INTERACTIVE MOTION PLANNING FOR MULTI-AGENT SYSTEMS WITH PHYSICS-BASED AND BEHAVIOR CONSTRAINTS

Andrew Best

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
2018

Approved by:

Dinesh Manocha

Ming Lin

Daniel Barber

Ron Alterovitz

Ketan Mayer-Patel

ABSTRACT

Andrew Best: Interactive Motion Planning for Multi-agent Systems with Physics-based and Behavior Constraints
(Under the direction of Dinesh Manocha)

Man-made entities and humans rely on movement as an essential form of interaction with the world. Whether it is an autonomous vehicle navigating crowded roadways or a simulated pedestrian traversing a virtual world, each entity must compute safe, effective paths to achieve their goals. In addition, these entities, termed *agents*, are subject to unique physical and behavioral limitations within their environment. For example, vehicles have a finite physical turning radius and must obey behavioral constraints such as traffic signals and rules of the road. Effective motion planning algorithms for diverse agents must account for these physics-based and behavior constraints.

In this dissertation, we present novel motion planning algorithms that account for constraints which physically limit the agent and impose behavioral limitations on the virtual agents. We describe representational approaches to capture specific physical constraints on the various agents and propose abstractions to model behavior constraints affecting them. We then describe algorithms to plan motions for agents who are subject to the modeled constraints.

First, we describe a biomechanically accurate elliptical representation for virtual pedestrians; we also describe human-like movement constraints corresponding to shoulder-turning and side-stepping in dense environments. We detail a novel motion planning algorithm extending velocity obstacles to generate collision-free paths for hundreds of elliptical agents at interactive rates. Next, we describe an algorithm to encode dynamics and traffic-like behavior constraints for autonomous vehicles in urban and highway environments. We describe a motion planning algorithm to generate safe, high-speed avoidance maneuvers using a novel optimization function and modified control obstacle formulation, and we also present a simulation framework to evaluate driving strategies. Next, we present an approach to incorporate high-level reasoning to model the motions and behaviors of virtual agents in terms of verbal interactions with other agents or avatars. Our approach leverages natural-language interaction to reduce uncertainty and generate effective plans. Finally,

we describe an application of our techniques to simulate pedestrian behaviors for gathering simulated data about loading, unloading, and evacuating an aircraft.

To my parents, who were not able to share this moment with me.

I love you, and I know you would be proud.

ACKNOWLEDGMENTS

I would first like to thank my loving wife, Kathryn, the rock upon which this accomplishment was built. Without your support, your encouragement, your dedication, and your sharing of sleepless nights, I would never have been able to accomplish this feat.

Next, to Sahil, my constant collaborator, best friend, and labmate. Many of the contributions in this dissertation are the product of your feedback, filtration, hard work, and refinement. Thank you for being a dedicated and dependable collaborator, an honest and helpful critic, and a great friend.

To my brothers, Nick and Terry, thank you for treating me as an equal from an early age. Thank you for supporting me and for keeping me in check when needed. Thank you for teaching me to survive. Nick, thank you especially for teaching me to roll with the punches, to appreciate life on the road, and to focus on what's important.

Thank you to my loving foster parents, Marianne and David, who taught me to think critically, pursue my dreams, and push myself to be better.

Thank you to my parents, Terry and Debbie. I'm sorry you didn't get to see this day, but I know you would be proud. Thank you for your love and guidance; especially to my father, who was always very proud of my education.

I have an extensive and immensely loving family. Thank you to my aunts and uncles who cared for me, to my in-laws who treat me like a son, and to my grandparents and cousins for tolerating my antics all these years.

Thanks to my advisor, Dr. Dinesh Manocha, for convincing me this was possible and pushing me to achieve it. Without your guidance, I would not have been able to manage this journey. Thank you as well to the members of my committee for their advice and support.

I have worked with many collaborators in my time at UNC, too many to name, but many of whom left a profound impact on me. Thanks to the members of the GAMMA lab for helpful feedback and intense practice sessions. Thank you to the students who shared the lab with me for being a sounding board and for

always being willing to help me test prototypes. Thanks especially to Sean Curtis for mentoring me early in my graduate career and helping me discover crowd simulation.

Finally, thank you to my friends and family I have met in Chapel Hill. I will dearly miss the socials, board games, get-togethers, and late night fooseball.

To the reader, I hope you find whatever knowledge you seek in reading this. If you are reading this as an acquaintance of mine, thank you especially for that. The legacy of a man is not only in the artifacts he leaves behind, but in the hearts and minds of the lives he touched. Thank you for remembering me. I hope you do so fondly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Safely traveling from one point to another is a problem faced daily by virtually everything that moves. Humans intuitively plan effective routes to their destinations and respond to changes in their environment with relative ease. They plan paths to achieve their desires, adapt those paths to changing conditions, and avoid other people, vehicles, or obstacles, often all at the same time. For computer-controlled entities, such as robots, video game characters, or autonomous cars, the problem of navigating a complex world remains challenging.

In this dissertation, we investigate scenarios in which many virtual entities must navigate dense shared spaces. These entites must avoid colliding with one another, navigate to their destinations, and, in some cases, communicate to achieve their desires. We propose novel algorithms inspired by observations of human crowd motion. We demonstrate their utility in recreating crowd behavior and understanding crowd phenomena. We additionally apply our insights to autonomous vehicles and demonstrate an algorithm for safely navigating an autonomous vehicle in typical traffic conditions.

Researchers refer to the problem of planning safe paths for a computer-controlled entity, called an *agent*, as motion planning. Each agent is described by its *configuration*, the set of quantities which describe its current state in the world. For a simple planar robot, such as an autonomous vacuum cleaner, a sufficient configuration could be its position, heading, radius, and current speed. Formally, motion planning is the process of computing a set of intermediate configurations for an agent which take it from some initial configuration, i.e. where it is currently, to a desired configuration, i.e. where it needs to be.

The motion-planning problem is typically solved by applying a set of transformations on the current state of the agent subject to sets of constraints. In the case of navigating the autonomous vacuum cleaner, these transformations typically represent actions the robot may take, such as driving straight or turning in place for some duration. Similarly, for the agents discussed in this dissertation, these transformations take the form of deliberate actions and planar motion. For more complex robots, such as robotic arms, these transformations may represent delicate combinations of angle changes in the many joints that make up the arm. The constraints on this motion include physical limits, e.g. the autonomous vacuum cleaner cannot

move up from the ground nor can it move sideways, and desired conditions of the motion, e.g. it should not collide with the furniture.

Motion planning is studied in many diverse fields of computation. In computer graphics, motion planning is used to plan paths for virtual characters in entertainment applications for collision-avoidance (Geraerts et al., 2008) or as a technique to generate natural-looking motion (Narang et al., 2018; Treuille et al., 2007). Motion planning is used as the basis for numerous techniques in robotics for navigation (Ziegler et al., 2014a; Park et al., 2012), manipulation (Kuffner and LaValle, 2000), locomotion (Dalibard et al., 2013), and is the subject of several highly informative textbooks (Latombe, 1991; LaValle, 2006).

It is often necessary and desirable to evaluate motion planning algorithms in the presence of other agents or a human user. Simulated virtual environments allow researchers to evaluate robotic platforms, character animation approaches, and navigation techniques rapidly and safely. *Multi-agent systems* is the term used to describe systems where many agents (and often human users) plan, act, and interact in a shared environment. *Multi-agent simulation* is the special case where these agents interact in a virtual environment. These simulations are used in a wide range of computational fields including motion planning, evacuation planning (Helbing et al., 2000; Fang et al., 2016), network modeling, security, and many others. In general, the multi-agent system approach maintains a common characteristic: each agent operates as an independent entity, capable of sensing the environment, planning actions to accomplish individual goals, and acting on those plans.

Two motivating applications at the intersection of motion planning and multi-agent systems research are crowd simulation and autonomous driving. Crowd simulation studies how humans move and interact in shared virtual or physical environments. Autonomous driving investigates the problem of safely navigating one or more vehicles from one location to another. These domains have many unique characteristics, but share several important core challenges. In this dissertation, we propose algorithms for motion planning in crowd simulation and autonomous driving subject to these core challenges.

First, whether the agent is an autonomous vehicle or a virtual pedestrian, the motion planning algorithm must account for the *physics-based* constraints of the underlying entity. These constraints represent physical limitations of the entity including the steering radius of a vehicle or the comfortable walking speed of a pedestrian (Gérin-Lajoie et al., 2005; LaValle, 2006). These constraints tend to satisfy the laws of physics, and failure to account for them may result in erratic or unsafe behavior of the agents. Autonomous vehicles

can lose control when given improper commands. Virtual pedestrians may display visual artefacts such as walking into walls and impair the utility of the simulation.

Second, algorithms must model the *behavioral* constraints of the entity and environment in order to generate effective plans for the agents. These constraints are not necessarily physical limitations, but capture the context of an agent's environment and generate more plausible motions for the agents. For example, in the United States, pedestrians tend to walk on the right side of corridors. Similarly, vehicles drive on the right side of the road. Failure to model behavioral constraints could result in collisions between virtual pedestrians and violation of traffic laws by vehicles leading to dangerous outcomes. In addition, in the presence of human users, violating behavior constraints leads to decreased plausibility and quality of user experience (Bailenson et al., 2005).

Finally, many applications in crowd simulation or autonomous vehicles require a motion planning algorithm to be computationally efficient enough to compute safe paths at interactive rates. Virtual crowds used in interactive applications such as video games and virtual-reality must plan paths sufficiently quickly to react to the user and one another without disrupting the overall experience. Autonomous vehicles must compute their paths at rates which enable emergency response maneuvers and account for the high-speed and rapidly changing environments in which they operate.

1.1 Crowd Simulation

Crowd simulation has applications to numerous domains. Architects and designers use insights from simulated crowds to design safer structures for human use. Crowds often form the background in entertainment applications such as movies, and video-games. Simulated crowds can also provide meaningful insight into how spaces are utilized and how to improve comfort and mobility in public and private spaces. Insights from the study of human crowds can also assist in the design of safer robots in the context of human-robot shared spaces.

In the field of pedestrian dynamics, researchers simulate pedestrian crowds in order to understand how individual behaviors and navigation decisions combine to generate emergent phenomena such as lane-

[1]Mona Lisa image by Victor Grigas, from Wikimedia Commons, `https://en.wikipedia.org/wiki/Mona_Lisa#/media/File:Crowd_looking_at_the_Mona_Lisa_at_the_Louvre.jpg`. Concert image from the Telegraph, `https://www.telegraph.co.uk/music/what-to-listen-to/ariana-grande-one-love-manchester-concert-live/`. Love Parade image by Arne Museler, from Wikimedia Commons, `https://commons.wikimedia.org/wiki/File:2010_07_24_arne_mueseler_0223.jpg`

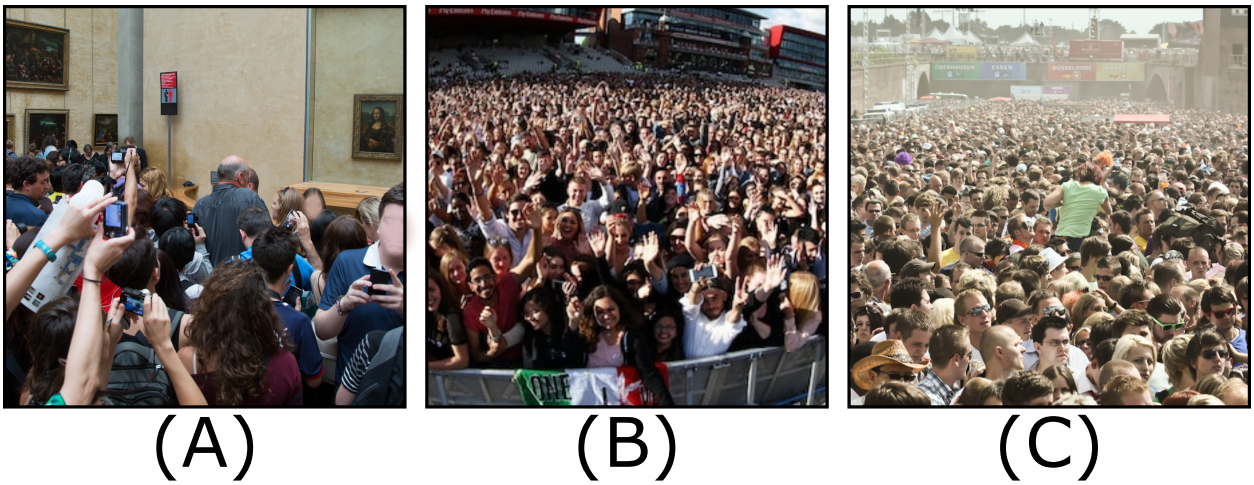


Figure 1.1: **Understanding Human Crowd Behavior:**[1]Human crowds exhibit many interesting behaviors encompassing physics-based and behavior constraints. **(A):** A crowd presses in to observe the Mona Lisa at the L'ouvre museum in Paris. Despite the dense crowd and physical movement constraints, the pedestrians respect and avoid nearby artwork. **(B):** Crowds press against the barricades at a concert in the U.K. The pedestrians create extreme densities without injuries. **(C):** In some instances, crowding can become deadly, such as the case of the infamous Love Parade disaster in 2010 (Love Parade, 2010). Understanding and simulating how pedestrians move and navigate in constrained spaces, as well as modeling the physics-based and behavior constraints that impact their decision-making, can provide insights into how to avoid disasters and design safer spaces for crowds.

formation (Helbing et al., 2000), turbulent flows (Kim et al., 2013; Krausz and Bauckhage, 2012), and congestion (Seyfried et al., 2009; Curtis et al., 2011). Effectively simulating these phenomena requires understanding both the physical constraints, such as movement speed and stride, and behavior constraints, such as mental, social, and other factors which govern the pedestrian's ultimate path. It is also necessary to model how the pedestrians affect one another. For example, pedestrians tend to move more slowly in dense-spaces (Fruin, 1971), and can employ side-steps or shoulder turns (Hughes et al., 2014) to avoid each other in congestion. Pedestrians also exhibit social norms such as a preference for personal space. Figure 1.1 details some of the behaviors studied by crowd simulation researchers.

A recent application of crowd and pedestrian simulation is immersive virtual reality experiences. Plausible behaviors for virtual agents in virtual reality environments increase the sense of realism of the environment and provide positive impacts in the perceived experience of users. Some research has shown the positive effects of virtual crowds on treating social phobias (Rothbaum et al., 1995, 2001), and in training simulations (Rickel and Johnson, 1999). Capturing the physics-based and behavior constraints of pedestrians can provide

clear benefits to these applications and enable the virtual agents to interact with users and one another in more meaningful ways.

1.1.1 Motion Planning for Crowd Simulation

There has been extensive work in motion planning for virtual crowds or pedestrians in the fields of computer graphics, robotics, and pedestrian dynamics for more than three decades. We limit our discussion to agent-based methods which treat each pedestrian in the crowd as an independent agent which plans an individual goal. Agent-based crowd simulation models often fall in the global-local planner paradigm (GLP). In GLP simulators, two different motion planning algorithms are coupled to create the plan for each agent. First, a motion planning algorithm, referred to as the global planning algorithm, generates a path through the static obstacles and environment of a simulation. this plan is communicated to a second motion planning algorithm, referred to as the local planner, which adapts the static plan to respect dynamic obstacles, agents, and other non-static phenomena. The plan is communicated as a *preferred velocity*, the instantaneous velocity which optimally follows the agent's path to its goal. The GLP paradigm is well studied in crowd simulation literature (Hughes et al., 2014; Narang et al., 2015; Funge et al., 1999; Ulicny and Thalmann, 2002).

Many techniques exist to determine the preferred velocity. A common approach involve discretizing the traversable space into connected components on a graph and using graph search algorithms (e.g. A*). Among the graph-based approaches are roadmaps (Latombe, 1991) which compute a network of connected way-points in environment to enable navigation over the graph of waypoiints. Snook (2000) described a navigation-mesh approach. Navigation-meshes decompose the environment into a set of connected polygons and utilize a funnel algorithm to find a set of connected polygons from a position to a goal. Similarly, Delaunay triangulation approaches triangulate the environment and navigate over the connected polygons (Lamarche and Donikian, 2004). Corridor maps (Geraerts et al., 2008) compute corridors of obstacle-free space the agents can navigate through to find their goal position. Other techniques, such as potential fields, do not explicitly compute a path but rather discretize the space into a field that is the gradient of some cost function (Khatib, 1986). The preferred velocity in this case is the value of the field at the agent's location.

Local navigation algorithms are used to adapt the preferred velocity computed by the global navigation algorithm to a feasible velocity which respects local, dynamic conditions. Different methods and paradigms have been proposed for local navigation. One class of local navigation algorithms is force-based models (Helbing and Molnár, 1995; Johansson et al., 2007; Zanlungo et al., 2011; Chraibi et al., 2010). These

algorithms treat agents as mass-particles and determine a new velocity by application of a set of modeled forces. Another commonly used approach is velocity-space based techniques (Van den Berg et al., 2008b; Paris et al., 2007; Van den Berg et al., 2011). These approaches use geometric optimization to compute a collision-free, feasible velocity directly in the velocity space by considering nearby agent and obstacles as constraints on the agent's velocity. A velocity is selected from the available velocity space based on an optimization criteria. Other methods evolve the agent state by specific steering rules in continuous space (Reynolds, 1987) or by decomposing the space into a grid of connected cells (Schadschneider, 2002). Some algorithms consider a synthetic-vision approach in which agents process "visual" information (Ondřej et al., 2010).

1.1.2 Modeling Behaviors for Crowd Simulation

Behavior modeling for crowd simulation is typically decomposed into high-level behavior and local behavior models. High-level behavior models generate the goals used by the global planning algorithms detailed previously. Many crowd simulation approaches rely on a finite-state machine approach to generate sequential goals for the virtual agents (Ulicny and Thalmann, 2002; Bandini et al., 2006; Curtis et al., 2016). Some alternate approaches have also been proposed to generate diverse, context-specific actions. For example, Paris and Donikian (2009) proposed an affordance-based behavior-tree approach to generate non-locomotion actions including animations of accessing machines and purchasing items. Shao and Terzopoulos (2005) described a categorical, hierarchical approach to generate diverse behaviors such as finding an empty seat, or queuing for a vending machine in a group of virtual pedestrians. Some approaches have proposed the use of decision networks or cognitive models (Funge et al., 1999; Yu and Terzopoulos, 2007). Kim et al. (2016) proposed a data-driven method to learn the high-level goals for virtual pedestrians from trajectories extracted from videos.

Local behaviors for crowd simulation typically model agent-agent interactions and locally emergent behavior. Density-sensitive behavior including aversion (Narang et al., 2015) and slower movement (Curtis and Manocha, 2012) have been used to model observed human movements. Some methods have encoded social deference by asymmetric local navigation (Curtis et al., 2012) and by adding agent proxies to represent aggressiveness and authority (Yeh et al., 2008). Other approaches have modeled personality factors influencing crowds (Guy et al., 2011; Durupinar et al., 2010) and stress-response (Kim et al., 2012).

1.1.3 Modeling Communication for Crowd Simulation

Some prior approaches have introduced communication capabilities in multi-agent systems using message passing techniques. Kullu et al. (2017) demonstrated a message packet approach based on the FIPA message structure. Huang et al. (2013) used a packet-based approach to model auditory cues in an environment such as banging. Chao and Li (2010) demonstrated a signal accumulation approach for transfer of a social contagion such as rioting. Pelechano et al. (2005) demonstrated implicit communication through the sharing of map information between agents with diverse social roles. Recently, Mordatch and Abbeel (2017) demonstrated a multi-agent system in which agents develop a symbolic language for agent cooperation in an emergent fashion using reinforcement learning.

1.1.4 Modeling Evacuations in Crowd Simulation

Prior work in evacuation modeling has demonstrated the utility of crowd simulation for designing better evacuation plans and understanding bottlenecks in evacuations. Helbing et al. (2000) studied the effect of panic on evacuations in simulation. Burghardt et al. (2012) performed stadium exit experiments with human subjects. The density and emergent phenoma displayed were replicated in simulation by Narang et al. (2015). Kim et al. (2013) modeled pushing interactions between dense crowds. Nilsson and Johansson (2009) studied evacuation experiments performed in a cinema theater. Studies on mass transit evacuation have been performed as well. Galea et al. (2011) performed experiments in the proposed blended wing body aircraft. Galea et al. (2012) examined evacuations on large sea vessels and Liu et al. (2012) performed simulations of evacuating a train tunnel.

1.2 Autonomous Driving

Similar to pedestrians, autonomous vehicles (AVs) are subject to specific physics-based and behavioral constraints. AVs are subject to kinematic constraints such as their minimum turning radius (LaValle, 2006). AVs are also subject to dynamic constraints such as their traction and braking limits (Gustafsson, 1997; Singh and Taheri, 2015). Aside from these physical constraints, AVs are subject to numerous behavioral constraints in the form of traffic laws and norms. AVs must be capable of planning paths, following signals, and safely navigating amongst other vehicles, cyclists, pedestrians, and any number of emergent and unpredictable

hazards. Furthermore, each category of interaction is governed by different laws, norms, and standards. Figure 1.2 details some of the challenging and complex situations faced by real-world vehicles.

The problem of autonomous driving has been widely studied in robotics, computer vision, intelligent transportation systems, and related areas. Extensive surveys can be found in (Pendleton et al., 2017; Katrakazas et al., 2015; Saifuzzaman and Zheng, 2014; Chen and Cheng, 2010). We limit our discussion to prior methods which address motion planning and navigation, dynamics, simulation, and collision avoidance.

1.2.1 Motion Planning for Autonomous Driving

Many techniques have been proposed to safely navigate an autonomous vehicle, termed the *ego-vehicle* along roadways. Prior approaches to path planning for autonomous vehicles have been based on occupancy grids (Kolski et al., 2006), which discretize free-space into a connected grid of free or occupied cells. Other approaches construct driving corridors (Hardy and Campbell, 2013; Ziegler et al., 2014a), which represent safe corridors a vehicle can occupy within a lane or roadway. Potential-field methods (Galceran et al., 2015b) are similar to force-based methods for crowd simulation. The vehicle experiences repulsive forces from nearby obstacles and vehicles to maintain safe distances. Some approaches employ game-theory (Sadigh et al., 2016), Bayesian behavior modeling (Galceran et al., 2015a), or path prediction (Sun et al., 2014) to frame the multi-agent navigation problem and to predict the behaviors of other drivers. Techniques to extend the velocity-obstacle formulation and enable velocity-obstacles for vehicle planning have been demonstrated. Van Den Berg et al. (2011) applied velocity-space reasoning with acceleration constraints to generate safe and collision-free velocities.

In order to plan effective paths for an autonomous vehicle, the motion planning algorithm must model the kinematic and dynamic constraints of the vehicle. A number of approaches have been developed to model these constraints, offering trade-offs between simplicity or efficiency of the approach, and physical accuracy. Simpler models are typically based on linear dynamics and analytical solutions to the equations of motion (LaValle, 2006). More accurate models provide a better representation of the physical motion, but require more computational power to evaluate and incorporate non-linear forces in the vehicle dynamics (Borrelli et al., 2005). The Reeds-Shepp formulation is a widely used car model with simple kinematics and forward and backward gears (Reeds and Shepp, 1990). Margolis and Asgari (1991) presented several representations of a car including the widely used single-track bicycle model. The single-track model represents the vehicle

Figure 1.2: **Understanding Conditions Faced by Autonomous Vehicles:**[2]**(A):** An extreme traffic jam outside of New Delhi, India. As the number of vehicles increases globally, traffic conditions will continue to become more challenging. **(B):** In many road conditions, cars navigate amongst cyclists with limited signaling and traffic control, increasing the chance for collisions. **(C):** Multi-modal traffic often includes pedestrians, cyclists, carts, and other obstructions which drivers must account for and respect. Often, behavioral norms and rules of the road form constraints on a driver's decisions.

as a single front and rear tire, neglecting load-transfer, balance, and body roll. Borrelli et al. (2005) extended this model by including detailed tire-forces.

1.2.2 Simulation for Autonomous Driving

Simulation has been an integral tool in the development of motion planners for autonomous vehicles. Simulation has played a crucial role in the development of many successful autonomous vehicle systems (Anderson et al., 2011; Likhachev and Ferguson, 2009; Aeberhard et al., 2015). Techniques have been proposed to simulate typical traffic behaviors in simulation as well, such as Human Driver Model (Treiber et al., 2006) and data-driven models such as (Hidas, 2005). Logic-based approaches with safety guarantees have also been demonstrated (Tumova et al., 2013).

Recent studies support the use of high-fidelity microscopic simulation for data-gathering and training of vision systems. Richter et al. (2016) and Johnson-Roberson et al. (2017) leveraged Grand Theft Auto 5 to train a deep-learning classifier at comparable performance to manually annotated real-world images. Other projects seek to enable video games to train end-to-end driving systems, including ChosunTruck[3]

[2]New Delhi image from The Telegraph, `https://www.independent.co.uk/news/world/asia/is-this-the-biggest-traffic-jam-ever-extraordinary-26-lane-gridlock-photographed-10495111.html`. Multi-modal traffic from Strong and Beyond, via. Youtube, `https://www.youtube.com/watch?v=mDLCd70iTeE`

[3]`https://github.com/bethesirius/ChosunTruck`

and DeepDrive-Universe[4] which leverages the OpenAi Universe system. Using video game data enables researches to gather high-quality training data including rendering and physics modeling. However, a video game based approach limits the ability to implement sensing systems and access data beyond visual data. A fully dedicated high-fidelity simulator can address these and provide access to point-cloud data, visual data, and other vehicle sensors without the limitations imposed by adapting gaming software. Research in this area suggests that high-fidelity microscopic simulators can be effective for training deep-learned end-to-end driving approaches (Dosovitskiy et al., 2017).

1.3 Thesis Statement

Our thesis statement is as follows:

Interactive motion planning for multi-agent systems can be enhanced by modeling physics-based and behavior constraints, and can generate plausible interactions and collision-free trajectories for agents of varying shapes and dynamics.

In this dissertation, we present novel algorithms to generate effective, collision-free motion for many virtual agents navigating shared virtual environments. Our algorithms seek to capture aspects of the behavioral and physics-based constraints for virtual agents. We describe representational approaches to capture specific physics-based constraints of the various agents and propose abstractions to model behavior constraints affecting them. We describe algorithms to plan motions for agents subject to these modeled constraints.

Our work builds on prior contributions in biomechanics, physics, pedestrian dynamics, and autonomous navigation to generate models which better reflect the characteristics of the modeled entities. We describe a model to enable virtual pedestrians to more accurately display high-density pedestrian movements in dense environments. We describe a model to enable virtual pedestrians to leverage verbal interaction to improve motion planning and the user's perception of the virtual agents in interactive environments. We describe a model for enabling high-speed reactive behaviors for an autonomous vehicle. We additionally describe a model for applying pedestrian simulation techniques to evaluating common phenomena observed in loading, unloading, and evacuating aircraft. Figure 1.3 demonstrates how our approaches are applied to these diverse and complex domains.

[4] `https://github.com/OSSDC/deepdrive-universe`

1.4 Main Results

1.4.1 Modeling Local Interactions in Dense Crowds using Elliptical Agents

Most prior work in pedestrian simulation relies on approximating the agents with discs in a 2D plane. Motion planning algorithms which represent pedestrians as discs are computationally efficient, but they are insufficient to model some commonly observed phenomena in dense crowds. For example, movements such as shoulder-turning, backpedaling, and side-stepping depend on the orientation of the agent. Prior studies in pedestrian dynamics, psychology and biomechanics have argued that 2D ellipses are more appropriate approximations of a human body in terms of shape and movement (Gérin-Lajoie et al., 2005; Fruin, 1971; Pauls, 2004; Chraibi et al., 2010). However, algorithms must account for increased complexity of simulation with orientable, elliptical agents.

We present a practical approach for interactive crowd simulation based on elliptical agents. Each agent is represented using a tight-fitting 2D ellipse in the plane. We use this biomechanically accurate pedestrian representation to simulate different local interactions, including backpedaling, side-stepping, and shoulder-turning.

We extend the reciprocal velocity obstacle formulation by using conservative linear approximations of ellipses and derive sufficient conditions for collision-free motion based on low-dimensional linear programming. We present efficient techniques to update the orientation to compute collision-free trajectories. Furthermore, we describe techniques to link the orientation of each elliptical agent to its velocity to automatically generate human-like turning and lateral movements. We use precomputed Minkowski Sum approximations for real-time collision avoidance in large multi-agent environments. We demonstrate that these approximations are conservative, i.e. the entire agent is guaranteed to lie within the bounding-shape, and we provide error bounds. Our approach provides significant speedups over prior algorithms for elliptical agents.

In practice, our approach can simulate dense crowds of hundreds or thousands of pedestrians at interactive rates on a single desktop CPU core.[5] We compare the runtime performance and behavior with disc-based agents on different benchmarks. We highlight the performance in complex scenarios and validate our simulation results by comparing with real-world crowd videos and experiments corresponding to dense bidirectional flows and a stadium evacuation.

[5]Results were gathered on an Intel Core I7 4790 except where otherwise noted.

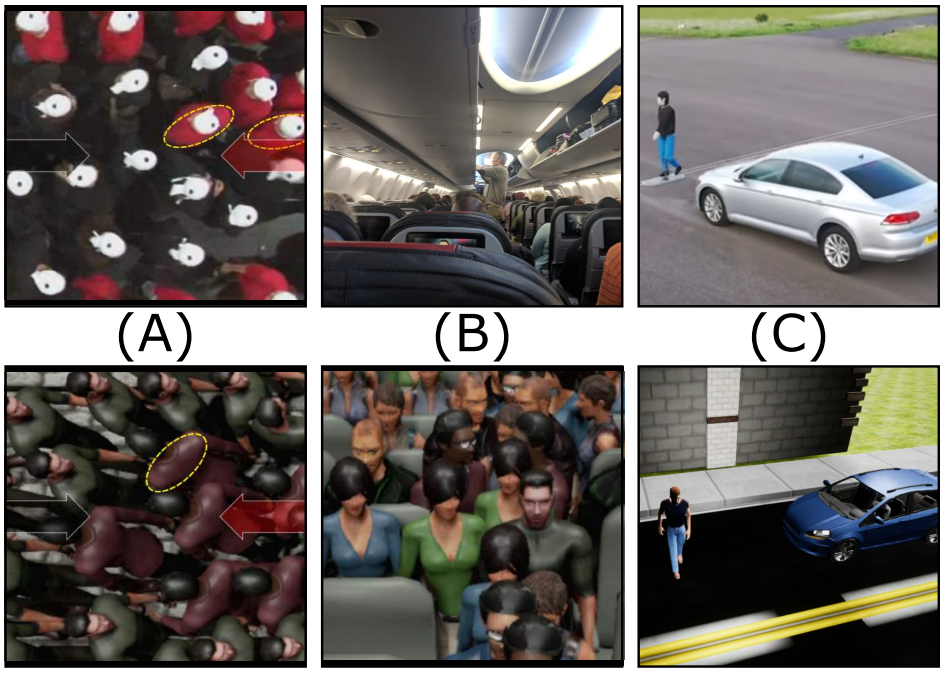


Figure 1.3: **Simulating Virtual Pedestrian and Autonomous Vehicle Behaviors:**[6]**(A): (above)** Bidirectional flow experiment performed by Zhang et al. (2012) **(below)** Virtual pedestrians simulated using our algorithms demonstrate side-stepping and shoulder-turning behaviors consistent with the captured data. **(B): (above)** Passengers load a 737 aircraft. **(below)** We demonstrate algorithmic techniques to model pedestrian behavior in loading, unloading, and evacuating aircraft. Our agents demonstrate side-stepping to reach the aisles, defering into a row to avoid other passengers, and finding and placing luggage based on variable strategies. **(C): (above)** A vehicle shown performing an emergency braking maneuver as a simulated pedestrian runs into its path. This test was performed by the European New Car Assessment Programme. **(below)** Our proposed autonomous driving algorithm performing the braking for pedestrian scenario, as modeled in our simulation framework.

1.4.2 Generating Collision-free Dynamic Maneuvers for Autonomous Vehicles

Prior work in motion planning for autonomous vehicles relies on conservative approximations of the vehicle's kinematics and dynamics, and limits the maneuver choice for the vehicle. Often, algorithms tend to limits hazard responses to either steering only (Borrelli et al., 2005), (Eidehall et al., 2007) or braking only (Distner et al., 2009). Few algorithms demonstrate combined control of throttle and steering and typically do so in constrained navigation scenarios (Turri et al., 2013). In addition, current approaches tend to limit the lane-changing behaviors, precluding their use for progressing more quickly to a goal or better utilization of the road conditions. These limitations have led to the perception that autonomous cars behave more like student drivers taking their driving test than actual skilled human drivers (Ziegler et al., 2014b).

[6]NCAP image from Euro NCAP, via YouTube, `https://www.youtube.com/watch?v=FTKxCE5qmQM`

We present a novel algorithm for autonomous vehicle navigation that supports dynamic maneuvers and integrates traffic constraints and norms. We describe a novel multi-objective optimization approach for evaluating the dynamic maneuvers. Our algorithm encodes passenger comfort, per-entity safe passing distances, maneuver constraints in terms of dynamics, and global route progress in order to compute appropriate trajectories for an autonomous vehicle. Our optimization-based maneuver planner supports dynamic lane-changes, swerving, and braking in traffic scenarios consisting of other vehicles, cyclists, and pedestrians. We take into account various traffic constraints, including collision avoidance with other vehicles, pedestrians, and cyclists using control velocity obstacles.

We encode traffic rules and road behaviors including stopping at intersections and lane-following by computing arcs along the center-line of the current lane. Our algorithm leverages the arc-approximation to generate an initial path that satisfies these constraints. We apply collision avoidance constraints and leverage an optimization-based maneuver approach to transform the initial path into the effective path the vehicle will follow.

We use a data-driven vehicle dynamics formulation that encodes feasible accelerations, steering rates, and decelerations into a set of per-vehicle profile functions, which can be quickly evaluated. These profiles are generated by simulating the vehicle through a series of trials to obtain lateral and longitudinal slip profiles. Our data-driven model generalizes to multiple vehicles and configurations.

We also present a high-fidelity simulation platform, AutonoVi-Sim, for autonomous driving data generation and driving strategy testing. Our simulation framework allows the rapid development and testing of vehicle sensor configurations and facilitates construction of complex traffic scenarios with weather variation. Our framework supports navigation for non-vehicle traffic participants such as cyclists and pedestrians in general and pre-scripted scenarios.

1.4.3 Planning Plausible Verbal Interactions between Virtual Agents and Avatars in Virtual Environments

A recent promising application of crowd-simulation is the generating of virtual characters for immersive social experiences. Such applications provide a user with the experience of embodying a digital *avatar* and sharing a virtual space with other user-controlled avatars as well as computer-controlled agents. Agents in multi-agent multi-avatar contexts must not only plan safe motions within the shared space, but be capable of engaging in meaningful interactions with avatars and other agents, either proactively or in response to

the actions of others. These interactions may include both verbal and non-verbal means of communication including movement and navigation, gesturing, gazing etc. However, prior work limits interactions to those between agents (Kullu et al., 2017) or between a single agent and single avatar (Cassell et al., 2000).

We introduce a novel least-commitment-based algorithm (Russell and Norvig, 2009) to plan motions as well as plausible verbal interactions between virtual agents and user-controlled avatars in immersive environments. The least-commitment approach allows the agent to maintain multiple concurrent plans and narrow their hypotheses as information becomes available. Our approach allows agents to generate plans with incomplete information and does not rely on complete knowledge of the environment. For example, agents may devise a plan which requires visiting a specific location without specifically knowing the path to that location. Or, an agent may plan to obtain an item and seek out the item's location from nearby agents. Our goal is to provide sufficient capability to the virtual agents to achieve plausible goals, respond to avatars as well as other agents, and plan proactive interactions when needed. We enable agents to not only respond to natural language, but to plan and ask verbal questions in order to resolve uncertainties in their chosen plan, as well generate informative responses based on their individual knowledge.

By creating plans which require interaction for uncertainty resolution, agents facilitate plausible interactions with avatars of users while accomplishing their own goals. The agents possess a semantic understanding of the environment and learn relationships based on their natural-language interactions as well as observations, i.e. object proximity. We demonstrate benefits of our approach in a set of simulations with tens of virtual agents that: proactively generate plans; interact with avatars and other agents; and ask and respond to questions verbally. We validate the benefits of our algorithm based on a user study simulating an immersive virtual environment and show that our method provides significant improvement over prior methods in the plausibility of simulations and the overall effectiveness of agent-avatar interactions. Participants in our study found that compared to a prior method, our approach generated significantly more plausible simulations and that the presence of our natural-language interactions was a significant positive factor on their preferences. We also observed that agents using our approach were able to complete their goals more quickly and effectively than prior approaches.

1.4.4 Evaluation of Pedestrian Models for Loading, Unloading, and Evacuating Aircraft

Prior work has demonstrated the efficacy of multi-agent simulation to evacuation modeling in large, open spaces. However, the use of agent-based models for constrained spaces, such as an aircraft, has been

proven to be challenging. Passengers in aircraft display a large range of dispositions and intentions. Behavior and motion planning aboard the aircraft is also inherently social, as the agents are governed not only by their own goals, but by customs and social norms. For example, during unloading, flight staff will assist infirm passengers, and during loading, passengers will temporarily move out of the aisle seat to allow another passenger to sit in a neighboring seat. Furthermore, the density of the aircraft makes motion planning especially sensitive to poor motion planning decisions. Agents with opposing goals, such as one traveling up the aisle to place luggage and the other down the aisle to their seat, can easily create deadlocks without techniques to resolve the head-on collision. An effective simulation aboard the aircraft must account for these phsyics-based and behavior constraints when generating plans for the virtual agents.

We demonstrate the use of our proposed algorithms for effective multi-agent simulation to model passenger behaviors in loading, unloading, and evacuating commercial aircraft. We have constructed a framework (Ped-Air), which not only addresses the navigation issues in these scenarios, but incorporates the behavior and characteristics of and interactions between passengers aboard an aircraft. We highlight the performance of Ped-Air across different scenarios corresponding to typical loading with and without luggage, unloading with and without luggage, and evacuation with obstructed exits. We also demonstrate the benefits of our approach to reducing deadlocks and generating more plausible agent behavior aboard the aircraft.

1.5 Organization

The rest of the dissertation is organized as follows:

Chapter 2 presents our novel elliptical pedestrian formulation and motion planning algorithm, and demonstrates its efficacy for planning safe trajectories for hundreds of agents and replicating captured human data.

Chapter 3 presents our optimization-based maneuver planning approach for autonomous driving subject to traffic rules and physics-based constraints, and demonstrates its efficacy on a set of diverse road conditions.

Chapter 4 presents our approach for generating motions and natural-language interaction behaviors for virtual agents in the presence of other agents and avatars, and demonstrates its positive impact on multi-agent multi-avatar interaction plausibility.

Chapter 5 Demonstrates applications of our proposed pedestrian algorithms to the simulation of loading, unloading, and evacuating commercial aircraft.

Chapter 6 concludes the dissertation, offers limitations and potential future challenges in the field of motion planning for multi-agent systems.

CHAPTER 2: E-ORCA: Real-time Reciprocal Collision Avoidance with Elliptical Agents

2.1 Introduction

Pedestrian and crowd simulation has received considerable attention in different fields. Besides computer graphics, there is extensive work in biomechanics, psychology, robotics, and pedestrian dynamics. Biomechanics researchers evaluate various factors related to human locomotion; psychologists investigate the spatial and behavioral relationships of pedestrians and how those relationships affect their movement; robotics and vision researchers study various aspects of locomotion, including path planning, collision-avoidance, spatial flow, and visual appearance; the pedestrian dynamics community is interested in modeling the pedestrian flows for architectural design and evacuation planning. Many ideas from these fields are also used in computer graphics, including interactive crowd simulation for games and virtual reality, and are employed in generating special effects for movies and animation.

Some widely used crowd simulation algorithms are based on multi-agent models. In these simulations, each individual is represented as an independently sensing, planning, and moving *agent*. However, simulating the behavior and movement of crowds can be challenging. Many applications model large crowds consisting of hundreds or thousands (or more) agents. Simple, individual movements can result in emergent crowd behaviors, and it is important to simulate these collective patterns. Furthermore, many interactive applications must perform these computations in tens of milliseconds or less. Given these challenges, most prior work in crowd simulation uses a simple representation for each agent as a 2D circular disc in a plane. There is extensive work on computing agent trajectories and behaviors based on 2D disc representation using force-based (Helbing et al., 2000; Karamouzas et al., 2014), velocity-based (Van den Berg et al., 2011; Pettré et al., 2009) rule-based (Reynolds, 1987), and cell-based (Schadschneider, 2002) methods, among others. Most commercial and research systems for pedestrian and crowd simulation use such disc shapes and navigation algorithms.

Human pedestrians exhibit a large variety of behaviors and local interactions in tight spaces and dense situations, including movements and salient behaviors that are used to avoid other agents or obstacles and

navigate through congestion and constriction (Imanishi and Sano, 2014; Hughes et al., 2014). It is not uncommon for pedestrians to twist their shoulders and torsos to reduce their profile when passing one-another. They also perform side-step movements or backpedal to avoid unexpected obstacles and make way for others. Current crowd simulation algorithms are unable to model many of these interactions using radially-symmetric discs. Furthermore, disc-based representations tend to overestimate the volume exclusion and restrict the maximum density in a crowd simulation.

Movements such as shoulder-turning, backpedaling, and side-stepping depend on the orientation of the agent. Some researchers impose additional restrictions with radially symmetric discs, including applying an implicit orientation used only to model these complex behaviors or one that assumes that the agent is always facing along its velocity vector (Karamouzas et al., 2009; Johansson et al., 2007). However, these techniques tend to be non-intuitive and are not able to generate many of the orientation-specific local interactions. Furthermore, rendering scenes with such disc-based agents can be difficult because artificial orientation constraints can result in inconsistent and erratic animation. Some of these rendering and foot-skating problems can be overcome using footstep planning (Berseth et al., 2015), although most footstep models tend to be non-convex and dynamic.

Prior studies in pedestrian dynamics, psychology and biomechanics have argued that 2D ellipses are more appropriate approximations of a human body in terms of shape and movement (Gérin-Lajoie et al., 2005; Fruin, 1971; Pauls, 2004; Chraibi et al., 2010). Ellipses are orientable shapes and the orientation information can be used to model local interactions. However, the use of elliptical shapes for crowd simulation typically yields many navigational and computational challenges. Force-based methods (Karamouzas et al., 2009; Helbing et al., 2000; Karamouzas et al., 2014) may encounter non-radial forces as the closest point between two arbitrarily oriented ellipses is often not along the line connecting their centers, which can vary agent behaviors based on their orientation. Velocity obstacle based techniques (Van den Berg et al., 2011) perform geometric optimization in velocity space to compute collision-free trajectories. These methods must compute the Minkowski sums (De Berg et al., 2008) of two ellipses, which is considerably more expensive than the Minkowski sum of discs. Cell-based methods (Schadschneider, 2002; Burstedde et al., 2001) require computing complex decompositions of 2D planes into complementary and overlapping grids, while updating them for dynamic obstacles.

Main Results: We present an efficient agent-based crowd simulation algorithm for collision-free navigation of elliptical agents in dynamic environments. We use a decentralized approach based on velocity obsta-

cles (Van den Berg et al., 2011; Fiorini and Shiller, 1998) and present fast algorithms for conservative collision avoidance. To overcome the complexity of exact Minkowski Sum computation, we use conservative linear approximations of ellipses and reduce the collision avoidance problem to solve a lower dimensional linear programming problem and show that our formulation is reciprocally maximal. To ensure that we can handle large environments consisting of hundreds or thousands of agents in real-time, we use precomputed tables of Minkowski Sums and guarantee that feasible trajectories computed using our algorithm would be collision-free. In practice, the use of conservative linear approximations and precomputed Minkowski Sums tables improves the runtime performance by more than an order of magnitude. We also present a technique to update the orientation of ellipses in dense environments to compute collision-free trajectories among static obstacles or to generate human-like trajectories, including computing side-stepping and shoulder-turning movements based on empirical observations. Our overall algorithm can perform collision-free navigation of thousands of agents at interactive rates on a single core and is only $4 - 5$X slower than collision avoidance algorithms for disc-based agents. Compared with prior methods, some of the contributions of our work include:

1. Generating collision-free trajectories for hundreds or thousands of elliptical agents at interactive rates.
2. A pre-computation approach to reduce the computational complexity of simulating elliptical agents by an order of magnitude.
3. Simulating human-like local interactions and collision-avoidance behaviors, including side-stepping, shoulder-turning, and backpedaling.
4. A method to link the orientation computation to velocity, which reduces the simulation cost and dimensionality of motion planning.
5. Validation with pedestrian experiments including comparison to video data and behavior classification experiments.

Our overall formulation is simple. As compared to other geometric algorithms (Lee et al., 1998) for collision avoidance between elliptical agents, our overall algorithm for local collision avoidance and orientation computation results in *one order* of magnitude performance improvement. We demonstrate the effectiveness of our algorithm by testing it on multiple, dense scenarios. We also validate our method by reproducing the trajectories observed in real-world pedestrian videos and comparing emergent behaviors and interactions with captured pedestrian data.

The remainder of the chapter is organized as follows. In Section 2.2, we provide a brief overview of prior work in crowd simulation and the use of orientable shapes. In Section 2.3, we introduce the notation and describe the elliptical representation. In Section 2.4, we describe local collision-avoidance algorithm, and in Section 2.5 our approach for accelerating our method with pre-computation. In Section 2.6, we describe our approach to update the orientation of an elliptical agent. The overall crowd simulation algorithm and implementation details are described in Section 2.6.5. We highlight the performance of our algorithm and compare the results with real-world crowd videos and prior simulation methods in Sections 2.7 and 2.8 and conclude in Section 2.9.

2.2 Related Work

In this section, we provide a brief overview of prior work in planning collision-free trajectories, crowd simulation, and the usee of elliptical shapes for modeling agents in pedestrian dynamics and robotics.

2.2.1 Robotics

At a broad level, prior algorithms for collision-free navigation in robotics can be classified as centralized and decentralized. Centralized planners treat the set of all robots as a single system in a (very) high-dimensional configuration space, and many well-known methods for single-robot motion planning can be used (LaValle et al., 1998; Sanchez and Latombe, 2002; Švestka and Overmars, 1998). These planners have the advantage of being complete (in theory), but practical algorithms are limited to systems with a few robots. Decentralized planners tend to compute a path for each robot or agent independently, and use some coordination mechanism or local navigation techniques to avoid collisions between them. Many techniques have also been proposed for collision-avoidance, navigation, and planning among moving obstacles (Fox et al., 1997; Hsu et al., 2002; Jaillet and Simeon, 2004; Petty and Fraichard, 2005), and these can be extended to elliptical agents. Given an agent, all the other agents can be treated as dynamic obstacles along with other objects in the scene. However, these methods do not consider the reactive behavior of other agents as part of multi-agent planning.

Many decentralized planners tend to rely on high-level planning modules to generate paths through the static environment and on local collision-avoidance algorithms to adapt those plans to the environment. Priority-based methods assign order to the robots, and plan the paths sequentially (Erdmann and Lozano-

Perez, 1987). Velocity-obstacle based methods compute locally collision-free velocities in velocity space (Van den Berg et al., 2008a; Guy et al., 2009; Van den Berg et al., 2011; Fiorini and Shiller, 1998). Whereas many local collision-avoidance algorithms take advantage of the disc representation of the robots, some algorithms also attempt to model additional dynamics constraints, such as differential-drive (Alonso-Mora et al., 2013), double-integrator (Lalish and Morgansen, 2012), car-like (Alonso-Mora et al., 2012), or linear (Bareiss and Van den Berg, 2013), etc.

2.2.2 Crowd Simulation

There has been extensive work in simulating crowds or pedestrians in the fields of computer graphics and pedestrian dynamics for more than three decades. Some widely used approaches are based on cellular-automata (Schadschneider, 2002; Burstedde et al., 2001), force-based computations (Helbing et al., 2000; Pelechano et al., 2007; Karamouzas et al., 2014), field-based methods (Patil et al., 2010; Sung et al., 2004), geometric computations based on velocity obstacles (Van den Berg et al., 2011), steering models (Reynolds, 1987), vision-based (Ondřej et al., 2010), continuum techniques (Treuille et al., 2006; Narain et al., 2009; Kapadia et al., 2009), cognitive models and decision networks (Funge et al., 1999; Yu and Terzopoulos, 2007), and data-driven simulation (Lee et al., 2007; Lerner et al., 2007; Pettré et al., 2009), among other approaches. These methods model different aspects of crowd simulation, including local collision avoidance, emergent behaviors, high-level behavior modeling, etc. However, almost all these methods model each agent as a 2D circle or disc on a plane to compute its trajectory or behavior. Most of these efficient local navigation and collision-avoidance methods exploit the fact that the agent can be conservatively approximated by a disc.

2.2.3 Lateral Motion and Orientable Shapes

Earlier work in pedestrian dynamics and biomechanics has argued for the use of elliptical shapes. Fruin (1971) proposed the use of elliptical pedestrians for planning and design. Borghese et al. (1996) described a relationship between human kinematics and motion. Gérin-Lajoie et al. (2005) demonstrated the relationship between stride-length, density, and pedestrian movement shape to make a case for elliptical personal space. Templer (1995) showed the presence of the sensory zone around a pedestrian using elliptical models. Imanishi and Sano (2014) showed the importance of side-stepping and shoulder-turning in avoidance behaviors from laboratory experiments. Singh et al. (2011) present an algorithm for human-like steering in dynamic crowds

based on a simple biomechanical footstep model and combine it with space–time planning to generate subtle navigation behaviors, including side-stepping.

There has been some research performed with respect to representing each agent as an orientable shape. Giese et al. (2014) used the concept of reciprocal rotation to extend the reciprocal velocity obstacle (RVO) algorithm to polygons. Chraibi et al. (2010) applied social forces to deformable discs to simulate ellipse-like local behaviors. Johansson et al. (2007) applied elliptical forces to disc-shaped agents. Karamouzas and Guy (2015) used velocity obstacles to navigate disc-based agents in deformable orientable formations. There is some work on modeling lateral agent motion in crowd simulation. Hughes et al. (2014) used pedestrian experiments to generate discrete side-stepping interruptions. Choi et al. (2011) used precomputed motion patches to generate appropriate animation in complex environments. Van Basten et al. (2011) used interpolation techniques to generate parametrized footsteps for articulated characters. Our approach is based on explicit representation of 2D elliptical agents and differs from these methods.

2.3 Elliptical Agent Representation

In this section, we introduce our notation and describe our elliptical representation and simulator state evolution.

2.3.1 Elliptical Representation

We assume that the underlying multi-agent simulation algorithm uses a high-level module that computes a new preferred velocity for each agent during each step of the simulation. The preferred velocity is the velocity that optimally leads the agent toward its current goal if there were no other agents or obstacles in the scene. The current velocity is the actual velocity computed at the current time step to avoid collisions with the obstacles and other agents in the scene. Our agents are represented in terms of their shape dimensions, current position, current velocity, and preferred velocity.

Notation: For an agent A, $\vec{p}_A$, $\vec{v}_A$, and $\vec{v}_A^0$, denote the current position of the agent's center of mass, current velocity, and preferred velocity. Each elliptical agent is represented using the following 10-dimensional state vector $[\vec{p}, \vec{v}, \vec{v}^0, \vec{o}, s^{maj}, s^{min}]$, where $\vec{o}_A$ denotes the agent's orientation. Following Brscic et al. (2013), we define orientation as "the angle of the vector perpendicular to the line connecting the shoulders and directed towards the front of the person". s_A^{maj} and s_A^{min} denote the length of the agent's semi-major and semi-minor

axes, respectively (see Figure 2.1). By contrast, a circular agent X is represented as a 7-dimensional vector $[\vec{p}_X, \vec{v}_X, \vec{v}_X^0, r_X]$, where r_X is its radius. In the rest of the chapter, we use symbols A and B to denote elliptical agents and X and Y to denote disc-shaped agents. For some computation, it can be useful to represent the agent's orientation as a scalar quantity. For such computation. we use o, defined as the angle between the agent's orientation vector and the positive x-axis in the range $(-\pi, \pi]$. Our approach can also be extended to robots or agents with larger configuration spaces. Let $\mathbb{R}^d$ represent the physical workspace of the robots or agents, where $d \geq 2$. We project the geometric representation $\mathcal{O}^d$ of the robot in $\mathbb{R}^d$ space to $\mathbb{R}^2$, represented by $\mathcal{O}^2$, and bound it with an ellipse in $\mathbb{R}^2$.

Disc-based techniques often use a diameter that reflects the average shoulder width of a pedestrian (Helbing et al., 2000). However, it has been shown that an elliptical approximation (Weidmann, 1993) is more appropriate for representing human motion, particularly in cases of high-density or tight spaces, which is illustrated in Figure 2.1; disc-based pedestrians overestimate the volume exclusion substantially more than a corresponding ellipse. Moreover, the use of discs restricts the maximum density and prevents the simulated crowds from reaching real measured densities.

2.3.2 Simulator State

We assume that there is a high-level module $\mathcal{K}_i : t \times \mathbb{S} \rightarrow \mathbb{R}^2$ that maps the time t and simulator state $\mathbb{S}$ into an instantaneous preferred velocity that can be expressed as the composition of simpler functions such as

$$\mathcal{K}_i(t) = \mathcal{P}_i(\mathcal{G}_i(t)), \tag{2.1}$$

where $\mathcal{G}_i : t \times \mathbb{S} \rightarrow \mathbb{R}^2$ maps the time and simulator state into a goal position and $\mathcal{P}_i : \mathbb{S} \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that maps the simulator state and the agent's goal position into a instantaneous preferred velocity for agent i, denoted by $\vec{v}_i^o$. The function $\mathcal{K}_i$ computes the collision-free path to the goal with respect to static obstacles in the simulation. Let $\mathcal{LCA}_i : \mathbb{S} \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2 \times \mathbb{R}^2$ denote a local collision avoidance function that maps the simulator state, the instantaneous preferred velocity, the instantaneous preferred orientation, and time horizon, τ, into a collision-free orientation (o_i) and velocity ($\vec{v}_i$) with respect to other robots and obstacles in the environment for at least time τ.

Our first goal is to formulate a generalized local collision avoidance function $\mathcal{LCA}$ for elliptical agents, which seeks to independently and simultaneously compute a velocity $\vec{v}_i$ and orientation $\vec{o}_i$ for each elliptical

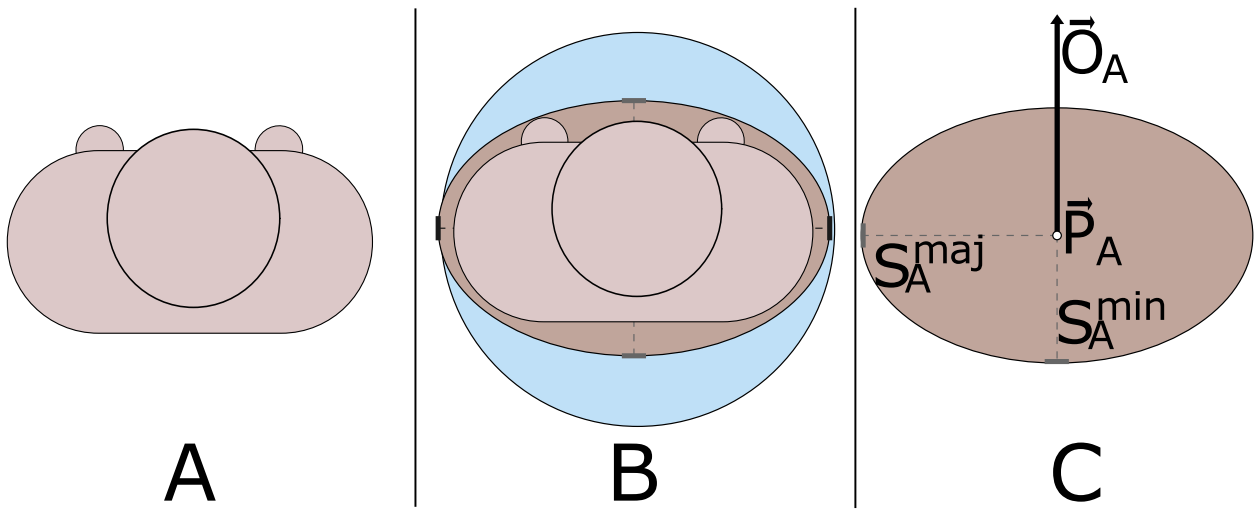


Figure 2.1: **Approximating Pedestrians**. (A). A sample human pedestrian of approximately average shoulder-width and body depth, 0.228 m and 0.149 m, respectively. (B) The typical disc-based approximation (blue) of the human overestimates its depth by over 50%. Our proposed elliptical approximation (brown) captures the pedestrian shape accurately and can keep track of the orientation. (C) Representation: Each agent A's state vector consists of position $\vec{p}_A$, orientation $\vec{o}_A$, current velocity $\vec{v}_A$, preferred velocity $\vec{v}_A^o$, semi-major axis s_A^{maj} and semi-minor axis s_A^{min}.

robot i in the simulation. Thus, the instantaneous velocity and orientation of an agent can be given by:

$$(\vec{v}_i(t), \vec{o}_i(t)) = \mathcal{LCA}_i(\mathcal{H}_i(\mathcal{P}_i(\mathcal{G}_i(t)))), \tag{2.2}$$

where $\mathcal{H}_i : \mathbb{S} \times \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^2$ maps the simulator state, preferred velocity, and time to the preferred orientation. The function $\mathcal{LCA}$ must guarantee that all elliptical agents can follow a collision-free trajectory with the new configuration for at least a predefined horizon window of time τ. Furthermore, the agent's new velocity should be as close as possible to its preferred velocity.

Given the preferred velocity, the underlying approach uses a local collision-avoidance technique to compute a new velocity, $\vec{v}^{new}$, and orientation, $\vec{o}^{new}$, for each elliptical agent. Furthermore, each agent has access to the directly observable properties of its neighbors to compute its current velocity. These properties include $\vec{p}$, $\vec{v}$, $\vec{o}$, s^{maj} and s^{min}. However, properties such as $\vec{v}^0$ are considered internal or intrinsic to the agent and cannot be observed by other agents. We assume that all the agents in the simulation are using the same local collision-avoidance and navigation strategies. To ensure consistency, the current velocity $\vec{v}$, is set to the new velocity $\vec{v}^{new}$ for each agent simultaneously at the end of the simulation step. Likewise, the current orientation $\vec{o}$, is set to the new orientation $\vec{o}^{new}$.

2.4 Local Collision Avoidance

In this section, we present our local collision-avoidance algorithm for elliptical agents. Our algorithm is based on velocity obstacles that have been frequently used for collision-avoidance in robotics and crowd simulation (Fiorini and Shiller, 1998; Van den Berg et al., 2011). However, as with most prior work in multi-agent navigation, these methods are limited to agents represented as 2D discs.

Velocity Obstacles: For two agents, A and B, centered at $\vec{p}_A$ and $\vec{p}_B$, respectively, the velocity obstacle of A induced by B, is denoted by $VO^{\tau}_{A|B}$. The velocity obstacle constitutes the set of velocities for A that would result in a collision with B at some time before τ. By definition, agents A and B are guaranteed to be collision-free for at least time τ, if $\vec{v}_A - \vec{v}_B \notin VO^{\tau}_{A|B}$ (Fiorini and Shiller, 1998). More formally,

$$VO^{\tau}_{A|B} = \{\vec{v} \mid \exists t \in [0, \tau] :: \vec{p}_A + t(\vec{v}_A - \vec{v}_B) \in M\}, \tag{2.3}$$

where $t \geq 0$ and M denotes the Minkowski Sum of $-A$ and B.

In general, let V denote the set of all velocities for an agent. At each simulation step, the agent must choose a velocity $\vec{v}^{new} \in V$ s.t. $\vec{v}^{new}$ lies outside the velocity obstacles defined by all the neighboring agents and obstacles, which is a sufficient condition for collision-free navigation for at least time τ.

In the case of disc-shaped agents, the Minkowski Sum for two agents can be implicitly computed with a few floating point operations. Finding the closest point on a disc from a query point is also trivial. For elliptical agents, these operations are non-trivial. Computing the Minkowski Sum requires either computing convolution curves, which is considerably more expensive (Lee et al., 1998), or using a closed-form implicit equation (Yan and Chirikjian, 2014). Moreover, computing the closest points on two arbitrarily oriented ellipses requires computing the roots of a fourth-order polynomial. These operations are costly, so we present faster algorithms based on conservative linear approximations.

2.4.1 Approximating Ellipses

Instead of computing the exact Minkowski Sum of two ellipses, we use a piece-wise linear approximation (PL) that can provide conservative guarantees for collision avoidance. For an ellipse $\mathcal{C}$, a piece-wise linear approximation can be computed by uniformly sampling $\mathcal{C}$ at intervals of $\delta\theta \in (0, 90)$ to yield the set of

sample points:

$$\mathcal{S} = \{(s_{maj} \times cos(\delta\theta \times i), s_{min} \times sin(\delta\theta \times i) : 0 \leq i \leq \lfloor \frac{2\pi}{\delta\theta} \rfloor | i \in \mathbb{I}\}. \tag{2.4}$$

Let $\mathcal{L} = \{\lambda_{\vec{p}} | \vec{p} \in \mathcal{S}\}$ denote the set of tangents to the curve $\mathcal{C}$, defined at each sample point as:

$$\lambda_{\vec{p}} = \mathcal{C}(\vec{p}) + t\mathcal{C}'(\vec{p}) \; \forall \; \vec{p} \in \mathcal{S}, \tag{2.5}$$

where $t \geq 0$. We can now define the set of vertices $\mathcal{V}$ of the bounding polygon by solving $\lambda_{\vec{p}_A} = \lambda_{\vec{p}_{A+1}}$ i.e., the point of intersection of two tangents where $\vec{p}_A, \vec{p}_{A+1} \in \mathcal{S}$.The computational cost is thus $O(m)$ for m samples. Next, we use this property to show that the linear approximation computes a conservative shape for collision avoidance.

Theorem 1: *For any ellipse $\mathcal{C}$, a piecewise linear approximation $\mathcal{L}$ of the curve defined by the tangents at the sampled points overestimates the curve.*

Proof: Let $\mathcal{C}$ denote an axis-aligned ellipse defined at the origin as:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \tag{2.6}$$

where $a^2 > b^2$ and $a, b \in \mathbb{R}^+$. The first and second derivative of the curve, $\dot{\mathcal{C}}$ and $\ddot{\mathcal{C}}$ resp., are given by:

$$\frac{dy}{dx} = -\frac{b^2 x}{a^2 y}, \tag{2.7}$$

$$\frac{d^2 y}{dx^2} = -\frac{b^4}{a^2 y^3}. \tag{2.8}$$

It follows that:

$$\begin{cases} \ddot{\mathcal{C}} > 0 & \text{if } y < 0, \\ \ddot{\mathcal{C}} < 0 & \text{if } y > 0. \end{cases}$$

By definition, $\mathcal{C}$ is concave downward for $y > 0$, which implies that the tangent lines $\lambda_{\vec{p}} \in \mathcal{L}$ lie above the curve when $\vec{p}$ lies in the first or second quadrant. Similarly, $\mathcal{C}$ is concave upward for $y < 0$, which implies that the tangent lines $\lambda_{\vec{p}} \in \mathcal{L}$ lie below the curve in the interval when p lies in the third or fourth quadrant.

Hence, the linearization $\mathcal{L}$ overestimates the curve $\mathcal{C}$. Without loss of generality, the same proof can be used for oriented ellipses.

Theorem 2: *We are given an ellipse $\mathcal{C}$ in the form $y = \mathcal{C}(x)$ and a piecewise linear approximation $\mathcal{L}$ of the curve defined by the tangents at the ordered set of samples $\mathcal{S}$. Let $\lambda_A \in \mathcal{L}$ denote a linearization centered at $\vec{p}_A = (x_A, \mathcal{C}(x_A))$ that approximates the curve in the interval $\mathcal{I} = (\vec{p}_A, \vec{p}_j)$ where $\vec{p}_A = (x_A, \mathcal{C}(x_A))$ and $\vec{p}_j = (x_j, \mathcal{C}(x_j))$. Then the error bounding function $B(t)$ over $\mathcal{I}$ can be given by :*

$$\mathcal{B}(x_s) = \frac{K}{(1 - \frac{x_s^2}{a^2})^{3/2}}, \tag{2.9}$$

where $K = \frac{b(x_A - x_j)^2}{2a^2}$ and $x_A < x_S < x_j$. Furthermore, the maximum approximation error, $\mathcal{E}_{max}$, can be exactly computed.

Proof: As before, let us consider an axis-aligned ellipse defined at the origin. We also assume that the set of samples $\mathcal{S}$ includes the points where $y = 0$. Let $\mathcal{E}(x_j)$ represent the approximation error for point $(x_j, \mathcal{C}(x_j))$, which is expressed as:

$$\mathcal{E}(x_j) = \mathcal{C}(x_j) - \lambda_A|_{x=x_j} \tag{2.10}$$

$$\lambda_A|_{x=x_j} \geq \mathcal{C}(x_j) \, \forall \, x_j \in \mathbb{R} \quad (Theorem 1) \tag{2.11}$$

$$\therefore \mathcal{E}(x_j) \leq 0 \tag{2.12}$$

It can be proven that there is some $x_s \in \mathcal{I}$ such that:

$$\mathcal{E}(x_j) = \frac{\ddot{\mathcal{C}}(x_s)}{2}(x_j - x_A)^2. \tag{2.13}$$

We can derive an expression for the error-bounding function using 2.6 , 2.8, & 2.12, as:

$$B(x_s) = |\mathcal{E}(x_j)| = \frac{K}{(1 - \frac{x_s^2}{a^2})^{3/2}} \tag{2.14}$$

Differentiating $B(x_s)$ gives:

$$\dot{B}(x_s) = \frac{3K}{a^2(1 - \frac{x_s^2}{a^2})^{\frac{5}{2}}} x_s. \tag{2.15}$$

It follows that the error bound is monotonically increasing when $x_s > 0$ i.e. $x_A, x_j \in \mathbb{R}^+$. Similarly, it is monotonically decreasing when $x_s < 0$ i.e. $x_A, x_j \in \mathbb{R}^-$. Thus the maximum approximation error, $\mathcal{E}_{max}$, over the open interval $\mathcal{I} = (\vec{p}_A, \vec{p}_j)$ can be expressed as:

$$\mathcal{E}_{max} = \begin{cases} \left.\frac{3K}{a^2(1-\frac{x_s^2}{a^2})^{\frac{5}{2}}} x_s\right|_{x_s=x_j} & \text{if } x_A, x_j \in \mathbb{R}^+, \\ \left.\frac{3K}{a^2(1-\frac{x_s^2}{a^2})^{\frac{5}{2}}} x_s\right|_{x_s=x_A} & \text{if } x_A, x_j \in \mathbb{R}^-. \end{cases}$$

2.4.2 Velocity Obstacles For Elliptical Agents

In this section, we extend the ORCA algorithm (Van den Berg et al., 2011) to elliptical agents based on the linear approximation. We refer to the new algorithm as EORCA.

2.4.2.1 Computing Neighboring Agent Constraints

To compute the velocity obstacle for an elliptical agent, we first compute a tangent from the origin of the velocity space to the boundaries of the Minkowski Sum scaled by the inverse of τ. For a Minkowski Sum with m samples, we can find tangents in $O(lg(m))$ using binary search on the vertices. The forward face of the truncated cone lies between the tangent points, and the nearest point can be computed by another binary search. Figure 2.2(A, B) illustrates the construction of the velocity obstacle of the elliptical agents using the Minkowski Sum and the tangents.

Next, we use the velocity obstacle to compute the set of permitted velocities for an agent. Given the velocity obstacle, $VO^{\tau}_{A|B}$, we construct the set of permitted velocities for A for reciprocal collision avoidance (Van den Berg et al., 2011), which is denoted as EORCA$^{\tau}_{A|B}$. Consider a vector $\vec{u}$ from the relative velocity $\vec{v}_A - \vec{v}_B$ to the nearest point on the truncated velocity obstacle. Also, let $\vec{n}$ be the outward normal of the boundary of $VO^{\tau}_{A|B}$ at point $(\vec{v}_A - \vec{v}_B) + \vec{u}$. We assume that all agents use the same collision-avoidance strategy. Therefore, agent A is responsible for adapting its velocity by $\frac{1}{2}\vec{u}$ assuming that B will do the same. In this manner, the set $EORCA^{\tau}_{A|B}$ of permitted velocities for A is defined by the half-plane pointing in the direction of $\vec{n}$ centered at the point $\vec{v}_A + \frac{1}{2}\vec{u}$ (see Figure 2.2(C)). Hence, one half-plane constraint is constructed for each neighboring agent.

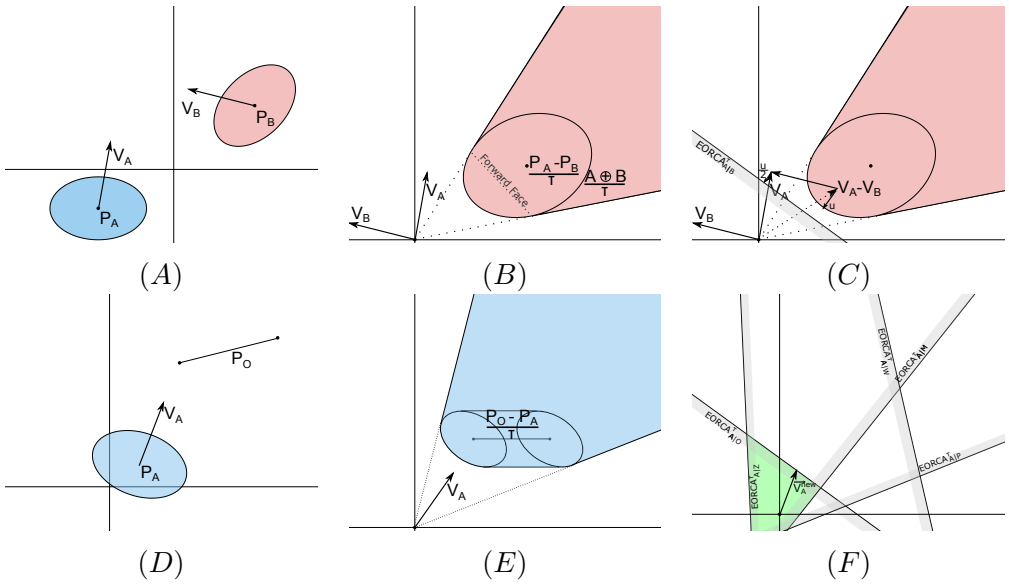


Figure 2.2: **Computing Collision-free Velocity** (A) Two agents are moving toward one another in space. The approximating polygons are shown. (B) The velocity obstacle for A induced by B takes the shape of a truncated cone. The apex of the cone is located at the origin in velocity space. The forward arc of the cone is the forward face of the Minkowski Sum of $-A$ and B scaled by τ and centered at $\vec{p}_B - \vec{p}_A/\tau$. (C) The set of permitted velocities for agent A w.r.t. B represented by the half-plane constraint $EORCA^{\tau}_{A|B}$. (D) An agent moves toward a line segment obstacle. (E) To compute $VO_{A|O}$, we rotate the frame of reference such that O is parallel to the positive x-axis. In this case, the forward arc of the cone is the forward face of the Minkowski Sum of $-A$ and O scaled by τ and centered at $\vec{p}_O - \vec{p}_A/\tau$. We also show the EORCA constraint as a half-plane perpendicular to the vector connecting v_A and $VO_{A|O}$ and passing through the nearest point on $VO_{A|O}$. After the constraint is computed, it is rotated back to the axis-aligned reference frame. (F) After all the constraints have been computed, we can determine a feasible velocity inside the union of all the EORCA sets. A new velocity is chosen from the region of feasible velocities.

2.4.2.2 Computing Neighboring Obstacle Constraints

Without loss of generality, we can assume that all obstacles in the scene are triangulated and their projections on the 2D plane are given as a collection of line segments. To compute the velocity obstacle $VO^{\tau}_{A|O}$ for agent A induced by a line segment O, we implicitly compute the Minkowski Sum of $-A$ and O scaled by the inverse of τ. An agent A will collide with obstacle O within the time τ if its velocity $\vec{v}_A$ is inside $VO^{\tau}_{A|O}$. Geometrically, the Minkowski Sum is equivalent to sliding the reflected A along the scaled line segment (each scaled by τ). Because discs are unaffected by orientation issues, the computations related to determining the shape of the velocity obstacle, finding tangents, and determining the closest point on the obstacle can be implemented by using a small number of floating point operations. With elliptical agents, the shape of the velocity obstacle, the tangents, and the nearest point operations are governed by the orientation of the agent as well as of the obstacle.

Let $-\theta \in [0, -2\pi]$ represent the angle between O and the positive x-axis. We can simplify the computation by rotating the coordinate system by θ, i.e., we rotate O and agent A by θ and compute the appropriate constraints in the transformed space. Let $\vec{o}_A^{rot}$ denote the orientation of the ellipse after the rotation transformation. For the remainder of this section, we can assume that O is parallel to the positive x-axis. Therefore, the shape of the velocity obstacle depends only on the orientation $\vec{o}_A^{rot}$ of the agent. We can also accelerate the computation of tangents and the closest points (Section 2.5).

Once rotated, we determine whether the agent's velocity projects onto the left tangent, right tangent, left face, right face, or line segment. Next, we compute the point $F_{A|O}$ on the velocity obstacle closest to $\vec{v}_A$. The half plane defined using the tangent to the velocity obstacle at $F_{A|O}$ yields the set $\text{EORCA}^{\tau}_{A|O}$ of permitted velocities for A with respect to O (Figure 2.2(E)). We rotate the final computed constraint by $-\theta$ to transform back into the original space (Figure 2.2(F)).

2.4.2.3 Choosing a Collision-free Velocity

At every simulation step, we construct the half-plane constraints for each neighboring agent and obstacle. The set of neighboring agents and obstacles can be computed efficiently by using a spatial data structure, such as a kD-tree. The set of permitted velocities for agent A is simply the convex region – EORCA^{τ}_{A} – given by the intersection of the half-planes of the permitted velocities that are induced by all the neighboring agents and obstacle (Figure 2.2(F)).

$$\text{EORCA}^{\tau}_{A} = \bigcap_{B \neq A} \text{EORCA}^{\tau}_{A|B} \tag{2.16}$$

The agent is responsible for selecting a new velocity v_A^{new} from EORCA^{τ}_{A} that minimizes the deviation from its preferred velocity $\vec{v}_A^0$.

$$\vec{v}_A^{new} = \underset{\vec{v} \in \text{EORCA}^{\tau}_{A}}{\arg\min} \|\vec{v} - \vec{v}_A^0\| \tag{2.17}$$

Equation (2.16) and Equation (2.17) can be solved efficiently with an expected runtime of $O(n)$ using linear programming, where n is the total number of constraints.

2.4.2.4 Collision-free Guarantees

If the linear programming algorithm can compute a feasible solution for each agent, we can guarantee that the resulting trajectories will be collision-free. This follows from our conservative, bounding

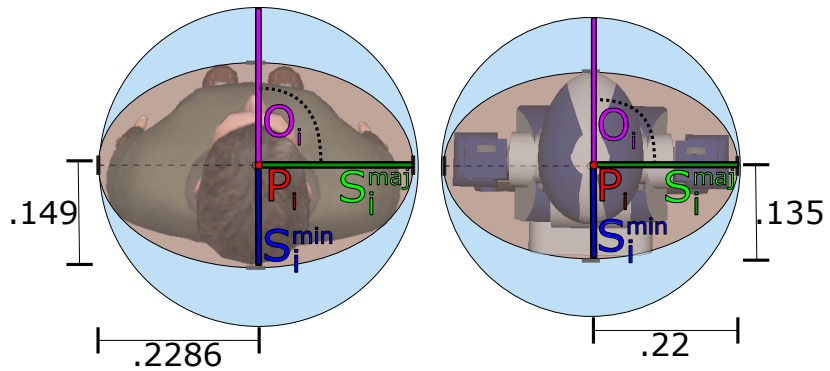


Figure 2.3: **Bounding Disc vs Ellipse.** Disc (Blue) and ellipse (Brown) for (Left) a human of average body width and depth; (Right) HRP-4 robot.

linear approximation (Section 2.4.1) for each ellipse. Furthermore, we extended the original ORCA algorithm (Van den Berg et al., 2011) to elliptical agents by formulating appropriate constraints (Section 2.4.2.1 and Section 2.4.2.2) that preserve the properties of the velocity obstacles. The set of permitted velocities $EORCA^{\tau}_{A|B}$ and $EORCA^{\tau}_{B|A}$ for agents A and B, respectively, are reciprocally maximal. The overall approach is conservative, but it guarantees collision-free navigation. In densely packed conditions, the EORCA set of feasible velocities may be empty, in which case the 2D linear program will not find a solution. One possibility is to change the orientation of the ellipses (see EORCA-F in Section 2.6) to find a solution. If that does not find a feasible solution, we can select the velocity that minimally penetrates the constraints generated by neighboring robots, by using 3D linear programming (Van den Berg et al., 2011).

2.5 Accelerating EORCA With Precomputation

The EORCA algorithm described above can compute collision-free trajectories for elliptical agents. However, in practice it is computationally two orders of magnitude more expensive compared to the original ORCA algorithm for disc-based agents (see Table 2.5). The main bottleneck is the computation of the Minkowski Sum of the polygons and determining the forward arc on its boundary while constructing the half-plane constraints for each neighboring agent and obstacle.

In order to accelerate the computation, we use precomputed Minkowski Sums for different orientations of ellipses and still guarantee collision-free navigation. In particular, we use a precomputed table of Minkowski Sums such that the runtime computation is $O(1)$, corresponding to a table lookup. We use a discrete angular resolution of θ_E for this precomputation. Let $\mathcal{P} = \{\theta_E \times i : 0 \leq i \leq \lfloor \frac{2\pi}{\theta_E} \rfloor | \theta_E \in (0, 2\pi)\}$ denote an ordered set of angles. Also, let $\mathcal{K} = \{ROT(A(\theta_i), \theta_{i+1}))|\theta_i, \theta_{i+1} \in \mathcal{P}\}$ denote the set of surfaces

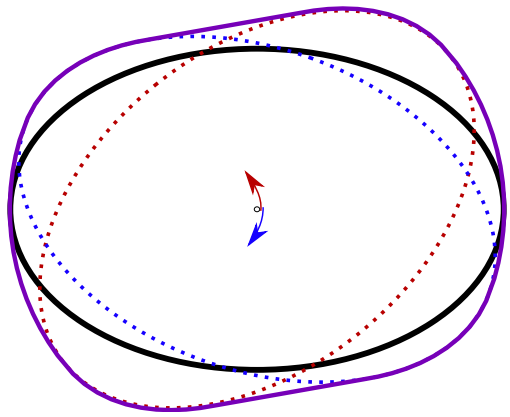


Figure 2.4: **Swept Ellipse:** An ellipse (shown in black) swept along an interval with padding in the positive direction (in red) and negative direction (in blue) equal to the agent's maximum angular velocity. The convex-hull of this swept surface (in purple) is used in place of the agent to construct Minkowski Sums and produce collision-free rotations for the EORCA-F algorithm. A table of these swept surfaces is maintained for acceleration of the EORCA-F algorithm.

generated by rotating an ellipse A from orientation θ_i to θ_{i+1} for each ordered pair (θ, θ_{i+1}) in $\mathcal{P}$. Likewise, let $\mathcal{K}'$ denote the set of swept volumes (Figure 2.4) generated by rotating an ellipse at orientation θ_i by $\theta_{alpha} = \pm\delta t \times \alpha_{max}$, where δt and α_{max} denote the simulation timestep and maximum angular acceleration of the robot, respectively. Each surface in $\mathcal{K}$ and $\mathcal{K}'$ is parametrized by the angle θ_i.

We precompute and store the pairwise Minkowski Sums for surfaces in $\mathcal{K}$, as well as $\mathcal{K}'$. Each such Minkowski Sum is parameterized by the corresponding pair of angles, corresponding to the two ellipses. Additionally, we also store the extreme points of the polygon which facilitates the construction of tangents and reduces the complexity of finding the closest point on the Velocity Obstacle. Consider an elliptical agent A with orientation θ_A. The precomputed table can be used to find the surface $\mathcal{K}_A$ s.t. $\theta_i \leq \theta_A \leq \theta_{i+1}$ where $\theta_i, \theta_{i+1} \in \mathcal{P}$. By definition, the ellipse representing the agent A is contained within $\mathcal{K}_A$ and is thus, conservative. When computing the constraints for two agents A and B, we lookup the surfaces $\mathcal{K}_A$ and $\mathcal{K}_B$ respectively, and the corresponding Minkowski Sum. The collision-free guarantees (Section 2.4.2) still hold but the solution may not be optimal since the EORCA sets of collision-free velocities are not maximal. Overall, our precomputed structures with an interval size of 5 degrees provide a 40x improvement in performance. The precomputation time is on the order of minutes and spatial complexity is $O(m * o^2)$ where m is the number of samples on the ellipse and o is the number of orientation intervals ($120MB$ for $o = 72, m = 100$).

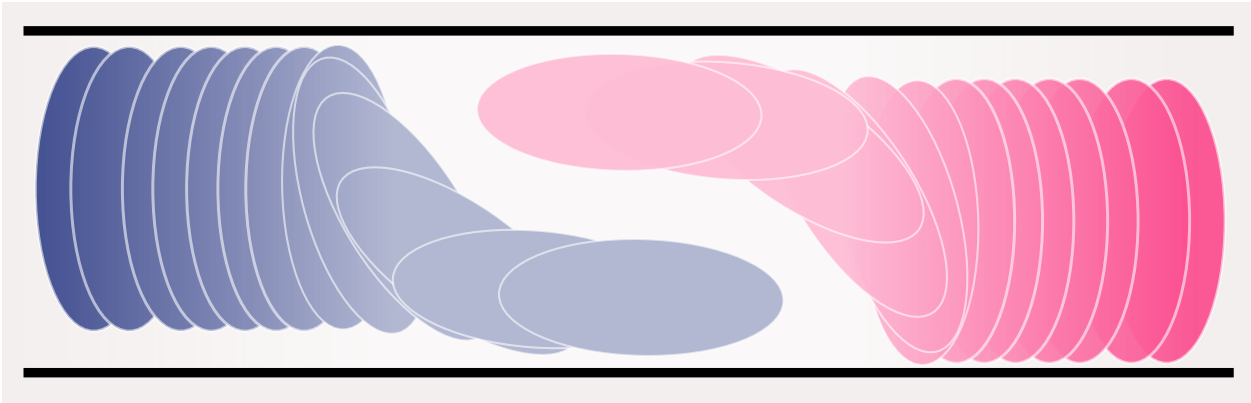


Figure 2.5: **Elliptical Agents in a Narrow Passage:** Two ellipses approach one another in a narrow hallway. In order to pass safely, each ellipse must rotate to reduce their profile. These behaviors are not possible with disc-based agents and are demonstrated with our EORCA-F and EORCA-P algorithm.

2.6 Orientation Computation

The use of ellipses increases the configuration space of each to three dimensions - $[x, y, \theta]$. In order to maintain interactive simulation, we decompose the problem in two parts: update the orientation of each ellipse and compute the optimal collision-free velocity for that orientation. By using swept surfaces in place of rotating ellipses for the construction of neighboring agent constraints, we ensure collision-free rotations as agents navigate. Figure 2.4 illustrates the swept surfaces. We first present two simple approaches to orientation computation, and then detail our heuristic orientation computation approach based on pedestrian observations. These methods are given as function definitions $\mathcal{H}$, for orientation update:

- **EORCA-C:** In the simplest case, the agents maintain their orientation between successive simulation time-steps:

$$o_i^0 = o_i. \tag{2.18}$$

- **EORCA-F:** In some cases, the agent must change its orientation in order to find a collision-free velocity. For example, in Figure 2.5, the agent must change its orientation to navigate through a narrow hallway. In this method, we determine whether agent i can travel along its current velocity $\vec{v}_i$ with its current orientation o_i by estimating the scalar space, c_i, available to the agent w.r.t its current orientation at a point slightly ahead in the direction of travel. Let $\mathcal{MC}(o)$ denote the minimum clearance required for an

agent with orientation o. Then, o_i^0 is given by:

$$o_i^0 = \begin{cases} o_i & \text{if } c_i \geq \mathcal{MC}(o_i), \\ o_i^E & \text{otherwise.} \end{cases}$$

where o_i^E denotes closest orientation to o_i such that $c_i \geq \mathcal{MC}(o_i^E)$. Clearance can be computed as the distance to the nearest point on the nearest neighboring agent or obstacle from the extreme point on either side of the agent with respect to its velocity.

- **EORCA-P:** In the subsequent section, we describe our data-driven orientation update function based on pedestrian experiments.

In the EORCA-F algorithm, at each time step, we evaluate $\mathcal{H}$ and $\mathcal{K}$ (Section 2.5) to determine the preferred orientation $\vec{o}^0$ and preferred velocity $\vec{v}^0$ for each agent. Next, we compute the EORCA constraints (Section 2.4.2) using "swept" surfaces from the set $\mathcal{K}'$ for agents for which $\vec{o} \neq \vec{o}^0$ and the less conservative surfaces from the set $\mathcal{K}$ for the remaining ones. The agents for which the 2D linear program computes a feasible solution can update their orientation. The remaining robots maintain their orientation and we use 3D linear programming to compute the "safest possible" velocity. Figure 2.6 details how we compute clearance.

2.6.1 EORCA-P: Data-driven Orientation Update for Virtual Pedestrians

In general, pedestrians face – or are oriented toward – their direction of motion. This fact is used by current crowd simulation algorithms that typically extrapolate the orientation from an agent's current velocity. However, pedestrians often twist their shoulders and/or take sidesteps to maneuver through dense or tight spaces. They often backpedal momentarily to give way to oncoming traffic. In these cases, their orientation is no longer in line with the direction of their movement. These local behaviors are not modeled by current simulation algorithms because, among other reasons, the commonly used disc representation inherently lacks the capability to model orientation. Some techniques augment the disc with an orientation vector, which is used for animation and rendering (Karamouzas et al., 2009; Johansson et al., 2007), but do not offer sufficient capability to model these local interactions.

We present a novel orientation computation algorithm for elliptical agents that accounts for shoulder-turning, lateral motion, and backpedaling. Due to their inherent radial asymmetry, elliptical agents can change

their orientation to decrease the portion of body width in the walking direction, allowing them to navigate through small gaps and generate more human-like trajectories.

2.6.2 Empirical Observations

Imanishi and Sano (2014) investigated pedestrian avoidance behaviors in crossing flows under laboratory conditions. They conducted experiments in which a pedestrian, the "traverser", crosses a crowd of pedestrians, referred to as "pedestrians in flow"(see Figure 2.8). Each experiment was parametrized based on the density of pedestrians in a flow and the angle at which the *traverser* approaches the crowd flow. They observed that both traversers and pedestrians in flow avoided collisions by adjusting their walking speed, walking route (detouring), and shoulder-turning. Moreover, Imanishi and Sano (2014) quantitatively classified each of these behaviors into *four levels of avoidance*: no, potential, weak, and strong avoidance. They observed that pedestrians did not use shoulder-turning when there was enough margin space around them. For example, with a density of two people/m^2 and crossing angle of 180, pedestrians slightly reduced their speeds and avoided each other primarily by controlling their shoulder angles. They observed shoulder-turns as high as 80 degrees. However, in situations with a crossing angle of 45 degrees with the same density, pedestrians mostly relied on speed reduction and only infrequently performed shoulder-turns. Their results also demonstrate that higher pedestrian densities result in strong avoidance. Hughes et al. (2014) also conducted experiments to observe the conditions under which humans exhibit lateral motion and concluded that lateral motion occurs when a collision is imminent and is mostly in tight spaces in which the pedestrian may not be able to navigate with a non-lateral orientation. Our approach is driven by these empirical observations.

2.6.3 Shoulder-Turning

Given its new velocity, each agent evaluates the scalar space, termed as "clearance", available to it at a sequence of points along the line joining its current position and its estimated position in the next second ($\vec{p} + \vec{v}$). Essentially, the line denotes the positions of the center of the ellipse during the interval :$[t, t+1]$. In general, the agent is expected to continue along with its current orientation, effectively translating along this line. Prior studies investigating lateral motion in pedestrian walking have shown a tendency to decide on lateral motion during the last gait cycle before arriving at an obstruction, approximately one second before (Gérin-Lajoie et al., 2005). We therefore choose an offset of one second in the future to provide the agent sufficient time to accomplish a shoulder-turn if it is required. Figure 2.6 describes the clearance computation

at a given point with respect to the agent. The estimated clearance, c, available to the agent with respect to its current orientation is set to be the minimum clearance at the sampled points. In addition, the agent only considers nearby agents and obstacles that are ahead with respect to its new velocity and are also visible to the agent at its current position. Once clearance is computed, the algorithm chooses a new orientation o^{new} as given by the optimization:

$$\arg\min_{o_{new}} \|o_{new} - o\| \quad \text{such that} \quad w(o_{new}) \leq c, \tag{2.19}$$

where w defines the cross-section width of the new orientation perpendicular to its new velocity. This is encapsulated in the function *getEllipseForClearance* described in Algorithm 1. For a given clearance, there are at most two ellipses that satisfy Equation (2.19). As part of our implementation, the agents bias counterclockwise rotations over clockwise rotations.

Computing Clearance: Let $\vec{CP}$ denote the point at which we wish to compute the clearance. We define $\hat{n}_i$ to be a unit vector perpendicular to the new velocity $\vec{v}_i^{new}$. Let c_i^+ and c_i^- denote the clearance available for agent i at point $\vec{CP}$ in the direction $\hat{n}_i$ and $-\hat{n}_i$ respectively. For each nearby agent j, we compute the nearest point $\vec{p}_{ij}$ on j with respect to $\vec{CP}$. We then update c_i^+ and c_i^- according to:

$$a = (\vec{p}_{ij} - \vec{CP}) \cdot \hat{n}_i \tag{2.20}$$

$$c = \|a\| \tag{2.21}$$

if $a \geq 0$

$$c_i^+ = \min(c, c_i^+) \tag{2.22}$$

else

$$c_i^- = \min(c, c_i^-) \tag{2.23}$$

end

where c_i^+ and c_i^- are initialized to infinity. Clearance c_i at point $\vec{CP}$ can then be estimated as $c_i = c_i^+ + c_i^-$ (Figure 2.6). Without loss of generality, the same equations can be applied to compute clearance with respect to a nearby obstacle j, where $\vec{p}_{ij}$ represents the closest point on the obstacle to $\vec{CP}$. Doing so for each nearby agent and obstacle, we can estimate the scalar space available to an agent at the query point $\vec{CP}$.

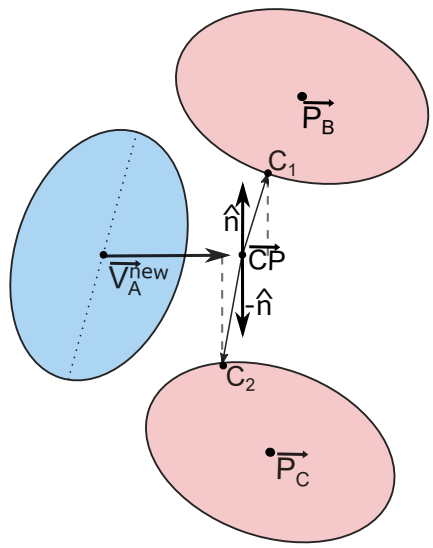


Figure 2.6: **Computing Clearance.** We compute the clearance at a point $\vec{CP}$ that is nearby (e.g., 1 meter ahead) in the direction of the new velocity $\vec{v}_A^{new}$. $\vec{c}_1$ denotes a vector from $\vec{CP}$ to the closest point on the nearest agent such that $\vec{c}_1 \cdot \hat{n} \geq 0$ where $\hat{n}$ is a unit vector perpendicular to $\vec{v}_A^{new}$. Similarly, $\vec{c}_2$ denotes a vector from $\vec{CP}$ to the closest point on the nearest agent such that $\vec{c}_2 \cdot (\hat{n}) < 0$. Clearance at $\vec{CP}$ is set to $\vec{c}_1 \cdot \hat{n} + \vec{c}_2 \cdot -\hat{n}$, denoted by the dotted lines.

To determine whether the new orientation is feasible, each agent projects their own position and the positions of their neighbors one step forward in simulation, assuming agent i adapts its new velocity $\vec{v}_i^{new}$ and new orientation $\vec{o}_i^{new}$ while other agents continue with their current velocity and orientation. We then explicitly check for collisions, denoted by the function *CollisionCheck* inAlgorithm 1. If there is a collision, the orientation update is discarded and the agent maintains its prior orientation. It should be noted that the clearance is computed for a set of sampled times in the next second while discrete collision checks are only performed for the next time step. These collision checks are necessary to ensure that the preferred orientation with respect to the computed clearance can be safely achieved. As such, it is sufficient and efficient to perform these checks only for the next time step. The detailed algorithm is described in Algorithm 1.

2.6.4 Lateral Motion and Backpedaling

Agent i estimates the time to collision with respect to nearby agents and obstacles based on its new velocity v_i^{new}. As described in Section 2.4.2.2, obstacles are represented as a collection of line segments in the 2D plane and thus we can analytically determine the closest point on the line segments (function *ClosestPtOnObstacle* in Algorithm 2) in order to compute the time to collision. The computed time to collision is not entirely accurate as it computes the time in the future at which the center of two agents (or an

Algorithm 1 EvalShoulderTurn: Compute the preferred orientation for shoulder-turning (if needed) for agent i.

Input: Position $\vec{p}_i$, new velocity $\vec{v}_i^{new}$, orientation $\vec{o}_i$, number of clearance samples to evaluate m_{clear}, neighboring agents $agts$, neighboring obstacles obs, and simulation timestep Δt

Output: Boolean indicator for shoulder-turn, $result$, and the preferred orientation $\vec{o}^{pref}$

```
result = false
c = ∞
for i in 1 to m_clear do
    CP_i ⇐ p_i + v_i^new · i/m_clear
    c_i ⇐ getClearanceAtPoint(CP_i, obs, agts)
    c ⇐ min(c_i, c)
end for
o_i^pref ⇐ getEllipseForClearance(o_i, c)
if (o_i^pref ≠ o_i) then
    collision ⇐ CollisionCheck(p_i, v_i^new, o_i^pref, agts, obs, Δt)
    if (collision ≠ true) then
        result = true
    end if
end if
```

agent and the closest point on a obstacle) collide and in doing so, does not account for the shape of the agent. However, in practice, we find this to be a reasonable estimate of the actual time to collision.

Agent i uses the new velocity to compute the angle of deviation from the preferred direction of travel. If the deviation from the preferred direction is greater than threshold θ_{thresh} and there is an imminent collision – i.e., the time to collision is less than threshold ttc_{thresh} – the resulting algorithm performs lateral movement or backpedaling i.e. the agent adapts its new velocity without changing its orientation. We base our algorithm, described in Algorithm 2, on the observations of Hughes et al. (2014) and illustrate it in Figure 2.7.

2.6.5 Overall Algorithm

Algorithm 3 presents the overall simulation update mechanism for each agent. First, we compute the half-plane constraints for nearby agents and obstacles, as described in Section 2.4.2. These steps are denoted by the functions *ComputeAgentConstraints* and *ComputeObsConstraints* in Algorithm 3. The constraints for neighboring agents are constructed with respect to the neighbor's orientation and velocity at the beginning of the simulation step. In doing so, the constraints guarantee collision-avoidance for at least the planning horizon time τ, under the assumption that neighboring agents maintain the velocities computed in the previous time-step. Next, we compute the set of permissible velocities (Equation (2.16)) and use linear programming to compute the new velocity for the next time step (or the relaxation described above). This is encapsulated

Algorithm 2 EvalLateralMotion(): Determine if agent i should execute lateral motion and backpedal.

Input: Position $\vec{p}_i$, new velocity $\vec{v}_i^{new}$, preferred velocity $\vec{v}_i^0$, threshold for deviation from preferred direction θ_{thresh} threshold for time to collision ttc_{thresh}, the set of neighboring agents $agts$, and the set of neighboring obstacles, obs.

Output: Boolean indicator for side-stepping $result$.

```
ttc ⇐ ∞
result = false
for each neighboring agent j in agts do
    k̂ ⇐ (p⃗_j − p⃗_i) / ‖p⃗_j − p⃗_i‖
    v_ij^tang ⇐ (v⃗_j − v⃗_i^new) · k̂
    ttc_j ⇐ ‖p⃗_j − p⃗_i‖ / v_ij^tang
    if ttc_j ≤ ttc then
        ttc ⇐ ttc_j
    end if
end for
for each neighboring obstacle ob in obs do
    p⃗_j ⇐ ClosestPtOnObstacle(p⃗_i, ob)
    k̂ ⇐ (p⃗_j − p⃗_i) / ‖p⃗_j − p⃗_i‖
    v_ij^tang ⇐ (−v⃗_i^new) · k̂
    ttc_j ⇐ ‖p⃗_j − p⃗_i‖ / v_ij^tang
    if ttc_j ≤ ttc then
        ttc ⇐ ttc_j
    end if
end for
θ_dev_i ⇐ acos(v⃗_i^new · v⃗_i^0 / (‖v⃗_i^new‖ · ‖v⃗_i^0‖))
if abs(θ_dev_i) ≥ θ_thresh and ttc ≤ ttc_thresh then
    result = true
end if
```

by the function *SolveLinearProgram*. After computing the new velocity, $\vec{v}^{new}$, the algorithm determines whether the agent must side-step (Section 2.6.4), turn its shoulders (Section 2.6.3), or align its orientation with the new velocity. In any case, we use a threshold for the maximum angular acceleration α_{thresh} to bound the change in orientation, denoted by the function *ClampAngularAcc*.

It is important to note that the current velocity and orientation of each agent is not updated until all agents have been evaluated. This enables us to independently compute the new velocity and the new orientation of each agent. Updates are performed as described by the following equations. In the case of EORCA-C and EORCA-F, Algorithm 3 holds with the substitution of the $evalLateralMotion$ and $EvalShoulderTurn$.

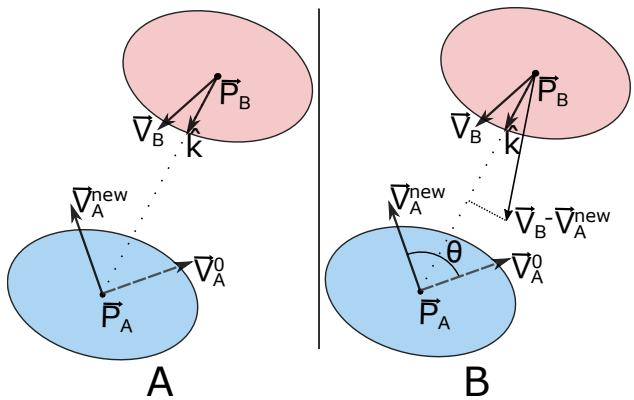


Figure 2.7: **Lateral motion computation** (A) At each timestep, agent A at position $\vec{p}_A$ computes a new velocity, $\vec{v}_A^{new}$, which is collision-free with respect to all neighboring agents and obstacles. For each neighboring agent or obstacle B, additional computations are performed to determine if lateral motion is required for collision avoidance. (B) Agent A computes the relative velocity with respect to each neighboring agent B's velocity, as well as the relative approach speed v_{AB}^{tang} by projecting onto the displacement direction, $\hat{k}$. If the deviation between v_A^{new} and the desired velocity, v_A^0, denoted by θ, is greater than θ_{thresh} and the minimum time to collision with a neighboring agent or obstacle is below ttc_{thresh}, the agent will engage in lateral steps to avoid the collisions.

$$\vec{v}_i \Leftarrow \vec{v}_i^{new}, \tag{2.24}$$

$$\vec{o}_i \Leftarrow \vec{o}_i^{new}, \tag{2.25}$$

$$\vec{p}_i \Leftarrow \vec{p}_i + \Delta t \cdot \vec{v}_i. \tag{2.26}$$

2.7 Validation with Real-World Human Trajectories

In this section, we validate the EORCA-P algorithm by reproducing the live pedestrian experiments reported by Imanishi and Sano (2014) and Zhang et al. (2012) that highlight pedestrian interaction in crossing flows and bidirectional flows respectively.

2.7.1 Crossing Flows

Recently, Imanishi and Sano (2014) sought to study pedestrian avoidance behaviors in crossing flows. They devised an experiment in which a group of 16 pedestrians with density $\rho \in \{0.25, 1, 2\}$ people/m^2 move through an open room. A single person crosses the group at an angle $\theta \in \{45, 90, 135, 180\}$ deg.

Algorithm 3 Simulation Update for agent i

Input: State Vector [$\vec{p}_i$, $\vec{v}_i$, $\vec{v}_i^0$, $\vec{o}_i$, s_i^{maj}, s_i^{min}] for agent i, simulation time step Δt, threshold for deviation from preferred direction θ_{thresh}, threshold for time to collision ttc_{thresh}, threshold for angular acceleration α_{thresh}, amount of time the agent has been engaged in lateral motion $lateralSeq$, number of clearance samples to evaluate m_{clear}, and threshold for lateral motion time $lateral_{thresh}$.

Output:

```
obs ⇐ GetNeighboringObstacles(p⃗_i)
agts ⇐ GetNeighboringAgents(p⃗_i)
agtLines_i ⇐ ComputeAgentConstraints(agts)
obsLines_i ⇐ ComputeObsConstraints(obs)
v⃗_i^new ⇐ SolveLinearProgram(agtLines_i, obsLines_i)
o⃗_i^new ⇐ v⃗_i^new / ||v⃗_i^new||
lateral ⇐ EvalLateralMotion(p⃗_i, v⃗_i^new, v⃗_i^0, θ_thresh, ttc_thresh, agts, obs)
if (lateralSeq ≤ lateral_thresh && lateral == true) then
    lateralSeq = lateralSeq + Δt
    o⃗_i^new ⇐ o⃗_i
else
    lateralSeq = 0
    (turn, o⃗^pref) ⇐ EvalShoulderTurn(p⃗_i, v⃗_i^new, o⃗_i, m_clear, agts, obs, Δt)
    if (turn == true) then
        o⃗_i^new ⇐ o⃗^pref
    end if
end if
if (o⃗_i^new ≠ o⃗_i) then
    o⃗_i^new ⇐ ClampAngularAcc(o⃗_i, o⃗_i^new, α_thresh, Δt)
end if
```

Three trials were conducted for each combination of density and crossing angle. The authors measured the level of occurrence of each of three local collision-avoidance behaviors: speed reduction, detouring, and shoulder-turning. For each combination of density and crossing angle, the authors further classify the degree of collision avoidance via shoulder-turns into one of four categories: (S)Strong avoidance ($36 - 90$ deg), (W)Weak avoidance ($24 - 36$ deg), (P)Potential avoidance ($12 - 24$ deg) and (N)No avoidance (≤ 12 deg). Figure 2.8 demonstrates the experimental setup utilized by Imanishi and Sano (2014).

To validate our algorithm, we faithfully reproduced their experiments. Pedestrians were modeled as elliptical agents with average human shoulder-width and body-depth and a preferred speed of 1.3 m/sec. For each combination of density, ρ and crossing angle θ, we recorded the maximum shoulder turn specifically made by the traverser. We classify the shoulder-turns in accordance with the classification presented by the authors. We report these observations in Table 2.1 along with the observations made by the authors. For our model, we also report the maximum degree of shoulder turn made by the traverser.

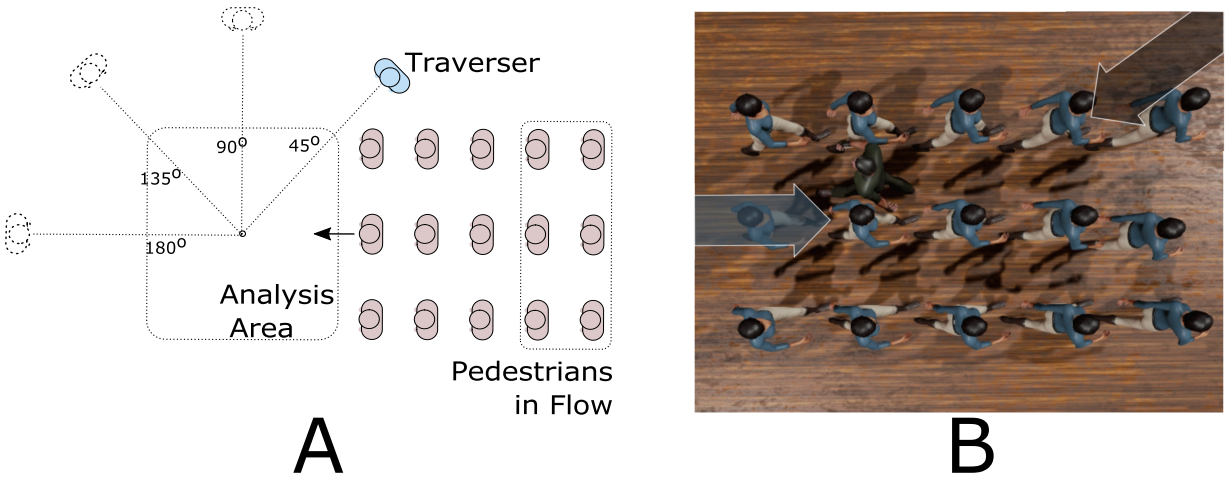


Figure 2.8: **Pedestrian interactions in crossing flows.** (A) Experimental Setup by Imanishi and Sano (2014). A pedestrian, the "traverser", crosses "pedestrians in flow" with density $\rho \in \{0.25, 1, 2\}$ people/m^2 at an angle $\theta \in \{45, 90, 135, 180\}$ deg. (B) Reproduction of the experiment with elliptical agents for density $1 \ people/m^2$ and crossing angle $45\,\mathrm{deg}$.

Our results of our simulation are summarized as follows:

- Consistent with real world observations, shoulder-turning was negligible in cases where the traverser had enough margin of space to travel with its current orientation ($\rho \in \{0.25, 1\}$ people/m^2, $\theta \in \{45, 135\}\,\mathrm{deg}$).
- The degree of shoulder-turning increased as the available space decreased in higher densities ($\rho \in \{1, 2\}$ people/m^2, $\theta = 90\,\mathrm{deg}$) as was observed by Imanishi and Sano (2014).
- Our model was able to match the observed level of shoulder turn in every case except when the traverser attempts to cross the crowd of density 2 people/m^2. In this case, the velocity constraints caused the traverser to detour around the crowd flow thus, allowing him the clearance to continue without shoulder-turning. However, in the study, the traverser was instructed to "walk straight towards the destination", thus walking through the crowd which required him/her to shoulder turn.

2.7.2 Bidirectional Flow

We also validated our model by reproducing the experiments of Zhang et al. (2012). This experiment tracked two groups of pedestrians moving in opposing directions through a hallway. The researchers varied the initial density of the pedestrians and observed the relationship between speed and density in the hallway.

Table 2.1: **Validation results for crossing flow.** We validate the EORCA-P algorithm with the study conducted by Imanishi and Sano (2014) (shown as IS14). For each combination of density and crossing angle, the authors classify the degree of collision avoidance via shoulder-turns into one of four categories: S)Strong avoidance ($36-90\,\mathrm{deg}$), (W)Weak avoidance ($24-36\,\mathrm{deg}$), (P)Potential avoidance ($12-24\,\mathrm{deg}$) and (N)No avoidance ($\leq 12\,\mathrm{deg}$). The table presents the classification of the observed degree of collision avoidance shown by the pedestrian crossing the flow (IS14) as well as the one simulated by our algorithm (shown as Ours). We also provide the maximum angle (max) of shoulder turn observed in our simulation. Most prior disk-based methods do not model orientation specific behaviors and thus, are incapable of simulating local interactions such as shoulder-turning. In contrast, our algorithm generates orientation specific behaviors for elliptical agents and generates results that are consistent with human data.

	Crossing Angle							
Density	180		135		90		45	
(People / m^2)	IS14	Ours(max)	IS14	Ours(max)	IS14	Ours(max)	IS14	Ours(max)
0.25	-	W[3.4]	-	W[2.1]	-	W[3.75]	-	W[2.91]
1	-	W[1.63]	W	W[25.1]	S	S[90]	W	W[30.01]
2	S	N[0.12]	S	S[45.5]	S	S[83]	W	W[29.7]

Behaviors such as shoulder-turning and sidestepping can be frequently seen in cases of high densities ($2-2.5$ people/m^2). We reproduced the experiment and ran our simulation using elliptical agents with initial density $\rho \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$ people/m^2. We observed emergent behaviors such as lane formation for densities as high as 2 people/m^2. This implied that most agents could smoothly navigate through the hallway without the need for shoulder-turns and sidesteps. However, at a higher density of 2.5 people/m^2, shoulder-turning and sidestepping were very frequent, in accordance with the captured data. We generated a visual rendering of the high-density experiment ($\rho = 2$ people/m^2) and highlighted the occurrence of shoulder-turning and side-stepping behaviors using our algorithm (Figure 2.9).

2.7.2.1 Prior Methods

We have compared the performance of our algorithm using elliptical agents to prior disc-based methods, such as:

- **Helbing:** This is a social forces based method which treats the crowd as a collection of mass particles, and applies Newtonian-like physics to drive the agents towards their goals (Helbing et al., 2000). Agents under such a model experience repulsive forces from neighboring agents and obstacles, and an attractive force towards the goal. The formulation of forces is distance specific i.e. these forces account for the

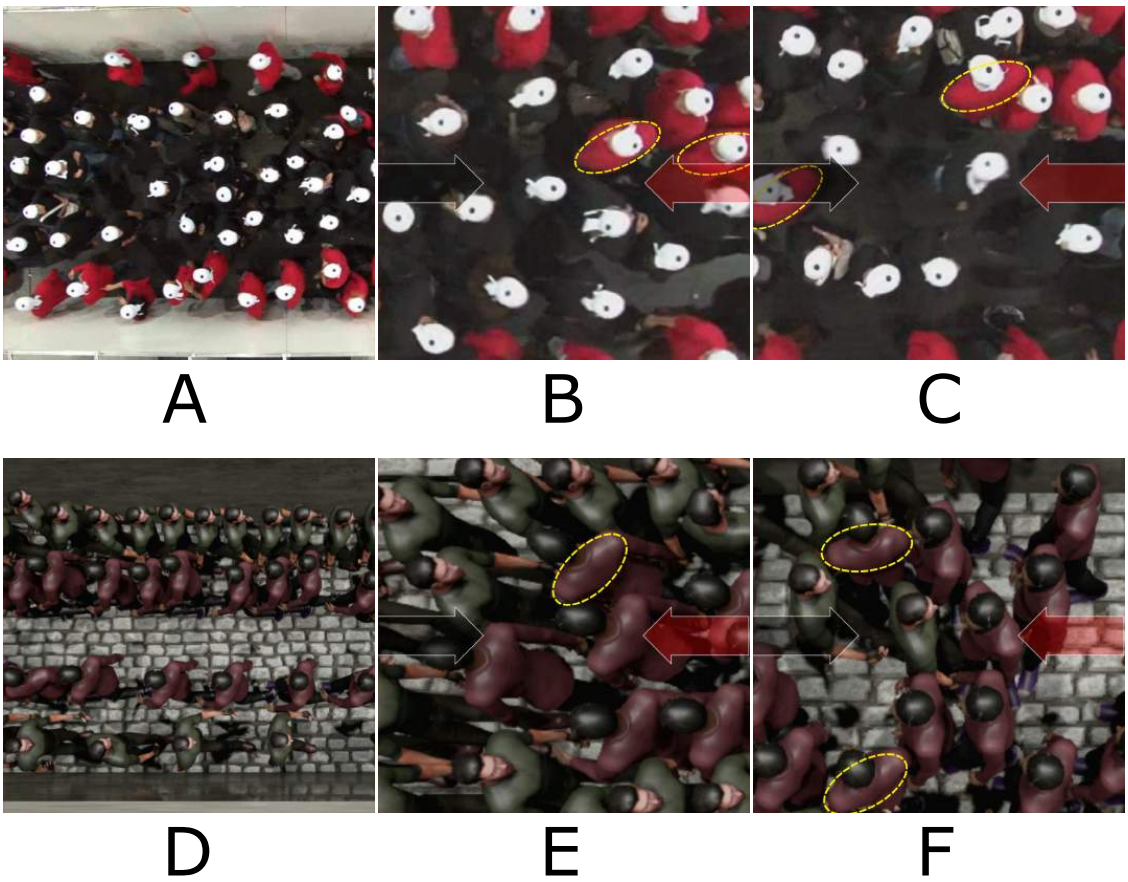


Figure 2.9: **Validation results for bidirectional flow.** (Top) Frames from captured crowd footage (Zhang et al., 2012). Shoulder-turning is a commonly observed phenomenon in high-density crowds as shown in these still frames. In each frame, several twisting agents have been highlighted in yellow with ellipses. (Bottom) We use 2D elliptical agents to generate such local behaviors automatically. The 2D trajectories are visualized using 3D characters during post processing. The agents shoulder turn and sidestep (E and F) to make their way through regions of high densities.

position of neighboring agents and obstacles. At each time step, the superposition of various forces are computed for each agent, ultimately determining a viable velocity.

- **Power Law:** Karamouzas et al. (2014) recently proposed a a social forces based approach which takes into account the time to collision in its formulation of repulsive forces. The proposed model is validated using crowd data sets and is shown to reproduce many known crowd phenomena.

- **ORCA:** The ORCA (Van den Berg et al., 2011) model uses the concept of velocity obstacles and geometric optimization to find collision-free velocity for disc-based agents.

For each of these methods, we model the discs with a radius equal to the semi-major axis for our elliptical agents. We ensure that parameters such as preferred speed, maximum acceleration are consistent in all

Table 2.2: **Simulation time.** We define simulation time as the time duration between initialization and the moment at which all agents have reached their goals. We compare the simulation time with elliptical agents, EORCA-P, to that with prior disc-based approaches such as ORCA, Helbing and Power Law on three benchmark scenarios. In general, EORCA agents reach their goals significantly faster by side-stepping and shoulder-turning around each other. In the case of the bidirectional flow benchmark, all three disc-based methods generated congestion or deadlocks and agents were unable to reach their goals in the allotted maximum time of 400 seconds. In case of EORCA-P, we also indicate the fraction of total simulation time, averaged per agent, spent executing orientation related behaviors such as side-stepping, and shoulder-turning.

	Solution Time (s)				Orientable Behaviors (% of Sol. Time)
Scene	EORCA-P	ORCA	Helbing	Power Law	EORCA-P
Anti-podal Circle	20.95	28.90	25.90	19.18	25.47
Cross Flow	58.30	114.30	77.11	76.91	18.31
Bidirectional Flow	85.55	∞	∞	∞	20.88

simulations. In addition, we have taken the exact formulation of these models, including parameter values, from the original work of the respective authors.

Some disc-based approaches use cellular automata (Schadschneider, 2002) to generate human-like crowd behaviors. These method divide the workspace into discrete grid cells which can be occupied by at most one agent. Agents then follow simple rules to move towards their goals through adjacent grid cells. Some of of these methods also use a secondary grid of cells, often referred to as floor fields (Burstedde et al., 2001), which can be used to alter the transition rules. These approaches have been used successfully for simulating disc-based agents. However, it is unclear how the discretized grid can be effectively used to simulate elliptical agents.

2.7.2.2 Comparisons with Prior Methods

Prior disc-based agents simulated with ORCA, Helbing, and Power Law are unable to find a collision-free path and lead to deadlocks. This is evident from the simulation time noted in Table 2.2. Elliptical agents, simulated with EORCA-P, are able to find collision-free paths to their goals. In addition, they also show behaviors seen in the recorded human footage such as lane-formation, shoulder-turning and side-stepping (Figure 2.9). We also visualize the observed densities in the human data, and compare it with simulations to further validate our approach. Figure 2.10(A) depicts the density at the instant when 20% of the population has crossed the midpoint of the scene with respect to initial positions. The Power Law method leads to

Table 2.3: **Maximum density values during bidirectional flow.** Our approach with elliptical agents, EORCA-P, results in densities that are similar to those observed in captured human data as compared to prior disc-based methods such as ORCA (Van den Berg et al., 2011), and Power Law (Karamouzas et al., 2014). The ORCA method initially shows density similar to that of the human data, but results in congestion and deadlocks as the simulation progresses. The Power Law model produces deadlocks and significant collisions causing densities to reach unrealistic values. Our approach results in more consistent density distribution, as depicted in Figure 2.10.

	Maximum Density (people/m^2)			
	Fraction of Simulation Completed			
Method	20%	40%	60%	80%
Human	0.876	1.15	1.322	1.350
EORCA-P (ours)	1.567	2.005	2.138	2.559
ORCA	1.505	2.445	3.355	-
Powerlaw	13.154	12.859	12.242	12.112

Table 2.4: **Simulation Parameters.** The parameters used in our experiments. In order: semi-major axis length(m), semi-minor axis-length(m), number of ellipse samples, time horizon for agent-agent collision avoidance(sec), time horizon for agent-obstacle collision avoidance(sec), maximum angular acceleration($\deg$/sec), threshold for deviation from preferred direction($\deg$), threshold for time to collision(sec), threshold for lateral motion time(sec), and number of clearance samples to evaluate.

s^{maj}	s^{min}	m	τ	$\tau_{obstacle}$	α_{thresh}	θ_{thresh}	ttc_{thresh}	$lateral_{thresh}$	m_{clear}
0.2286	0.149	100	1.0	1.0	360	10	6	3	10

extreme congestion, producing extremely high densities of approx. 4 people/m^2. ORCA performs better, but also shows congestion at the midpoint of the corridor. Our method, EORCA-P, generates the closest results to the captured human data, producing smooth densities across the corridor. Similarly, Figure 2.10(B) depicts the density at the instant when 60% of the population has crossed the midpoint of the scene. Both ORCA and Power Law lead to substantial congestion on either side of the corridor midpoint, leading to empty space and a lack of flow. On the other hand, EORCA-P maintains flow despite congestion near the midpoint and better matches the human data. Table 2.3 details specific measured densities at various simulation milestones. The model proposed by Helbing et al. (2000) was found to be unstable even at a small timestep of 0.01 second, and is therefore, not included in the comparisons.

2.8 Performance on Synthetic Benchmarks

In this section, we highlight the performance of our EORCA-C, EORCA-F, and EORCA-P on different benchmarks and also provide illustrative demonstrations based on real-world scenarios in which lateral motion and high-density situations are common.

We set the sampling size at $m = 100$ points for our piece-wise linear approximation and orientation intervals of 5 degrees for precomputed surfaces. Using Theorem 2 (Section 2.4.1), the maximum approximation error for a point on the bounding ellipse for a human was found to be 0.005. The aggregate error, defined as the difference between the area of the exact ellipse and approximated ellipse, was $0.0002m^2$. For the HRP-4 humanoid robot benchmark, we set the following elliptical parameters: ($s^{maj} = .22$, $s_{min} = .135$, $r = .22$). Figure 2.3 details the difference between the modeled pedestrians and HRP-4 robot.

2.8.1 Benchmarks

We use the following benchmarks to evaluate our algorithm:

- **Hallway Squeeze:** In this scenario, an agent must navigate through a narrow hallway of width .3 meters in order to reach its goal position, which requires the agent to rotate and change its orientation. Elliptical agents simulated with EORCA-F and EORCA-P smoothly change their orientation and find a collision-free path, whereas disc-based agents are unable to find a path to their goals.

- **Hallway Head-on:** In this case, two agents approach each other in a narrow hallway of width .7 meters with goals at opposite ends of the hallway. Agents must rotate to pass through (Figure 2.11(A)). Elliptical agents independently decide to rotate and find a collision-free path to their respective goals, where disc-based agents are unable to find a path.

- **Anti-podal Circle:** This commonly used crowd simulation benchmark depicts agents initialized on the circumference of a circle with anti-podal goal positions. Thus, it leads to congestion and high probability of deadlocks and collisions at the center of the circle. Elliptical agents, simulated with EORCA, shoulder-turn, and side-step to maneuver around neighboring agents. On average, EORCA-P agents show such behaviors for 25.67% of the total simulation time (Table 2.2). Thus, EORCA agents reach their goals significantly faster as compared to disc-based agents simulated with ORCA and Helbing.

However, agents simulated with EORCA are slightly slower to reach their goals compared to Power law agents. Table 2.5 details the results of our simple orientation update methods on this benchmark.

- **Cross Flow:** In this benchmark, two populations, each with 51 agents, cross each other orthogonally. On average, elliptical agents, simulated with EORCA-P, shoulder-turn, and side-step 18.31% of the total simulation time. This allows them to reach their goals in a significantly shorter time period, compared to disc based agents simulated with ORCA, Helbing, and Power Law (Table 2.2). This scenario demonstrates EORCA's capability to utilize space more effectively, and thus, decreases the overall simulation time that corresponds to all the agents reaching their goal position (Figure 2.11(C)). Table 2.5 details the results of our simple orientation update methods on this benchmark.

- **HRP-4 Crossflow**: In this benchmark, the agents are configured to the dimensions of the HRP-4 humanoid robot platform ($s^{maj} = 0.22m$, $s^{min} = 0.135m$). We demonstrate two groups of agents walking in opposing directions through a hallway (Figure 2.11(D)). EORCA-F agents are able to change their orientation and navigate through the high density region at the crossing region. Table 2.5 details the results of our simple orientation update methods on this benchmark.

It is evident from our results (Table 2.5) that both EORCA-C and EORCA-F are computationally comparable to ORCA. Table 2.6 demonstrates a higher cost to our heuristic function. EORCA-F and EORCA-P significantly reduce the solution time as agents utilize the space more effectively by changing their orientation (Table 2.2 and Table 2.5). The average collision, measured as the average of the interval penetration depth at each time step, is negligible in each case.

2.8.2 EORCA Performance: Benefit of Pre-computation

We have evaluated the performance of the EORCA algorithm described in Section 2.3 and the acceleration structure that pre-computes the Minkowski Sums in Section 2.5. In particular, we compared the performance of EORCA, with and without precomputation, to compute collision-free trajectories of a large number of elliptical agents and also compared them with ORCA, which uses disc-based agents, on the anti-podal benchmark. In this scenario, the agents are initialized on a circle and their goal position is set to the antipodal position. We plot the average frame update time as function of the number of agents in Figure 2.12.

Scene	Num. Agents	Model	Sim Time (s)	Avg. Update Time (ms)	Avg. Collisions (m)
Circle	100	ORCA	38.19	0.1972	8.9e-05
		EORCA-C	33.54	1.056	8.1e-04
		EORCA-F	27.5	5.38	7.4e-04
Cross Flow	100	ORCA	54.52	0.2918	3.05e-07
		EORCA-C	55.56	1.1012	7.52e-05
		EORCA-F	53.13	3.95608	9.82e-05
HRP-4 Crossflow	88	ORCA	29.7	0.233	0
		EORCA-C	29.58	0.899	1.55e-06
		EORCA-F	29.64	3.954	0

Table 2.5: **Comparison Results with Simple Orientation Update**. EORCA reduces total solution time with a small increase in computation time (aprox. 5x).

2.8.3 Applications

We devised a set of benchmark scenarios to highlight how elliptical pedestrians can be used to generate real-world crowd behaviors. These demonstrations clearly illustrate frequently seen avoidance behaviors such as shoulder-turns, lateral motion and back pedaling.

Aircraft Unloading: Our first benchmark shows the agents exiting a commercial aircraft (Figure 2.13 (A)(B)(C)). The agents stand up, side-step to the aisle, fetch their luggage, and exit the aircraft. The seat clearance is not large enough for agents to turn and walk to the aisle, thus forcing the elliptical agents to side-step to the aisle. Furthermore, the aisles are not wide enough for two disc-based agents to pass each other. However, two elliptical agents can pass each other by turning their shoulders and side-stepping.

Subway Station: In this scenario, depicted in Figure 2.13(D,E.F), a group of agents rush the platform and wait for the train. When the train arrives, these agents attempt to board the train while those on the train attempt to exit. This creates a bottleneck and is an ideal scenario for the target behaviours. Using our models, agents backpedal and shoulder-turn to allow those agents disembarking to pass before entering the subway car. The elliptical agents are able to make room for the exiting passengers with limited motion and without unnecessary rotations, resulting in natural looking motion.

2.8.4 Timing Analysis

As the cost of the orientation heuristic is higher, we include the average frame computation time for crowd simulation with EORCA-P separately and compare that with prior disc-based techniques. Table 2.6 shows the

Table 2.6: **Average computation time comparison.** We present the average computation time per simulation step for our simulation algorithm with elliptical agents, EORCA-P, and compare it with prior disc-based methods. In general, we find EORCA-P to be 1 order of magnitude more expensive than the compared disc-based methods .

		Computation Time Per Frame (ms)			
Scene	Num. agents	EORCA-P	ORCA	Helbing	Power Law
Anti-podal Circle	50	2.35	0.14	0.19	0.14
Cross flow	102	5.64	0.16	0.34	0.18
Bidirectional Flow	200	11.26	0.34	0.63	0.27

running time of each scenario for EORCA-P agents vs. disc-based agents using ORCA (Van den Berg et al., 2011), Helbing (Helbing et al., 2000), and Power Law (Karamouzas et al., 2014). As mentioned earlier, exact geometric computations using ellipses can be very expensive operations for interactive simulation of large number of agents. The combination of approximations, setting up EORCA constraints, linking orientation to the velocity, and the precomputed data structures improves the overall performance by one order of magnitude. EORCA-P is more computationally expensive than EORCA-F or EORCA-C, but generates results which more closely match pedestrian data.

2.9 Limitations, Conclusions, and Future Work

We have developed an efficient collision-free navigation algorithm for elliptical agents. We use a piecewise linear approximation of ellipses and reduce the velocity computation problem to linear programming. Furthermore, we use a precomputed Minkowski Sum table to reduce the runtime overhead and present techniques to update the orientation. We compute agent orientation based on real-world observations or by efficient update methods. In particular, we present algorithms that link the orientation of each agent to its velocity and compute appropriate side-stepping and turning motions. We have validated the results by comparing the local interactions with real-world behaviors. Overall, we have presented a practical algorithm for multi-agent crowd simulation with many elliptical agents.

Our approach has some *limitations*. The collision-avoidance and navigation algorithms that are based on elliptical agents are more expensive (specifically, one order of magnitude) and more complex than those using discs. As a result, their utility may be limited to dense situations in which such local interactions are important. The collision-avoidance algorithm tends to be conservative and makes some simplifying assumptions in cases

where a unique solution for the optimum velocity or orientation does not exist. Moreover, the orientation computation is only able to model some, and not all, of the local interactions related to turning, lateral motion and backpedaling. The overall approach is also limited to crowd simulation algorithms that reduce trajectory and behavior computation to the preferred velocity specification at each frame. In addition, the orientation update is decoupled from the velocity computation, and we would like to simultaneous optimize both the velocity and the orientation.

There are many avenues for future work. In addition to overcoming these limitations, we would like to validate our algorithm on other real world pedestrian data-sets and evaluate its performance. We would like to further investigate the concept of optimal velocity in cases where a globally unique solution does not exist. It would be useful to develop techniques for flocking or well-known behaviors for elliptical agents (similar to Reynolds (1987)) and also combine these with high-level behavior modeling and footstep planning (Singh et al., 2011). The present computation of shoulder-turning is based on clearance and heuristics; we intend to continue to explore the use of psychological theories and biomechanics to motivate choices in shoulder-turning based on comfort and impending density, as well as the current clearance. These ideas can be useful in determining an appropriate orientation in cases our optimization does not yield a unique solution. We intend to explore the application of more direct collision checking in the lateral step algorithm as well. Moreover, we would like to further improve the runtime performance using parallelization techniques.

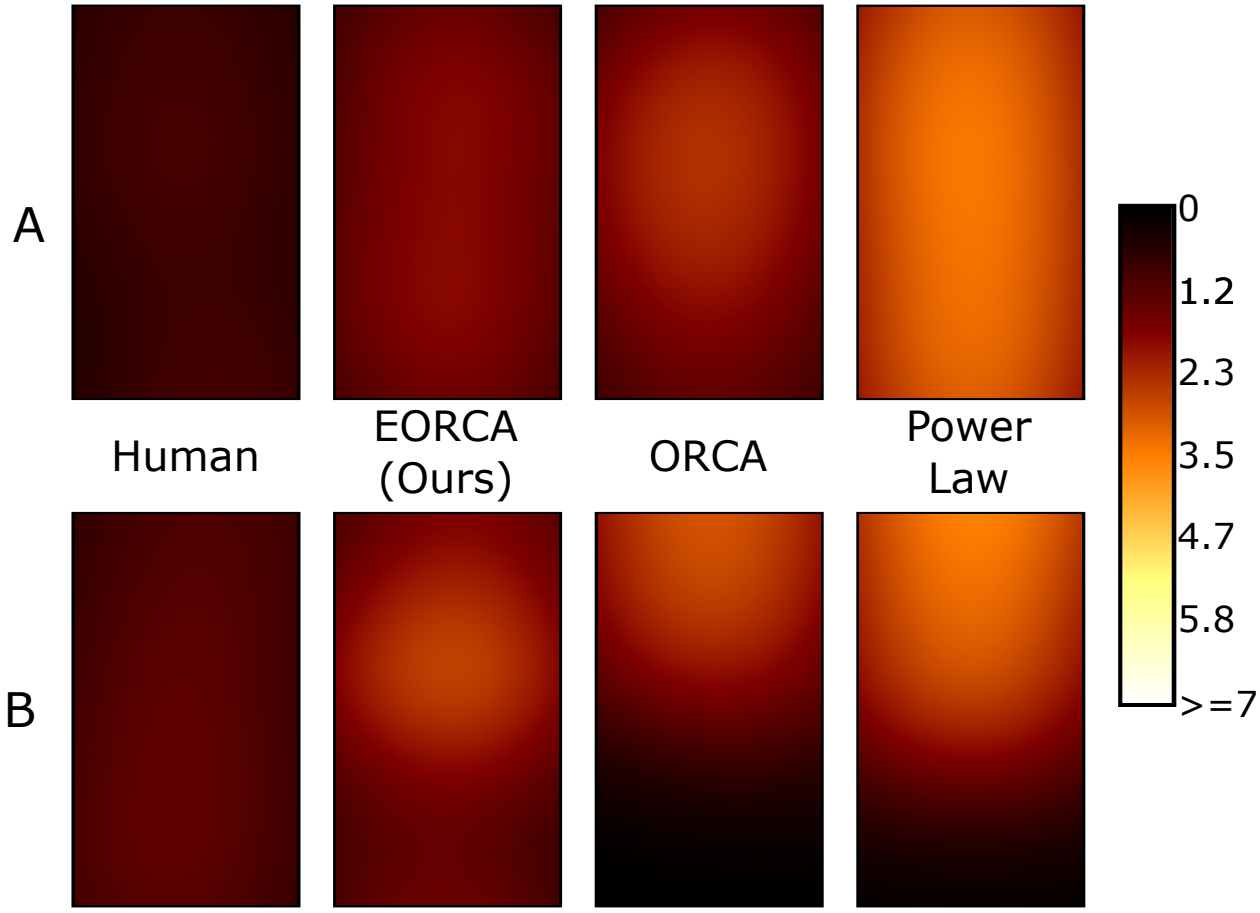
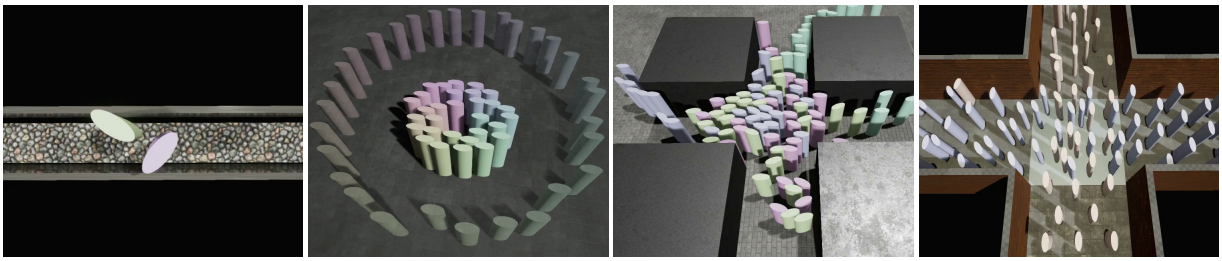


Figure 2.10: **Observed density for bidirectional flow experiment.** We compare the densities observed in the captured human data during a bidirectional flow experiment with simulations using: our method EORCA, ORCA (Van den Berg et al., 2011), and Power Law (Karamouzas et al., 2014). Each image represents a heatmap constructed from a 3m x 8m section of the corridor at a specific point during the experiment / simulation. **A) 20% complete**: As agents amass at the center of the corridor, the Power Law method experiences congestion, creating unnaturally high pedestrian densities. ORCA produces density more consistent with human data but shows congestion in the midpoint of the corridor. EORCA generates the closest results to the captured human data, with a smooth density consistent across the corridor. **B) 60% complete**: Both ORCA and Power Law experience substantial congestion on either side of the corridor midpoint, leading to empty space and a lack of flow. EORCA maintains flow despite congestion near the midpoint and better matches the human data.



(A) (B) (C) (D)

Figure 2.11: **Experimental Scenarios.** (A) Two agents approach and rotate in the narrow hallway. (B) 30 agents cross the antipodal circle. (C) 100 Agents cross through the center of the 4-square scene. (D) Two groups of 44 HRP-4 form-factor agents cross one-another.

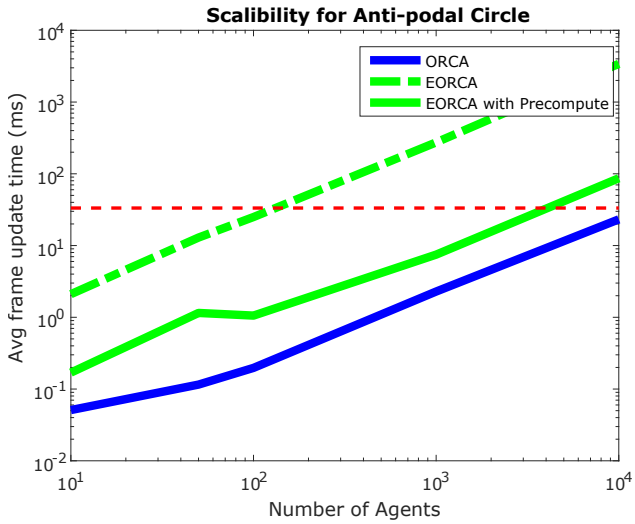


Figure 2.12: **Relative Performance of ORCA and EORCA-C (with and without precomputation).** The average frame update time for EORCA with precomputation is 4-5x slower than ORCA but still interactive for 1000's of agents, shown by the red line denoting 30 FPS. It is also two orders of magnitude faster than EORCA without precomputation.

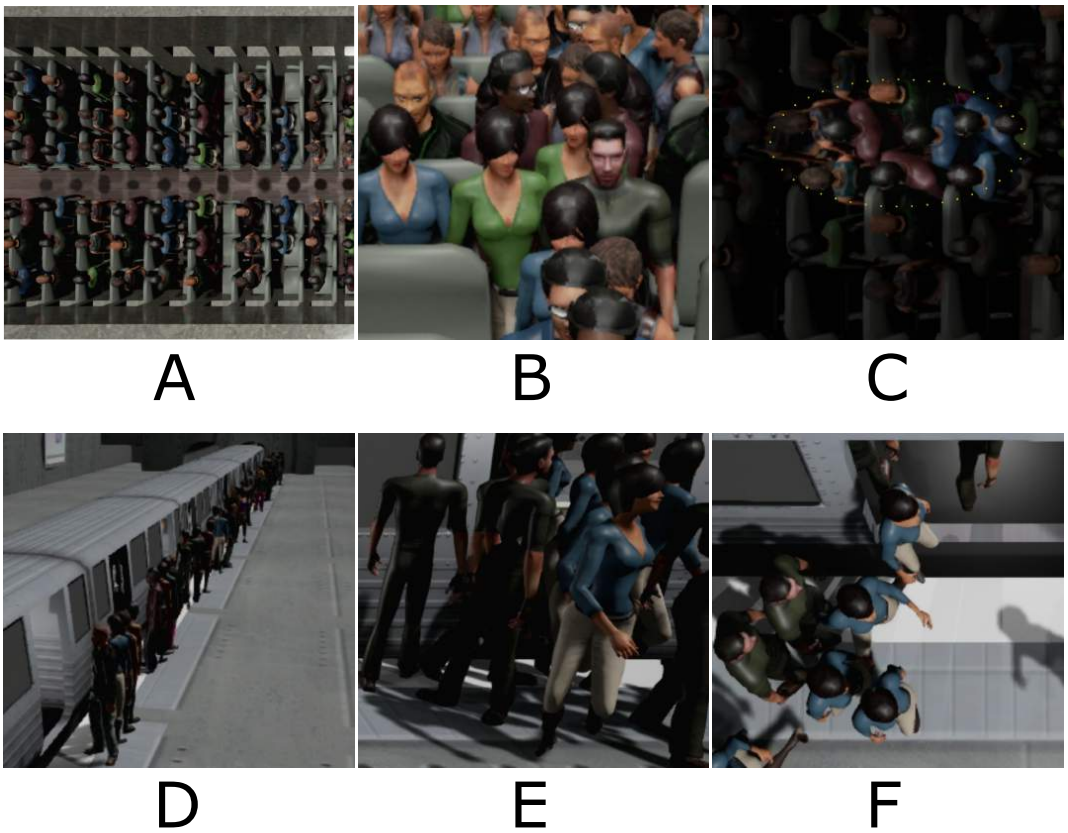


Figure 2.13: **Applications.** We simulate pedestrians in real time with 2D ellipses and visualize the results in 3D during post processing. (Top) Pedestrians exit a commercial aircraft. Our algorithm is able to capture the lateral steps individuals take when exiting from their row and the shoulder-turning that occurs as they pass one another in the aisle. (Bottom) Pedestrians enter a subway station and wait for the train to arrive. When the doors open, the waiting pedestrians step back to let those aboard exit before entering the subway train. Agents side-step, backpedal (F) and shoulder-turn (E) as they make their way into and out of the train.

CHAPTER 3: Generating Collision-free Dynamic Maneuvers for Autonomous Vehicles

3.1 Introduction

Autonomous driving is a difficult and extremely complex task that has immense potential for impacting the lives of billions of people. In order to develop autonomous capabilities to perform the driving task, we need appropriate capabilities to sense and predict the traffic and road obstacles, as well as for planning, control, and coordination of the vehicle (Ziegler et al., 2014a; Pendleton et al., 2017). There is considerable research in this area that borrows ideas from different disciplines including computer vision, machine learning, motion planning, mechanical engineering, intelligent traffic simulation, human-factors psychology, etc.

Research in the field of autonomous driving has progressed considerably in both sensing and planning, but many problems remain to be solved. Vehicle sensors have the capability to detect many relevant obstacles, vehicles, and other traffic entities including bicycles and pedestrians. However, current techniques are not yet robust and are sensitive to lighting conditions, weather, distortion, and motion. In addition, automatic planning in different scenarios and the computation of the appropriate response to vehicle and non-vehicle entities, such as bicycles and pedestrians, are still the subjects of ongoing research. Moreover, the uncertainties in the sensor data, the capability, and response of the autonomous vehicle, typically referred to as the *ego-vehicle* (Ziegler et al., 2014b), have led to the development of behavior and navigation algorithms that impose conservative limits on the acceleration, deceleration, and steering decisions. For example, algorithms tend to limit hazard responses to either steering only (Borrelli et al., 2005; Eidehall et al., 2007) or braking only (Distner et al., 2009). Few algorithms demonstrate combined control of throttle and steering and typically do so in constrained navigation scenarios (Turri et al., 2013). In terms of planning the routes and navigating the roads, current algorithms tend to limit the lane-changing behaviors, precluding their use for progressing more quickly to a goal or better utilization of the road conditions. These limitations have led to the perception that autonomous cars behave more like student drivers taking their driving test than actual skilled human drivers (Ziegler et al., 2014b). One of the goals is to extend the capabilities of current autonomous vehicles in

terms of planning, control, and navigation, making them less conservative but still allowing safe performance during driving.

There are numerous barriers to developing novel autonomous driving algorithms. Safety concerns pose a significant constraint on the development novel approaches. In order to facilitate acceptance and guarantee safety, vehicles must be tested not only in typical, relatively safe scenarios, but also in dangerous, less frequent scenarios, which can pose a risk to the public or the researchers themselves. Aside from safety concerns, costs pose an additional challenge to the testing of autonomous driving algorithms. Each new configuration of a vehicle or new sensor requires re-calibration of a physical vehicle, which is labor intensive. Furthermore, the vehicle can only be tested under conditions limited either by a testing track, or the current traffic conditions if a road test is being performed. This means the vehicle can be tested no faster than real-time and without any speedups or parallel testing.

Many recent approaches to autonomous driving additionally rely on machine-learning via Bayesian networks or deep-learning to provide entity detection (Mendes et al., 2016), entity prediction (Galceran et al., 2015a), and end-to-end control (Bojarski et al., 2016). However, such approaches rely on substantial amounts of annotated data in safe, as well as dangerous scenarios. The dataset must also encompass varied weather and lighting conditions. In addition, not all autonomous vehicles are equipped with identical or equivalent sensing capability; training data must be available for the specific configuration or sensors of the vehicle being tested. Gathering such data by physical tests can be expensive, difficult, and even dangerous. In contrast, a high-fidelity simulator can augment and improve training of algorithms, and allow for testing safely and efficiently. Insights gained from simulation could provide critical training data and information on algorithmic inefficiencies before actual vehicle testing.

Main contributions: In this chapter, we present a novel navigation algorithm for autonomous vehicles, AutonoVi, which utilizes a data-driven vehicle dynamics model and optimization-based maneuver planning to compute a safe, collision-free trajectory with dynamic lane-changes. Our approach is general, makes few assumptions about the traffic conditions, and plans dynamically feasible maneuvers in traffic consisting of other vehicles, cyclists, and pedestrians. In order to develop an autonomous vehicle planning approach with these capabilities, we present four novel contributions:

- **Optimization-based Maneuvering:** We describe a novel multi-objective optimization approach for evaluating the dynamic maneuvers. Our algorithm encodes passenger comfort, safe passing distances,

maneuver constraints in terms of dynamics, and global route progress in order to compute appropriate trajectories.

- **Data-driven Vehicle Dynamics:** We use a data-driven vehicle dynamics formulation that encodes feasible accelerations, steering rates, and decelerations into a set of per-vehicle profile functions, which can be quickly evaluated. These profiles are generated by simulating the ego-vehicle through a series of trials to obtain lateral and longitudinal slip profiles. This data-driven model generalizes to multiple vehicles and configurations.

- **Collision Avoidance with Kinematic and Dynamic Constraints:** We present a collision avoidance algorithm that combines collision-free constraints with specific kinematic and dynamic constraints of the autonomous vehicle. Our approach allows the autonomous vehicle to steer away from collisions with other vehicles, pedestrians, and cyclists as well as to apply brakes, or use a combination of steering and braking.

- **Trajectory Planning with Traffic Rules and Behaviors:** We present a trajectory planning algorithm that encodes traffic rules and road behaviors along with lane-following for computing safe trajectories. Our approach is based on computing arcs along the center-line of the current lane to generate an initial path that satisfies all the constraints. This initial path is computed and refined according to collision avoidance and maneuver optimization computations.

In addition, in an effort to facilitate progress in autonomous driving generally, we present AutonoVi-Sim, a simulation framework for training and testing autonomous driving algorithms and sensors. AutonoVi-Sim is a collection of high-level, extensible modules designed to allow researchers and engineers to rapidly configure novel road networks, driving scenarios, and vehicle configurations, and to test these in a variety of weather and lighting conditions. AutonoVi-Sim captures a variety of autonomous driving phenomena and testing requirements including:

- **Data Generation**: Autonovi-Sim facilitates data analysis by allowing exports of relevant data for traffic proximate to the autonomous vehicle as well as data from each virtual sensor on the vehicle. Sensor and local traffic data can be used in training deep-learning approaches by generating automatically labeled classification and decision data efficiently.

- **Varying Vehicle, Cyclist, Pedestrian, and Traffic Conditions**: AutonoVi-Sim includes various vehicle and sensor models, pedestrians, and cyclists. Diversity of these traffic entities allows for training classification on differing shapes, sizes, colors, and behaviors of cyclists, pedestrians, and other drivers.

- **Dynamic Traffic, Weather and Lighting Conditions**: AutonoVi-Sim provides high fidelity traffic simulation, supporting dynamic changes in traffic density, time of day, lighting, and weather including rain and fog.

- **Rapid Scenario Construction**: Typical road networks can be easily laid out using spline painting and are automatically connected for routing and navigation purposes. AutonoVi-Sim supports many lane configurations and atypical road geometry such as cloverleaf overpasses. In addition, other vehicles and entities can be scripted to generate repeatable erratic behavior, e.g. cutting in front of the ego-vehicle, walking into the road.

Leveraging AutonoVi-Sim, we evaluate our algorithm in a set of traffic scenarios in both sparse and dense traffic conditions with tens of other vehicles, pedestrians, and cyclists. We demonstrate collision-avoidance events including a vehicle suddenly driving into the road, traffic suddenly stopping ahead of the ego-vehicle while travelling at high speed, and a pedestrian jaywalking in front of the ego-vehicle, representing typical accident scenarios (Eidehall et al., 2007). Our approach enables advantageous use of lane changes (e.g., overtaking) and adherence to traffic rules in typical traffic conditions. It also exhibits safe maneuvering in the presence of heavy traffic, pedestrians, and cyclists.

The rest of the chapter is organized as follows. We detail relevant related work in Section 3.2. In Section 3.3, we introduce the vehicle kinematic model, define relevant assumptions, and introduce the terminology used in the rest of the chapter. In Section 3.4, we present our navigation algorithm, AutonoVi, and its components. In Section 3.5 and Section 3.6, we describe AutonoVi-Sim, our simulation framework for testing autonomous driving algorithms. We present the details of our simulation benchmarks, describe results of our algorithm, and detail performance results of our simulation framework in Section 3.7.

3.2 Related Work

The problem of autonomous driving has been widely studied in robotics, computer vision, intelligent transportation systems and related areas. In this section, we give a brief overview of prior methods which

address motion planning and navigation, dynamics, behavior generation, and collision avoidance. More detailed surveys are given in Pendleton et al. (2017); Katrakazas et al. (2015); Saifuzzaman and Zheng (2014); Chen and Cheng (2010).

3.2.1 Vehicle Kinematics and Dynamics Modeling

A number of approaches have been developed to model the motion of a moving vehicle, offering trade-offs between simplicity or efficiency of the approach, and physical accuracy. Simpler models are typically based on linear dynamics and analytical solutions to the equations of motion (LaValle, 2006). More accurate models provide a better representation of the physical motion, but require more computational power to evaluate and incorporate non-linear forces in the vehicle dynamics (Borrelli et al., 2005; Peters et al., 2011; Rucco et al., 2015). The Reeds-Shepp formulation is a widely used car model with forward and backward gears (Reeds and Shepp, 1990). Margolis and Asgari (1991) present several representations of a car including the widely used single-track bicycle model. Borrelli et al. (2005) extend this model by including detailed tire-forces. Our simulator leverages the NVIDIA PhysX engine for the underlying vehicle model.

3.2.2 Path Planning and Collision Avoidance

Prior approaches to path planning for autonomous vehicles are based on occupancy grids (Kolski et al., 2006), random-exploration (Kuwata et al., 2009), driving corridors (Hardy and Campbell, 2013), potential-field methods (Galceran et al., 2015b), avoiding inevitable-collision states (ICS) (Martinez-Gomez and Fraichard, 2009) etc. Recent approaches seek to incorporate driver behavior prediction in path planning using game-theoretic approaches (Sadigh et al., 2016) and Bayesian behavior modeling (Galceran et al., 2015a). In addition, a variety of algorithms have been proposed for planning paths for automobiles for navigation outside of road conditions and traffic rules (Ziegler et al., 2008; Hoffmann et al., 2007). Several techniques have been proposed to specifically avoid hazards while remaining in a target lane. These techniques can be coupled with a path planner to avoid vehicles (Fritz et al., 2004) and other hazards in the ego-vehicle's lane (Turri et al., 2013).

Many continuous approaches for collision-avoidance have been proposed based on spatial decomposition or velocity-space reasoning. Van Den Berg et al. (2011) apply velocity-space reasoning with acceleration constraints to generate safe and collision-free velocities. Bareiss and Van den Berg (2015) extend the concept of velocity obstacles into the control space to generate a complete set of collision-free control inputs. Ziegler

et al. (2014a) utilize polygonal decomposition of obstacles to generate blockages in continuous driving corridors. Sun et al. (2014) demonstrate the use of prediction functions and trajectory set generation to plan safe lane-changes.

3.2.3 Modeling Traffic Rules

Aside from planning the appropriate paths to avoid collisions, autonomous vehicles must also follow applicable laws and traffic norms. Techniques have been proposed to simulate typical traffic behaviors in traffic simulation such as Human Driver Model (Treiber et al., 2006) and data-driven models such as that of Hidas (2005). Logic-based approaches with safety guarantees have also been demonstrated (Tumova et al., 2013). An extensive discussion on techniques to model these behaviors in traffic simulation can be found in Chen and Cheng (2010). Our simulator allows for modeling such traffic behaviors as well as traffic control strategies at the infrastructure level.

3.2.4 Autonomous Driving Systems

Many autonomous systems have been demonstrated that are able to navigate an autonomous vehicle in traffic along a specific route. Ziegler et al. (2014b) demonstrated an autonomous vehicle which drove the historic Bertha Benz route in southern Germany. They use a conservative navigation approach, which specifically encodes *lanelets* for lane changing and does not account for dynamic lane changes. In contrast, our algorithm allows the vehicle to change lanes when our maneuver optimization method deems it appropriate and does not rely on pre-encoded changes. Geiger et al. (2012) demonstrate a planning and control framework that won the Grand Cooperative Driving Challenge in 2011. This vehicle was designed for platooning and employed controls over acceleration only. Our navigation algorithm plans maneuvers using both steering and acceleration to operate in more generic traffic scenarios. The DARPA Urban Grand Challenge included a number of autonomous vehicle navigating examples of driving scenarios (Buehler et al., 2009). While overtaking was allowed as an intended capability in these systems, the vehicles were not evaluated in dense, high-speed traffic conditions where the benefits of lane changes could be demonstrated.

3.2.5 Data Generation

Recent studies support the use of high-fidelity microscopic simulation for data-gathering and training of vision systems. Richter et al. (2016) and Johnson-Roberson et al. (2017) leveraged Grand Theft Auto 5 to

train a deep-learning classifier at comparable performance to manually annotated real-world images. Several recent projects seek to enable video games to train end-to-end driving systems, including ChosunTruck[1] and DeepDrive-Universe[2] which leverages the OpenAi Universe system. Using video game data provides benefits in the fidelity of the vehicle models but limits the ability to implement sensing systems and access data beyond visual data. A fully dedicated high-fidelity simulator can address these limitations and provide access to point-cloud data, visual data, and other vehicle sensors without the limitations imposed by adapting gaming software. Research in this area has begun to emerge (Dosovitskiy et al., 2017). Our work is complimentary to such systems and can be combined with generated data from other simulators to increase robustness of training data.

3.3 Problem Formulation

In this section, we introduce the notation, the kinematic and dynamics model of the car and the state space of the vehicle in terms of both physical configuration and behavior space.

3.3.1 Vehicle State Space

We represent the kinematic and dynamic constraints of the vehicle separately. In terms of trajectory planning, steering and throttle controls that could lead to skidding or a loss of control are first excluded in our dynamics model (see Section 3.4.6) and future trajectories are computed according to our vehicle kinematic model described in Equation (3.1).

We extend the simple-car kinematic model (LaValle, 2006; Laumond et al., 1998). The vehicle has three degrees of freedom in a planar coordinate space. These are the position of the center of mass $\vec{p} = (p_x, p_y)$, and the current heading or orientation θ. We represent the speed of the vehicle as v and steering as ϕ. L_f and L_r represent the distance from the center of mass to the front and rear axles, respectively. The geometry of the ego-vehicle is represented as $\mathcal{O}_e$.

The vehicle has two degrees of control, throttle (u_t) and steering (u_ϕ). We define throttle $-1 \leq u_t \leq 1$, where -1 indicates maximum braking effort for the vehicle and 1 represents maximum throttle. $-1 \leq u_\phi \leq 1$ describes the steering effort from $-\phi_{max}$ to ϕ_{max}.

[1] https://github.com/bethesirius/ChosunTruck

[2] https://github.com/OSSDC/deepdrive-universe

We also use acceleration and steering functions, $A(v, u_t)$ and $\Phi(v, u_s)$, respectively, which describe the relationship between the vehicle's speed, steering, and control inputs and its potential for changes in the acceleration and steering (see Section 3.4.6). A and Φ can be chosen to be constants in the simplest model, or may be represented using complex functions corresponding to tire dynamics and load transfer. We describe our choice for A and Φ in Section 3.4.6. The vehicle's motion can be described by:

$$\dot{p}_x = v\cos(\theta) \qquad \dot{p}_y = v\sin(\theta) \qquad \dot{\theta} = \frac{\tan(\phi)}{L_f + L_r}v \tag{3.1a}$$

$$\dot{v} = A(v, u_t) \qquad \dot{\phi} = \Phi(\phi, u_s) \tag{3.1b}$$

In addition to the physical state of the vehicle, we describe its behavior b as a label from a set of all behaviors $\mathcal{B}$, such as driving straight, turning left, merging right, etc. The behavior state is used to modify parameters of each stage of the algorithm. Each behavior state can encode a set of weights of the maneuver optimization function, bias the generation of a guiding path, and adjust the sampling bias of our augmented control-obstacle approximation and acceleration when necessary (see Section 3.4.1). The full state of a vehicle is defined as $X_e = \{p_x, p_y, v, \phi, u_t, u_\phi, b\}$. The vehicle updates its plan at a fixed planning rate Δt. At each planning step, the vehicle computes a target speed v' and target steering ϕ' to be achieved by the control system. We refer to Equation (3.1) compactly as the *state evolution function* $X_{t+\Delta t} = q(X_t, u, t)$. We also define a function $S(u, X)$ which determines if a set of controls is feasible. Given the current state of the vehicle, $S(u, X)$ will return false if the given input u will cause a loss of traction or control. We describe this function further in Section 3.4.6.

3.3.2 Sensing and Perception

We assume the vehicle is equipped with a sensing module capable of providing information regarding the surrounding environment. For each lane on a road, the sensing module provides an approximation of the center line of the lane, l, the closest point on the lane center to the ego-vehicle, $\vec{l}_p$, and a reasonable value of the friction coefficient μ of the road. Approaches have been presented to evaluate μ from sensor data (Gustafsson, 1997). Our navigation algorithm utilizes the set of nearby vehicles, pedestrians, bicycles, or other obstacles, collectively referred to as neighbors, N within the sensing range. For each neighbor $n \in N$, the sensing system provides the neighbor's shape, $\mathcal{O}_n$, position, $\vec{p}_n$, and velocity $\vec{v}_n$. Where possible,

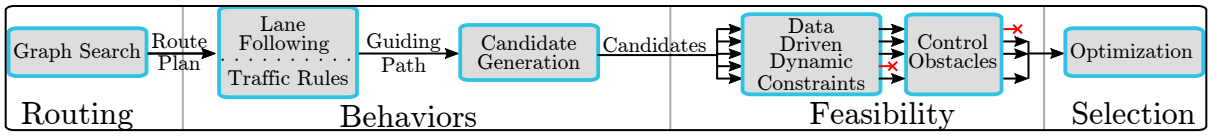


Figure 3.1: **AutonoVi Algorithmic Pipeline:** Our autonomous vehicle planning algorithm operates in several sequential steps. First, a route is planned using graph-search over the network of roads. Secondly, traffic and lane-following rules are combined to create a guiding path for the vehicle for the next planning phase. This guiding path is transformed to generate a set of candidate control inputs. These controls are evaluated for dynamic feasibility using our data-driven vehicle dynamics modeling and collision-free navigation via augmented control obstacles. Those remaining trajectories are evaluated using our optimization technique to determine the most appropriate set of controls for the next execution cycle.

the sensing module provides the current lane l_n, current acceleration $\dot{v}_n$, and current rate of turn, $\dot{\theta}$ for the neighbor. We define a set of neighbor types, $\mathcal{T}$, including vehicle, pedestrian, cyclist, and obstruction. Each neighbor is assigned a type $\mathcal{T}_n$ corresponding to the detected neighbor type. The complete state of a neighbor is denoted as $X_n = \{\vec{p}_n, \vec{v}_n, l_n, \dot{v}_n, \dot{\theta}_n\}$.

3.4 Navigation Algorithm

In this section, we describe our navigation algorithm. Our algorithm operates in four sequential stages, shown in Figure 3.1. First, a route is constructed over the space of roads in the environment. Secondly, a *Guiding Path* that follows the current lane is computed that provides input to the collision-avoidance and optimization-based maneuver stages. The collision avoidance stage leverages an augmented control obstacle formulation to determine the set of feasible *candidate controls* that represent dynamically feasible, collision-free controls for the vehicle. Finally, a new control is chosen for the vehicle based on the optimization-based maneuver function.

3.4.1 Route Choice and Behavior State

In the first step of the algorithm, a global route for the vehicle to follow to the goal is determined. This step is performed only once unless special conditions (e.g., missing a turn) force the vehicle to recompute a route. The ego-vehicle is provided a connected graph of roads in the environment from a GIS database. Each road in the graph contains information on the number and configuration of lanes in the road and the speed limits. When a destination is chosen, we use A* search to compute the shortest route to the goal and construct a route plan. Other recent approaches have been proposed to compute routes on large-scale road

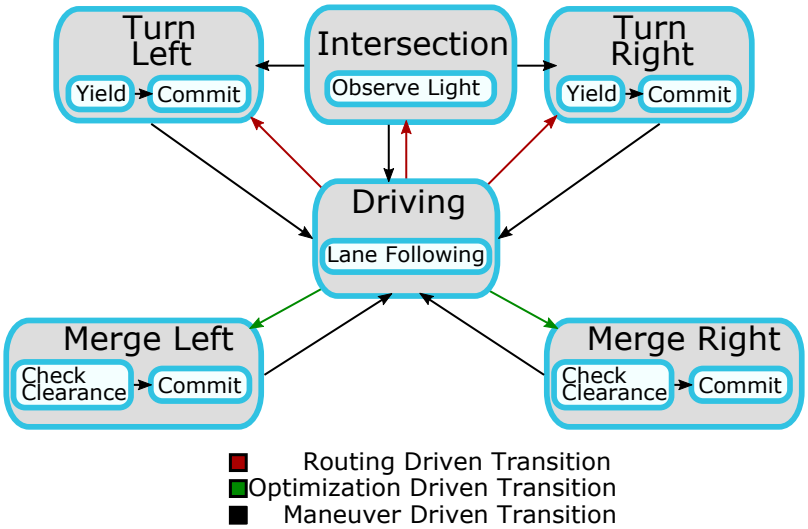


Figure 3.2: **Hierarchical Finite State Machine:** We conceptualize local decision-making as a hierarchical finite-state machine. We highlight different behavior states that are determined by the routing and optimization algorithms. When executing turns, the routing algorithm transitions the behavior state to a turning state. When the optimization-based maneuver algorithm plans a lane change, the behavior state is transitioned to merging. Behavior states can bias control sampling and can encode specific weights for the optimization function.

networks (Delling et al., 2009; Bast et al., 2016). Our multi-stage navigation approach is complementary to these approaches and can leverage them for the first stage of route computation.

Each step of the route plan encodes how the vehicle transitions from one road to the next. We denote these as *road-transition maneuvers*. A road-transition maneuver consists of the valid source lanes, valid destination lanes, the position along the road at which the maneuver begins, denoted $\vec{p}_m$, and the behavior implied by the road transition. The set of behaviors includes merging, right turns, left turns, and driving straight. Once the road-change maneuver is completed, the vehicle navigates along the lanes of the new road until the next maneuver node is reached. Lane changes are not encoded in the maneuver nodes, but they are performed implicitly based on the optimization function described in Section 3.4.5.

The behavior state of the vehicle is described by a limited hierarchical finite-state machine shown in Figure 3.2. It is used to restrict potential control decisions and adjust the weight of the cost-function for specific maneuvers, such as turning. This allows our algorithm to force the vehicle to be more conservative when performing delicate maneuvers. For example, the valid steering space is constrained in turns to guarantee that the vehicle moves closely along the center line.

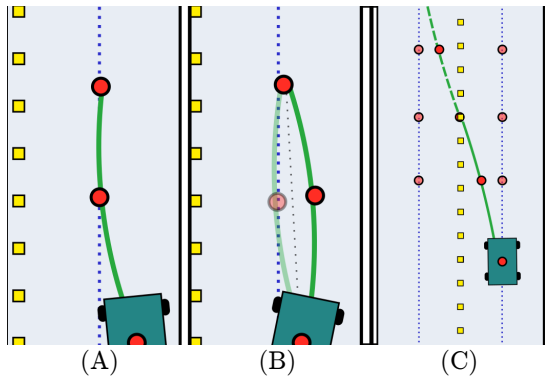


Figure 3.3: **Guiding Path Computation:** The vehicle computes a guiding path to the center of its current lane based on a circular arc. **(A):** When the vehicle tracks a path off the center of its current lane, the guiding path leads it smoothly back to center. **(B):** In cases where the guiding path represents abrupt changes to heading, the center point is reflected about the axis formed by the car's position and the final waypoint. **(C):** In the case of lane changes, the guiding path is computed by a weighted average of the waypoints on the departure and destination lanes.

3.4.2 Guiding Path

The ego-vehicle computes a set of waypoints, denoted $\{\vec{w}_1, \vec{w}_2 ... \vec{w}_k\}$, along the center-line of its current lane at fixed time intervals, which represent the expected positions for a planning horizon, τ. Using its own position, the median point, and the final waypoint ($\vec{p}, \vec{w}_{\frac{k}{2}}, \vec{w}_k$), we compute a circular arc on the road plane which sets the initial target speed and steering, v' and ϕ' respectively, and acts as the guiding path for the next planning phase. We use circular arc approximations because they implicitly encode the radius of curvature needed for slip computation, making it easy to check whether the dynamic constraints are violated. Absent discontinuities in the center-line of the lane, the guiding paths exhibit first-order C^1 continuity. Figure 3.3(a) demonstrates a guiding path constructed for a sample lane.

We constrain the arcs to lie within the first two quadrants of the circle that is represented by three waypoints. In cases when the vehicle's trajectory tracks away from the center of the lane, e.g. during collision avoidance maneuvers, this constraint may be violated, as shown in Figure 3.3(b). In such cases, the point $\vec{w}_{\frac{k}{2}}$ is reflected about the axis formed between $\vec{p}$ and $\vec{w}_k$ to correct the arc angle. In case of lane changing, waypoints are constructed from a weighted average of points sampled ahead on both the departure lane and the destination lane. Figure 3.3(c) demonstrates a set of lane-change arcs.

Given a guiding path, a target steering ϕ' is computed from Equation (3.1a). The radius of the arc, r, is substituted into Equation (3.11) to determine the maximum safe speed for the current road curvature. A target speed, v', is computed from the minimum value of the current speed limit and the maximum safe speed.

The target steering and speed form the basis of the augmented control obstacle exploration in the subsequent stage.

3.4.2.1 Traffic Rules

Many rules of the road, such as maintaining safe distances, driving on the correct side of the road, avoiding collisions, obeying speed limits, and yielding are captured implicitly in our algorithm. Traffic rules such as stopping at red lights and stop signs are encoded explicitly. When choosing a target speed v', the sensing system is referenced to determine if an intersection, round-about, merge, etc. is being approached and whether the vehicle needs to stop. In cases where the vehicle must stop, the edge of the intersection is used to compute a stopping point and v' is set to the speed that will reach the stopping point at τ seconds. In case of stoplights, the green light signals v' to return to its original value. In the case of stops with continuous cross-traffic, the vehicle waits until the collision-avoidance algorithm indicates safety. In the case of all-way stops, the vehicle maintains a queue of vehicle arrival order, but defers to other drivers if they enter the intersection out of turn.

3.4.3 Collision Avoidance

We leverage the theory of *Control Obstacles* for collision avoidance (Bareiss and Van den Berg, 2015). Control obstacles construct constraints in the control space and are an extension of acceleration-velocity obstacles (Van Den Berg et al., 2011). For each neighbor of the ego-vehicle, n, we define the control obstacle for the neighbor as the union of all controls that could lead to collisions with the neighbor within the time horizon, τ. Given t, where $0 \leq t \leq \tau$, the relative position of the ego-vehicle and neighbor $\vec{p}_{en}$ must remain outside the Minkowski Sum given by the formulation, which is defined as

$$\mathcal{O}_{en} = \mathcal{O}_n \oplus -\mathcal{O}_e. \tag{3.2}$$

The complete derivation for control obstacles can be found in Bareiss and Van den Berg (2015).

In order to adapt to autonomous vehicles, we modify the original control obstacle formulation (Bareiss and Van den Berg, 2015) in the following manner: (1) we do not assume reciprocity in collision avoidance and the ego-vehicle must take full responsibility for avoiding collisions; (2) we do not assume the control inputs of other vehicles are observable, which is consistent with the first point; (3) we do not assume bounding

discs for the neighboring entities, but rather a tight bounding rectangle. The Minkowski Sum for two convex polygons can be computed in linear time in the number of edges; (4) we choose a new feasible control by minimizing the objective function defined in Section 3.4.5, rather than by minimizing the deviation from v' and ϕ'.

The union of all control obstacles and the set of dynamically infeasible controls form the boundary of the space of admissible controls for the ego-vehicle. As long as a new control set is chosen from this space, the ego-vehicle will be collision-free for the next τ seconds. We refer to our novel control obstacle formulation as *augmented control obstacles*. For an ego agent e, a set of neighbors $j \in N$, the state evolution function q, a control-evolution function g_e given by integrating $A(v, u_t)$ and $\Phi(\phi, u_s)$, and g_j given by the estimations in Section 3.4.4, we can describe the augmented control obstacle set by the following constraint formulas:

$$\forall(j \in N, 0 < t < \tau) \quad :: \quad (\mathcal{O}_e \oplus \{q_e(g_e(t, x_e, u_e + \Delta u_e))\}) \cap (\mathcal{O}_j \oplus \{q_j(g_j(t, x_j, u_j + \Delta u_j))\}) = \emptyset \tag{3.3}$$

$$\forall(0 < t < \tau) \quad :: \quad S(u, x_{e,t}) = 1 \tag{3.4}$$

This approach is conservative and it is possible that there may be no feasible solution. In that case, we reduce τ and search for a feasible solution.

3.4.4 Trajectory Sampling

Computing the exact boundary of the augmented control obstacle space is computationally expensive. Moreover, depending on the choice of A and Φ, the boundary computation will typically not have an analytical solution. In order to ensure that the vehicle can plan within a specific time bound, we use a sampling strategy around ϕ' and v' to determine a feasible control that the vehicle will adopt for the next τ seconds. Each sample is referred to as a *candidate control* and represented as u_c.

First, the closest collision-free velocity to v' is determined where $\phi = \phi'$ by forward projection. This represents the largest speed the vehicle could take without deviating from the center-line of its lane and is always included in the set of candidates. Next, we compute evenly spaced samples around the point (v', ϕ') in the control space.

For each neighbor $n \in N$, we compute a set of states for that neighbor for the next τ seconds by forward integration of $q(X_n, \emptyset, t)$. We assume the neighbor moves along its current velocity $\vec{v}_n$ subject to the current

observed values of turning and acceleration, $\dot{\theta}$ and $\dot{v}_n$, respectively. If the lane information is available, we assume that the neighboring vehicle will follow its current lane at the current speed subject to the current acceleration during this time interval.

For each candidate control, u_c, we determine whether Equation (3.11) is violated by the candidate control inputs and immediately discard it if that is the case. If not, the sample points are computed at even time intervals along $0 \leq t \leq \tau$ by forward integration of $q(X_t, u_c, t)$. For each position in time, $\vec{p}_t$, we compute the relative position with each neighboring position at that time and determine if the relative position lies inside the Minkowski Sum. If so, we discard the candidate controls. After all the candidates are evaluated, the new control sequence is chosen by minimizing the objective function described in the subsequent section.

3.4.5 New Trajectory Computation

Once a set of suitable control candidates has been computed, the vehicle selects the valid controls that minimize the following cost function at each sample point $i \in I$:

$$C = \sum_{i=0}^{I} c_{path}(i) + c_{cmft}(i) + c_{mnvr}(i) + c_{prox}(i). \tag{3.5}$$

This function corresponds to producing paths which are comfortable for passengers, provide safe passing-distances from other vehicles, and respect the constraints of upcoming maneuvers the vehicle must perform. Each term consists of several cost evaluation functions, each with its own weight $e \in W$, which are described in the following sections.

3.4.5.1 Path Costs

c_{path} encodes costs associated with the vehicle's success at tracking its path and the global route. This is given as:

$$c_{path} = c_{vel} + c_{drift} \tag{3.6}$$

$$c_{vel} = (v' - v)^2; \qquad c_{drift} = ||\vec{p} - \vec{l}_p||^2 \tag{3.7}$$

c_{vel} is the squared difference between desired speed and current speed and c_{drift} is the squared distance between the center line of the vehicle's lane and its current position. If the path crosses a lane boundary, c_{drift} is computed with respect to the new lane. These terms drive the vehicle to choose trajectories that maximally progress the ego-vehicle along its computed route between steps.

3.4.5.2 Comfort Costs

Comfort costs are computed similar to Ziegler et al. (2014a) and penalize motions which are uncomfortable for passengers in the vehicle. c_{accel} penalizes large accelerations and decelerations. c_{yawr} penalizes large heading changes and discourages sharp turning. The comfort costs are given as:

$$\begin{aligned} c_{cmft} = \quad & c_{accel} + c_{yawr}; \\ c_{accel} = ||\dot{v}_i||; \quad & c_{yawr} = ||\dot{\theta}|| \end{aligned} \tag{3.8}$$

.

3.4.5.3 Maneuver Costs

The novel maneuvering cost function discourages lane-changes without excluding them and guides the vehicle to occupy the necessary lane for its next maneuver. The formulation is given as:

$$\begin{aligned} c_{mnvr} &= c_{lane} + c_{mdist} \\ c_{lane} &= 1 \cdot LaneChanged \\ c_{mdist} &= min(MaxLaneCost, \frac{1}{\vec{p} - \vec{p}_m}) \cdot WrongLane \end{aligned} \tag{3.9}$$

$LaneChanged$ is a boolean variable representing whether a candidate path crosses a lane boundary. $\vec{p}_m$ is the position of the next maneuver change, e.g. the beginning of a right turn. This position is determined by the point at which the maneuver starts in the desired lane for the maneuver. $WrongLane$ is a boolean that evaluates to true if the vehicle's lane does not match the lane for the next maneuver. If a candidate control is chosen where for some point $i \in I$, $LaneChanged$ evaluates to true, a lane change behavior is initiated in

the hierarchical finite-state machine. $MaxLaneCost$ is a constant representing the maximum absorbing cost of being in the wrong lane.

3.4.5.4 Proximity Costs

While the collision avoidance stage prevents the vehicle from colliding with neighbors, the proximity cost term is designed to prevent the vehicle from passing close to neighboring entities based on the identified type of the neighbor, T_n. This cost is represented as a cost distance term with exponential decay based on the relative positions of the ego-vehicle and its neighbor.

$$c_{prox} = \sum_{n=0}^{N} d(N_j, \vec{p}) \tag{3.10}$$
$$d(N_n, \vec{p}) = C_{type_{\mathcal{T}_n}} \cdot e^{-||\vec{p}_n - \vec{p}_e||}$$

C_{type} is a per-type constant cost value. C_{type} is larger for pedestrians and bicycles than for vehicles, and guides the ego-vehicle to pass those entities with greater distance.

3.4.6 Data-driven Vehicle Dynamics Model

In order to determine values for $A(v, u_t)$ and $\Phi(\phi, u_s)$, we use a novel data-driven approach to model the dynamics of the vehicle. For each ego-vehicle, data are collected by driving the vehicle from $v = 0$ to $v = v_{max}$ at the highest possible throttle without loss of traction. Similarly, for braking, the vehicle is decelerated from $v = v_{max}$ to $v = 0$ using the highest braking effort possible without loss of traction. Data are collected at 60Hz for these values: current speed, acceleration, and throttle/braking values. From these data, piecewise-quadratic functions are constructed by least squares fitting to represent the maximum available acceleration and braking given the current vehicle state. These values also define thresholds for the control safety function $S(u, X)$.

We determine $\Phi(\phi, u_s)$ by fixing the vehicle's speed and collecting data for instantaneous changes to the steering angle for a given u_s. We construct a piecewise- quadratic function by least-squares fitting to represent the vehicle's steering dynamics. Based on the value of μ given by the sensors, we determine the

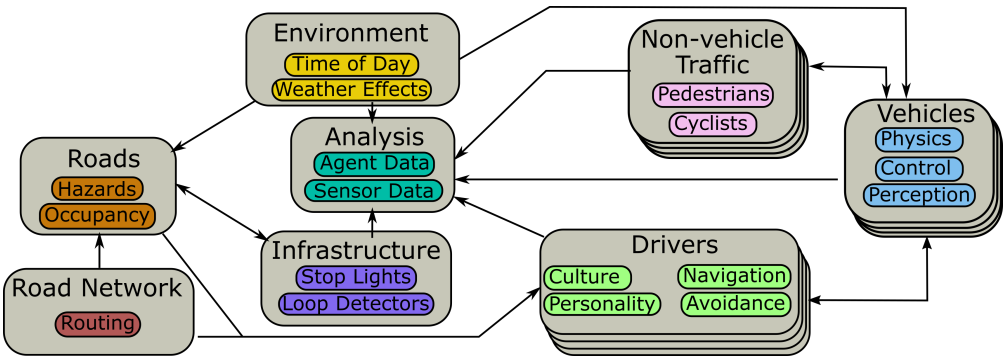


Figure 3.4: **AutonoVi-Sim Platform Overview:** The eight modules composing AutonoVi-Sim encompass varying aspects of autonomous driving. The *Road*, *Road Network*, and *Infrastructure* modules define the driving environment. The *Environment* module allows engineers to specify specific environment conditions including time of day and weather. The *Non-vehicle Traffic* module allows engineers to specify navigation goals for pedestrians and cyclists, or setup specific triggered behaviors. The *Drivers* and *Vehicles* modules work as a pair to define current traffic conditions and specific driving destinations and decisions for the vehicles in the simulation. Each vehicle in the simulation has a unique set of sensing capabilities and a single driver which operates the vehicle during the simulation. Finally, the *Analysis* module is used to catalog and export data, including agent positions and sensor readings, for analysis.

maximum feasible speed for a given curvature from the centripetal force equation:

$$v = \sqrt{\mu r g} \tag{3.11}$$

where r is the radius of curvature that is computed from Equation (3.1a). By substituting Equation (3.1a) into Equation (3.11), and the angular velocity formula $v = \omega \cdot r$, we can determine feasible steering for a given speed as

$$\phi = tan^{-1}(\frac{(L_f + L_r) \cdot \mu \cdot g)}{v^2}). \tag{3.12}$$

Given the generated functions S, A, and Φ, the future path of the vehicle can be evaluated quickly for planning future controls.

3.4.7 Control Input

The set of controls chosen by the optimization function is applied by a pair of PID controllers. One PID controller drives the current speed to match the target speed. The other drives the current steering angle to match the target steering angle. By limiting the choice of candidate controls to kinematically and dynamically feasible controls using our data-driven vehicle dynamics model, the PID controllers are sufficient to achieve the desired values.

3.5 AutonoVi-Sim: Simulation Modules

The subsequent two sections describe our novel autonomous driving simulation framework, AutonoVi-Sim. Drawing from recent work in crowd simulation, namely that of Curtis et al. (2016), AutonoVi-Sim is divided into eight extensible modules, each with various sub-components. The modules are Environment, Road Network, Road, Drivers, Infrastructure, Vehicles, Non-vehicle Traffic, and Analysis. Each module captures some aspect of autonomous driving simulation and can be extended and modified to suit the specific needs of a particular algorithm. Figure 3.4 shows the connection between components in AutonoVi-Sim. In this section, we will detail the modules which make up the basic simulation system, reserving discussion of the vehicle and driving strategy modules for Section 3.6.

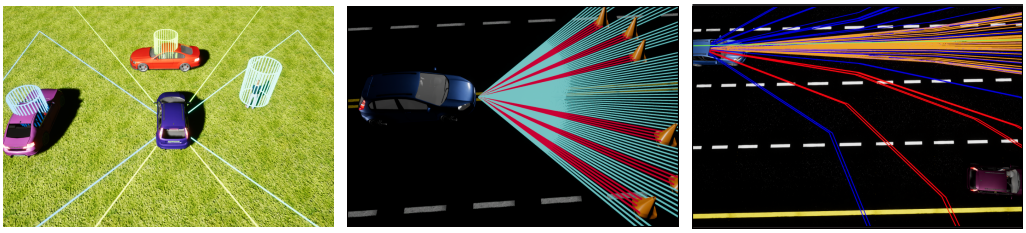
3.5.1 Roads

Roads in AutonoVi-Sim are represented by their center line, a number of lanes and directions thereof, and the surface friction of the road. Roads are placed interactively by drawing splines on a landscape which allows quick construction. Each road maintains occupancy information, average flow, and can maintain hazard information. The road module also maintains the set of hazards such as potholes or debris, which can be specified by density (number of hazards per km) or interactively by placing them on the road.

Alternately, roads can be specific pieces of geometry as in the case of intersections. This provides the flexibility to place specific intersections and model atypical road constructions for modelling specific environments.

3.5.2 Infrastructure

Infrastructure controllers represent traffic lights, signage, and any other entity which modifies the behaviors of vehicles on the road. These controllers can be added specifically to roads, as in the case of intersections, or placed independently as in signage or loop detectors. Vehicles implement their own detection of these entities as is described in Section 3.6.1.2. Infrastructure components are provided with basic detection capability and agency. For example, traffic lights can determine which lanes are congested and adjust the light cycle accordingly to move traffic more effectively.



(A) (B) (C)

Figure 3.5: **AutonoVi-Sim Sensor Modules: (A)**: Sensors on the vehicle are placed interactively. The most basic sensors, shown in this figure, provide ray-cast detection with configurable noise, detecting time, angle, and range. The displayed sensors are configure to provide debugging detection information. **(B)**: This configuration demonstrates a hatchback with a laser rangefinder navigating around traffic cones. Returned beams are illustrated in red. Beams which do not return data are illustrated in cyan for debugging. **(C)**: Once sensors are placed, the vehicle's navigation algorithm can be tested and examined interactively. The AutonoVi algorithm (Section 3.4) samples potential controls and projects forward in time. Red control paths indicate predicted collisions with the nearby vehicle. The data analysis module allows for exporting sensor data as the vehicle navigates.

3.5.3 Road Network

The road network in AutonoVi-Sim provides the basic connectivity information for the traffic infrastructure to the vehicles in the simulation. At run-time, the network is automatically constructed by connecting roads into a directed graph. The road network provides GPS style routing to vehicles and localization for mapping purposes. Coupled with the road and infrastructure modules, the Road Network also provides information about upcoming traffic and current road conditions. As part of the Road Network, vehicle spawners are provided which generate vehicles and can provide specific destinations for each vehicle. The Road Network can be used to specify per-road initial density as well or to specify a general initial traffic density over the network.

3.5.4 Environment

The environment module allows engineers to specify the specific environmental conditions for a given driving scenario. This currently includes time of day and weather. The system implements varying levels of fog and rain conditions. Basic environmental effects such as road friction reduction are controlled by the environment module. The interaction between weather and specific sensors is implemented by the sensor module.

Figure 3.6: **Simulated Camera Outputs in AutonoVi-Sim:** Camera data captured from different cameras while (**top row**) passing a pedestrian in the road and (**bottom row**) passing a cyclist. **(A)**: A high-resolution camera. **(B)**: Low-resolution camera. **(C)**: A black-and-white camera. **(D)**: A fish-eye 150 degree view camera.

3.5.5 Non-Vehicle Traffic

AutonoVi-Sim implements two non-vehicle traffic participants: pedestrians and cyclists. Pedestrians operate separately from the road network and can be given specific destinations. By default, pedestrians follow safe traffic rules to navigate to their goal. They can also be setup to trigger specific occurrences. For example, as the ego-vehicle nears, a pedestrian can be triggered to walk into the street in front of the vehicle to test its reaction time.

Cyclists operate similarly to vehicles in AutonoVi-Sim. Cyclists are given destinations and route over the road network. Similarly to pedestrians, cyclists can be programmed to trigger erratic behavior under specified conditions. For example, as the ego-vehicle approaches, a cyclist can be triggered to stop in the road, suddenly change direction, or enter the road in an unsafe fashion.

3.5.6 Analysis and Data Capture

AutonoVi-Sim implements a module for logging positions, velocities, and behaviors of the various traffic participants. It also supports logging egocentric data from the vehicle, such as relative positions of nearby entities at varying times during simulation. Camera-based sensors can record out the video data captured during simulation as can LIDAR based sensors Section 3.6.1.2 describes sensors in more detail.

3.6 AutonoVi-Sim: Autonomous Driving Modules

The simulation modules described in Section 3.5 serve as the basis for AutonoVi-Sim. This section describes the two core modules which allow for testing autonomous driving and sensing algorithms under varying conditions, the Drivers and Vehicles modules.

3.6.1 Vehicles

The vehicle in AutonoVi-Sim is represented as a physics-driven entity with specific tire, steering, and sensor parameters. Physics parameters include the base tire coefficient of friction, the mass of the vehicle, engine properties such as gear ratios, and the physical model for the vehicle. Each of these parameters can vary between vehicles and relevant properties such as tire friction or mass can vary at runtime as needed.

3.6.1.1 Control and Dynamics

Vehicle control is provided on three axes: steering, throttle, and brake inputs. The specific inputs are chosen each simulation step by the driver model, described in Section 3.6.2. The vehicle's dynamics are implemented in the NVidia PhysX engine. This allows the simulator to model the vehicle's dynamics and communicate relevant features such as slipping as needed by the driving algorithm.

3.6.1.2 Perception

The perception module provides the interface to gather and store information about the vehicle's surroundings. The basic sensing module in AutonoVi-Sim employs a ray-cast with configurable uncertainty, detection time, classification error rate, and sensor angle / range. This module is sufficient to test scenarios such as late detection or misclassification of pedestrians with minimal intervention. A vehicle can be equipped with multiple sensors with varying angles and fidelity. This allows the vehicle to equip high-fidelity sensors in the longitudinal directions and broader, less accurate sensors in lateral directions. In addition, the perception module specifies how the sensors interact with environmental conditions, including performance impacts and uncertainty caused by weather effects.

The perception module provides interfaces to a generic camera interface and Monte-Carlo scanning ray-casts to simulate various sensor types. These interfaces can be extended to implement LIDAR or camera-based neural network classifiers in simulation. The LIDAR can be configured to change the scanning

range, angle, and resolution. Similarly, the camera resolution, color parameters, and refresh rate can be configured for each camera sensor. Figure 3.5 shows an example of sensor configuration and laser scanner in AutonoVi-Sim. Figure 3.6 demonstrated varying camera setups in AutonoVi-Sim.

3.6.1.3 Modeling Actual Sensors

A core challenge to generating effective perception data is the capacity to replicate the parameters of sensors found on typical autonomous vehicles. Each sensing modality presents unique refresh rates, error rates, and interactions with the vehicle platform itself. The specific data format of a particular camera or LIDAR must be modeled, and the latency expected of the physical sensor must accounted for. We believe AutonoVi-Sim can provide a platform for exploring these issues.

The sensor system in AutonoVi-Sim is modular, and allows for specifying parameters of the sensors. A sensor's refresh rate can be configured independently of the vehicle, and the expected and output data formats are configurable for sensors of different types. In addition, cameras can be configured for color range and camera intrinsics such focal length and distortion, and they can be modeled under varying noise conditions.

The perception systems built on top of imperfect sensor systems can be modeled as well. Misclassification is a typical problem in vision in which an object is assigned an incorrect category. For example, a pedestrian could be misclassified as debris in the road, or a vehicle misidentified as part of the background. By exploiting simulated data from imperfect cameras, we can model classification error and observe and correct the relevant features which cause the vehicle to misidentify and respond inappropriately to nearby entities.

3.6.2 Drivers

Driving decisions in AutonoVi-Sim, including routing and control inputs, are made by driver models. A driver model fuses information from the road network and the vehicle's sensors to make appropriate decisions for the vehicle. The specific update rate of the driver model can be configured as well as what sensors the model supports and prefers. Each model can implement any necessary parameters needed for the specific approach. Aside from the model described in Section 3.4, AutonoVi-Sim implements two other driver models.

The **Basic Driver** is a simple lane-following approach which employs control methods similar to a driver assistance lane-keeping system. This driver model is used to generate passive vehicles traveling along their

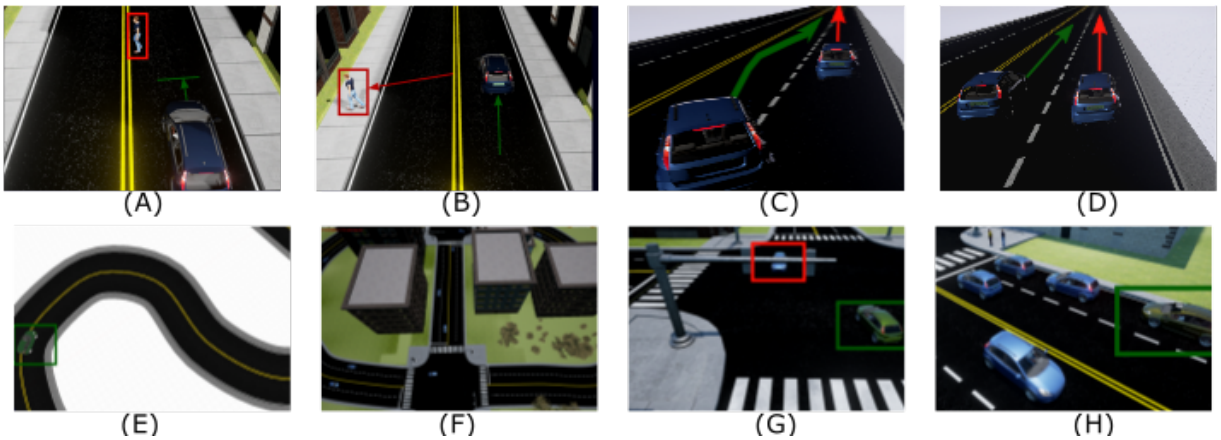


Figure 3.7: **Results: (A) and (B)**: The ego-vehicle is forced to stop as a pedestrian enters the roadway during the **Jaywalking** benchmark due to the proximity costs. Once the pedestrian has moved away, the vehicle resumes its course. **(C) and (D)**: The ego-vehicle approaches a slower moving vehicle from behind. The path and maneuver costs drive the ego-vehicle to plan a lane-change around the slower vehicle. The trajectory of the ego-vehicle is shown in green. **(E)**: The Hatchback ego-vehicle during the **S-Turns** benchmark. The vehicle plans the highest speed it can safely maintain during the tight turns. Each ego-vehicle plans an appropriate speed based on their data-driven vehicle dynamics functions. **(F)**: An overview of the **Simulated City** benchmark. The ego-vehicle navigates amongst typical traffic to a set of randomly assigned destinations. **(G)**: The ego-vehicle (outlined in green) yields to an oncoming vehicle (outlined in red) during the **Simulated City** benchmark. Once the vehicle clears the intersection, the ego-vehicle proceeds with a left turn. **(H)**: The ego-vehicle (outlined in green) stops in traffic waiting for a stoplight to change during the **Simulated City** benchmark.

destinations without aberrant or egocentric behaviors. These vehicles are capable of lane-changes and turns, but follow simple rules for these maneuvers and rely on perfect sensing models to accomplish them.

At each planning step, the Basic Driver projects the positions of nearby entities into the future by a pre-determined time threshold. It leverages these projections to exclude choices of control inputs which would lead to a collision with its neighbors. It then chooses the closest speed to its target speed that avoids potential collisions.

Finally, the simulator implements a manual driving mode, which can be activated from any autonomous driver. Manual mode allows an engineer to drive the vehicle using a keyboard, game-pad, or steering wheel and pedal combination. Barber and Best (2017) demonstrate using this manual operation to test vehicle signaling and connected vehicle operation. It can also be used to collect data for neural-network methods, as shown in figure 3.6.

3.7 Experimental Evaluation

In this section, we detail the evaluation scenarios for our navigation algorithm within Autonovi-Sim. Each scenario is chosen to test different aspects of the algorithm including response time, safety, and handling different traffic situations. Figures 3.7 and 3.8 demonstrate several simulated scenarios in AutonoVi-Sim.

3.7.1 Ego-Vehicles

To demonstrate the generality of our approach, we tested each experimental scenario on each of three vehicles. Vehicle 1, the hatchback, has a mass of 1365 kg, a length of 3.8m, and a maximum steering angle of $60°$. Vehicle 2, the sports car, has a mass of 1750 kg, a length of 4.6m, and a maximum steering angle of $63°$. Vehicle 3, the SUV, has a mass of 1866 kg, a length of 4.8m, and a maximum steering angle of $55°$.

3.7.2 Benchmarks

We conducted a series of simulations with each vehicle representing a variety of the challenging traffic scenarios faced while navigating city roads and highways.

Passing a bicycle: The ego-vehicle must pass a bicycle on a four-lane road. The vehicle should maintain a safe distance from the bicycle, changing lanes if possible to avoid the cyclist. We perform the evaluation twice, once featuring a vehicle in the adjacent lane preventing the vehicle from moving to avoid the cyclist without first adjusting its speed.

Jaywalking Pedestrian: The vehicle must react quickly to safely decelerate or stop to avoid a pedestrian stepping into the road in front of the vehicle.

Sudden Stop at High Speed: The vehicle must execute an emergency stop on a highway at high speeds when the vehicle in front of it stops suddenly. We evaluate this scenario in two conditions. First, we evaluate performance with no other traffic aside from the ego-vehicle and stopping vehicle. In this condition, swerving can be performed simply. Secondly, we evaluate this scenario with surrounding traffic, complicating any swerving maneuvers as the vehicle must account for nearby traffic.

High-Density Traffic Approaching a Turn: The ego-vehicle approaches a stoplight at which it must execute a turn, but the ego-vehicle's lane is congested by slow traffic. To make optimal progress, the ego-vehicle should execute a lane change to the adjoining lane and return to the correct lane with sufficient time to execute the turn.

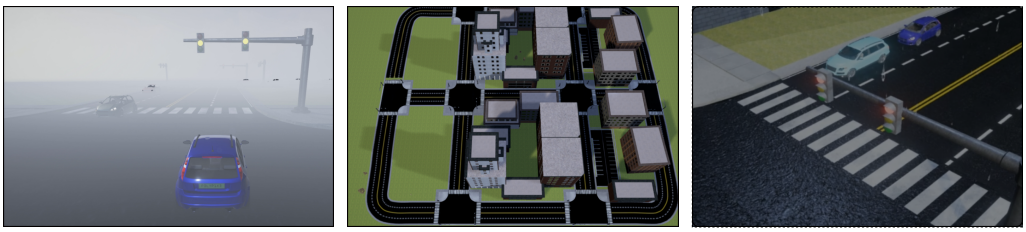


Figure 3.8: **Simulated scenarios and conditions in Autonovi-Sim: (A)**: Heavy fog obstructs the view of the vehicle. **(B)**: A simulated city modelled in AutonoVi-Sim. Closed circuit road networks allow engineers to test driving algorithms over long timescales by assigning new navigation goals periodically. **(C)**: Vehicles pass through a slick intersection during rainy conditions.

Car Suddenly entering Roadway: The ego-vehicle travels along a straight road at constant speed when a vehicle suddenly enters the roadway, blocking the ego-vehicle's path. The ego-vehicle must decelerate and swerve to avoid colliding with the blocking vehicle. We demonstrate this scenario with the ego-vehicle travelling at 10, 30, and 50 mph and with the blocking vehicle obstructing either the right lane or both lanes in the ego-vehicle's travel direction.

S-turns: We demonstrate the ego-vehicle navigating a set of tight alternating turns, or S turns. Each ego-vehicle computes an appropriate safe speed depending on the specific kinematic and dynamic limits of the vehicle.

Simulated City: We demonstrate the ego-vehicle navigating to several key points in a small simulated city. The vehicle must execute lane changes to perform various turns as it obeys traffic laws and navigates to its goals. The vehicle encounters bicycles, pedestrians, and other vehicles as it navigates to its waypoints.

3.7.3 Benchmark Results

We evaluated our navigation algorithm in these simulated scenarios. The computational cost of the algorithm scales linearly in the number of neighbors. The algorithm can avoid tens of vehicles at interactive rates. As expected, the sports-car and the hatchback were able to maintain their preferred speeds more effectively in turns, whereas our SUV was forced to reduce speed. Each of the vehicles was able to pass other vehicles, pedestrians, and bicycles safely. We did not observe the ego-vehicle colliding with simulated vehicles in traffic.

Figure 3.7 details some interesting behaviors we observed while testing our navigation algorithm. As expected, the ego-vehicle utilizes lane-changes to pass slower vehicles when no traffic prevents it. In traffic, the ego-vehicle slows down until it is safe to pass in the adjoining lane. When interacting with pedestrians,

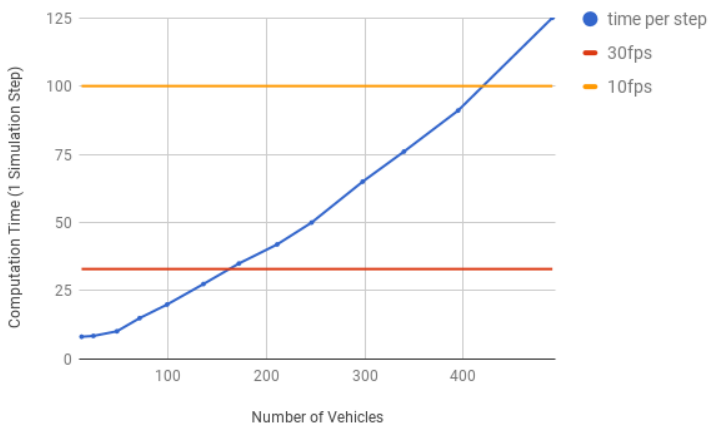


Figure 3.9: **AutonoVi-Sim Performance Graph**: We conducted repeated simulations on a traffic network, increasing the number of vehicles in each scenario. This graph details the computation time for each simulation step as a function of the number of vehicles simulated. The limit of 30 frames per second and 10 frames per second are shown for reference. In this scenario, each vehicle is equipped with two basic ray-cast sensors with perfect accuracy. We find that the computation time scales linearly in the number of vehicles simulated, with the ability to simulate 160 vehicles at 30 frames per second and up to 420 vehicles at 10 frames per second.

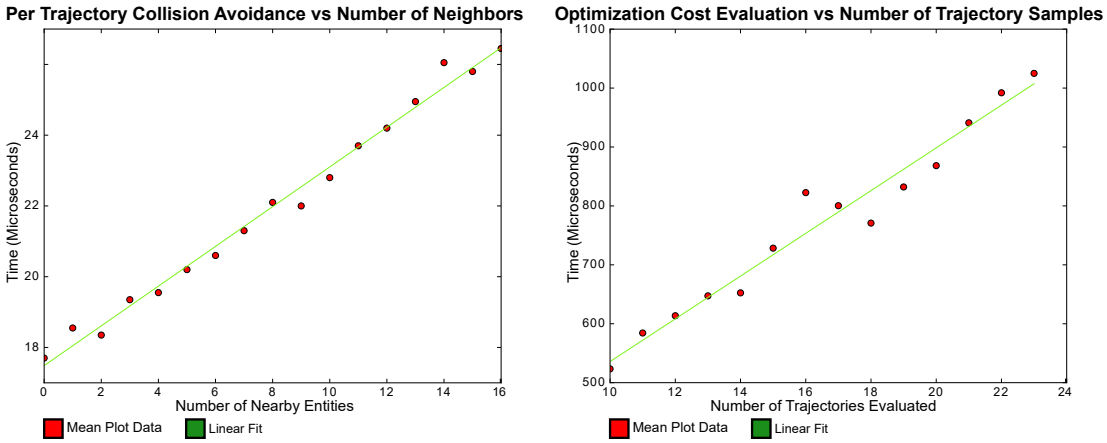
the high proximity cost discourages the vehicle from changing lanes as the pedestrian passes, and the vehicle instead waits until the pedestrian has moved considerably.

3.7.4 Performance Timing

3.7.4.1 AutonoVi Algorithm

We collected data from a simulation designed to gradually increase the density of other vehicles encountered by a vehicle navigating with AutonoVi. Figure 3.10 (A) demonstrates the relationship between collision avoidance cost for a specific trajectory sample and the number of nearby vehicles and entities. We observe the cost of collision avoidance grows linearly in the number of neighbors.

Figure 3.10 (B) demonstrates the relationship between the number of trajectories sampled and the computational expense of the AutonoVi optimization evaluation. The observed relationship is linear with greater variance than that of collision avoidance computation. The overall computation time for a typical navigation update including guiding path computation, control-obstacle sampling, collision-avoidance and cost evaluation is on the order of milliseconds, typically between 1 and 2 milliseconds. This suggests that the cost of the algorithm is dominated by the optimization time.



(A) (B)

Figure 3.10: **AutonoVi Algorithm Timing Results: (A) Collision Avoidance Timing**: We detail the relationship between the cost of collision checking for a trajectory and the number of nearby entities considered. We observe a linear relationship between collision checking and number of neighbors. **(B) Cost Function Evaluation Timing**: The computational cost of computing the optimal trajectory for the vehicle varies linearly with the number of collision-free trajectories evaluated.

3.7.4.2 AutonoVi-Sim

We conducted a series of repeated traffic trials to determine the expected performance of AutonoVi-Sim. We find that the computational costs scale approximately linearly with the number of vehicles simulated. We have successfully simulated over 400 vehicles simultaneously at high-densities at interactive simulation rates. Figure 3.9 shows the results of our performance tests.

3.7.5 Generating Training Data

AutonoVi-Sim can be used to generate labeled training data for typical as well as atypical and dangerous situations. We can simulate many scenarios involving pedestrians, cyclists, and other vehicles, such as jaywalking or passing in traffic. The vehicle can be driven automatically using the driver models, or manually by an engineer. Camera, ray-cast (LIDAR approximation), relative position, detection, and control data are exported from each trial of the simulation. The controls of the vehicle combined with local conditions can be used for reinforcement learning in the autonomous driving case or imitation learning in the manual case. These scenarios can be repeatedly run under varying lighting and weather conditions; different surroundings, i.e. buildings, trees, etc; and with different pedestrians, cyclists, and vehicle shapes and sizes. Figure 3.6 demonstrates a vehicle with 4 co-located cameras of varying properties capturing interactions with a cyclist and pedestrian.

3.8 Conclusion and Limitations

In this chapter, we have presented AutonoVi, a navigation algorithm for autonomous vehicles. Our approach uses a data-driven vehicle dynamics model and optimization-based maneuver planning to compute safe, collision-free trajectories with dynamic lane changes under typical traffic conditions. We have demonstrated our algorithm on a varied set of vehicles under varying dense and sparse traffic conditions with pedestrians and cyclists. We have also demonstrated that our vehicles follow traffic laws, and utilize both braking and steering simultaneously when avoiding collisions. We highlight many benefits over prior methods in our simulations.

Our approach has some limitations. Though our data-driven dynamics functions A, Φ, and S could generalize to arbitrary complexity of underlying dynamics, our current approach requires computing new vehicle dynamics functions for different values of μ. In future work, we can address this by learning a transfer function between various road frictions to produce more general data-driven vehicle dynamics functions. In addition, we have assumed perfect sensing, which limits the applicability of the approach. In future work, we will expand our algorithm to account for sensing errors and uncertainties. These could incorporate predictive behavior models to overcome imperfect state estimations for neighboring entities (Galceran et al., 2015a; Sun et al., 2014). The control obstacle approach could then anticipate levels of reciprocity from predictable vehicles. In addition, the use of circular arcs may not be appropriate for vehicles with substantially different geometries, such as trucks pulling trailers. We would also like to incorporate real-world driving patterns and cultural norms to improve our navigation algorithm, as well as techniques to predict pedestrian motion from sensors for safer navigation (Bera et al., 2017).

In addition, we have detailed AutonoVi-Sim, a platform for autonomous vehicle simulation with the capacity to represent various vehicles, sensor configurations, and traffic conditions. We have demonstrated AutonoVi-Sim's applicability to a number of challenging autonomous-driving situations and detailed the ways in which AutonoVi-Sim can be used to generate data for training autonomous-driving approaches. AutonoVi-Sim is a modular, extensible framework. While many modules currently represent preliminary implementations of advanced functionality, the extensible nature of the framework provides the basis for progress in the various disciplines which define autonomous driving.

Our work is in active development and still faces a number of limitations. AutonoVi-Sim contains basic implementations of the various modules such as sensors for perception, a physics engine to simulate dynamics,

etc. However, each of these modules can be extended to more accurately reflect real world conditions. For example, our sensor models currently do not model noise or uncertainty in the exported data. The basic driver behavior is also quite limited; in the future we intend to model additional driver models to provide more rich behaviors for other vehicles.

AutonoVi-Sim currently lacks calibration information to replicate specific sensors and sensor configurations. In the future we hope to model specific sensing packages and algorithms to test specific real-world configurations. In addition, it will be beneficial to explore the transfer between algorithms trained on AutonoVi-Sim and actual test vehicles. Our current driver models are limited to hierarchical, rule-based driving approaches. In future work, we intend to include exploration of end-to-end approaches, which can be represented by a novel Driver model. The current simulator supports a few hundreds of vehicles. By combining our simulator with macroscopic or hybrid traffic simulation approaches, we seek to increase the size of supported traffic conditions.

CHAPTER 4: Planning Plausible Verbal Interactions Between Virtual Agents and Avatars in Virtual Environments

4.1 Introduction

There is great recent interest in generating immersive social experiences. Increasingly, games, training, and entertainment seek to provide a user with the experience of embodying a digital *avatar* and sharing a virtual space with other user-controlled avatars as well as computer-controlled characters, or *agents*. Such multi-agent multi-avatar applications range from immersive games, social virtual-reality (VR) hangouts, training simulations (Rickel and Johnson, 1999; Hill Jr et al., 2003), treating social-phobias (Pertaub et al., 2002) or visiting virtual spaces such as museums or landmarks.

The plausibility and effectiveness of multi-avatar simulations can be improved by the presence of interactive human-like virtual agents (Garau et al., 2005). However, virtual agents that do not interact in plausible ways can reduce the sense of presence in virtual environments (Slater et al., 2009; Bailenson et al., 2005). Moreover, the context of the simulation may necessitate agents that have their own independent goals and are not purely focused on the co-present avatars. The virtual world should feel like a place the avatar is visiting, as opposed to one constructed purely for the avatar. In such cases, agents must be capable of engaging in meaningful interactions with avatars and other agents, either proactively or in response to the actions of others. These interactions may include both verbal as well as non-verbal means of communication including movement and navigation, gesturing, gazing etc. Recent studies have highlighted the critical role of verbal communication and its significant impact on the perceived naturalness of user-agent interactions, and the overall effectiveness of the application (Novielli et al., 2010; Nass et al., 1994).

Most prior work in enabling interactions between avatars and agents is limited to embodied conversational agents (ECA), wherein an anthropomorphic virtual agent demonstrates human-like face-to-face communication (Cassell et al., 2000). However, ECA is generally restricted to single agent-avatar pairwise interactions and is often avatar-centric. The agent participates in interaction with the aim of assisting the avatar in achieving a goal, or foiling the avatar, but does not plan its own intentions outside the context of the

avatar-agent interaction. There is also prior work in multi-agent navigation that has explored communication behaviors (Kullu et al., 2017; Pelechano et al., 2005). However, these methods rely on message-passing or implicit communication which preclude verbal interaction with user-controlled avatars. Overall, simulating plausible verbal interactions in shared multi-avatar multi-agent environments remains a challenge.

There are several core challenges in simulating the behaviors of virtual agents in such multi-avatar multi-agent environments. First, agents must be capable of independently planning egocentric behaviors in potentially uncertain conditions. Much like the real world, an agent may possess an imperfect understanding of the world and must be capable of proactively communicating with other entities to derive knowledge such that it can accomplish its goal. This may include seeking out information from nearby agents or avatars.

Second, agents must be capable of communicating with avatars and other agents in unstructured conditions. In effect, agents must be able to interpret language, generate meaningful responses and exchange information, agnostic of whether the other entity is a user-controlled avatar or another virtual agent. This enables multi-agent multi-avatar environments in which the agents are communicating in ways which are interpretable to the receiver and present plausible interactions to nearby observers.

Third, agents must be capable of generating plausible behaviors, including asking and answering questions, based on their interpretable understanding of the virtual world. In effect, agents should be able to absorb information through communication and behave appropriately based on the new information. Creating an appropriate knowledge representation enables the agents to interpret information communicated by other agents and query their own understand to generate responses for questions from agents and avatars.

Main Results: In this chapter, we seek to address the problem of simulating many virtual agents that can effectively plan individual actions, interact, and communicate with avatars and other agents using natural language. To this end, we present *Sense-Plan-Ask (SPA)*, an interactive approach to enable virtual agents to accomplish their individual goals with uncertain information in complex multi-agent multi-avatar environments. The SPA approach consists of following novel contributions:

- **Propositional-planning with Automatic Uncertainty Resolution**: We present a least-commitment-based planning approach (Russell and Norvig, 2009) to generate agent action plans with uncertain information. Agents automatically generate uncertainty resolution actions that may include navigational actions to explore the environment, or asking questions. Moreover, agents re-plan based on new information.

- **Multi-agent Natural language Interaction**: We present a natural-language communication approach that can parse utterances received from other agents and avatars, generate natural-language responses as well as construct queries and new beliefs based on propositional logic.

- **Proactive Agents**: Our approach, SPA, allows agents not only to react to avatars and agents, but to proactively seek out interaction and engagement. The agents learn from interaction and respond accordingly, generating diverse and comprehensive simulations.

- **Interactive Simulation of Tens of Agents and Avatars**: Our approach allows tens of agents and avatars to interact in immersive virtual environments and can be combined with passive agents to simulate large scenes.

- **User Evaluation**: We present the results of a user study which demonstrates our method's advantages over prior approaches. Compared to methods which do not enable natural-language interaction between agents and avatars, participants showed significant preference for our method in terms of the plausibility of the scenarios and quality of agent-avatar interactions.

We demonstrate our approach's ability to generate plausible behaviors for tens of virtual agents on a set of simulated benchmarks including typical social settings such as visiting a museum and emergency evacuations. The rest of the chapter is organized as follows: In Section 4.2, we detail relevant related work in multi-agent systems and task planning. We give an overview of SPA in Section 4.3. In Section 4.4, we detail our propositional-planning framework which enables planning under uncertainty via interaction. Section 4.5 details our natural-language processing and generation approach. We describe our simulation benchmarks, offer performance results, and detail the results of a user evaluation of our method in Section 4.6.

4.2 Related Work

In this section, we give an overview of relevant work in action-planning, multi-agent simulation, and verbal communication.

4.2.1 Action Planning

Action planning, sometimes referred to as classical planning, task planning, or logical planning in the literature (Russell and Norvig, 2009; Ghallab et al., 2004), deals with constructing a set of actions taken by an

intelligent agent to solve a problem or achieve a goal. Fikes and Nilsson (1971) introduced STRIPS, a generic problem solving framework which has formed the basis of many classic approaches. STRIPS introduces a domain description formalism which allows many distinct domains to be solvable with a common approach. Many extensions to STRIPS have been proposed including ADL, Action Description Language (Pednault, 1989), and PDDL, Planning Domain Definition Language (McDermott et al., 1998; Fox and Long, 2003), which form the basis of many modern approaches. Our planning formalism described in Section 4.4 is a subset of ADL. Additional details on action planning can be found in Section 4.4.

Leveraging the logical description of the problem domain, the agents must internalize some set of knowledge about the environment. Many formalisms have been proposed to represent the agent's beliefs and understanding of the world. These include ACT-R (Anderson et al., 1997), Belief-Desire-Intention (BDI) (Rao and Georgeff, 1995), and Agent-Oriented Programming (Shoham, 1993) among others. A common characteristic of these agent frameworks is adherence to a logical-formalism for representing not only the state of the environment, but the agent's knowledge and beliefs about that environment. Our algorithm leverages the theory of BDI agents, but would be compatible with the Agent-Oriented Programming approach as well.

Recent work in task planning has addressed planning under uncertainty through continual planning (Brenner and Nebel, 2009), or by encoding sensing actions for the agents (Petrick and Bacchus, 2004; Bonet and Geffner, 2000). The latter approach encodes a specific "sensing" action for each kind of information the agent may need. Our approach seeks to minimize explicit sensing actions, instead employing natural-language interactions and generic exploration to resolve uncertainty.

4.2.2 Human and Single Agent Interactions

Interactions between humans and automated agents can be subdivided into human-robot interactions (HRI) and human interactions with a digital or virtual agent, also called Embodied Conversational Agents (ECA).

In the domain of HRI, robots typically plan dialog to clarify a human operator's intentions in a collaborative context (Doshi and Roy, 2008; Clodic et al., 2007; Bisk et al., 2016). The robot does not maintain independent goals, nor does it maintain a non operator-centric representation of the domain. Its understanding is purely grounded in the operator and the robot's perception of the operator's world model. The robot works as a subservient collaborator, typically limited to interaction with a single human operator. Recent

work has demonstrated the ability for a robot to request assistance in a specific collaborative-task using inverse-semantics (Tellex et al., 2014). Our planning framework is complimentary to the language generation model demonstrated, and could be used with such a system for embodied conversation on a physical robot. Some work has incorporated models of sensor uncertainty (Tenorth and Beetz, 2017), but modeling unknown information remains a challenge across domains. Our approach allows agents to act on their own goals and employ natural-language interactions with multiple avatars or agents by specifying a range of domain information types without fully specifying the agent's knowledge.

Prior work in Embodied Conversation Agents (ECA) has demonstrated the ability of a single agent to communicate with a human avatar using complex but well-structured dialogue (Graesser et al., 2014). Kopp et al. (2005) demonstrated an embodied agent acting as a museum guide. Rickel and Johnson (1999) demonstrated positive effects on team training when using virtual agents in a mixed agent-avatar environment. Hall et al. (2005) demonstrated positive impacts on empathic engagement using structured dialog and conversation agents. However, in all of these cases, the agent's behavior is fundamentally user-centric. Rather than behaving as an independent entity, the agent's plan revolves around the avatar.

Prior work found positive impacts from the use of proactive agents in a personal assistant capacity (Liao et al., 2016; Kim et al., 2006). In each case the agent's goal was to serve the user in a productivity or learning task. The agent maintains a model of the user's mental state, and utilizes that model to determine appropriate moments for interruption / proactive interaction. The agents were not pursuing egocentric goals in an independent fashion, and were additionally limited to single agent environments. Our approach seeks to enable many agents to pursue individual goals leveraging interaction to reduce uncertainty and accomplish their goals more effectively, and is not avatar-centric.

Some recent work has demonstrated capabilities in learning from natural language interaction. Wang et al. (2016) demonstrated an agent learning concepts from natural language interaction with a single human. Thomason et al. (2015) demonstrated learning from natural language by generating a large example database and generalizing the concepts. However, these methods limit the interactions to a single user and do not generalize to unstructured interactions.

4.2.3 Multi-agent Planning

Action planning for interactive multi-agent systems has typically been limited to locomotion-based actions, and comprises of choosing an appropriate goal position for each agent and computing its trajectory. A

large body of research exists on planning paths to an agent's goal (Snook, 2000; van Toll et al., 2012; Latombe, 1991) and local-avoidance behaviors (Van den Berg et al., 2011; Karamouzas et al., 2014; Schadschneider, 2002; Reynolds, 1987) for agents. Our work is complimentary to these approaches and our agents can use any of these methods for path planning. However, many of these widely used approaches rely on relatively rigid finite-state machines to generate goals for the agents (Curtis et al., 2016; Ulicny and Thalmann, 2002).

Some alternate approaches have also been proposed to generate more diverse, contextual actions. Paris and Donikian (2009) proposed an affordance-based behavior-tree approach to generate non-locomotion actions including accessing machines and purchasing items. Shao and Terzopoulos (2005) described a categorical, hierarchical approach to generate diverse behaviors in a group of virtual pedestrians. In each case, the behaviors of the agents are pre-encoded and activated when specific conditions are met. Instead, our approach allows for dynamically creating action plans for agents without a rigid or pre-encoded structure. Our agents generate and execute plans as needed to satisfy their goals. In addition, our agents' goals are diverse. Agents can plan simple goal positions, or complex goals, i.e. interacting with a specific agent or acquiring some item from the environment.

4.2.4 Communication in Multi-agent Systems

Some prior approaches have introduced communication capabilities in multi-agent systems. Kullu et al. (2017) demonstrated a message packet approach based on the FIPA message structure. Huang et al. (2013) used a packet-based approach to model auditory cues in an environment such as banging. Chao and Li (2010) demonstrated a signal accumulation approach for transfer of a social contagion such as rioting. Pelechano et al. (2005) demonstrated implicit communication through the sharing of map information between agents with diverse social roles. In each of these prior approaches, agents communicate using a strict message structure, or implicitly through the sharing of data and thus preclude the use of verbal communication.

Recently, Mordatch and Abbeel (2017) demonstrated a multi-agent system in which agents develop a symbolic language for agent cooperation in an emergent fashion using reinforcement learning. However, the language is not directly interpretable and does not allow for communication with avatars. Sun et al. (2012) generated animated conversation for agents in a simulated crowd. These conversations were generated post-simulation as an animation feature. By contrast, our approach allows the agents to communicate with other agents and avatars using plausible verbal communication in real time. The most relevant recent work is that of (Brenner and Kruijff-Korbayová, 2008) which enables multiple agents to collaborate with a simulated

user (i.e. not an actively controlled avatar) in a shared-task. However, this method generates language as a post-process and does not rely directly on natural-language processing, which limits its applicability to interactive simulations with avatars.

4.3 Sense-Plan-Ask Overview

Our approach, Sense-Plan-Ask, couples propositional planning with natural-language processing to enable many agents to interact with other agents and avatars in shared environments. This section introduces relevant terms and notation used throughout the chapter, and provides an overview of our approach.

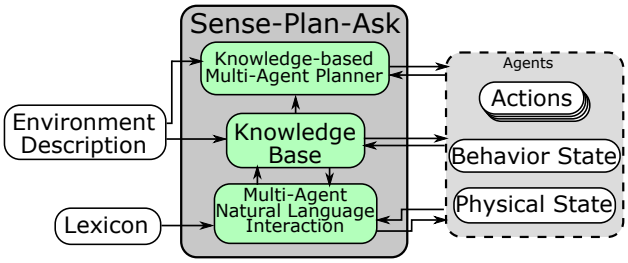


Figure 4.1: **Algorithmic Pipeline**: We present a novel interactive approach called Sense-Plan-Ask (SPA) which enables agents to interact with other agents and user-controlled avatars using natural language communication. Each agent seeks to accomplish its individual goals and uses SPA to compute explainable action plans based on potentially uncertain information. The agent can engage in natural language conversations with nearby agents and avatars to resolve uncertainty in its plans. The communication model increases the plausibility of the simulation, and also improves the user's experience in immersive virtual environments.

4.3.1 Agent Model

Each agent in the virtual environment can be defined by its *physical state* and its *behavioral state*. The physical state comprises of physical properties such as position, velocity and its current action. The *behavior state* regulates its decision making, including its current knowledge and the set of available actions.

4.3.1.1 Physical State

For the purpose of behavior and movement planning, we treat each agent i as a bounded disc in a 2D plane with scalar radius r_i. We denote the agent's position and velocity at time t by $\vec{p}_i^t$, and $\vec{v}_i^t$ respectively. Furthermore, we denote the current action being executed by the agent as $a_i^t | a_i^t \in \mathcal{A}_i$. $\mathcal{A}_i$ is the set of all actions available to the agent and may include movement or navigation as well as non-movement actions, such as speaking, or interacting with the environment, i.e. affecting change to the simulation domain. We

assume a path planner is available for locomotion and is used to compute the path and avoid collisions with other agents, avatars, and obstacles in the environment. Collectively, the position, velocity, and current action describe the agent's time-varying physical state $\{\vec{p}_i^t, \vec{v}_i^t, a_i^t\}$.

4.3.1.2 Behavior State

We conceptualize our agents as Belief Desire Intention (BDI) agents (Rao and Georgeff, 1995). Each agent maintains an independent understanding of the world represented by a set $\mathcal{B}$ of facts called *beliefs*. We use the term belief to reflect the fact that the agent's knowledge may be wrong. Moreover, each agent is given a set $\mathcal{D}$ of high-level *desire*s, or goals, to achieve in the simulation. Each agent plans a set of intermediate actions called *intentions* to accomplish these desires.

We encode the agent's beliefs, desires, and intentions in a first-order propositional planning language and store the agent's knowledge in relational database. Our formalism is a subset of ADL (Pednault, 1989) with the exception that we do not allow disjunction in desires, and limit desire specification to grounded literals. First-order propositional planning provides sufficient representational power for our interactive benchmarks (see Section 4.6) and fits well with the types of questions we seek to support in our interaction. We define a set Σ comprising of *entities* that are symbolic constants through the simulation. These may include agents, items, physical locations, etc. We apply a type to each entity and a set of attributes associated with the type.

We describe predicates as first-order formulae over Σ. We allow constraints on the type of arguments in the predicate schema to reduce the search space, and we explicitly allow negated predicates in state specification. Furthermore, we categorize predicates in two categories. *Knowledge predicates* describe relationships or facts the agent might know. *Fluent predicates* describe transitive properties the agent may hold, i.e. being at a specific location (Russell and Norvig, 2009). A belief-state for agent i at time t, denoted $\mathcal{B}_i^t$, consists of the set of all beliefs known to be true or false to the agent. For compactness, we assume Σ is known to all agents and is implicitly included in $\mathcal{B}$. Consider for example, a problem-solving domain consisting of a series of keys and locks, the following would be a valid state specification:

$$\begin{aligned}\mathcal{B}_i^t = \{&Have(Key_1), \neg Have(Key_2), Opens(Key_1, Safe_1),\\ &\neg Locked(Safe_2)\}\end{aligned}$$

We do not assume a complete state specification. That is, B_i^t contains all *known* information. Missing predicates from the state specification are considered unknown as opposed to false. In the example above, agent i knows that Key_1 opens $Safe_1$, but does not know of anything which opens $Safe_2$. In Section 4.4, we detail how our planning approach allows agents to plan uncertainty resolution and in Section 4.5 we describe how they use verbal communication to resolve uncertainty.

We define a set of operators, or *actions* $\mathcal{A}$, over beliefs of the form $O < parameters, conditions, effects >$. Parameters describe elements in Σ passed to the action, subject to constraints on the type of the element. Conditions are predicates which must hold in $\mathcal{B}_i^t$ for the action to be applicable. Effects are predicates added to $\mathcal{B}_i^t$ upon application of the action. Should a predicate in $\mathcal{B}_i^t$ be contradicted by new information, it is removed. This corresponds to the agent updating its beliefs. Continuing from the example above, the following would be a valid action schema:

$$
\begin{aligned}
Open <& \{key : X, safe : Y\}, \\
& \{Locked(Y), Opens(X, Y), Have(X)\}, \\
& \{\neg Locked(Y)\} >
\end{aligned}
$$

The desires of the agent are a time-varying set $\mathcal{D}_i^t$ of beliefs which must be achieved by actions over the agent's initial state, $\mathcal{B}_i^0$. The agent's behavior state for time t is compactly described as $\{\mathcal{B}_i^t, \mathcal{D}_i^t, \mathcal{A}_i^t\}$.

4.3.2 Sense-Plan-Ask (SPA) Algorithm

Our proposed approach, Sense-Plan-Ask, generates plausible interactions between virtual agents by coupling a novel propositional planner with a natural-language processing framework. SPA is an agent-based simulation algorithm which enables the agents to communicate, plan, and interact with other agents and avatars. Figure 4.1 details our algorithmic pipeline.

Sense: We conceptualize the simulator as a discrete in time and continuous in space. The simulation updates at a fixed rate, Δt. Each update, the agents in the simulator "sense" their surroundings. They observe relevant predicates associated with nearby entities within a range, and "hear" utterances produced by agents and avatars in their vicinity. Based on the observations, the agents update their internal knowledge representation and react accordingly.

Plan: Each agent plans a series of actions to accomplish its goals. Section 4.4 describes how these plans are constructed based on uncertain information, and how the planner generates disambiguation actions such as asking questions of nearby agents or avatars to resolve the uncertainty. The planner allows the agents to process new information and re-plan rapidly, as is detailed in Section 4.6.

Ask: The natural-language approach combines shallow semantic parsing with template-based generation to produce intelligible verbal questions and answers between agents and avatars. Section 4.5 details how agents learn new information from their interactions and use that information to resolve uncertainties in their plans. In addition, agents can interact with avatars and other agents verbally by engaging in question and answer-based dialogue.

Each simulation environment is configured from a domain specification file, which contains the set of entities, predicate schema, action schema, and initial knowledge for each agent in the environment. This specification also gives initial conditions of agents and other objects in the environment. This domain description is paired with an English lexicon to automatically generate training data for a shallow-semantic parser as described in Section 4.5. This domain specification is also used to construct a queryable knowledge-base, encoded in a SQL database, for each agent which allows the agents to recall and respond to changes in the environment.

4.4 Knowledge-based Multi-Agent Planning with Incomplete Information

Figure 4.2: **Two-stage Action Planning with Incomplete Information**: Each agent is given a set of desires to achieve during simulation. We propose a two-stage planner which generates action plans despite uncertainties in the agent's knowledge. The first stage generates a plan template and a set of candidates for each argument in the plan. The second stage generates a set of candidate bindings. The algorithm selects the plan with the least uncertainty and generates an action plan from the bindings. If any uncertain information is present, a disambiguation action is created, yielding the final action plan which may include asking questions, or exploring the environment.

Our proposed planning approach enables each agent to plan actions to achieve its desires based on its current, potentially incomplete set of beliefs. Our algorithm tracks the uncertainty in the agent's plan and determines new actions such as verbal interaction or exploration of the environment to resolve the uncertainty. Moreover, it enables agents to update their beliefs in real-time and re-plan based on their new set of beliefs.

4.4.1 Two-stage Action Planning with Incomplete Information

In many propositional planning languages, predicates absent in the state description are considered false, and uncertainty is prohibited (Ghallab et al., 2004). Typical first-order propositional planning approaches may allow incomplete specifications, but require an explicit action for determining the truth state of each individual pre-condition of an action. This can lead to a combinatorial explosion in the number of actions and subsequently the planning time of the approach. (Russell and Norvig, 2009). Our approach addresses these limitations and allows for efficient planning under uncertainty by leveraging a two-stage planning algorithm.

Given agent i's desire set, $\mathcal{D}_i$, a set of actions must be constructed to satisfy all of the desires. We apply a least-commitment, backward state-space search approach (Russell and Norvig, 2009). Each planning step may comprise of multiple iterations. In each iteration j, the agent chooses an action which satisfies the first desire $d_0 \in \mathcal{D}_i^j$. A new desire set $\mathcal{D}_i^{j+1}$ is created consisting of the remaining unsatisfied desires and any unsatisfied pre-conditions of the chosen action.

As described in Section 4.3, we differentiate between knowledge and fluent predicates when computing $\mathcal{D}_i^{j+1}$. Knowledge predicates present in $\mathcal{B}_i$ are considered satisfied, and any absent from $\mathcal{B}_i$ are added to an uncertainty set, U_i, as opposed to the new desire set, $\mathcal{D}_i^{j+1}$. Any other arguments of the action aside from those needed to satisfy the desire are left unbound, i.e. they are not assigned any entity. For each unbound argument k, a candidate set of all entities which may satisfy k is constructed, termed C_k. Candidates are chosen based on two criteria: they must satisfy any type and property constraints of the predicate, and, given the candidate binding, all pre-conditions of the action must be either held in B_i (true) or absent from B_i (unknown).

Our planning algorithm continues in this fashion, achieving desires until either the desire set is empty, $D_i^j = \emptyset$, or the desire set represents a subset of the agent's current belief state, $D_i^j \subset B_i$. The result is a plan-template, $\mathcal{P} = \{A_i^{t_0}, ..., A_i^{t_n}\}$, i.e. a sequence of actions which satisfy the agent's desires, a set of candidates for all unbound arguments $\mathcal{C} = \{C_k | k \in K\}$, and a set of unknown predicates, U_i, associated with the plan. The action order is inverted for execution, consistent with the backward state-space search.

The second stage of our planning algorithm finds appropriate bindings for the candidates in $\mathcal{C}$. We first construct a set of grounded plans such that all candidates are given a specific binding. These plans are sorted according to the number of predicates from U_i found in B_i given the candidate bindings. In effect, the agent prefers plans with the least uncertainty. For each predicate in U_i not found in B_i, an uncertainty resolution

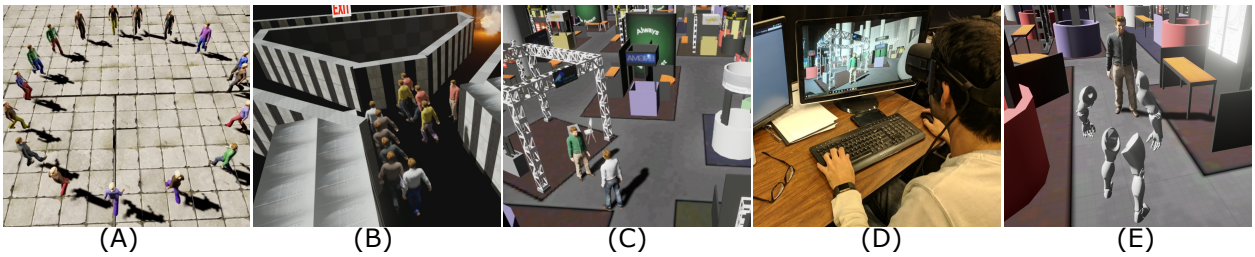


Figure 4.3: **Simulation Benchmarks**: We demonstrate our algorithm's performance in a set of simulated environments. **(A)** Agents in the anti-podal circle scenario search for items while requesting information from one another. **(B)** Agents evacuate a building during an emergency. The right-most agent acts as a first-responder, warning the other agents to avoid the fire-blocked hallway. **(C)** Agents explore a densely-packed tradeshow scene, visiting booths and exhibits. The agents request location and booth information from one-another to resolve uncertainty in their plans and reach their desired goals more effectively. **(D)**: A user explores the tradeshow with a first-person view in immersive settings. Our approach allows virtual agents to interact with both agents and avatars simultaneously. **(E)**: The user's avatar (shown from a third-person view) interacts with a virtual agent face-to-face in the tradeshow.

action is inserted into the plan prior to the first occurrence of the predicate. Uncertainty resolution actions include exploring the environment or asking questions. The final plan now consists of the original actions in $\mathcal{P}$ and an uncertainty resolution for each unknown predicate. Figure 4.2 details our planning approach.

4.4.2 Plan Execution and Re-planning

Each action described in the problem domain is mapped to a simulation controller in the agent. The controller is responsible for executing the action in the simulation environment. These controllers include uttering, moving to a location, interacting with the environment, waiting a specific amount of time, etc.

If a controller fails to accomplish the specified action for the agent, or the agent is unable to acquire the the information needed for an uncertain belief, the plan binding fails. If other bindings are available for all entities in $\mathcal{C}$, the next binding in order of uncertainty is chosen. Each time an agent acquires new information, the uncertainty of remaining bindings is updated and the set of candidates adjusted to prevent repetitive questions.

If no suitable bindings are available, the planner discards the plan template and back-tracks to the prior branch in the backward state-space search. If no additional branches can be chosen, the plan is discarded and the planner fails. The agent waits a pre-defined amount of time before restarting the planning procedure. These failures are often caused by a failure to acquire information, and are likely to succeed on subsequent planning attempts as other agents and avatars move near the agent.

Waiting is the strategy used in our experiments, but it is only one of the potential failure strategies the agents may adopt. It may prove beneficial to provide the agents with a set of basic contingency actions. These may include specifically seeking out groups of other agents or traveling to a random known location where other agents may be. In general, increasing the potential for interaction with a novel set of agents should increase the likelihood of successfully finding missing information.

4.5 Natural-language Interaction between Virtual Agents and Avatars

This section details our natural-language processing and generation approach that allows virtual agents to engage in natural-language interactions with other agents and user-controlled avatars. This includes responding to agent or avatar questions and statements, as well as posing questions which resolve uncertainty in the agent's plans and facilitate achievement of the agent's goals.

4.5.1 Parsing Natural-language Utterances

To understand incoming utterances, our agents leverage state of the art shallow semantic parsing (Rasa.ai, 2017). Shallow semantic parsers are trained on a corpus of example sentences to label the sentence with an "intention" and extract from it natural-language *named-entities* (Manning et al., 2014). To avoid confusion with knowledge entities, we refer to named-entities as NLE (natural-language entity) and intentions as NL-I (natural-language intention).

The NL-I of an utterance is a label which is used to categorize the utterance and determine an appropriate response. We provide training examples for specific questions and answers our agents should be capable of responding to. We refer to these as in-domain NL-I. They are: predicate question, predicate answer, attribute question, and attribute answer. Aside from domain specific NL-I, we provide example data for five generic NL-I, which we refer to as out-of-domain NL-I. These are: greeting, thanks, farewell, affirmation, and fallback (i.e. random dialogue). Section 4.5.1.1 details how we generate training sentences automatically for our in-domain NL-I.

NLEs are specific words in an utterance which are considered important for understanding its meaning. Typically, example sentences for each NL-I are also annotated with the relevant NLEs. We specifically provide training data for five NLEs: attribute instances, attribute types, predicate types, knowledge entities, and addressees. Section 4.5.1.1 details how we generate training sentences automatically for our target NLEs.

As an example, the utterance "Where is object A?" would receive the NL-I *predicate question* as it relates to the location of the object. The parser would also recognize two NLEs, "object A" of type *knowledge entity* and "where" of type *knowledge predicate*.

4.5.1.1 Training the parser

To generate training data, we couple our domain descriptions with an English lexicon. The lexicon provides part of speech information for the set of words in our problem-domain as well as subjective and objective verb tense information. The lexicon also provides a set of sample usage sentences which we annotate with NL-Is and NLEs. To train the parser, sample sentences are drawn from the lexicon and the template parameters are bound to corresponding entries from the knowledge base. We also provide a set of basic responses for the set of out-of-domain NL-Is. Limiting the shallow parser to few NL-Is and NLEs allows us to train the parser using tens of examples rather than hundreds or thousands used for modern voice assistants.

For the results demonstrated in Section 4.6.2, we created a custom, limited lexicon. However, our method would generalize to a common lexicon provided that the sentence templates could be extracted. Recent work (Lindsay et al., 2017) has demonstrated the ability to extract planning domains automatically from text and may provide a potential avenue for automatic tagging of lexical entries.

4.5.2 Understanding Utterances from Avatars and Other Agents

As mentioned in Section 4.3, Each agent "hears" utterances issued by other avatars and agents that are visible with respect to static obstacles, and are within a tunable hearing range. Each utterance is passed to the shallow semantic parser and the NL-I and NLEs are returned. For out-of-domain NL-I, the agent can respond with one of the example responses provided in the lexicon.

To map an utterance to the planning framework, the recognized NL-I must be an in-domain NL-I and the utterance must contain at-least one NLE of type *knowledge predicate* or *knowledge attribute*. The utterance should also contain at least one *knowledge entity* (NLE). The agent maps the recognized NLEs to their knowledge-base equivalents. The agent then attempts to construct a belief from the detected entities according to whether a predicate or attribute was detected. For predicates, entities are matched to the slots of the predicate. For attributes, entites are mapped to the attribute relationship.

Consider this statement generated from the example above, “Key one opens Safe one”. The NLE “opens” would map to knowledge *knowledge predicate*. “Key one” and “Safe one” would map to *knowledge instance*s. The planner would construct the complete knowledge predicate $Opens(Key_1, Safe_1)$.

In the case of questions, if a slot is missing from the constructed predicate or attribute belief, the missing information is used as the subject of the question. If no information is missing and a complete belief can be constructed, the question is assumed to be a confirmation question.

The question “Does key one open safe one?” would receive the NL-I *predicate question* and the same NLEs as the statement form. The NLE “opens” would map to knowledge *knowledge predicate*. “Key one” and “Safe one” would map to knowledge *knowledge instance*s. In this case, the complete predicate $Opens(Key_1, Safe_1)$ would be interpreted as a confirmation question. Similarly, the question “Which key opens safe one” would map to the predicate $Opens(?, Safe_1)$ and be interpreted as a question.

In some cases, such as the question ”Is object A in location B?”, no specific predicate information is given. However, if the agent can find a predicate which accepts all the detected entities, it can be inferred from the utterance. Table 4.1 provides several example mappings for NL-Is, NLEs, and constructed beliefs from the museum benchmark (see Section 4.6).

Once a belief is constructed, the agent queries its knowledge base to determine an appropriate response to the question. For confirmation questions, if the agent finds a belief matching the query, the agent responds in affirmation or negation, i.e. “Yes, Key one opens safe one.” For information questions, the agent will issue a response for each candidate found which satisfies the query belief, i.e. “Key one opens safe one. The master key opens safe one.” For utterances labeled as answers, if a complete belief is constructed, it is added to the agent’s knowledge base.

4.5.3 Generating Questions and Statements

Questions: Each uncertain predicate in the agent’s plan must be resolved in order to satisfy the agent’s desires. To generate a question for an uncertain item, the agent first maps the attribute or predicate in question to the lexicon to discover potential template questions for the given item. The production templates are augmented with appropriate slots for entities, predicates, etc. The agent binds the entities from its uncertain predicate to the NLE slots in the production template. If all entities in the question are bound and all NLE slots in the production are complete, the production sentence can be uttered. We maintain several production templates for each predicate and attribute to generate variation in the agents’ utterances. The agent may

Table 4.1: Sample mapping from the shallow parser to knowledge queries in the museum benchmark (Section 4.6). Utterances are parsed into NL-Is (natural-language intentions) and NLEs (natural language entities). Entities are matched to relationships and a knowledge query is constructed.

Utterance	NL-I	NLEs	Mapped Belief
where is the venus de milo	predicate question	knowledge entity: venus de milo predicate: where	InSpace(?,Venus)
What material is the venus de milo	attribute question	knowledge attribute: material knowledge entity: venus de milo	statue(Venus).material=?
venus de milo is located in gallery a	predicate answer	knowledge entity: venus de milo predicate: located knowledge entity: gallery a	InSpace(Venus, GalleryA)

optionally determine to whom the question is addressed. If there is one nearby agent, the agent can specifically address the other agent using an arbitrarily assigned, unique phonetic name.

Statements: Similar to questions, statements are bound by matching the knowledge predicate to a sample production template. Once a question is received, the agent performs a query into the knowledge-base. If an answer is found, i.e. a predicate or attribute belief which satisfies the query, the agent matches the belief into production templates for the attribute or predicate. If no answer is found, the agents are given a set of generic responses representing a lack of knowledge, e.g. "I'm sorry. I don't know."

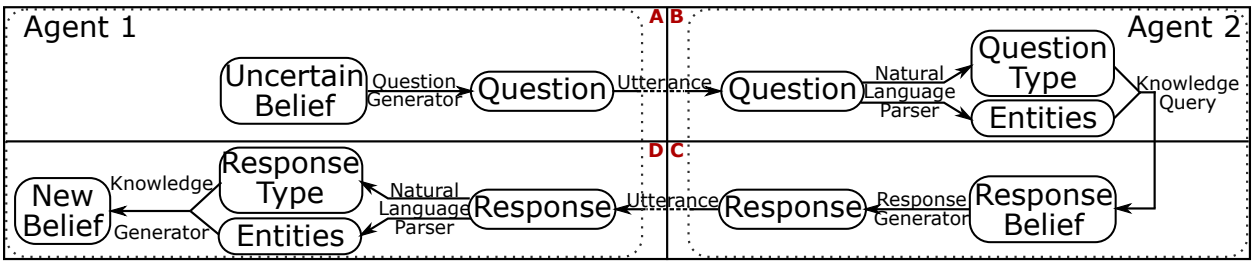


Figure 4.4: **Natural Language Communication for Virtual Agents**: This figure illustrates a sample interaction between two agents using our natural-language approach (clockwise from top). **(A)** Agent 1's plan yields an uncertain belief. The agent generates a question from the belief template. The question is communicated as a natural language utterance. **(B)** Agent 2 receives the utterance and parses it into the relevant question type and entities. The agent queries its knowledge-base for an answer to the question, yielding a response predicate. **(C)** Agent 2 uses our approach to generate a response utterance. **(D)** Agent 1 receives the utterance, generates the appropriate response type and entities, and processes these into a new belief which is stored in the knowledge-base.

The agents process and produce natural language utterances as needed to respond to questions or proactively seek information. No implicit information is transmitted between agents, which allows the agents

Table 4.2: **Performance Benchmark Details**. We detail number of agents, desires, and the natural-language interaction details of the benchmark scenarios including how many questions were asked, statements made, facts overheard by agents, and parser failures. We observe, as expected, that as the number of agents and desires increases, the amount of information gained from overhearing nearby agents increases.

Scene	Agents	Desires	Statements	Questions	Facts Overheard	Parser failures
Anti-podal Circle	10	30	14	5	28	0
Evacuation	11	10	4	0	20	0
Museum	5	9	20	13	3	0
Trade Show	4	1	3	2	0	0

to communicate with agents or avatars without distinction between the methods of interaction. Figure 4.4 details an example exchange between two agents.

4.6 Results

In this section, we provide implementation details and highlight the effectiveness of our novel approach in generating interactions among virtual agents and between virtual agents and avatars given uncertain information, as well as the role of verbal communication in resolving uncertainty.

4.6.1 Implementation

Our experiments were conducted on a desktop PC with an Intel Xeon E5 CPU, NVIDIA TitanX GPU and 16 GB of ram. We coupled our propositional planner with Rasa NLU (Rasa.ai, 2017) for semantic parsing. User utterances were captured via microphone and automated speech recognition. Our algorithm was implemented in python, and our VR experiments were performed with the Occulus Rift HMD. We couple our approach with the 3D animation system described in Narang et al. (2018).

4.6.2 Benchmarks

We demonstrate the results of our approach on several challenging multi-agent scenarios (Figure 4.3) including:

Anti-podal Circle: In this scenario, 10 agents and an equal number of 'goal-objects' are distributed on the circumference of a circle (Figure 4.3(a)). Each agent is given a desire to retrieve a set of randomly assigned goal-objects. However, the agent's initial belief set may not include information about the desired

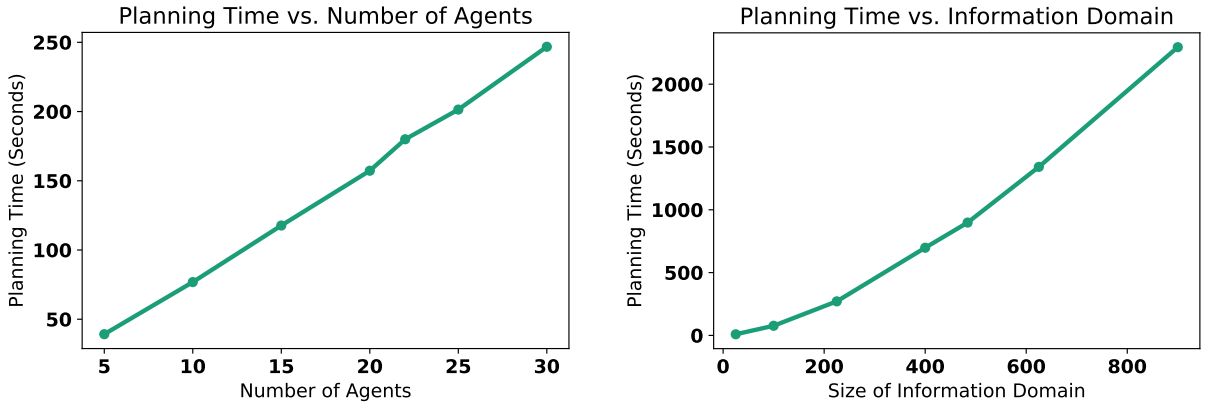
Table 4.3: **Benchmark comparisons with and without natural-language interactions**. This table compares performance of our algorithm with and without natural-language interactions. We find that agents who are able to communicate accomplish their goals more quickly and benefit from nearby communication of other agents.

Scene	Agents	Planning Time	Replans	Replan Time (S)	Solution Time(S)
Anti-podal Circle	10	76.390	4	0.007006	67.85
Without NL-I	10	75.917	4	0.005508	73.95
Evacuation	11	1.009	10	0.000278	36.60
Without NL-I	11	1.771	10	0.000263	53.70
Museum	5	6.811	9	0.013800	159.35
Without NL-I	5	12.361	2	0.006314	209.40
Trade Show	4	0.123	1	0.001375	52.65
Without NL-I	4	0.128	1	0.001116	98.95

goal-objects. In such cases, the agent resorts to engaging in verbal communication with other agents in order to resolve uncertainties regarding the location of the goal-object. This benchmark illustrates the ability of our algorithm to plan with incomplete information, automatically generate uncertainty resolution actions, and facilitate verbal agent-agent communication. We find that asking questions reduced overall simulation time compared to our method without the ability to resolve uncertainty via interaction.

Evacuation: We simulate a fire evacuation scenario in a Y-shaped corridor (Figure 4.3 (b)). A group of 10 agents approach a junction and must choose one of two passageways to reach their goals. Prior to their approach, each agent randomly selects its desired passageway. Unknown to the agents, the right passageway is obstructed due to a fire breakout. As the agents approach the junction, another agent acting as a first-responder redirects them to the leftmost passageway using verbal communication. Without our algorithm, agents are unable to communicate and some take the rightmost passageaway forcing them to retreat. This leads to a 47% increase in evacuation time as is shown in Table 4.3.

Avatar & Multi-agent Tradeshow: In this scenario, agents explore a complex tradeshow scenario with a large number of booths (Figure 4.3 (c)). One of the agents is given the desire to go to the 'registration desk' but does not know its location. Using our two-stage planning approach, the agent generates a plan template and a set of candidates for the location of the registration desk. It then verbally interacts with nearby agents to resolve the uncertainty and find the location of the registration desk. We also simulate this scenario with a user in immersive settings (Figure 4.3 (d)). The user controls a virtual avatar and is given the task of finding



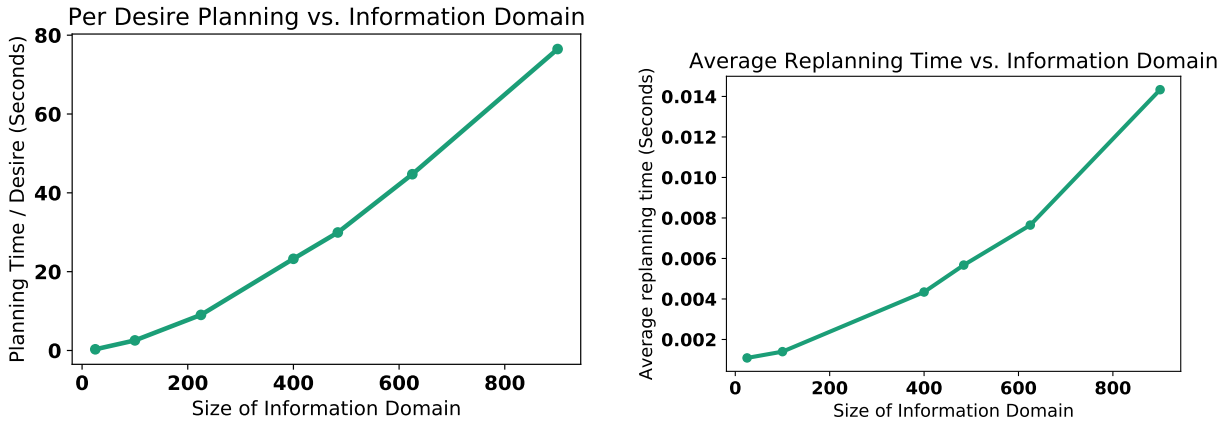
(A) (B)

Figure 4.5: **Performance Results Varying Number of Agents and Problem Size**: **(A) Varying agents on a fixed domain size (100 predicates):** We observe that our algorithmic approach's performance scales linearly in the number of agents. **(B) Varying domain size for a fixed number of agents (10 agents):** We observe that our algorithm's performance scales exponentially in the size of the problem domain. This is consistent with propositional approaches. However, our two stage planning approach enables rapid replanning after the initial planning step, reducing overall planning time.

the registration desk. The user asks questions, and receives meaningful responses from the agents (Figure 4.3 (e)).

Avatar and Multi-agent Museum: This scenario demonstrates an art museum. In the multi-agent case, each agent is given a set of statues to visit. However, knowledge of the locations of the statues is randomly assigned amongst the agents. Each agent must seek out other agents with whom they can interact to acquire the location of the statuary they are seeking. In the avatar case, the user is given a specific statue to find in the museum. During the user's exploration, virtual agents approach the user and request the locations of statues the user has previously visited. If the user responds with the appropriate information, the agents are able to complete their plans.

Multi-Avatar Hide-and-Seek: This scenario depicts multiple user avatars engaged in a hide-and-seek game in a virtual environment populated by many virtual agents. The hiding avatar chooses a room in which to hide, and the seeking avatar must interact with the virtual agents to obtain the location of the the hider. The necessary information is disbursed amongst several agents, requiring the seeker to interact with multiple virtual agents to find the hider.



(A) (B)

Figure 4.6: **Performance Results per Desire and Replanning Call**: **(A) Per desire planning time varying domain size:** We observe that the per-desire planning time scales exponentially as a function of the domain size. This is consistent with propositional planning approaches. **(B) Average time per replanning call:** Replanning using our two-stage algorithm can be performed on the order of milliseconds even in complex information domains. This reduces the overall planning time substantially.

4.6.3 Performance Analysis

We conducted a series of repeated trials on the Anti-podal circle benchmark described in Section 4.6.2. In each trial, we increased the number of virtual agents or the number of potential goal-objects for the agents. Overall, we find that our planning algorithm scales linearly in the number of potential targets for a single agent and linearly in the number of agents. Consistent with other propositional planning approaches, we find our algorithm scales exponentially in the size of the information domain. Figure 4.5 details our performance evaluation results. In addition, we evaluated the replanning time for each agent as the agents resolve uncertainty in their plans. We find that our two-stage approach yields negligible replanning times even though agents must resolve new information and choose new candidates. Figure 4.6 details how our two-stage approach reduces overall computation time be enabling rapid replanning.

In a typical scenario, the generation of candidates is only performed once, and can done as a pre-processing step for the scenario, leading to interactive agents capable of replanning as a response to verbal communication with extremely small overhead.

4.6.4 User Evaluation

We conducted a user study to evaluate the plausibility of agent-avatar interactions and the overall simulation generated as a result of our algorithm. Following from (Nass et al., 2000), we sought to evaluate

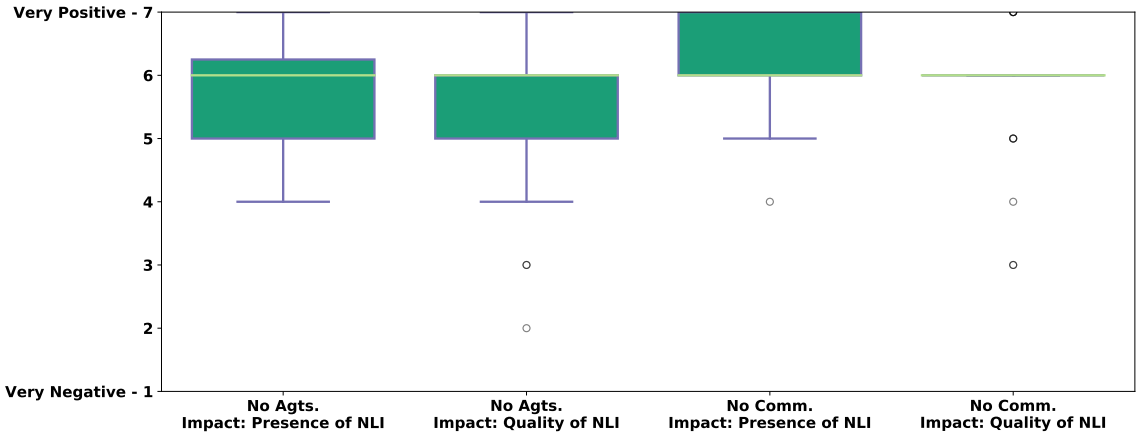


Figure 4.7: **Participant Impact Perception:** This figure shows the perceived impact on participant preference for our method over several factors. **No Agts.**: Compared against the no agents condition, participants perceived the presence and quality of natural language interactions (NLI) provided significant positive impacts on their preference (5.82 ± 0.94 and 5.29 ± 1.24). **No Comms.**: Compared against the no communication agents condition, the presence and quality of NLI was an even more important factor in the participants preference (6.18 ± 0.77 and 5.75 ± 1.00). This is consistent with expectations, as the presence of the virtual agents is consistent across both examples and is less impactful.

whether our approach generated improvement in the overall perception of simulations between virtual agents and avatars.

Experiment Goals & Expectations: We hypothesize that verbal communication between agents and avatars will enhance the perceived plausibility of the simulation, and generate positive impressions as compared to the control conditions. Moreover, we expect that more effective plans will be generated as a result of agent-avatar communication.

4.6.4.1 Comparison Conditions

- **No Agents**: In the no agents case, a user avatar explores a virtual environment without any virtual agents present.
- **No Communication**: In the no communication case, a user avatar explores a virtual environment with agents who could not interact using natural-language communication.

4.6.4.2 Experimental Design

This study was conducted based on a within-subjects, paired-comparison design. Each scenario was displayed with a text-based prompt to provide the appropriate context. Participants were shown two pre-

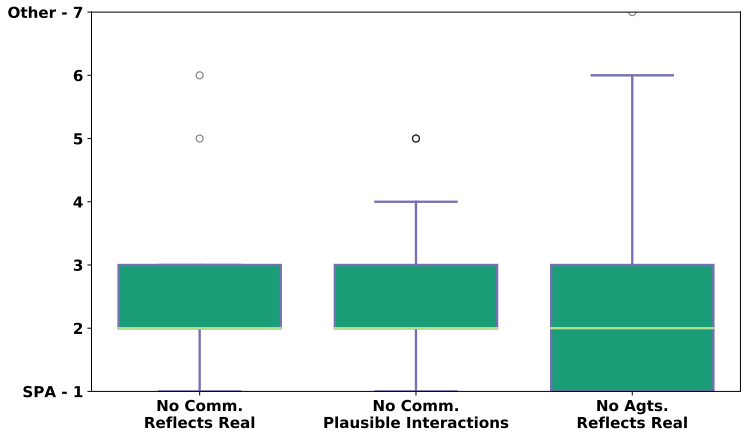


Figure 4.8: **Participant Preference in User Evaluation** Participants clearly found our method produced simulations more consistent with real world scenarios. Compared to the no agents condition, **No Agts.**, participants felt SPA generated more realistic simulations (2.29 ± 1.15). Compared to agents without natural-language interaction, **No Comm.**, participants felt SPA generated more realistic simulations as well (2.29 ± 0.81). In addition, participants felt the natural-language interactions increased the overall plausibility of the agent-avatar interactions compared to agents without the ability to interact using natural-language interaction (2.46 ± 1.20).

recorded videos in a side-by-side comparison of our method and one of the comparison methods. They were then asked to answer a short questionnaire before moving on to the next scenario. The order of scenario and the positioning of the methods was counterbalanced.

4.6.4.3 Environments

The multi-agent tradeshow scenario and multi-agent museum (Section 4.6.2) were used in this study. Three confederates were recruited to participate as the avatar in the environments. In trials using our method, the confederate was allowed to interact with the agents using natural-language communication. In each case, the avatar was piloted from a first-person view. Their interactions were recorded via screen capture and a microphone.

Tradeshow: The avatar was instructed to find the "registration booth". They were shown a picture of the booth before beginning their task but were not told its location. In the SPA case, virtual agents in the environment were able to interact and provide the location of the booth to the avatar. Figure 4.10(C) and (D) demonstrate this scenario.

Museum: The avatar was instructed to find a specific statue in the museum but was not told the location of the statue. The statue in question was Lucy, courtesy of the Stanford University Computer Graphics

Laboratory. In the SPA case, a virtual agent near the avatar's starting position was provided knowledge of the location. The avatar was able to ask this agent the location of the statue. In addition, two agents were placed along the path to the goal who would interrupt the avatar's progress and ask the avatar for the locations of other statues as they passed. Figure 4.10(A) and (B) demonstrate this scenario.

4.6.4.4 Metrics

Participants were asked a set of common questions for both comparison methods, with specific additions for each comparison method.

Common Metrics: Participants were asked to indicate which simulation more closely reflected a real-world scenario on a Likert scale with 1 indicating strong preference for the method presented on the left, 7 indicating strong preference for the method presented on the right, and 4 indicating no preference. They were then asked the impact of the following items on their preference: the presence of natural language communication, the quality of the verbal responses from the agents, and the quality of the animation. These were answered on a Likert scale with 1 indicating strong negative impact, 7 indicating strong positive impact, and 4 indicating no impact.

No Agent Metrics: Participants were additionally asked what impact the presence of the virtual agents had on their preference.

No Communication Metrics: Participants were additionally asked which of the methods demonstrated more plausible interactions, in which simulation did the agents appear to benefit more from their interactions with the avatar, and in which simulation did the avatar appear to benefit more from their interactions with the virtual agents.

4.6.4.5 Results

The study was taken by 14 participants. We normalized the data for comparative questions such that a response of 1 indicates strong preference for our method. We collapsed the common metrics across trials as well as plausibility of interactions question for the No Communication metric and the presence of virtual agents from the No Agents metric. We performed a one-sample t-test comparing the mean of each question with a hypothetical mean of 4 (no preference or no impact). We limit our discussion below to questions which directly deal with natural-language interaction and preference for the methods.

Table 4.4: **Frequency of Responses in User Evaluation**. This table shows the frequency of participant responses in the user evaluation, as well as the means and p-value for a one-sample t-test with a hypothetical mean of 4. For comparative questions, responses less than 4 indicate preference for our agents. For impact questions, responses greater than 4 indicate positive impacts.

Question	1	2	3	4	5	6	7	mean	std	p-value
NL-I Agents vs Non-Interactive Agents										
Comparative Questions (NL-I Agents left)										
More closely reflects real scenario	6	13	7	0	1	1	0	2.29	± 1.15	< 0.000
Agents benefit more from interaction	11	4	1	11	0	1	0	2.57	± 1.53	< 0.000
User benefits more from interaction	17	10	0	0	0	1	0	1.54	± 1.00	< 0.000
More plausible interactions	5	13	5	2	3	0	0	2.46	± 1.20	< 0.000
Impact Questions										
Presence of natural Language	0	0	0	1	3	14	10	6.18	± 0.77	< 0.000
Quality of the verbal interactions	0	0	2	1	3	18	4	5.75	± 1.00	< 0.000
Animation of the virtual agents	0	0	4	13	3	3	5	4.74	± 1.36	0.010
NL-I Agents vs No Agents										
Comparative Questions (NL-I Agents left)										
More closely reflects real scenario	9	10	6	1	0	1	1	2.29	± 1.46	< 0.000
Impact Questions										
Presence of the virtual agents	0	0	0	0	8	10	10	6.07	± 0.81	< 0.000
Actions of the virtual agents	0	0	0	6	9	10	3	5.36	± 0.95	< 0.000
Presence of natural Language	0	0	0	3	6	12	7	5.82	± 0.94	< 0.000
Quality of the verbal interactions	0	1	2	3	7	12	3	5.29	± 1.24	< 0.000
Animation of the virtual agents	0	0	3	11	10	2	2	4.61	± 1.03	0.004

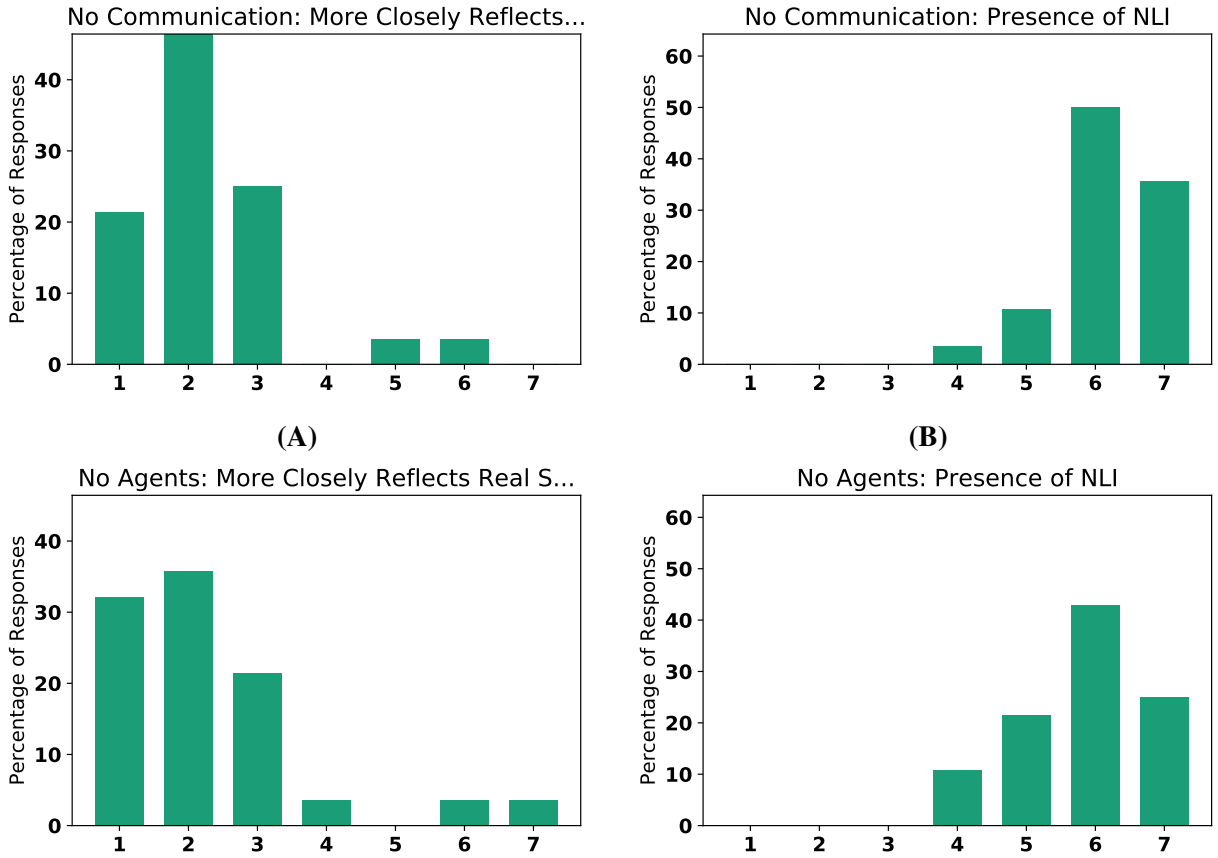
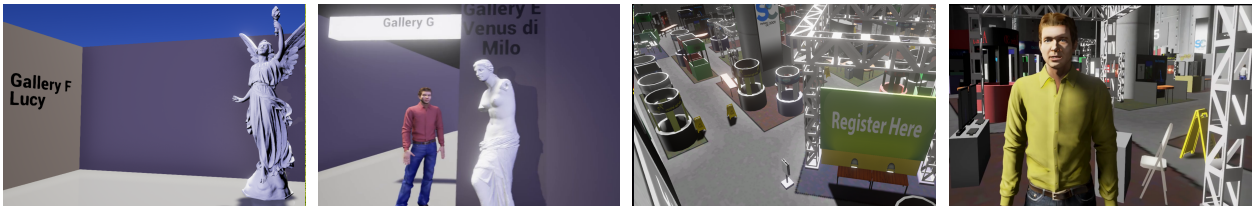


Figure 4.9: **Histogram data of User Responses (A)** Participants in our evaluation found simulations using SPA significantly more plausible compared to simulations with a prior approach (above) and simulations with no agents (below). This indicates that the presence of agents has a positive impact on plausibility and that our agents behave sufficiently well to increase plausibility with respect to agents lacking SPA. **(B)** Participants in our evaluation found the presence and quality of the natural language interactions had a significant impact on their preference for our approach to simulations with agents lacking SPA (above) and simulations without agents (below).

We found the question "Which simulation more closely reflects a real-world scenario" significant in both the no agents condition ($t(27) = -6.204, p < 0.000$), and the no communication condition ($t(27) = -7.887, p < 0.000$). We found the question "What impact did the presence of natural language interaction have on your answer" significant in both the no agents condition ($t(27) = 10.200, p < 0.000$), and the no communication condition ($t(27) = 14.925, p < 0.000$). We found the question "What impact did the quality of the verbal responses from the agents have on your answer" significant in both the no agents condition ($t(27) = 5.473, p < 0.000$), and the no communication condition ($t(27) = 9.218, p < 0.000$). We found the question "In which simulation did the interactions between the user and the agents seem more plausible" significant in the no communication condition ($t(27) = -6.765, p < 0.000$). It was not asked of the no



(A) (B) (C) (D)

Figure 4.10: **User Study Scenarios (A)**: In the museum, the avatar searches for the statue of Lucy in a gallery at the far-side of the museum. **(B)**: While exploring in the SPA condition, agents approach the avatar to ask the location of other statues the avatar has passed. The avatar can choose to provide information to the agents. **(C)**: In the tradeshow, the avatar must find the registration booth pictured here. **(D)**: In the SPA case, the avatar can ask virtual agents for the location of the registration booth.

agents condition. We found the question "What impact did the presence of virtual agents have on your answer" significant in the no agents condition ($t(27) = 13.478, p < 0.000$). It was not asked of the no communication condition. Figure 4.7 shows the participants' preference across the comparative questions. Figure 4.8 shows the participants' perception of the impact of natural language interactions. In addition, Figure 4.9 details specific question analyzed. The complete set of responses is presented in Table 4.4.

Analysis: The participant responses clearly demonstrate the benefits of our algorithm in terms of generating plausible agent-avatar interactions (2.46 ± 1.20). Expanding the interaction capacity of the virtual agents beyond movement interactions has a positive impact, and participants believed the quality of the natural-language interaction generated by SPA was a positive factor when comparing against agents without the ability to interact (mean impact:5.75 ± 1.00) or cases with no agents (mean impact: 5.29 ± 1.24). In both comparative conditions, either without agents or without communication, participants found our method to generate simulations and interactions with better reflect real-world scenarios (2.29 ± 1.15 and 2.29 ± 1.46).

Overall, the results of our study show that participants find agents with natural-language interaction capability to be significantly more plausible than those without. More importantly, they show that participants found the natural-language interactions generated by SPA to be a significantly positive factor on their preferences. This indicates that SPA yields natural-language interactions which are plausible and effective.

4.7 Conclusion

We have presented a novel algorithm for generating virtual agent plans under uncertain conditions and natural language interactions between virtual agents and avatars for multi-agent multi-avatar environments. Our approach allows agents to plan with uncertain information, engage in question and answer-based dialog

and effectively accomplish their individual goals while facilitating plausible avatar-agent interactions. We have demonstrated how our approach can be used to provide significant improvements to behavior plausibility for virtual agents in a shared environment and detailed a user-study which verified our approach's advantage. Moreover, our approach can simulate dozens of interactive virtual agents in real time. Overall, SPA seeks to improve limitations of interactivity in typical multi-agent planning approaches and addresses limitations of single agent-avatar interactions in typical conversation agent approaches.

Our algorithm is part of ongoing research and has several limitations. First, as with many propositional planning approaches, the performance of our planning algorithm degrades with the complexity of the agents' desires and the problem domain. As our agents need to generate a full plan template typically only once, we will explore the use of planning as a pre-computation step to simulation and caching plan templates for subsequent simulations to address this. In addition, while the use of shallow-semantic parsing enables verbal interactions, it is very sensitive to training data, making the communication quality sensitive to lexicon quality. Further, shallow semantic parsing is limited in the sentence structures it successfully interprets. While our work focuses on simple interactions, extending to more conversational agents may require exploring alternate parsing strategies. Additionally, determining the most appropriate question is a non-trivial problem, and our approach can generate somewhat repetitive dialog when translated to natural language. Also, when interacting, agents are able to produce greetings, questions, answers, and arbitrary dialog only. In the future, we will seek to expand the interaction capability of our virtual agents to include conversational context keeping and other dialog actions.

In addition to addressing the limitations, there are several avenues for future work. First, our planning formalism would support partial-order planning (Russell and Norvig, 2009), which may give us speedups in the plan computation time. We will also work to improve our attention models. Our current attention models are simple and rule-based. All agents not actively engaged in a task or utterance hear and process nearby utterances. Research in human-agent interaction offers avenues for exploring different attention models (Narang et al., 2016). In the future, we would additionally seek to gather data on real human social interactions and develop a heuristic model for agent responses to questions.

CHAPTER 5: Evaluation of Pedestrian Models for Loading, Unloading, and Evacuating Aircraft

5.1 Introduction

Air travel is an integral part of transportation infrastructure. Over 2.8 billion passengers traveled by airplane in 2013, and that number is expected to double over the next fifteen years[1]. Balancing passenger safety, passenger comfort, and airline profitability is a challenging, evolving task. One factor in coping with expanding volumes of flights and passengers is reducing the amount of time a plane stays on the ground at an airport (turning time); it allows fewer aircraft to service more passengers and makes the experience more pleasant by reducing delays and layovers. Reducing loading and unloading times can improve turning time, but airlines need the ability to experiment with boarding orders, seating arrangements, and overhead layouts without interrupting commercial service. Simulation is one way to achieve this.

For the certification of any aircraft designed to carry more than forty-four passengers, US federal law requires a live demonstration of the aircraft being safely evacuated in under 90 seconds (14 C.F.R.25.803, 1990). The flight is certified for commercial use only upon a successful demonstration. Failed demonstrations require costly mitigation efforts before a new attempt can be made. The ability to perform test evacuations in a simulated environment, before engaging in live testing or, possibly, in lieu of a live test, would be extremely useful for aircraft designers and manufacturers; it would save time and and money by reducing the cost of design evaluation.

We propose the use of agent-based simulation to model passenger behaviors in loading, unloading, and evacuating commercial aircraft. We have constructed a framework, Ped-Air, which not only addresses the navigation issues in these scenarios, but incorporates the behavior and characteristics of and interactions between passengers aboard an aircraft. We highlight the performance of Ped-Air across different scenarios.

The rest of the chapter is organized as follows. We briefly relate work on pedestrian simulation and evacuation in Section 5.2. We highlight some major challenges in simulating passenger behaviors on an

[1] http://www.bbc.com/news/business-26148702

aircraft and present our agent-based simulation framework in Section 5.3. Finally, we detail our results, and offer conclusions and future direction.

5.2 Related Work

5.2.1 Pedestrian Simulation

In agent-based simulations, each pedestrian is modeled as a discrete entity, capable of planning and acting on its own. Ped-Air implements a particular abstraction of agent-based simulation. This abstraction is based on a coupled global and local navigation mechanism connected by a *preferred velocity*. The preferred velocity, sometimes called *desired velocity*, is the velocity an agent would take if not affected by local dynamic conditions (Van den Berg et al., 2008b; Helbing et al., 2000; Zanlungo et al., 2011). Agents plan a path with respect to the fixed environment, continuously adapting that plan to avoid dynamic hazards.

5.2.2 Global Navigation

The global navigation mechanism devises a plan for optimally progressing towards a goal while avoiding collisions with the static environment. This plan serves as the basis for computing an instantaneous preferred velocity for the agent. The preferred velocity is the "optimal" velocity for the agent to realize the navigation plan, considering the agent's current position and state. Its optimality can be defined according to arbitrary criteria based on the static environment (Latombe, 1991; Snook, 2000; Geraerts et al., 2008).

Many techniques exist to determine the preferred velocity. Some involve discretizing the traversable space into connected components on a graph and using graph search algorithms (e.g. A*). This type of techniques includes roadmaps (Latombe, 1991), navigation meshes (Snook, 2000), and corridor maps (Geraerts et al., 2008). Other techniques, such as potential fields, do not explicitly compute a path but rather discretize the space into a field that is the gradient of some cost function (Khatib, 1986). The preferred velocity in this case is the value of the field at the agent's location.

5.2.3 Local Navigation

Because global navigation only considers the static environment, it may be insufficient when the scene is populated with dynamic elements such as other agents. Local navigation algorithms are used to adapt the preferred velocity computed by the global navigation algorithm to a feasible velocity which respects local,

dynamic conditions. Local navigation is also referred to as “pedestrian model,“ “steering algorithm,” or “collision avoidance” (Reynolds, 1999; Van den Berg et al., 2008b; Helbing and Molnár, 1995).

Different methods and paradigms have been proposed for local navigation. One class of local navigation algorithms is force-based models (Helbing and Molnár, 1995; Johansson et al., 2007; Zanlungo et al., 2011; Chraibi et al., 2010). These algorithms treat agents as mass-particles. The preferred velocity is achieved through a driving force which accelerates the agent towards the preferred velocity as necessary. The actual velocity taken by the agent is computed through the application of the driving force, as well as repulsive and attractive forces depending on local conditions.

Another commonly used approach is velocity-space based techniques (Van den Berg et al., 2008b; Paris et al., 2007; Van den Berg et al., 2011). These approaches use geometric optimization to compute a collision-free, feasible velocity directly in velocity space. Nearby agents and obstacles serve as constraints on an agent’s velocity. A velocity is selected from the available velocity space, based on an optimization criteria.

5.2.4 Behavior Modeling

Ped-Air uses a Behavioral Finite State Machine (BFSM) to represent the mental state (including such factors as immediate goals and mood) of agents in the simulation. FSMs are well-studied in crowd and pedestrian simulation (Ulicny and Thalmann, 2002; Bandini et al., 2006) and animation, and provide an effective mechanism for establishing time-dependent behaviors and goals for agents to accomplish.

5.2.5 Evacuation

Evacuations and panic modeling have been studied by researchers in many fields. Helbing et al. (2000) studied the effect of panic on evacuations in simulation. Burghardt et al. (2012) performed stadium exit experiments with human subjects. Kim et al. (2013) modeled physical interactions between dense crowds. Nilsson and Johansson (2009) studied evacuation experiments performed in a cinema theater. Studies on mass transit evacuation have been performed as well. Galea et al. (2011) performed experiments in the proposed blended wing body aircraft. Galea et al. (2012) examined evacuations on large sea vessels and Liu et al. (2012) performed simulations of evacuating a train tunnel.

5.3 Ped-Air: Simulating Loading, Unloading, and Evacuations

In this section, we highlight the advantages of agent-based methods for behavioral simulation in aircraft, and highlight some of the goals and challenges in terms of the design of Ped-Air. We present the architecture of Ped-Air and techniques used to model different behaviors.

5.3.1 Agent-Based Simulation

Agent-based simulation is not the only paradigm which could be applied to simulating passengers on an airplane. Continuum crowds (Narain et al., 2009), or event-based simulations could also be used. However, agent-based simulation provides unique advantages. First, agent-based models inherently support modeling heterogeneity of agent goals, behaviors, and characteristics. This is important because every passenger in the aircraft has unique goals and targets (e.g., luggage or seat location). Passengers also display a wide range of personal characteristics, including moving at different speeds, and response to stress.

Agent-based methods also naturally support modeling complex interactions and relationships. During unloading, flight staff will assist infirm passengers. During loading, passengers will temporarily move out of the aisle seat to allow another passenger to sit in a neighboring seat. In each case, these actions affect how passengers respond to others. Agent-based simulation uniquely captures not only the behavioral coordination, but the physical coordination that takes place between passengers.

Agent-based methods additionally can be parallelized on current commodity CPUs and GPUs. As the behavioral complexity and simulation domain expand, it will be important to have a simulator which supports many independent entities as well as individual actions and plans. Additionally, efficient simulation techniques would allow safety engineers the ability to test many possible emergency situations in rapid succession or parallel. Because of the nature of agent-based simulations to provide both the intentions of the agent (e.g., getting a bag, waiting for the isle to clear, evacuating) and the trajectory, they provide a strong foundation for behavioral analysis.

5.3.2 Challenges in Simulating Passenger Behavior and Navigation

Simulating passengers in an aircraft provides some unique challenges. One of these challenges arises from the sensitivity to preferred velocity computation. Many global planning mechanisms involve discretization of the traversable space and can introduce small errors in preferred velocity which lead the agent into collision

with an obstacle. The preferred velocity does not necessarily point directly into an obstacle, but rather leads the agent on a path with insufficient clearance. In cases with few nearby agents to consider and a relatively large amount of traversable space, we find the local planner is able to correct these errors. However, we observe in the aircraft that these issues lead to the agents failing to find a collision-free, non-zero velocity. Agents stuck in these conditions become obstructions for others, often causing a cascade effect and a deadlock in the simulation.

A further challenge in this environment is the transient nature of preferred velocities and agent goals. Agents change preferred velocities often, turning corners or changing direction completely to return to their seats. Agent-based models tend to use simple extrapolation on current velocities to make predictions and plan for future states. These assumptions become particularly problematic as agents seek to move into the aisles of the aircraft, where agents can reach a stalemate waiting for one another to turn a corner.

Pedestrian models tend to assume symmetric relationship between pedestrians; both pedestrians experience a deviation from preferred velocity to avoid collision. However, the social conventions of passengers aboard aircraft suggest that, in some occasions, one passenger yields to another. It is important to model these types of *asymmetric* relationships in simulation. Passengers also engage in cooperative and sometimes counter-intuitive behaviors which are difficult to capture, but are necessary in order to successfully load and unload the aircraft. For example, a seated passenger will rise and leave their row to allow others to enter. Other passengers not involved with the exchange respect the motion and wait until the aisle clears to resume moving. Also, passengers moving against the flow of the aircraft slide into an unoccupied seat to allow others to pass.

5.3.3 Ped-Air Architecture

Ped-Air is built on the Menge crowd simulation framework (Curtis et al., 2016). Menge is a modular, agent-based framework for research in crowd and multi-agent simulation. Menge models the work of simulating crowds in complex environments as the composition of various computational elements (e.g., global navigation algorithms, spatial query structures, etc.) While the framework provides default implementations of all of the element types, it provides a plug-in architecture to allow for arbitrary element implementations. Ped-Air exploits this plug-in architecture and introduces novel elements, designed specifically to bring about agent behaviors deemed consistent with actual passenger behaviors (see Figure 5.1).

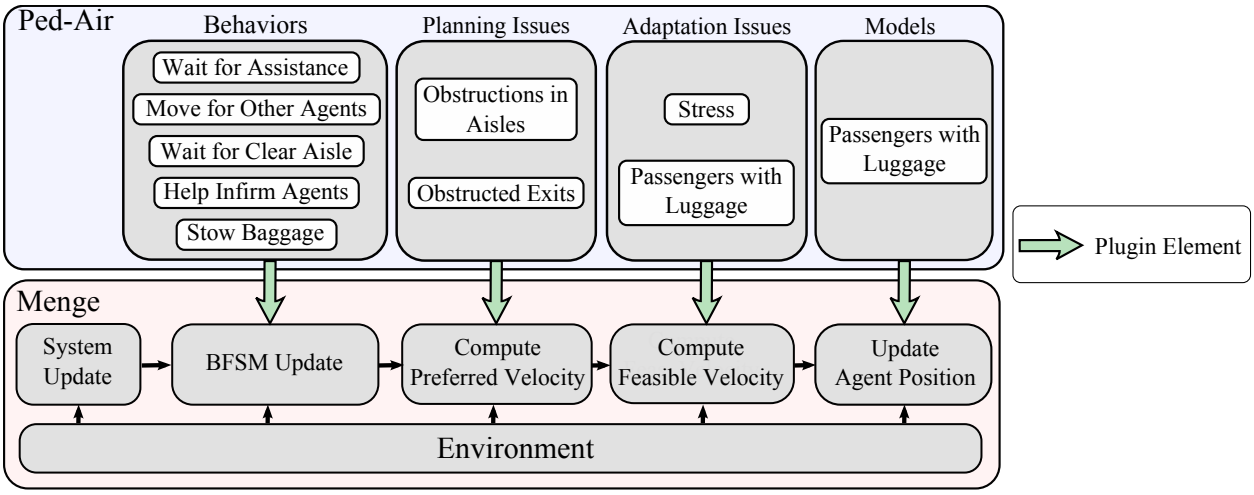


Figure 5.1: The architecture of Ped-Air and how it is built on top of Menge (Curtis et al., 2016). We highlight different behaviors modeled in Ped-Air.

5.3.4 Behavior Modeling

We have modeled several behaviors and phenomena typical of airline loading and unloading in our simulations. These behaviors include:

- *Waiting*: Some passengers wait until the space around them and the aisle is clear to exit the plane. The agents only advance from their waiting state to an exiting state when a region forward of the passenger is clear.

- *Infirm*: Passengers wait until a flight attendant comes to assist them off the plane. Flight attendants wait until the aisles clear, then move through the plane assisting infirm passengers. When the attendant reaches an infirm passenger, both agents move into a state which causes them to move together toward the exit – the flight attendant "assisting" the passenger.

- *Bin Searching*: Passengers stow their carry-on luggage in the overhead bins. However, the bin directly overhead may not be available and the passenger must find an alternative location to stow their bag. Bin selection mechanisms such as nearest, farthest, and nearest behind accomplish the search. Each bin has a certain capacity, and each agent's carry-on luggage is assigned a size value. The agent must find a bin that can hold their luggage.

- *Physical Carry-on*: Carry-on luggage is attached to the agent and takes up space in the aisles. The luggage is modeled using a proxy-agent. The agent attaches to its owner and follows them like a normal

agent. The agent and proxy are exempt from colliding with one-another. However, other agents must avoid the luggage.

- *Evacuation Stress*: Evacuations cause stress and increased stress affects how efficiently the passengers can evacuate the plane. We model stress by modifying navigation parameters of the agents over time as stress increases.

- *Aisle Obstructions*: Exits on the plane and aisles can be obstructed increasing evacuation complexity. Obstructions are treated like other obstacles in the simulation; agents must plan and navigate around them.

- *Physiological Variance*: Agents in the aircraft vary in characteristics, reflecting models of gender and age. Agents are assigned a "profile" which determines their simulation characteristics (preferred speed, max neighbors, etc.). By crafting these profiles according to parameter research (Zheng et al., 2013), we can model various types of passenger.

- *Loading Order Variation*: The aircraft can be loaded front to back, back to front, randomly, and by zones. Each agent is assigned a seat according to which strategy is employed, and new strategies can be implemented in the framework.

We illustrate several behaviors realized in the simulator in Figure 5.4. Figure 5.2 shows an example set of BFSM states and transitions for loading the aircraft. For more details on how these elements fit into Menge, we refer the reader to Curtis et al. (2016).

5.3.5 Coordination and Navigation

Ped-Air employs a navigation mesh for global navigation. The navigation mesh decomposes the traversable space into connected, convex polygons. We found this representation to be more amenable to consistently producing error-free preferred velocity, virtually eliminating the observed deadlocks discussed in Section 5.3.2. While it is certainly possible to use other methods (e.g., roadmaps or guidance fields), we found that these approaches required excessive engineering effort and/or computation resources to achieve a comparable quality in preferred velocity.

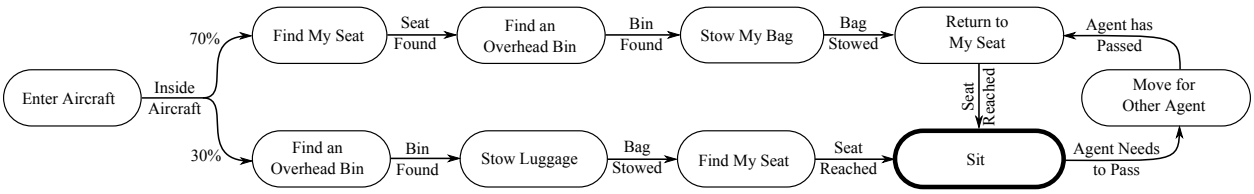


Figure 5.2: An example BFSM for aircraft loading. Agents enter the aircraft, and are split into two groups based on a distribution. Agents either find their seat first, or place their bag first, and proceed to sit. Should another agent need the seated agent to move, the agent can stand and leave their row, returning to their seat once the other agent has passed. The "Sit" state is a final state; once all agents reach this state, the simulation ends.

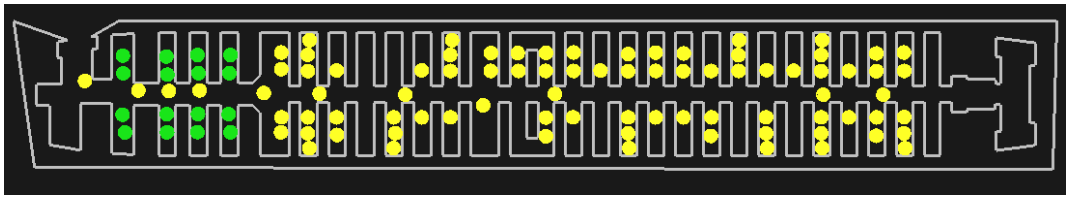
Guidance fields implicitly define paths from any location to a single goal. In the aircraft, each seat is a goal and would therefore require its own guidance field. Furthermore, each guidance field must be computed with a very small cell size to robustly sample the traversable space.

In contrast, roadmaps provide a mechanism to plan paths from any point to any other point. However, roadmaps represent the traversable space as a graph in which the vertices are discrete *samples* of free space. Two samples are connected iff there is a straight, collision-free path between them. The planning is largely constrained to paths along the graph and is therefore sensitive to the number and quality of the samples. This sensitivity requires extensive effort in producing a high quality roadmap. Furthermore, we found that typical techniques for producing *smooth* trajectories from the piecewise-linear paths implied by the graph failed in the aircraft's narrow passages.

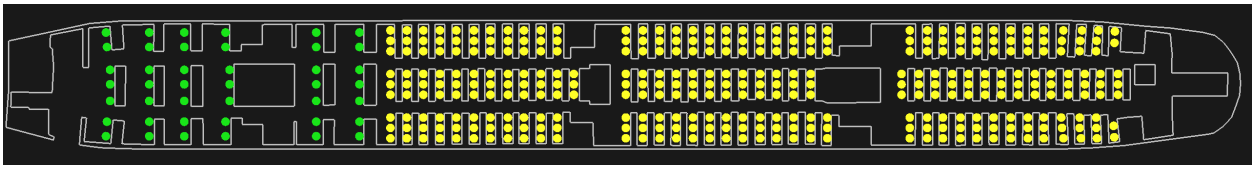
While the underlying framework supports a number of local navigation algorithms, we constrained our exploration to two specific models, a simple, representative social-force-based model from Helbing et al. (2000), and ORCA, a velocity-space based model proposed by Van den Berg et al. (2011). Using typical simulation parameters, both models suffered from severe simulation artefacts; often agents remain on the aircraft indefinitely as evidenced in Table 5.1.

The force based method suffered from deadlocks when unloading the aircraft. The obstacle forces tend to cause oscillation in the narrow rows. As agents reach the aisles, the repulsive forces from nearby agents can cancel the driving force, keeping the agents in the rows indefinitely or deadlocked with the agents across the aisle. In simulations where we lowered force distances sufficiently to prevent this, we had difficulty preventing inter-penetration between agents.

The velocity-space based method suffers from deadlocks as well. When two agents arrive at the aisle from opposite rows, they tend to meet exactly in the center of the aisle without sufficient room for either to



(*a*)



(*b*)

Figure 5.3: **The simulated aircraft layouts in a loading and unloading scenario**. Business class passengers are modeled in green and coach passengers in yellow. (*a*) Agents load onto the 737 aircraft model in random order. (*b*) 376 agents ready to disembark the Boeing 777-300.

advance. This also occurs when agents turn into occupied rows. Agents tend to stop at the boundary of the row because of the local navigation constraints.

In order to successfully unload the aircraft and prevent deadlocks, we use the symmetry breaking algorithm proposed by Curtis et al. (2012). Right of Way shifts the burden of avoiding collision from being evenly shared between agents, to being unevenly distributed. The mechanism allows us to capture asymmetric relationships among the passengers. We assign priority to agents near the front of the plane, and agents who have arrived at the aisle, causing those in the row to yield. For example, while fetching luggage from an overhead bin, an agent is given higher priority; this allows them to largely ignore nearby neighbors, relying on the neighbors to take responsibility for avoiding collisions. This mechanism also gives us the ability to force agents to yield to flight staff, modeling the deference to authority in tense situations and the crew's ability to push through the crowd. We apply a small noise in the base priority values of agents in the aircraft to generate the desired tendency for agents meeting head-on to resolve the deadlock; one agent yields to the other, allowing for progress.

5.4 Experimental Results

Below we highlight some results and experiments performed using Ped-Air.

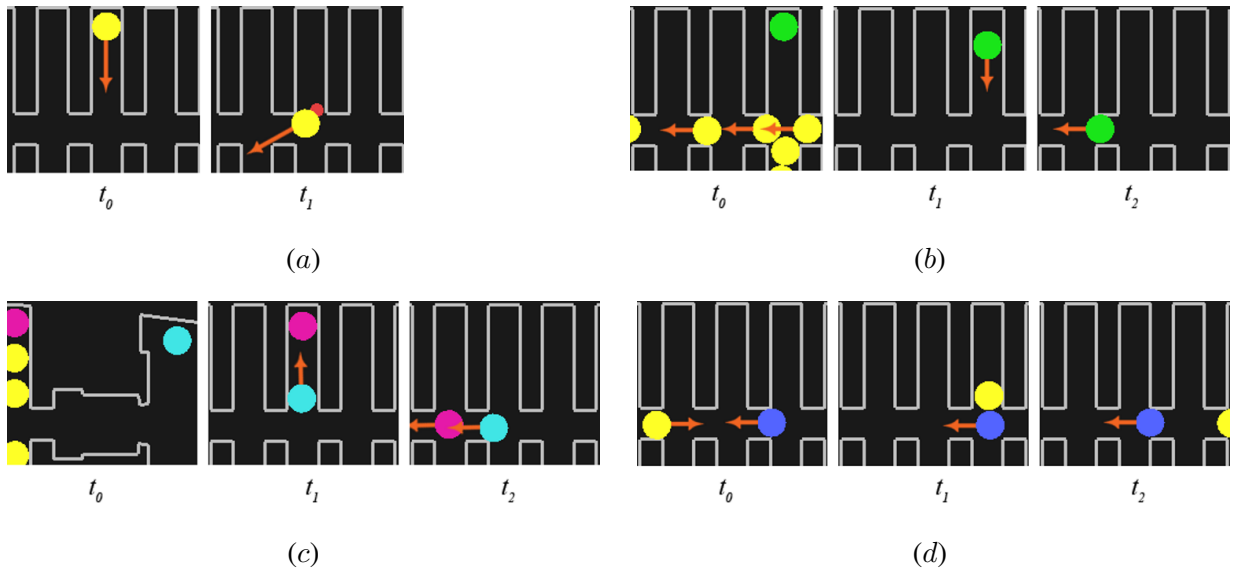


Figure 5.4: **Several behaviors displayed in Ped-Air** (shown at elapsed time t). (a) Typical agents exiting a plane leave their seat (t_0), collect their luggage from the overhead (t_1) and exit. (b) Waiting agents wait until the space around them is clear (t_0) before leaving their seat (t_1), collecting their luggage, and exiting (t_2). (c) Flight attendants (cyan) wait for the aisles to clear (t_0), then move through the plane, and assist infirm agents out of the aircraft (t_1 & t_2). (d) Particularly in loading, agents meet head-on in the aisles (t_0). Ped-Air forces one agent to yield to the other (t_1), allowing both to proceed through the aircraft (t_2).

5.4.1 Modeled Aircraft

We have applied Ped-Air to simulating single aisle and dual aisle aircraft, but its techniques can be applied to any model. Figure 5.3 illustrates the models we used, based on reconstructions of two Boeing Commercial aircraft. The space between seat rows has been increased to accommodate the circular agents typically found in agent-based simulation.

5.4.2 Experimental Timings

Federal regulations require the following distribution of passengers in a simulated evacuation: 40% must be female, 35% must be over 50 years of age, 15% must be both female and over 50 years of age. Additionally, half of all usable exits must be blocked or unused during evacuation (no more than one exit from each pair) (14 C.F.R.25 Appendix J , 2004). Using the model of physiological heterogeneity proposed by Zheng et al. (2013), we are able to produce an evacuation simulation corresponding to this distribution of passengers. Figure 5.5 illustrates the evacuation, and Table 5.2 highlights the results from our simulator. We observe that the combination of ORCA, Right of Way, and GAS unloads the simulator in under 90 seconds.

Table 5.1: **Baseline timing for unloading 158 passengers from the single aisle aircraft** (averaged over 10 iterations). Results for ORCA (Van den Berg et al., 2011), Social-Forces (Helbing and Molnár, 1995), and ORCA + Right of Way (Curtis et al., 2012). In this baseline scenario, there are no passengers who need assistance and no carry-ons. As discussed in Section 5.3.5, without symmetry-breaking techniques, both local navigation methods suffer from deadlocks and agents getting stuck. The max simulation time for this experiment was 600 seconds. Neither local navigation method was able to complete the unloading in this time frame. By contrast, with Right of Way, ORCA is able to unload the plane in 481.94 seconds.

Model	Agents unloaded at 60 seconds (t)	At 180 seconds	At 300 seconds	Total Unload Time (t)
ORCA	8	28	41	-
SF	52	52	52	-
ORCA + Right of Way	29	70	107	481.94

Table 5.2: **Baseline timing for an evacuation of 158 passengers from the single aisle aircraft** (averaged over 10 iterations). Results shown for ORCA, Social Forces, ORCA + Right of Way, and ORCA + Right of Way + GAS (Kim et al., 2012), a stress model. The max time given to these simulations was 120 seconds. Similar to the results in Table Table 5.1, without the use of symmetry-breaking techniques, we were unable to evacuate the aircraft in the full 120 seconds. With the application of Right of Way, agents are nearly able to evacuate in the time alloted by regulation. Modeling of stress in the aircraft, based on (Kim et al., 2012), generates the impetus to evacuate in under regulated time. As stress builds, agents move more quickly and are less concentrated on collision avoidance.

Model	Agents Evacuated at 30 Seconds	at 60 Seconds	at 90 Seconds	Total Evacuation Time
ORCA	20	20	20	-
SF	57	57	57	-
ORCA + Right of Way	78	125	155	92.65
ORCA + Right of Way + GAS	78	90	158	89.89

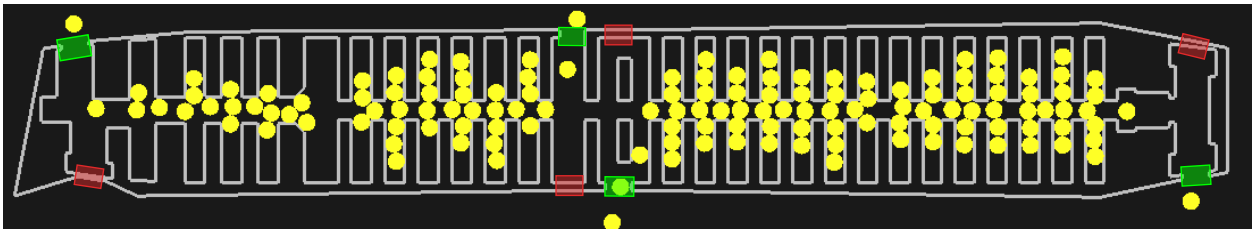


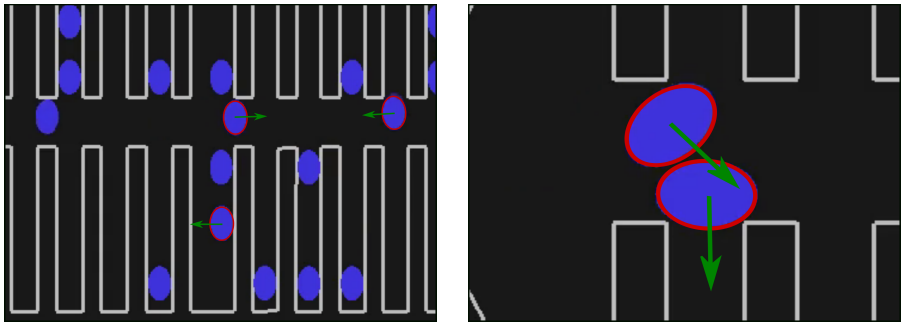
Figure 5.5: A simulation of an evacuation of the 737-800 with 50% of the exits obstructed. Available exits are highlighted in green. Obstructed or unavailable exits are highlighted in red. This evacuation simulation took 89.89 seconds in Ped-Air.

We additionally observe that agents with Right of Way but without the GAS model, (ORCA + Right of Way), initially deplane more rapidly than those experiencing stress. This creates a significant difference at the 60 second observation, 28% fewer ORCA + Right of Way + GAS agents have evacuated. We hypothesize that this difference is explained by the model of stress accumulation. We apply a linear stress accumulation function, which creates a saturation of stress between the 30 and 60 second observations. This creates sub-optimal agent behavior as the agent's navigation model changes due to stress. At the point of stress saturation, the agent's navigation stabilizes, allowing the agents to ultimately deplane more efficiently.

5.4.2.1 Data-driven Loading

We partnered with a U.S. airline to demonstrate how Ped-Air can be used to simulate loading an aircraft based on observed flight data. The airline provided anonymized data consisting of the boarding order at a per-minute fidelity for a series of flights on the single-aisle aircraft. Each seat in the aircraft was designated as a goal position and assigned a label. A goal-selector was created which leveraged a SQL database containing the flight data to assign agents goals in the proper boarding order from the aircraft.

In order to resolve seat conflicts, e.g. when an aisle seat agent boards before a window-seat agent, agents were given the ability to engage in pair-wise seat-swapping actions as described in Figure 5.2. Agents in the seated state are signaled by arriving agents, and change states to the seat-swapping state. During seat-swapping, collisions are ignored between the virtual agents, reflecting the coordination of the passengers to move for one-another. By combining the goal-selection database connector and seat-swapping behavior, we were able to successfully reproduce the loading behavior of typical flights. Figure 5.6(A) demonstrates elliptical agents boarding the aircraft in the data-driven seating order. In many cases, the aisle and window



(*a*) (*b*)

Figure 5.6: **Data-driven and elliptical agents in Ped-Air. (A)** Elliptical agents load the aircraft in actual boarding order of a real flight. The data-driven boarding approach demonstrates the pairwise seat-swapping actions triggered when agents need to pass in the narrow rows. Several agents' orientations have been highlighted in green to demonstrate the orientation constraints on elliptical agents. **(B)** Elliptical agents are able to twist their shoulders to pass in the narrow aisles. In this image, an agent twists to pass an agent in a row placing luggage. The agent orientations have been highlighted in green to demonstrate the rotation of the passing agent.

seats are loaded first. This may reflect that earlier boarding groups in the loading process typically choose their seats.

5.4.2.2 Elliptical Agent Simulation

Leveraging the work presented in Chapter 2, we produced elliptical agent simulations for aircraft loading and unloading. Utilizing elliptical agents enables the enforcement of orientation constraints. For example, agents were required to move laterally to their seats in the aircraft as is typically observed in real-world loading. In addition, the use of elliptical agents enables modeling passing behaviors in the aisles. Passengers with anti-parallel goals along the aircraft aisle were able to rotate their shoulders to pass one-another in the narrow aisle space, which better reflects observed passenger behavior. Figure 5.6(B) demonstrates an elliptical passenger rotating their shoulders to pass a passenger stowing luggage.

5.5 Conclusion

We have presented the challenges of agent-based simulation in aircraft and motivation for why agent-based simulation is a suitable method for exploring this domain. We have additionally presented Ped-Air, a framework capable of addressing these challenges and generating meaningful simulations of aircraft loading,

unloading, and evacuation. We have described the architecture of Ped-Air and highlight different passenger behaviors that can be simulated.

5.5.1 Limitations

At present, Ped-Air lacks some important behaviors and categories of passengers needed for a complete simulation. Groups, children, and passengers in a hurry are not modeled in the current simulator. In addition, our framework only admits the BFSM for behavioral modeling. We do not currently support other methods for representing passenger behavior.

5.5.2 Future Work

In addition to the constraints we placed on evacuation simulation, federal regulation requires partial aisle obstructions and four passengers to be carrying simulated infants. Further study would be required to appropriately model how the presence of infants affects behavior of the carrying adult and those around the infant. We would also like to include techniques for partial aisle obstruction.

Ped-Air is currently based on intuition and observation. Although we have demonstrated simulations based on boarding data, it would be useful to further validate its performance by statistical comparison with real-world loading and unloading data. Additionally, the air travel experience extends beyond simulations in the aircraft; the simulator provides a foundation for modeling entire airport terminals. Extending the simulations to include more aircraft and airport terminals would extend the usefulness of the simulation beyond safety and design considerations of aircraft and would provide a cohesive modeling framework for the design of efficient infrastructure, leading to and a more pleasant and safer overall travel experience.

CHAPTER 6: Conclusion and Future Work

In this dissertation, we have presented our work on modeling behavior constraints and physics-based constraints to enhance motion-planning for multi-agent systems. Our focus is on capturing the specific constraints of agents with varying shapes and behavioral contexts to generate more effective simulations and ultimately further applications such as autonomous driving, immersive virtual reality, and pedestrian simulation. We have demonstrated our approach for generating more effective models of virtual pedestrians (Chapter 2), safe, high-speed maneuvers for simulated autonomous vehicles (Chapter 3), generating plausible interactions for virtual agents (Chapter 4), and simulating behaviors aboard aircraft in loading, deplaning, and emergency evacuation (Chapter 5). Table 6.1 details several constraints modeled in this dissertation. Specifically, we have presented the following approaches:

Collision-free Trajectories for Elliptical Pedestrians: We proposed a novel model for representing virtual pedestrians as 2D ellipses. We demonstrated our algorithm's ability to generate trajectories and local behaviors which better match pedestrian data than prior approaches. We described three approaches to modeling orientation updates for elliptical agents. including a heuristic approach which enables shoulder-turning, backpedaling, and side-stepping in dense conditions. We additionally proved collision-free guarantees and offered pre-computation structures for accelerating elliptical pedestrian simulation. We demonstrated our method's ability to model commonly observed crowd phenomena and compared our model to real-world pedestrian data. In this chapter we addressed physics-based constraints including modeling the agent as an ellipse, incorporating limits in human biomechanics including shoulder-turning, and side-stepping. We addressed specific behavioral constraints such as when to utilize shoulder-turns and replicating human dense motion.

Safe Dynamic Maneuvers for Autonomous Vehicles: We proposed an optimization-based approach for generating safe, effective driving behaviors for simulated autonomous vehicles. We demonstrated a data-driven dynamics approach which allows the vehicle to maneuver safely in reactive scenarios at low or high speeds. We demonstrated our algorithm's ability to generalize to a set of vehicles with different physical profiles including steering and braking parameters. We also detailed a high-fidelity simulation framework

Table 6.1: **Modeled Constraints.** We detail the contexts, physic-based constraints, and behavior constraints modeled in this dissertation, and communication models associated with each.

Context	**Physics-based Constraints**	**Behavior Constraints**	**Communication**
Elliptical Agents	Human Bio-mechanics Collision-avoidance	Plausible Orientation Update	None
Autonomous Vehicles	Vehicle Dynamics Collision-avoidance Vehicle Kinematics Vehicle Sensing	Traffic Laws Traffic Norms Lane-changing for Passing and Avoidance	None
Interactive Agents	Attention limits	Planning Under Uncertainty Proactive Interaction Natural-language Interaction	Natural-language
Aircraft	Physiological Profiles Physical Luggage Placement Narrow Passages	Psychological Profiles Social deference Flight-staff Interaction Stress Modeling	None

for autonomous vehicle simulation which allows for testing novel driving strategies. Our framework allows simulation of various driving strategies, vehicle sensor configurations, and weather and environmental conditions. In this chapter we addressed physics-based constraints including modeling the vehicle as a tightly-fitting bounding polygon, incorporating kinematic constraints into the planner, and modeling the vehicle dynamics with a data-driven approach. We addressed specific behavioral constraints such as respecting traffic rules, traffic norms, and giving priority to vulnerable road users.

Natural-language Interactions between Virtual Agents and Avatars: We demonstrated an approach to augment motion-planning for virtual agents and enable natural-language interactions between agents and virtual avatars. Our approach enables agents to plan under uncertainty, and pro-actively seek out interaction to resolve uncertain information. We detailed the results of a user evaluation which clearly demonstrated the advantages of our approach over prior simulation methods. We additionally demonstrated advantages in replanning times using our simulated approach. In this chapter, we addressed specific behavior constraints including planning under uncertainty, interacting with agents and avatars, and modeling natural-language interaction to expedite planning.

Simulation of Loading, Unloading, and Evacuating Aircraft: We demonstrate an application of our approach to simulating behaviors aboard aircraft. We have detailed the advantages of agent-based simulation for the aircraft context and presented approaches to address the challenges of the dense, behavior-rich domain. Our approach can simulate evacuations with physiological, psychological, and obstruction distributions, and represents social relationships aboard the aircraft. We model flight-staff offering assistance, and passengers waiting in their seats for aisle clearance before deplaning. In this chapter, we specifically addressed physics-based constraints including navigating in dense-spaces, physiological variation, and agents traveling with physical luggage. We addressed specific behavior constraints including socially cooperative behaviors, social deference, and stress.

There are many avenues for future work in addition to addressing the limitations presented in each chapter. Our work seeks to address physics-based and behavior constraints in multi-agent simulation environments. There are many additional physics-based constraints that can be considered in future work. In the context of virtual pedestrians, we do not model non-linearity in velocity changes for the agents. In addition, we can explore the use of multiple shapes to represent an agent's body configuration. We additionally neglect physical contact forces, i.e. pushing, and could incorporate such forces in dense conditions such as the bidirectional flow modeled in Chapter 2.

In the context of autonomous vehicles, our data-driven dynamics model does not account for changes in the friction surface of the road. We additionally neglect wear on the vehicle and assume performance is consistent. We approximate tire forces, which are critical at high-speeds, and could incorporate a more accurate model of tire parameters to increase the accuracy of our dynamics approach.

There are many additional behavior constraints that can be modeled as well. Our virtual elliptical pedestrians neglect social conventions such as preferring the right-side of walkways or tending to pass on the left. These can be modeled as soft-constraints to guide the avoidance algorithm. We additionally do not account for social cues and signaling between virtual agents. Often, predicting the behavior of other pedestrians is an important facet of decision making. More behaviorally accurate models would incorporate prediction in the decision making.

Similarly, in the autonomous vehicle context, driver anticipation is an important behavior constraint not yet modeled by our approach. Driving is an inherently social activity, and learning and replicating the signals given by human drivers would enable to vehicle to operate more comfortably in traffic. In addition, our models of traffic-rule respect are simple and rule-based. In the future, we can incorporate more complex behavioral representations of the traffic context or data driven models. Our approach assumes perfect knowledge of the local environment. While effective for developing novel algorithms and for simulation, we need to model uncertainty as a behavior constraint for the vehicles in the future.

In the future, we can leverage captured data to improve our model heuristics. Our elliptical pedestrian and natural-language interaction approaches are modeled on pedestrian observations, but do not reflect potential benefits from directly learning from captured data. We would like to explore a purely data-driven approach to determining when the use of shoulder-turns, side-steps, and backpedaling is appropriate. By incorporating state of the art pedestrian tracking and pose-estimation, we can generate a more effective model for elliptical pedestrian local behaviors. Similarly, the attention model for our natural-language interactions is purely based on line-of-sight and distance. Humans exhibit much more complicated attention to natural-language, and we would like to leverage human conversation data to improve our virtual agent responses.

Our elliptical agent approach additionally relies on pre-computation which limits its applicability for agents of different sizes. In the future, we would like to explore simulations with elliptical agents of various sizes and eccentricities. We believe that modeling different human shapes, i.e. adults and children, separately, will provide advantages in capturing the diversity of pedestrian behaviors observed in crowds.

An important avenue for future work is expanding the behavioral tolerance of our autonomous driving algorithm. AutonoVi presently relies on a robust understanding of the behaviors of other drivers on the road. We are interested in techniques to model uncertainty in the prediction to build a more robust reaction model to other driver's behaviors. We additionally would like to explore other methods of defining the safety functions for the vehicle. Our current approach relies on repeated simulation which is limited by the fidelity of the simulator and may not generalize well to use in real vehicles. We would also seek to model more accurate representations of the vehicle's sensors for our autonomous driving framework to enable end-to-end learning for our driving conditions.

The conversational depth of our natural-language interaction is limited to basic questions and answers. Having established a general approach to modeling uncertainty in the environment, we would like to apply a conversational model to our interactions to generate more robust, context sensitive dialog. We believe that such a model would further increase the benefits of our interactive agent approach. We also would seek to model the interaction between natural-language and non-verbal communication in social scenarios. We neglect the non-verbal cues provided by humans in typical interaction scenarios. Humans rely on multi-modal interaction, and incorporating gestures, gazing, etc. would provide a more robust social framework for agent-avatar interaction.

BIBLIOGRAPHY

14 C.F.R.25 Appendix J (2004).

14 C.F.R.25.803(1990).

Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., Homm, F., Huber, W., and Kaempchen, N. (2015). Experience, Results and Lessons Learned from Automated Driving on Germany's Highways. *IEEE Intelligent Transportation Systems Magazine*, 7(1):42–57.

Alonso-Mora, J., Breitenmoser, A., Beardsley, P., and Siegwart, R. (2012). Reciprocal collision avoidance for multiple car-like robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 360–366. IEEE.

Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., and Siegwart, R. (2013). *Optimal reciprocal collision avoidance for multiple non-holonomic robots*. Springer.

Anderson, J. R., Matessa, M., and Lebiere, C. (1997). Act-r: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction*, 12(4):439–462.

Anderson, S. J., Peters, S. C., Pilutti, T. E., and Iagnemma, K. (2011). Design and development of an optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *Springer Tracts in Advanced Robotics*, 70(STAR):39–54.

Bailenson, J. N., Swinth, K., Hoyt, C., Persky, S., Dimov, A., and Blascovich, J. (2005). The independent and interactive effects of embodied-agent appearance and behavior on self-report, cognitive, and behavioral markers of copresence in immersive virtual environments. *Presence: Teleoper. Virtual Environ.*, 14(4):379–393.

Bandini, S., Federici, M., Manzoni, S., and Vizzari, G. (2006). Towards a methodology for situated cellular agent based crowd simulations. *Engineering societies in the agents world VI*, pages 203–220.

Barber, D. and Best, A. (2017). Connected and automated vehicle simulation to enhance vehicle message delivery. In *International Conference on Applied Human Factors and Ergonomics*, pages 38–46. Springer.

Bareiss, D. and Van den Berg, J. (2013). Reciprocal collision avoidance for robots with linear dynamics using lqr-obstacles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3847–3853. IEEE.

Bareiss, D. and Van den Berg, J. (2015). Generalized reciprocal collision avoidance. *The International Journal of Robotics Research*, 34(12):1501–1514.

Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., and Werneck, R. F. (2016). Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer.

Bera, A., Randhavane, T., and Manocha, D. (2017). Sociosense: Robot navigation amongst pedestrians with social and psychological constraints. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE.

Berseth, G., Kapadia, M., and Faloutsos, P. (2015). Robust space-time footsteps for agent-based steering. *Computer Animation and Virtual Worlds*.

Bisk, Y., Yuret, D., and Marcu, D. (2016). Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars. *CoRR*, abs/1604.07316.

Bonet, B. and Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, pages 52–61. AAAI Press.

Borghese, N. A., Bianchi, L., and Lacquaniti, F. (1996). Kinematic determinants of human locomotion. *The Journal of physiology*, 494(3):863–879.

Borrelli, F., Falcone, P., Keviczky, T., Asgari, J., and Hrovat, D. (2005). MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2/3/4):265.

Brenner, M. and Kruijff-Korbayová, I. (2008). A continual multiagent planning approach to situated dialogue. *Proceedings of the Semantics and Pragmatics of Dialogue (LONDIAL)*, page 61.

Brenner, M. and Nebel, B. (2009). Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331.

Brscic, D., Kanda, T., Ikeda, T., and Miyashita, T. (2013). Person tracking in large public spaces using 3-d range sensors. *Human-Machine Systems, IEEE Transactions on*, 43(6):522–534.

Buehler, M., Iagnemma, K., and Singh, S. (2009). *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer.

Burghardt, S., Klingsch, W., and Seyfried, A. (2012). Analysis of flow-influencing factors in mouths of grandstands. In *Pedestrian and Evacuation Dynamics*.

Burstedde, C., Klauck, K., Schadschneider, A., and Zittartz, J. (2001). Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295:507–525.

Cassell, J., Bickmore, T., Campbell, L., Vilhjálmsson, H., and Yan, H. (2000). Conversation as a system framework: Designing embodied conversational agents. *Embodied conversational agents*, pages 29–63.

Chao, W.-M. and Li, T.-Y. (2010). Simulation of social behaviors in virtual crowd. In *Computer Animation and Social Agents*.

Chen, B. and Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):485–497.

Choi, M. G., Kim, M., Hyun, K. L., and Lee, J. (2011). Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum*, 30(2):445–453.

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

Clodic, A., Alami, R., Montreuil, V., Li, S., Wrede, B., and Swadzba, A. (2007). A study of interaction between dialog and decision for human-robot collaborative task achievement. In *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pages 913–918. IEEE.

Curtis, S., Best, A., and Manocha, D. (2016). Menge: A modular framework for simulating crowd movement. *Collective Dynamics*, 1(0):1–40.

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*.

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

Curtis, S., Zafar, B., Gutub, A., and Manocha, D. (2012). Right of way: Asymmetric agent interactions in crowds. *The Visual Computer*, pages 1–16.

Dalibard, S., El Khoury, A., Lamiraux, F., Nakhaei, A., Taïx, M., and Laumond, J.-P. (2013). Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. *The International Journal of Robotics Research*, 32(9-10):1089–1103.

De Berg, M., Cheong, O., Van Kreveld, M., and Overmars, M. (2008). *Computational Geometry: Algorithms and Applications*. Springer, Heidelberg, 3rd edition.

Delling, D., Holzer, M., Müller, K., Schulz, F., and Wagner, D. (2009). High-performance multi-level routing. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 74:73–92.

Distner, M., Bengtsson, M., Broberg, T., and Jakobsson, L. (2009). City Safety- A System Addressing Rear-End Collisions At Low Speeds. *21st Enhanced Safety Vehicles Conference*, pages 1–7.

Doshi, F. and Roy, N. (2008). Spoken language interaction with model uncertainty: an adaptive human–robot interaction system. *Connection Science*, 20(4):299–318.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*.

Durupinar, F., Pelechano, N., Allbeck, J., Gudukbay, U., and Badler, N. (2010). The impact of the ocean personality model on the perception of crowds. *Computer Graphics and Applications, IEEE*, 31(99).

Eidehall, A., Pohl, J., Gustafsson, F., and Ekmark, J. (2007). Toward autonomous collision avoidance by steering. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):84–94.

Erdmann, M. and Lozano-Perez, T. (1987). On multiple moving objects. *Algorithmica*, 2(1-4):477–521.

Fang, Z.-M., Lv, W., Jiang, L.-X., Xu, Q.-F., and Song, W.-G. (2016). Modeling and assessment of civil aircraft evacuation based on finer-grid. *Physica A: Statistical Mechanics and its Applications*, 448:102–112.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–762.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33.

Fox, M. and Long, D. (2003). Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*.

Fritz, H., Gern, A., Schiemenz, H., and Bonnet, C. (2004). CHAUFFEUR Assistant: a driver assistance system for commercial vehicles based on fusion of advanced ACC and lane keeping. *IEEE Intelligent Vehicles Symposium, 2004*, (Vc):495–500.

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

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

Galceran, E., Cunningham, A. G., Eustice, R. M., and Olson, E. (2015a). Multipolicy Decision-Making for Autonomous Driving via Changepoint-based Behavior Prediction. *Robotics: Science and Systems*.

Galceran, E., Eustice, R. M., and Olson, E. (2015b). Toward Integrated Motion Planning and Control using Potential Fields and Torque-based Steering Actuation for Autonomous Driving. *IEEE Intelligent Vehicles Symposium*, (Iv).

Galea, E., Deere, S., Brown, R., and Filippidis, L. (2012). An evacuation validation data set for large passenger ships. *Pedestrian and Evacuation Dynamics*, pages 109–123.

Galea, E., Filippidis, L., Wang, Z., Lawrence, P., and Ewer, J. (2011). Evacuation analysis of 1000+ seat blended wing body aircraft configuration: computer simulations and full-scale evacuation experiment. *Pedestrian and Evacuation Dynamics*, pages 151–161.

Garau, M., Slater, M., Pertaub, D.-P., and Razzaque, S. (2005). The responses of people to virtual humans in an immersive virtual environment. *Presence: Teleoper. Virtual Environ.*, 14(1):104–116.

Geiger, A., Lauer, M., Moosmann, F., Ranft, B., Rapp, H., Stiller, C., and Ziegler, J. (2012). Team AnnieWAY's Entry to the 2011 Grand Cooperative Driving Challenge. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1008–1017.

Geraerts, R., Kamphuis, A., Karamouzas, I., and Overmars, M. (2008). Using the corridor map method for path planning for a large number of characters. In *Motion in Games*, pages 11–22. Springer, Heidelberg.

Gérin-Lajoie, M., Richards, C. L., and McFadyen, B. J. (2005). The negotiation of stationary and moving obstructions during walking: Anticipatory locomotor adaptations and preservation of personal space. *Motor Control*, 9:242–269.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

Giese, A., Latypov, D., and Amato, N. M. (2014). Reciprocally-rotating velocity obstacles. In *ICRA*, pages 3234–3241. IEEE.

Graesser, A. C., Li, H., and Forsyth, C. (2014). Learning by communicating in natural language with conversational agents. *Current Directions in Psychological Science*, 23(5):374–380.

Gustafsson, F. (1997). Slip-based tire-road friction estimation. *Automatica*, 33(6):1087–1099.

Guy, S. J., Chhugani, J., Kim, C., Satish, N., Lin, M. C., Manocha, D., and Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH/Erographics Symposium on Computer Animation*, pages 177–187. ACM.

Guy, S. J., Kim, S., Lin, M. C., and Manocha, D. (2011). Simulating heterogeneous crowd behaviors using personality trait theory. In *SCA '11*, pages 43–52.

Hall, L., Woods, S., Aylett, R., Newall, L., and Paiva, A. (2005). Achieving empathic engagement through affective interaction with synthetic characters. In *International Conference on Affective Computing and Intelligent Interaction*, pages 731–738. Springer.

Hardy, J. and Campbell, M. (2013). Contingency Planning Over Probabilistic Obstacle Predictions for Autonomous Road Vehicles. *IEEE Transactions on Robotics*, 29(4):913–929.

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

Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286.

Hidas, P. (2005). Modelling vehicle interactions in microscopic simulation of merging and weaving. *Transportation Research Part C: Emerging Technologies*, 13(1):37–62.

Hill Jr, R. W., Gratch, J., Marsella, S., Rickel, J., Swartout, W. R., and Traum, D. R. (2003). Virtual humans in the mission rehearsal exercise system. *Ki*, 17(4):5.

Hoffmann, G. M., Tomlin, C. J., Montemerlo, M., and Thrun, S. (2007). Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. *Proceedings of the American Control Conference*, pages 2296–2301.

Hsu, D., Kindel, R., Latombe, J., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. 21(3):233–255.

Huang, P., Kapadia, M., and Badler, N. I. (2013). Spread: sound propagation and perception for autonomous agents in dynamic environments. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 135–144. ACM.

Hughes, R., Ondrej, J., and Dingliana, J. (2014). Holonomic collision avoidance for virtual crowds. In Koltun, V. and Sifakis, E., editors, *Symposium on Computer Animation*, pages 103–111. Eurographics Association.

Imanishi, M. and Sano, T. (2014). Level of avoidance in crossing pedestrian flow. *Transportation Research Procedia*, 2(0):367 – 375. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands.

Jaillet, L. and Simeon, T. (2004). A PRM-based motion planner for dynamically changing environments. pages 1606–1611.

Johansson, A., Helbing, D., and Shukla, P. K. (2007). Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems*, 10:271–288.

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., and Vasudevan, R. (2017). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *IEEE International Conference on Robotics and Automation*, pages 1–8.

Kapadia, M., Singh, S., Hewlett, W., and Faloutsos, P. (2009). Egocentric affordance fields in pedestrian steering. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 215–223.

Karamouzas, I. and Guy, S. J. (2015). Prioritized group navigation with formation velocity obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Karamouzas, I., Heil, P., van Beek, P., and Overmars, M. H. (2009). A predictive collision avoidance model for pedestrian simulation. In *Motion in Games*, volume 5884 of *Lecture Notes in Computer Science*, pages 41–52. Springer.

Karamouzas, I., Skinner, B., and Guy, S. J. (2014). Universal power law governing pedestrian interactions. *Physical Review Letters*, 113(23):238701.

Katrakazas, C., Quddus, M., Chen, W.-H., and Deka, L. (2015). Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5:90–98.

Kim, S., Bera, A., Best, A., Chabra, R., and Manocha, D. (2016). Interactive and adaptive data-driven crowd simulation. *Proc. of IEEE VR*.

Kim, S., Guy, S. J., and Manocha, D. (2013). Velocity-based modeling of physical interactions in multi-agent simulations. In *Symposium on Computer Animation*, pages 125–133.

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

Kim, Y., Baylor, A. L., Group, P., et al. (2006). Pedagogical agents as learning companions: The role of agent competency and type of interaction. *Educational Technology Research and Development*, 54(3):223–243.

Kolski, S., Ferguson, D., Bellino, M., and Siegwart, R. (2006). Autonomous Driving in Structured and Unstructured Environments. In *2006 IEEE Intelligent Vehicles Symposium*, pages 558–563. IEEE.

Kopp, S., Gesellensetter, L., Krämer, N., and Wachsmuth, I. (2005). A Conversational Agent as Museum Guide – Design and Evaluation of a Real-World Application. pages 329–343.

Krausz, B. and Bauckhage, C. (2012). Loveparade 2010 : Automatic video analysis of a crowd disaster. *Computer Vision and Image Understanding*, 116(3):307–319.

Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE.

Kullu, K., Güdükbay, U., and Manocha, D. (2017). Acmics: an agent communication model for interacting crowd simulation. *Autonomous Agents and Multi-Agent Systems*, pages 1–21.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J. P. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118.

Lalish, E. and Morgansen, K. A. (2012). Distributed reactive collision avoidance. *Autonomous Robots*, 32(3):207–226.

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

Latombe, J.-C. (1991). *Robot Motion Planning*. Springer, Heidelberg.

Laumond, J.-p., Sekhavat, S., and Lamiraux, F. (1998). Guidelines in nonholonomic motion planning for mobile robots. In *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, New York, NY, USA.

LaValle, S. M., Hutchinson, S., et al. (1998). Optimal motion planning for multiple robots having independent goals. *Robotics and Automation, IEEE Transactions on*, 14(6):912–925.

Lee, I.-K., Kim, M.-S., and Elber, G. (1998). Polynomial/rational approximation of minkowski sum boundary curves. *Graphical Models and Image Processing*, 60(2):136 – 165.

Lee, K. H., Choi, M. G., Hong, Q., and Lee, J. (2007). Group behavior from video: a data-driven approach to crowd simulation. In *Symposium on Computer Animation*, pages 109–118.

Lerner, A., Chrysanthou, Y., and Lischinski, D. (2007). Crowds by example. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3).

Liao, Q. V., Davis, M., Geyer, W., Muller, M., and Shami, N. S. (2016). What can you do?: Studying social-agent orientation and agent proactive interactions with an agent for employees. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, pages 264–275. ACM.

Likhachev, M. and Ferguson, D. (2009). Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. *The International Journal of Robotics Research*, 28(8):933–945.

Lindsay, A., Read, J., Ferreira, J. F., Hayton, T., Porteous, J., and Gregory, P. J. (2017). Framer: Planning models from natural language action descriptions. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS 2017)*.

Liu, S. B., Lo, S. M., and Ma, J. (2012). On the simulation for rail tunnel evacuation with cross-passageways. *Pedestrian and Evacuation Dynamics*, pages 425–433.

Love Parade (2010). Dokumentation der Ereignisse zur Loveparade 2010 in Duisburg.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Margolis, D. L. and Asgari, J. (1991). Multipurpose models of vehicle dynamics for controller design. In *SAE Technical Paper*. SAE International.

Martinez-Gomez, L. and Fraichard, T. (2009). Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 100–105.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language.

Mendes, C. C. T., Frémont, V., and Wolf, D. F. (2016). Exploiting fully convolutional neural networks for fast road detection. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3174–3179. IEEE.

Mordatch, I. and Abbeel, P. (2017). Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

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

Narang, S., Best, A., Curtis, S., and Manocha, D. (2015). Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. *PLoS ONE*, 10(4):1–17.

Narang, S., Best, A., and Manocha, D. (2018). Simulating movement interactions between avatars & agents in virtual worlds using human motion constraints. *Proc. of IEEE VR*.

Narang, S., Best, A., Randhavane, T., Shapiro, A., and Manocha, D. (2016). Pedvr: Simulating gaze-based interactions between a real user and virtual crowds. In *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, VRST '16, pages 91–100.

Nass, C., Isbister, K., Lee, E.-J., et al. (2000). Truth is beauty: Researching embodied conversational agents. *Embodied conversational agents*, pages 374–402.

Nass, C., Steuer, J., and Tauber, E. R. (1994). Computers are social actors. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 72–78. ACM.

Nilsson, D. and Johansson, A. (2009). Social influence during the initial phase of a fire evacuation—analysis of evacuation experiments in a cinema theatre. *Fire Safety Journal*, 44(1):71–79.

Novielli, N., de Rosis, F., and Mazzotta, I. (2010). User attitude towards an embodied conversational agent: Effects of the interaction mode. *Journal of Pragmatics*, 42(9):2385–2397.

Ondřej, J., Pettré, J., Olivier, A.-H., and Donikian, S. (2010). A synthetic-vision based steering approach for crowd simulation. In *Proc. SIGGRAPH*, pages 123:1–123:9.

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

Paris, S., Pettre, J., and Donikian, S. (2007). Pedestrian reactive navigation for crowd simulation: a predictive approach. 26.

Park, C., Pan, J., and Manocha, D. (2012). Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In *ICAPS*.

Patil, S., Van den Berg, J., Curtis, S., Lin, M., and Manocha, D. (2010). Directing crowd simulations using navigation fields. *IEEE TVCG*, pages 244–254.

Pauls, J. L. (2004). Suggestions on evacuation models and research questions. In *Conference Proceedings of the 3rd International Symposium on Human Behaviour in Fire*.

Pednault, E. P. (1989). Adl: Exploring the middle ground between strips and the situation calculus. *Kr*, 89:324–332.

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

Pelechano, N., O'Brien, K., Silverman, B., and Badler, N. (2005). Crowd simulation incorporating agent psychological models, roles and communication. *First International Workshop on Crowd Simulation*.

Pendleton, S., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y., Rus, D., and Ang, M. (2017). Perception, Planning, Control, and Coordination for Autonomous Vehicles. *Machines*, 5(1):6.

Pertaub, D.-P., Slater, M., and Barker, C. (2002). An experiment on public speaking anxiety in response to three different types of virtual audience. *Presence: Teleoperators and virtual environments*, 11(1):68–78.

Peters, S. C., Frazzoli, E., and Iagnemma, K. (2011). Differential flatness of a front-steered vehicle with tire force control. *IEEE International Conference on Intelligent Robots and Systems*, 02139:298–304.

Petrick, R. P. and Bacchus, F. (2004). Extending the knowledge-based approach to planning with incomplete information and sensing. In *ICAPS*, pages 2–11.

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 *Symposium on Computer Animation*, pages 189–198.

Petty, S. and Fraichard, T. (2005). Safe motion planning in dynamic environments. In *IROS*, pages 3726–3731.

Rao, A. S. and Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. *Proceedings of the First International Conference on Multiagent Systems*, 95:312–319.

Rasa.ai (2017). Language understanding with rasa nlu.

Reeds, J. and Shepp, L. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393.

Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proc. of SIGGRAPH*.

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

Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016). *Playing for Data: Ground Truth from Computer Games*, pages 102–118. Springer International Publishing, Cham.

Rickel, J. and Johnson, W. L. (1999). Virtual humans for team training in virtual reality. In *Proceedings of the ninth international conference on artificial intelligence in education*, volume 578, page 585.

Rothbaum, B. O., Hodges, L. F., Kooper, R., Opdyke, D., Williford, J. S., and North, M. (1995). Virtual reality graded exposure in the treatment of acrophobia: A case report. *Behavior therapy*, 26(3):547–554.

Rothbaum, B. O., Hodges, L. F., Ready, D., and Alarcon, R. D. (2001). Virtual reality exposure therapy for vietnam veterans with posttraumatic stress disorder. *The Journal of clinical psychiatry*, 62(8):1–478.

Rucco, A., Notarstefano, G., and Hauser, J. (2015). An efficient minimum-time trajectory generation strategy for two-track car vehicles. *IEEE Transactions on Control Systems Technology*, 23(4):1505–1519.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Sadigh, D., Sastry, S., Seshia, S. A., and Dragan, A. D. (2016). Planning for Autonomous Cars that Leverage Effects on Human Actions. *Proceedings of Robotics: Science and Systems*.

Saifuzzaman, M. and Zheng, Z. (2014). Incorporating human-factors in car-following models: A review of recent developments and research needs. *Transportation Research Part C: Emerging Technologies*, 48:379–403.

Sanchez, G. and Latombe, J.-C. (2002). Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 2112–2119. IEEE.

Schadschneider, A. (2002). Cellular automaton approach to pedestrian dynamics - theory. *Pedestrian and Evacuation Dynamics*, pages 75–86.

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

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

Shoham, Y. (1993). Agent-oriented programming. *Artificial intelligence*, 60(1):51–92.

Singh, K. B. and Taheri, S. (2015). Estimation of tire-road friction coefficient and its application in chassis control systems. *Systems Science & Control Engineering*, 3(1):39–61.

Singh, S., Kapadia, M., Reinman, G., and Faloutsos, P. (2011). Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds*, 22(2-3):151–158.

Slater, M., Lotto, B., Arnold, M. M., and Sanchez-Vives, M. V. (2009). How we experience immersive virtual environments: the concept of presence and its measurement. *Anuario de psicología*, 40(2).

Snook, G. (2000). Simplified 3d movement and pathfinding using navigation meshes. In *Game Programming Gems*, chapter 3, pages 288–304. Charles River, Hingham, Mass.

Sun, H., Deng, W., Zhang, S., Wang, S., and Zhang, Y. (2014). Trajectory planning for vehicle autonomous driving with uncertainties. *ICCSS 2014 - Proceedings: 2014 International Conference on Informative and Cybernetics for Computational Social Systems*, pages 34–38.

Sun, L., Shoulson, A., Huang, P., Nelson, N., Qin, W., Nenkova, A., and Badler, N. I. (2012). Animating synthetic dyadic conversations with variations based on context and agent attributes. *Computer Animation and Virtual Worlds*, 23(1):17–32.

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

Švestka, P. and Overmars, M. H. (1998). Coordinated path planning for multiple robots. *Robotics and autonomous systems*, 23(3):125–152.

Tellex, S., Knepper, R. A., Li, A., Rus, D., and Roy, N. (2014). Asking for help using inverse semantics. In *Robotics: Science and systems*, volume 2.

Templer, J. (1995). *The Staircase: Studies of Hazards, Falls, and Safer Design*. MIT Press.

Tenorth, M. and Beetz, M. (2017). Representations for robot knowledge in the knowrob framework. *Artificial Intelligence*, 247:151–169.

Thomason, J., Zhang, S., Mooney, R., and Stone, P. (2015). Learning to interpret natural language commands through human-robot dialog. *IJCAI International Joint Conference on Artificial Intelligence*, 2015-Janua(Ijcai):1923–1929.

Treiber, M., Kesting, A., and Helbing, D. (2006). Delays, inaccuracies and anticipation in microscopic traffic models. *Physica A: Statistical Mechanics and its Applications*, 360(1):71–88.

Treuille, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. In *Proc. of ACM SIGGRAPH*, pages 1160–1168.

Treuille, A., Lee, Y., and Popović, Z. (2007). Near-optimal character animation with continuous control. *ACM Trans. Graph.*, 26(3).

Tumova, J., Hall, G. C., Karaman, S., Frazzoli, E., and Rus, D. (2013). Least-violating control strategy synthesis with safety rules. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 1–10. ACM.

Turri, V., Carvalho, A., Tseng, H. E., Johansson, K. H., and Borrelli, F. (2013). Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, (Itsc):378–383.

Ulicny, B. and Thalmann, D. (2002). Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum*, 21(4):767–775.

van Basten, B. J. H., Stuvel, S. A., and Egges, A. (2011). A hybrid interpolation scheme for footprint-driven walking synthesis. *Graphics Interface*, pages 9–16.

Van den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Inter. Symp. on Robotics Research*, 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 multiple agents in crowded environments. In *Proc. Symp. Interact. 3D Graph. Game.*, pages 139–147.

Van Den Berg, J., Snape, J., Guy, S. J., and Manocha, D. (2011). Reciprocal collision avoidance with acceleration-velocity obstacles. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3475–3482.

van Toll, W. G., Cook, A. F., and Geraerts, R. (2012). Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds*, 23(1):59–69.

Wang, S. I., Liang, P., and Manning, C. D. (2016). Learning Language Games through Interaction. *The 54th Annual Meeting of the Association for Computational Linguistics*, pages 2368–2378.

Weidmann, U. (1993). Transporttechnik der fussgaenger. Technical Report 90.

Yan, Y. and Chirikjian, G. S. (2014). Closed-form characterization of the minkowski sum and difference of two ellipsoids. *Geometriae Dedicata*, pages 1–26.

Yeh, H., Curtis, S., Patil, S., Van den Berg, J., Manocha, D., and Lin, M. (2008). Composite agents. *Proc. of SCA*, pages 39–47.

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.

Zanlungo, F., Ikeda, T., and Kanda, T. (2011). Social force model with explicit collision prediction. *Europhysics Letters*, 93:68005.

Zhang, J., Klingsch, W., Schadschneider, A., and Seyfried, A. (2012). Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. *J. Stat. Mech.*, 2012(02):P02002.

Zheng, L., Wang, L., Liu, L., and Liu, X. (2013). Heterogeneous crowd behaviors simulation: a physiological perspective. In Lu, K., Mei, T., and Wu, X., editors, *ICIMCS*, pages 195–198. ACM.

Ziegler, J., Bender, P., Dang, T., and Stiller, C. (2014a). Trajectory planning for Bertha - A local, continuous method. *The International Journal of Robotics Research*, 35(April):450–457.

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C. G., et al. (2014b). Making bertha drive-an autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20.

Ziegler, J., Werling, M., and Schröder, J. (2008). Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 787–791.