979]. 32

2.2 Performance timings per frame. 32

3.1 Different interaction parameters for intentional behaviors and responsive behaviors . . 48

3.2 Performance on a single core for different scenarios. Our algorithm can handle all of them at interactive rates. 60

4.1 **Comparisons of different Bayesian learning algorithms** Root mean squared error between the predicted position and ground truth position in meters (best shown in bold). 82

5.1 Performance on a single core for different scenarios. We highlight the number of real and virtual pedestrians, the number of static obstacles, the number of frames of extracted trajectories and the time (in seconds) spent in different stages of our algorithm. Our learning and trajectory computation algorithms can be used for interactive crowd simulations. 118

5.2 Comparison of similarity scores (higher is more similar). These preliminary results indicate that the use of our pedestrian dynamics learning algorithm to compute entry points and movement flows considerably improves the perceptual similarity of our simulation to the pedestrian movements in the original video. 122

LIST OF FIGURES

1.1 Components in Crowd Sim (colored outline) as related to techniques (rounded rectangles) used in this thesis. 7

1.2 Opposing Group scenario. (a) Two opposing groups approach each other. (b) Agents initially form natural lanes. (c) After experiencing stress from the alarm the lane formation breaks down into uncooperative, clogged and congested behavior. 9

1.3 Average speed of agents in the Opposing Groups scenario at various levels of stress. The Yerkes-Dodson Law states that stress should increase performance up to a point then decrease it, a result which is matched by the proposed model. 10

1.4 **Interactive Simulation of Physical Interactions between Agents, Furniture in Office, and User-thrown Obstacles**. Agents navigate to avoid office furniture (left). As users insert flying pink boxes into the scene, the agents get pushed, collide into each other, and avoid falling objects (right). 11

1.5 **Train Station scenario:** (a) Original Video. We use the tracklet data computed by the KLT algorithm as an input to our algorithm. We compute the pedestrian dynamics from the noise trajectories. (b) A frame from our data-driven crowd simulation algorithm in the same layout. (c) Crowd simulation in a modified layout as compared to the original video. (d) We increase the number of pedestrians in the scene. They have similar movement patterns as the original video and are combined with density-dependent filters for plausible simulation. 14

2.1 Shibuya Crossing Scenario. A scenario simulating the scramble crossing near Shibuya metro station in Tokyo. (a) Agents start to walk quickly or jog when the walk signal begins flashing indicating little time left to cross. (b) When the light turns red, indicating no time left to cross safely, agents experience a high level of stress and run aggressively to cross as quickly as possible. 15

2.2 **General Adaptation Syndrome and approximation** (a) GAS model of the human response to stress. After an initial disturbance, the resistance level increases up to maximum capacity. If the stress is unresolved, the resistance gets weaker and is no longer effective (causes illness or death). (b) Our approximation of GAS model. We assume acute stress, i.e. no exhaustion stage. Instead, agents are relieved from stress when the stress is resolved. The shape of the response is determined by the stress accumulation rate α and maximum capacity β 19

2.3 **System Overview** Different levels of stressed behaviors are simulated by updating agent parameters. 23

2.4 Evacuation scenario. Agents evacuate an office building in the presence of a fire stressor and crowding stressors. 30

2.5 Chasing scenario. (a) A red shirted agent is chased through a crowd by a green monster causing a positional stressor. (b) If not being chased (dashed line), the agent's path drifts as it navigates through the crowd. With our model, the agent takes a faster, more direct path through the crowd (red line). This aggressive behavior is due to the effect of the stressor. 31

2.6 Comparison of simulated crossing speeds and real-world data. The less time left to cross, the faster agents move. 32

3.1 **Simulation of Tawaf**: We simulate pilgrims performing the Tawaf ritual. In our scene, about 35,000 agents circle around the Kaaba, performing a short prayer at the starting line while some of the agents try to get towards the Black Stone at the eastern corner of Kaaba. We model the interactions between the agents in a dense crowd, such as when the agents are pushed by crowd forces (see video). 34

3.2 **Wall Breaking**. We demonstrate the physical forces applied by cylindrical agents to breakable wall obstacles. Our algorithm can model such interactions between the agents and the obstacles in dense scenarios at interactive rates. 36

3.3 **System Overview.** The motions for objects and agents are computed by a rigid-body dynamics solver and a constrained optimizer, respectively. Physical interactions between agents and obstacles determine forces. For obstacles, the forces serve as inputs to the rigid-body system; for agents, they become force constraints. These force constraints are combined with the original ORCA planning constraints and serve as inputs to optimization algorithm. 39

3.4 **Overview of FSM-based behavior modeling** FSM states are used to specify a set of available actions, along with the parameters used for physics-based interaction and local collision avoidance. Transitions between the FSM states are made based on the result of our physical interaction model combined with local collision avoidance method. 47

3.5 **Rushing through still agents**: The red agent tries to rush through a group of standing agents, simulated (a) with only anticipatory collision avoidance and (b) with physical interactions. Using our method, the forces are propagated among the agents, resulting in a new distribution pattern (b). 49

3.6 **Two bottlenecks scenario** We simulate and compare crowd behavior at two narrow bottlenecks, which are marked with red dotted lines. Bottleneck (1) is barely wide enough for one person to pass through; bottleneck (2) is about twice that width and allows two agents to pass through it at a time. The result from collision-avoidance-only simulation results in an arch-shaped arrangement of agents in the crowd (highlighted with a yellow circle), which causes congestion at the bottleneck. Our method breaks the congestion by allowing the agents to push one other in congested conditions. 50

3.7 **Wall Breaking Simulations with Different Wall Properties**. (a) When the blocks are tightly attached, the wall is not broken. Instead, it is moved and rotated by crowd forces, and made a gap for the crowd to escape through. (b) When the wall consists of much heavier blocks, it does not break or move easily even after all the crowds (1200 agents) entered the isle. In this example, sometimes crowd makes a wave-like movement where sparse density crowd movement is propagated front to back and vice versa. . . . 51

3.8 **FSM for Dodge-ball Scenario** We use a simple two-state FSM to specify the game rule. The states consists of defense state and an attack state. During the attack-state, the character chases a ball and kicks it to its opponent. A character in defense-state tries to avoid the ball until the ball fell down on the ground. State transitions occur based on the location of the ball and the kicking (applying force to the ball) action performed by the character. 52

3.9 **Behavior examples modeled by our method** (a) Green character (user-control, left) kicks the red ball to the orange character (computer control, right). Orange character is in the defense-state at that moment, and tries to avoid the ball. (b) When the ball falls down, the orange character's state is changed to the attack-state. Collision avoidance behavior is changed just to meet non-overlapping condition with the ball and the character's preferred velocity is updated towards the ball. The orange character approaches the ball and kicks the ball to the green character. 53

3.10 **Agent States and Transitions**. The Tawaf states are represented as blue circles and transition condition between these states are marked with arrows. We associate different properties like walking speed, pushing condition, etc., with the agent behavior. 56

3.11 **Density from the Simulated Result**. Reported densities on the Mataf floor can be as high as 8 $people/m^2$ (Curtis et al., 2011); our method gives a maximum density around 7.4 $agents/m^2$ 57

3.12 **Pushed by crowd**. Green circles represent the agents in the queue waiting to touch the black stone. The green agents slow down and the result in heavy congestion at the beginning of the queue. Without physical interactions, agents are stuck in the beginning of the queue although there is a space in front of the queue to proceed. By adding physical interactions, the agents in the queue are pushed by the crowds, and move towards the black stone without breaking the queue. 58

3.13 **Region Speed from the Simulation Result** Average speed of each region of the Mataf area. It matches the overall trend corresponding to higher average speed (region 6) and lower speeds (regions 1 and 6) observed by (Koshak and Fouda, 2008). 59

3.14 **Comparison of average region speeds**. Blue bars correspond the average speeds of the agents in each region when we introduce excessive pushing behavior to the exiting agent and queuing agents. Red bars correspond the average speeds when only the queuing agents pushes forwards while moving towards the Black Stone. Increasing number of pushing behaviors brought a 20% to 40% increase in average speed. 60

3.15 **Number of constraint optimization failures**. We analyze the stability of our method by measuring the number of constraint optimization failures. At the peak of congestion in the simulation, 95% of the agents are able to find velocities that satisfy all their constraints. Importantly, this stability holds across a variety of timesteps. As the timestep size varies from 0.01 up to 0.2, most of the agents are still able to find constraint-satisfying velocities. 61

4.1 (a) **RVO Simulation Overview** shows agent A's position ($\mathbf{p}$), preferred velocity ($\mathbf{v}_{pref}$) and actual velocity ($\mathbf{v}$), along with a neighboring agent B (both represented as circles). If the ORCA collision-avoidance constraints prevent an agent A from taking its preferred velocity, as shown here, the actual velocity will be the closest allowable velocity. Taken together, these elements form an agent's RVO state $\mathbf{x}$. (b) **Agent State Estimation** As new data is observed (blue dot) BRVO refines its estimate of a distribution of likely values of the RVO states (dashed ellipse). These parameters are then used with RVO to predict likely paths (as indicated by arrow). 67

4.2 **Overview of the Adaptive Motion Model.** We estimate current state $\mathbf{x}$ via an iterative process. Given noisy data observed by the sensor, RVO as a motion model, and the error distribution matrix $\mathbf{Q}$, we estimate the current state. The error distribution matrix $\mathbf{Q}$ is recomputed based on the difference between the observed state and the prediction $f(\mathbf{x})$, and is used to refine the current estimation of $\mathbf{x}$. 71

4.3 **Benchmarks used for our experiments** (a) In the Campus dataset, students walk past a fixed camera in uncrowded conditions. (b) In the Bottleneck dataset, multiple cameras track participants walking into a narrow hallway. (c) In the Street dataset, a fixed camera tracks pedestrian motion on a street. (d) In the Students dataset, a fixed camera tracks students' motion in a university. 75

4.4 **Comparison of Error on Campus Dataset.** BRVO produces less error than Constant Velocity and Constant Acceleration predictions. Additionally, BRVO results in less error than using a static version of RVO whose parameters were fitted to the data using an offline global optimization. 84

4.5 **The effect of the EM algorithm.** (a) This figure shows the trajectory of each agent and the estimated error distribution (ellipse) for the first five frames of Campus-3 data. The estimated error distributions gradually reduces as agents interact. For two highlighted agents (orange and blue), their estimated positions are marked with 'X'. (b) The improvement provided by the EM feedback loop for various amounts of noise. As the noise increases, this feedback becomes more important. 84

4.6 **Path Comparisons** These diagrams shows the paths predicted using BRVO, Constant Velocity, and Constant Acceleration. For each case, ground-truth data with 0.05m noise, 0.1m noise and 0.15m are given as input for learning (blue dotted lines in shaded regions). BRVO produces better predictions even with a large amount of noise. 86

4.7 **Mean prediction error (lower is better).** Prediction error after 7 frames (2.8s) on Campus dataset. As compared to the Constant Velocity and Constant Acceleration models, BRVO can better cope with varying amounts of noises in the data. 87

4.8 **Prediction Accuracy (higher is better)** (a-c) Shows prediction accuracy across various accuracy thresholds. The analysis is repeated at three noise levels. For all accuracy thresholds and for all noise levels BRVO produces more accurate predictions than the Constant Velocity or Constant Acceleration models. The advantage is most significant for large amounts of noise in the sensor data, as in (c). 88

4.9 **Error at Various Densities (lower is better).** In high-density scenarios, the agents move relatively slowly, and even simple models such as constant velocity perform about as well as BRVO. However, in low- and medium-density scenarios, where the pedestrians tend to move faster, BRVO provides more accurate motion prediction than other models. KF fails to quickly adjust to rapid speed changes in the low-density region, resulting in larger errors than other methods. In general, BRVO performs consistently well across various densities. 89

4.10 **Error vs Sampling Interval** As the sampling interval increases, the error of Constant Velocity and Constant Acceleration estimations grows much larger than that of KF or BRVO. BRVO results have the lowest error among the four methods. 90

4.11 **Comparison to State-of-the-art Offline Methods** We compare the average error of LIN (linear velocity), LTA, ATTR, KF and BRVO. Our method outperforms LTA and ATTR, with an 18-40% error reduction rate over LIN in all three different scenarios. Significant improvement is made; compare the 4-16% and 7-20% error reduction rate of LTA and ATTR over LIN, respectively. 91

4.12 **Kinematic model of the robot** The robot is modeled as a simple car at position (x, y) and with orientation θ. ϕ is a steering angle and L is a wheelbase. Our robot has the wheelbase $L = 1m$. 94

4.13 **Performance of the robot navigation** We measured the percentage of the trajectories in which the robot reached farther than halfway to the goal position without any collision during the data sequence, using only GVO (blue bars), GVO with KF (red bars), and GVO with BRVO (green bars), both with 15cm sensor noise. In many of these scenarios, using GVO for navigation often caused a robot to stop moving through the crowd to avoid collisions (the freezing robot problem). GVO-KF shows lower performance in very sparse scenario, but outperforms GVO-only method as the number of pedestrians increases. GVO-BRVO algorithm further improves the navigation, especially for more challenging scenarios with more pedestrians. 95

4.14 **Example robot trajectory** navigating through the crowd in **Students** dataset. Blue circles represent current pedestrian positions, red circles are the current position of the robot, and orange dotted lines are the previous positions of the robot. 96

5.1 **Interactive data-driven simulation pipeline:** Our method takes extracted trajectories of real-world pedestrians as input. We use the Bayesian inference technique to estimate the most likely state of each pedestrian. Based on the estimated state, we learn time-varying behavior patterns. These behavior patterns are used as behavior rules for data-driven simulation and can also be combined with other multi-agent simulation algorithms. All these computations can be performed in tens of milliseconds. 102

5.2 **Pedestrian Dynamics Learning:** A one-frame example from the **zara01** dataset: (a) input consists of the manually-annotated trajectories (green) from the video; (b) probabilistic distributions of entry points at one frame, computed using the Gaussian Mixture Model (shown as elliptical regions); and (c) movement flows grouped by the characteristics of pedestrian dynamics, in which each grouping is represented by the same color. 107

5.3 **Manko scenario:** We highlight the benefits of entry point and movement flow learning. (a) A frame from a Manko video, which shows different flows corresponding to lane formation (shown with white arrows); (b) and (c) We compute collision-free trajectories of virtual pedestrians. For (b), we use random entry points for virtual pedestrians and goal positions on the opposite side of the street. White circles highlight the virtual pedestrians who are following the same movement flow as neighboring real pedestrians. For (c), we use TVPD (entry point distribution and movement flow learning) to generate virtual pedestrians' movement. The virtual agent's follow the lane formation, as observed in the original video. 110

5.4 **Marathon Scenario:** We compare the performance of different algorithms used to generate the trajectories of 500 pedestrians in the Marathon scenario: (a) The original video frame with 18 extracted trajectories (white); (b) A multi-agent simulation using five intermediate goal positions along the track; (c) We run the same simulation with optimized parameters using an offline optimization method; and (d) Instead of the intermediate goals and/or optimized parameters, we use only TVPD. Notably, TVPD captures the pedestrian flow in the original video. 111

5.5 (a) A frame from a video of pedestrians in a street with extracted trajectories (shown in red); (b) Our simulation algorithm computes collision-free trajectories of virtual pedestrians (shown in blue) in the 3D virtual environment, which have the same movement flows as extracted trajectories (red) 115

5.6 **ATC Shopping Mall Scenario:** The real pedestrian trajectories are computed using a 3D range sensor. (a) We show the flows highlighted with different colors in the original data. (b) We populate the scenario with 207 virtual pedestrians. The virtual pedestrians exhibit the characteristics observed in the original data. 116

5.7 **Black Friday Shopping Mall scenario:** (a) We use six simple trajectories generated in an empty space. (b) and (c) We create a new layout by adding obstacles interactively, and also increase the number of pedestrians in the scenario. (b) has 271 pedestrians, (c) has the twice higher number of pedestrians generated at the same rate. 117

5.8 **Crossing Explosion Scenario:** (a) Initially, virtual pedestrians follow the TVPD computed from the extracted trajectories. (b) The user places an explosion during the simulation. The pedestrians who perceive the explosion begin running away from it in different directions. 118

LIST OF ABBREVIATIONS

BRVO	Bayesian-RVO
ORCA	Optimal Reciprocal Collision Avoidance
PDL	Pedestrian Dynamics Learning
RVO	Reciprocal Velocity Obstacle

CHAPTER 1
Introduction

1.1 Motivation

The problem of understanding and modeling how a human crowd would behave in different situations arises in many areas. For example, during the planning stages, city, traffic, and evacuation engineers use crowd behavior modeling to predict the flows and usage pattern to evaluate their designs. Many other applications, including virtual reality, computer animation, games, and training, also need the capability to simulate the trajectory of pedestrians as well as the crowd behaviors. These applications require not only a realistic rendering of the virtual environment, but also realistic simulations of virtual crowds that become an essential element for an immersive user experience.

Human crowd behavior is often understood and modeled in two ways - macroscopic view, which looks at overall crowd behavior as a whole, and microscopic view, which looks at individual level behaviors and their interactions (Al-nasur and Kachroo, 2006). In a macroscopic view, human crowds generally exhibit a collective and structured formation. However, when we look at individual behaviors, we see many variations depending on various factors such as the person, time, or environment. Currently, it is very challenging to simulate wide variety of crowd behaviors in virtual environments due to following reasons. First of all, there is no known mathematical model or algorithm that can describe all aspects of crowd behaviors. Psychologists and social scientists have proposed many different hypothesis about human behavior. Biomechanics have studied how physical conditions, including the terrain and obstacles, can affect human movement. It is important to model these biomechanical and psychological variabilities and conditions.

Another issue is related to simulating individual behaviors, when they interact with the rest of the crowd. The behavior of an individual can vary depending on whether they are alone or part of a dense crowd. For example, when multiple people cross a street, they will sometimes take a detour to avoid bumping into others. This means an individual's movement needs to be adjusted when they

are interacting with other pedestrians. In a dense crowd, if a person pushes someone by mistake, the pushing force might be propagated through the crowd and thereby causes a domino effect. In extremely dense crowds, the forces from crowds sometimes become very large and can completely change the trajectory of an agent or make them fall. In these cases, crowd disasters can occur (Still, 2013). As the density of the crowd increases, it is more likely that even small motions can cause physical interactions with neighboring agents. For these reasons, understanding and simulating interaction between agents is necessary to simulate and analyze crowd scenarios.

As well as the problem of modeling the variations and interactions of crowds, designing a plausible crowd scenarios can be another challenging and important problem. In many applications, the goal is to generate trajectories of individual walking humans that match with real human movement and display emerging behaviors observed in real-world crowds. For these applications, there are many important problems to solve – such as a local interaction model for individual microscopic level behaviors, a higher-level behavior model that controls local behaviors and determines a macroscopic shape of crowd, and an interface to design the scenario. The solution for these problems should work well individually and also when combined together.

Last but not least, the simulation should be numerically stable and efficient. We often need to simulate dense, sometimes even massive (thousands to tens of thousands of agents in) crowds. The simulation should provide stable results in such cases, and be efficient enough to run at interactive rates for games, virtual reality, and computer-generated animation. Moreover, stability and efficiency in crowd simulation can also benefit applications in other areas outside computer graphics, such as robot navigation, real-time crowd tracking and scene analysis in computer vision.

1.2 Crowd Simulation

We limit the scope of the crowd simulation discussed in this thesis to mainly computing trajectory-level movements or behaviors of each independent individual, which we call *agent* or *pedestrian*, in the crowd. Since Reynolds introduced simulation of flocking behavior of boids using a few simple rules, many approaches have been proposed to simulate trajectory behaviors (Reynolds, 1987). Continuum-based methods focus more on global or macroscopic view of crowds, by computing common movement characteristics for groups of agents (Hughes, 2003; Narain et al., 2009; Treuille

et al., 2006). These methods are able to simulate a large number of agents, but it is hard to control behavior of individual agents or small groups of agents. Agent-based techniques model interactions between the agents, mainly collision avoidance behavior. Force-based methods (Helbing and Molnar, 1995) use various forces to model attraction and repulsion between the agents; vision-based technique models steering behavior as a response to visual information (Ondřej et al., 2010). Other techniques model collision avoidance behavior using velocity-space reasoning (Karamouzas and Overmars, 2012; van den Berg et al., 2011). Among these techniques, we use the Optimal Reciprocal Collision Avoidance (ORCA) algorithm (van den Berg et al., 2011), which is based on a velocity-space reasoning, to simulate the local navigation of each agent. We use the term *local behavior* to refer to the local navigation behaviors that models interactions with nearest neighbors.

1.2.1 Local Collision Avoidance in Velocity-Space Reasoning

We briefly summarize ORCA algorithm that we use as an underlying multi-agent simulation technique to model individual goals and interactions between people. We chose a velocity-space reasoning technique based on ORCA (van den Berg et al., 2011), which implements an efficient variation of Reciprocal Velocity Obstacles (RVO) (van den Berg et al., 2008a).

ORCA uses a set of linear constraints on an agent's velocities in order to ensure that agents avoid collisions. The constraints are represented as the boundary of a half plane containing the space of feasible, collision-free velocities called *permitted velocities*. Agents are assumed to have a *preferred velocity*, which is the velocity at which the agent would travel if there were no collisions to avoid. At each timestep, an agent computes a new velocity that satisfies the velocity constraints, then updates its position based on the new velocity. See (Curtis and Manocha, 2012) for the various benefits of using velocity-based reasoning for crowd simulation. RVO-based collision avoidance has previously been shown to reproduce important pedestrian behaviors such as lane formation, speed-density dependencies, and variations in human motion styles (Guy et al., 2010, 2011).

1.2.2 Modeling Local Interactions

The agent-based algorithms, including ORCA, focus on computing local collision avoidance behavior of agents. More specifically, these algorithms compute collision-free trajectories of each agent taking into account the interactions with the neighboring agents and obstacles in the environment.

These algorithms provide a general formulation that can be used for every agent in the environment. However, when all the agents exhibit the same behavior patterns, the resulting simulation can look unnatural since it does not capture variations of individual behaviors that we can observe in the real-world. (Peters and Ennis, 2009; Pražák and O'Sullivan, 2011) study the perceptual effect of variation of crowd behaviors and different size of groups in crowd simulation. Lerner et al. point out the restricted number of behavior rules as one of the limitations of crowd simulation algorithms and propose an example-based method to add a variety of behaviors (Lerner et al., 2007).

Some agent-based methods approach this problem by defining different behavior rules to multiple groups of agents. For example, Yeh et al. describe the geometric notion of a composite agent, which can model different behaviors including aggression, social priority, authority, protection, and guidance (Yeh et al., 2008). Pelechano et al. simulate different wayfinding behaviors of trained/untrained leaders and the followers in emergency situations (Nuria Pelechano, 2005). These behavior patterns are chosen based on a given role, and may not change during simulation. More recently, methods based on psychological theories have been proposed. Pelechano et al. define behavior rules for five different types of personalities (Durupinar et al., 2011). Guy et al. use personality trait theory to model heterogeneous crowd behaviors (Guy et al., 2011). Curtis et al. propose an asymmetric collision avoidance method that can handle some limitations from prior symmetric collision avoidance techniques (i.e., where each agent compute collision avoidance behavior in the same way as other agents) (Curtis et al., 2012b).

Simulating dense crowd behaviors is another important problem related to local interaction models. In a dense crowd, agent-agent interactions and agent-object interactions can affect the overall crowd behavior significantly since the frequency of collisions or contacts increases than in a sparse crowd. Dense crowd behaviors are often modeled by continuum methods (Treuille et al., 2006), or hybrid methods (Narain et al., 2009; Golas et al., 2013) becuase of small computational overhead of the algorithm. Modeling high-density crowd behaviors also have been used for safety analysis. Golas et al. propose a continuum-based crowd turbulence model (Golas et al., 2014). Some force-based techniques are also used to simulate the interactions between agents in a dense crowd. Helbing et al. model panic behavior with two additional physical forces (body force and sliding friction) in addition to the social forces (Helbing et al., 2000). Yu and Johansson propose a force-based technique to model the turbulence-like motion of a dense crowd by increasing the

repulsive force (Yu and Johansson, 2007). Curtis et al. use velocity-space collision avoidance technique to model dense crowds performing a religious ritual (Curtis et al., 2011).

1.2.3 High-level Behavior Modeling

Higher-level behavior models provide a way to control local navigation and collision avoidance behaviors of agents. Unlike local interaction behaviors, which can be modeled with mathematical formulations, higher-level behaviors can hardly be formulated due to the complexity of human behaviors. Instead, it has been a common practice to script or manually specify behavior rules. For example, cognitive approaches focus on defining rules or behavior to mimic cognition or the decision making process of the crowd (Shao and Terzopoulos, 2005; Ulicny and Thalmann, 2002). These methods can simulate very specific and detailed behaviors when used with corresponding scenarios such as buying a ticket or waiting in line. Also, Finite State Machine (FSM) has also commonly used to encode procedural behaviors or set of goals based on an agent's state (Bandini et al., 2006; Paris and Donikian, 2009; Sean Curtis, 2013).

On the other hand, data-driven or example-based crowd simulation algorithms synthesize behaviors from real-world behavior examples. Some higher-level behaviors, such as formations or group behaviors, are implicitly exhibited from the examples, rather than explicitly specified behavior rules. Motion capture or real-world trajectory data are commonly used source of examples. For example, trajectory data extracted using semi-automatic trackers are used to generate group behaviors (Lee et al., 2007), and extended to build group formation models (Ju et al., 2010). Populating the virtual scenes can be done by copying and pasting small pieces of synthesized motions (Li et al., 2012; Lee et al., 2006; Kim et al., 2012a; Shum et al., 2012). The emergence of data-driven methods grows out of the increasing availability of real-world datasets of individual humans and crowds, driven in part by improvements in high-resolution cameras and motion-capture systems.

1.2.4 Crowd Simulation for Robotics

Crowd simulation algorithms are not only used for creating graphics animation, but also can be used for understanding and analyzing crowd behaviors in many areas outside computer graphics. Robotics is one area where better understanding of human crowd movement can contribute for applications related to human-robot interaction.

Robots are becoming increasingly common in everyday life. As more robots are introduced into human surroundings, it becomes increasingly important to develop safe and reliable techniques for human-robot interaction. Robots working around humans must be able to successfully navigate to their goal positions in dynamic environments with multiple people moving around them. A robot in a dynamic environment thus needs the ability to sense, track, and predict the position of all people moving in its workspace to navigate complex environments without collisions.

Sensing and tracking the position of moving humans has been studied in robotics and computer vision, e.g. (Luber et al., 2010; Rodriguez et al., 2009; Kratz and Nishino, 2011). These methods often depend upon an a priori motion model fitted for the scenario in question. More recently, local collision avoidance models have been used for pedestrian tracking algorithms (Pellegrini et al., 2009; Yamaguchi et al., 2011). Multi-agent interaction models effectively capture short-term deviations from goal-directed paths. However, in order to do so, they must already know each pedestrian's destination; they often use handpicked destination information, or other heuristics that require prior knowledge about the environment. As a result, these techniques have many limitations: they are unable to account for unknown environments with multiple destinations, or times when pedestrians take long detours or make unexpected stops.

1.3 Thesis Statement

Our thesis statement is as follows:

Velocity space planning can be enhanced by incorporating modeling of human behaviors using psychological, physics-based and pedestrian dynamics learning principles, and these models can each be combined to provide plausible crowd behavior for interactive applications and robotics.

In this thesis, we propose interactive algorithms to simulate crowd behaviors, where a large number of human-like agents interact with each other. Our formulations are developed on top of the velocity-space reasoning method that models local collision avoidance. A large part of our research borrows ideas from other fields including psychology, physics, and machine learning related to understanding and observing human-like behaviors and their interactions. As a result, both short-term, local interaction and long-term, high-level behavior models are improved.

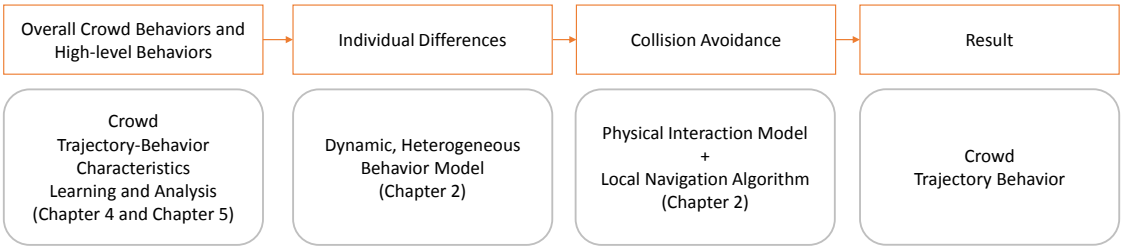


Figure 1.1: Components in Crowd Sim (colored outline) as related to techniques (rounded rectangles) used in this thesis.

1.4 Main Results

Figure 1.1 visualizes how each chapter in this thesis is related to crowd simulation as a building block. As highlighted in the thesis statement, the techniques are based on psychology, physics, and learning principles. These can cover individual differences, local navigation, and high-level behavior learning, respectively.

Our focus is on increasing the realism for simulating the movement and behavior of individual agents and the overall crowd. This includes simulating the interactions and the effects of an agent's movements on other agents in the crowd or within an environment. We first start with local behavior models, which are used to model specific behaviors related to local collision avoidance or local interactions. In Chapter 2, we use well-established studies from psychology to model dynamic behavior changes of a crowd in different situations. In Chapter 3, we introduce a physical interaction model that can simulate new types of force-based interactions as well as collision avoidance behaviors. Next, we suggest algorithms to simulate high-level behaviors which control individual's local behaviors. In Chapter 4, We propose online, adaptive, and individualized motion model that can learn motion model characteristics from real-world data. We are particularly interested in learning from real pedestrian trajectory data and using the learned behavior model for crowd simulation. Finally in Chapter 5, we show a data-driven framework to generate crowd simulation using our learning algorithms. This chapter also discuss about how we integrate the local motion models discussed in the earlier chapters, as well as how the improvement made by the adaptive motion model discussed in Chapter 4. We provide brief summary of each topics below:

Dynamic, Heterogeneous Crowd Behaviors: We present a method to simulate dynamic patterns and variations of individual behaviors based on psychological theories. Our method can handle spatially- and temporally-varying situational factors in a unified framework and can generate several emergent behaviors in different scenarios that match real-world observations.

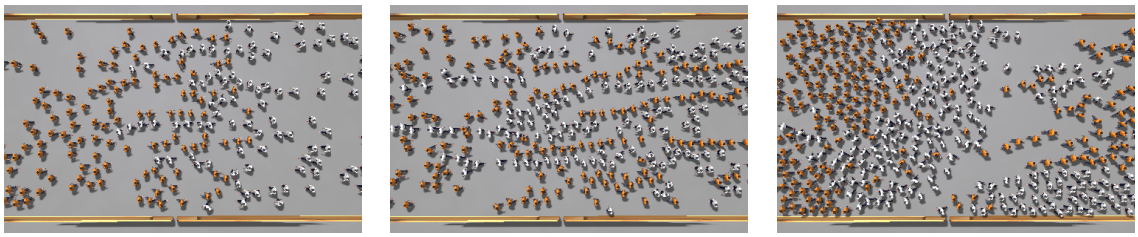
Physical Interaction Models: We present a technique to model various interactions involving physical contacts, such as collision response and intentional physical interactions. Our method handles physical interactions while still preserving the ability of the crowd to avoid collisions with each other. In addition, agent-object interactions and agent-user interactions are performed at interactive rates using our approach. Our formulation can be useful for games, augmented reality or virtual reality applications, where the user can directly interact with other virtual agents or the objects in the scene.

Adaptive Motion Model: We present a method to improve the motion model by learning from real-world trajectory data generated using the sensors. Our approach is able to learn individual characteristics in an online manner, without any prior knowledge to the scene. We use the results to predict the motion of the individuals in the scene, and this can be used to improve the quality of the interactions between the users and the computer system for human-computer interactions, human-robot interactions, computer tracking applications, or augmented reality.

Interactive Data-driven Crowd Simulation: We present a method to learn high-level crowd behaviors from real-world data. Our method retains some of the advantages of data-driven methods that use trajectory patterns from real-world data, but no longer requires building large database or pre-processing of the input data. Also, our method can be used with agent-based simulation techniques without specifying complex behavior rules. Our method can handle both structured and unstructured crowd behaviors, with temporal/spatial behavior changes.

1.4.1 Modeling Dynamic, Heterogeneous Behaviors

Multi-agent simulations typically use a general motion model that applies to all of the agents. However, it is not realistic to see uniform behaviors in a small group of agents for different scenarios.



(a) Initial Approach (b) Lane Formation (c) Alarm Response

Figure 1.2: Opposing Group scenario. (a) Two opposing groups approach each other. (b) Agents initially form natural lanes. (c) After experiencing stress from the alarm the lane formation breaks down into uncooperative, clogged and congested behavior.

Sometimes designers can carefully adjust simulation parameters to model different crowd behaviors that fit to a specific scenario, but the process can be time-consuming.

Our approach is based on observations and understandings from psychology literature. We define a general motion model to describe the variability and the dynamic changes of individual behavior. Our method is based on Attribution theory, that finds a cause of a behavior from a dispositional attributes and situational attributes. Dispositional attributes are internal factors such as personality, and situational attributes are external factors such as stimulus from current situation (Heider, 1982). For example, a very polite, patient person who yields others for a way, but might show aggressive pushing behavior towards the other people in a building with a fire.

As a first step, Guy et al. proposed a mapping between simulation parameters and personality space of the virtual agents, acquired from a user study (Guy et al., 2011). This enables us to simulate agents with different personalities. In Chapter 2, we propose a model that takes account situational stimuli. The agents perceive stimuli from current situation and respond dynamically based on their perceived amount of stress. Our resulting system generates different behaviors of an agent ranging completely based on their personality when they are not exposed to any external stimuli, to a behavior under maximum stress when their accumulated stress reaches threshold.

Fig. 1.2 shows a sequence of images from one of the results. We can see the dynamic changes of crowd behavior from stress-free condition, where the crowd navigates to their destination orderly, to a chaotic behavior as a response to an external stimulus. The wide variety of such dynamic behaviors are modeled by a uniform motion model incorporating personality and situational difference. We also perform a validation of the proposed model in many ways; by providing side-by-side comparison

with other models and reproducing empirical data collected from the real crowd, and to show that the resulting behavior conforms psychology theories, such as Yerkes-Dodson's Law (See Fig. 1.3).

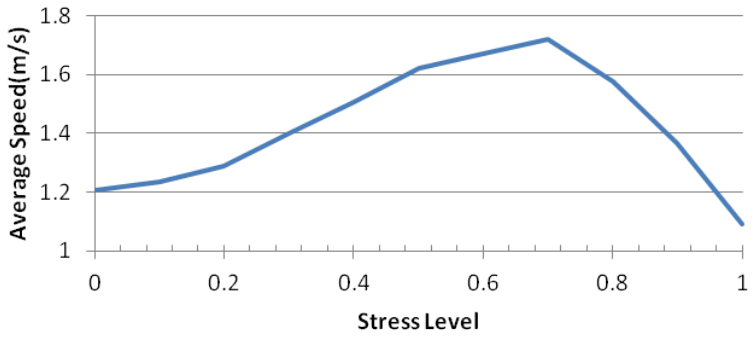


Figure 1.3: Average speed of agents in the Opposing Groups scenario at various levels of stress. The Yerkes-Dodson Law states that stress should increase performance up to a point then decrease it, a result which is matched by the proposed model.

1.4.2 Modeling Physical Interactions

Most of the multi-agent simulations focus mainly on local collision avoidance. These techniques model agent's behavior by defining how they will perceive or acquire knowledge about impending collisions, and how each of them will respond (change their behavior) to avoid impending collision. In other words, it is expected from an agent with a certain level of intelligence that when they see other agents or obstacles on their way to the destination, then they should take a detour for safe navigation. However, very little attention was given to the behavior of an agent when collision actually happens, though such situation is observed frequently in real world, especially in dense settings. We can easily imagine some people who are pushed by somebody in a hurry, for example, in a crowded subway station. Moreover, many other interactions are initiated by contact, that is by physically touching or applying force to a person or an object.

In Chapter 3, we describe a method that models such physical interactions between agents or between agents and obstacles, while preserving the ability of anticipated collision avoidance. We model forces generated from interactions with nearby agents and convert each force into a constraint in the velocity-space. We compute the final velocity of an agent using constrained optimization algorithms using both the force constraint and collision avoidance constraints. We model interaction forces for agents, which can capture human-like agent's behavior different from the rigid-body

motion. For example, we can model balance-recovery motion of the agents, or a propagation of a force or a motion after the interaction through nearby agents in a dense crowd. Moreover, our method is able to handle both collision avoidance and physical interactions with dynamic obstacles (e.g., movable rigid-bodies), as opposed to the previous techniques only considered static obstacles.

Our method provides a numerically stable and fast simulation of a large (a few thousands), dense (up to 7-8 $agents/m^2$) crowd. Since our method is able to handle dynamic obstacles interacting with the agents, and is fast enough for real-time interactive simulation, users can participate in the simulation by moving rigid bodies inside the scene. This movement dynamically changes the environment for the moving agent. Fig. 1.4 shows an example scenario that consists of several pieces of furniture and 1200 agents. In combination with the Bullet Physics library (AMD, 2012), we were able to simulate complex interactions between agents and dynamic obstacles in the environment.

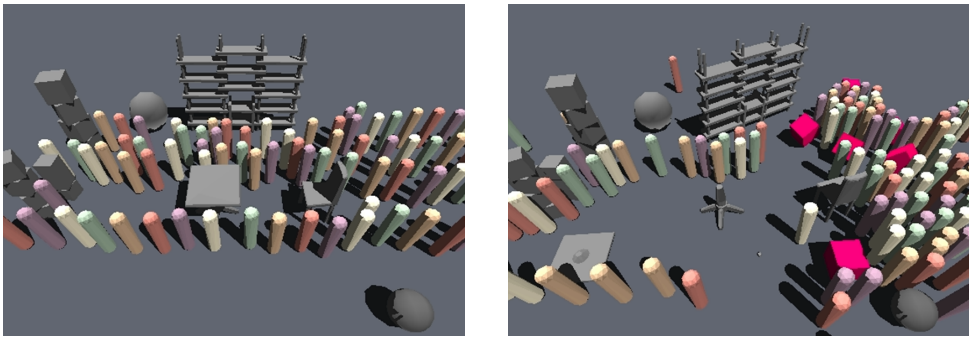


Figure 1.4: **Interactive Simulation of Physical Interactions between Agents, Furniture in Office, and User-thrown Obstacles**. Agents navigate to avoid office furniture (left). As users insert flying pink boxes into the scene, the agents get pushed, collide into each other, and avoid falling objects (right).

We extend the physical interaction model by combining with Finite State Machines (FSM) to model high-level behaviors involving decision making process. In addition, we we have simulated real world examples of massive crowd such as those in the Tawaf ritual. Our method was able to generate many emergent behaviors compared with real-world behaviors.

1.4.3 Adaptive Motion Model: Learning Individual Motion Parameters from Real World Crowd

In Chapter 4, we present a method to fit different crowd motion models as close as possible to give real-world examples. More specifically, the goal is to compute the simulation parameters

for each individual that can best describe the observed motion trajectory. We present an online, individualized and adaptive system that learns behavior parameters and evolves itself as it gets more observations. The fitted motion model can be used to predict the motion of each individual in the near future (a few seconds to a few minutes), and can be used for virtual reality, augmented reality, human-robot interactions, or tracking applications.

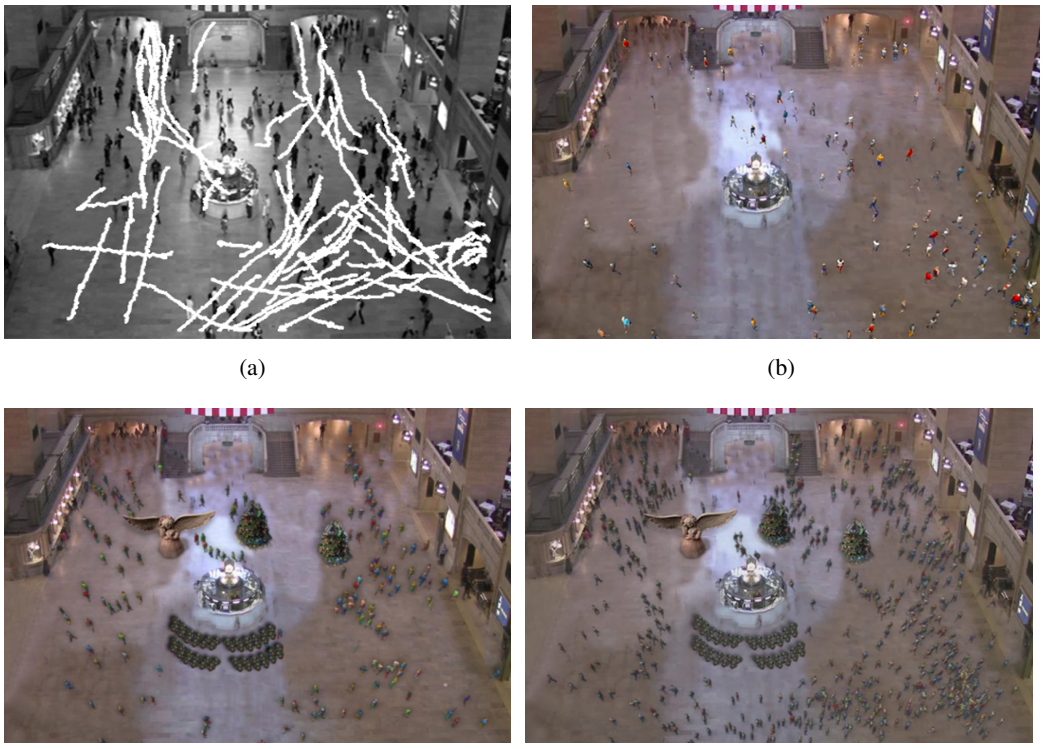
Our approach has unique properties and benefits compared to prior approaches. First, our method can learn unique characteristics of each individuals. Prior techniques find a global, uniform parameters for everybody during the entire sequence of data. Instead, we learn time-varying parameters for each individual, which preserves individual characteristics. Second, our method can generate better representation of time-varying changes of motion by adapting itself to the new observations. We use maximum likelihood estimation technique to improve the accuracy of the motion model. Thus, our method provides better local prediction at a certain time, also resulting in increased overall accuracy for the entire sequence. Finally, our method does not need prior knowledge about the scene and can be used to learn and predict online. Note that prior approaches require manual work or preprocessing to find a global parameters from the same scenario.

We perform series of experiments to demonstrate the performance of the proposed method. The method performs very well with varying number of individuals, varying noise level, scenarios with varying density (low-high), and sparsely sampled data (sampled every 0.4 - 1.6 seconds). Moreover, we extend the prediction method for a robot navigation problem and show the performance improvement.

1.4.4 Time-varying Crowd Behavior Learning for Crowd Simulation

In many applications using crowd simulation, the goal is to generate trajectories of individual walking humans that align with real human movement and display emerging behaviors observed in real-world crowds. The most common approaches for simulating crowd movement use agent-based models that treat individuals as autonomous agents that can perceive the environment and make independent decisions about their behavior or movement. However, using agent-based methods to simulate the dynamic nature, variety, and subtle aspects of real-world crowd motions can be rather challenging.

In Chapter 5, we present an adaptive algorithm that learns high-level behavior rules from real-world human data. The main motivation of our work is to address some of the limitations of existing data-driven crowd simulation and also propose an interactive system for crowd simulation design. Our first goal is develop automatic and interactive approaches that can extract the movement patterns and motion dynamics of individuals from crowd videos. There is a large collection of crowd videos available on the internet and our approach can be combined with tracking algorithms to generate a large library of dynamic movement patterns for different real-world situations. Our second goal is to develop techniques that make it easier to use these movement patterns in slightly different environments or settings than those captured in the original video. Our work has three main applications: replicating real-world crowds in virtual environments, populating the replicated scenarios with virtual crowds, and analyzing time-varying behavior patterns in different scenarios generated interactively by the users. Figure 1.5 shows an example of these applications. From an input video containing real-world crowd behavior (Figure 1.5 (a)), we can create a replicated virtual scenario (Figure 1.5 (b)), and in a different layout designed interactively by the user (Figure 1.5 (c) and (d)).



(a) (b)

(c) (d)

Figure 1.5: **Train Station scenario:** (a) Original Video. We use the tracklet data computed by the KLT algorithm as an input to our algorithm. We compute the pedestrian dynamics from the noise trajectories. (b) A frame from our data-driven crowd simulation algorithm in the same layout. (c) Crowd simulation in a modified layout as compared to the original video. (d) We increase the number of pedestrians in the scene. They have similar movement patterns as the original video and are combined with density-dependent filters for plausible simulation.

CHAPTER 2

Interactive Simulation of Dynamic Crowd Behaviors using General Adaptation Syndrome Theory

2.1 Introduction

Simulating the wide variety of behaviors seen in real-world crowds is important for many interactive applications, including games and virtual environments. There are no known computational models that can simulate different types of crowd behaviors. At a broad level, crowd behaviors are governed by the characteristics of the individual humans and the surrounding environment.

Psychologists have extensively studied human characteristics and behaviors. Differences in human behaviors are governed by multiple factors, including differences in stimuli, genetic endowment, physiological state, cognitive state, social environment, cultural environment, previous life experiences, and personal characteristics (Eysenck, 2002). Despite this diversity, factors affecting human behaviors can be categorized into some basic types.

Attribution theory, for example, divides these causes into dispositional attributes and situational attributes. Dispositional attributes capture internal factors such as personality or characteristics, while situational attributes capture external factors such as current situation (Heider, 1982). Cattell (Cattell, 1952) suggested a similar divide in the causes of behavior termed *personality* and *situational factors*.

(a) Crossing immediately before light change (b) Crossing after red light

Figure 2.1: Shibuya Crossing Scenario. A scenario simulating the scramble crossing near Shibuya metro station in Tokyo. (a) Agents start to walk quickly or jog when the walk signal begins flashing indicating little time left to cross. (b) When the light turns red, indicating no time left to cross safely, agents experience a high level of stress and run aggressively to cross as quickly as possible.

Our proposed technique for simulating dynamic crowd behaviors is based on this dichotomy, and we use separate models for an agent's personality and another one to account for situational factors.

In crowd simulation literature, techniques to model heterogeneous behaviors using personality models have been proposed (Durupinar et al., 2011; Guy et al., 2011). Resulting simulations can successfully generate a variety of behaviors happening in a scene, but may not be able to model *dynamic behaviors*. These dynamic behaviors correspond to changes in individual and crowd behaviors *in response to a situation*. For example, a calm or composed person walking through a pedestrian crossing may become aggressive when the light turns red from green. Similarly, the same person may cut through a crowd when a train approaches the platform at a train station. The dynamic nature of human and crowd behaviors is also observed during fearful or panicked situations, such as fire evacuations, where individuals change behaviors in response to emergencies and alarms.

Our objective is to model such dynamic crowd behaviors at interactive rates. We model these behaviors as a reaction to meet certain demands or cope with the changes in a situation or environment. These situational factors will be referred to as *stressors*, and the effect of these stressors on the agents will be measured as *stress*. Our approach is build on the psychological theory of General Adaptation Syndrome (Selye, 1956) that provides a well-established behavior model of how humans react to stress.

Main Result: We present a new technique to simulate dynamic patterns of crowd behaviors by considering various types of stressors. Our model accounts for both stable, consistent aspects of behaviors, influenced by personality, and the dynamic changes in behavior due to situational factors. Our main contribution is a method that incorporates well-established psychological models of stress into crowd simulation. Our algorithm generates realistic, dynamically-changing crowd behaviors based on a few high-level parameters that model how individuals vary in their response to stress.

We model stressors as functions of space and time that can generate various real-world crowd behaviors. These include prototypes of many common stressors based on time constraints, agent positions, and other environmental factors. These stressors can be used in conjunction with goal-directed, multi-agent simulations to easily generate a rich variety of behaviors. An agent's reaction to a stressor depends on both an agent's personality and the stressor itself. More specifically, we model the measured value of stress that a agent receives from stressors as a function of both (personality-dependent) internal weight and (situation-dependent) external weights.

The resulting simulation shows a variety of dynamic crowd behaviors under stress, such as cutting through the crowds, walking with variable speeds, and breaking lane-formation over time or in different situations. We perform both qualitative and quantitative comparison between our simulation results and real-world observations. Our method has a small computational overhead and can simulate thousands of agents responding to dynamic stressors in real time on a single-core CPU.

The rest of this chapter is organized as follows. Section 2.2 provides an overview of related work both from crowd simulation and psychology. Section 2.3 gives an overview of psychological models and of our approach. Section 2.4 discusses our approach to modeling stress accumulation. Section 2.5 presents the overall dynamic crowd simulation algorithm. We highlight the results in Section 2.6. We direct the readers to the project webpage (`http://gamma.cs.unc.edu/GAScrowd/`) for the videos as well as the related publication (Kim et al., 2012c).

2.2 Related Work

In this section, we give a brief overview of prior work from both psychology and crowd simulation literature. Our discussion here focuses on modeling dynamic behaviors due to stress. For a broader coverage of human psychology, we refer the reader to Eysenck (Eysenck, 2002).

2.2.1 Stress

Several researchers have attempted to characterize how humans respond to stress in terms of both internal, physiological changes and external, behavioral changes. Early attempts to model how human behavior changes in different situations include the work of Cattell (Cattell, 1952), who proposed a mathematical formula to predict human behavior as a function of personality and situation. More recently, Leon (Leon, 2010) have extended this work to the pedestrian behavior, modeling the increased aggression people exhibit when stressed.

Selye (Selye, 1956) proposed a broad framework for understanding how the response to stress changes over time. His General Adaptation Syndrome theory presents a non-specific, non-unique model of stress response to stimuli. Since this work, other studies about stress in a broad aspect have been performed, investigating its relationship to general activities, physiological effects, emotional, behavior, and cognitive performance (Eriksen et al., 1999; Mordkoff, 1964). The connection between

stress and aggression has been particularly well established (Evans, 1984; Anderson, 2001; K.B. and Rasmussen, 1979) and holds across a variety of stressors (Berkowitz, 1990; Miller, 1941).

2.2.2 Crowd Simulation

The area of crowd simulation and multi-agent navigation is an active area of research, with a wide variety of methods and results. We primarily focus on interactive methods for modeling crowd behaviors. For a broader coverage of the field we refer the reader to a recent survey (Pelechano et al., 2008).

2.2.2.1 Interactive Crowd Simulation

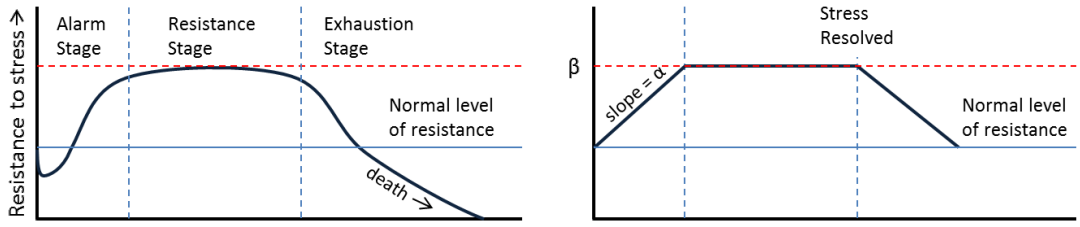
There are several frameworks proposed for simulating and rendering large number of crowds. The Virtual Dublin project simulated crowds in an urban simulation at interactive frame rates (Dobbyn et al., 2005). Yersin et al. (Yersin et al., 2009) proposed a method using pre-computed crowd patches to populate a large-scale virtual environment for real-time simulations. Parallel GPU-based algorithms have also been proposed for both crowd simulation and high-quality rendering (Shopf et al., 2008).

Many real-time techniques have also been suggested for goal-directed multi-agent simulations. Among them the HiDAC system is able to simulate various behaviors (Pelechano et al., 2007). Reciprocal Collision Avoidance based methods have been successfully applied to simulate crowds (van den Berg et al., 2011). Patil et al. (Patil et al., 2011) proposed an interactive algorithm based on navigation fields, where users can directly control the crowd movement.

2.2.2.2 Behavior Modeling

Rule-based approaches are commonly used to model complex behaviors. These include frameworks based on motor, perceptual, behavioral, and cognitive components for modeling pedestrian behavior (Shao and Terzopoulos, 2005) and modeling decision-making process (Yu and Terzopoulos, 2007).

Reynolds (Reynolds, 1999) modeled flocking behaviors as those of individuals and pairs, by combined behavior as group behavior. Yeh et al. (Yeh et al., 2008) described the geometric notion



(a) Selye's General Adaptation Syndrome model

(b) Response generated by our model

Figure 2.2: **General Adaptation Syndrome and approximation** (a) GAS model of the human response to stress. After an initial disturbance, the resistance level increases up to maximum capacity. If the stress is unresolved, the resistance gets weaker and is no longer effective (causes illness or death). (b) Our approximation of GAS model. We assume acute stress, i.e. no exhaustion stage. Instead, agents are relieved from stress when the stress is resolved. The shape of the response is determined by the stress accumulation rate α and maximum capacity β

.

of a composite agent, which can model different behaviors including aggression, social priority, authority, protection, and guidance.

Many researchers have used psychological factors in crowd simulation. Pelechano et al. (Nuria Pelechano, 2005) simulated different wayfinding behaviors of trained/untrained leaders and the followers in emergency situations. These behavior patterns are chosen based on the given role, and may not change during simulation. Sakuma et al. (Sakuma et al., 2005) proposed a local collision avoidance method that switches discreetly between smooth and urgent avoidance behaviors based on the urgency of collisions. Nygren (Nygren, 2007) proposed a system which models the effects of psychological factors by using artist derived rules to change behaviors when the agents are fearful, fatigued or happy.

More recently, there have been attempts to create realistic, heterogeneous crowd behavior based on human psychology, especially personality traits (Guy et al., 2011; Durupinar et al., 2011). These approaches provide a way to model heterogeneous behaviors, but the behavior patterns do not change over time. Our approach builds on these works to model dynamic crowd behaviors.

2.3 Preliminaries and Overview

In this section, we introduce some important concepts from psychology literature on modeling stress and highlight the link between stress and behaviors. We also give an overview of our method for applying this established psychological literature to multi-agent simulations for generating dynamic crowd behaviors.

2.3.1 Psychological Models of Stress

There are multiple definitions of stress in psychology literature. In a broad sense, stress can be defined as any change caused by interactions between the environment and individuals. Generally, stress is caused by a discrepancy between environmental demands and the abilities of individuals (Cox, 1978). In other words, people become stressed when they feel they are challenged or they need to cope with the current situation. Stressors are what cause the stress, they can be a situation, an object, or even other individuals. There are a number of sources that cause stress. In this chapter, we focus on the following types of stressors:

1. loads given to individuals (challenging situations), e.g. time constraints associated with the goal of each agent;
2. perceived threats, e.g. fire, threatening animals or objects;
3. unpleasant events, e.g. heat, noise, air pollution (smoke, malodor), and over-crowding.

The emotional or behavioral effect of stress is generally associated with increased aggression. This link can be found is both psychological models of emotion (Berkowitz, 1990) and empirical studies of human behavior (Evans, 1984; Anderson, 2001) However, the result of stress is not always negative. In some situations increased aggression can have positive effects, and can improve performance up to a point (Yerkes and Dodson, 1908).

By measuring the connection between how people act (measured through recorded observation) and how people feel (measured indirectly via heart rate, skin temperature and self reporting) psychologists have established a consistent relationship between increased stress and increased aggressive and impulsive behavior (Anderson, 2001). This result has held across various stressors, different settings, cultures, and genders (Evans, 1984; K.B. and Rasmussen, 1979). Our approach uses the result of these empirical studies to model various stressors and their effects as increasingly aggressive and impulsive crowd behaviors.

2.3.2 General Adaptation Syndrome (GAS)

To simulate the behavioral effects under a given value of stress, we use the General Adaptation Syndrome (GAS) formulation proposed by Hans Selye (Selye, 1956). Selye proposed the GAS

model as a general response to any stressor (toxins, cold, injury, fatigue, fear, etc.) The GAS model has three stages of response: alarm, resistance, and exhaustion (see Figure 2.2(a)). When individuals perceive a stress, in the alarm stage, they ready themselves for "fight" or "flight". In the resistance stage, they work to resolve the stress at their full capacity. If the stressor is not removed (i.e. a chronic stressor), they reach the exhaustion stage and resistance becomes ineffective.

Research shows that the GAS model also applies to various physiological changes and general activities (Eriksen et al., 1999). Additionally, there is a stable relationship between psychological response of how stress makes a person feel and the physiological response of how it changes their behavior (Mordkoff, 1964).

2.3.3 Approximation of the GAS Model

While the GAS model suggests the shape of a person's stress responses, it does not provide quantitative values for the level of response to different stressors. We propose a quantitative approximation of the GAS model, which produces a stress response consistent with that model.

We first assume an agent is experiencing a perceived stress with a value of ψ. Our goal is to compute a stress response for an agent, denoted as S. This value will be a function of the perceived stress, ψ. To maintain consistency with the shape of the GAS response, our model has two main attributes. First, an agent's change in stress response is capped by a maximum rate, denoted as α. This is to ensure that an agent's stress response does not jump suddenly in response to a sudden stress. Secondly, an agent's stress response is capped at some maximum amount, denoted as β. This is to ensure that if the perceived stress increases unboundedly, there will be a limit on an agent's response.

Taken together, α and β will map the perceived stress ψ to a stress response S as follows:

$$\frac{dS}{dt} = \begin{cases} \alpha & \text{if } \psi > S \\ \{-\alpha \leq \frac{d\psi}{dt} \leq \alpha\} & \text{if } \psi = S \\ -\alpha & \text{if } \psi < S \end{cases} \tag{2.1}$$

where S is capped at maximum of β. For efficiency reasons, we model α and β as constants per agent per stressor, though, in general, these values can be a function of time or of the number of exposures to a stressor.

Figure 2.2(b) shows the resulting stress response induced by a instantaneous, large value of ψ (which lasts until the stress is resolved). This is similar to the stress induced by a sudden, loud warning alarm sounding. The stress response, which results from Eqn.2.1, shows a similar shape to that corresponding the GAS model (shown in Figure 2.2(a)).

In general, an individual's stress response (parameterized by α and β) can vary between different people and across different situations. The values for these parameters can be chosen by an artist for a specific, potentially exaggerated effect (as discussed in Section 2.5.2) or chosen to match real-world data (as discussed in Section 2.5.3).

Our purpose is to provide a general framework that can simulate dynamically changing behavior triggered by stress response in real-time crowd simulations. To that end, we make some simplifying assumptions, notably, treating the parameters α and β as constants, and a further assumption that agents will not be exposed to a stressor long enough to reach the exhaustion or death stage.

2.3.4 Overview of our Approach

Our overall system has three main components. Firstly, the stressor provides a source of stress for the agents. Secondly, the stress accumulation function Eqn.2.1 which is determined by the GAS model. Thirdly, a multi-agent simulation algorithm that is capable of changing an agent behavior by increasing its aggressiveness and impulsiveness.

The interaction between a stressor and an agent's perceived level of stress from that stressor is determined by the *perceived stress function*, ψ. The form of the perceived stress function varies based on the type of stress, and we highlight several examples in Section 2.4. The interactions between the agents are updated as a function of the accumulated stress, S. This aspect is discussed in Section 2.5. Figure 2.3 provides an overview of the system. In the next two sections we discuss details of how we model each component.

2.4 Modeling Stress and Stressors

In this section, we describe how we model stress in our simulator. We show how measurable physical quantities of a stressor are mapped to the perceived amount of stress and give specific examples for various prototype stressors.

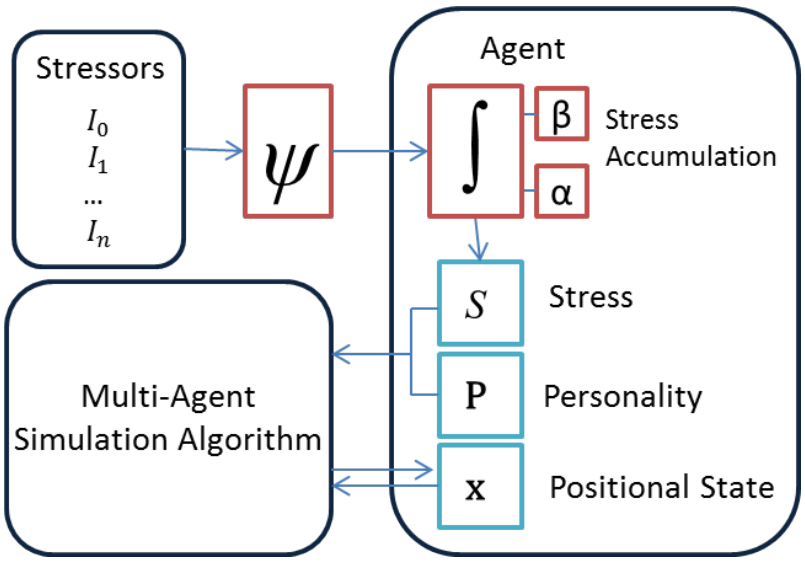


Figure 2.3: **System Overview** Different levels of stressed behaviors are simulated by updating agent parameters.

2.4.1 Perceived Stress

In order to define the perceived amount of stress experienced by an agent, we adapt Stevens' psychophysical power law (Stevens, 1957). This law states the relationship between the perceived magnitude of a stress and the physical measurement of the stimulus intensity, e.g. the relationship between sound intensity and perceived loudness, luminance and perceived brightness, and density and perceived crowding.

Steven's Law states that the relationship between the perceived intensity of the stressor, ψ, and the magnitude of the physical intensity of the stressor, I, has the form:

$$\psi(I) = kI^n, \tag{2.2}$$

with two parameters, a scale factor k and an exponent n, which depend on the type of stressor. Stevens (Stevens, 1957) provides approximate exponent values for various stimulus. Additionally, the computation of the intensity I also depends on the type of the stressor. Steven's Law was originally formulated for low level stressors, such as noise and heat. However, subsequent studies found that similar power laws can be applied to a wide range of stressors (Teghtsoonian and Frost, 1982;

Middlemist et al., 1976; Oswald and Bratfisch, 1969). Inpired by this observation, we use Eqn.2.2 as a generic model of the effect of the stressors.

2.4.2 Stressor Prototypes

We define four different prototype stressors, which can be used to model a variety of forms of stress. These are time pressure, positional stressors, environmental stressors, and interpersonal stressors.

Time pressure: These are stressors associated with an attempt to reach a goal by a particular time. Examples could include crossing the street at a timed light, boarding a train before the doors close, or evacuating a building during a fire.

To model these type of stressors, agents are given a goal position and a time constraint of $t_{allowed}$. We model the intensity of the time pressure, I_t, as a function of the difference between the allowed time and the estimated arrive time. Formally:

$$I_t = \max(t_{estimated} - t_{allowed}, 0), \tag{2.3}$$

where $t_{estimated}$ is the estimated amount of time an agent will take to reach its goal, i.e. $t_{estimated} = distRemaing/avgSpeed$.

Area stressors: These are stressors that arise from a condition in the environment. Examples include noise, heat, bright lights, and smoke. The intensity of these types of stressors is almost constant over a area, that is

$$I_a = \begin{cases} c & \text{if } p_a \in A \\ 0 & \text{if } p_a \notin A \end{cases} \tag{2.4}$$

where A is the area under effect from the stressor and p_a is the agent's current position.

Positional stressors: These are stressors associated with a localized source of stress. Unlike area stressors, these correspond to stressors whose intensity grows as an agent approaches them. Examples include both static stressors, such as fire, or dynamically moving stressors, such as a runaway car or an assailant.

Formally, we define the intensity as

$$I_p = \|p_a - p_s\|, \tag{2.5}$$

where p_a and p_s indicate the position of the agent and the stressor, respectively.

Some stressors, such as fire, have a high intensity over a large area. For these stressors, we use a Gaussian distribution with a standard deviation of σ to compute the intensity:

$$I_p = \mathcal{N}(p_a - p_s, \sigma). \tag{2.6}$$

Interpersonal stressors: These are stressors associated with the stress coming from other agents. A common example includes crowding, where some people feel stress due to too many people being too close. These interpersonal stressors have been found to follow a similar exponential law (Middlemist et al., 1976; Oswald and Bratfisch, 1969). We model these stressors' intensity as a function of the difference between the preferred density of neighbors, and the actual density of neighbors:

$$I_i = \max(n_{cur} - n_{pref}, 0), \tag{2.7}$$

where n_{cur} is number of current neighbors in a unit space and n_{pref} is the preferred number of neighbors in the same area.

2.4.3 Stress Model

Each of the above stressor prototypes define an intensity I which, when combined with Eqn. (2.2), is used to define the perceived stress ψ. Given the current ψ, an agent's stress response, S, is determined by Eqn. (2.1).

The values of α and β can be changed to control the stress response of an agent. The way people respond to stress varies based on the condition of body defense system and the coping style of the individual. In our model, agents with a high α can be thought of as tense and react very quickly to the introduction of a new stressor. Agents with a high β react very intensely to stressors and show a much stronger reaction to the same stressor as compared to agents with a low β.

Multiple stressors: When exposed to multiple stressors, we compute a weighted sum of each stress value to find the total stress experienced by the agent. This model conforms with the discussion in Lazarus (Lazarus, 1993) that people selectively pay attention to the stimulus.

Formally, we define the total perceived stress s as

$$S = \sum \omega_i S_i, \tag{2.8}$$

where ω_i is the weight of each individual stressor S_i. The values of ω_i can be chosen to weigh the stresses equally or can be used to give priority to more important stressors.

2.5 Behavior Mapping

In Section 4, we defined our psychologically-motivated model of how an agent's level of perceived stress, S, changes in response to various stressors. We now discuss how an agent's behavior changes in response to these changes in the stress level.

As discussed in Section 2.3.1, the primary observable response to increasing levels of stress is an increase in aggressive and impulsive behavior. Our method for modeling the dynamic changes in behavior that arise from stress relies on a multi-agent system capable of simulating changes in the levels of aggression and impulsiveness displayed by the agents. We use the multi-agent simulation algorithm proposed by Guy et al. (Guy et al., 2011), though our methods could be easily applied to other approaches with similar capabilities.

2.5.1 Incorporation of Behavior Changes

Recently, Guy et al. (Guy et al., 2011) proposed a method of reparameterizing the reciprocal-collision-avoidance based simulation method described in van den Berg et al. (van den Berg et al., 2011) to achieve a perceptually based variation in agent behavior. The method was based on the results of a user study that asked participants to classify the apparent behavior of agents in terms of how aggressive, impulsive, shy, tense, active or assertive the agents appeared. The result of the study suggested a method for choosing simulation parameters that could achieve a desired agent behavior. These parameters include agent radius, preferred speed, planning horizon, number of neighbors, and

agents sight distance. Different parameter values generate different goal-directed behaviors and local interaction with neighboring agents, which are perceived as personality.

They asserted that there are two primary dimensions (or principle components) of crowd behavior, and that these can be regarded as high-level parameters. The first, denoted PC1, was correlated to an increased level of of "extraverted" or more "intense" behavior from the agents. The second dimension, PC2, was associated with increasingly "careful" behavior. Using this parameterization, we are able to determine how to automatically select simulation parameters for the multi-agent modeling system to produce behaviors that appear to be as increasingly aggressive and impulsive. We denote this change in parameters as the stress behavior vector, $\mathbf{B_{stress}}$ since adding it to an agents current simulation parameters will increase their perceived level of stress. For the results in this chapter, we use

$$\mathbf{B_{stress}} = (\mathrm{PC1}\,\mathrm{PC2}) \begin{pmatrix} 0.95 \\ -0.3 \end{pmatrix} \tag{2.9}$$

because it produces behavior which is predominately aggressive (very high "egocentricity") and somewhat impulsive (negative "carefulness").

2.5.2 Coupling with Personality Attributes

Guy et al. (Guy et al., 2011) also provided a matrix, A_{pc}, which gives a linear mapping between the values of PC1 and PC2 and the simulation parameters. The same work further suggested a matrix, A_{adj} which maps a variety of different personality descriptors to simulation parameters. We determine the final behaviors of our agents as a linear combination of these two parameter matrices: the first representing an agent's situational response and the second the agent's stable personality. The effect of the situational response is scaled by the amount of perceived stress $\bar{s}$ that an agent is current experiencing.

The resulting equation for determining simulation parameters that depicting the agent's behaviors due to inherent personality traits and dynamically changing stress response is

$$SimParams = SA_{pc}\mathbf{B_{stress}} + A_{adj}\mathbf{P}, \tag{2.10}$$

where $\mathbf{P}$ is a vector representing an agents stable personality. When a stress is induced on an agent and the value of S increases, the effect of $\mathbf{B_{stress}}$ will grow and thereby increase the stressful behaviors displayed by an agents. When the stress is removed, S will decrease and the agents behavior will return to its personality $\mathbf{P}$.

2.5.3 Modeling from Real-world Data

Eqn. 2.10 provides a complete description of how stress affects crowd behaviors in a multi-agent simulation. Given the definitions A_{adj}, S, A_{pc}, and $\mathbf{B_{stress}}$ from above, the free parameters to be determined for the model are α and β, which control how an agent responds to stress. One option is to let the simulation designer determine these parameters to create controlled or exaggerated crowd behaviors. However, they can also be found by fitting the stress response to real-word data.

Studies such as Crompton (Crompton, 1979) have found mappings between quantifiable changes behaviors, such as increasing speed, and quantifiable stressors such as the amount of time remaining on a pedestrian crossing signal. Matching the largest observed speeds for agents with little time left to cross provides a lower bound on the value of β. Likewise measuring the change in velocity over the time to cross the street will provide an estimate for α. Finally for determining the form of the perceived stress function $\psi()$ we use a least-square power-law fitting.

Section 2.6.2 shows the result of this process on the data from (Crompton, 1979). In our simulations, we do not use the same value of α and β for every agent, but rather sample them from a normal distribution centered at the values estimated from the data.

2.6 Results

We tested our method on several scenarios described below. For each scenario, we saw a variety of behaviors exhibited by the agents under stress, which matched the results from psychology literature both qualitatively and quantitatively.

Opposing Groups: Two large groups of agents move through a hallway in opposite directions. An alarm is triggered that provides an area stressor for the agents (Figure 1.2).

Street Crossing: A single agent must cross the street with a time pressure given by the crossing signal (see accompanying video)

Chasing: An agent is being chased by a monster. The chased agent experiences a positional stressor from the monster (Figure 2.5(a)).

Evacuation: One thousand agents must evacuate an office environment. All agents experience stress from the fire in the center of the building. Additionally some agents, shown in red shirts, also experience an interpersonal stressor from crowding (Figure 2.4).

Shibuya Crossing: One thousand agents participate in a scrabble crossing modeled after the Shibuya metro station in Tokyo. The agents experience two stressors. The first is a pressure stressor when the crossing signal begins flashing, as a signal to clear the intersection. The second is a more intense area stress, when the crossing signal turns red indicating the vehicular traffic will resume soon (Figure 2.1).

2.6.1 Emergent Behaviors

In the above scenarios agents exhibit several different behaviors commonly associated with increasing stress. These behaviors are not explicitly coded into the system, but rather emerge from the dynamic increase in aggressive and impulsive behavior that occurs from increasing the stress level.

A basic effect that can be seen in all of the scenarios is that agents will increase their speed as their stress increases. An example of this can be observed in the *Evacuation* scenario where agents near the fire run quickly to escape its spread. Likewise in the *Shibuya Crossing* scenario agents who are in the middle of the intersection increase their speed to a jog when the crossing light starting to flash and those who are still in the street when the light turns yellow or red switch to an all out run.

Another stressed behavior displayed by agents is an increase in ”selfish” behavior as stress increases. In the *Evacuation* scenario the agents who experience crowding stress have a higher level of stress than others who are only concerned with evacuating. The effect of this can be seen clearly in the accompanying video where these agents push their way out of the crowded doorway to exit faster than others. Likewise, in the *Chasing* scenario, the agent being chased cuts a near straight line through the crowd. In the same scenario, without a source of stress, the agent's path is deflected by the crowd of people it is trying to cut through (see Figure 2.5(b)).

Additionally, stressed agents show noticeably more impulsive or reckless behavior. In the *Evacuation* scenario stressed agents push and bump into each other as they crowd together to get

Figure 2.4: Evacuation scenario. Agents evacuate an office building in the presence of a fire stressor and crowding stressors.

through the exit. Similarly, agents becoming increasingly stressed leads to the breakdown of the cooperative behavior which usually occurs between agents. This can be most clearly seen in the *Opposing Groups* scenario. Here, after the alarm sounds the natural lane formation that forms between the groups breaks down and the agents degenerate into a chaotic, inefficient flow (see Figure 1.2). When the alarm is turned off, and the stress subsides the agents eventually return to normal flow patterns.

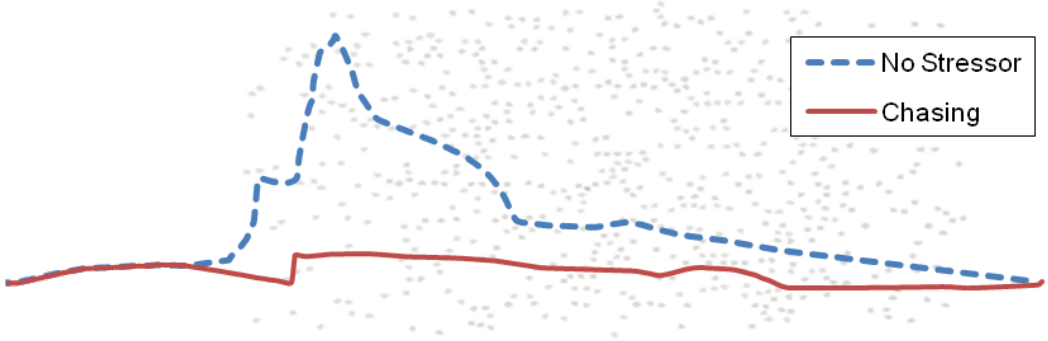
2.6.2 Validation

Due to the dynamic, incoherent nature of human crowds, preforming a precise validation of our method is a challenging problem. However, we can examine some quantitative and qualitative features of stress reported in the psychological literature and compare them to our results.

For example, in a study of pedestrians crossing roads, Crompton (Crompton, 1979) measured average crossing speed against how much a pedestrian has been delayed entering a street crossing from the start of the crossing signal. The study found that the later a pedestrian entered the street, the faster they walked. This effect can be seen with our method by presenting agents with a time pressure stressor with a $t_{allowed}$ for the time it takes to cross the street at an average walking speed. Using the method discussed in Sec. 2.5.3, we derived the values shown in Table 2.1.

Another important result from the psychology of stress is known as the Yerkes-Dodson Law (Yerkes and Dodson, 1908). This is the observation that while for very simple tasks stress is usual

(a)



(b)

Figure 2.5: Chasing scenario. (a) A red shirted agent is chased through a crowd by a green monster causing a positional stressor. (b) If not being chased (dashed line), the agent's path drifts as it navigates through the crowd. With our model, the agent takes a faster, more direct path through the crowd (red line). This aggressive behavior is due to the effect of the stressor.

beneficial to performance, when a task is sufficiently complicated too much stress causes a decrease in performance. We can see similar effects in our simulation. For example, in the *Opposing Groups* scenario, we can change the value of β for the agents, changing the amount of stress they receive from the alarm. Here we measure performance as the average velocity of the agents over a small time window. With small amounts of stress agents velocities increase along with the aggressiveness of their behavior. However, when the stress level becomes too high, the lane formation breaks down causing agents to jam up and bump into each other, and slows down the overall progress. The "inverted U" shaped performance graph in Figure 1.3 is consistent with the Yerkes-Dodson Law.

Param	Value	Description
α	.38	Stress accumulation rate
β	.80	Maximum stress
k	.012	Power law coefficient
n	2	Power law exponent

Table 2.1: Data-driven stress parameters. Values derived from fitting our stress model to street crossing data from Crompton [1979].

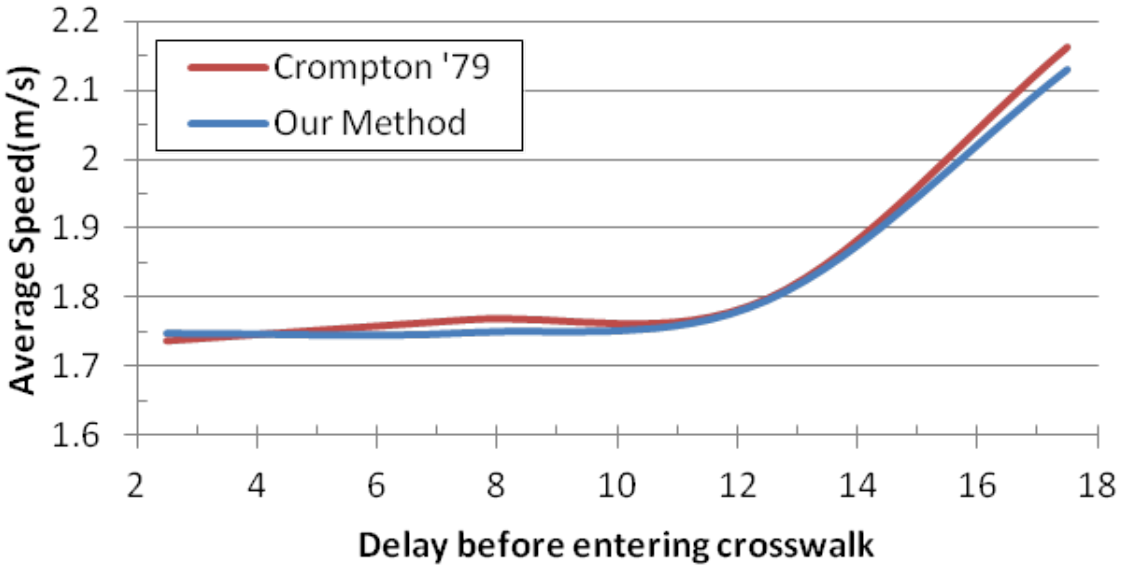


Figure 2.6: Comparison of simulated crossing speeds and real-world data. The less time left to cross, the faster agents move.

2.6.3 Performance Results

The Shibuya Crossing and Evacuation scenarios demonstrate the ability of our method to scale to large complex scenarios with thousands of agents and obstacles. Table 2.2 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 (ms)
Chasing	602	0	18.8
Opposing Groups	800	2	17.2
Evacuation	1000	219	21.8
Shibuya Crossing	1000	205	20.6

Table 2.2: Performance timings per frame.

2.7 Conclusion

We have introduced a method for modeling the dynamic changes in behavior that result from situational factors. Our method derives a linearized approximation of the well-established theory of Generalized Adaptation Syndrome for modeling stress response. Our approach is able to match quantitative studies of human behavior, reproduce important phenomena such as the Yerkes-Dodson Law, and display a variety of emergent dynamic behaviors, all at interactive rates.

There are, however, some limitations to our approach. We have made a few simplifying assumptions, and have limited our application to short-term stressors ignoring the effects exhaustion or death. Additionally, we do not address how the stress response should interact with the crowd rendering system (e.g. changing the facial expression of agents under stress). We would like to investigate these areas in future work. Additionally we would like to integrate our system in a video game engine to allow for stressors to arise from user interaction.

CHAPTER 3

Velocity-Based Modeling of Physical Interactions in Dense Crowds

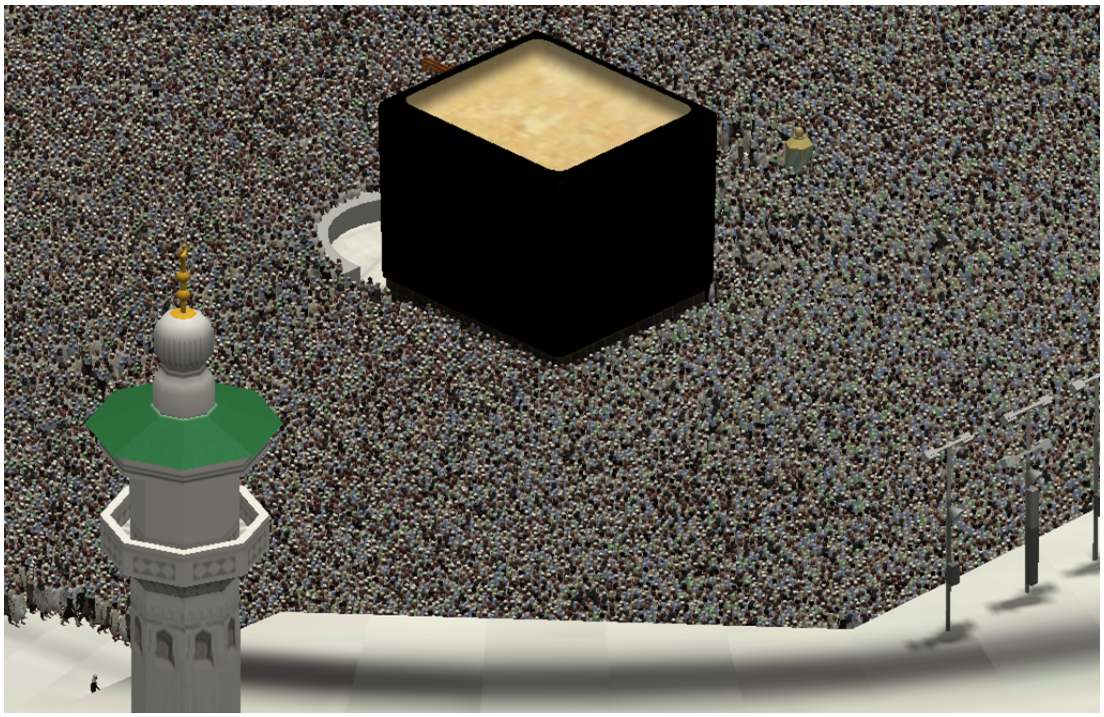


Figure 3.1: **Simulation of Tawaf**: We simulate pilgrims performing the Tawaf ritual. In our scene, about 35,000 agents circle around the Kaaba, performing a short prayer at the starting line while some of the agents try to get towards the Black Stone at the eastern corner of Kaaba. We model the interactions between the agents in a dense crowd, such as when the agents are pushed by crowd forces (see video).

3.1 Introduction

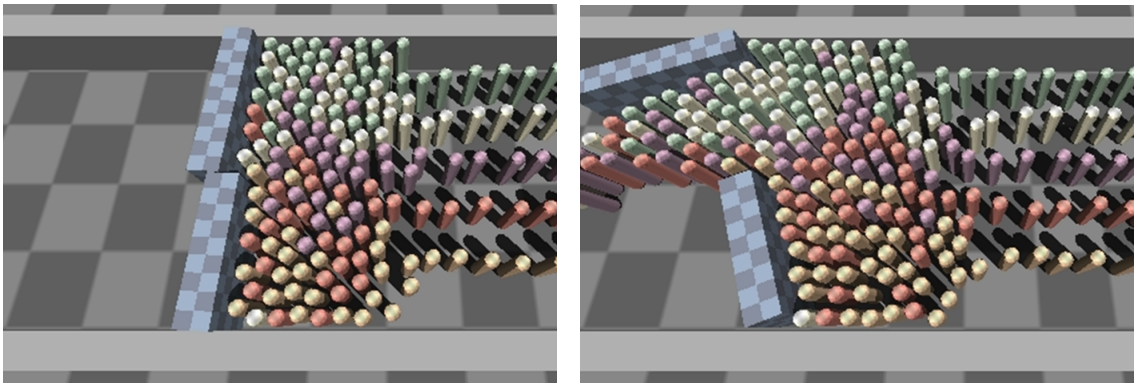
Multi-agent simulations are frequently used to model a wide variety of natural and simulated behaviors, including human crowds, traffic, groups of birds, bees, fish, ants; etc. In many of these applications, it is important for the agents to interact socially and physically with each other and the environment. In Chapter 2, we discussed how we model variety of the behaviors in terms of individual differences and also based on situational factors. In this chapter, we focus on modeling

physical interactions. Agents often collide, push, and impart forces on other agents and on the obstacles in the environment, changing their trajectory or behavior. The challenge is to model these interactions in large multi-agent systems at interactive rates. Many multi-agent simulation techniques focus mainly on local navigation based on anticipatory collision avoidance and does not explicitly take into account physical interactions between agents or between agents and obstacles in the environment. Moreover, collision avoidance behavior towards the obstacles are limited to the static obstacles.

Methods which focus only on collision avoidance can work well in scenarios with low to medium density (e.g. less than 2 $agents/m^2$), where there is enough space for the agents to navigate freely without collisions. However, there are many situations such as political rallies, religious gatherings, or public subway stations, where agents can get very close to each other, and physical interactions between the agents frequently occurs. An agent may be pushed, or bumps into other agents in dense scenarios. For these kinds of dense crowds, it is important to model the direct physical interaction between the agents. Additionally, the indirect effect of the physical impact transferred to neighboring individuals in the crowd, such as the domino effect of people leaning against each other, may impact the trajectory of a high number of agents in a crowd. In extremely dense crowds, the forces from crowds sometimes become very large and can completely change the trajectory of an agent or make them fall. In these cases, crowd disasters can occur (Still, 2013). For all these reasons, understanding and simulating physical interaction between agents is necessary to simulate and analyze dense crowd scenarios. As the density of the crowd increases, it is more likely that even small motions can cause physical interactions with neighboring agents.

Similarly, the forces from many individual agents combine to produce a large effect on the environment. For example, crowds may push stacked boxes while moving through a narrow corridor and somebody may be hit by a falling boxes. Dense, aggressive crowds bend fences or break walls. In order to simulate such scenarios, we need to develop appropriate two-way coupling techniques between autonomous agents and the obstacles in the environment.

Main Results: In this chapter, we present a new method to model physical interactions between agents and objects in an interactive velocity-based multi-agent framework. Our approach incorporates both an agent's ability to anticipate and avoid upcoming collisions, while also modeling physical responses to external forces in a single unified framework.



(a) (b)

Figure 3.2: **Wall Breaking**. We demonstrate the physical forces applied by cylindrical agents to breakable wall obstacles. Our algorithm can model such interactions between the agents and the obstacles in dense scenarios at interactive rates.

We compute the velocity of each agent as a linear programming problem in the velocity space. The resulting approach is efficient and can be used to simulate dense scenarios with thousands of agents at interactive rates. We further extend our method to model more complex behaviors involving social and cultural rules. We use finite state machines to define a series of behaviors as well as parameters for our physical interaction model. These parameters are used to distinguish between responsive collision avoidance behaviors and force-based physical interactions. For example, we use our approach to simulate various behaviors during the Tawaf ritual. We show that our velocity-based formulation can reliably simulate tens of thousands of agents in very dense scenarios (maximum density 8 $agents/m$), and model the physical interactions. Furthermore, we show that our approach is quite robust and we can use large time steps. We have also integrated our approach with the Bullet Physics Engine (AMD, 2012), and highlight the performance in many scenarios.

The rest of this chapter is organized as follows. Section 3.2.1 gives a brief review of related work, Section 3.3 describe velocity-based physical interaction model combining anticipatory collision avoidance and physical forces, and Section 3.4 discuss high-level behavior modeling using a finite-state machine. We highlight the performance on different scenarios focusing on physical interactions in Section 3.5. We direct the readers to the project webpage (`http://gamma.cs.unc.edu/CrowdInteractions/`) for the videos as well as related publications (Kim et al., 2013, 2014b).

3.2 Related Work

3.2.1 Multi-Agent Motion Models

Many approaches have been proposed to simulate the motion of large number of agents and crowds. Often these models are based on rules, which are used to guide the movement of each agent. An early example of such an approach is the seminal work of Reynolds (Reynolds, 1999), which uses simple rules to model flocking behavior.

Force-based methods, such as the social force model (Helbing and Molnar, 1995), use various forces to model attraction and repulsion between agents. These forces are not physically based; rather, they provide a mechanism to model the psychological factors that govern how agents approach each other. Other approaches model collision-avoidance behavior with velocity-based techniques (van den Berg et al., 2011; Pettré et al., 2009; Karamouzas and Overmars, 2012) or vision-based steering approaches (Ondřej et al., 2010).

Other techniques have been proposed to model complex social interaction. HiDAC (Pelechano et al., 2007) uses various rules and social forces to model interactions between agents and obstacles; collision avoidance and physical interactions between agents and objects are handled using repulsive forces. The composite agent formulation (Yeh et al., 2008) uses geometric proxies to model social priority, authority, guidance, and aggression. Many other multi-agent simulation algorithms exist, using techniques inspired by different fields such as sociology (Musse and Thalmann, 1997), biomechanics (Guy et al., 2012a), and psychology (Sakuma et al., 2005; Durupinar et al., 2011; Guy et al., 2011) to model different aspects of agent behaviors and decision models. These approaches are able to generate realistically heterogeneous behaviors for agents. Our approach to model physical interactions can also be combined with many of these approaches.

Other techniques use cognitive and decision-making models to generate human-like behaviors (Shao and Terzopoulos, 2005; Yu and Terzopoulos, 2007; Ulicny and Thalmann, 2002), or use data-driven approaches to the problem (Lee et al., 2007; Lerner et al., 2007).

3.2.2 Dense Crowd Simulation

Density and crowd behaviors are closely related. The fundamental diagram is an empirically measured relationship between the pedestrian density and speed (Seyfried et al., 2005). Some

crowd simulation algorithms tend to adhere to the fundamental diagram. Curtis et al. (Curtis and Manocha, 2012) propose a method to simulate density-dependent behaviors for velocity-based collision avoidance technique. Lemercier et al. (Lemercier et al., 2012) focus on generating realistic following behaviors based on varying densities.

Other approaches for modeling crowds are based on continuum or macroscopic models (Hughes, 2003; Treuille et al., 2006; Narain et al., 2009). In particular, Narain et al. (Narain et al., 2009) present a hybrid technique using continuum and discrete method for aggregate behaviors in large and dense crowds. These continuum methods are mainly used to simulate the macroscopic flow and may not model the detailed interactions between the individuals and the obstacles. In contrast, our approach simulates agent-agent and agent-obstacles physical interaction.

Some force-based techniques are used to simulate the interactions between agents in a dense crowd. Helbing et al. (Helbing et al., 2000) model panic behavior with two additional physical forces (body force and sliding friction) in addition to the social forces. Yu and Johansson (Yu and Johansson, 2007) propose a force-based technique to model the turbulence-like motion of a dense crowd by increasing the repulsive force.

3.2.3 Force-Based Techniques for Character Animation

There has been extensive work on using physics-based models to improve character animation. Sok et al. (Sok et al., 2010) use a force-based approach to ensure that the resulting motions are physically plausible. Other approaches consider geometric and kinematic constraints (Shum et al., 2012) or use interactive methods for character editing (Kim et al., 2009). These techniques, which are primarily based on enhancing motion-captured data, can be used to simulate behaviors of and interactions between the characters and obstacles in their environment.

Many hybrid techniques have been proposed that bridge the gap between physics-based simulation of character motion and pre-recorded animation of characters to model responsive behavior of character (Shapiro et al., 2003; Zordan et al., 2005). Muico et al. (Muico et al., 2011) propose a composite method to improve the responsiveness of physically simulated characters to external disturbances by blending or transitioning multiple locomotion skills.

Our approach is quite different from these methods. Unlike character animation techniques that mainly focus on generating the full-body motion of a relatively small number of characters, we focus on generating physically plausible interactions between a large number of agents in dense scenarios.

3.2.4 Crowd Simulation in Game Engines

Some commercial game engines or middleware products can simulate character motion or crowd behavior. This includes Natural Motion's Euphoria, which simulates realistic character behavior based on biomechanics and physics simulation. There are also commercial AI middlewares for game engines that combine crowd and physics simulation: Kynapse, Havok AI, and Unreal Engine are examples of these. These systems primarily focus on the local and global navigation of each agent using navigation meshes and local rules. Other crowd simulation software such as Miarmy, Massive, and Golaem are integrated with modeling and rendering tools, and used to create character animation. Our approach to generating physical interactions can be combined with these systems to improve local interactions between the agents and the obstacles in the scene.

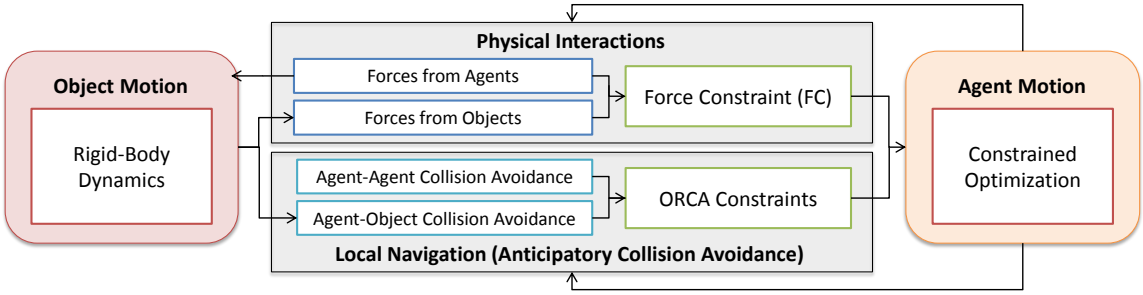


Figure 3.3: **System Overview.** The motions for objects and agents are computed by a rigid-body dynamics solver and a constrained optimizer, respectively. Physical interactions between agents and obstacles determine forces. For obstacles, the forces serve as inputs to the rigid-body system; for agents, they become force constraints. These force constraints are combined with the original ORCA planning constraints and serve as inputs to optimization algorithm.

3.3 Velocity-based Modeling of Physical Interactions

Our approach models the physical interactions between a large number of agents and obstacles. In this section, we give an overview and a summary of techniques used to compute the forces.

3.3.1 Overview

Local navigation and anticipatory collision avoidance of agents can be efficiently modeled using reciprocal velocity obstacles, which imposes linear constraints on an agent's velocity to help it navigate its environment. We extend this framework by representing the effect of physical forces on agents also as linearized velocity constraints. This allows us to use linear programming to compute a new velocity for each agent – one which takes into account both the navigation and force constraints imposed upon that agent. Fig. 3.3 gives an overview of the full simulation system.

Agents are assumed to have a *preferred velocity*. This is the velocity at which the agent would travel if there were no collisions to avoid or physical forces acting on the agent. At each timestep, an agent computes a new velocity that satisfies the velocity constraints, then updates its position based on the preferred velocity. There are two types of constraints which we impose on an agent's velocity:

- **ORCA Constraints** guide the collision avoidance by specifying the space of velocities which are guaranteed to remain collision-free for a given period of time (van den Berg et al., 2011).
- **Force Constraints** account for forces which arise through physical interactions with other agents and objects.

Given an agent A with neighbors B, the *permitted velocities* for A, PV_A is the union of ORCA constraints and force constraints. We can state our agent update algorithm as an optimization problem. Formally:

$$PV_A = FC_A \cap \bigcap_{B \neq A} ORCA_{A|B}, \tag{3.1}$$

$$\mathbf{v}^{new} = \underset{\mathbf{v} \in PV_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}^{pref}\|, \tag{3.2}$$

where $\mathbf{v}^{new}$, $\mathbf{v}^{pref}$ and FC_A are the new velocity, preferred velocity, and force constraints of A, respectively. $ORCA_{A|B}$ is ORCA constraints of A given its neighbors B.

3.3.2 Anticipatory Collision Avoidance

There are some significant differences between an agent's interaction with a neighboring agent and a dynamic obstacle, in terms of the motion computation. The motion of obstacles (e.g. rigid bodies) is governed by Newtonian physics, since these objects have no will and are unable to initiate movement on their own. As a result, the agents cannot assume that the obstacles will anticipate collisions and change trajectory to avoid them. We take account such difference into agent's collision avoidance behavior.

3.3.2.1 Agent-agent collision avoidance

ORCA constraints are defined by a set of velocities that are guaranteed to avoid upcoming collisions with other nearby agents. The constraints are represented as the boundary of a half plane containing the space of feasible, collision-free velocities. Given two agents, A and B, which we represent as 2D discs, we compute the minimum vector $\mathbf{u}$ of the change in relative velocity needed to avoid collision. ORCA enforces this constraint by requiring each agent to change their current velocity by at least 1/2 $\mathbf{u}$. The ORCA constraint on A's velocity induced by B would be:

$$ORCA_{A|B} = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \frac{1}{2}\mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}, \tag{3.3}$$

where $\mathbf{v}_A$ is A's current velocity and $\hat{\mathbf{u}}$ is the normalized vector $\mathbf{u}$.

If A has multiple neighboring agents, each will impose its own ORCA constraint on A's velocity. Local navigation is computed by finding the new velocity for A ($\mathbf{v}^{new}$) which is closest to its preferred velocity ($\mathbf{v}^{pref}$) while respecting all the ORCA constraints.

3.3.2.2 Agent-dynamic obstacle collision avoidance

The dynamic object O is represented, like the agents, as an open disc that is a 2D projection of the bounding sphere of the object. We use this bounding shape for collision avoidance since the agent's navigation is performed in 2D space, but the underlying rigid body simulation uses an 3D object shape for handling collisions with other rigid bodies in the scene.

Agents try to avoid collisions with dynamic obstacles whenever the dynamic obstacles are within agent's visual range. However, agents do not assume objects will reciprocate in avoiding collisions. Therefore, assuming that a change in velocity of $\mathbf{u}$ (Section 3.2.1) is required to avoid an anticipated collision with an obstacle, the collision avoidance constraint for agent A induced by object O is:

$$ORCA^{\tau}_{A|O} = \{\mathbf{v}|(\mathbf{v} - (\mathbf{v}_A + \mathbf{u})) \cdot \hat{\mathbf{u}} \geq 0\}. \tag{3.4}$$

3.3.3 Constraints from Physical Forces

We give brief description of the forces in this section. *Contact forces* include pushing forces and collision forces, and model collision response force or an attempted pushing. *inferred forces* include deceleration force and resistive force, and model the impact of forces on agent's motion.

3.3.3.1 Force computation

Pushing Forces: Pushing is one of the ways for agents to *physically* interact with each other (Pelechano et al., 2007). In our formulation, the pushing force $\mathbf{f}^{p}_{i|k}$ exerted by an agent i pushing another agent k can be given as:

$$\mathbf{f}^{p}_{i|k} = \rho_k f_p \frac{\mathbf{p}_k - \mathbf{p}^{+}_{i}}{\|\mathbf{p}_k - \mathbf{p}^{+}_{i}\|}, \tag{3.5}$$

where $\mathbf{p}_i$ and $\mathbf{p}_k$ indicate the positions of agent i and k, respectively, and $\mathbf{p}^{+}_{i} = \mathbf{p}_i + \mathbf{v}_i \Delta t$ is the pushing agent's future position at the next time step. f_p is a magnitude of total pushing force of agent i towards all interacting agents. It can be defined by the designer, but in our examples, we compute this value to be proportional to agent i's current speed. ρ_k is used to define the weight of pushing force towards each interacting agent k. For our examples, we formulate it as an inverse of number of agents that are pushed.

Collisions: In case of collisions between agents, a collision resolution force is applied. This force is computed based on the physically-based simulation approach proposed by (Baraff, 1997). We consider only linear momentum and simulate agents as radially symmetric disks. For an agent i

colliding with agent k, the collision force $\mathbf{f}^c$ is computed as follows:

$$\mathbf{f}^c = (\frac{-(1+\epsilon)\mathbf{v}^{rel}}{1/m_i+1/m_k} \cdot \mathbf{n})/\Delta t, \tag{3.6}$$

where $\mathbf{n}$ is the collision normal, pointing towards agent i from agent j; v^{rel} is relative velocity; and m_i and m_k are the mass of agent i and agent k, respectively. ϵ is the coefficient of restitution. In our examples, we assign a uniform mass to each agent, but any reasonable mass value can be used for the simulation.

In case of a collision between an agent and a dynamic object, the impulse force is computed in the same way. A force with the same magnitude but with the opposite direction is applied to the object, which also results in change of angular motion generated by the torque τ^c:

$$\tau^c = \mathbf{f}^c \times \mathbf{r}_o, \tag{3.7}$$

where $\mathbf{r}_o$ is the displacement vector for the contact point of the object.

Deceleration Forces: When an agent reduces speed while preserving direction to within a certain threshold (θ_d), we introduce a force into the system based on this velocity change. The deceleration force generated by agent i's deceleration is defined as:

$$\mathbf{f}_i^d = \begin{cases} k_{thresh} m_i \Delta \mathbf{v}_i / \Delta t & \text{if } (\Delta \hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_i) < -cos(\theta_d), \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

where $\Delta \mathbf{v}_i = \mathbf{v}_i - \mathbf{v}_i^-$ is the change in velocity from the previous time step to the current time step. Agents can absorb or transform forces, which are approximated by a parameter k_{thresh}. We assume that the speed reduction arises from one of two sources: self-will (e.g. sudden change of preferred velocity) or agent interaction (e.g. impending collision avoidance). When there is no interacting agent, we assume it is the former case, and the deceleration force is applied back to the agent itself. In the latter case, we distribute the deceleration force among the neighbors. A neighboring agent k causes such behavior if it lies within a cone centered on $\mathbf{v}_i^-$ and is within an angular space of $2\theta_d$ degrees.

Resistive Forces: Resistive forces occur when an agent's computed velocity does not account for the entire change in velocity expected from the external force. This difference is propagated to neighboring agents via the resistive forces. This force is computed by the difference between the velocity $\mathbf{v}$ computed by (3.2) and the velocity $\mathbf{v}^f$ computed only from the net force applied to the agent. The resistive force of an agent i experiencing the discrepancy between $\mathbf{v}^f$ and $\mathbf{v}$ is:

$$\mathbf{f}_i^r = \begin{cases} k_{thresh} m_i (\mathbf{v}_i - \mathbf{v}_i^f)/\Delta t & \text{if } \mathbf{v}_i^f \neq \mathbf{0} \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

As in the case of deceleration force, the resistive force is applied to the agent i when there is no interacting agent. Otherwise, the resistive force is distributed equally among the interacting agents, whose position is inside a cone centered on $\mathbf{v}_i^f$ and with an angular span of $2\theta_r$ degrees.

The resistive force and deceleration force can be viewed as complementary to one another. The resistive force is non-zero only in the presence of external physical forces on an agent, and the deceleration force is non-zero only in the absence of such forces.

3.3.3.2 Force constraints

The net force $\mathbf{f}$ is the sum of all the forces applied to the agent:

$$\mathbf{f} = \sum \mathbf{f}^c + \sum \mathbf{f}^d + \sum \mathbf{f}^r + \sum \mathbf{f}^p. \tag{3.10}$$

The force constraint FC induced by the net force $\mathbf{f}$ is computed as follows:

$$\mathbf{v}^f = \mathbf{v} + \frac{\mathbf{f}}{m}\Delta t \tag{3.11}$$

$$FC = \{\mathbf{v} | (\mathbf{v} - \mathbf{v}^f) \cdot \hat{\mathbf{f}} \geq 0\}. \tag{3.12}$$

FC is a half plane whose boundary, a line through $\mathbf{v}^f$, is perpendicular to the normalized force $\hat{\mathbf{f}}$. It contains a set of velocities that is equal to or greater than the minimum velocity change required by the force $\mathbf{f}$.

3.3.4 Benefits of Force Constraints

By introducing inferred forces, our method can model balance recovery motions that cannot be captured by physics-based rigid body dynamics.

Balance recovery motion: When forces are applied, rigid body motion changes accordingly to the Newtonian dynamics. However, humans have the ability to absorb and resist the external forces even from unexpected events such as sudden pushes or an impact from an obstacle. In these situations, humans take effort to keep their balance creating a behavior known as *balance recovery* in Biomechanics (Maki et al., 2003). Balance recovery is important to model human locomotion, and has been studied in other fields like robotics for humanoid robots (Wei et al., 2009) and in computer graphics for animated characters (Shiratori et al., 2009).

Typical balance recovery motions include taking additional steps or reaching and grasping an object for support. When humans fails to recover the balance, they take further adjustments to refine their initial responses. In other words, the balance recovery can affect the motion, including the trajectory, for a period of time. The balance recovery motion is a result of both physical and cognitive activity, which also depends on the environmental constraints and affordances (e.g., space to step, objects to grasp for support) (Maki et al., 2003).

We define two forces, deceleration force and resistive force that are used to simulate the behavior corresponding to balance recovery. We infer these forces from the agent's motion at a given time, based on our assumption that the motion of an agent can be decomposed into two components: collision avoidance and Newtonian dynamics. Loosely speaking, we treat the different between the velocity implied by the physics forces and the resultant velocity produced by the simulation as a recovery force which is applied on the nearby environment and agents.

Force propagation: Forces applied to an agent can propagate through a dense crowd, since one agent is likely to exert forces on others for support in order to recover from the external pushing force. The propagation forces can be inferred when the motion computed using constrained optimization does not match the motion expected from external physical forces. In this case, we assume that the agent's action of balance recovery took place to resist the external physical forces. For example, when an agent decelerates at a faster rate than that implied by the external forces, we infer that the agent must be pushing against other agents or obstacles in order to be able to slow down so quickly.

Likewise, when an agent accelerates at a rate less than that implied by external forces, we infer the agent must be pushing against other agents or obstacles, while resist the effect of the forces. These inferred propagation forces are applied to the appropriate neighboring agents during the subsequent timestep.

3.4 Higher-Level Behavior Modeling

In many cases, the crowd or individual behaviors change over a period of time. Cultural, social norms, as well as personal goals and intentions can change a person's behaviors over time. Likewise, individuals can exhibit role-specific behaviors, in a variety of situations which can fundamentally change how they interact with each other. A clear example of this can be seen in sporting events, where the behavior towards the players of their own team members are cooperative (and avoid collisions), whereas the behavior towards members of the opposite team often includes blocking, tackling and other forms of physical collisions. Ideally, we would like to model a full variety of such behaviors using our physically-based interactions, while accounting for changes in behavior.

Incorporating such a variety of behaviors changing over time, and depending on the situation or social, cultural norms requires a way to model higher-level decision making process and behavior rules. A common approach to achieve state-dependent behaviors in general is to use Finite State Machines (FSM). In this section, we show how our simulation approach can be combined with FSMs to model such complex physical behaviors in multi-agent simulations. The resulting framework provides a natural way to define different behavior patterns and changes of those behaviors over time and in different situations.

3.4.1 The Behavior Finite State Machine

An FSM is a machine or a model which has finite number of state and transitions between the states. FSMs have been widely used as a way to model intention and decision making process for agents in Crowd Simulation and Games (Thalmann and Musse, 2013; Buckland, 2005). For example, FSMs can describe a set of behaviors for an agent with certain social status (e.g., a leader) along with the transition of these behaviors in certain situations (e.g., safe state or dangerous state) based on the leader agent's perceived information. Recent work has integrated such an FSM-based behavior

specification with velocity-based local collision avoidance schemes to simulate crowds displaying various behaviors (Curtis et al., 2011). We present an improved algorithm that extends this framework to produce complex simulations with physically-based agent interactions.

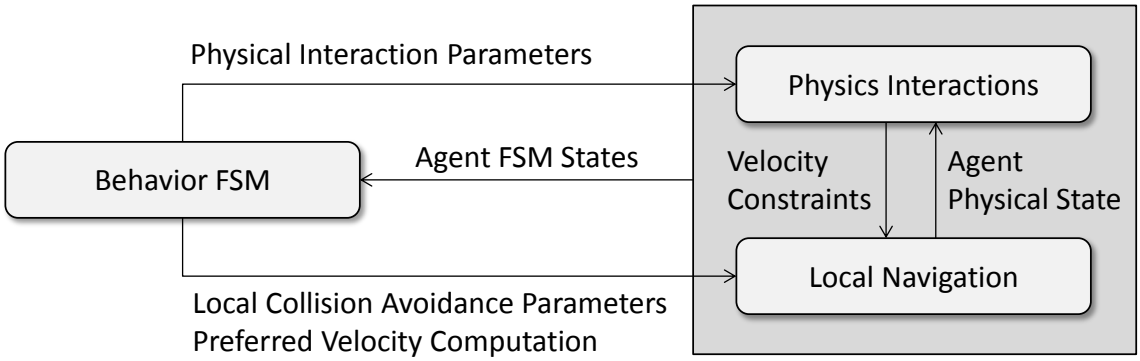


Figure 3.4: **Overview of FSM-based behavior modeling** FSM states are used to specify a set of available actions, along with the parameters used for physics-based interaction and local collision avoidance. Transitions between the FSM states are made based on the result of our physical interaction model combined with local collision avoidance method.

Fig. 3.4 shows the overall architecture of the FSM based behavior modeling for our physical interaction model. We specify a set of actions (behaviors) for each state, along with interaction parameters that change the behavior of the individual agents and the crowd. Transitions between the FSM states are made based on the result of our physical interaction model combined with local collision avoidance method. This corresponds to the decision making process of an agent, which is based on the perceived information about the agent itself and its neighbors. The perceived information and the decision of current action altogether are used to compute agents' local navigation planning. In other words, we model agents behavior using FSM by specifying the interaction parameters that define perception and local planning.

Importantly, it's the combined interplay between an agent's perceived information and its behavioral simulation parameters that determine its actions in a simulation. For example, if we reduce an agent's perception by only allowing it to sense very close neighbors, the agent will be less able to plan ahead to avoid collisions and more likely to run into neighbors. However, when the two agents collide, the magnitude of the interaction will be controlled by the physical interaction parameters. Therefore, a rude or hurried agent state can be created by reducing the perception range, and increasing the pushing forces, and a more polite or relaxed agent state can by made by increasing

the perception range and decreasing pushing. In this way, we use the same motion model (Eq. 3.1 and Eq. 3.2) across all the states of the FSM, but leverage the agent parameters to diversify the agent interactions.

Because of this relationship between collision avoidance parameters and physical interaction parameters, both should be considered together when designing the FSM. Table 3.1 shows what both of these components should be like in terms of intentional behaviors and responsive behaviors. For intentional interactions, the agent's primary goal is to approach the target and apply intended forces. In this case, higher anticipatory collision avoidance behavior prevents the agent from getting closer to the target. Rather, the agent should be able to approach the target even when they perceive impending collision with the target. For example, a soccer player would even run towards the ball even when the ball is approaching the player at a high speed. To model such behavior, local collision avoidance behavior should be minimized to allow the agent to physically interact with the target, but still be able to prevent overlap with the target object.

	Intentional physical interaction	Responsive behaviors
Preferred velocity	towards the target	(local, global) destination
Collision avoidance	no overlap	anticipated collision avoidance
Physical interaction	applying forces (varying magnitude and direction)	collision response or balance recovery

Table 3.1: Different interaction parameters for intentional behaviors and responsive behaviors

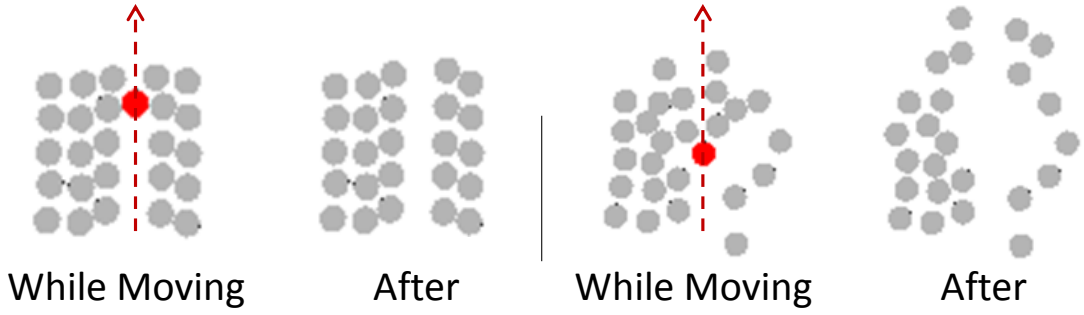
3.5 Results

In this section, we highlight the performance of our algorithm in different scenarios. We first show some results of our physical interaction model, and then present FSM-based behavior modeling with dodge-ball game scenario and Tawaf ritual scenario. We also analyze the approach and compare it with other techniques.

3.5.1 Agent-Agent Interaction

We demonstrate a few scenarios which highlight the effect of physical interactions between agents and how those effects propagate through crowds.

Running Through Scenario: We demonstrate a scenario where an agent runs at a high speed and push through a dense crowd of 25 agents that are standing still. Fig. 3.5 compares the result of our method to those achieved using multi-agent simulation without any physical interactions.



(a) without physical interactions (b) with physical interactions

Figure 3.5: **Rushing through still agents**: The red agent tries to rush through a group of standing agents, simulated (a) with only anticipatory collision avoidance and (b) with physical interactions. Using our method, the forces are propagated among the agents, resulting in a new distribution pattern (b).

The left side of each image shows a pushing agent (red) passing through the crowd, and the right side of each image shows the position of all other agents in the crowd after the fast-moving agent has passed. As Fig. 3.5 demonstrates, agents simulated without physics-based interaction use minimal motion to avoid collisions. In contrast, agents simulated using our physically-based formulation resist the pushing motion (in an attempt to stand still) and propagate the effects of being pushed to other agents.

Two Bottlenecks Scenario: In this scenario, long lines of closely spaced walking agents attempt to pass through two narrow bottlenecks, as illustrated in Fig. 3.6. The first bottleneck (denoted in the figure as (2)) is about the width of two agents; the second is narrower, about wide enough for one agent (denoted as (1)). A local navigation algorithm that performs collision avoidance frequently results in congestion at both the bottlenecks due to stable-arch formation of agents (highlighted with a yellow circle) in Fig. 3.6 (a). However, agents simulated by our physically-based method are able to break this congestion at the bottleneck area by pushing the blocking agents. The ability to break

through bottlenecks also results in a quantitatively higher rate of flow for agents using our approach. After seconds, twice as many agents make it through both the bottlenecks, using our algorithm.

(a) Multi-agent simulation with no physical interaction

(b) Physical interaction amongst agents and with the walls

Figure 3.6: **Two bottlenecks scenario** We simulate and compare crowd behavior at two narrow bottlenecks, which are marked with red dotted lines. Bottleneck (1) is barely wide enough for one person to pass through; bottleneck (2) is about twice that width and allows two agents to pass through it at a time. The result from collision-avoidance-only simulation results in an arch-shaped arrangement of agents in the crowd (highlighted with a yellow circle), which causes congestion at the bottleneck. Our method breaks the congestion by allowing the agents to push one other in congested conditions.

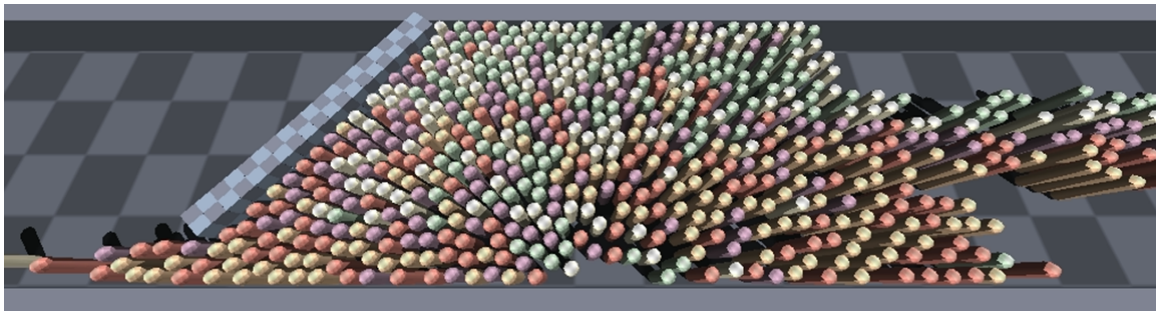
3.5.2 Agent-object Interaction

We also demonstrate the effect of forces between dynamic objects and agents. We used the Bullet Physics engine (AMD, 2012) to compute the motion of dynamic obstacles (3D rigid body dynamics). The results demonstrate several features of our approach:

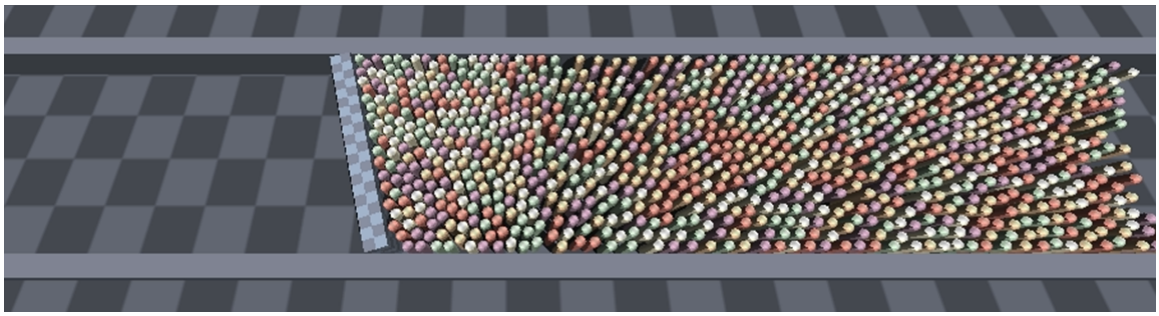
- *Dynamic Obstacle Avoidance:* Agents try to avoid collisions with other agents and with dynamic obstacles.

- *Agent-Object Interactions:* Our method takes into account the collisions which occur between the agents and the objects. The forces generated by these collisions affect both the objects and the agents.

- *User Interactions:* Our method is fast enough for real-time interactive simulation. Users can participate in the simulation by moving rigid bodies inside the scene; this movement dynamically changes the environment for the moving agent.



(a) Tightly attached wall blocks



(b) Tightly attached heavier blocks (zoom out view)

Figure 3.7: **Wall Breaking Simulations with Different Wall Properties**. (a) When the blocks are tightly attached, the wall is not broken. Instead, it is moved and rotated by crowd forces, and made a gap for the crowd to escape through. (b) When the wall consists of much heavier blocks, it does not break or move easily even after all the crowds (1200 agents) entered the isle. In this example, sometimes crowd makes a wave-like movement where sparse density crowd movement is propagated front to back and vice versa.

Wall Breaking Scenario: In this scenario, long lines of agents come at a constant rate into the simulated region, which is blocked off with a movable wall made of 200 blocks glued together. This wall can be broken into separate blocks if a large external force is applied by the agents. Agents initially stop to avoid hitting the wall, but as other agents start to push from behind, the wall breaks apart and gets carried away with the agents. Fig. 3.2 shows stills from the simulation.

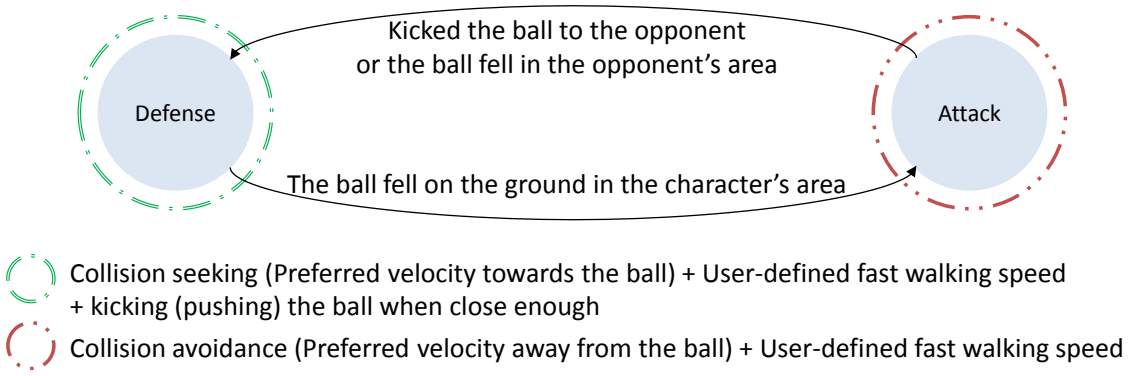


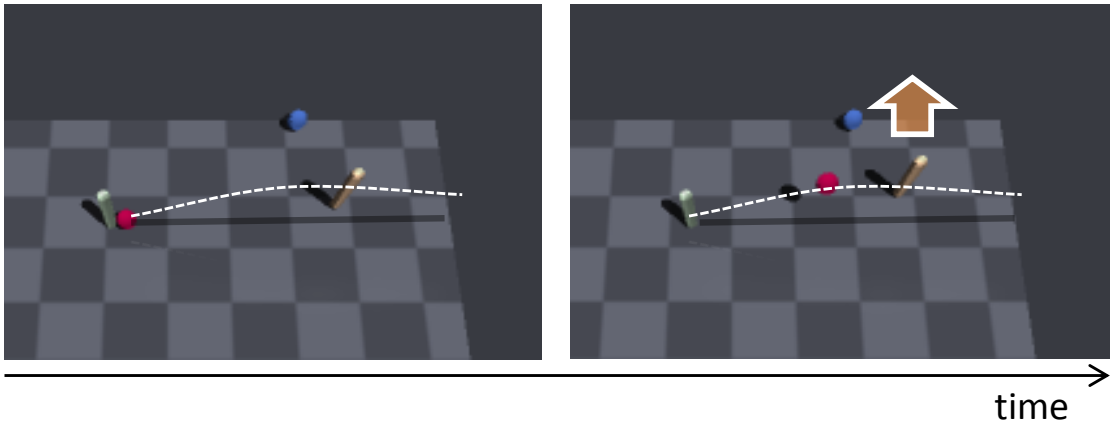
Figure 3.8: **FSM for Dodge-ball Scenario** We use a simple two-state FSM to specify the game rule. The states consists of defense state and an attack state. During the attack-state, the character chases a ball and kicks it to its opponent. A character in defense-state tries to avoid the ball until the ball fell down on the ground. State transitions occur based on the location of the ball and the kicking (applying force to the ball) action performed by the character.

Changing the various properties of the wall changes how the crowd interacts. Fig. 3.7 shows the result of simulation with two different configurations of the wall. In the first configuration, when the blocks are tightly attached, the wall is not broken. Instead, it is moved and rotated by crowd forces, and makes a gap for the crowd to escape through. In the second configuration, when the wall consists of much heavier blocks that are glued together tightly, it does not break or move easily even after the crowd (1200 agents) has entered the isle. In this configuration, the crowd sometimes makes a wave-like movement where the sparse density crowd movement is propagated from front to back and vice versa.

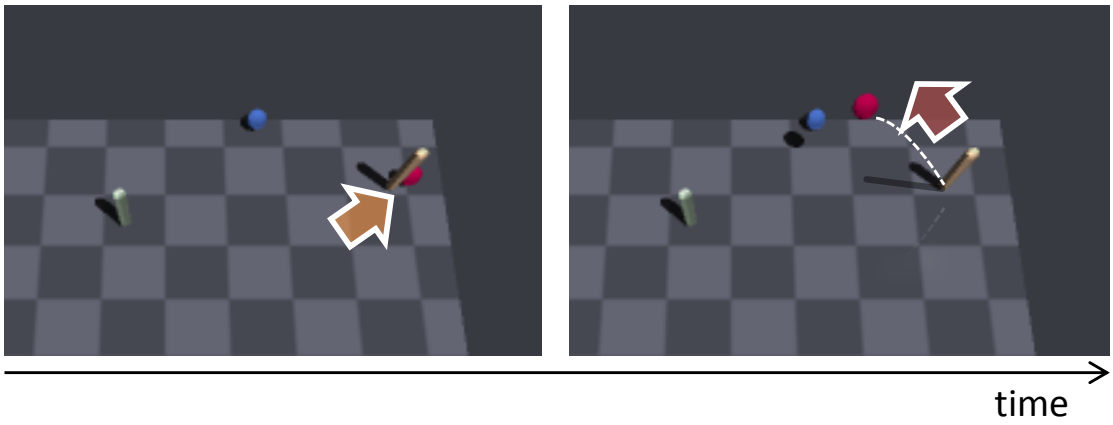
Cluttered Office Scenario: In this scenario, several decomposed 3D models - a table, a chair, and a shelf, and several rigid bodies (e.g. boxes) stacked on top of each other – are placed in the way of the agents. A long stream of agents attempts to navigate past the obstacles. Users can throw boxes, which push the agents and knock over objects in the environment. Fig. 1.4 shows a still from the simulation.

3.5.3 Dodge-ball scenario

As an example of state-based, physical interaction, we show how our FSM based algorithm can be used to control behavior changes in simulated game of dodge ball. Here, we created an interactive dodge-ball game, where a user can control one of the game characters, and the computer program controls the other character(s). A two-state FSM is used to specify the behaviors (See Fig. 3.8), with



(a) Orange character avoids a ball



(b) Orange character approaches and kicks a ball

Figure 3.9: **Behavior examples modeled by our method** (a) Green character (user-control, left) kicks the red ball to the orange character (computer control, right). Orange character is in the defense-state at that moment, and tries to avoid the ball. (b) When the ball falls down, the orange character's state is changed to the attack-state. Collision avoidance behavior is changed just to meet non-overlapping condition with the ball and the character's preferred velocity is updated towards the ball. The orange character approaches the ball and kicks the ball to the green character.

the states consisting of a defense state and an attack state. During the attack state, a character chases a ball and kicks it to its opponent. A character in defense-state tries to avoid the ball until the ball rolls on the ground. State transitions occur based on the location of the ball and the kicking action performed by the characters.

Figure 3.9 shows part of the scenario highlighting the change in interaction between the agent and the ball. The first two images, Fig. 3.9(a), show the agent behavior in the defense-state. The user-controlled green character on the left kicks the red ball at the FSM-controlled orange character on the right. Initially the red character tries hard to avoid the ball, with the local collision avoidance algorithm for this character considering a large perception radius, with a long time duration, when computing its motion. In cases where the agent is not able to avoid the collision (e.g. the ball is moving too fast), there is a physically simulated collision response between the ball and the agent. Due to relatively smaller mass (0.4kg) of the ball compared to the characters (70kg), the effect of collision and resulting forces is much larger on the ball. If the FSM-agent is successful in avoiding the collision, its state will change to the attack state (Fig. 3.9). Here, the agent's collision avoidance behavior is changed allowing it to approach the ball as fast as possible (e.g., small sight radius and large preferred velocity), then to kick the ball towards the green character. The force applied to the ball is computed based on the speed of the character, and its direction is towards the user-controlled agent.

As can be seen in this example, our overall approach can model collision avoidance, collision response, applying intentional forces, and decision making for the character (e.g., goal position and transition of the state). In the supplementary video, we show an expanded version of this scenario with increased number of balls. We can observe a character chasing a ball while avoiding other dynamic obstacles, and, at times, pushing through them to attack the other character. Properly simulating these interacting behaviors requires physical interaction, local collision avoidance, and behavioral states to be combined together in the same framework as we have presented here.

3.5.4 Large-Scale Simulation: Tawaf Scenario

Because our method has only a small computational cost per-agent, and is stable even in dense scenarios, it can be used to produce complex, large-scale simulations with agents physically

interacting across a variety of different behaviors. To illustrate this, we performed a case-study in simulating the large, dense crowd performing the Islamic ritual of pilgrimage called the Tawaf.

During the Tawaf, pilgrims walk in a circle around the Kaaba, the large central structure, seven times counter-clockwise for prayers. While circulating, many pilgrims try as part of the ritual to reach the central black stone located at the eastern corner of the Kaaba; alternatively, they perform a short prayer while facing the Kaaba at the beginning of each circuit. The walkable area surrounding the Kaaba, is known as the Mataf and can support upwards of 35,000 pilgrims gather to perform the Tawaf (Curtis et al., 2011).

We have extended the simulation of Curtis et al. (Curtis et al., 2011), to produce a simulation of the Tawaf ritual with about 35,000 FSM-driven, physically-interacting agents as described in Section 3.4.

The FSM we use, both sets the goals of the agents in order to follow the steps of the ritual and modifies the agent's behavioral parameters to help achieve these goals. For example, some of the agents will probabilistically choose to move closer to the central Kaaba structure in order to approach a religiously significant black stone. Agent in the "Move to Black Stone" state are allowed to exert physical pushing forces on their nearby neighbors in order to successfully move through the dense crowd to reach the stone. Figure 3.10 shows all the FSM states and transition conditions, along with the descriptions about a few important variables conditions for the local navigation and physical interactions for the corresponding state. Most notably, agents exert pushing forces on the crowd if they are trying to touch to the black stone on the Kaaba or are trying to exit the Mataf after completing the ritual.

Flow Analysis: We measured the average speed and density of the agents from our simulation. First, we computed the average speed in different regions of Mataf shown in (Curtis et al., 2011); there is an overall trend towards higher speed in region 6 and towards a lower speed in regions 1 and 7 when compared to the speeds of their neighboring regions (see Fig. 3.13). These highest speed and lowest speed regions also match with the real world data provided by (Koshak and Fouda, 2008). Second, we computed the density in different regions of Mataf based on our Tawaf simulator. Empirically, the density on the Mataf floor can be as high as 8 $people/m^2$ (Curtis et al., 2011); our method gives a maximum density around 7.4 $agents/m^2$ (see Fig.3.11).

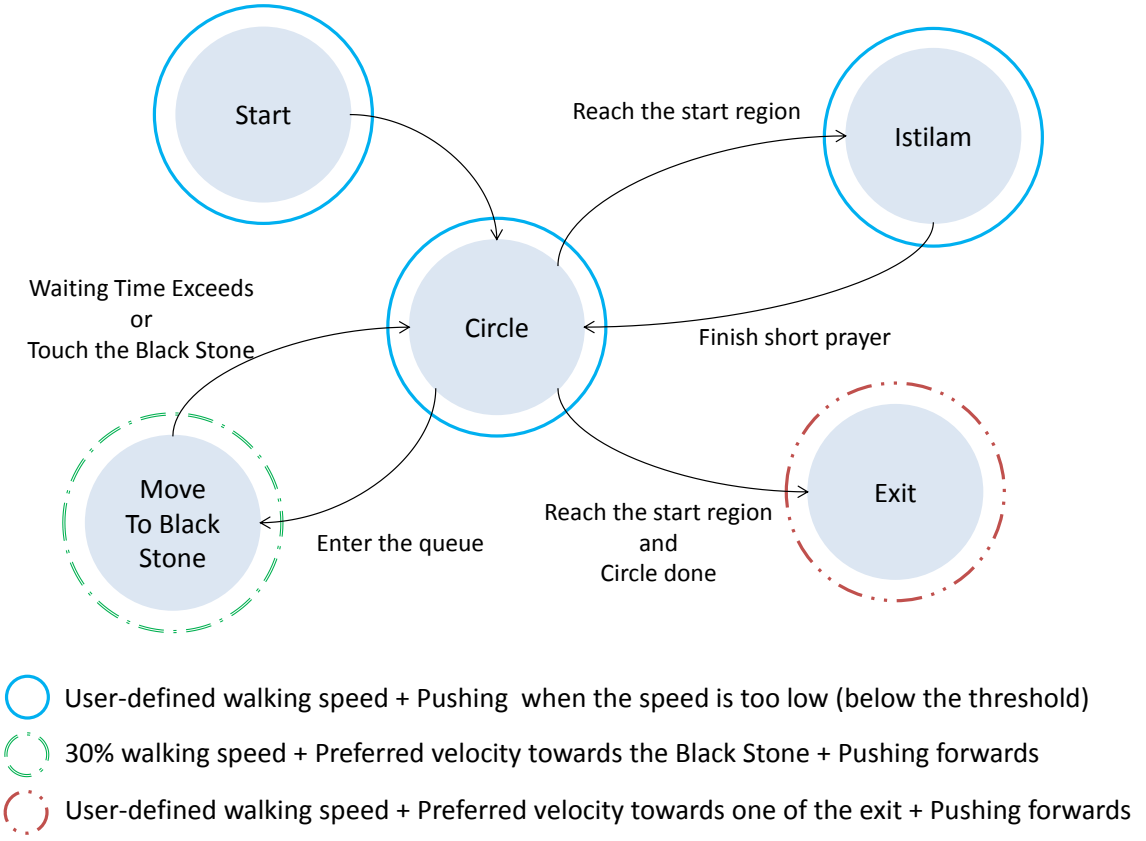


Figure 3.10: **Agent States and Transitions**. The Tawaf states are represented as blue circles and transition condition between these states are marked with arrows. We associate different properties like walking speed, pushing condition, etc., with the agent behavior.

3.5.4.1 Effect of physical interactions

We perform two experiments to show the benefit of physical interactions in large, dense crowd settings. The first experiment shows crowd forces acting on agents. The second experiment compares the overall crowd flow simulated during the Tawaf ritual under increased pushing behaviors between agents.

Pushing in the Queue to the Black Stone : As part of the Tawaf, we simulate the movement of pilgrims waiting in lines to touch or kiss the Black Stone (the eastern cornerstone of Kaaba). Pilgrims in this region makes distinctively slow motion patterns compared to the other pilgrims circling around the Kaaba. After they touch the Black Stone, these pilgrims join the rest of circling flow and adjust to the speed of other neighbors.

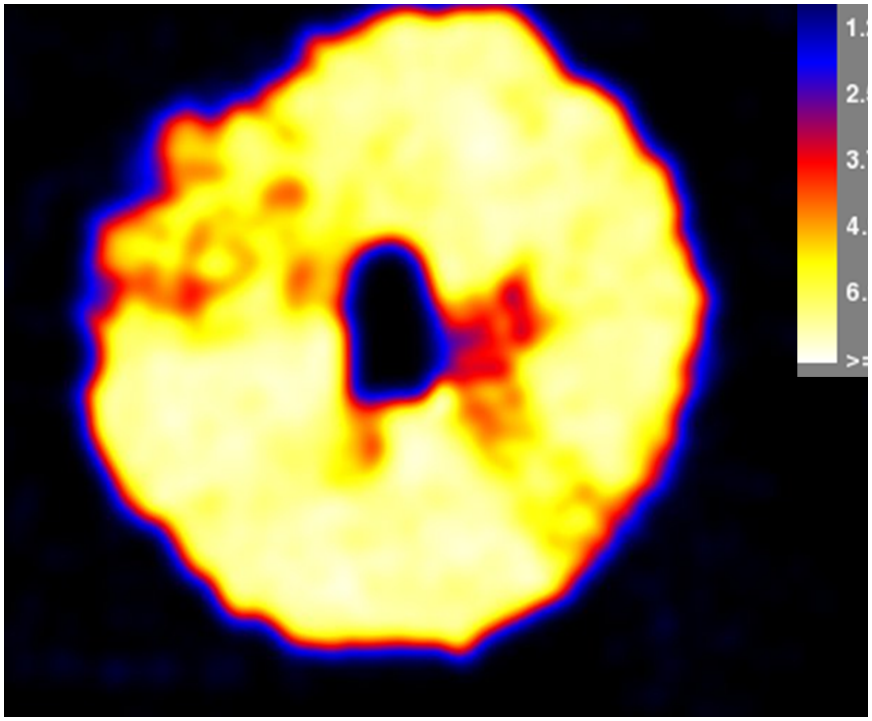
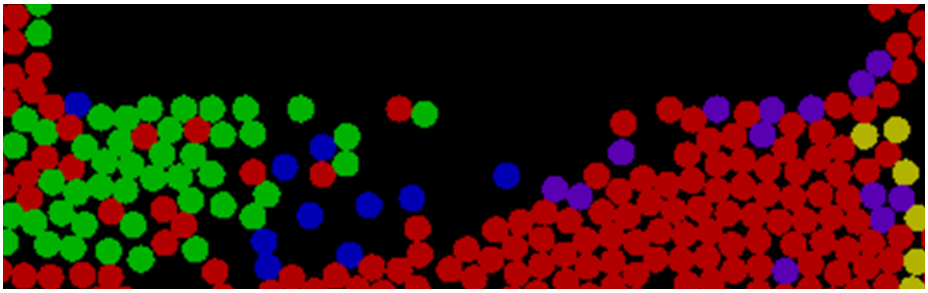


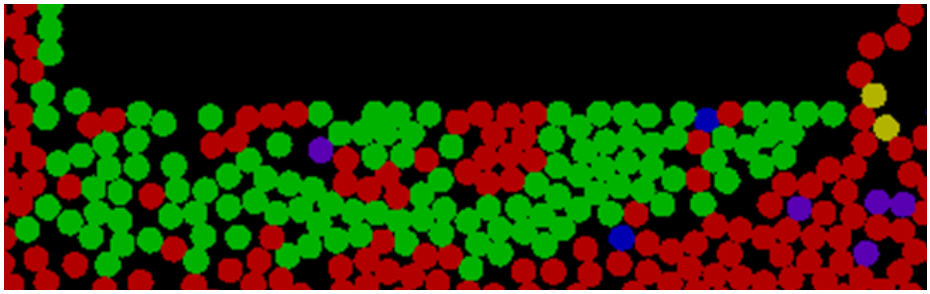
Figure 3.11: **Density from the Simulated Result**. Reported densities on the Mataf floor can be as high as 8 $people/m^2$ (Curtis et al., 2011); our method gives a maximum density around 7.4 $agents/m^2$

In the real-world video (see the supplementary video), we can observe that some of the pilgrims in the queue are often pushed by neighboring pilgrims. We attempted to simulate such crowd force that are applied to the agents in a dense crowd. To see the effect of crowd force, we assign lower preferred speeds to the agents who have entered the beginning of the queue. Figure 3.12 shows the 2D comparison between the simulated queuing behavior for the Black Stone, (a) when no physical interactions added and (b) with physical interactions. Green circles represent the agents in the queue, red circles represent the agents circling around the Kaaba, blue circles represent the agents leaving the queue and start circling after given waiting time.

Due to the sudden slowdown caused by the green agents, heavy congestion is made at the beginning of the queue. Without adding physical interactions, we cannot capture the effect of crowd force applied to these agents even in such high density. Agents are stuck in the beginning of the queue although there is a space in front of the queue to proceed. By adding physical interactions, the agents in the queue are pushed by the crowds, and move towards the black stone without breaking the queue.



(a) Without Physical Interaction



(b) With Physical Interaction

Figure 3.12: **Pushed by crowd**. Green circles represent the agents in the queue waiting to touch the black stone. The green agents slow down and the result in heavy congestion at the beginning of the queue. Without physical interactions, agents are stuck in the beginning of the queue although there is a space in front of the queue to proceed. By adding physical interactions, the agents in the queue are pushed by the crowds, and move towards the black stone without breaking the queue.

Pushing Towards Exits : We also evaluate the effect of physical interactions in large crowds. First, we compare the average speed of the agents with and without physical interactions. Fig. 3.13 shows the average speed of the agents measured in the several different regions as proposed in (Koshak and Fouda, 2008). The overall trend of relative speeds between regions is the same with and without pushing, but the average speed increase as agents push more. In both cases, the trends match well with those reported in the Tawaf literature (Koshak and Fouda, 2008).

Additionally, we run the same scenario with increased number of aggressive pushing agents. When an agent finishes the ritual, the agent is assigned a randomly selected exit (from the five exits in the Mataf area) as their goal position, and tries to push through the crowd. Since the exiting agents have to escape through a very dense crowd while also moving in the circular flow, their pushing forces affects the average speed of entire region. At any given point in the simulation, about 2% of the total number of agents are trying to exit. By adding more pushing agents, the average speed increased about $0.2m/s$. Figure 3.14 summarizes the speed of the agents in each of the Tawaf regions.

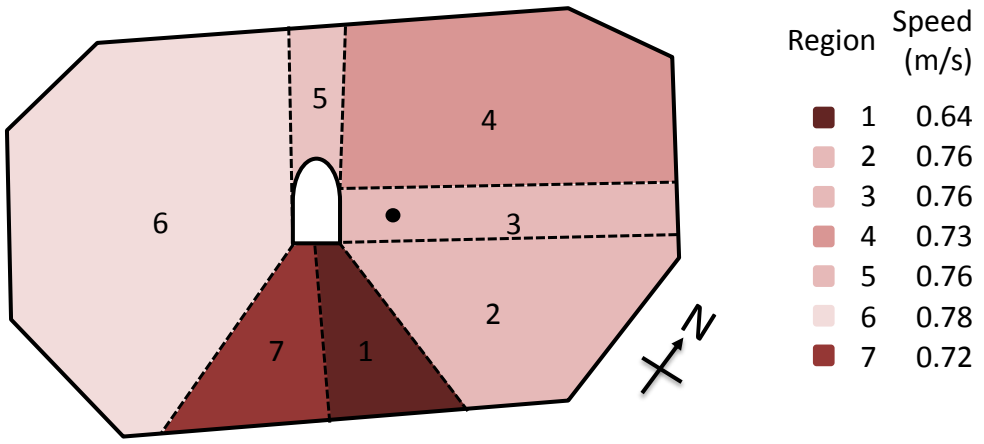


Figure 3.13: **Region Speed from the Simulation Result** Average speed of each region of the Mataf area. It matches the overall trend corresponding to higher average speed (region 6) and lower speeds (regions 1 and 6) observed by (Koshak and Fouda, 2008).

3.6 Analysis

Our approach is mainly designed for interactive applications that require plausible physical behavior (e.g. games or virtual worlds) as well as real-world scenarios with high crowd densities. By using a combination of force and navigation constraints that affect agents' behavior, our approach can simulate many useful effects and emergent behaviors. For example, our formulation allows for intentionally uncooperative agents to physically push their way through a crowd by imparting physical forces to nearby agents. Additionally, agents can use navigation constraints to avoid collisions with dynamic obstacles as well as other agents. By expressing all interactions as linear velocity constraints, we can naturally combine the two different simulation paradigms of forces and navigation into a unified framework and compute the new velocity for each agent using linear programming.

Performance: We measured the simulation timings for the demos we presented in earlier sections (see Table 3.2). The timings were computed on a 3.4 GHz Intel i7 processor with 8GB RAM. Our method efficiently simulates large numbers of agents, and also exhibits interactive performance when integrated with the Bullet Physics library.

Stability Analysis: It is well known that many forced based simulation models, such as social force models commonly used to simulate crowd (e.g. (Helbing and Molnar, 1995)) are prone to stability problems that can even occur at small step sizes (Köster et al., 2013). These problems

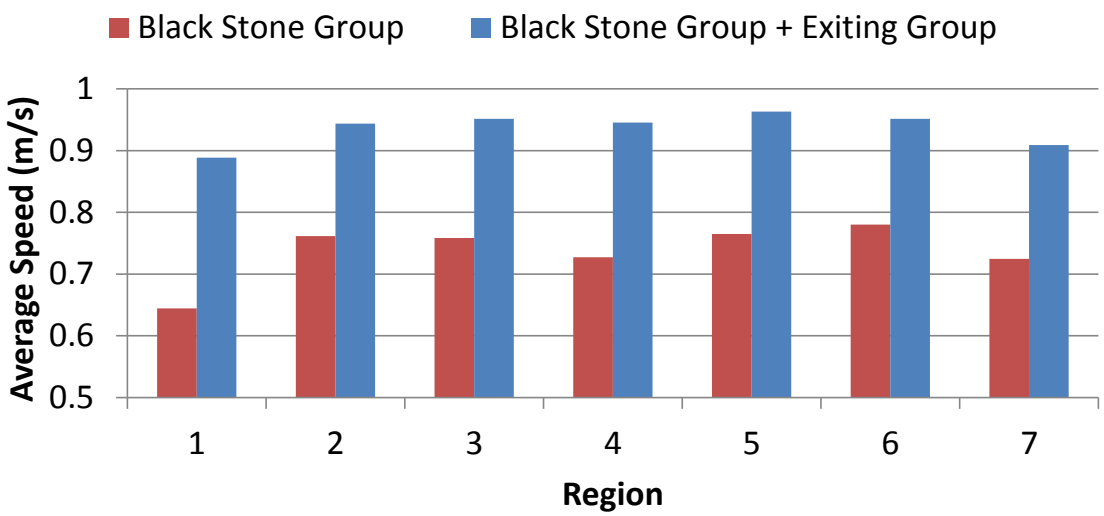


Figure 3.14: **Comparison of average region speeds**. Blue bars correspond the average speeds of the agents in each region when we introduce excessive pushing behavior to the exiting agent and queuing agents. Red bars correspond the average speeds when only the queuing agents pushes forwards while moving towards the Black Stone. Increasing number of pushing behaviors brought a 20% to 40% increase in average speed.

Scenario	Num. Agents	Dynam. Obsts.	Static Obsts.	fps
Two Bottlenecks	1000	0	20	829.7
Wall Breaking	1200	200	2	50.1
Office	1200	65	0	69.0
Dodge Ball	2	500	4	90.9
Tawaf Sim.	35000	0	23	5.7

Table 3.2: Performance on a single core for different scenarios. Our algorithm can handle all of them at interactive rates.

include oscillation and loss of accuracy in terms of trajectory computation. Using bigger time steps can make the problem worse. In contrast, velocity based collision avoidance techniques have been shown to produce stable simulations in large, dense crowds (Curtis et al., 2012a). Our approach preserves this stability across timesteps while still accounting for physical forces.

One way to analyze the stability of our approach is by analyzing the number of times we are unable to find a feasible velocity that satisfies all the constraints, e.g. both anticipated collision avoidance constraints and force constraints. When this occurs, it means an agent can choose a potentially colliding velocity or is not respecting the physical constraints.

We perform a test using a scenario similar to Fig. 3.5. An aggressive agent pushes through 50 standing agents, and the pushed agents sequentially exert forces on adjacent neighbors by generating physical interactions. During the simulation, we measure the number of times when linear optimization fails to find the feasible velocity satisfying all the constraints. In this case, we perform higher-order optimization to find a solution.

As can be seen in Fig. 3.15, at the peak of congestion in the simulation 95% of the agents are able to find velocities that satisfy all their constraints. Importantly, this stability holds across a variety of timesteps. As the timestep size varies from 0.01 up to 0.2, most of the agents are still able to find constraint-satisfying velocities. In general, the behavior does not change significantly across this wide range of time steps.

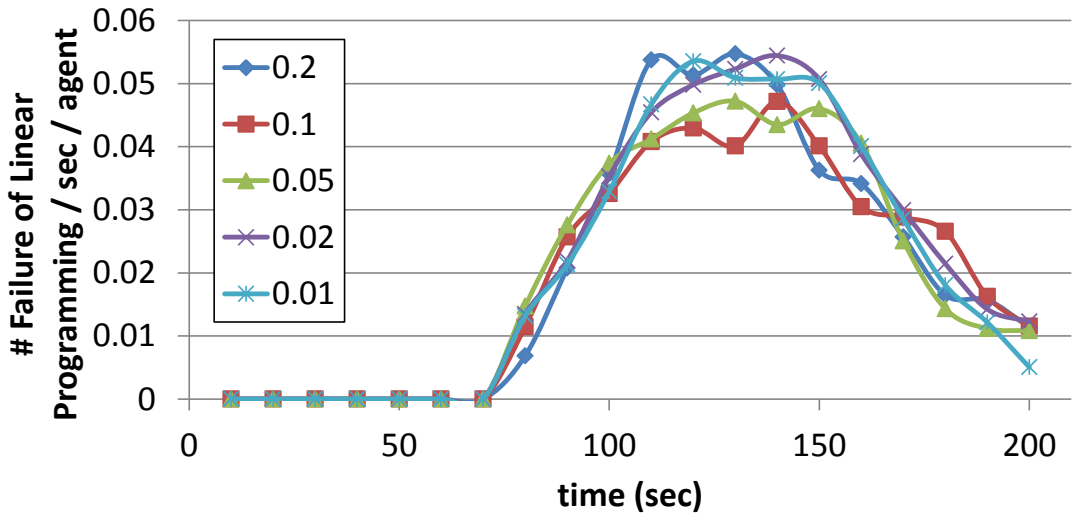


Figure 3.15: **Number of constraint optimization failures**. We analyze the stability of our method by measuring the number of constraint optimization failures. At the peak of congestion in the simulation, 95% of the agents are able to find velocities that satisfy all their constraints. Importantly, this stability holds across a variety of timesteps. As the timestep size varies from 0.01 up to 0.2, most of the agents are still able to find constraint-satisfying velocities.

Benefits of Our Method:

Many techniques have been proposed in the literature for simulating large numbers of agents that display a wide variety of emergent behaviors. However, the primary emphasis of these methods is on collision avoidance – avoiding any physical contact between the agents. In other words, they model

how agents move around each other, but do not usually model explicit physical contacts, interactions, and external forces.

Force-based methods such as (Helbing and Molnar, 1995) use forces to model social factors (e.g. attraction and repulsion) between the agents, not physical interactions. Most closely related to our work are methods such as (Helbing et al., 2000; Yu and Johansson, 2007; Pelechano et al., 2007). These methods model crowd turbulence or physical interactions among panicking agents by adding explicit physical force or by increasing repulsive forces. These methods are capable of reproducing some important emergent crowd phenomena, but do not account for the anticipation needed to efficiently avoid upcoming collisions with other agents and obstacles.

Force-based methods can also suffer from stability issues in dense scenarios, which require careful tuning and small time steps in order to remain stable (Curtis et al., 2011; Köster et al., 2013). Our method provides stable anticipatory motion for agents while incorporating agent responses to forces. It can be easily combined with other velocity-based approaches.

In terms of large, dense crowd simulation, continuum-based methods such as (Hughes, 2003; Treuille et al., 2006) or hybrid method coupling continuum-based method and velocity-based method (Narain et al., 2009) can be effective solutions. However, these methods do not model physical interactions between the agents or obstacles. Moreover, it is hard to extend these methods to model individually varying behaviors or high-level social behaviors.

Limitations: We use a physically-inspired approach to simulate the interactions between a high number of agents and the obstacles. However, it is only an approximation and may not be physically accurate. Secondly, we assume that agents are constrained to move along a 2D plane, and we use the projected positions of 3D dynamic objects to compute the interactions. Third, like other agent-based simulation methods, we use a rather simple approximation for each agent (a 2D disc). This means that we cannot accurately simulate physical interactions with human-like articulated models and 3D objects.

3.7 Conclusions and Future Work

We have proposed a novel method to combine physics-based interactions with anticipatory collision-avoidance techniques that use velocity-based formulation. Our method can generate many

emergent behaviors, physically-based collision responses, and propagation of forces to the agent's nearby neighbors. In combination with the Bullet Physics library, we were able to simulate complex interactions between agents and dynamic obstacles in the environment. We also showed that our approach can be extended to model more complex behaviors involving a decision making process. In addition, we simulated real world examples of massive crowds such as in the Tawaf ritual. Our method was able to generate many emergent behaviors compared with real-world behaviors.

As future work, we would like to further explore our method by comparing the results with real-world crowd behaviors and performing more validation. Moreover, in many scenarios, the external forces could change an agents behavior. For example, applying intentional forces such as pushing, kicking can slow down an agent. Such a phenomena could be well incorporated in our approach given studies about how such forces can limit human behavior (e.g., from biomechanics). Furthermore, we need better techniques to collect data about real-world crowds in dense settings and use them to validate the simulation algorithms. We would also like to extend our model to agents moving in 3D space or multi-layer frameworks, and to consider using more complex shapes, or even articulated body models, to represent agents, as this would allow for more accurate force computation. Finally, we would like to use more accurate physically-based modeling algorithms to generate appropriate behaviors.

CHAPTER 4

BRVO: Predicting Pedestrian Trajectories using Velocity-Space Reasoning

4.1 Introduction

In Chapter 2 and Chapter 3, we discussed about modeling local navigation behaviors to generate trajectories of virtual agents. In this chapter, we discuss about how we can use such local navigation model to learn and predict real human trajectory behaviors. More specifically, we focus on using our methods for robot navigation problem in crowds.

Robots are becoming increasingly common in everyday life. As more robots are introduced into human surroundings, it becomes increasingly important to develop safe and reliable techniques for human-robot interaction. Robots working around humans must be able to successfully navigate to their goal positions in dynamic environments with multiple people moving around them. A robot in a dynamic environment thus needs the ability to sense, track, and predict the position of all people moving in its workspace to navigate complicated environments without collisions.

Sensing and tracking the position of moving humans has been studied in robotics and computer vision, e.g. (Luber et al., 2010; Rodriguez et al., 2009; Kratz and Nishino, 2011). These methods often depend upon an a priori motion model fitted for the scenario in question. However, such motion priors are typically based on globally-optimized parameters for all the trajectories during the entire sequence of data, rather than only taking into account the specific individual's movements. As a result, current motion models are not able to accurately capture or predict the trajectory of each pedestrian. For example, we frequently observe atypical pedestrian motions, such as moving against the flow of other agents in a crowd, or quick velocity changes to avoid collisions. In order to address these issues, many pedestrian tracking algorithms use multi-agent or crowd motion models based on local collision avoidance (Pellegrini et al., 2009; Yamaguchi et al., 2011). These multi-agent interaction models effectively capture short-term deviations from goal-directed paths, but in order to do so, they must already know each pedestrian's destination; they often use handpicked destination

information, or other heuristics that require prior knowledge about the environment. As a result, these techniques have many limitations: they are unable to account for unknown environments with multiple destinations, or times when pedestrians take long detours or make unexpected stops. In general, the assumption that the destination information remains constant can often result in large errors in predicted trajectories. Moreover, environmental factors, such as static or dynamic obstacles, dense crowds, and social/cultural behavior rules, make predicting the behavior or movement of each individual pedestrian more complicated, thereby making it more difficult to compute a motion from a single destination.

In this work, we seek to overcome these limitations by presenting a novel online prediction method (BRVO) that is built on agent-based, velocity-space reasoning combined with Bayesian statistical inference; BRVO can provide an individualized motion model for each agent in a robot's environment. Our approach models each pedestrian's motion with their own unique characteristics, while also taking into account interactions with other pedestrians. This can be contrasted with the previous methods that use general motion priors or simple motion models. Moreover, our formulation is capable of dynamically adjusting the motion model for each individual in the presence of sensor noise and model uncertainty. We address the problems associated with fixed destination by integrating learning into our predictive framework and by adjusting short-term steering information.

Our approach works as follows. We generalize the problem by assuming that at any given time the robot has past observations for each person in the environment and wants to predict agent motion in the next several timesteps (i.e., to aid in navigation or planning tasks). We apply Ensemble Kalman Filtering (EnKF) to estimate the parameters for a human motion model based on *Reciprocal Velocity Obstacles* (RVO) (van den Berg et al., 2008a). We use this combination of filtering and parameter estimation in crowded environments to infer the most likely state for each observed person: its position, velocity, and goal velocity. Based on the estimated parameters, we can predict the trajectory of each person in the environment, as well as their goal position. Our experiments with real-world pedestrian datasets demonstrate that our online per-agent learning method generates more accurate motion predictions than prior methods, especially for complex scenarios with noisy, low-resolution sensors, and missing data. Using our model, we also present an algorithm to compute collision-free trajectories for robots, which takes into account kinematic constraints; our approach can compute these paths at interactive rates in scenarios with dozens of pedestrians.

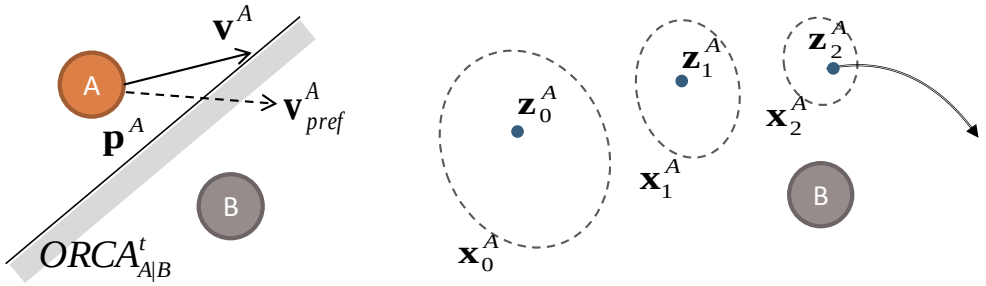
The rest of this chapter is organized as follows. Section 4.2 reviews related work. Section 4.3 provides an overview of motion-prediction methods and the RVO crowd simulation method. Section 4.4 describes how our approach combines an adaptive machine-learning framework with RVO, and Section 4.5 presents our emperical results using real-world (video recorded) data, along with analysis on these results. Section 4.6 presents a method to integrate our pedestrian prediction approach with modern mobile robot navigation techniques to improve collision avoidance for robots navigating in environments with moving pedestrians. We direct the readers to the project webpage (`http://gamma.cs.unc.edu/BRVO/`) for the related publications (Kim et al., 2012b, 2015b).

4.2 Related Work

In this section, we give an overview of prior work on motion models in robotics and crowd simulation and on their applications to pedestrian tracking and robot navigation.

4.2.1 Motion Models

There is an extensive body of work in robotics, multi-agent systems, crowd simulation, and computer vision on modeling pedestrians' motion in crowded environments. Here we provide a brief review of some recent work in this field. Many motion models have come from the fields of pedestrian dynamics and crowd simulation (Schadschneider et al., 2009; Pelechano et al., 2008). These models are broadly classifiable into few main categories: potential-based models, which models virtual agents in a crowd as particles with potentials and forces (Helbing and Molnar, 1995); boid-like approaches, which create simple rules to steer agents (Reynolds, 1999); geometric models, which compute collision-free velocities (van den Berg et al., 2008b, 2011); and field-based methods, which generate fields based on continuum theory or fluid models (Treuille et al., 2006). Among these approaches, velocity-based motion models (Karamouzas et al., 2009; Karamouzas and Overmars, 2010; van den Berg et al., 2008b, 2011; Pettré et al., 2009) have been successfully applied to the simulation and analysis of crowd behaviors and to multi-robot coordination (Snape et al., 2011); velocity-based models have also been shown to have efficient implementations that closely match real human paths (Guy et al., 2010).



(a) RVO State ($\mathbf{p}$, $\mathbf{v}$, $\mathbf{v}_{pref}$) (b) Adaptive Refinement and Prediction

Figure 4.1: (a) **RVO Simulation Overview** shows agent A's position ($\mathbf{p}$), preferred velocity ($\mathbf{v}_{pref}$) and actual velocity ($\mathbf{v}$), along with a neighboring agent B (both represented as circles). If the ORCA collision-avoidance constraints prevent an agent A from taking its preferred velocity, as shown here, the actual velocity will be the closest allowable velocity. Taken together, these elements form an agent's RVO state $\mathbf{x}$. (b) **Agent State Estimation** As new data is observed (blue dot) BRVO refines its estimate of a distribution of likely values of the RVO states (dashed ellipse). These parameters are then used with RVO to predict likely paths (as indicated by arrow).

4.2.2 People-Tracking with Motion Models

Much of the previous work in people-tracking, including (Fod et al., 2002; Schulz et al., 2003; Cui et al., 2005), attempts to improve tracking quality by making simple assumptions about pedestrian motion. In recent years, robotics and computer vision literature have developed more sophisticated pedestrian motion models. For example, long-term motion planning models have been proposed to combine with tracking. Bruce and Gordon (Bruce and Gordon, 2004) and Gong et al. (Gong et al., 2011) propose methods to first estimate pedestrians' destinations and then predict their motions along the path towards the estimated goal positions. Liao et al. propose a method to extract a Voronoi graph from the environment and predict people's motion along the edges, following the topological shape of the environment (Liao et al., 2003). Methods that use short-term motion models for people-tracking are also an active area of development. Luber et al. apply Helbing's social force model to track people using a Kalman-filter based tracker (Luber et al., 2010) . Mehran et al. also apply the social force model to detect people's abnormal behaviors from video (Mehran et al., 2009). Pellegrini et al. use an energy function to build up a goal-directed short-term collision-avoidance motion model, that they call Linear Trajectory Avoidance (LTA), to improve the accuracy of their people-tracking from video data (Pellegrini et al., 2009). More recently, Yamaguchi et al. present a pedestrian-tracking algorithm that uses an agent-based behavioral model called ATTR, with additional social and

personal properties learned from the behavioral priors, such as grouping information and destination information (Yamaguchi et al., 2011).

4.2.3 Robot Navigation in Crowds

Robots navigating in complex, noisy, and dynamic environments have prompted the development of other forms of trajectory prediction. For example, (Fulgenzi et al., 2007) use a probabilistic velocity-obstacle approach combined with the dynamic occupancy grid; this method's robot navigation, however, assumes the constant linear velocity motion of the obstacles, which is not always borne out in real-world data. (Du Toit and Burdick, 2010) present a robot planning framework that takes into account pedestrians' anticipated future location information to reduce the uncertainty of the predicted belief states. Other work uses potential-based approaches for robot path planning in dynamic environments (Pradhan et al., 2011; Svenstrup et al., 2010).

Some methods use collected data to learn the trajectories. Ziebart et al. use pedestrian trajectories collected in the environment for prediction (Ziebart et al., 2009). (Henry et al., 2010) use reinforced learning from example traces, estimating pedestrian density and flow with a Gaussian process. Bennewitz et al. apply Expectation Maximization clustering to learn typical motion patterns from pedestrian trajectories, and then guide a mobile robot using Hidden Markov Models to predict future pedestrian motion (Bennewitz et al., 2005). Broadly speaking, our work differs from these approaches in that we combine the established pedestrian simulation method (RVO) with online learning to produce individualized motion predictions for each agent.

4.3 Background

In this section, we give an overview of motion-prediction methods, including offline and online methods. We use the term 'offline prediction methods' to refer to the techniques that perform some preprocessing, either manual or automatic, of the same scenarios (Gong et al., 2011; Pellegrini et al., 2009; Yamaguchi et al., 2011; Rodriguez et al., 2009; Kratz and Nishino, 2011); in other words, offline methods use global knowledge about past, present, and future inputs. We use the term 'online prediction methods to refer to the techniques that use past observed data only without using information learned from the scene, prior to the prediction. For example, online people/object tracking

systems can be designed using filtering-based methods (e.g., Kalman filter with a linear motion model) or feature-based tracking such as the mean-shift (Collins, 2003) or KLT algorithm (Baker and Matthews, 2004). Our method, BRVO, is an online method that is based on a non-linear motion-model. BRVO is also an adaptive method that adjusts the parameters according to the environment. We briefly summarize the RVO motion model and discuss its advantages when handling sensor-based inputs.

4.3.1 Motion Prediction Methods using Agent-Based Motion Models

Our approach optimizes existing crowd simulation methods by integrating machine learning; it adapts the underlying crowd-simulation models based on what it learns from captured real-world crowd data. Some recent work has explored the use of crowd simulation as motion priors. Two notable examples of motion-prediction methods that use agent-based motion models are the previously mentioned approaches of LTA (Pellegrini et al., 2009) and ATTR (Yamaguchi et al., 2011). We briefly discuss these two methods in order to highlight how our approach differs from them and to better understand the source of our performance improvement at predicting paths in real-world datasets (discussed in Section 4.5).

Motion Model LTA and ATTR use motion models based on an energy-minimization function. Their motion computation function takes into account various terms like steering to the destination, avoiding collisions with other agents, regulations in current speed, etc. ATTR also includes a parameter to capture the effect of group behaviors. Minimization of this motion function is used to compute the desired velocity.

Offline Learning Both these methods contain an offline pre-processing step that trains their motion models using real pedestrian video data, captured in the same scene in which the system is deployed. This offline learning is used to determine free parameters (such as the weight of various terms used to compute an agents desired velocity) and for collision avoidance. In addition, ATTR learns about group behavior and destination locations, properties that affect individual behavior. The need for an offline learning phase reduces the potential applicability of these methods to mobile robots navigating in potentially unknown environments.

Destination Prediction LTA and ATTR both separate trajectory prediction from destination determination. In other words, destination information is not itself part of the prediction; instead,

it must be given a priori in order to predict each pedestrian's motion. Both methods use offline, scene-specific processes to determine the destinations. In LTA, the destination is assumed to always lie on the opposite side of the scene, while ATTR uses an offline preprocessing step based on clustering to learn a fixed number of destinations within the scene.

Our method differs from LTA and ATTR in all three categories. First, our motion model is based on a velocity-space planning technique. Parameters and interaction information are used to compute geometric information in velocity space, rather than terms in an energy function. Second, our method learns parameters online rather than offline; this allows it to learn unique parameters for each individual, as well as to dynamically and adaptively refine each individual's parameters across time. Finally, our method learns the preferred velocity (medium-term, interim information) rather than the destination (long-term, final information) which is more appropriate given our online parameter-learning paradigm.

The flexible adaptation of preferred velocity offers many advantages over techniques that rely on global information. Global, constant parameters can give excellent results when, for example, the motion of a pedestrian is linear, or when its speed is constant over time. However, human motion changes dynamically, temporally and spatially. Adjusting to the changes dynamically, for example steering directions or speeds, reduces the deviation from the actual path, as compared to motion models that use constant parameters. We include a detailed discussion about the benefits of our method in Sec. 4.5.4.

4.3.2 RVO Multi-Agent Simulation

As part of our approach, we use an underlying multi-agent simulation technique to model individual goals and interactions between people. For this model, we chose a velocity-space reasoning technique based on Reciprocal Velocity Obstacles (RVO) (van den Berg et al., 2011). RVO-based collision avoidance has previously been shown to reproduce important pedestrian behaviors such as lane formation, speed-density dependencies, and variations in human motion styles (Guy et al., 2010, 2011).

Our implementation of RVO is based on the publicly available RVO2-Library (`http://gamma.cs.unc.edu/RVO2`). This library implements an efficient variation of RVO that uses a set of linear constraints on an agent's velocities, known as *Optimal Reciprocal Collision Avoidance* (ORCA)

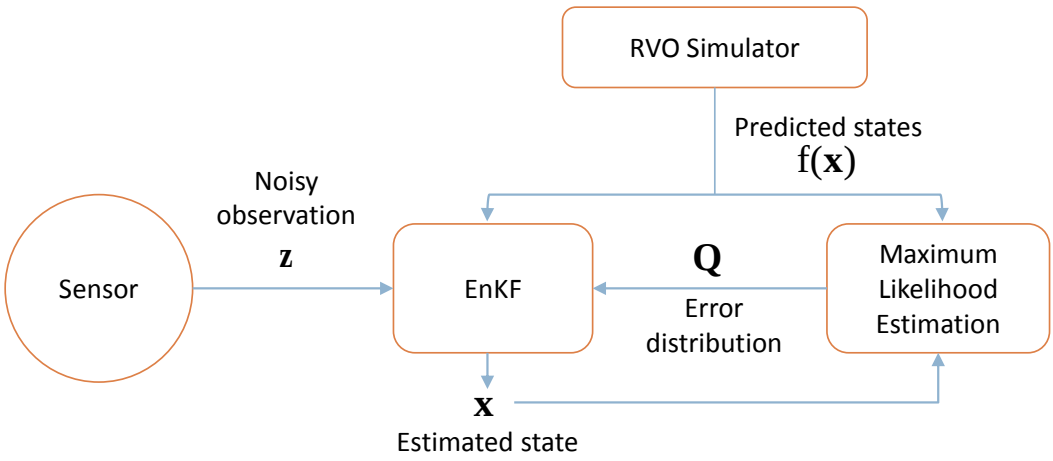


Figure 4.2: **Overview of the Adaptive Motion Model.** We estimate current state $\mathbf{x}$ via an iterative process. Given noisy data observed by the sensor, RVO as a motion model, and the error distribution matrix $\mathbf{Q}$, we estimate the current state. The error distribution matrix $\mathbf{Q}$ is recomputed based on the difference between the observed state and the prediction $f(\mathbf{x})$, and is used to refine the current estimation of $\mathbf{x}$.

constraints, in order to ensure that agents avoid collisions (van den Berg et al., 2011). Each agent is assumed to have a position, a velocity, and a preferred velocity; if an agent's preferred velocity $\mathbf{v}_{pref}$ is forbidden by the ORCA constraints, that agent chooses the closest velocity which is not forbidden. Formally:

$$\mathbf{v}^{new} = \underset{\mathbf{v} \in ORCA}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}_{pref}\|. \tag{4.1}$$

This process is illustrated in Fig. 4.2.1.

An agent's position is updated by integration over the new velocity $\mathbf{v}^{new}$ computed in Eqn. (4.1). An agent's preferred velocity is assumed to change slowly, and is modeled by a small amount of noise during each timestep. More details of the mathematical formulation are given in Sec 4.4. A thorough derivation of ORCA constraints is given in (van den Berg et al., 2011).

BRVO uses RVO combined with a filtering process. It can replace the entire framework of offline prediction methods: LTA or ATTR, for example, and the subsequent machine learning or pre-processing of data that they require. RVO was chosen as our motion model because it is especially suitable for sensor-based applications. For example, as a velocity-based method, BRVO can handle a wide range of sampling rates and crowd densities. Force-based multi-agent simulation methods, like

LTA and ATTR, compute energy potentials from the proximity of pedestrians and use these potentials as steering forces for their agents. These force-based methods thus require careful parameter tuning and small time steps to remain stable, and even with small simulation time steps, force-based methods are prone to generating oscillatory motions (Köster et al., 2013; Curtis et al., 2011). These force-based methods are thus especially problematic when using video data, which has a sensing frequency (or frame rate) of 25 - 30 fps, or 0.033 to 0.04 seconds per frame; this high sampling rate must be reduced to decrease the computational burden for interactive applications, and force-based methods are unable to cope with the larger time steps created by reducing the sample rate. Velocity-based methods, however, remain stable even when sampling rates are decreased and relatively big time steps are used. RVO, which is a velocity-based method, is shown to be stable in large-time steps, as well as when simulating large, dense crowds(Curtis et al., 2012a).

While we chose RVO for its stability in large time-steps and dense scenarios, the motion model part of BRVO is not specific to RVO. In other words, the motion computation formulation of LTA, ATTR, or any other agent-based motion model that uses preferred velocity or short-term destination information can be used instead of RVO. Our main goal is to demonstrate how an online filtering process can be combined with an RVO-based motion model to accurately predict pedestrian trajectories.

4.4 Bayesian-RVO

In this section, we provide the mathematical formulation of our Bayesian-RVO motion model, or **BRVO**. This model combines an Ensemble Kalman Filtering approach with the Expectation-Maximization (EM) algorithm to best estimate the approximate state of each agent, as well as the uncertainty in the model.

4.4.1 Problem Definition

In this section, we introduce our notation and the terminology used in the rest of this chapter.

A *pedestrian* is assumed to be a human entity that shares the environment with a mobile robot. We treat pedestrians as autonomous entities that seek to avoid collisions with each other (but not necessarily with the robot). We use n to denote the number of pedestrians in the environment. We

assume that a robot's *sensor* produces a (noisy) estimate of the position of each pedestrian. Lastly, we assume that the robot has an estimate of impassable obstacles in the environment, represented as a series of connected line segments.

We define the computation of the BRVO motion model as a state-estimation problem. Formally, the problem we seek to solve is as follows. Given a set of observations, denoted as $\mathbf{z}_0 \cdots \mathbf{z}_k$, for each pedestrian at timestep k, we solve for the RVO state $\mathbf{x}_k$ that best reproduces the motion seen so far. Given this estimate, we can then use the RVO simulation model to determine the likely future path of each pedestrian.

We propose an iterative solution to this problem based on the EM-algorithm. Assume that a robot is operating under a sense-plan-act loop: the robot first measures new (noisy) positions of each pedestrian, then iteratively updates its estimate of each person's state. To do this, we run EnKF individually for each pedestrian. The joint state can be estimated using an RVO motion model. This model uses for its input the latest estimated positions of every pedestrian in the scene, and uses that input to compute the local collision avoidance of each pedestrian. Based on the prediction, the robot can create a local collision-free navigation plan for each pedestrian.

4.4.2 Model Overview

We use the RVO algorithm to represent the state of each sensed pedestrian in a robot's environment. The state of the pedestrian at timestep k is represented as a 6D vector $\mathbf{x}_k$, which consists of an agent's 2D position, 2D velocity, and 2D preferred velocity. It is important to note that the preferred velocity is included as a part of the state vector and does not need to be manually pre-determined in a scene-specific fashion. We use a Bayesian inference process to adaptively find the state that best represents all previously observed motion data (accounting for sensing uncertainty) for each pedestrian in the robot's environment. In our case, we assume that the robot has an ability to observe the relative position of the pedestrians.

Given an agent's RVO state $\mathbf{x}_k$, we use the RVO collision-avoidance motion model, denoted here as f, to predict the agent's next state $\mathbf{x}_{k+1}$. We denote the state prediction error of f at each time step as $\mathbf{q}$. This leads to our motion model of:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{q}. \tag{4.2}$$

Additionally, we assume that the sensing of the robot can be represented by a function h that provides an observed state $\mathbf{z}_k$, which is a function of the person's true state $\hat{\mathbf{x}}_k$ plus some noise from the sensing processing, which is denoted as $\mathbf{r}$. That is:

$$\mathbf{z}_k = h(\hat{\mathbf{x}}_k) + \mathbf{r}. \tag{4.3}$$

An overview of this adaptive motion prediction model is given in Fig. 4.2.

Simplifying Assumptions The model given by Eqns. (4.2) and (4.3) is very general. In order to efficiently estimate the state $\mathbf{x}_k$ from the observations $\mathbf{z}_k$, we must make some simplifying assumptions, which allow us to develop a suitable learning approach for our adaptive model. First we assume that the error terms q and r are independent at each timestep, and that they follow a zero-meaned Gaussian distribution with covariances Q and R, respectively. That is:

$$\mathbf{q} \sim N(0, Q), \tag{4.4}$$

$$\mathbf{r} \sim N(0, R). \tag{4.5}$$

We further assume that the sensor error r is known or can be well-estimated. This is typically possible by making repeated measurements of known data points to compute the average error. In many cases, this error will be provided by the manufacturer of the sensing device. When using a tracking or recognition system to process the position input, R should be estimated from the tracking and recognition error.

To summarize, the function h is specified by the robot's sensors, and the matrix R characterizes the estimated accuracy of these sensors. The f function is the motion model that will be used to predict the motion of each agent (here RVO), and Q represents the accuracy of this model.

Our BRVO formulation uses the RVO-based simulation model to represent the function f and EnKF to estimate the simulation parameters that best fit the observed data. At a high level, EnKF operates by representing the potential state of an agent at each timestep as an ensemble (or collection) of discrete samples. Each sample is updated according to the motion model f. A mean and standard deviation of the samples is computed at every timestep in order to estimate the new state.

(a) Campus

(b) Bottleneck

(c) Street

(d) Students

Figure 4.3: **Benchmarks used for our experiments** (a) In the Campus dataset, students walk past a fixed camera in uncrowded conditions. (b) In the Bottleneck dataset, multiple cameras track participants walking into a narrow hallway. (c) In the Street dataset, a fixed camera tracks pedestrian motion on a street. (d) In the Students dataset, a fixed camera tracks students' motion in a university.

In addition, we adapt the EM-algorithm to estimate the model error Q for each agent. Better estimation of Q improves the Kalman filtering process, which in turn improves the predictions given by BRVO. This process is used iteratively to predict the next state and to refine the state estimation for each agent, as depicted in Fig 4.2.1. More specifically, we perform the EnKF and EM steps for each agent separately while simultaneously taking into account all agents present in the motion model $f(x)$. This gives more flexibility in dynamic scenes, allowing the method to handle cases when the robot moves or when pedestrians arrive or depart and change the number of agents in the sensed area. Because each inference step is performed per-agent on a fixed-size state (as opposed to a varying size state vector which considers all agents together), agents entering or leaving can be easily incorporated.

4.4.3 State Representation

The above representation can be more formally summarized with the following notation. We represent each agent's state, $\mathbf{x}$, as the six-dimensional set of RVO parameters (see Sec 4.3.2):

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{v}_{pref} \end{bmatrix}, \tag{4.6}$$

where $\mathbf{p}$ is the agent's position, $\mathbf{v}$ the velocity, and $\mathbf{v}_{pref}$ the preferred velocity. The crowd dynamics model f is then:

$$f(\begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{v}_{pref} \end{bmatrix}) = \begin{bmatrix} \mathbf{p} + \mathbf{v}\Delta t \\ \text{argmin}_{\mathbf{v} \in ORCA} \|\mathbf{v} - \mathbf{v}_{pref}\| \\ \mathbf{v}_{pref} \end{bmatrix}. \tag{4.7}$$

Also, the robot can observe the relative positions of the pedestrians, which can be represented as a function h of the form:

$$h(\begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{v}_{pref} \end{bmatrix}) = \mathbf{p}. \tag{4.8}$$

The EnKF use some assumptions for the observation model. In general though, our BRVO framework makes no assumption about the linearity of the sensing model. More complex sensing functions (for example, advanced computer vision techniques or the integration of multiple sensors) can be represented by modifying the function h in accordance with the desired sensing model.

4.4.4 State Estimation

When f and h are linear functions, optimal estimation of the simulation state $\mathbf{x}_k$ is possible using Kalman Filtering. However, because f corresponds to the RVO simulation model and we can use any arbitrary sensing model h, the resulting system is a non-linear system and there is no simple way to compute the optimal estimate.

Therefore, we use EnKF, a sampling-based extension of Kalman Filtering, to estimate the corresponding agent state for each pedestrian. EnKF is a Bayesian filtering process, which takes as input an estimate of the prediction error $\mathcal{Q}$ and observations $\mathbf{z}_0 \cdots \mathbf{z}_k$ and produces an estimate of the true pedestrian states $\mathbf{x}_k$ as a distribution of likely states $\mathcal{X}_k$. Given n pedestrians to track, each with a d dimensional state vector, the distributions $\mathcal{X}_k$ are represented in an ensemble representation where each distribution is represented by m samples. EnKF provides an estimate of the true distribution of likely pedestrian states by representing it with these m samples.

One benefit of performing EnKF for each pedestrian is that we can easily handle entering and leaving pedestrians, unlike a joint-state estimation. Instead, crowd interactions such as collision avoidance behavior is estimated by our motion model f, using the latest estimation of the pedestrians in the scene.

The ensemble representation for distributions is similar to a particle representation used in particle filters (Arulampalam et al., 2002), with the distinction that the underlying distribution is assumed to be Gaussian. The ensemble representation is widely used for estimation of systems with very large dimensional state spaces, such as climate forecasting (Hargreaves et al., 2004) and traffic conditions (Work et al., 2008). In general, the EnKF algorithm works particularly well for high-dimensional state spaces (Evensen, 2003) and can overcome some of the issues presented by non-linear dynamics (Blandin et al., 2012). For a more detailed explanation of EnKF, we refer readers to a standard textbook in statistical inference such as (Casella and Berger, 2002).

The motivation for this algorithmic choice is twofold. First, EnKF makes the same assumptions we laid forth in Sec 4.4.2: that is, a (potentially) non-linear f and h combined with a Gaussian representation of error. Second, as compared to methods such as particle filters, EnKF is very computationally efficient, providing higher accuracy for a given number of samples. This is an important consideration for the low-to-medium power onboard computers commonly found on a mobile robot, especially given a goal of online, real-time estimation.

4.4.5 Maximum Likelihood Estimation

The accuracy of the states estimated by the EnKF algorithm is a function of the parameters defined in Eqns 4.2-4.5: f, Q, h, and R. Three of these parameters are pre-determined given the problem set-up: the sensing function h and the error distribution R are determined by the sensor's

specifications, and f is determined by the motion model chosen. However, the choice of motion prediction error distribution Q is still a free parameter. We propose a method of estimating the optimal value for Q based on the Expectation Maximization or EM-algorithm.

The EM-algorithm is a generic framework for learning (locally) optimal parameters iteratively with latent variables. The algorithm alternates between two steps: an E-step, which computes expected values for various parameters, and an M-step, which computes the distribution maximizing the likelihood of the values computed during the E-step (for a more thorough discussion see (McLachlan and Krishnan, 2008)).

In our case, the E-step is performed via the EnKF algorithm. This step estimates the most likely state given the parameters Q. For the M-step, we need to compute Q that maximizes the likelihood of values estimated from EnKF. This probability will be maximized with Q that best matches the observed error between the predicted state and the estimated state. We can compute this value simply by finding the average error for each sample in the ensemble at each timestep for each agent.

By iteratively performing the E and M steps, we continuously improve our estimate of Q, which will in turn improve the quality of the learning and the predictiveness of the method. Ideally, one should iterate over the E and M steps until the approach converges. In practice, the process converges fairly rapidly. Due to the online process nature of our approach, we limit the number of iterations to three, which we found to be empirically sufficient. We analyze the resulting improvement produced by the EM algorithm in Section 4.5.1.3.

4.4.6 Implementation

Pseudocode for our BRVO algorithm is given in Algorithm 14. We represent the distribution of likely RVO states as an ensemble of m samples. We set $m = 1000$ for the results shown in Section 4.5. Lines 2 through 6 perform a stochastic prediction of the likely next states. Lines 7 through 10 correct this predicted value based on the observed data from the robot's sensor. Z_k is a measurement error covariance matrix, and $\Sigma_k Z_k^{-1}$ is an ensemble version of Kalman Gain matrix. Lines 11 through 14 perform a maximum likelihood estimation of the uncertainty in the prediction.

Algorithm 1: Bayesian-RVO

Input: Observed positions over time $\mathbf{z}_1 ... \mathbf{z}_t$, crowd motion simulator f, estimated initial error covariance $\mathbf{Q}$, sensor function h, sensor noise $\mathbf{R}$, and the number of samples m.

Output: Estimated agent's state distributions $\mathbf{x}_1 ... \mathbf{x}_t$

foreach $k \in 1 \dots t$ **do**

  `// EnKF Predict Step`

  foreach $i \in 1 \dots m$ **do**

    Draw $\mathbf{q}_{k-1}^{(i)}$ from $\mathbf{Q}$, $\hat{\mathbf{x}}_k^{(i)} = f(\hat{\mathbf{x}}_{k-1}^{(i)}) + \mathbf{q}_{k-1}^{(i)}$;

    Draw $\mathbf{r}_k^{(i)}$ from $\mathbf{R}$, $\hat{\mathbf{z}}_k^{(i)} = h(\hat{\mathbf{x}}_k^{(i)}) + \mathbf{r}_k^{(i)}$;

  $\bar{\mathbf{z}}_k = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{z}}_k^{(i)}$;

  $Z_k = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)(\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)^T$;

  `// EnKF Correct Step`

  $\bar{\mathbf{x}}_k = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{x}}_k^{(i)}$;

  $\Sigma_k = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{x}}_k^{(i)} - \bar{\mathbf{x}}_k)(\hat{\mathbf{z}}_k^{(i)} - \bar{\mathbf{z}}_k)^T$;

  foreach $i \in 1 \dots m$ **do**

    $\hat{\mathbf{x}}_k^{(i)} = \hat{\mathbf{x}}_k^{(i)} + \Sigma_k Z_k^{-1} (\mathbf{z}_k - \hat{\mathbf{z}}_k^{(i)})$;

  `// Maximum Likelihood Estimation`

  $\mathbf{Q}_{k-1} = \mathbf{Q}$;

  foreach $i \in 1 \dots m$ **do**

    $\mathbf{Q}_k += (\hat{\mathbf{x}}_k^{(i)} - f(\hat{\mathbf{x}}_{k-1}^{(i)}))(\hat{\mathbf{x}}_k^{(i)} - f(\hat{\mathbf{x}}_{k-1}^{(i)}))^T$;

  $\mathbf{Q} = \frac{k-1}{k} \mathbf{Q}_{k-1} + \frac{1}{k} \mathbf{Q}_k$

4.5 Results and Analysis

In this section, we show the results from our BRVO approach when applied to real-world datasets. We begin by isolating the effect of each component of BRVO separately, namely: EnKF, EM and individual parameter learning (Section 4.5.1). We then measure the robustness of our approach across a variety of factors commonly found in real world situations including: sensor noise, low sampling rates, and spatially varying densities. Finally, we provide a quantitative comparison to other recent techniques across a variety of datasets.

Our analysis here is performed on four different datasets with various levels of noise and different sampling rates, illustrated in Fig. 4.3. We briefly describe each dataset below:

Campus Video data of students on campus recorded from the top of the ETH main building in Zurich, extracted manually every 0.4 second (Pellegrini et al., 2009). We extracted three sequences from this data, each containing about 10 seconds of pedestrian interaction: Campus-1 (7 pedestrians), Campus-2 (11 pedestrians), Campus-3 (18 pedestrians).

Bottleneck Optical tracking equipment captured the motion of participants in a lab environment (Boltes et al., 2010). Participants moved through a bottleneck opening into a narrow passage. Data was collected with a passage width of 2.5 meters. The study has about 170 participants, and we use the trajectory data subsampled at a rate of 0.4 second.

Street This video of low-density pedestrian traffic was recorded from a street view (Lerner et al., 2009). Manually extracted pedestrian trajectories are provided with the video. The dataset contains two sequences. Street-1 contains motion of 148 pedestrians over a period of roughly 5 minutes, and Street-2 contains the motion of 204 pedestrians over a period of roughly 7 minutes.

Students This video was recorded from a street view (Lerner et al., 2009). The dataset contains the motion of 434 students over a period of roughly 3 minutes, tracked manually.

To explore the relative performance of our method we will compare it to other typical prediction approaches commonly found in robotics applications as described below:

Constant Velocity (ConstVel) This model assumes that all agents will continue their most recent instantaneous velocity (inferred from the last two positions) for the immediate future (Eqn

4.9).

$$f(\begin{bmatrix}\mathbf{p}\\ \mathbf{v}\end{bmatrix}) = \begin{bmatrix}\mathbf{p}+\mathbf{v}\Delta t\\ \mathbf{v_{k-1}}\end{bmatrix} = \begin{bmatrix}\mathbf{p}+\mathbf{v}\Delta t\\ \frac{\mathbf{p_{k-1}}-\mathbf{p_{k-2}}}{\mathbf{\Delta t}}\end{bmatrix}. \tag{4.9}$$

Constant Acceleration (ConstAcc) This model assumes that all agents will continue their most recent acceleration (inferred from the last two velocities) for the immediate future (Eqn 4.10).

$$f(\begin{bmatrix}\mathbf{p}\\ \mathbf{v}\\ \mathbf{a}\end{bmatrix}) = \begin{bmatrix}\mathbf{p}+\mathbf{v}\Delta t\\ \mathbf{v}+\mathbf{a}\Delta t\\ \mathbf{a_{k-1}}\end{bmatrix} = \begin{bmatrix}\mathbf{p}+\mathbf{v}\Delta t\\ \mathbf{v}+\mathbf{a}\Delta t\\ \frac{\mathbf{v_{k-1}}-\mathbf{v_{k-2}}}{\mathbf{\Delta t}}\end{bmatrix}. \tag{4.10}$$

Kalman Filtering (KF) We use a position and velocity based Kalman Filter (i.e., using Eqn 4.9 as the model) which uses standard Gaussian filtering techniques to account for potential noise in the environment.

Unlike the ConstVel and ConstAcc models, using a Kalman filter will account for noise in the sensing and motion models. However, this comes at the cost of a delayed response to changes in the environment (due to data smoothing) leading to KF performing worse than ConstVel in situations where there is little noise and smooth, consistent motion. BRVO avoids this issue as demonstrated in Section 4.5.2.

We also compare the results against two recently proposed *offline* pedestrian tracking methods: LTA (Pellegrini et al., 2009) and ATTR (Yamaguchi et al., 2011). Both methods use offline training to learn models of typical pedestrian goals and speeds for a given observation area. Because of this training step, both methods are inappropriate for use in a mobile robot. However, they provide a good understanding of the state-of-the-art in pedestrian tracking. Surprisingly, in all of the above datasets, BRVO outperforms even these offline model suggesting the flexibility of learning a model per-agent outweighs the advantages from offline training. More discussion of this comparison is given in Section 4.5.3.

In all cases, we report errors as the distance between ground-truth (manually annotated) positions and the estimated positions. We also provide various quantitative and qualitative performance measures including the long-term prediction success rate, shapes of the trajectories and prediction error distributions.

4.5.1 Method Analysis

BRVO has three main components: a probabilistic Bayesian inference framework, per-agent online state estimation, and a continuing refinement of filtering parameters through the EM algorithm. Each of these components provides a substantial contribution to the overall performance.

4.5.1.1 EnKF for Online Learning

Different approaches to Bayesian inference can be used to perform the online learning task. Here we compare EnKF, Particle filter, and Unscented Kalman filter (Wan and Van der Merwe, 2000), all using RVO as the motion model, all without EM. For this experiment, we use the three sequences from various points in the the Campus dataset.

The first five frames of ground truth positions are used as sensor input $\mathbf{z}$, and each method is issued to perform prediction for the remaining frames with the learned parameters. We use a fixed $\mathbf{Q}$ for each learning and prediction step: $[0.5m] * I$. The number of samples for both EnKF and Particle filter is 50000. We measured the mean error (squared distance) between the predicted position and ground truth position. The results are summarized in Table 4.1.

	Campus-1	Campus-2	Campus-3
EnKF	**0.359**	**0.462**	**0.388**
Particle filter	0.366	0**.462**	0.389
Unscented KF	0.476	0.501	0.420

Table 4.1: **Comparisons of different Bayesian learning algorithms** Root mean squared error between the predicted position and ground truth position in meters (best shown in bold).

Both EnKF and Particle Filter outperform Unscented Kalman Filter, with EnKF having the best average performance. While other sampling based approaches such as Particle filters may provide similar results, EnKF's consistent performance across these and other datasets motivated our choice of EnKF for Bayesian inference throughout this work.

4.5.1.2 Online vs Offline Learning

We can measure the improvements made by introducing online parameter-estimation technique, by comparing our method to a closely related offline approach. Following the approach proposed in

LTA (Pellegrini et al., 2009) and ATTR (Yamaguchi et al., 2011), we developed an offline prediction technique based on RVO referred to here as **RVO-offline**. At a high level, RVO-offline uses a genetic algorithm to optimize RVO parameters to best match the training dataset of the ground truth motion of individuals in the environment. The destination of each agent is set side of the environment opposite of where the agent entered (as done in (Pellegrini et al., 2009)). For ConstVel, ConstAcc, and BRVO, we assume that we get good sensor inputs for first two frames and compute the initial velocity for each pedestrian from the first two groundtruth positions. For RVO-Offline, we assume that we have good global information about destinations and motion parameters including preferred speed, and compute the initial velocity from these parameters.

We compare this RVO-offline model with the constant velocity model, the constant acceleration model, and with BRVO using the above mentioned sequences from the Campus dataset. To compute the error, we use the first half of the data set for training and the second half for testing. Each method attempts to predict the final agent position from the position at the halfway point. Fig. 4.4 shows the mean error for different approaches.

While the static, offline-learning RVO model performs slightly better than the simpler motion models, BRVO offers an even more significant reduction in error, demonstrating the advantage of learning individual time-varying parameters for each agent. Additionally, unlike offline models, BRVO does not need prior knowledge of the current environment (such as likely goals), instead learning each agents destination dynamically through EnKF with RVO. The difficultly in choosing an individual's goal position, is one of the main challenges to using offline methods in an autonomous robot system. More discussion of the impact of goals in offline methods is given in Section 4.5.3

4.5.1.3 EM based estimation refinement

As discussed in Section 4.4.5, our proposed BRVO approach uses the EM-algorithm to automatically refine the parameters used in the online Bayesian estimation. The effect of EM can be demonstrated visually by graphing the estimated uncertainty in state as shown in Fig. 4.5(a); as new data is presented and the pedestrians interact with each other, the estimated uncertainty, $\mathbf{Q}$, decreases. Without the EM step, the initially given estimated uncertainty $\mathbf{Q}$ must be used without refining its values.

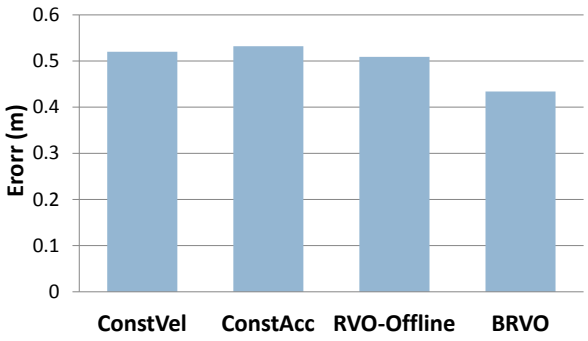
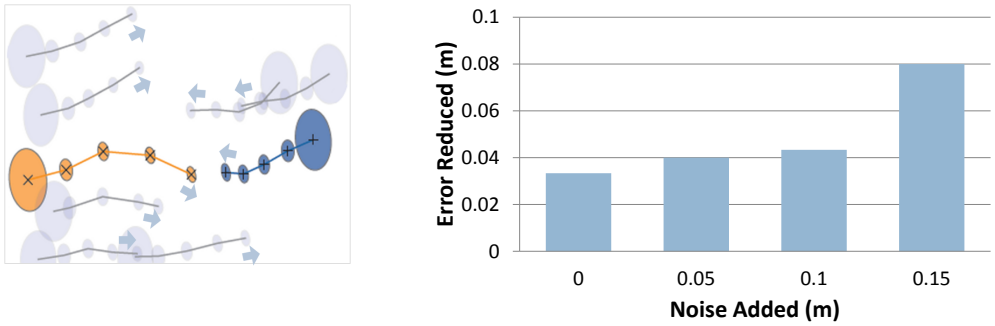


Figure 4.4: **Comparison of Error on Campus Dataset.** BRVO produces less error than Constant Velocity and Constant Acceleration predictions. Additionally, BRVO results in less error than using a static version of RVO whose parameters were fitted to the data using an offline global optimization.



(a) Estimation Refinement

(b) Improvement from EM feedback Loop

Figure 4.5: **The effect of the EM algorithm.** (a) This figure shows the trajectory of each agent and the estimated error distribution (ellipse) for the first five frames of Campus-3 data. The estimated error distributions gradually reduces as agents interact. For two highlighted agents (orange and blue), their estimated positions are marked with 'X'. (b) The improvement provided by the EM feedback loop for various amounts of noise. As the noise increases, this feedback becomes more important.

Figure 4.5(b) shows the quantitative advantage of using EM, by comparing our results in the Campus sequences with EM to those without. When using high quality data with little noise, EM shows a slight improvement in the results. If we artificially add noise to the data (while keeping all other parameters constant), the effect of the active feedback loop becomes even more important and the EM algorithm offers increasing improvements in error estimation.

4.5.2 Environmental Variations

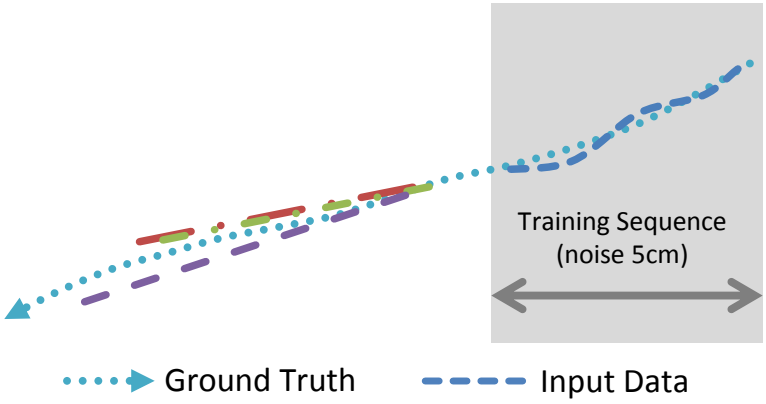
Many factors in a robot's environment can impact how well a predictive tracking method such as BRVO works. For example, poor lighting conditions can reduce the efficacy of vision based sensors increasing the noise in the sensor data. Likewise, in dense scenarios, people can tend to change their velocities frequently to avoid collisions creating increasing the uncertainty in their motion. Similarly, when a robot's energy gets low it may choose to reduce the frequency of sampling sensor data to save power. In all of these scenarios, the difficulty in predicting a pedestrians motion can increase significantly. Importantly, the advantage that BRVO has over simple prediction methods grows larger as scenarios grow in complexity through additional sensor noise, more dense environments, or less frequent data readings as shown in the following experiments.

4.5.2.1 Noisy Data

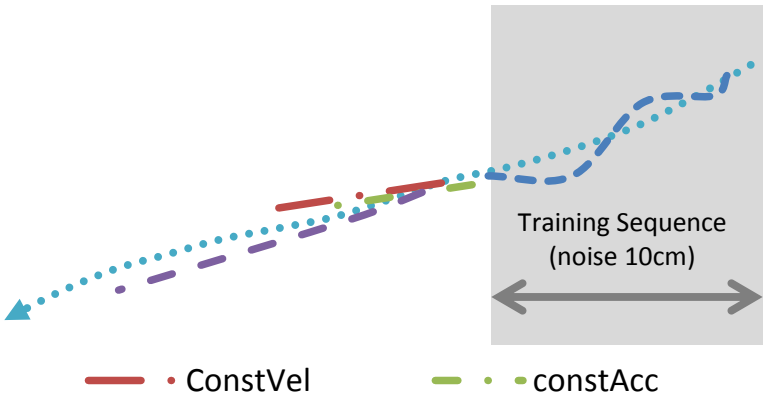
To analyze how BRVO responds to noise in the data, we compare its performance across varying levels of synthetic noise added to the ground truth data in the Campus dataset. Like before, BRVO learns for the first 5 frames, and uses that information to predict the position for remaining frames.

Fig. 4.6 compares predictions from BRVO, constant velocity, and constant acceleration for one pedestrian's moving right-to-left across the campus. The figure shows the prediction given with varying amounts of noise levels added to the training data (respectively, 0.05m noise, 0.1m noise and 0.15m noise). After the training frames, we assume that no further sensor information is given and predict the paths using only the motion models. As can be seen in the figure, BRVO reduces sensitivity to this type of noise and performs better overall at predicting both the direction of the path and the absolute speed of the pedestrian.

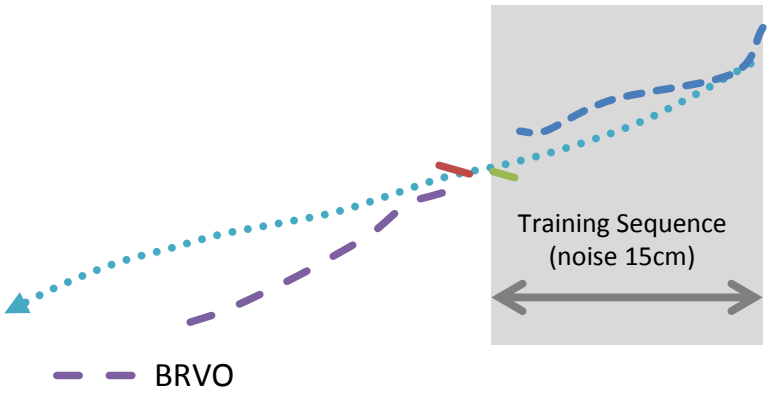
A comparison of the average prediction error across all of the Campus sequences is given in Fig. 4.7 which compares BRVO with the constant velocity model, constant acceleration model,



(a) 0.05m



(b) 0.1m



(c) 0.15m

Figure 4.6: **Path Comparisons** These diagrams shows the paths predicted using BRVO, Constant Velocity, and Constant Acceleration. For each case, ground-truth data with 0.05m noise, 0.1m noise and 0.15m are given as input for learning (blue dotted lines in shaded regions). BRVO produces better predictions even with a large amount of noise.

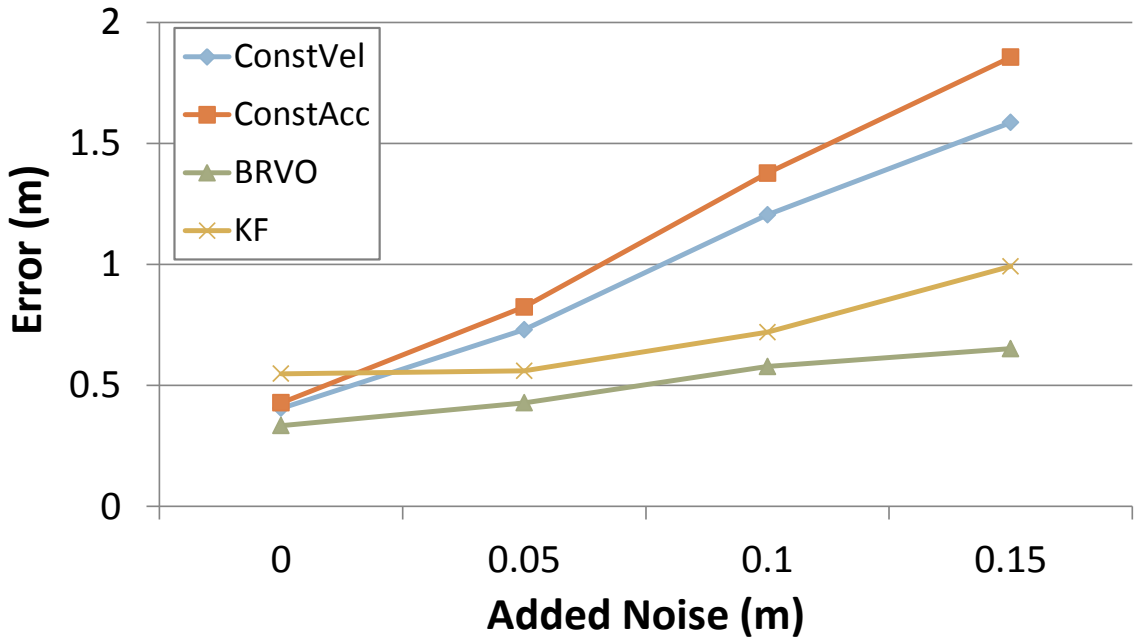
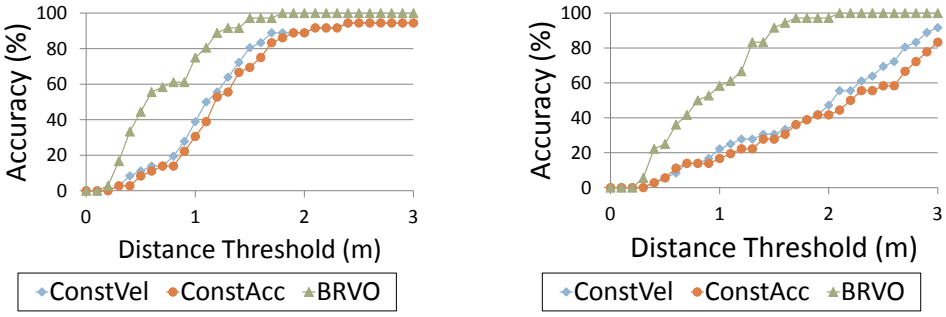


Figure 4.7: **Mean prediction error (lower is better).** Prediction error after 7 frames (2.8s) on Campus dataset. As compared to the Constant Velocity and Constant Acceleration models, BRVO can better cope with varying amounts of noises in the data.

and KF. Both KF and BRVO are initialized with the same conditions (e.g., Sensor noise and initial error covariance matrix). While adding noise always increases the prediction error, BRVO was less sensitive to the noise than other methods. Fig. 4.8 shows the percentage of correctly predicted paths within varying accuracy thresholds. At an accuracy threshold of 0.5m, BRVO far outperforms the constant velocity and constant acceleration models (44% vs 8% and 11% respectively) even with little noise. With larger amounts of noise, these differences tend to increase.

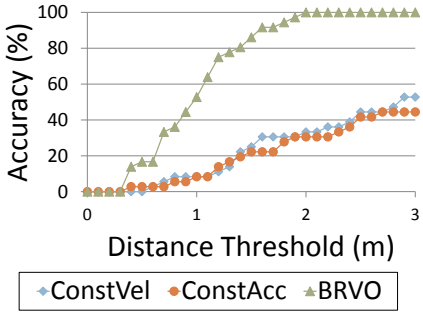
4.5.2.2 Density Dependance

In a dense crowd, an individual's motion is restricted significantly by his or her neighbors. It is thus difficult to predict pedestrian trajectories without taking into account the neighbors. Because individuals in the bottleneck scenario undergo many different densities, it provides the clearest effect of density has on prediction accuracy. For analysis purposes, the scenario was spilt in three sections: a dense front section where people backup trying to make their way through the narrow bottleneck opening (3.6 $people/m^2$), a more moderately dense section at the entrance (3 $people/m^2$), and the



(a) 0.05m

(b) 0.1m



(c) 0.15m

Figure 4.8: **Prediction Accuracy (higher is better)** (a-c) Shows prediction accuracy across various accuracy thresholds. The analysis is repeated at three noise levels. For all accuracy thresholds and for all noise levels BRVO produces more accurate predictions than the Constant Velocity or Constant Acceleration models. The advantage is most significant for large amounts of noise in the sensor data, as in (c).

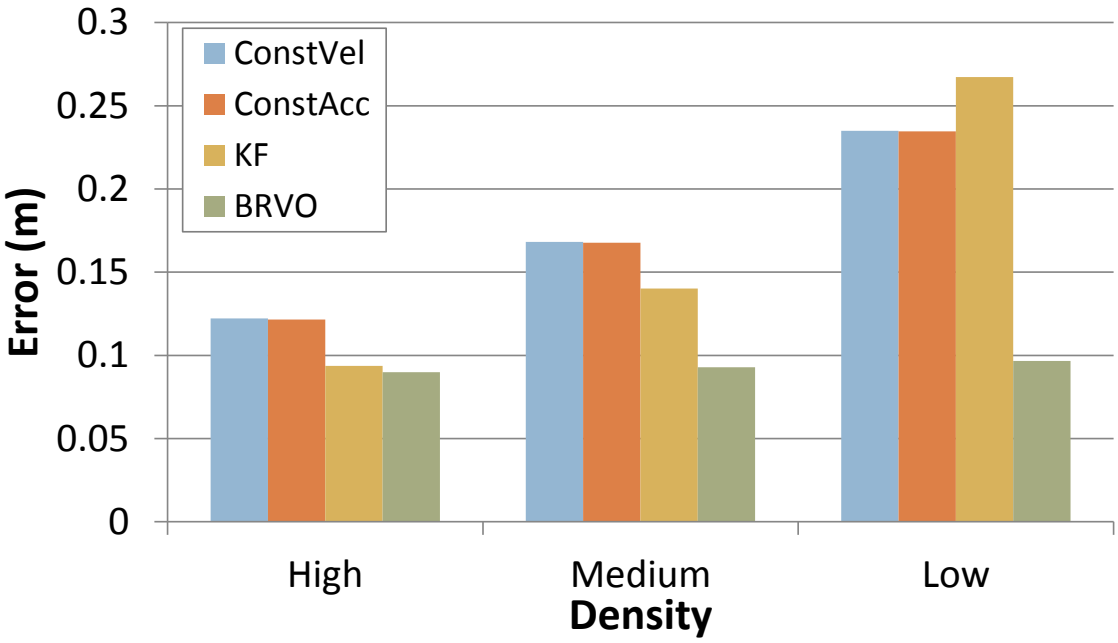


Figure 4.9: **Error at Various Densities (lower is better).** In high-density scenarios, the agents move relatively slowly, and even simple models such as constant velocity perform about as well as BRVO. However, in low- and medium-density scenarios, where the pedestrians tend to move faster, BRVO provides more accurate motion prediction than other models. KF fails to quickly adjust to rapid speed changes in the low-density region, resulting in larger errors than other methods. In general, BRVO performs consistently well across various densities.

hallway itself which is less dense (2 $people/m^2$). Note that this last region is not only the least dense, but has the simplest motion consisting primarily of moving straight through the hallway.

Fig. 4.9 compares BRVO's error to that of the constant velocity, constant acceleration, and Kalman filter models. The constant velocity, constant acceleration, and Kalman filter models have large variations in error for different regions of the scenario with different densities. The error is smallest in the highest density region. This is because the speed of the pedestrians are relatively slow, and the effect of the wrong prediction does not produce a large deviation from the ground-truth. As the density gets lower and the speed increases, we observe that the error increases. Unlike constant velocity and constant acceleration, which instantly adopts the speed, Kalman filter model have latency adapting to the abrupt speed changes, which may have resulted in a bigger error in the lower density region. In contrast, the BRVO approach performs well across all densities because it can dynamically adapt the parameters for each agent for each frame.

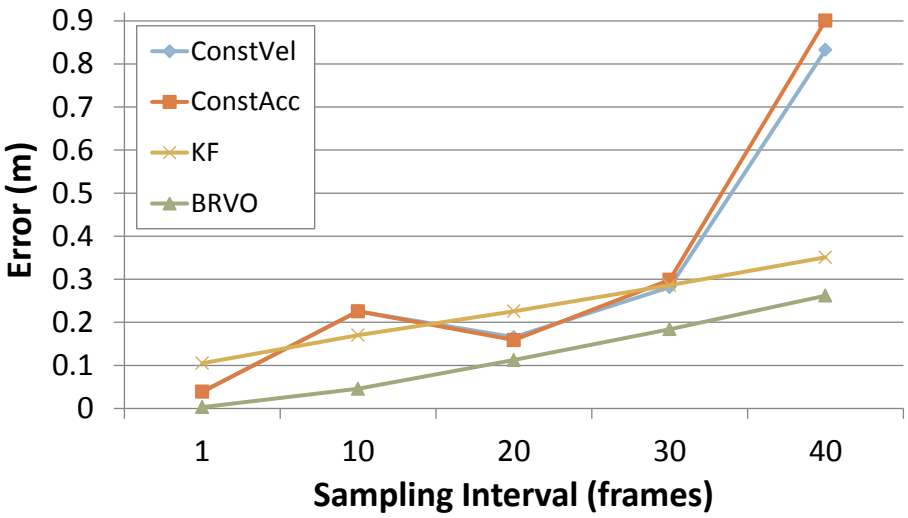


Figure 4.10: **Error vs Sampling Interval** As the sampling interval increases, the error of Constant Velocity and Constant Acceleration estimations grows much larger than that of KF or BRVO. BRVO results have the lowest error among the four methods.

4.5.2.3 Sampling Rate Variations

Our method also works well with very large timesteps, i.e., when sensor data is collected at a sparse rate over long intervals. To demonstrate this, we show the results on the Street dataset with varying sampling intervals to sub-sample the data. We chose Street-1 scenario, which has the highest framerate (at 0.04s per frame) of all the datasets, and then sub-sampled the data to reduce the effective framerate. Fig. 4.10 shows the graph of the mean error versus the sampling interval. The results show that our method performs very well compared to the constant velocity and constant acceleration model across all the sampling intervals, and has less sensitivity to low framerates.

4.5.3 Model Performance Comparison

Taken together, the results provided in this section demonstrate the ability of BRVO to provide robust, high-quality motion prediction in a variety of difficult scenarios. We can directly compare our results with the results of LTA (Pellegrini et al., 2009) and ATTR (Yamaguchi et al., 2011), which report performance numbers for some of the same benchmarks. Because LTA and ATTR require offline training, we also measure the performance of KF alone in the same scenarios, in order to provide a comparison with an online method, but with a simple linear motion model.

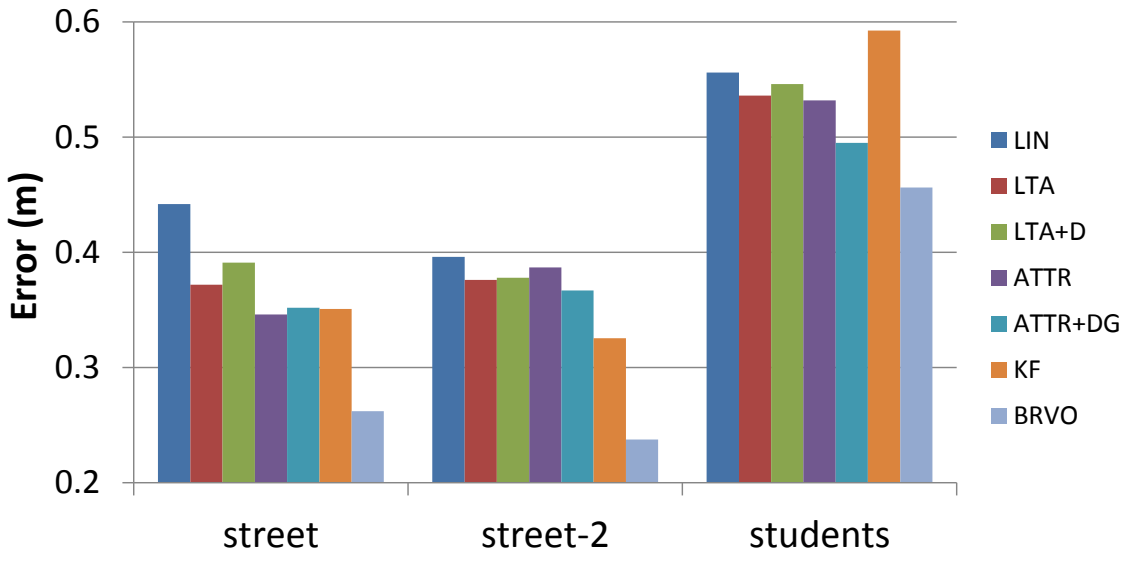


Figure 4.11: **Comparison to State-of-the-art Offline Methods** We compare the average error of LIN (linear velocity), LTA, ATTR, KF and BRVO. Our method outperforms LTA and ATTR, with an 18-40% error reduction rate over LIN in all three different scenarios. Significant improvement is made; compare the 4-16% and 7-20% error reduction rate of LTA and ATTR over LIN, respectively.

We use the Street-1, Street-2 and Students datasets, all sampled every 1.6 seconds, and measure mean prediction error for every agent in the scene during the entire video sequence. Fig. 4.11 compares the prediction accuracy of LTA, ATTR in various configurations, KF, and BRVO with LIN (the constant velocity model). Our method outperforms LTA and ATTR with 18-40% error reduction rate in across the three different scenarios. LTA and ATTR use the ground truth destinations for prediction; LTA+D and ATTR+DG use destinations learned offline, as explained in (Yamaguchi et al., 2011); ATTR+DG uses grouping information learned offline. Even though BRVO is an online method, it shows significant improvement in prediction accuracy on all three datasets, producing less error than other approaches.

We observe a relatively large error when using the students dataset. This dataset is especially difficult to estimate, as explained in (Yamaguchi et al., 2011); the irregular behavior of the pedestrians, including sudden stops, wandering, and chatting, reduce the predictive performance. This reduced predictability also affects the KF test, since it fails to quickly adjust to the irregular behaviors.

4.5.4 Analysis

The ability to learn the preferred velocity on the fly comes from the BRVO framework. BRVO can be compared with other prediction methods that use motion models followed by data pre-processing and offline parameter learning (e.g., LTA or ATTR). We have shown quantitative comparisons with LTA and ATTR on three different datasets. Overall, the differences make our approach better suited to the domain of mobile robot navigation.

We can also estimate how much of BRVO's performance improvement comes from the addition of the filtering process and how much from using the RVO motion model, although we believe that combining the motion model and filtering algorithm results in considerable improvement. To quantify the contributions of each, we provided additional comparisons: one with the constant-velocity model and one with the Kalman filter and the constant-velocity model. These two models are both online methods that use a linear motion model. The Kalman filter method is more resilient with the noisy inputs than is the constant velocity model, but adjusts more slowly when inputs change rapidly. Thus, the performance is influenced by the characteristics of the data. We have shown that our method actually outperforms both the constant-velocity and Kalman-filter methods in various scenarios.

4.6 Robot Navigation with BRVO

One potential application of our BRVO is for safer navigation for autonomous robot vehicles through areas of dense pedestrian traffic. Here we describe a simple method to integrate BRVO with the GVO (Generalized Velocity Obstacle) motion-planning algorithm proposed in (Wilkie et al., 2009) to achieve more effective navigation of a simulated car-like robot through a busy walkway. The GVO navigation method is a velocity obstacle based technique for navigating robots with kinematic constraints. In our case, we use car-like kinematic constraints, and we assume the robot has the ability to sense the positions of nearby moving obstacles (such as pedestrians) though with some noise. The robot uses our BRVO technique to predict the motion of each pedestrian as it navigates through the crowded walkway to its goal position.

The robot's configuration is represented as its position (x, y) and the orientation ϕ. The robot has controls u_s and u_ϕ, which are the speed and steering angle of the robot, respectively. Fig. 4.12

shows the kinematic model of the robot. Its constraints are defined as follows:

$$\begin{aligned} x'(t) &= u_s \cos\theta(t), \\ y'(t) &= u_s \sin\theta(t), \\ \theta'(t) &= u_s \frac{\tan u_\phi}{L}, \end{aligned} \tag{4.11}$$

where L is the wheelbase of the robot. We assume $L = 1\text{m}$.

Assuming that the control remains constant for the time interval, the robot's position $R(t, u)$ at time t given the control u can be derived as follows:

$$R(t,u) = \begin{pmatrix} \frac{1}{\tan(u_\phi)} \sin(u_s \tan(u_\phi)t) \\ -\frac{1}{\tan(u_\phi)} \cos(u_s \tan(u_\phi)t) + \frac{1}{\tan(u_\phi)} \end{pmatrix}. \tag{4.12}$$

GVO samples the space of controls for a robot to determine which controls lead to collisions with obstacles. This is done using an algebraic formulation for the minimum distance between the robot, given its kinematic model, and an obstacle, given its predicted trajectory, up to a time horizon. For our work, the minimum distance between a robot using the simple-car kinematic model and a linearly-moving obstacle is derived. We limit the maximum velocity of the robot to $1.5m/s$. For each control sample, this minimum distance is solved numerically. Every control that yields a distance greater than the sum of the robot and obstacle widths is considered free. Of these free controls, a control is then selected that brings the robot closest to its goal. Formally, the actual control u is:

$$u = \underset{u' \notin VO}{\operatorname{argmin}}\, dist(R(t,u'), g, t')), \tag{4.13}$$

where u' is subject to a velocity obstacle constraint for each moving obstacle, and VO is the velocity obstacle of the robot. $dist(R(t,u'), g, t')$ is the minimum distance between the robot and the robot's goal position g, over a time horizon t'. We use 5 seconds for t', which is bigger than the simulation update time horizon to give some a 'look ahead' effect. The computation of u is a slight modification from the original formulation (Wilkie et al., 2009), where u is selected from u' which is closest to the optimal control $u*$. Instead of selecting the closest control, we choose the control which actually brings the robot closest to the goal.

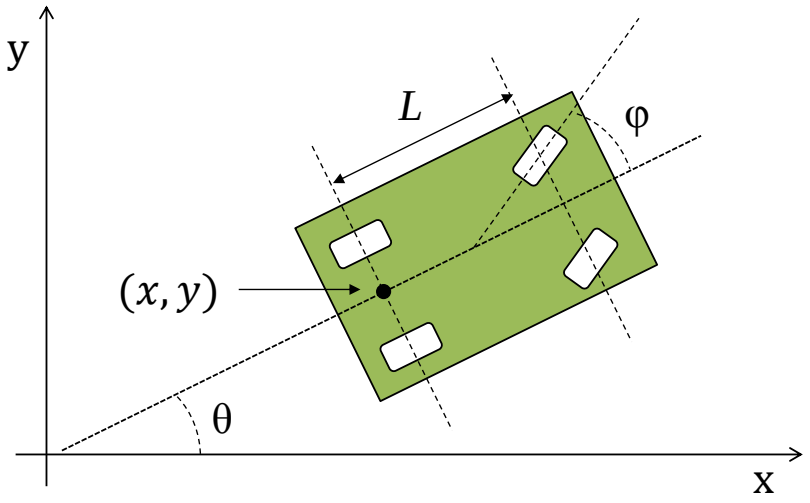


Figure 4.12: **Kinematic model of the robot** The robot is modeled as a simple car at position (x, y) and with orientation θ. ϕ is a steering angle and L is a wheelbase. Our robot has the wheelbase $L = 1m$.

When navigating, the robot uses BRVO to predict upcoming pedestrian motion and must avoid any steering inputs which (as determined by Equation 4.12) will collide with the predicted pedestrian positions. In essence, we assume a non-cooperative environment, where the pedestrians may not actively avoid collisions with the robot and the robot must assume 100% responsibility for collision avoidance (i.e., asymmetric behavior).

We use the Campus-1 (7 pedestrians), Campus-2 (11 pedestrians), and Campus-3 (18 pedestrians) data sequences from the **Campus** dataset (See Fig.4.3 (a)) to measure the performance of the robot navigation. The robot is given an initial position at one side of the walkway and is asked to move through the pedestrians to the opposite side. Given the importance of safety in pedestrian settings, if at any point the robot fails to find collision free trajectory which moves forward along the path, the robot will stop or turn back towards the start. Given this setup, we evaluate the percentage of times the robot is able to make it more than halfway through the pedestrian crossing without colliding with a pedestrian, or needed to stop or turn back.

We run experiments on all the three sequences, assuming a sensing error of $\pm$15cm noise in positional estimates, and a sampling rate of 2.5Hz to allow adequate time for any visual processing needed to detect pedestrians in the sensed area. We collect the mean of 30 runs for each sequence; the robot's initial position and goal position are randomly chosen for each run.

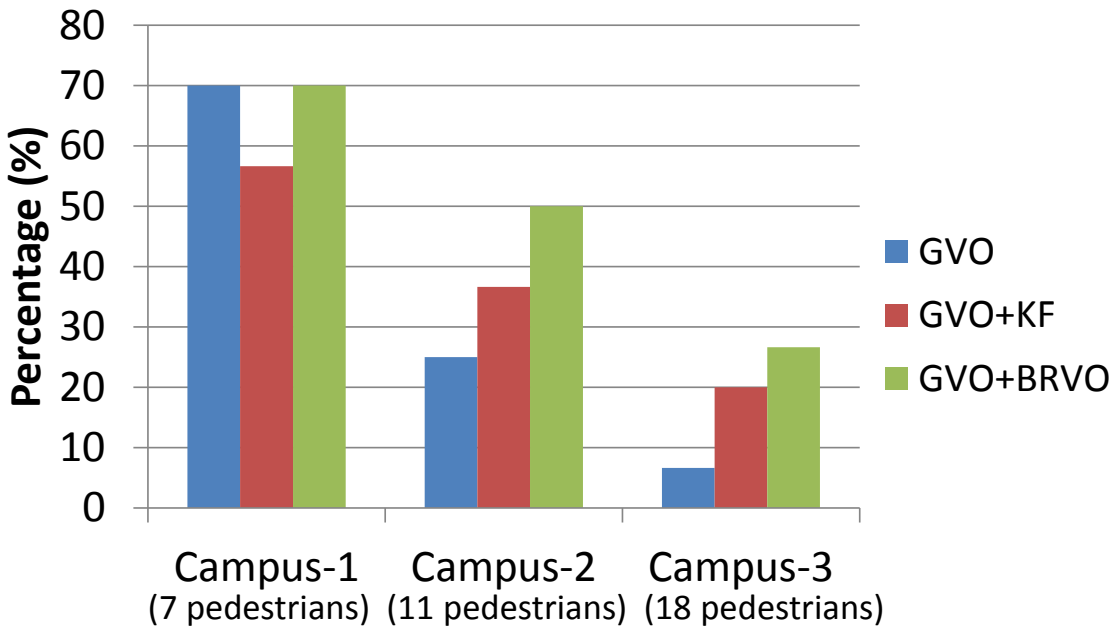


Figure 4.13: **Performance of the robot navigation** We measured the percentage of the trajectories in which the robot reached farther than halfway to the goal position without any collision during the data sequence, using only GVO (blue bars), GVO with KF (red bars), and GVO with BRVO (green bars), both with 15cm sensor noise. In many of these scenarios, using GVO for navigation often caused a robot to stop moving through the crowd to avoid collisions (the freezing robot problem). GVO-KF shows lower performance in very sparse scenario, but outperforms GVO-only method as the number of pedestrians increases. GVO-BRVO algorithm further improves the navigation, especially for more challenging scenarios with more pedestrians.

We compare the robot navigations in various combinations: GVO alone (using a constant-velocity model), GVO with GVO-KF, and GVO with BRVO. Fig. 4.13 shows the results with GVO only (blue bars), GVO-KF (red bars), and GVO-BRVO (green bars). As the scenarios get denser, the robots navigating with GVO alone tended to avoid collisions by staying still, displaying the same freezing robot problem as discussed by other researchers (see for example (Trautman and Krause, 2010)). In very sparse crowds, GVO + a linear method performs slightly better than GVO-KF, but the performance rapidly drops as the number of pedestrians increases. The GVO-BRVO algorithm improves the navigation more than GVO-KF, especially for more challenging scenarios. Compared to GVO-alone, GVO-BRVO more than doubled the task-completion rate.

We also performed the same experiments using the ground-truth data without any sensor noise added. In this case, GVO-BRVO achieved 23% and 26% better performance over GVO-only and GVO-KF, respectively.

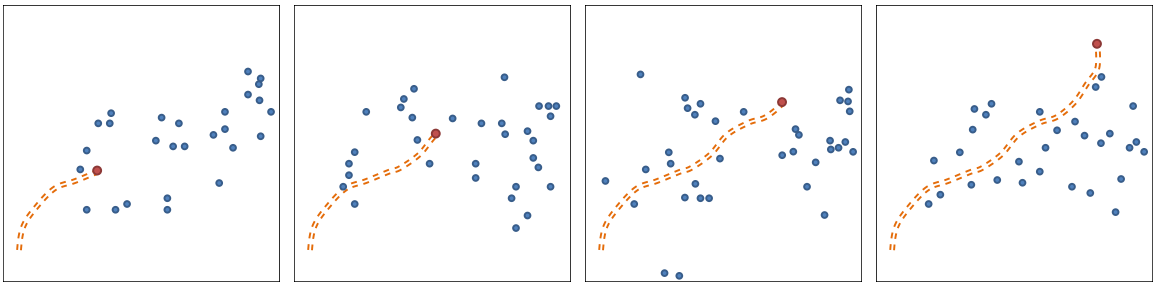


Figure 4.14: **Example robot trajectory** navigating through the crowd in **Students** dataset. Blue circles represent current pedestrian positions, red circles are the current position of the robot, and orange dotted lines are the previous positions of the robot.

Similar results were seen in other datasets. Fig.4.3 shows an example path of the robot navigating through pedestrian motion taken from the **Students** dataset (with a pedestrian density of .35 ppl/m^2). The result shown comes from an experimental setup with an even lower sampling rate (.6Hz) and using only 100 ensemble samples in EnKF. We believe that this low sampling rate, and low processing requirement, may be appropriate for many types of low-powered mobile robots. Even with these reduced computational requirements, we still achieved about a 50% task completion rate.

This series of experiments indicates that BRVO can improve planning in uncertain environments: environments with dynamic obstacles or with sensor limitations, including noisy or sparse sensor inputs. Though better prediction does not guarantee better navigation, and the freezing robot problem can still occur (Trautman and Krause, 2010), we believe that better prediction algorithms can improve the robotic navigation.

We also believe that BRVO can be combined with other robot navigation methods in a cooperative setup, since it improves the performance of RVO-based motion models in noisy-data situations, as discussed in (Trautman et al., 2013). More importantly, BRVO does not need prior knowledge about the scene, such as destinations or goal positions and motion priors. BRVO-GVO, however, requires static obstacles to compute robot navigation. Motion prediction using BRVO does not require knowledge about static obstacles, although static obstacles that strongly constrain the motion of the pedestrians may improve its accuracy (e.g., **Bottleneck** dataset). Instead of the **Bottleneck** dataset, our experiments are performed without specifying the static obstacles. These features of BRVO can be a considerable benefit for navigating robots or autonomous wheelchairs in real-world scenes.

4.7 Conclusion and Future Work

We have presented a method to predict pedestrian trajectories using agent-based velocity-space reasoning. The BRVO model that we have introduced is an online motion-prediction method that learns per-agent motion models. Even without prior knowledge about the environment, it performs better than offline approaches that do have prior knowledge. We have demonstrated the benefits of our approach on several datasets, showing our model's performance with varying amounts of sensor noise, interaction levels, and densities. Specifically, we have shown that our approach performs very well with noisy data and in low framerate scenarios. We also highlight BRVO's performance in dynamic scenarios with density (spatial) and speed (temporal) variation.

BRVO assumes no prior knowledge of the scenario which it will navigate; it is thus uniquely well-suited for mobile robots that may frequently encounter new obstacles and that must handle pedestrians entering and leaving the environment. We have shown that by learning an individualized motion model for each observed pedestrian, our online motion-prediction model can perform better than less responsive offline motion models. We also showed that our method can be integrated with recent local navigation techniques to improve task completion rates and reduce instances of the freezing robot problem.

BRVO has also been used for pedestrian tracking applications and shown to increase tracking performance in various scenarios (Bera and Manocha, 2014; Liu et al., 2014). In the future, we would like to explore a method for local, dynamic group behavior estimation, to further improve the performance of BRVO.

CHAPTER 5

Interactive Data-Driven Crowd Simulation using Time Varying Pedestrian Dynamics

5.1 Introduction

Crowd simulation has received considerable attention in multiple fields, including animation, games, robotics, computer vision, evacuation planning, and computer-aided design. In many applications, the goal is to generate trajectories of individual walking humans that align with real human movement and display emerging behaviors observed in real-world crowds. The most common approaches for simulating crowd movement use agent-based models that treat individuals as autonomous agents that can perceive the environment and make independent decisions about their behavior or movement. In previous chapters, we have presented various ways to model different aspects of human behaviors, including individual differences, dynamic behavior changes and physical interactions. However, using agent-based methods to simulate the subtle aspects of real-world crowd motions can be still challenging.

To address these issues, many data-driven or example-based crowd simulation algorithms (Lerner et al., 2007; Lee et al., 2007) have been proposed. Their emergence grows out of the increasing availability of real-world datasets of individual humans and crowds, driven in part by improvements in high-resolution cameras and motion-capture systems. Furthermore, advances in computer vision and tracking algorithms have made it possible to extract pedestrian trajectories, full-body movements, and pairwise interactions from these datasets, and to use them to generate simulations that match real-world human movements in both indoor and outdoor scenes. However, current implementations of these data-driven methods have several limitations. The accuracy of tracking methods varies with the number of pedestrians and crowd density. As a result, use of these methods is mostly limited to isolated pedestrians or sparse crowds. Many behavior learning algorithms require many training videos to learn the patterns offline and typically extract a fixed set of parameters that are used as a global characterization of pedestrian behavior or trajectories; thus, they may not be suited to capturing

the dynamic nature or time-varying behaviors of pedestrians. Finally, techniques utilizing full-body movements and pairwise interaction datasets are either limited to non-interactive applications or involve user editing.

In this chapter, we address some of these limitations. Our first goal is develop automatic and interactive approaches that can extract the movement patterns and motion dynamics of individuals from crowd videos. There is a large collection of crowd videos available on the internet and such approaches might make it easier to generate a large library of dynamic movement patterns for different real-world situations. Our second goal is to develop techniques that make it easier to use these movement patterns in slightly different environments or settings than those captured in the original video.

Main Results: We present an interactive approach to learning the characteristics of pedestrian dynamics from 2D trajectories extracted from videos. These characteristics consist of three components that correspond to movement patterns and motion dynamics: the entry point or starting position of each individual human in the scene, the movement flow used to estimate its preferred velocity or intermediate goal position, and the local collision avoidance technique. We show that these three components can generate crowd movements that are similar to those observed in the original videos and can also be used in slightly varying environments for interactive applications. As the individual humans in the original video change their speed or movement, each virtual pedestrian in the simulation can dynamically adapt its trajectory to interact with other pedestrians and obstacles in the scene.

We present statistical algorithms to estimate these characteristics using Bayesian inferences from noisy or sparse trajectory data. Furthermore, we present algorithms to combine these characteristics with other agent-based models, which allows us to adapt their movement to a new environment or a new situation.

1. Robustness with sparse and noisy data: Current algorithms for interactive tracking are limited to sparse or low-density crowds and result in noisy data. Our statistical approach is able to reliably extract the movement patterns and motion dynamics from such trajectories.

2. Interactive applications: Our approach is interactive and can compute dynamic patterns and time-varying pedestrian dynamics. As a result, we can easily update crowd movements as the

environment changes. We highlight their benefit for evaluating different layouts of obstacles in a scene.

3. Dense simulations: The characteristics of pedestrian dynamics can be used to add an arbitrary number of virtual pedestrians to a given environment that have a similar movement pattern as the original video. We combine the characteristics with density-dependent filters (Best et al., 2014) to ensure that the resulting speed/density relationships in the data-driven simulation are consistent with the Fundamental Diagrams.

4. Situation response: The characteristics of pedestrian dynamics can be combined with other agent-based models that can change each individual's behavior in response to an external event (such as to external stressors). We demonstrate its application to a panic situation.

We have implemented our approach and tested it on several indoor and outdoor scenarios with varying pedestrian movement patterns, using trajectory data from a variety of video datasets. The original videos of these scenes have tens of real-world pedestrians, and we are able to generate data-driven simulations with tens and even hundreds of pedestrians in slightly different environments or situations. We also present results from a user-study that compares our approach with other data-driven methods. Overall, we show that the characteristics of time-varying pedestrian dynamics can facilitate the broader use of data-driven crowd simulation methods.

Th rest of this chapter is organized as follows. Section 5.2 provides an overview of related work in crowd simulation and behavior learning. We introduce the terminology and present our interactive pedestrian dynamics–learning algorithm in Section 5.3. In Section 5.4, we use these characteristics for data-driven crowd simulation. Then, we describe our implementation and highlight its performance in various scenarios in Section 5.5. Finally, we analyze its performance and compare it with prior approaches in Section 5.6. We direct the readers to the project webpage (`http://gamma.cs.unc.edu/PDL/`) for the videos and the related tech. report (Kim et al., 2015a) and publications.

5.2 Related Work

In this section, we offer a brief overview of related work, including motion models, data-driven crowd simulation and behavior learning.

5.2.1 Crowd Simulation

Some of the commonly used techniques for simulating crowd behaviors and trajectory computation are derived from agent-based models, including rule-based methods, which use a set of behavioral rules to guide the behavior of each pedestrian (Reynolds, 1987; Pelechano et al., 2007). Another group of models include force-based methods that model interactions among pedestrians using attraction or repulsion forces (Helbing and Molnar, 1995); velocity-based (van den Berg et al., 2011; Karamouzas and Overmars, 2012) and vision-based approaches (Ondřej et al., 2010), which are useful for collision-free local navigation. Another set of algorithms is based on continuum techniques, which compute fields for pedestrians to follow based on continuum flows (Treuille et al., 2006) or fluid models (Narain et al., 2009). Crowd behaviors can also be simulated based on cognitive and decision-making models (Shao and Terzopoulos, 2005; Yu and Terzopoulos, 2007), ocean personality models (Durupinar et al., 2011), etc. Still other techniques have been proposed to handle large-scaled simulation and crowd rendering (Dobbyn et al., 2005; Yersin et al., 2009).

5.2.2 Data-driven Crowd Simulation

Data-driven methods use real-world motion specifications or captured data to generate the trajectory or behavior of each pedestrian. At a broad level, prior work in data-driven methods can be classified into offline methods (which involve preprocessing) and interactive algorithms.

Offline Methods: There is a large body of work on scalable multi-character motion synthesis based on *Motion Patches* (Lee et al., 2006) and its extensions that can be used to generate a dense crowd with multiple interacting characters (Yersin et al., 2009; Kim et al., 2012a; Shum et al., 2012). These methods can also model close interactions between pedestrians.

Other data-driven methods emphasize generating trajectory level behaviors using video data or recorded trajectories. For example, data extracted using semi-automatic trackers are used to generate group behaviors (Lee et al., 2007), and others extend these approaches by building formation models (Ju et al., 2010). The virtual scenes can be populated by copying and pasting small pieces of real-world crowd data (Li et al., 2012) or by using efficient data structures to represent sequences of motion in a large database for motion retrieval (Kapadia et al., 2013). A different class of data-driven algorithms use real-world crowd data to learn or optimize the motion-model parameters

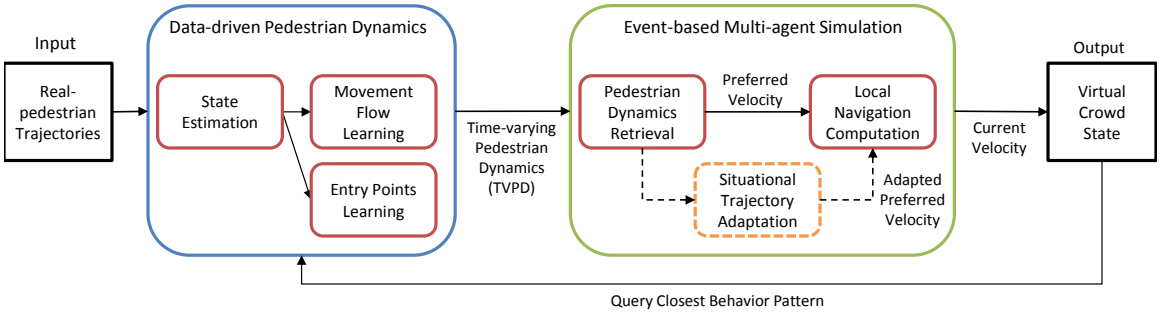


Figure 5.1: **Interactive data-driven simulation pipeline:** Our method takes extracted trajectories of real-world pedestrians as input. We use the Bayesian inference technique to estimate the most likely state of each pedestrian. Based on the estimated state, we learn time-varying behavior patterns. These behavior patterns are used as behavior rules for data-driven simulation and can also be combined with other multi-agent simulation algorithms. All these computations can be performed in tens of milliseconds.

for agent-based simulation algorithms. This class includes a density-based measure (Lerner et al., 2009), a similarity-based entropy metric (Guy et al., 2012b) to learn or evaluate parameters from a given scenario; offline optimization methods that use real-world trajectory data to compute the best parameters for simulated motion models (Wolinski et al., 2014; Berseth et al., 2014); and a data-driven framework that analyzes the quality and anomalies of crowd simulations by comparing them to given training data (Charalambous et al., 2014).

Interactive Methods: There is large body of computer vision literature on realtime pedestrian tracking from videos, and these can be used to generate 2D agent trajectories for data-driven simulation (Zhang et al., 2012; Bera et al., 2015). However, current methods are limited to generating 2D trajectories and cannot handle any changes in the environment or simulate different trajectory behaviors from those observed in the videos. There is a large body of work on interactive editing of crowd trajectories and animations (Kim et al., 2014a; Jordao et al., 2014; Kwon et al., 2008). These can be directly used on extracted trajectories or on the full body motion of animated characters to generate plausible simulations for different environments.

5.2.3 Video Based Crowd Analysis

There is extensive work in computer vision, multimedia, and robotics that analyzes the behavior and movement patterns in crowd videos, as surveyed in (Li et al., 2015; Borges et al., 2013), where the main objectives include human behavior understanding and recognition and crowd activity

recognition for detecting abnormal behaviors (Hu et al., 2004; Kim et al., 2015a). Many of these methods use a large number of training videos to learn the patterns offline (Zen and Ricci, 2011; Solmaz et al., 2012). Other methods utilize motion models to learn crowd behaviors (Mehran et al., 2009; Pellegrini et al., 2012) or machine learning methods (Zhou et al., 2012; Sun et al., 2013). In contrast, our goal is to develop improved techniques for interactive data-driven crowd simulation.

5.3 Time Varying Pedestrian Dynamics

In this section, we present our interactive algorithm that learns time-varying pedestrian dynamics from real-world pedestrian 2D trajectories. We assume that these trajectories are generated using standard tracking algorithms.

5.3.1 Pedestrian State

We first define specific terminology used in this chapter. We use the term *pedestrian* to refer to independent individuals or agents in the crowd. We use the notion of *state* to specify the trajectory and behavior characteristics of each pedestrian. The components used to define a state govern the fidelity and realism of the resulting crowd simulation. Because the input to our algorithm consist of 2D position trajectories, our state vector consists of the information that describes the pedestrian's movements on a 2D plane. We use the symbol $\mathbf{x} \in \mathbb{R}^6$ to refer to a pedestrian's state:

$$\mathbf{x} = [\mathbf{p}\ \mathbf{v}^c\ \mathbf{v}^{pref}]^{\mathbf{T}}, \tag{5.1}$$

where $\mathbf{p}$ is the pedestrian's position, $\mathbf{v}^c$ is its current velocity, and $\mathbf{v}^{pref}$ is the preferred velocity on a 2D plane. The preferred velocity is the optimal velocity that a pedestrian would take to achieve its intermediate goal if there were no other pedestrians or obstacles in the scene. In practice, $\mathbf{v}^{pref}$ tends to be different from $\mathbf{v}^c$ for a given pedestrian. We use the symbol $\mathbf{S}$ to denote the current state of the environment, which corresponds to the state of all other pedestrians and the current position of the obstacles in the scene. The state of the crowd, which consists of individual pedestrians, is a union of the set of each pedestrian's state $\mathbf{X} = \bigcup_i \mathbf{x_i}$, where subscript i denotes the i^{th} pedestrian. Our state formulation does not include any full body or gesture information. Moreover, we do not explicitly model or capture pairwise interactions between pedestrians. However, the difference between $\mathbf{v}^{pref}$

and $\mathbf{v}^c$ provides partial information about the local interactions between a pedestrian and the rest of the environment.

5.3.2 Pedestrian Dynamics

Pedestrian dynamics consist of those factors that govern pedestrians' trajectory behaviors, i.e., the factors that change the state of the pedestrians. We model pedestrian dynamics using three components: starting position or entry point; movement flow; and the local collision-free navigation rule. Formally, we represent the characteristics of these dynamics for each pedestrian with a vector-valued function, $f()$, with an initial value determined by the function, $E()$:

$$\mathbf{x}^{t+1} = f(t, \mathbf{x}^t) = [P(\mathbf{x}^t)\ I(\mathbf{x}^t)\ G(t, \mathbf{x}^t)]^{\mathbf{T}}; \quad \mathbf{x}^0 = E(t^0). \tag{5.2}$$

For each pedestrian in the crowd, the function $G : \mathbb{R} \times \mathbb{R}^6 \times \mathbb{S} \rightarrow \mathbb{R}^2$ maps time t, current state of the pedestrian $\mathbf{x} \in \mathbf{X}$, and current state of the simulation environment $\mathbf{S} \in \mathbb{S}$ to a preferred velocity $\mathbf{v}^{pref}$. Function $I : \mathbb{R}^6 \times \mathbb{S} \rightarrow \mathbb{R}^2$ computes the interactions with other pedestrians and obstacles in the environment and is used to compute the collision-free current velocity $\mathbf{v}^c$ for local navigation. The function $P : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ computes the position, given $\mathbf{v}^c$; $E : \mathbb{R} \rightarrow \mathbb{R}^2$ computes the initial position for time t_0, which is the time at which a particular pedestrian enters the environment. The three components of the pedestrian dynamics – entry point, movement flow, and local collision-free navigation – can be mapped to the functions $E()$, $G()$ and $I()$, respectively. We learn $E()$ and $G()$ from the 2D trajectory data. The local collision-free navigation rule $I()$ can be chosen by the data-driven algorithm.

We refer to our interactive method as learning time-varying pedestrian dynamics (TVPD). Fig. 5.1 gives an overview of our approach, including computation of TVPD and using them for crowd simulation. The input to our method consists of the trajectories extracted from a sensor. The trajectories are time-series observations of the positions of each pedestrian in a 2D plane. The output TVPD consists of entry point distributions and movement flows learned from the trajectory data. Notably, our approach is interactive and operates based on current and recent states; in other words, it does not require future knowledge of an entire data sequence and does not have to re-perform offline training steps whenever new real-world pedestrian trajectory data is acquired or generated. As a

result, our approach can effectively capture local and/or individual variations and the characteristics of time-varying trajectory behaviors. We use TVPD for data-driven crowd simulation in Section 4.

Algorithm 2: Crowd Simulation

Input: Observed position of each real agent $\mathbf{z}_i^t, i \in R$, Simulated state of each virtual agent $\mathbf{x}_i^t, i \in V$, Pedestrian local navigation model I, Composed situational trajectory adaption function J of n number of separate situational trajectory adaption function modules $J = J_1 \circ J_2 \circ \cdots \circ J_{n-1} \circ J_n$, $J(x) = x$ if $n = 0$, current time step t, last frame of the real-world data t_{end}, size of the time window w for TVPD

Output: State of virtual crowds $\mathbf{X}$

if $t \leq t_{end}$ **then**
  // State Estimation
  // 1. TVPD computation
  foreach $i \in R$ **do**
    Compute $\mathbf{x}_i^t$ from $\mathbf{z}_i^t$;
  // Update TVPD
  if $t\%w == 0$ **then**
    foreach $i \in R$ **do**
      // compute pedestrian dynamics feature $\mathbf{b}_i$ and add to the closest behavior cluster B_k
      Compute $\mathbf{b}_i$ from $\mathbf{x}_i^t$ and $\mathbf{x}_i^{t-w}$;
      $\mathbf{B}_k = \{\mathbf{b}_i : \mathrm{argmin}_k \, dist(\mathbf{b}_i, \mu_k)$;
    // Update Clusters
    while *not converged* **do**
      $\mathbf{B}_k = \{\mathbf{b}_i : dist(\mathbf{b}_i, \mu_k) \leq dist(\mathbf{b}_i, \mu_l) \forall l, 1 \leq l \leq K\}$;
      $\mu_k = \frac{1}{|\mathbf{B}_k|} \sum_{\mathbf{b}_i \in B_k} \mathbf{b}_j$;
// 2. Virtual Crowd State Computation
foreach $i \in V$ **do**
  // Pedestrian Dynamics
  Compute $\mathbf{b}_i$ from $\mathbf{x}_i^t$ and $\mathbf{x}_i^{t-w}$;
  $\mathbf{x}_i^t.\mathbf{v}^{pref}$=QueryPrefVelocityFromPDL($\mathbf{b}_i$);
  // Situational Trajectory Adaption
  $\mathbf{x}_i^t.\mathbf{v}^{pref}$=J($\mathbf{x}_i^t$);
// Local Collision Avoidance
$\mathbf{X}_V = \mathbf{x}_i^t | i \in V$;
$\mathbf{X}_V^{t+1} = I(X_V)$;

5.3.3 State Estimation

The trajectories extracted from a real-world video tend to be noisy and may have incomplete tracks (Enzweiler and Gavrila, 2009); thus, we use Bayesian-inference technique presented in Chapter 4 to compensate for any errors and to compute the state of each pedestrian.

At each time step, the observation of a pedestrian computed by a tracking algorithm is the position of each pedestrian on a 2D plane, denoted as $\mathbf{z}^t \in \mathbb{R}^2$. The observation function $h()$ provides $\mathbf{z}^t$ of each pedestrian's true state $\hat{\mathbf{x}}^{\mathbf{t}}$ with sensor error $\mathbf{r} \in \mathbb{R}^2$, which is assumed to follow a zero-mean Gaussian distribution with covariance Σ_r:

$$z^t = h(\hat{\mathbf{x}}^t) + \mathbf{r}, \mathbf{r} \sim N(0, \Sigma_r). \tag{5.3}$$

$h()$ can be replaced with any tracking algorithms or synthetic algorithms that provides the trajectory of each pedestrian.

The state-transition model $f()$ is an approximation of true real-world crowd dynamics with prediction error $\mathbf{q} \in \mathbb{R}^6$, which is represented as a zero-mean Gaussian distribution with covariance Σ_q:

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t) + \mathbf{q},\ \mathbf{q} \sim N(0, \Sigma_q). \tag{5.4}$$

We can use any local navigation algorithm or motion model for function $f()$, such as social forces, Boids, or velocity obstacles. The motion model computes the local collision-free paths for the pedestrians in the scene.

We use an Ensemble Kalman Filter (EnKF) and Expectation Maximization (EM) with the observation model $h()$ and the state transition model $f()$ to estimate the most likely state $\mathbf{x}$ of each pedestrian. EnKF uses an ensemble of discrete samples assumed to follow a Gaussian distribution to represent the distribution of the potential states. EnKF is able to provide state estimation for a non-linear state-transition model. During the prediction step, EnKF predicts the next state based on the transition model and Σ_q. When a new observation is available, Σ_q is updated based on the difference between the observation and the prediction, which is used to compute the state of the pedestrian. In addition, we run the EM step to compute the covariance matrix Σ_q to maximize the

Figure 5.2: **Pedestrian Dynamics Learning:** A one-frame example from the **zara01** dataset: (a) input consists of the manually-annotated trajectories (green) from the video; (b) probabilistic distributions of entry points at one frame, computed using the Gaussian Mixture Model (shown as elliptical regions); and (c) movement flows grouped by the characteristics of pedestrian dynamics, in which each grouping is represented by the same color.

likelihood of the state estimation. For more details about EM-based state computation, please see the supplemental materials.

5.3.4 Dynamic Movement Flow Learning

We compute the movement features, which are used as descriptors for local pedestrian movement. These movement features are grouped together and form a cluster of a movement flow.

Movement Feature The movement features describe the characteristics of the trajectory behavior at a certain position at time frame t. The characteristics include the movement of the agent during the past w frames, which we call *time window*, and the intended direction of the movement (preferred velocity) at this position.

The movement feature vector is represented as a six-dimensional vector:

$$\mathbf{b} = [\mathbf{p}\ \mathbf{v}^{avg}\ \mathbf{v}^{pref}]^T, \tag{5.5}$$

where $\mathbf{p}$, $\mathbf{v}^{avg}$, and $\mathbf{v}^{pref}$ are each two-dimensional vectors representing the current position, average velocity during past w frames, and estimated preferred velocity computed as part of state estimation, respectively. $\mathbf{v}^{avg}$ can be computed from $(\mathbf{p}^t - \mathbf{p}^{t-w})/w * dt$, where dt is the time step.

The duration of the time window, w, can be set based on the characteristics of a scene. Small time windows are good at capturing details in dynamically changing scenes with many rapid velocity changes, which are caused by some pedestrians moving quickly. Larger time windows, which tend to smooth out abrupt changes in motion, are more suitable for scenes that have little change in pedestrians' movement. For our results, we used 0.5 to 1 second of frames to set the value of w.

Movement Flow Clustering

At every w steps, we compute new behavior features for each agent in the scene, using Equation 5.5. We group similar features and find K most common behavior patterns, which we call *movement flow clusters*. We use recently observed behavior features to learn the time-varying movement flow.

We use the k-means data clustering algorithm to classify these features into K movement flow clusters. K and N^f are user-defined values that represent the total number of the clusters and the total number of collected behavior features, respectively, and $K \leq N^f$. A set of movement-flow clusters $B = \{B_1, B_2, ..., B_K\}$ is computed as follows:

$$\underset{B}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{b_i \in B_k} dist(b_i, \mu_k), \tag{5.6}$$

where b_i is a movement feature vector, μ_k is a centroid of each flow cluster, and $dist(b_i, \mu_k)$ is a distance measure between the arguments. In our case, the distance between two feature vectors is computed as

$$\begin{aligned} dist(b_i, b_j) = c_1 \left\| \mathbf{p}_i - \mathbf{p}_j \right\| \\ + c_2 \left\| (\mathbf{p}_i - \mathbf{v}_i^{avg} w\ dt) - (\mathbf{p}_j - \mathbf{v}_j^{avg} w\ dt) \right\| \\ + c_3 \left\| (\mathbf{p}_i + \mathbf{v}_i^{pref} w\ dt) - (\mathbf{p}_j - \mathbf{v}_j^{pref} w\ dt) \right\|, \end{aligned} \tag{5.7}$$

which corresponds to the weighted sum of the distance among three points: current positions, previous positions and estimated future positions (which are extrapolated using v^{pref}, c_1, c_2, and c_3 as the weight values). Comparing the distance between the positions rather than mixing the points and the vectors eliminates the need to normalize or standardize the data.

Each movement-flow cluster contains adjacent features that have similar average velocity and preferred velocity (See Figure 5.2 (c)). The pseudo-code of the overall pedestrian movement flow–learning algorithm is given in the appendix (See Algorithm 2).

5.3.5 Entry-Points Learning

Entry points are a component of pedestrian dynamics we want to learn to estimate when real pedestrians enter the scene. These starting positions and timings for each agent are very important and govern their overall trajectory. We use a multivariate Gaussian mixture model to learn the time-varying distribution of entry points, which will be used as the initial position $\mathbf{x}^0$ for a newly added pedestrian in a data-driven crowd simulation. We define $E()$ as the function that provides a position sampled from the learned distributions. The Gaussian distribution works better for non-spherical distribution; the distribution of entry points, which are scattered near the scene's boundary and often correspond to long elliptical regions, are frequently non-spherical (see Figure 5.2 (b)).

We assume that the distribution of entry points, e, that the function $E()$ samples from, is a mixture of J components and that each of the components is a multivariate Gaussian distribution of a two-dimensional random variable, $\mathbf{p}$, with a set of parameters $\Theta = (\alpha_1, \cdots, \alpha_J, \theta_1, \cdots, \theta_J)$:

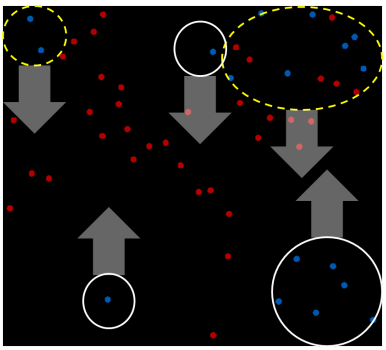
$$e(\mathbf{p}|\Theta) = \sum_{j=1}^{J} \alpha_j e_j(\mathbf{p}|\mu_j, \theta_j), \tag{5.8}$$

$$e_j(\mathbf{p}; \theta_j) = \frac{1}{2\pi|\Sigma_j|^{1/2}} exp(-\frac{1}{2}(\mathbf{p} - \mu_j)^T \Sigma_j^{-1}(\mathbf{p} - \mu_j)). \tag{5.9}$$

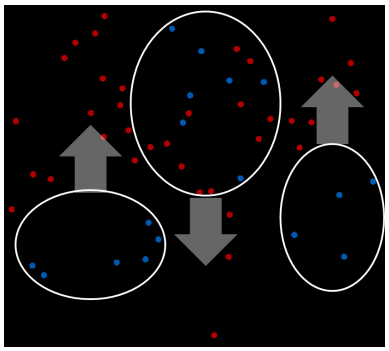
Each component e_j is a Gaussian distribution given by the parameters $\theta_j = (\mu_j, \Sigma_j)$, where μ_j is mean of the component j and Σ_j is a 2×2 covariance matrix. α_j is a mixture weight, which is the probability of a point $\mathbf{p}$ that belongs to the component j. $\alpha_j \in [0, 1]$ for all i and sum of α_j's are constrained to 1 ($1 = \sum_{j=1}^{J} \alpha_j$). From an initial guess of the parameters θ_j, we perform EM to learn these parameters $\theta_j = (\mu_j, \Sigma_j)$ from the given entry points collected from the real pedestrian trajectories. More details are provided in the supplemental material.

The entry point distribution is updated whenever we have a new observation of a pedestrian entering near the boundary of the scene (i.e., the starting positions of a trajectory). We use only the recent N^e observations of entry positions from trajectories and discard old observations. A large value for N^e can capture the global distribution of entry points, whereas a smaller value for N^e can better capture the dynamic changes of the distribution (See Appendix for further discussion). Although we update the model frequently, we can exploit the locality in distributions because the new

(a) Original Video



(b) Without TVPD



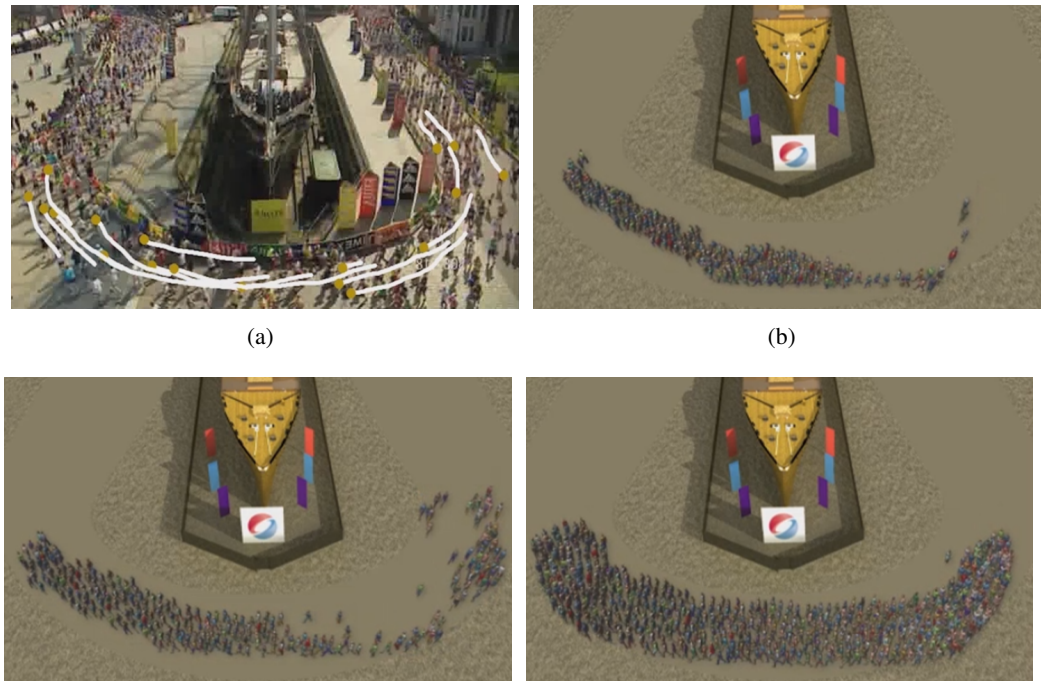
(c) With TVPD

Figure 5.3: **Manko scenario:** We highlight the benefits of entry point and movement flow learning. (a) A frame from a Manko video, which shows different flows corresponding to lane formation (shown with white arrows); (b) and (c) We compute collision-free trajectories of virtual pedestrians. For (b), we use random entry points for virtual pedestrians and goal positions on the opposite side of the street. White circles highlight the virtual pedestrians who are following the same movement flow as neighboring real pedestrians. For (c), we use TVPD (entry point distribution and movement flow learning) to generate virtual pedestrians' movement. The virtual agent's follow the lane formation, as observed in the original video.

distribution is evolving from the previous distribution. We use the previous parameters and choose cluster j, which satisfies $\text{argmin}_j ||\mathbf{p} - \mu_j||$, as our initial guess for the new distributions.

5.4 Data-driven Crowd Simulation

In this section, we use TVPD to generate data-driven crowd simulations. In particular, TVPD are used to compute the starting position and the steering or preferred velocity of each pedestrian in the scene, which correspond to $E()$ and $G()$, respectively. During the simulation, the local collision avoidance rule, $I()$, is employed to perform collision-free navigation and compute the actual velocity of each pedestrian. Any of the local navigation agent-based models that compute the interaction



(a) (b)

(c) (d)

Figure 5.4: **Marathon Scenario:** We compare the performance of different algorithms used to generate the trajectories of 500 pedestrians in the Marathon scenario: (a) The original video frame with 18 extracted trajectories (white); (b) A multi-agent simulation using five intermediate goal positions along the track; (c) We run the same simulation with optimized parameters using an offline optimization method; and (d) Instead of the intermediate goals and/or optimized parameters, we use only TVPD. Notably, TVPD captures the pedestrian flow in the original video.

between a pedestrian and the rest of the environment, i.e., other pedestrians and obstacles, can be used. Some widely used choices include social forces (Helbing and Molnar, 1995; Pelechano et al., 2007), vision-based choices (Ondřej et al., 2010), hybrid method choices (Narain et al., 2009), velocity-obstacle-based choices (van den Berg et al., 2011; Karamouzas and Overmars, 2012), and rule-based choices (Reynolds, 1987). This decoupled usage of different components of TVPD enables us to use the original movement patterns in different or varying environments compared with those that were captured in the original video.

5.4.1 Pedestrian Dynamics Retrieval

Given a scene description, new virtual agents can be added to the scene at any time during the simulation. The initial position of each newly added virtual agent is sampled from the entry point distributions, which are modeled as a mixture of Gaussian distributions (see Equation 5.8). First, we select one of the Gaussian distributions, e_j, based on weight, α_j, and sample a point from the

chosen distribution (Equation 5.9). Because we compute TVPD adaptively to the new observations, the parameters of the mixture model e that is used for entry point distribution also vary over time.

After a virtual agent is added to the scenario, we must compute its trajectory beginning from the entry point location. We use the notion of *preferred velocity* that is employed to specify the intermediate goal position for a given pedestrian in agent-based models (see Section 3.1). We use TVPD to compute the preferred velocity of each agent. At runtime, we compute this velocity by querying to which movement cluster that pedestrian belong to (set B in Equation 5.6). More specifically, we query the closest cluster using the behavior feature $\mathbf{b}_i$ of an agent i at time t, computed from its state estimation $\mathbf{x}^t$ and $\mathbf{x}^{t-w}$:

$$\underset{k}{\operatorname{argmin}}\, dist(\mathbf{b}_i, \mu_k), \tag{5.10}$$

where k is the label of the closest behavior cluster of the agent and μ_k is the centroid feature of the cluster k. We use the preferred velocity of the centroid feature to update the preferred velocity of the agent. Because TVPD captures time-varying characteristics, the generated virtual pedestrian trajectories exhibit similar spatio-temporal characteristics in terms of the resulting trajectories.

5.4.1.1 Adapting to Different Environments and Situations

TVPD can be combined with other agent-based models that can change the trajectory or behavior of an agent depending on the environment or the situation. We refer to these as *situational trajectory adaption modules*; these modules are used to generate variations in the crowd simulation. We define the situational trajectory adaption function A as a composition of N^a separate situational trajectory adaption function modules:

$$A = A_1 \circ A_2 \circ \cdots \circ A_{N^a-1} \circ A_{N^a}, \tag{5.11}$$

where $A(x) = x$ (identity function), if $N^a = 0$. The situational trajectory adaption modules A_i act as a filtering method on the preferred velocity of a pedestrian based on its state and on the state of the environment. The output of a situational trajectory adaption A_i is an updated, adapted preferred velocity of the agent. As a result, our approach can be combined with any agent-based model (A_i)

that incorporates the change in behavior or the trajectory by updating the preferred velocity. In our current system, we have integrated TVPD with a Density-dependent Filtering (DDF) module and General Adaptation Syndrome (GAS) behavior modules.

Density-dependent Filters Density-dependent filters (DDF) are used to ensure that the trajectories of pedestrians in a dense crowd satisfy the speed/density relationships that are typically expressed using the Fundamental Diagram (Best et al., 2014). The density is often defined in terms of the number of pedestrians per square meter. In many data-driven simulations, we populate the scene with a large number of virtual pedestrians. It is important that their movement or density/velocity flow resembles that of real-world crowds, which is captured by the Fundamental Diagram. The DDF computes a new steering angle and the preferred speed of a pedestrian based on the crowd density in its neighborhood.

We can easily combine TVPD with DDF for dense crowd simulation. A pedestrian with radius r chooses θ to minimize the distance to its intermediate goal $\mathbf{g}$ during the time period τ

$$\arg\min_{\theta} \| \mathbf{g} - (\mathbf{p} + \mathbf{v}^{FD_\theta})\tau \|, \tag{5.12}$$

where $\mathbf{v}^{FD_\theta}$ represents the Fundamental Diagram adherent velocity (Best et al., 2014), which can be computed as:

$$\mathbf{v}^{FD_\theta} = \frac{\mathbf{v}^{\theta}}{s^{pref}}(\rho^{\theta}/2r), \tag{5.13}$$

where $\mathbf{v}^{\theta}$ is the input preferred velocity, s^{pref} is natural walking speed of the pedestrian, and ρ^{θ} is the density around the pedestrian. Finally, $\mathbf{v}_i^{FD_\theta}$ is the new preferred velocity of that pedestrian that is used as an input to the local navigation module.

Situational Trajectory Adaptation

We use a variation of a stressor-based behavior model proposed in Chapter 2 to generate heterogeneous behaviors and dynamic behavior changes in response to an environment or a situation. In Chapter 2, the behavior changes are modeled by updating various simulation parameters (e.g., personal radius, speed, number of neighbors, etc.) of an agent who is affected by one or more stressors of many kinds, such as time pressure, area stressors, positional stressors, interpersonal stressors, etc. We approximate this model to use with various multi-agent methods in general.

We assume that a pedestrian is experiencing a perceived stress with a value of ψ. The perceived intensity of the stressor, ψ, can be different for each pedestrian, depending on the type of the stressor and/or the distance between the pedestrian and the stressor. Our goal is to compute a stress response for an agent, denoted as S, that follows the General Adaptation Syndrome (GAS) theory. We use an approximated GAS model that is formulated as follows:

$$\frac{dS}{dt} = \begin{cases} \alpha & \text{if } \psi > S \\ \{-\alpha \leq \frac{d\psi}{dt} \leq \alpha\} & \text{if } \psi = S \\ -\alpha & \text{if } \psi < S \end{cases} \tag{5.14}$$

where S is capped at a maximum rate α and at a maximum amount β. We also assume that pedestrians tend to move away from the stressor. Let us denote the vector with the direction away from the stressor as v^{st}. The new adjusted preferred velocity v^{adj} is represented as a weighted sum of v^{st} and the agent's preferred velocity v^{pref}, with preferred speed s^{pref} scaled and based on the magnitude of S:

$$v^{adj} = S(1 + s^{pref})((1 - \frac{S}{\beta})\|v^{pref}\| + \frac{S}{\beta}\|v^{st}\|). \tag{5.15}$$

Finally, $\mathbf{v}^{adj}$ is the new preferred velocity of the pedestrian who is affected by the stressor.

5.5 Results

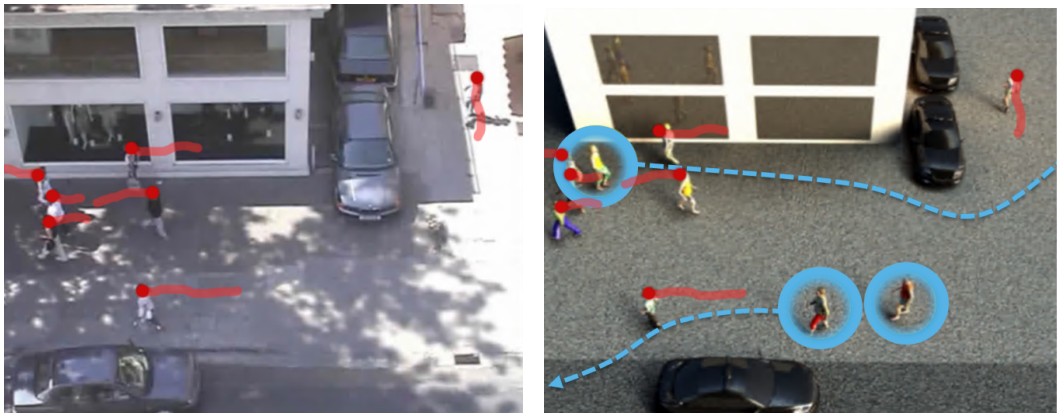
In this section, we describe our implementation and highlight its performance on different scenarios. Our system runs at interactive rates on a desktop machine with a 3.4 GHz Intel i7 processor and 8GB RAM. For state estimation, we use velocity-based reasoning as the state transition model, $f()$. For collision-avoidance computation, we use a publicly available library (van den Berg et al., 2011), and we use different real pedestrian tracking datasets corresponding to indoor and outdoor environments as the input for the TVPD computation algorithm. These datasets are generated using manual tracking, online multiple-person tracker, KLT tracker, synthetic data and 3D range sensor tracking (Lerner et al., 2009; Zhou et al., 2012; Brscic et al., 2013). Our approach makes no assumptions in connection with the underlying tracking algorithm that is used to generate these datasets. Table 5.1 presents more details on these datasets along with the number of tracked pedestrians and the number of virtual pedestrians in the data-driven simulation. Our algorithms

compute collision-free trajectories for the virtual pedestrians. We modeled the objects in these scenes using Maya and rendered the results using GOLAEM, a commercially available crowd rendering platform (see video for details).

5.5.1 Scenarios

We used different indoor and outdoor scenarios to highlight the performance of our crowd simulation algorithm.

Street: This scenario illustrates the basic notion of our data-driven algorithm. We use smooth, manually annotated trajectories of 147 pedestrians (Lerner et al., 2007). In the resulting simulation, we compute the trajectories of pedestrians whose entry points and movement patterns are similar to those of the tracked pedestrians. In this case, TVPD captures both time-varying entry-point distributions and movement patterns and show that the virtual pedestrians have the same characteristics (See Figure 5.5 in Appendix) .



(a) (b)

Figure 5.5: (a) A frame from a video of pedestrians in a street with extracted trajectories (shown in red); (b) Our simulation algorithm computes collision-free trajectories of virtual pedestrians (shown in blue) in the 3D virtual environment, which have the same movement flows as extracted trajectories (red)

Manko: We highlight the similar behaviors of the virtual pedestrians compared with those of tracked pedestrians. Furthermore, we generate crowd trajectories and behaviors using different options: (a) crowd simulation with a similar scene-setup, but the virtual pedestrian trajectories are generated without TVPD, and we instead use random positions for the entry points; (b) computing the entry points and movement flow for virtual pedestrians using TVPD. The benefits of our learning

(a) (b)

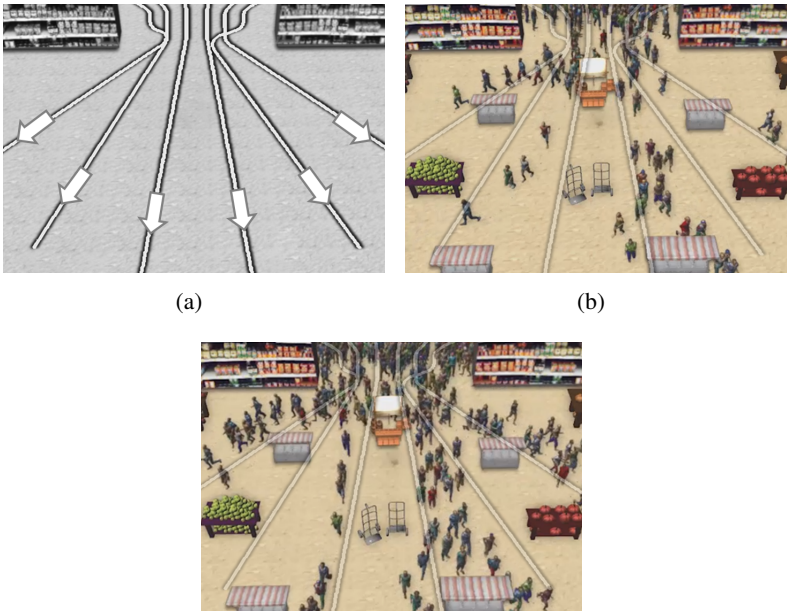
Figure 5.6: **ATC Shopping Mall Scenario:** The real pedestrian trajectories are computed using a 3D range sensor. (a) We show the flows highlighted with different colors in the original data. (b) We populate the scenario with 207 virtual pedestrians. The virtual pedestrians exhibit the characteristics observed in the original data.

algorithm are shown in the video (see Figures 5.3) as the virtual pedestrian follow the lanes in the original video. In contrast, virtual agents simulated using method (a) often result in collisions when going against the flow.

Marathon: We show a *structured scenario* in which pedestrian movement is somewhat uniform. In the marathon scenario, the runners follow an arc-shaped route (see Figure 5.4). Although the motion pattern looks simple, it is hard to model this scenario using one fixed set of entry points and goal positions, which is commonly used in agent-based models. Our TVPD algorithm eliminates manually adding and adjusting entry/goal points and automatically computes the preferred velocity over different time periods. We use only 18 trajectories from the original video and employ them to compute TVPD. These characteristics are used to compute the trajectories of 500 virtual pedestrians as part of the marathon. We further compare our approach with other agent-based simulation algorithms that use many set of intermediate goals and offline parameter learning instead of TVPD (Figures 5.4 (b) and (c)). The different simulation results are shown in the video.

Black Friday Shopping Mall: In this scenario, we show that our TVPD approach can be combined with other agent-based methods that simulate various aspects of human behavior, as described in Section 4.1.1. We take a shopping mall scene generated from given trajectories and vary the pedestrian density. We combine TVPD with DDFs (Best et al., 2014) and highlight the results in the video (See Figure 5.7). We also highlight the benefits of *interactive computation* of TVPD.

In particular, we add new obstacles during the simulation, and the data-driven crowd simulation algorithm can vary the trajectories to avoid collisions with the obstacles.



(a) (b)

(c)

Figure 5.7: **Black Friday Shopping Mall scenario:** (a) We use six simple trajectories generated in an empty space. (b) and (c) We create a new layout by adding obstacles interactively, and also increase the number of pedestrians in the scenario. (b) has 271 pedestrians, (c) has the twice higher number of pedestrians generated at the same rate.

Explosion: We combine our method with GAS model *dynamic behavior changes and situational trajectory adaptation*. This scenario highlights how the trajectories and behaviors of the pedestrians change in response to an explosion. In this scenario, the pedestrians initially follow the TVPD computed from the extracted trajectories. We represent an explosion as an external event and model that using a stressor. We use the stressor-based behavior model 2 to compute the new preferred velocity for each pedestrian in the scene. We model a stressor from an explosion as a positional stressor and change the behavior of the virtual pedestrians based on the amount of stress that the explosion generates (See Figure 5.8).

Train Station: We demonstrate the performance of TVPD on noisy, tracked data. For this scenario, we use tracklets generated by a KLT tracking algorithm. Instead of tracking each individual agent, the KLT tracker computes particles that belong to one or more tracked pedestrians in the video. The length of the trajectories range from a few to tens of frames. The tracklets are noisy, and we

(a) (b)

Figure 5.8: **Crossing Explosion Scenario:** (a) Initially, virtual pedestrians follow the TVPD computed from the extracted trajectories. (b) The user places an explosion during the simulation. The pedestrians who perceive the explosion begin running away from it in different directions.

compute the TVPD from these tracklets, using them to generate the trajectories of virtual pedestrians (see Figure 1.5 and the supplementary video). Our approach can produce the same movement flow as the original video. Moreover, we also highlight gradual density changes in the same environment and/or when we change the layout.

ATC Mall: This scenario demonstrates the performance of our algorithm with different sensor data. In this case, we use the 50 trajectories extracted in a shopping mall scenario using 3D range sensors, which comprises a sparse dataset with noisy trajectories. We compute the TVPD to generate the trajectories of 207 virtual pedestrians (see Figure 5.6 in Appendix) that exhibit similar time-varying movement patterns.

Scenario	Sensor	# Tracked Peds.	# Virtual Peds.	# Static Obst.	# Input Frames	Avg. time TVPDL	Avg. time Traj. Comp.
Manko	Online Tracking	42	70	0	373	0.075	5.63e-05
Marathon	Online Tracking	18	500	33	450	0.04	1.56E-03
Explosion	Online Tracking	19	110	0	238	0.03	2.56E-04
Street	Manual Tracking	147	167	0	9014	0.012	3.27E-06
Train Station	KLT (Tracklets)	200	200-943	37-47	999	0.05	4.93E-04
ATC Mall	3D Range Sensors	50	207	38	7199	0.023	2.97E-03
BlackFriday	Synthetic Data	6	271-1000	20-28	109	8.58E-03	2.75E-05

Table 5.1: Performance on a single core for different scenarios. We highlight the number of real and virtual pedestrians, the number of static obstacles, the number of frames of extracted trajectories and the time (in seconds) spent in different stages of our algorithm. Our learning and trajectory computation algorithms can be used for interactive crowd simulations.

5.6 Analysis and Comparisons

In this section, we analyze the performance of our approach and compare it with previous methods. The TVPD were able to capture the movement patterns and motion dynamics from the extracted trajectories. We have demonstrated the benefit of our pedestrian dynamics learning algorithm on several challenging benchmarks, including many structured and unstructured benchmarks. Furthermore, we demonstrate its benefits on different scenarios: robust to noisy and varying sensor data (ATC Mall and Train Station scenarios), interactive computations (Black Friday Shopping Mall scenario), handling structured environments (Marathon scenario), adapting to a situation (Explosion scenario), high-density simulation (Train Station scenario).

5.6.1 Comparisons

Many prior agent-based techniques in the literature are known to compute collision-free trajectories of virtual pedestrians. The simplest methods use random points to compute entry points and goal destinations and compute collision-free paths for virtual pedestrians using any collision avoidance algorithm (e.g., social forces or reciprocal velocity obstacles). However, the resulting trajectories may not exhibit the same movement patterns or trajectories as real pedestrians, as observed in some benchmarks (see Figure 5.4(c)).

Parameter Learning: Recently, many offline optimization techniques have been proposed to learn the optimal motion parameters for parameterized multi-agent simulation algorithms using real-world trajectories (Pettré et al., 2009; Wolinski et al., 2014; Berseth et al., 2014). These techniques improve the trajectory behaviors of virtual pedestrians with respect to resembling the tracked pedestrians. However, we still need good techniques to estimate the entry point, movement flow and/or the goal position for each agent (see Figure 5.4(d)). In many cases, entry points and destinations are specified or selected from manually pre-annotated regions in the scene (Pellegrini et al., 2012; Zhou et al., 2012). These parameter learning methods can be used to improve the interaction model $I()$ of our pedestrian dynamics model.

Offline Behavior Learning : Most prior work in computer vision and robot-human interactions uses offline learning methods from the training data to compute a fixed set of parameters that correspond to a global characterization (Zhou et al., 2012; Ikeda et al., 2012). However, these

methods may not work when there are large variations in individual behaviors or trajectories. Unlike prior offline methods, our approach does not learn fixed sets of destination points or sub-goals and can capture time-varying movement characteristics (see Figure 5.4(e)). Many offline methods based on multi-character motion synthesis (Lee et al., 2006; Yersin et al., 2009) can generate more realistic crowd simulations with multiple interacting characters. By contrast, our approach only uses extracted 2D trajectories and only generates the trajectories of virtual pedestrians. Our approach cannot perform simulations of full body motions or gestures.

Interactive applications: Compared with prior interactive data-driven methods, our approach captures the motion dynamics of the tracked pedestrians. This makes it possible to generate denser crowd simulations as well as using slightly varying environments and situations. Furthermore, our approach is automatic and involves no user editing.

5.6.2 User Study

To evaluate the potential benefits of our pedestrian dynamics learning algorithm, we performed a user study that sought to analyze how well the crowd simulations generated using TVPD matched the movements or trajectory characteristics in the original video in terms of perceptual similarity. Our hypothesis was that the movement patterns generated by *our* TVPD method would have a higher perceptual similarity to the movement patterns in the original video, as compared to the results generated by other crowd simulation algorithms.

Scenario 1: Outdoor walking scene with pedestrian cross flow. Around 40 pedestrians in the original video and around $60 - 70$ in the synthetically generated and rendered crowd simulation videos. We use scenarios with and without the two components of TVPD, entry point distributions and movement flows: (i) We use random entry points for pedestrians and simulate their movements towards the other side of the street. (ii) We use entry point distributions to generate initial positions of the pedestrians, and use the learned movement flows to simulate pedestrians.

Scenario 2: Outdoor running scene with 500 pedestrians in both the original video and the other generated using crowd simulation algorithms. We specifically compare the effect of movement flows (i) We use the local navigation algorithm with optimized motion parameters and intermediate sub-goals; (ii) We use the learned movement flows to simulate pedestrians. (i) and (ii) uses the same entry point distributions learned from the data and the same number of pedestrians.

Scenario 3: Outdoor scene with sudden changes in pedestrian velocity. There were 50 pedestrians in the original video and around 100 in the video generated using different crowd simulation algorithms. We compare the effects of movement flow without using entry point distributions. (i) We use manually specified entry positions and randomly chosen goal positions. (ii) We use the same entry positions as (i), but use the learned movement flows to simulate pedestrian trajectories.

For each scenario, we used one original video and two synthetically generated crowd simulation videos. All the videos were about $15-20$ seconds long. Every user is asked to compare the movement patterns in the original video with the synthetically generated crowd simulations, by playing them simultaneously. There were 39 participants (51.61% female, 64.52% in the age group of $20-30$) in the study. The study had no time constraints and the participants were free to take breaks in-between the benchmarks as long as the web-session did not expire.

Every user was randomly presented with these scenario studies and different synthetic videos for each scenario. We asked the users to first watch the original video and then rate each synthetic video on a scale of $1-5$ in terms of comparing the similarity of movement patterns between the original video and a synthetic video. A score of 1 indicated most dissimilar and a score of 5 indicated most similar movement pattern. Since this was a perceptual study related to movement patterns, we intentionally did not explicitly define or asked the user to classify the crowd behavior. Furthermore, we had asked the users not to rate the video based on the rendering quality, foot skidding, orientation issues, etc. We also encouraged the users to watch the original video and the synthetic video as many times as he/she wants and finally provide some (optional) feedback.

We measured the mean, variance of their scores, and compute p-values using a two-tailed t-test. We find that the crowd simulations generated by our TVPD learning algorithm scored much higher than the other two approaches for all scenarios, at a statistically significant rate (p-value< 0.05).

5.7 Conclusions, Limitations and Future Work

We present an interactive approach to learning the characteristics of pedestrian dynamics from trajectories extracted from real videos. These characteristics are used to compute collision-free trajectories of virtual pedestrians whose movement patterns resemble those of pedestrians in the original video. Our approach is automatic and interactive and captures the dynamically changing

Scenario	Approach	Mean Score	Variance	p-value
Scenario 1	(i) without TVPD	2.28	0.58	2.53e-05
	(ii) with TVPD	3.10	0.73	
Scenario 2	(i) without TVPD	1.44	0.25	5.06e-21
	(ii) with TVPD	3.59	0.67	
Scenario 3	(i) without TVPD	1.41	0.35	1.71-16
	(ii) with TVPD	3.49	0.99	

Table 5.2: Comparison of similarity scores (higher is more similar). These preliminary results indicate that the use of our pedestrian dynamics learning algorithm to compute entry points and movement flows considerably improves the perceptual similarity of our simulation to the pedestrian movements in the original video.

movement behaviors of real pedestrians. We demonstrate its applications for many data-driven crowd simulations, where we can easily add hundreds of virtual pedestrians, generate dense crowds, and change the environment or the situation.

Limitations: The performance of our learning algorithm is governed by the accuracy of the input trajectories. Current algorithms for automatic pedestrian tracking can only handle low-to-medium density crowds. Our learning algorithm makes some assumptions about sensor error and statistical distributions and is only useful for capturing the characteristics of local pedestrian dynamics for each pedestrian, whereas offline learning methods can compute many global characteristics. We consider only some movement characteristics to compute the trajectories of virtual pedestrians and do not consider other aspects of pedestrian behaviors or state, full body actions or the interactions among pedestrians. Our approach only computes the trajectories, but we need to combine with techniques that can generate plausible animation and rendering. For example, GOLAEM can result in some artifacts.

Future Work: There are many avenues for future work. In addition to overcoming the limitations of our work, our interactive TVPD can also be combined with other data-driven crowd simulation algorithms and offline behavior learning methods. We would like to combine the pedestrian dynamics characteristics with other techniques that can model complex crowd behaviors or multi-character motion synthesis techniques (Lee et al., 2006; Yersin et al., 2009).

CHAPTER 6
Conclusion and Future Work

In this thesis, we have presented our work on crowd behavior model. Our focus is on increasing the realism for simulating the movement and behavior of individual agents and the overall crowd. This includes simulating the interactions and effect of an agent with other agents in a crowd or within an environment. These models are used to improve various aspects of crowd simulation, such as individual differences and dynamic behavior changes (Chapter 2), collision avoidance behaviors (Chapter 3), and high-level behaviors: learning motion model parameters (Chapters 4 and learning spatio-temporal trajectory-level behavior characteristics 5). More specifically, we have presented following approaches:

Dynamic Crowd Behaviors: We proposed a method to simulate dynamic patterns and variations of individual behaviors based on psychological theories. The method can handle spatially- and temporally-varying situational factors in a unified framework. The proposed method is able to generate several emergent behaviors in different scenarios that match real-world observations.

Physical Interaction Models: We proposed a method to model new types of interactions involving physical contact such as collision response and intentional physical interactions. This method handles physical interactions while still preserving the ability of the crowd to avoid collisions with each other. In addition, agent-object interactions and agent-user interactions are available at interactive rates with the proposed method. The method can be useful for games, augmented reality or virtual reality applications where the user can directly interact with the virtual agents or the objects in the scene.

Adaptive Motion Model: We proposed a method to improve the motion model by learning from real-world data. Our approach is able to learn individual characteristics in an online manner, without any prior knowledge to the scene. The result can be used to predict the motion of the

individuals in the scene, which can improve the quality of the interaction between the users and the computer system for human-computer interactions, human-robot interactions, computer tracking applications, or augmented reality.

Online Data-driven Crowd Simulation: We proposed a method to learn high-level crowd behaviors from real-world data. This method keeps the advantage of data-driven method that uses behavior examples from real world data, but no longer requires building large database or pre-processing of the input data. Also, the method can be used with agent-based simulation techniques with or without specifying complex behavior FSMs or behavior trees to encode behavior rules. This handles both structured and unstructured data well, with temporal/spatial behavior changes.

Each method can be combined together or used independently to improve specific aspects of the crowd behavior. For example, one might want to model dynamic behavior changes based on stimuli but only considering anticipated collision avoidance. Also, we have shown the use of different local crowd behavior models in combination with high-level behavior models in chapter 5. Our adaptive motion model presented in chapter 4 can also be used with the local or high-level behavior models to learn and improve local navigation behaviors. In each chapter, we highlighted the real-time performance of our algorithms and discussed many benefits and broad applications.

There are many avenues for future work, in addition to overcoming the limitations of our work discussed in each chapter. Currently, our methods only compute the trajectory behaviors. For the local interaction models, we would like to address how our models can interact with the crowd rendering system (e.g., gesture animation or facial expression of agents). We would also like to extend our models to apply to crowds in 3D spaces or in a multi-layer framework.

For the higher-level behavior models, we have focused on capturing local, time-varying dynamics of crowd behaviors. We would like to explore combining our local model with global learning methods. In addition, we would like to incorporate learning the context of the environments and the static obstacles. Adding these components would greatly help understanding more long-term and common aspects of human behaviors.

For both local and high-level behavior models, we have not considered effects of social behaviors, for example grouping behaviors, effect of verbal/nonverbal communications, etc. Understanding and modeling these components would greatly extend the usage of the crowd simulation.

Finally, we would like to explore more applications of our behavior models. We have shown the use of BRVO technique for a car-like robot navigation. We believe that our TVPD algorithm can further improve the navigation and tracking problems in robotics and computer vision, by providing better understandings of crowd behavior. We also would like to explore more virtual reality and augmented reality applications that can benefit from our methods.

BIBLIOGRAPHY

Al-nasur, S. and Kachroo, P. (2006). A microscopic-to-macroscopic crowd dynamic model. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 606–611.

AMD (2012). Bullet Physics 2.80. http://bulletphysics.org.

AMD (2012). Bullet physics 2.80 http://bulletphysics.org.

Anderson, C. A. (2001). Heat and violence. *Current Directions in Psychological Science*, 10(1):pp. 33–38.

Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255.

Bandini, S., Federici, M., Manzoni, S., and Vizzari, G. (2006). Towards a methodology for situated cellular agent based crowd simulations. In Dikenelli, O., Gleizes, M.-P., and Ricci, A., editors, *Engineering Societies in the Agents World VI*, volume 3963 of *Lecture Notes in Computer Science*, pages 203–220. Springer Berlin Heidelberg.

Baraff, D. (1997). An introduction to physically based modeling: Rigid body simulation i - unconstrained rigid body dynamics. In *In An Introduction to Physically Based Modelling, SIGGRAPH '97 Course Notes*, page 97.

Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48.

Bera, A., Kim, S., and Manocha, D. (2015). Efficient trajectory extraction and parameter learning for data-driven crowd simulation. In *Proceedings of Graphics Interface*.

Bera, A. and Manocha, D. (2014). Realtime multilevel crowd tracking using reciprocal velocity obstacles. *IEEE International Conference on Pattern Recognition*.

Berkowitz, L. (1990). On the formation and regulation of anger and aggression: A cognitive-neoassociationistic analysis. *American Psychologist*, 45(4):494 – 503.

Berseth, G., Kapadia, M., Haworth, B., and Faloutsos, P. (2014). Steerfit: Automated parameter fitting for steering algorithms. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*.

Best, A., Narang, S., Curtis, S., and Manocha, D. (2014). Densesense: Interactive crowd simulation using density-dependent filters. *Symposium on Computer Animation*, pages 97–102.

Blandin, S., Couque, A., Bayen, A., and Work, D. (2012). On sequential data assimilation for scalar macroscopic traffic flow models. *Physica D: Nonlinear Phenomena*.

Boltes, M., Seyfried, A., Steffen, B., and Schadschneider, A. (2010). Automatic extraction of pedestrian trajectories from video recordings. *Pedestrian and Evacuation Dynamics 2008*, pages 43–54.

Borges, P., Conci, N., and Cavallaro, A. (2013). Video-based human behavior understanding: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(11):1993–2008.

Brscic, D., Kanda, T., Ikeda, T., and Miyashita, T. (2013). Person position and body direction tracking in large public spaces using 3d range sensors. *IEEE Transactions on Human-Machine Systems*, 43(6).

Bruce, A. and Gordon, G. (2004). Better motion prediction for people-tracking. In *Proc. of the International Conference on Robotics and Automation (ICRA), New Orleans, USA*.

Buckland, M. (2005). *Programming game AI by example*. Jones & Bartlett Learning.

Casella, G. and Berger, R. (2002). *Statistical inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning.

Cattell, R. (1952). *Factor analysis: an introduction and manual for the psychologist and social scientist*. Harper's psychological series. Harper.

Charalambous, P., Karamouzas, I., Guy, S. J., and Chrysanthou, Y. (2014). A data-driven framework for visual crowd analysis. *Computer Graphics Forum*, 33(7):41–50.

Collins, R. T. (2003). Mean-shift blob tracking through scale space. In *CVPR*.

Cox, T. (1978). *Stress*. Macmillan.

Crompton, D. (1979). Pedestrian delay, annoyance and risk: preliminary results from a 2 years study. In *Proceedings of PTRC Summer Annual Meeting*, pages 275–299.

Cui, J., Zha, H., Zhao, H., and Shibasaki, R. (2005). Tracking multiple people using laser and vision. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2116–2121. IEEE.

Curtis, S., Guy, S. J., Zafar, B., and Manocha, D. (2011). Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds. In *1st IEEE Workshop on Modeling, Simulation and Visual Analysis of Large Crowds*, pages 128–135.

Curtis, S. and Manocha, D. (2012). Pedestrian simulation using geometric reasoning in velocity space. In *Proceedings of Pedestrian and Evacuation Dynamics*.

Curtis, S., Snape, J., and Manocha, D. (2012a). Way portals: efficient multi-agent navigation with line-segment goals. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 15–22, New York, NY, USA. ACM.

Curtis, S., Zafar, B., Gutub, A., and Manocha, D. (2012b). Right of way. *The Visual Computer*, pages 1–16.

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

Du Toit, N. and Burdick, J. (2010). Robotic motion planning in dynamic, cluttered, uncertain environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 966–973.

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

Enzweiler, M. and Gavrila, D. M. (2009). Monocular pedestrian detection: Survey and experiments. *PAMI*, pages 2179–2195.

Eriksen, H. R., Olff, M., Murison, R., and Ursin, H. (1999). The time dimension in stress responses: relevance for survival and health. *Psychiatry Research*, 85(1):39 – 50.

Evans, G. (1984). *Environmental Stress*. Cambridge University Press.

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

Eysenck, M. (2002). *Simply psychology*. Psychology Press.

Fod, A., Howard, A., and Mataric, M. (2002). A laser-based people tracker. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 3024–3029.

Fulgenzi, C., Spalanzani, A., and Laugier, C. (2007). Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1610–1616.

Golas, A., Narain, R., and Lin, M. (2013). Hybrid long-range collision avoidance for crowd simulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, pages 29–36, New York, NY, USA. ACM.

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

Gong, H., Sim, J., Likhachev, M., and Shi, J. (2011). Multi-hypothesis motion planning for visual object tracking. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 619–626.

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

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

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

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

Hargreaves, J., Annan, J., Edwards, N., and Marsh, R. (2004). An efficient climate forecasting method using an intermediate complexity earth system model and the ensemble kalman filter. *Climate Dynamics*, 23(7):745–760.

Heider, F. (1982). *The psychology of interpersonal relations*. Lawrence Erlbaum Associates.

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

Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical review E*.

Henry, P., Vollmer, C., Ferris, B., and Fox, D. (2010). Learning to navigate through crowded environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 981–986.

Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352.

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

Ikeda, T., Chigodo, Y., Rea, D., Zanlungo, F., Shiomi, M., and Kanda, T. (2012). Modeling and prediction of pedestrian behavior based on the sub-goal concept. In *Robotics: Science and Systems'12*.

Jordao, K., Pettré, J., Christie, M., and Cani, M.-P. (2014). Crowd Sculpting: A space-time sculpting method for populating virtual environments. *Computer Graphics Forum*, 33(2):351–360.

Ju, E., Choi, M. G., Park, M., Lee, J., Lee, K. H., and Takahashi, S. (2010). Morphable crowds. *ACM Trans. Graph.*, 29(6):140:1–140:10.

Kapadia, M., Chiang, I.-k., Thomas, T., Badler, N. I., and Kider, Jr., J. T. (2013). Efficient motion retrieval in large motion databases. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 19–28.

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

Karamouzas, I. and Overmars, M. (2010). A velocity-based approach for simulating human collision avoidance. In *Intelligent Virtual Agents*, pages 180–186. Springer.

Karamouzas, I. and Overmars, M. (2012). Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):394–406.

K.B. and Rasmussen (1979). Annoyance from simulated road traffic noise. *Journal of Sound and Vibration*, 65(2):203 – 214.

Kim, J., Seol, Y., Kwon, T., and Lee, J. (2014a). Interactive manipulation of large-scale crowd animation. *ACM Trans. Graph.*, 33(4):83:1–83:10.

Kim, M., Hwang, Y., Hyun, K., and Lee, J. (2012a). Tiling motion patches. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 117–126. Eurographics Association.

Kim, M., Hyun, K., Kim, J., and Lee, J. (2009). Synchronized multi-character motion editing. *ACM Trans. Graph.*, 28(3):79:1–79:9.

Kim, S., Bera, A., and Manocha, D. (2015a). Interactive crowd content generation and analysis using trajectory-level behavior learning. Technical report, University of North Carolina at Chapel Hill.

Kim, S., Guy, S., Hillesland, K., Zafar, B., Gutub, A.-A., and Manocha, D. (2014b). Velocity-based modeling of physical interactions in dense crowds. *The Visual Computer*, pages 1–15.

Kim, S., Guy, S., Liu, W., Lau, R., Lin, M., and Manocha, D. (2012b). Predicting pedestrian trajectories using velocity-space reasoning. In *The Tenth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Kim, S., Guy, S. J., Liu, W., Wilkie, D., Lau, R. W., Lin, M. C., and Manocha, D. (2015b). Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, 34(2):201–217.

Kim, S., Guy, S. J., and Manocha, D. (2013). Velocity-based modeling of physical interactions in multi-agent simulations. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. Eurographics Association.

Kim, S., Guy, S. J., Manocha, D., and Lin, M. C. (2012c). Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Symposium on Interactive 3D Graphics*, I3D '12, pages 55–62. ACM.

Koshak, N. and Fouda, A. (2008). Analyzing pedestrian movement in mataf using gps and gis to support space redesign. In *The 9th International Conference on Design and Decision Support Systems in Architecture and Urban Planning*.

Köster, G., Treml, F., and Gödel, M. (2013). Avoiding numerical pitfalls in social force models. *Phys. Rev. E*, 87:063305.

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).

Kwon, T., Lee, K. H., Lee, J., and Takahashi, S. (2008). Group motion editing. *ACM Trans. Graph.*, 27(3):80:1–80:8.

Lazarus, R. S. (1993). From psychological stress to the emotions: A history of changing outlooks. *Annual Reviews Psychology*, 44:1–22.

Lee, K. H., Choi, M. G., Hong, Q., and Lee, J. (2007). Group behavior from video: A data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 109–118, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Lee, K. H., Choi, M. G., and Lee, J. (2006). Motion patches: Building blocks for virtual environments annotated with motion data. *ACM Trans. Graph.*, 25(3):898–906.

Lemercier, S., Jelic, A., Kulpa, R., Hua, J., Fehrenbach, J., Degond, P., Appert-Rolland, C., Donikian, S., and Pettré, J. (2012). Realistic following behaviors for crowd simulation. *Computer Graphics Forum*, 31(2pt2):489–498.

Leon, J. (2010). Pedestrian psychology and safety. http://www.drdriving.org/pedestrians/.

Lerner, A., Chrysanthou, Y., and Lischinski, D. (2007). Crowds by example. In *Computer Graphics Forum*, volume 26, pages 655–664. Wiley Online Library.

Lerner, A., Fitusi, E., Chrysanthou, Y., and Cohen-Or, D. (2009). Fitting behaviors to pedestrian simulations. In *Symp. on Computer Animation*, pages 199–208.

Li, T., Chang, H., Wang, M., Ni, B., Hong, R., and Yan, S. (2015). Crowded scene analysis: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 25(3):367–386.

Li, Y., Christie, M., Siret, O., Kulpa, R., and Pettré, J. (2012). Cloning crowd motions. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 201–210.

Liao, L., Fox, D., Hightower, J., Kautz, H., and Schulz, D. (2003). Voronoi tracking: Location estimation using sparse and noisy sensor data. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 723–728. IEEE.

Liu, W., Chan, A. B., Lau, R. W. H., and Manocha, D. (2014). Leveraging long-term predictions and online-learning in agent-based multiple person tracking. *IEEE Transactions on Circuits and System for Video Technology*.

Luber, M., Stork, J., Tipaldi, G., and Arras, K. (2010). People tracking with human motion predictions from social forces. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 464–469.

Maki, B., McIlroy, W., and Fernie, G. (2003). Change-in-support reactions for balance recovery. *Engineering in Medicine and Biology Magazine, IEEE*, 22(2):20–26.

McLachlan, G. J. and Krishnan, T. (2008). *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2 edition.

Mehran, R., Oyama, A., and Shah, M. (2009). Abnormal crowd behavior detection using social force model. In *CVPR*.

Middlemist, R. D., Knowles, E. S., and Matter, C. F. (1976). Personal space invasions in the lavatory: Suggestive evidence for arousal. *Journal of Personality and Social Psychology*, 33(5):541 – 546.

Miller, N. E. (1941). I. the frustration-aggression hypothesis. *Psychological Review*, 48(4):337–342.

Mordkoff, A. M. (1964). The relationship between psychological and physiological response to stress. *Psychosomatic Medicine*, 26(2):135–150.

Muico, U., Popović, J., and Popović, Z. (2011). Composite control of physically simulated characters. *ACM Transactions on Graphics*, 30(3).

Musse, S. R. and Thalmann, D. (1997). A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Proc. Workshop of Computer Animation and Simulation of Eurographics'97*, pages 39–51.

Narain, R., Golas, A., Curtis, S., and Lin, M. C. (2009). Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.*, 28(5):122:1–122:8.

Nuria Pelechano, Kevin O'Brien, B. S. N. B. (2005). Crowd simulation incorporating agent psychological models, roles and communication. In *First International Workshop on Crowd Simulation. (V-Crowds '05)*, pages 21–30.

Nygren, M. (2007). Simulation of human behaviour in stressful crowd situations.

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

Oswald and Bratfisch (1969). A further study of the relation between subjective distance and emotional involvement. *Acta Psychologica*, 29(0):244 – 255.

Paris, S. and Donikian, S. (2009). Activity-driven populace: A cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications*, 29(4):34–43.

Patil, S., van den Berg, J., Curtis, S., Lin, M., and Manocha, D. (2011). Directing crowd simulations using navigation fields. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):244 –254.

Pelechano, N., Allbeck, J., and Badler, N. (2008). Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation*, 3(1):1–176.

Pelechano, N., Allbeck, J. M., and Badler, N. I. (2007). Controlling individual agents in high-density crowd simulation. In *Symposium on Computer animation*, pages 99–108.

Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proc. of the (IEEE) 12th International Conference on Computer Vision*, pages 261–268.

Pellegrini, S., Gall, J., Sigal, L., and Gool, L. (2012). Destination flow for crowd simulation. In *Computer Vision – ECCV 2012. Workshops and Demonstrations*, volume 7585, pages 162–171.

Peters, C. and Ennis, C. (2009). Modeling groups of plausible virtual pedestrians. *Computer Graphics and Applications, IEEE*, 29(4):54–63.

Pettré, J., Ondřej, J., Olivier, A.-H., Cretual, A., and Donikian, S. (2009). Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 189–198, New York, NY, USA. ACM.

Pradhan, N., Burg, T., and Birchfield, S. (2011). Robot crowd navigation using predictive position fields in the potential function framework. In *American Control Conference (ACC)*, pages 4628–4633.

Pražák, M. and O'Sullivan, C. (2011). Perceiving human motion variety. In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization*, APGV '11, pages 87–92, New York, NY, USA. ACM.

Reynolds, C. (1999). Steering Behaviors for Autonomous Characters. In *Game Developers Conference 1999*.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA. ACM.

Rodriguez, M., Ali, S., and Kanade, T. (2009). Tracking in unstructured crowded scenes. In *Proc. of the IEEE 12th International Conference on Computer Vision*, pages 1389–1396.

Sakuma, T., Mukai, T., and Kuriyama, S. (2005). Psychological model for animating crowded pedestrians: Virtual humans and social agents. *Comput. Animat. Virtual Worlds*, 16:343–351.

Schadschneider, A., Klingsch, W., Kl′upfel, H., Kretz, T., Rogsch, C., and Seyfried, A. (2009). Evacuation dynamics: Empirical results, modeling and applications. pages 3142–3176.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2003). People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99.

Sean Curtis, Andrew Best, D. M. (2013). Menge: A modular framework for simulating crowd movement. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill.

Selye, H. (1956). *The stress of life*. Number v. 5 in McGraw-Hill paperbacks. McGraw-Hill.

Seyfried, A., Steffen, B., Klingsch, W., and Boltes, M. (2005). The fundamental diagram of pedestrian movement revisited. *J. Stat. Mech.*, 2005:P10002.

Shao, W. and Terzopoulos, D. (2005). Autonomous pedestrians. In *Symposium on Computer animation*, pages 19–28.

Shapiro, A., Pighin, F., and Faloutsos, P. (2003). Hybrid control for interactive character animation. In *Pacific Conference on Computer Graphics and Applications*, PG '03, Washington, DC, USA. IEEE Computer Society.

Shiratori, T., Coley, B., Cham, R., and Hodgins, J. K. (2009). Simulating balance recovery responses to trips based on biomechanical principles. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Shopf, J., Barczak, J., Oat, C., and Tatarchuk, N. (2008). March of the froblins: simulation and rendering massive crowds of intelligent and detailed creatures on gpu. In *ACM SIGGRAPH 2008 classes*, pages 52–101.

Shum, H. P. H., Komura, T., and Yamazaki, S. (2012). Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):741–752.

Snape, J., van den Berg, J., Guy, S., and Manocha, D. (2011). The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics (TRO)*, 27(4):696–706.

Sok, K. W., Yamane, K., Lee, J., and Hodgins, J. (2010). Editing dynamic human motions via momentum and force. In *Symposium on Computer Animation*, SCA '10, pages 11–20, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Solmaz, B., Moore, B. E., and Shah, M. (2012). Identifying behaviors in crowd scenes using stability analysis for dynamical systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10):2064–2070.

Stevens, S. S. (1957). On the psychophysical law. *Psychological Review*, 64(3):153 – 181.

Still, G. K. (2013). Proximate and distal causality `http://www.gkstill.com/ExpertWitness/CrowdDisasters.html`.

Sun, L., Li, X., and Qin, W. (2013). Simulating realistic crowd based on agent trajectories. *Computer Animation and Virtual Worlds*, 24(3-4):165–172.

Svenstrup, M., Bak, T., and Andersen, H. (2010). Trajectory planning for robots in dynamic human environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4293–4298.

Teghtsoonian, R. and Frost, R. O. (1982). The effects of viewing distance on fear of snakes. *Journal of Behavior Therapy and Experimental Psychiatry*, 13(3):181 – 190.

Thalmann, D. and Musse, S. (2013). Behavioral animation of crowds. In *Crowd Simulation*, pages 111–168. Springer London.

Trautman, P. and Krause, A. (2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 797–803.

Trautman, P., Ma, J., Murray, R. M., and Krause, A. (2013). Robot navigation in dense human crowds: the case for cooperation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2153–2160.

Treuille, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. In *ACM SIGGRAPH 2006*, pages 1160–1168. ACM.

Ulicny, B. and Thalmann, D. (2002). Towards interactive real-time crowd behavior simulation. In *Computer Graphics Forum*, volume 21, pages 767–775. Wiley Online Library.

van den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics Research: 14th ISRR (STAR)*, volume 70, pages 3–19.

van den Berg, J., Lin, M., and Manocha, D. (2008a). Reciprocal velocity obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1928 –1935.

van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. (2008b). Interactive navigation of individual agents in crowded environments. In *Symp. on Interactive 3D Graphics and Games (I3D)*.

Wan, E. and Van der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proc. of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium. AS-SPCC.*, pages 153–158.

Wei, Y., Gang, B., and Zuwen, W. (2009). Balance recovery for humanoid robot in the presence of unknown external push. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pages 1928–1933.

Wilkie, D., van den Berg, J., and Manocha, D. (2009). Generalized velocity obstacles. In *Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 5573–5578, Piscataway, NJ, USA. IEEE Press.

Wolinski, D., Guy, S. J., Olivier, A.-H., Lin, M. C., Manocha, D., and Pettré, J. (2014). Parameter estimation and comparative evaluation of crowd simulations. In *Eurographics*.

Work, D., Tossavainen, O., Blandin, S., Bayen, A., Iwuchukwu, T., and Tracton, K. (2008). An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. In *47th IEEE Conference on Decision and Control, CDC 2008.*, pages 5062–5068. IEEE.

Yamaguchi, K., Berg, A., Ortiz, L., and Berg, T. (2011). Who are you with and where are you going? In *Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1345–1352.

Yeh, H., Curtis, S., Patil, S., van den Berg, J., Manocha, D., and Lin, M. (2008). Composite agents. In *Symposium on Computer Animation*, pages 39–47.

Yerkes, R. M. and Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, 18(5):459–482.

Yersin, B., Maïm, J., Pettré, J., and Thalmann, D. (2009). Crowd patches: populating large-scale virtual environments for real-time applications. In *Interactive 3D graphics and games*, pages 207–214.

Yu, Q. and Terzopoulos, D. (2007). A decision network framework for the behavioral animation of virtual humans. In *Symposium on Computer animation*, pages 119–128.

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

Zen, G. and Ricci, E. (2011). Earth mover's prototypes: A convex learning approach for discovering activity patterns in dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3225–3232.

Zhang, K., Zhang, L., and Yang, M.-H. (2012). Real-time compressive tracking. In *ECCV*, pages 864–877.

Zhou, B., Wang, X., and Tang, X. (2012). Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2871–2878.

Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., and Srinivasa, S. (2009). Planning-based prediction for pedestrians. In *IROS*, pages 3931–3936.

Zordan, V. B., Majkowska, A., Chiu, B., and Fast, M. (2005). Dynamic response for motion capture animation. *ACM Trans. Graph.*, 24(3):697–701.