SMOOTH AND COLLISION-FREE NAVIGATION FOR MULTIPLE MOBILE ROBOTS AND VIDEO GAME CHARACTERS

Jamie R. Snape

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

Chapel Hill
2012

Approved by:

Dinesh Manocha

Ming C. Lin

Ron Alterovitz

Jan-Michael Frahm

Marc Niethammer

ABSTRACT

JAMIE R. SNAPE: Smooth and Collision-Free Navigation for Multiple Mobile Robots and Video Game Characters
(Under the direction of Dinesh Manocha)

The navigation of multiple mobile robots or virtual agents through environments containing static and dynamic obstacles to specified goal locations is an important problem in mobile robotics, many video games, and simulated environments. Moreover, technological advances in mobile robot hardware and video games consoles have allowed increasing numbers of mobile robots or virtual agents to navigate shared environments simultaneously. However, coordinating the navigation of large groups of mobile robots or virtual agents remains a difficult task. Kinematic and dynamic constraints and the effects of sensor and actuator uncertainty exaggerate the challenge of navigating multiple physical mobile robots, and video games players demand plausible motion and an ever increasing visual fidelity of virtual agents without sacrificing frame rate.

We present new methods for navigating multiple mobile robots or virtual agents through shared environments, each using formulations based on velocity obstacles. These include algorithms that allow navigation through environments in two-dimensional or three-dimensional workspaces containing both static and dynamic obstacles without collisions or oscillations. Each mobile robot or virtual agent senses its surroundings and acts independently, without central coordination or inter-communication with its neighbors, implicitly assuming the neighbors use the same navigation strategy based on the notion of reciprocity. We use the position, velocity, and physical extent of neighboring mobile robots or virtual agents to compute their future trajectories to avoid collisions locally and show that, in principle, it is possible to theoretically guarantee that the motion of each mobile robot or virtual agent is smooth. Moreover, we demonstrate direct, collision-free, and oscillation-free navigation in experiments using physical iRobot Create mobile robots, simulations of multiple differential-drive robots or simple-airplanes, and video games levels containing hundreds of virtual agents.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF ALGORITHMS

Algorithm

Chapter 1

INTRODUCTION

1.1 Motivation

Mobile robots are becoming increasingly common in everyday life. Applications have been as diverse as inspecting walls (Longo and Muscato, 2006), warehousing (Fiorini and Botturi, 2008), and acting as tour guides (Philippsen and Siegwart, 2003). Numerous mobile robots, such as unmanned aerial vehicles and autonomous underwater vehicles, have been deployed in military applications (Cheng, Kumar, Arkin, *et al*., 2009). Terrestrial mobile robots, moving in the two-dimensional workspace, often have differential-drive constraints; a simple drive mechanism that consists of two independently actuated drive wheels mounted on a common axis. From vacuum cleaners (Jones, Mack, Nugent, *et al*., 2005) to wheelchairs (Prassler, Scholz, and Fiorini, 1999), most mobile robots for domestic applications have differential-drive constraints. Aerial mobile robots, in the three-dimensional workspace, are also becoming more widely used. In addition to their deployment by the military, these robots, which are often propeller-driven with fixed wings, and may be as large as light aircraft, have seen civil use assisting humans in the localization of wildfires (Casbeer, Kingston, Beard, *et al*., 2006), as well as patrolling national borders (Girard, Howell, and Hedrick, 2004).

Mobile robots are beginning to be used as parts of a distributed system of multiple robots. Groups of coordinated mobile robots may be used for surveillance, environmental monitoring, and search and rescue (Michael, Fink, and Kumar, 2008). As their cost has decreased, multiple autonomous airplanes and unmanned aerial vehicles have been used for tracking dynamic targets or providing three-dimensional coverage (Cheng, Kumar, Arkin, *et al*., 2009; Grzonka, Grisetti, and Burgard, 2009). When multiple mobile robots share an environment, it is necessary to develop methods to compute collision-free paths for each of these robots with respect to other robots and dynamic and static obstacles. Moreover, the robot should move smoothly, without oscillations in velocity (Van den Berg, Lin, and Manocha, 2008). Smoothness is important for mobile robots since their motion must account for the physical limits of their actuators and other safety issues.

The same techniques are applicable to video games containing many virtual agents. Pathfinding is an important component of such games, and improved hardware, in particular the utilization of multi-core architectures in recent computers and consoles, has made available the computational resources to simulate large numbers of virtual agents with improved visual fidelity (Reynolds, 2006). As a result, there is an ever-increasing demand for effective and efficient techniques to generate plausible motion for such groups of virtual agents automatically.

1.2 Global Path Planning in Static Environments

The path planning problem for a single mobile robot or virtual agent in an environment containing static obstacles is to compute a path through the environment, from a given starting point to a given goal point, such that the mobile robot or virtual agent is never in collision with a static obstacle, that respects any additional constraints imposed on the motion of the mobile robot or virtual agent due, for instance, to its kinematics. A planning algorithm is said to be "complete" if it returns a path if one exists and returns that no such path exists if it does not. There is a wide variety of approaches to the path planning of a single mobile robot or virtual agent, complete or otherwise, but the most prevalent are, broadly, cell decomposition, potential fields, and roadmaps.

Cell Decomposition Global Planners. These algorithms divide the environment into cells such that a path always exists between points within the same cell (Brooks and Lozano-Pérez, 1985; Schwartz and Sharir, 1983). A connectivity graph between cells is generated, and a graph search, such as A* (Hart, Nilsson, and Raphael, 1968), is performed to compute a sequence of cells that will lead the mobile robot or virtual agent from the cell containing the starting point to the cell containing the goal point. This decomposition may take the form of a navigation mesh (Snook, 2000) or corridor map (Geraerts, Kamphuis, Karamouzas, *et al.*, 2008), which are popular methods for path planning in video games.

Potential Field Global Planners. These algorithms direct the mobile robot or virtual agent using potential functions (Khatib, 1986) that are defined over the environment. The potential functions are usually the sum of an attractive potential at the goal point and repulsive potentials at each obstacle in the environment. Often the potential function can be sampled locally rather than being precomputed for the whole of the environment. It follows that potential field planners often have low computation costs, but their disadvantage is that the mobile robot or virtual agent may reach a local minimum in the potential field and may need to resort to a random walk, or similar technique, to escape that point.

Roadmap Global Planners. These algorithms represent the connectivity of discrete points in the environment of the mobile robot or virtual agent as a roadmap (Canny, 1988; Lozano-Pérez and Wesley, 1979; Ó'Dúnlaing and Yap, 1985). Points in the environment are sampled, and the samples between which a collision-free path exists are connected. A graph search then calculates the sequence of roadmap nodes that must be traversed to move from the starting point to the goal point. Popular approaches that use random sampling of points include the probabilistic roadmap method (Amato and Wu, 1996; Kavraki, Švestka, Latombe, *et al.*, 1996) and rapidly-exploring random trees (LaValle and Kuffner, 2001).

1.3 Global Path Planning in Shared Environments

For multiple mobile robots or virtual agents, the path planning problem is expanded to compute paths from the starting point to the goal point for every mobile robot or virtual agent sharing an

environment such that no mobile robot or virtual agent is in collision with another mobile robot or virtual agent or a static obstacle at any time. Approaches to planning paths of multiple mobile robots or virtual agents are often categorized as either "centralized" or "decoupled," although there are hybrid approaches that combine some elements of both centralized and decoupled planners, and may add improved scalability to a centralized planner or completeness to a decoupled planner.

Centralized Global Planners. These algorithms compute a single path in a composite workspace in a higher dimension, effectively treating the mobile robots or virtual agents as a single, composite mobile robot or virtual agent. The approaches commonly used for the planning of a single mobile robot or virtual agent can then be directly applied to the planning of multiple mobile robots or virtual agents. It follows that centralized planners may be complete, yet the resulting, composite workspace may be of such high dimension as to make their use computationally impractical.

Decoupled Global Planners. These algorithms compute paths for each robot individually and independently, and then perform a velocity-tuning step or some coordination scheme to ensure that no collisions occur along each path. Decoupled planners are not, in general, complete, but they are faster, as planning in the individual workspaces of each mobile robot or virtual agent is usually easier. Prioritized planners plan for each mobile robot or virtual agent in some order of priority while ensuring that each mobile robot or virtual agent does not collide with mobile robots or virtual agents for which paths have previously been planned.

1.4 Local Collision Avoidance in Dynamic Environments

Instead of considering the global environment of each mobile robot or virtual agent and forming a complete path from the starting point to the goal point in the environment, local collision avoidance computes a trajectory for each mobile robot or virtual agent based on local observations of a dynamic environment such that the mobile robot or virtual agent is not in collision with other mobile robots or virtual agents and obstacles within some short, future window of time. The trajectories of the mobile robots or virtual agents are updated repeatedly as they progress through the environment. Early work was termed the "asteroid avoidance problem" (Reif and Sharir, 1985). Collision avoidance approaches are particularly well suited to navigating multiple mobile robots or virtual agents since the mobile robots or virtual agents must react to the unpredictable actions of other mobile robots or virtual agents and dynamic obstacles. Their computational costs are also often much lower than those of traditional path planners. However, due to their local nature, these planners are not complete. Many approaches to collision avoidance have been proposed, but the most popular are force-based and velocity-based.

Force-based Local Collision Avoidance. In video games and simulated environments, the prevalent approach has been to model virtual agents as particle systems (Helbing and Molnár, 1995) with each particle applying a force on nearby particles. The laws of physics are used to compute forces on and

the motion of the particles along with a set of specified behaviors that influence properties of the system, such as separation, alignment, and cohesion of the virtual agents as particles (Reynolds, 1987).

Velocity-based Local Collision Avoidance. Popularized in robotics (Van den Berg, Lin, and Manocha, 2008; Chakravarthy and Ghose, 1998; Fiorini and Shiller, 1998), these methods use the current position and velocity of each mobile robot or virtual agent to estimate its future trajectory for some short window of time and compute a new velocity that will be free of collisions over some short time interval. Often, the new velocity for each mobile robot or virtual agent may be computed independently without explicit communication, allowing for distributed systems of mobile robots or the parallelization of simulations of virtual agents on modern multi-core architectures.

1.5 Kinematic Constraints

Mobile robots and some virtual agents have kinematic constraints on their motion. Some of the earliest work in robotics concerned optimal paths for mobile robots with the kinematic constraints of a simplified car (Dubins, 1957). Due to their widespread applicability to physical robots, much research has focused on mobile robots with car-like constraints or those of a differential-drive robot.

Simple Car Kinematic Constraints. The control inputs of a simple car (Latombe, 1991; Laumond, Sekhavat, and Lamiraux, 1998; LaValle, 2006) are its varying speed and steering angle. Simple cars have four wheels that are attached to two axles separated by some nonzero, fixed wheelbase. The two wheels that are attached to the rear axle of a simple car are its drive wheels, and both wheels are always driven at the same speed. The front axle and its two wheels may be turned by some bounded steering angle. The motion of a simple car in the two-dimensional workspace is constrained by its speed, steering angle, and wheelbase. Aerial robots in the three-dimensional workspace have been modeled as simple cars with varying altitude (Chitsaz and LaValle, 2007).

Differential-Drive Kinematic Constraints. The individual wheel speeds are the control inputs of a differential-drive robot. Differential-drive robots have two fixed drive wheels, separated by a nonzero, fixed wheel track, that may be driven independently at different speeds. A differential-drive robot turns by varying the differential speed of its wheels. If the right wheel is driven at a greater speed than the left, a differential-drive robot will turn left, and vice versa. The motion of a differential-drive robot in the two-dimensional workspace is constrained by its two wheel speeds and its wheel track. The center of a differential-drive robot can follow any continuous path within the environment (LaValle, 2006), but not instantaneously since it must spin to reach a direction perpendicular to the direction of its wheels.

1.6 Thesis Statement and Contributions

Our thesis statement is as follows.

Multiple mobile robots or virtual agents with kinematic and dynamic constraints may navigate through shared environments in two-dimensional or three-dimensional workspaces containing static and dynamic obstacles without collisions or oscillations and without central coordination using formulations based on velocity obstacles.

In the remainder of this dissertation, we make the following main contributions in support of the thesis statement.

Independent Navigation of Multiple Mobile Robots or Virtual Agents with Hybrid Reciprocal Velocity Obstacles. In Chapter 2, we present the "hybrid reciprocal velocity obstacle" for oscillation-free navigation of multiple mobile robots or virtual agents in the two-dimensional workspace. Each mobile robot or virtual agent senses its surroundings and acts independently without central coordination or communication with other robots or virtual agents. Our approach uses both the current position and the velocity of other mobile robots or virtual agents to compute their future trajectories to avoid collisions locally. Our approach is reciprocal (Van den Berg, Lin, and Manocha, 2008), avoiding oscillations in velocity by explicitly taking into account that the other mobile robots or virtual agents in the environment equally sense their surroundings and change their trajectories accordingly. We apply hybrid reciprocal velocity obstacles to multiple iRobot Create mobile robots and virtual agents and demonstrate direct and oscillation-free navigation even in the presence of dynamic or static obstacles.

Smooth Navigation of Multiple Robots under Differential-Drive Constraints. In Chapter 3, we present a method for the smooth navigation of multiple independent differential-drive robots in the two-dimensional workspace. We adapt the optimal reciprocal collision avoidance formulation (Van den Berg, Guy, Lin, *et al.*, 2011) by enlarging the "effective radius" by which it bounds a differential-drive robot in a precise way to provide a reference point, the "effective center," that can be maneuvered in any direction instantaneously. Our approach theoretically guarantees both the smoothness of the trajectories of the differential-drive robots and paths that are locally free of collisions. We provide proofs of these guarantees, including, to the best of our knowledge, the first proof of smoothness for a reciprocal collision avoidance scheme for mobile robots or virtual agents. We show the effectiveness of our approach in experiments with multiple iRobot Create differential-drive robots, demonstrating visibly smooth motion.

Navigating Multiple Simple-Airplanes in Three-Dimensional Workspace. In Chapter 4, we present optimal reciprocal collision avoidance with "variable reciprocity" for navigating multiple simple-airplanes in the three-dimensional workspace. Our approach extends the kinematic and dynamic constraints of a simple car (Latombe, 1991; Laumond, Sekhavat, and Lamiraux, 1998; LaValle, 2006) to a model, which we name the "simple-airplane," that has constraints on both speed, steering angle, and changes in altitude. We use a reciprocal collision avoidance method that computes the trajectory of each simple-airplane independently without collisions with other simple-airplanes or dynamic

obstacles and without oscillations in velocity. Moreover, we use a sampling-based scheme to ensure the kinematic and dynamic constraints of each simple-airplane are satisfied. Generalizing the notion of reciprocity, we extend previous approaches (Van den Berg, Guy, Lin, *et al.*, 2011; Van den Berg, Lin, and Manocha, 2008) by explicitly considering the recent motion of nearby simple-airplanes to vary the level of reciprocity between pairs of simple-airplanes dynamically so that a simple-airplane that is observed to be more constrained needs to take less responsibility for avoiding collisions. We test our approach in experiments with simulated simple-airplanes and dynamic obstacles, demonstrating oscillation-free trajectories that satisfy the kinematic and dynamic constraints of each simple-airplane.

Goal Velocity Obstacles for Spatial Navigation of Multiple Virtual Agents. In Chapter 5, we present the "goal velocity obstacle" for the spatial navigation of multiple virtual agents, such as are found in video games and simulated environments, to planar goal regions in the two-dimensional workspace. Our approach uses the notion of velocity obstacles (Fiorini and Shiller, 1998) not only to compute collision-avoiding velocities with respect to other virtual agents, but also to specify velocities that will direct a virtual agent toward its spatial goal region. The goal velocity obstacle provides a unified formulation that allows for goals specified as points, line segments, and bounded, planar regions in two dimensions that may be static or moving. A virtual agent may have multiple goal regions without requiring an explicit goal allocation algorithm that would choose a particular goal region to navigate toward in advance. We have applied our approach to experiments with hundreds of virtual agents, demonstrating shorter path lengths and fewer collisions with only microseconds of additional computation, per virtual agent, per time step, than when using velocity-based methods that optimize on a single, preferred velocity toward the goal of each virtual agent.

Chapter 2

INDEPENDENT NAVIGATION OF MULTIPLE MOBILE ROBOTS OR VIRTUAL AGENTS WITH HYBRID RECIPROCAL VELOCITY OBSTACLES

2.1 Introduction and Motivation

Many recent works have considered the problem of navigating a mobile robot or virtual agent in an environment composed of dynamic obstacles (Chakravarthy and Ghose, 1998; Fiorini and Shiller, 1998; Fox, Burgard, and Thrun, 1997; Hsu, Kindel, Latombe, *et al.*, 2002; Petti and Fraichard, 2005). Some of the simplest approaches predict where the dynamic obstacles may be in the future by extrapolating their current velocities and let the mobile robot or virtual agent avoid collisions accordingly. However, such techniques are not sufficient when a robot encounters other robots because treating the other robots as dynamic obstacles overlooks the reciprocity between robots. In other words, the other robots are not passive, but are actively trying to avoid collisions. Therefore, the future trajectories of other robots cannot be estimated by simply extrapolating their current velocities since doing so would inherently cause undesirable oscillations in their trajectories (Van den Berg, Lin, and Manocha, 2008).

We present the "hybrid reciprocal velocity obstacle" for navigating multiple mobile robots or virtual agents that explicitly considers the reciprocity between the mobile robots or virtual agents. Informally, reciprocity lets a robot take half of the responsibility for avoiding collisions with another robot and assumes that the other robot takes responsibility for the other half. In an environment containing multiple robots, this concept extends to every pair of robots. Each robot executes an independent feedback loop in which it chooses its new velocity based on observations of the current positions and velocities of the other robots in proximity. The robots do not communicate with each other, but implicitly assume that the other robots use the same navigation strategy based on reciprocity. Our overall approach can also deal with both static and dynamic obstacles using a navigation roadmap.

The hybrid reciprocal velocity obstacle is an extension of the reciprocal velocity obstacle (Van den Berg, Lin, and Manocha, 2008) that was introduced to address similar issues in multiagent simulation. However, the reciprocal velocity obstacle formulation has some limitations, particularly that it frequently causes virtual agents to end up in a "reciprocal dance" (Feurtey, 2000) as they cannot reach agreement on which side to pass each other. To overcome this drawback, the hybrid reciprocal velocity obstacle enlarges the reciprocal velocity obstacle on the side that a robot should not pass by substituting the reciprocal velocity obstacle edge with the edge of a velocity obstacle (Fiorini and Shiller, 1998). Consequently, if a robot attempts to pass on the wrong side of another robot, then the

robot has to give full priority to the other robot. If the robot chooses the correct side, then it can assume the cooperation of the other robot and retains equal priority.

We have implemented and applied our approach to a set of iRobot Create mobile robots moving in an indoor environment using Bluetooth wireless remote control and centralized sensing from an overhead digital video camera. Our experiments show that our approach achieves oscillation-free navigation in an environment containing multiple mobile robots and dynamic obstacles even with some uncertainty in position and velocity. We also demonstrate the ability to handle static obstacles and the low computational requirements and scalability of the hybrid reciprocal velocity obstacle in simulations of multiple virtual agents.

2.2 Prior Work

Reactive navigation differs from traditional global path planning approaches, *e.g.*, Hsu, Kindel, Latombe, *et al.* (2002); Kavraki, Švestka, Latombe, *et al.* (1996); LaValle and Kuffner (2001), in that rather than planning complete paths to their goals, robots react only to their local environment at any moment in time. Well-known reactive formulations include the dynamic window approach (Fox, Burgard, and Thrun, 1997) and inevitable collision states (Petti and Fraichard, 2005), in addition to velocity obstacles (Fiorini and Shiller, 1998). Some approaches use a number of predefined discrete behaviors (Pallottino, Scordio, Bicchi, *et al.*, 2007) or parameter space transformations (Blanco, González, and Fernández-Madrigal, 2008). Multiple robots may cooperate implicitly by broadcasting their future intentions (Pedduri, Krishna, and Hexmoor, 2007) or with limited bidirectional communication (Bekris, Tsianos, and Kavraki, 2007).

A particularly successful concept for reactive navigation is the velocity obstacle (Fiorini and Shiller, 1998; Large, Sckhavat, Shiller, *et al.*, 2002) or collision cone (Chakravarthy and Ghose, 1998). Velocity obstacles have been used in practice for applications such as warning drivers of impending highway collisions (Shiller, Prasanna, and Salinger, 2008), navigating a robotic wheelchair through a crowded station (Prassler, Scholz, and Fiorini, 1999), and directing an autonomous robot within a pharmaceuticals plant (Fiorini and Botturi, 2008).

Several variations of velocity obstacles have been proposed for systems of multiple robots. Generally, these have attempted to incorporate the reactive behavior of the other robots in the environment. Formulations such as common velocity obstacles (Abe and Matsuo, 2001), recursive probabilistic velocity obstacles (Fulgenzi, Spalanzani, and Laugier, 2007; Kluge and Prassler, 2006), and reciprocal velocity obstacles (Van den Berg, Lin, and Manocha, 2008) use various means to handle reciprocity, but all have shortcomings. Specifically, the common velocity obstacle and reciprocal velocity obstacle are limited to dealing with only two robots, and the recursive probabilistic velocity obstacle may fail to converge.

Approaches such as Van den Berg, Guy, Lin, *et al.* (2011); Gal, Shiller, and Rimon (2009); Guy, Chhugani, Kim, *et al.* (2009); Tychonievich, Zaret, Mantegna, *et al.* (1989) truncate the collision cone to consider only collisions that will occur within a finite window of time.

2.3 Problem Description and Notation

We use the following problem description and notation in this chapter and, with small variations, the rest of this dissertation.

Let there be a nonempty set $\mathcal{R}$ of disc-shaped robots or disc-bounded virtual agents sharing an environment in the two-dimensional workspace $\mathbb{R}^2$ and the two-dimensional velocity space, which we denote $\mathbb{V}^2$. The environment may also contain a set $\mathcal{O}$ of dynamic, *i.e.*, moving, obstacles and static obstacles that we assume can be identified by each robot or virtual agent as not actively adapting their velocity to avoid collisions.

Each robot or virtual agent $A \in \mathcal{R}$ has a fixed radius $r_A \in \mathbb{R}^+$, a current position $\boldsymbol{p}_A \in \mathbb{R}^2$, and a current velocity $\boldsymbol{v}_A \in \mathbb{V}^2$, all of which are known to itself and may be measured by the other robots or virtual agents in the environment. Let each robot or virtual agent $A \in \mathcal{R}$ also have a point goal located at $\boldsymbol{p}_A^{\text{goal}} \in \mathbb{R}^2$ and a preferred speed $v_A^{\text{pref}} \in \mathbb{V}^+$ that are unknown to the other robots or virtual agents. The point goal may simply be a fixed point chosen in the workspace $\mathbb{R}^2$ or the result of some external criteria, such as the output of a global planning or scheduling algorithm. The preferred speed is the speed that a robot or virtual agent would take in the absence of other robots or virtual agents and obstacles, and may be similarly chosen manually or by some external algorithm. The robots may have kinematic and dynamic constraints on their motion in the workspace $\mathbb{R}^2$.

The objective of each robot or virtual agent $A \in \mathcal{R}$ is to choose a new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ at each time step independently and simultaneously to compute a trajectory toward its goal without collisions with any other robots or virtual agents and obstacles and with as few oscillations as possible. The robots or virtual agents should not communicate with each other or perform any sort of central coordination, but may assume that the other robots or virtual agents are using the same strategy to choose new velocities. We deem that a robot or virtual agent $A \in \mathcal{R}$ has reached its point goal if $\left\| \boldsymbol{p}_A - \boldsymbol{p}_A^{\text{goal}} \right\|_2 \leq d$, for some threshold $d \in \mathbb{R}^+$.

2.4 Hybrid Reciprocal Velocity Obstacles

In this section, we describe how robots avoid collisions with each other using velocity obstacles. We review the concepts of velocity obstacles and reciprocal velocity obstacles, and then introduce our formulation of hybrid reciprocal velocity obstacles that we use for navigating multiple mobile robots or virtual agents.

2.4.1 Velocity Obstacles

The velocity obstacle (Fiorini and Shiller, 1998) of a robot induced by a dynamic obstacle is the set of all velocities of the dynamic obstacle in the velocity space $\mathbb{V}^2$ that will result in a collision between the robot and the dynamic obstacle in the workspace $\mathbb{R}^2$ at some future moment in time, assuming that the dynamic obstacle maintains a constant velocity.

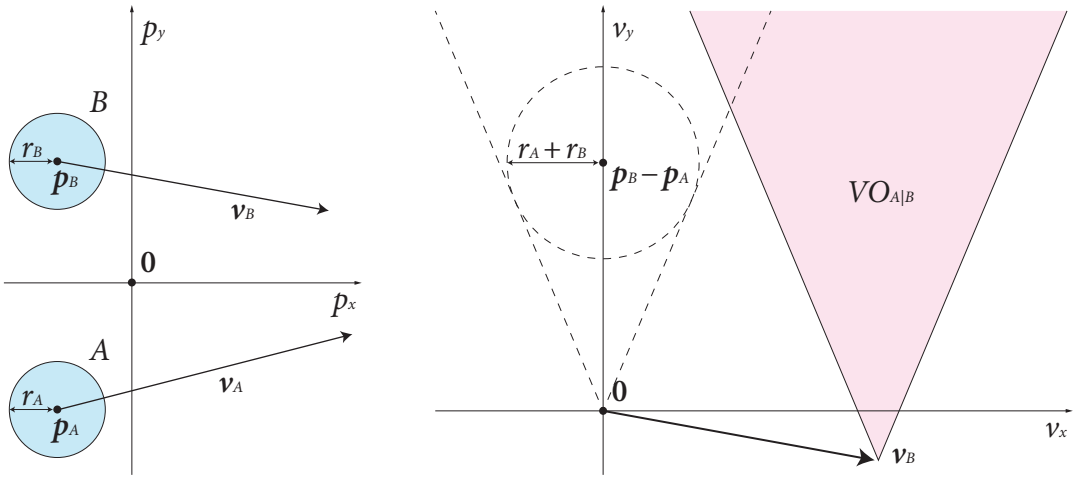


Figure 2.1: (a) Two disc-shaped robots A and B with positions $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$, velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$, and radii r_A and r_B, respectively, in the two-dimensional workspace. (b) The velocity obstacle $VO_{A|B}$ of robot A induced by robot B in the two-dimensional velocity space.

Definition 2.1. Let $A \oplus B \subseteq \mathbb{R}^2$ denote the Minkowski sum of robot A and dynamic obstacle B, let $-A \subseteq \mathbb{R}^2$ denote robot A reflected in its reference point, and let $\lambda(\boldsymbol{p}, \boldsymbol{v}) = \{\boldsymbol{p} + t\boldsymbol{v} \mid t > 0\} \subseteq \mathbb{R}^2$ be a ray starting at point $\boldsymbol{p}$ with direction $\boldsymbol{v}$. Then, the *velocity obstacle* of robot A induced by dynamic obstacle B is defined as

$$VO_{A|B} := \{\boldsymbol{v} \mid \lambda(\boldsymbol{p}_A, \boldsymbol{v} - \boldsymbol{v}_B) \cap B \oplus -A \neq \varnothing\} \subseteq \mathbb{V}^2.$$

A geometric interpretation of the region $VO_{A|B}$ appears in Figure 2.1(b). Note that the apex of the velocity obstacle is at $\boldsymbol{v}_B$.

Definition 2.2. If robot A and dynamic obstacle B are both disc-shaped with radii r_A and r_B, respectively, then Definition 2.1 simplifies to

$$VO_{A|B} := \{\boldsymbol{v} \mid \exists t > 0 :: t(\boldsymbol{v} - \boldsymbol{v}_B) \in \mathrm{D}(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B)\} \subseteq \mathbb{V}^2,$$

where $\mathrm{D}(\boldsymbol{p}, r) \subseteq \mathbb{R}^2$ is an open disc of radius $r \in \mathbb{R}^+$ centered at $\boldsymbol{p} \in \mathbb{R}$.

It follows that if robot A chooses a velocity inside $VO_{A|B}$, then robot A and dynamic obstacle B will potentially collide at some point in time. If the velocity chosen is outside $VO_{A|B}$, then a collision will not occur.

We note that the velocity obstacle is symmetric, *i.e.*, $\boldsymbol{v}_A$ is inside $VO_{A|B}$ if and only if $\boldsymbol{v}_B$ is inside $VO_{B|A}$, and translation invariant, *i.e.*, $\boldsymbol{v}_A$ is inside $VO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v})$ if and only if $\boldsymbol{v}_A + \boldsymbol{u}$ is inside $VO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v} + \boldsymbol{u})$. By the convexity of half-planes, the velocity obstacle is also convex, *i.e.*, if $\boldsymbol{v}$ and $\boldsymbol{u}$ are in the left half-plane extending from the left edge of $VO_{A|B}$, then $(1 - t)\boldsymbol{v} + t\boldsymbol{u}$ is in the left half-plane extending from the left edge of $VO_{A|B}$ for all $t \in [0, 1]$; and equivalently for the right half-plane extending from the right edge of $VO_{A|B}$.

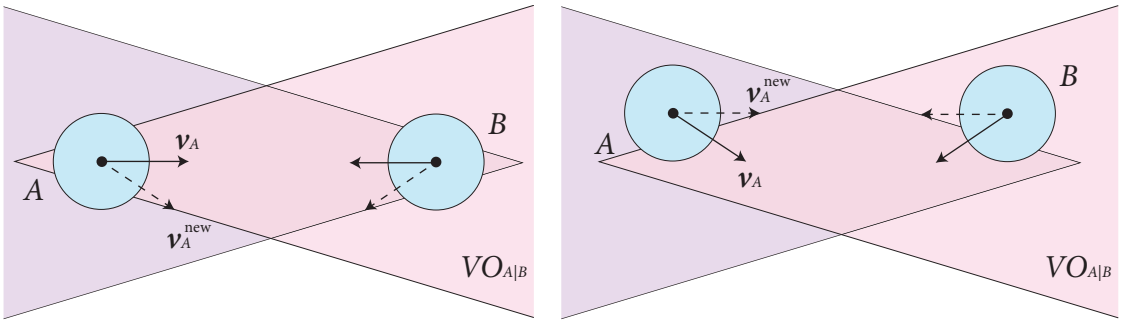


Figure 2.2: (a) Robots A and B each choose the velocity closest to their current velocity that is outside the velocity obstacle induced by the other robot. (b) At the next time step, the new velocity of each robot leaves its previous velocity outside the velocity obstacle so it returns to that velocity at the following time step.

The velocity obstacle has been successfully used to navigate one robot through an environment containing multiple dynamic obstacles by having the robot select a velocity at each time step that is outside any of the velocity obstacles induced by the dynamic obstacles (Fiorini and Shiller, 1998; Fulgenzi, Spalanzani, and Laugier, 2007; Prassler, Scholz, and Fiorini, 1999). Unfortunately, the velocity obstacle concept does not work well for navigating multiple robots, where each robot is actively adapting its velocity to avoid the other robots, since it assumes that the other robots never change their velocities (Abe and Matsuo, 2001). If all robots were to use velocity obstacles to choose a new velocity, then there would be inherent oscillations in the trajectories of the robots (Van den Berg, Lin, and Manocha, 2008). More precisely, if two robots each selected a new velocity outside the velocity obstacle of the other, then their old velocities would be valid with respect to the velocity obstacle based on the new velocities. Hence, the robots would oscillate back to the old velocities, as shown in Figure 2.2.

2.4.2 Reciprocal Velocity Obstacles

The reciprocal velocity obstacle (Van den Berg, Lin, and Manocha, 2008) addresses the problem of oscillations caused by the velocity obstacle formulation by incorporating the reactive nature of the other robots. Instead of having to take all the responsibility for avoiding collisions, as with velocity obstacles, reciprocal velocity obstacles let a robot take just half of the responsibility for avoiding a collision while assuming that the other robot involved reciprocates by taking care of the other half.

Definition 2.3. The *reciprocal velocity obstacle* of robot A induced by robot B is defined as

$$RVO_{A|B} := \left\{ \boldsymbol{v} \;\middle|\; 2\boldsymbol{v} - \boldsymbol{v}_A \in VO_{A|B} \right\} \subseteq \mathbb{V}^2.$$

The geometric interpretation of $RVO_{A|B}$ in Figure 2.3 illustrates that the velocity obstacle has been effectively translated such that its apex is at $\frac{1}{2}(\boldsymbol{v}_A + \boldsymbol{v}_B)$. In theory, the reciprocal velocity obstacle formulation guarantees that if both robots select a velocity outside the reciprocal velocity obstacle

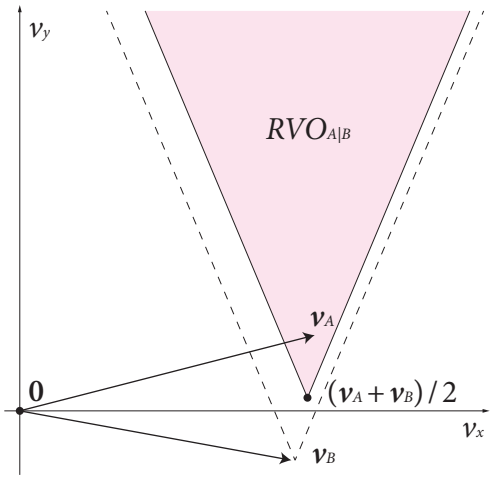


Figure 2.3: The reciprocal velocity obstacle $RVO_{A|B}$ of robot A induced by robot B in the two-dimensional velocity space.

induced by the other and both robots choose to pass each other on the same side, then the trajectories of both robots will be free of collisions and oscillations in the local time interval.

By the symmetry, translation invariance, and convexity of the velocity obstacle, it follows that if $\boldsymbol{v}_A^{\text{new}}$ is in the left half-plane extending from the left edge of $RVO_{A|B}$ and $\boldsymbol{v}_B^{\text{new}}$ is in the left half-plane extending from the left edge of $RVO_{B|A}$, then $\boldsymbol{v}_A^{\text{new}}$ is in the left half-plane extending from the left edge of $VO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v}_B^{\text{new}})$ and $\boldsymbol{v}_B^{\text{new}}$ is in the left half-plane extending from the left edge of $VO_{B|A}(\boldsymbol{v}_A = \boldsymbol{v}_A^{\text{new}})$. The equivalent statement holds for the right half-planes extending from the right edges of $RVO_{A|B}$ and $RVO_{B|A}$. Hence, there will not be collision if $\boldsymbol{v}_A^{\text{new}}$ and $\boldsymbol{v}_B^{\text{new}}$ are chosen, by the properties of the velocity obstacle (Van den Berg, Lin, and Manocha, 2008). The trajectories of the two robots can be shown to be free of oscillations by the translation invariance of the velocity obstacle (Van den Berg, Lin, and Manocha, 2008). More formally, if $\boldsymbol{v}_A^{\text{new}} = \boldsymbol{w} + \boldsymbol{u}$ and $\boldsymbol{v}_B^{\text{new}} = \boldsymbol{v} - \boldsymbol{u}$, then $\boldsymbol{w}$ is inside $RVO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v}, \boldsymbol{v}_A = \boldsymbol{w})$ if and only if $\boldsymbol{w}$ is inside $RVO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v} + \boldsymbol{u}, \boldsymbol{v}_A = \boldsymbol{w} + \boldsymbol{u})$. If each robot chooses the new velocity closest to its current velocity, then the robots will automatically pass each other on the same side (Van den Berg, Lin, and Manocha, 2008), *i.e.*, if $\boldsymbol{v}_A^{\text{new}} = \boldsymbol{v}_A + \boldsymbol{u}$ and $\boldsymbol{v}_B^{\text{new}} = \boldsymbol{v}_B - \boldsymbol{u}$, then $\boldsymbol{v}_A + \boldsymbol{u}$ is outside $RVO_{A|B}$ if and only if $\boldsymbol{v}_B - \boldsymbol{u}$ is outside $RVO_{B|A}$.

Rather than choosing new velocities closest to their current velocities, to make progress to their goals, robots A and B are typically required to select velocities closest to their own preferred velocities (Definition 2.4), *i.e.*, the velocities directed from the robots towards their goals, denoted $\boldsymbol{v}_A^{\text{pref}}$ and $\boldsymbol{v}_B^{\text{pref}}$, respectively, as in Figure 2.4. Furthermore, as shown in Figure 2.5, the presence of a third robot C may cause at least one of the robots to choose a velocity even farther from its current velocity. Unfortunately, this means that the robots may not necessarily choose the same side to pass, which may result in oscillations known as "reciprocal dances" (Feurtey, 2000).

While distinct from oscillations caused by the velocity obstacle formulation, reciprocal dances may be equally difficult for the robots to resolve and, in extreme circumstances, this behavior may become stable and the robots may oscillate forever. More precisely, there exists a configuration in

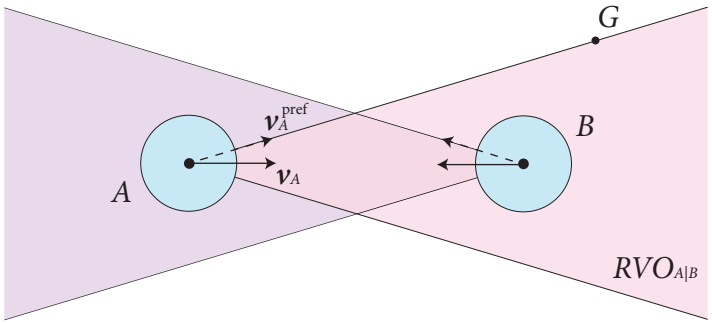


Figure 2.4: Situation when reciprocal dances may occur due to the preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ of robot A toward its goal G being oriented in a different direction to its current velocity $\boldsymbol{v}_A$.

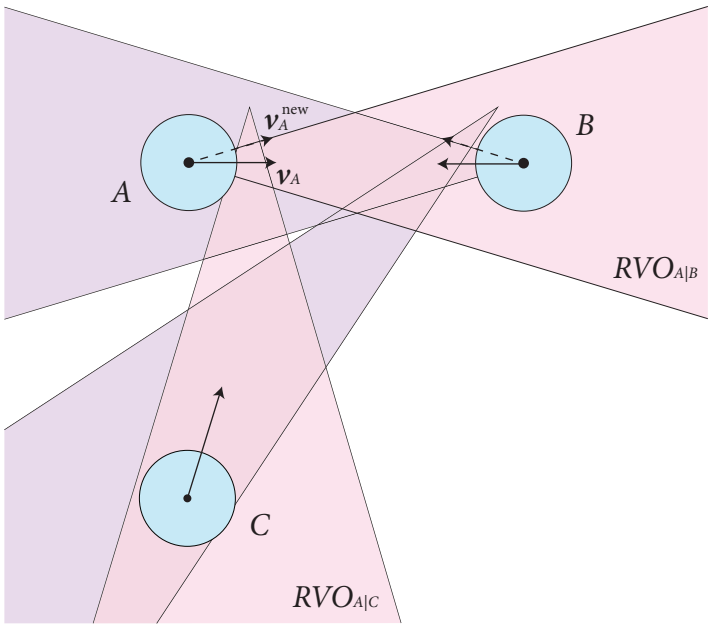


Figure 2.5: Situation when reciprocal dances may occur due to the presence a third robot C that causes the closest new velocity to the current velocity $\boldsymbol{v}_A$ outside the reciprocal velocity obstacle $RVO_{A|B}$ to be inside the reciprocal velocity obstacle $RVO_{A|C}$.

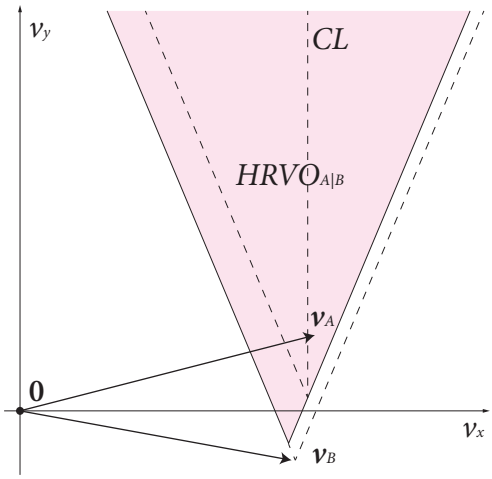


Figure 2.6: The hybrid reciprocal velocity obstacle $HRVO_{A|B}$ of robot A induced by robot B in the two-dimensional velocity space. Note that $\boldsymbol{v}_A$ is right of the centerline CL of $RVO_{A|B}$, so the apex of $HRVO_{A|B}$ is the intersection of the right side of $RVO_{A|B}$ and the left side of $VO_{A|B}$.

which if $\boldsymbol{v}_A^{\text{new}}$ is the closest velocity to $\boldsymbol{v}_A^{\text{pref}}$ that is outside $RVO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v}, \boldsymbol{v}_A = \boldsymbol{w})$ and $\boldsymbol{v}_B^{\text{new}}$ is the closest velocity to $\boldsymbol{v}_B^{\text{pref}}$ that is outside $RVO_{B|A}(\boldsymbol{v}_A = \boldsymbol{w}, \boldsymbol{v}_B = \boldsymbol{v})$, then $\boldsymbol{w}$ is the closest velocity to $\boldsymbol{v}_A^{\text{pref}}$ that is outside $RVO_{A|B}(\boldsymbol{v}_B = \boldsymbol{v}_B^{\text{new}}, \boldsymbol{v}_A = \boldsymbol{v}_A^{\text{new}})$ and $\boldsymbol{v}$ is the closest velocity to $\boldsymbol{v}_B^{\text{pref}}$ that is outside $RVO_{B|A}(\boldsymbol{v}_A = \boldsymbol{v}_A^{\text{new}}, \boldsymbol{v}_B = \boldsymbol{v}_B^{\text{new}})$.

2.4.3 Hybrid Reciprocal Velocity Obstacles

To counter this situation, we introduce the hybrid reciprocal velocity obstacle, shown in Figure 2.6. For robots A and B, if $\boldsymbol{v}_A$ is to the right of the centerline of $RVO_{A|B}$, which implies by symmetry that $\boldsymbol{v}_B$ is to the right of the centerline of $RVO_{B|A}$, we wish robot A to choose a velocity to the right of $RVO_{A|B}$. To encourage this, the reciprocal velocity obstacle is enlarged by replacing the edge on the side we do not wish the robots to pass, in this instance the left side, by the edge of the velocity obstacle $VO_{A|B}$. The apex of the resulting obstacle corresponds to the point of intersection between the right side of $RVO_{A|B}$ and the left side of $VO_{A|B}$. If $\boldsymbol{v}_A$ is to the left of the centerline, we mirror the procedure, exchanging left and right. As a hybrid of a reciprocal velocity obstacle and a velocity obstacle, we call the result a "hybrid reciprocal velocity obstacle," written $HRVO_{A|B} \subseteq \mathbb{V}^2$.

The hybrid reciprocal velocity obstacle formulation has the consequence that if robot A attempts to pass on the wrong side of robot B, then it has to give full priority to robot B, as with the velocity obstacle formulation. However, if it does choose the correct side, then it can assume the cooperation of robot B and retains equal priority, as with the reciprocal velocity obstacle formulation. This greatly reduces the possibility of oscillations while not unduly over constraining the motion of each robot. While it is still possible for the robots to pass on the wrong side of each other, this behavior is not stable because the robots may still pass on the correct side of each other in a future time step.

2.5 Navigating Multiple Mobile Robots

In this section, we show how we apply our hybrid reciprocal velocity obstacle formulation to navigating multiple mobile robots. We first describe the overall approach, and then explain how to take into account obstacles in the environment, uncertainty in radius, position, and velocity, and the kinematics and dynamics of the robots.

2.5.1 Overall Approach

We have defined the hybrid reciprocal velocity obstacle of a robot and one other robot or dynamic obstacle in an uncluttered environment. Suppose instead that we have a set of robots $\mathcal{R}$ sharing an environment with a set of dynamic and static obstacles $\mathcal{O}$. Each robot $A \in \mathcal{R}$ has a preferred velocity, which is the velocity toward its goal that the robot would take were there no other robot or dynamic obstacles in its path.

Definition 2.4. The *preferred velocity* of robot A toward its goal centered at $\boldsymbol{p}_A^{\text{goal}}$ with constant preferred speed v_A^{pref} is defined as

$$\boldsymbol{v}_A^{\text{pref}} := v_A^{\text{pref}} \frac{\boldsymbol{p}_A - \boldsymbol{p}_A^{\text{goal}}}{\left\| \boldsymbol{p}_A - \boldsymbol{p}_A^{\text{goal}} \right\|_2}.$$

As illustrated in Figure 2.7, the combined hybrid reciprocal velocity obstacle of robot A induced by all other robots and obstacles in the environment is the union of all hybrid reciprocal velocity obstacles induced by the other robots individually and all velocity obstacles induced by the obstacles.

Definition 2.5. The *combined hybrid reciprocal velocity obstacle* of robot A induced by robots $B \in \mathcal{R}$, with $A \neq B$, and obstacles $O \in \mathcal{O}$ is defined as

$$HRVO_A := \bigcup_{\substack{B \in \mathcal{R} \\ A \neq B}} HRVO_{A|B} \cup \bigcup_{O \in \mathcal{O}} VO_{A|O}.$$

It follows that each robot A should select as its new velocity $\boldsymbol{v}_A^{\text{new}}$ the velocity outside the combined hybrid reciprocal velocity obstacle that is closest to its preferred velocity:

$$\boldsymbol{v}_A^{\text{new}} = \underset{\boldsymbol{v} \notin HRVO_A}{\arg\min} \left\| \boldsymbol{v} - \boldsymbol{v}_A^{\text{pref}} \right\|_2. \tag{2.1}$$

We use the ClearPath efficient geometric algorithm (Guy, Chhugani, Kim, *et al.*, 2009) to compute the new velocity. Following the ClearPath approach, which is applicable to many variations of velocity obstacles, we represent the combined hybrid reciprocal velocity obstacle as a union of line segments. The line segments are intersected pairwise and the intersection points inside the combined hybrid reciprocal velocity obstacle are discarded. The remaining intersection points, shown by gold markers in Figure 2.7, are the permissible new velocities on the boundary of the combined hybrid

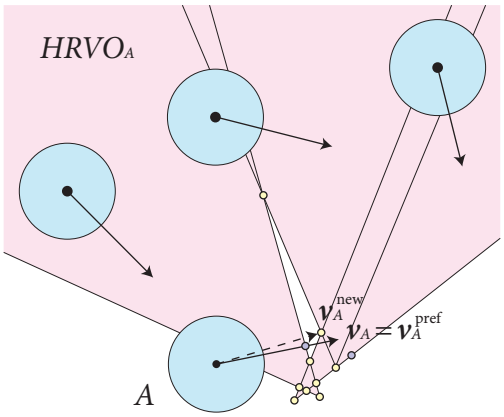


Figure 2.7: The combined hybrid reciprocal velocity obstacle $HRVO_A$ of robot A. Intersections of line segments not inside $HRVO_A$ are indicated by gold markers, and projections of preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ onto line segments not inside $HRVO_A$ are indicated by purple markers. These points are the permissible new velocities of robot A.

reciprocal velocity obstacle. In addition, we project the preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ onto the line segments and retain those points that are outside the combined hybrid reciprocal velocity obstacle, as indicated by the purple markers in Figure 2.7. It is guaranteed that the velocity that is closest to the preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ of the robot is in either of these two sets of points (Guy, Chhugani, Kim, *et al.*, 2009), and we choose that point as the new velocity of the robot.

If there are no permissible new velocities, then we discard the hybrid reciprocal velocity obstacle of the farthest away robot or obstacle and repeat the ClearPath algorithm until a velocity outside the combined hybrid reciprocal velocity obstacle becomes available. It is possible that there may be collisions between robots, or deadlocks if they both stop, however we have only observed this issue on occasion in simulations with several hundred virtual agents.

While the robot should take the new velocity $\boldsymbol{v}_A^{\text{new}}$ immediately, this may not be directly possible due to its kinematic constraints. Therefore, the velocity $\boldsymbol{v}_A^{\text{new}}$ is transformed into a control input of the robot that will let the robot reach velocity $\boldsymbol{v}_A^{\text{new}}$ as soon as possible. The overall approach is summarized by Algorithm 2.1. Note that we do not require the robots to communicate with each other; the robots use only the information that they can sense independently.

2.5.2 Dynamic and Static Obstacles

The presence of dynamic and static obstacles in the workspace $\mathbb{R}^2$ necessitates slight changes to our approach. Specifically, the combined hybrid reciprocal velocity obstacle uses the velocity obstacles, rather than hybrid reciprocal velocity obstacles, of each obstacle in the environment since a robot cannot assume the cooperation of the obstacles to avoid collisions whichever side of the relevant reciprocal velocity obstacle the current velocity of the robot lies. The velocity obstacle of a static line segment obstacle is shown in Figure 2.8.

An additional consideration is that an obstacle may block the path from the current position of a robot to its goal causing the preferred velocity to be directed toward or through the obstacle. To

Algorithm 2.1: Our method for navigating multiple mobile robots in the two-dimensional workspace using hybrid reciprocal velocity obstacles.

inputs
  List of robots $\mathcal{R} \neq \varnothing$
  List of static and dynamic obstacles $\mathcal{O}$
loop
  for all $A \in \mathcal{R}$ **do**
    Sense $\boldsymbol{p}_A$ and $\boldsymbol{v}_A$
    for all $B \in \mathcal{R}$ **such that** $A \neq B$ **do**
      Sense $\boldsymbol{p}_B$ and $\boldsymbol{v}_B$
      Construct $VO_{A|B}$
      Expand $VO_{A|B}$ to $VO^*_{A|B}$
      Construct $RVO_{A|B}$
      Expand $RVO_{A|B}$ to $RVO^*_{A|B}$
      Locate centerline CL^* of $RVO^*_{A|B}$
      if $\boldsymbol{v}_A$ is right of CL^* **then**
        Replace left side of $RVO^*_{A|B}$ with left side of $VO^*_{A|B}$ to construct $HRVO^*_{A|B}$
      else
        Replace right side of $RVO^*_{A|B}$ with right side of $VO^*_{A|B}$ to construct $HRVO^*_{A|B}$
      end if
    end for
    for all $O \in \mathcal{O}$ **do**
      Sense $\boldsymbol{p}_O$ and $\boldsymbol{v}_O$
      Construct $VO_{A|O}$
      Expand $VO_{A|O}$ to $VO^*_{A|O}$
    end for
    Construct $HRVO^*_A$ from all $HRVO^*_{A|B}$ and $VO^*_{A|O}$
    Compute preferred velocity $\boldsymbol{v}^{\text{pref}}_A$
    Compute new velocity $\boldsymbol{v}^{\text{new}}_A \notin HRVO^*_A$ closest to $\boldsymbol{v}^{\text{pref}}_A$ using ClearPath algorithm
    Compute control inputs from $\boldsymbol{v}^{\text{new}}_A$
    Apply control inputs to actuators of A
  end for
end loop

account for this, we can incorporate a global navigation strategy, such as a precomputed probabilistic roadmap (Kavraki, Švestka, Latombe, *et al.*, 1996) or rapidly-exploring random tree (LaValle and Kuffner, 2001), and use the nearest visible node of a covering roadmap as a waypoint or sub-goal toward which to direct the preferred velocity rather than directly toward the ultimate goal. An example of part of such a roadmap is also illustrated in Figure 2.8.

We have found the covering roadmap approach to be preferable to attempting to follow complete precomputed paths that are guaranteed not to collide with static obstacles since it is unclear how to compute a preferred velocity that allows a robot to rejoin the path without oscillations if it has deviated from the path to avoid another robot or dynamic obstacle. Additionally, the precomputed path may not remain visible if the deviation of a robot from the path is large. When using a covering roadmap, a robot may simply choose another node, resulting in a different path to the goal.

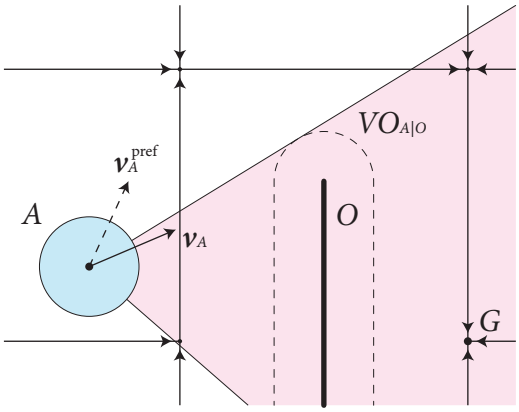


Figure 2.8: The velocity obstacle $VO_{A|O}$ of robot A induced by a static line segment obstacle O, and part of a roadmap of sub-goals for navigation to the goal G of the robot.

2.5.3 Uncertainty in Radius, Position, and Velocity

To calculate the hybrid reciprocal velocity obstacles induced by other robots, each robot requires the radius, current position, and current velocity of every robot, and the position and physical extent of every obstacle. Because this data is obtained using sensors, it inevitably contains uncertainty. This may jeopardize the correct functioning of our approach.

We assume that each robot has onboard sensing and is able to measure the positions, or relative positions, of itself and every other robot or obstacle, and that it has prior knowledge or is able to sense the radius of itself and the other robots. We then use a Kalman filter (Kalman, 1960) to obtain accurate estimates of the radii, positions, and velocities of the robots and obstacles. The Kalman filter also provides an estimate of the variance of the measured quantities.

Now, if $P_A \sim \mathcal{N}(\bar{\boldsymbol{p}}_A, \sigma_{\boldsymbol{p}\,A})$ is a Gaussian distribution of the measured position $\boldsymbol{p}_A$ with mean $\bar{\boldsymbol{p}}_A$ and variance $\sigma_{\boldsymbol{p}\,A}$, and $P_B \sim \mathcal{N}(\bar{\boldsymbol{p}}_B, \sigma_{\boldsymbol{p}\,B})$ is a Gaussian distribution of the measured position $\boldsymbol{p}_B$ with mean $\bar{\boldsymbol{p}}_B$ and variance $\sigma_{\boldsymbol{p}\,B}$, then

$$P_{B-A} \sim \mathcal{N}(\bar{\boldsymbol{p}}_B - \bar{\boldsymbol{p}}_A, \sigma_{\boldsymbol{p}\,B-A})$$

is a Gaussian distribution of the measured relative position $\boldsymbol{p}_B - \boldsymbol{p}_A$ with mean $\bar{\boldsymbol{p}}_B - \bar{\boldsymbol{p}}_A$ and variance $\sigma_{\boldsymbol{p}\,B-A}$. Moreover, if $R_A \sim \mathcal{N}(\bar{r}_A, \sigma_{r\,A})$ is a Gaussian distribution of the measured radius r_A and $R_B \sim \mathcal{N}(\bar{r}_B, \sigma_{r\,B})$ is a Gaussian distribution of the measured radius r_B, then

$$R_{A+B} \sim \mathcal{N}(\bar{r}_A + \bar{r}_B, \sigma_{r\,A+B})$$

is a Gaussian distribution of the measured $r_A + r_B$.

Assuming that $V_B \sim \mathcal{N}(\bar{\boldsymbol{v}}_B, \sigma_{\boldsymbol{v}\,B})$ is a Gaussian distribution of the measured velocity $\boldsymbol{v}_B$, we use these distributions to construct the "uncertainty-adjusted velocity obstacle," written $VO^*_{A|B} \subseteq \mathbb{V}^2$, as follows. First, we expand the relative angle of the sides of the velocity obstacle to encompass the area corresponding to the Minkowski sum of a disc of radius $\bar{r}_A + \bar{r}_B + \sigma_{r\,A+B}$ and a linear transformation

of a disc centered at $\bar{\boldsymbol{p}}_B - \bar{\boldsymbol{p}}_A + \bar{\boldsymbol{v}}_B$ by the variance $\sigma_{\boldsymbol{p}\,B-A}$. Secondly, we move the sides of the velocity obstacle out perpendicularly by an amount large enough to encompass the linear transformation of a unit disc by the variance $\sigma_{\boldsymbol{v}\,B}$. As the Gaussian distribution has infinite extent, it is necessary to choose a finite segment of the distribution. In practice, we have found one standard deviation around the mean to be acceptable, as shown in Figure 2.9.

The "uncertainty-adjusted reciprocal velocity obstacle," written $RVO^*_{A|B} \subseteq \mathbb{V}^2$, illustrated in Figure 2.10, is constructed in a similar manner. We expand the relative angle of the sides of the reciprocal velocity obstacle to encompass the area corresponding to the Minkowski sum of a disc of radius $\bar{r}_A + \bar{r}_B + \sigma_{r\,A+B}$ and a linear transformation of a disc centered at $\bar{\boldsymbol{p}}_B - \bar{\boldsymbol{p}}_A + \frac{1}{2}(\bar{\boldsymbol{v}}_A + \bar{\boldsymbol{v}}_B)$ by the variance $\sigma_{\boldsymbol{p}\,B-A}$. Assuming that

$$V_{(A+B)/2} \sim \mathcal{N}(\tfrac{1}{2}(\bar{\boldsymbol{v}}_A + \bar{\boldsymbol{v}}_B), \sigma_{\boldsymbol{v}\,(A+B)/2})$$

is a Gaussian distribution of the measured $\frac{1}{2}(\boldsymbol{v}_A + \boldsymbol{v}_B)$, where $V_A \sim \mathcal{N}(\bar{\boldsymbol{v}}_A, \sigma_{\boldsymbol{v}\,A})$ is a Gaussian distribution of the velocity $\boldsymbol{v}_A$, then we move the sides of the velocity obstacle out perpendicularly by an amount large enough to encompass the linear transformation of a unit disc by the variance $\sigma_{\boldsymbol{v}\,(A+B)/2}$. Note that if $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ are independent, then $RVO^*_{A|B}$ will be comparatively smaller than $VO^*_{A|B}$.

The "uncertainty-adjusted hybrid reciprocal velocity obstacle," written $HRVO^*_{A|B} \subseteq \mathbb{V}^2$, shown in Figure 2.11, is constructed from $VO^*_{A|B}$ and $RVO^*_{A|B}$. The "combined uncertainty-adjusted hybrid reciprocal velocity obstacle," written $HRVO^*_A$, is constructed in an analogous way to the combined hybrid reciprocal velocity obstacle (Definition 2.5). Any velocity obstacles $VO_{A|O}$ of each obstacle O in the environment are expanded to uncertainty-adjusted velocity obstacles $VO^*_{A|O}$ as part of the construction.

2.5.4 Kinematic Constraints

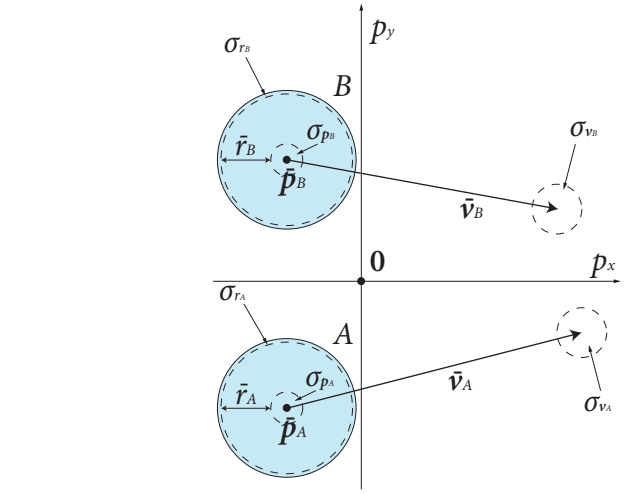
We can apply our approach to mobile robots with differential-drive kinematic constraints. Such robots, shown in Figure 2.12, use a simple drive mechanism that consists of two drive wheels mounted on a common axis with each wheel able to be independently driven in both forward and reverse directions.

Definition 2.6. The *kinematic constraints of a differential-drive robot* with position $(x, y) \in \mathbb{R}^2$, orientation $\theta \in \mathbb{S}^1$, and wheel track $L \in \mathbb{R}^+$ are described by

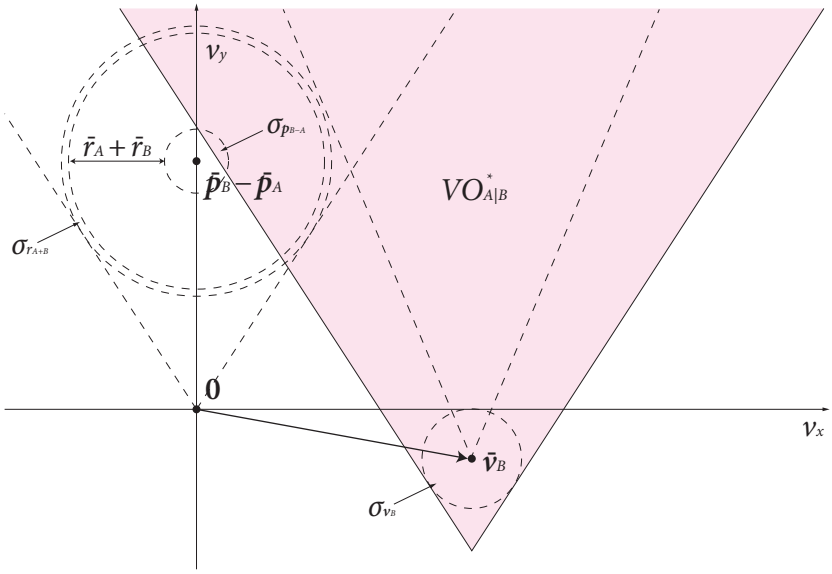
$$\dot{x} = \frac{v_{\text{left}} + v_{\text{right}}}{2} \cos\theta, \qquad (2.2)$$

$$\dot{y} = \frac{v_{\text{left}} + v_{\text{right}}}{2} \sin\theta,$$

$$\dot{\theta} = \frac{v_{\text{right}} - v_{\text{left}}}{L}, \qquad (2.3)$$



(a)



(b)

Figure 2.9: (a) Two disc-shaped robots A and B with uncertain positions, velocities, and radii in the two-dimensional workspace. (b) The uncertainty-adjusted velocity obstacle $VO^*_{A|B}$ of robot A induced by robot B in the two-dimensional velocity space.

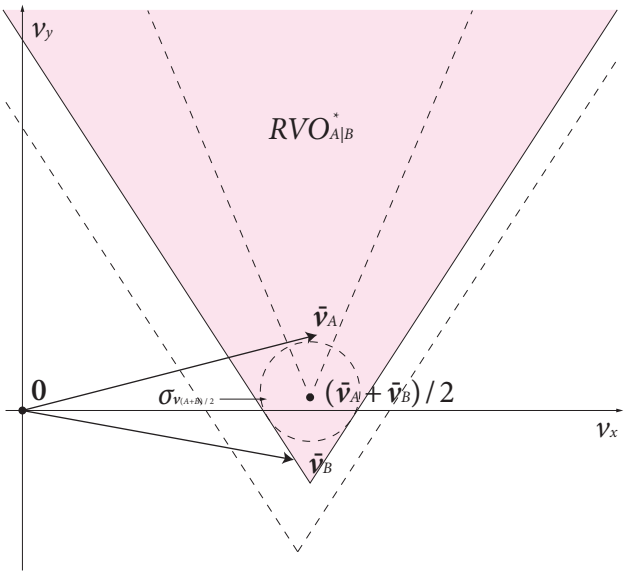


Figure 2.10: The uncertainty-adjusted reciprocal velocity obstacle $RVO^*_{A|B}$ of robot A induced by robot B in the two-dimensional velocity space.

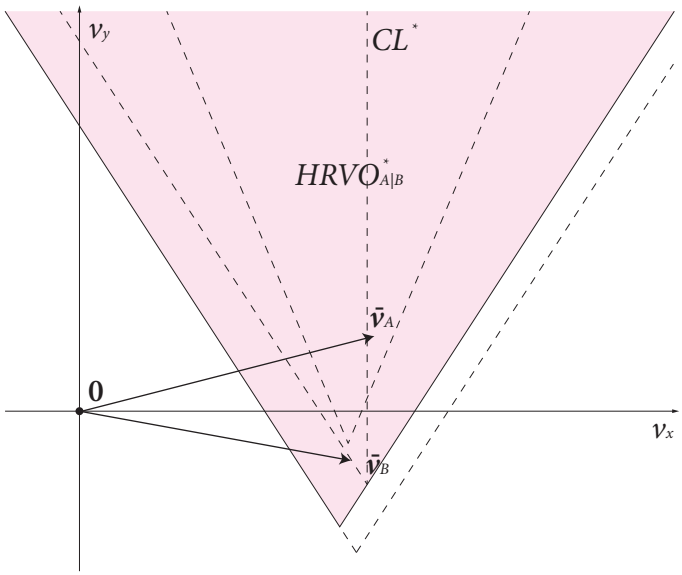


Figure 2.11: The uncertainty-adjusted hybrid reciprocal velocity obstacle $HRVO^*_{A|B}$ of robot A induced by robot B in the two-dimensional velocity space. Since $\bar{\boldsymbol{v}}_A$ is right of the centerline CL^* of $RVO^*_{A|B}$, the apex of $HRVO^*_{A|B}$ is the intersection of the right side of $RVO^*_{A|B}$ and the left side of $VO^*_{A|B}$.

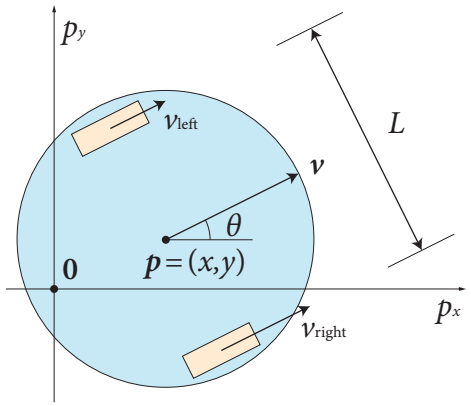


Figure 2.12: The kinematic model of a differential-drive robot with center $\boldsymbol{p} = (x, y)$, radius r, and wheel track L in the two-dimensional workspace. The control inputs are the left and right signed wheel speeds v_{left} and v_{right}, respectively.

where the signed left and right wheel speeds $v_{\text{left}}, v_{\text{right}} \in \mathbb{V}$, respectively, are bounded to a given maximum $v_{\max} \in \mathbb{V}^+$, such that

$$\begin{aligned} -v_{\max} \leq v_{\text{left}} \leq v_{\max}, \\ -v_{\max} \leq v_{\text{right}} \leq v_{\max}. \end{aligned} \tag{2.4}$$

The signed wheel speeds are the control inputs of the robot. When $v_{\text{left}} = v_{\text{right}} > 0$, the robot will move straight forward, when $v_{\text{left}} > v_{\text{right}} > 0$, it will arc right, and when $v_{\text{left}} = -v_{\text{right}} \neq 0$, it will spin in place. The center of the robot is able to follow any continuous path within the workspace $\mathbb{R}^2$ (LaValle, 2006), although not necessarily instantaneously.

Now, we must transform the computed new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ from (2.1) to wheel speeds $v_{\text{left}}, v_{\text{right}} \in \mathbb{V}$ given the current orientation θ of the robot. We choose to set v_{left} and v_{right} such that $\boldsymbol{v}_A^{\text{new}}$ is obtained after a prescribed amount of time $\omega > 0$ to ensure oscillation-free motion. More formally, suppose that $\boldsymbol{v}_A^{\text{new}} = (v_x, v_y)$. Then, the target orientation is $\phi = \arctan(v_y/v_x) \in \mathbb{S}^1$ and the target speed is $u = \left\|\boldsymbol{v}_A^{\text{new}}\right\|_2 \in \mathbb{V}$. Therefore, to obtain the target speed, it follows from (2.2) that

$$v_{\text{right}} + v_{\text{left}} = 2u.$$

Moreover, if the difference between the target orientation and current orientation is $\Delta = \phi - \theta \in \mathbb{S}^1$, such that $\Delta \in [-\pi, \pi]$, then to move from the current orientation to the target orientation in ω time it follows directly from (2.3) that

$$v_{\text{right}} - v_{\text{left}} = \frac{L\Delta}{\omega}.$$

The desired values of v_{left} and v_{right} may be calculated from the system formed by these two equations.

Now, if the constraints of (2.4) invalidate the computed values of v_{left} and v_{right}, we first attempt to move the values into the interval $[-v_{\max}, v_{\max}]$ while keeping $v_{\text{right}} - v_{\text{left}}$ constant, such that the target orientation is obtained after ω time. If, after this, v_{left} and v_{right} still do not satisfy the constraints

of (2.4), in which case $\left|v_{\text{right}} - v_{\text{left}}\right| > 2v_{\max}$, then v_{left} and v_{right} are clamped to the extremes of the interval, such that the robot maximally rotates in place.

The choice of ω must be small enough to allow the robot to react quickly to other robots and obstacles in its path. However, if set lower than the duration δt of each time step, the robot will overshoot its target orientation, leading to oscillations in its trajectory. A low value of ω may also result in less smooth paths, since the robot may have to frequently rotate in place to achieve its target orientation. In practice, we have found that a value of $\omega = 3\delta t$ yields good results.

2.5.5 Dynamic Constraints

Each robot is also likely to be subject to dynamic constraints that reduce the velocities that it can attain within a time step δt. Suppose robot A with current velocity $\boldsymbol{v}_A$ has a maximum speed $v_A^{\max}$ and maximum acceleration $a_A^{\max}$, then the set of velocities from which to choose $\boldsymbol{v}_A^{\text{new}}$ is reduced to

$$\{\boldsymbol{v} \mid \boldsymbol{v} \notin HRVO_A \wedge \|\boldsymbol{v}\|_2 \leq v_A^{\max} \wedge \|\boldsymbol{v} - \boldsymbol{v}_A\|_2 \leq a_A^{\max}\delta t\} \subseteq \mathbb{V}^2.$$

2.6 Experimentation and Performance

In this section, we describe the implementation of our approach and report the results of our experiments involving multiple mobile robots or virtual agents.

2.6.1 Implementation

We applied our approach to a set of iRobot Create mobile robots (see Figure 2.13) using wireless remote control and centralized sensing. The iRobot Create is a differential-drive robot based on the iRobot Roomba vacuum-cleaning robot (Jones, Mack, Nugent, *et al.*, 2005). It has two individually actuated wheels and a third passive caster wheel for balance. The maximum speed of the robot is 0.5 m/s in both forward and reverse directions, its shape is circular with radius 0.17 m, and it has a mass of less than 2.5 kg. The limited sensing power of the iRobot Create does not allow it to localize itself with any degree of accuracy.

For convenience, the robots were tracked centrally using fiducial markers within a 6 m² indoor space using an overhead Point Grey Flea2 digital video camera connected to a notebook computer via FireWire 800. Images were captured at a resolution of 1032x776 and a refresh rate of 30 Hz. They were processed using the ARToolKit augmented reality library (Kato and Billinghurst, 1999) to determine the position and orientation of each robot with an absolute error of less than 0.01 m. The velocity of each robot was inferred from the position and orientation measurements using a Kalman filter.

We implemented our approach in C++, and the code was compiled using the Intel C++ Compiler, version 11.1. All calculations were performed on a dual-core 2.53 GHz Intel Core 2 Duo processor within a standard notebook computer containing 4 GB of memory and running Microsoft Windows 7 Ultimate, 64-bit version. However, to ensure that our approach applies when each robot uses its own on-board sensing and computing, only the localization was carried out centrally. The other calculations

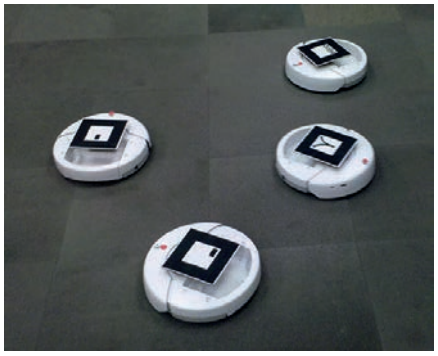


Figure 2.13: A photo of four iRobot Create mobile robots in our experimentation setting. Note the fiducial markers attached to the top of each robot for tracking via an overhead digital video camera.

for each robot were carried out in separate and independent threads and in parallel, where possible, using Intel Threading Building Blocks, version 4. The computed wheel speeds, encoded in 4 b serial data packets, were sent to the robots over a Bluetooth 2.1 virtual serial connection at a speed of 57.6 kb/s and an average latency of 0.5 s.

In the simulated experiments of virtual agents, for efficiency, only a subset of all other virtual agents within a fixed radius of each virtual agent, with respect to Euclidean distance in the workspace $\mathbb{R}^2$, were considered for the computation of hybrid reciprocal velocity obstacles, and these virtual agents were selected at the beginning of every time step using an algorithm based on k-D trees (De Berg, Cheong, Van Kreveld, *et al.*, 2008).

2.6.2 Experiments

Using the iRobot Create mobile robots, we tested our approach in the following experiments.

2.1. Five robots are spaced evenly on the perimeter of a circular environment. Their goals are to navigate to the antipodal positions on the circle. The robots will meet and have to negotiate around each other in the center.

2.2. One robot takes the role of a dynamic obstacle moving at a constant velocity. The other robots have to cross the path of the dynamic obstacle to navigate to their goals.

In addition, we tested the ability of our approach to handle static obstacles and the scalability of our approach in the following simulated experiments.

2.3. Four virtual agents must navigate from one side of a rectangular environment to the other, negotiating around each other in the center. Blocking their path are two static obstacles that form a passage through which they must pass.

2.4. From ten to one thousand virtual agents are spaced evenly on the perimeter of a circular environment. Their goals are to navigate to the antipodal positions on the circle. The virtual agents will meet and have to negotiate around each other in the center.

Videos of these experiments are available online at `http://gamma.cs.unc.edu/HRVO/`.

2.6.3 Discussion

Figure 2.14 shows traces of the five robots in Experiment 2.1 for three variations of the velocity obstacle formation. In Figure 2.14(a), when the robots use velocity obstacles, the traces are not smooth due to oscillations, while in Figure 2.14(b), for reciprocal velocity obstacles, the beginnings of the traces are not smooth due to reciprocal dances. The traces in Figure 2.14(c), with robots navigating using hybrid reciprocal velocity obstacles, show no oscillations or reciprocal dances over their entire lengths for any robot. In each experiment, the velocities of the robots were updated at a rate of 30 Hz, limited by the refresh rate of the tracking camera.

Figure 2.15 shows that, in Experiment 2.2, the hybrid reciprocal velocity obstacle formulation can naturally deal with the presence of a dynamic obstacle that may not necessarily adapt its motion to the presence of other robots. Two robots increase speed to cross ahead of the dynamic obstacle, while the third slows and crosses behind. The combined hybrid reciprocal velocity obstacle of each robot is the union of the hybrid reciprocal velocity obstacles of the other two robots and the velocity obstacle of the dynamic obstacle. Note that we do not consider how to identify between a robot and a dynamic obstacle, simply that our formulation is capable of handling the distinction should it be made.

Figure 2.16 shows the traces, in Experiment 2.3, of four virtual agents navigating though a passage while avoiding collisions with the static obstacles and each other. The virtual agents merge into two lines before the passage and pass through in double file. Once they have negotiated the passage, they move toward their goals.

Figure 2.17 shows three screenshots of Experiment 2.4, our simulation with one hundred virtual agents. Figure 2.17(a) shows the starting configuration, Figure 2.17(b) shows the virtual agents approaching the center of the circle, and Figure 2.17(c) shows the virtual agents moving towards the perimeter of the circle having passed the center. All computations were completed in less than 15 µs per virtual agent, per time step on one core. The timing of Experiment 2.4 for three variations of velocity obstacles is shown in Table 2.1. Given the reactive nature of the hybrid reciprocal velocity obstacle formulation, it is difficult to calculate any formal bound on the computation time.

Table 2.2 and Figure 2.18 show the timing of Experiment 2.4 with an increasing number of virtual agents moving across a circular environment with a radius that has been increased proportionally to the number of virtual agents. This shows that our formulation can navigate up to one thousand virtual agents before the computation time per time step exceeds the 30 Hz refresh rate of a sensor such as the tracking camera used in our experiments with iRobot Create mobile robots. All timings are for one thread running on a single processor core.

Table 2.3 and Figure 2.19 show the collisions in Experiment 2.4 with an increasing number of virtual agents moving across a circular environment of fixed radius so that the density of virtual agents is increased and free space reduced. As the number of virtual agents exceeds one hundred, a small, increasing number of collisions per time step are observed as there is insufficient space left uncovered with hybrid reciprocal velocity obstacles for some virtual agents.

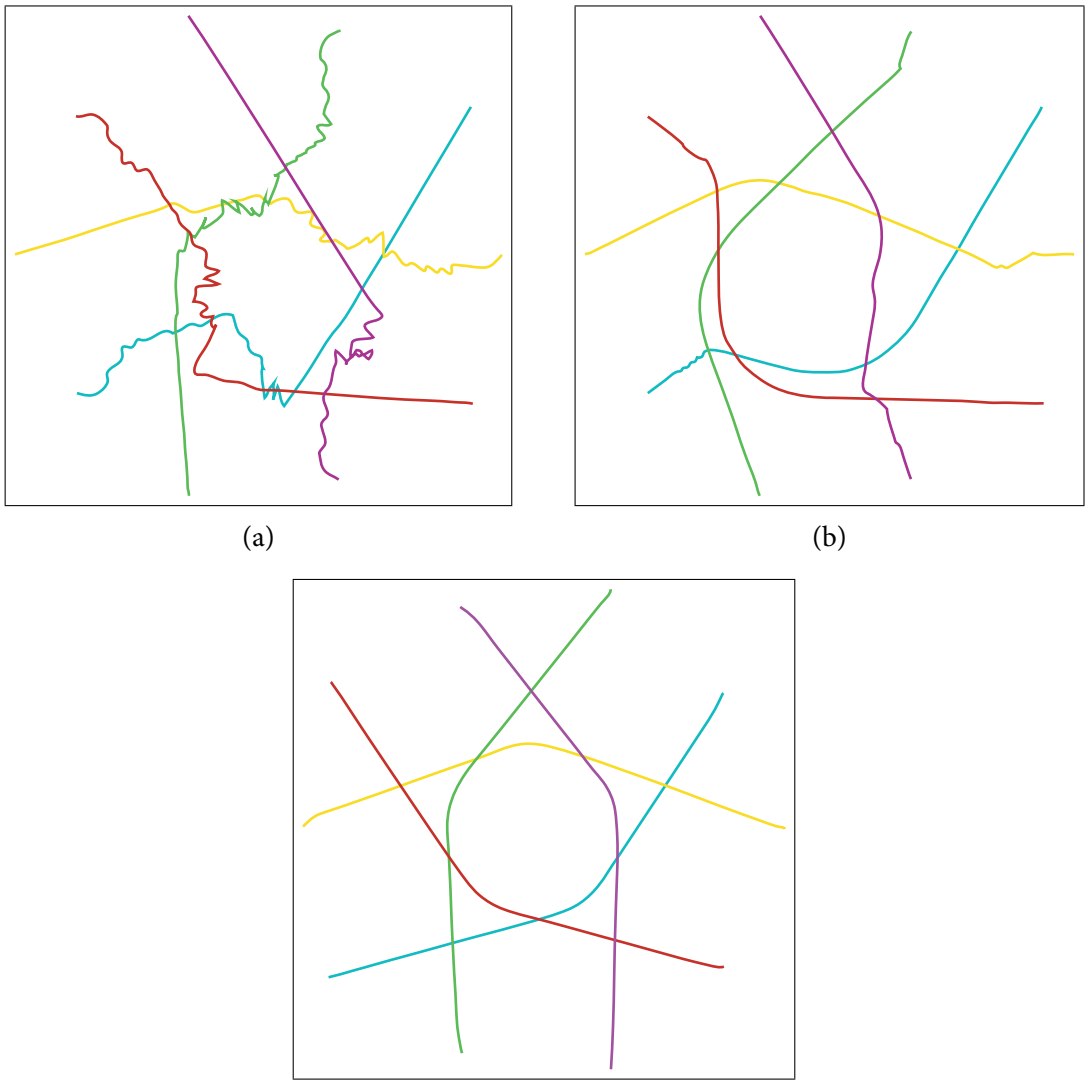


Figure 2.14: Traces of the trajectories of five mobile robots navigating simultaneously across a circular environment in the two-dimensional workspace using (a) velocity obstacles, (b) reciprocal velocity obstacles, and (c) hybrid reciprocal velocity obstacles (Experiment 2.1).

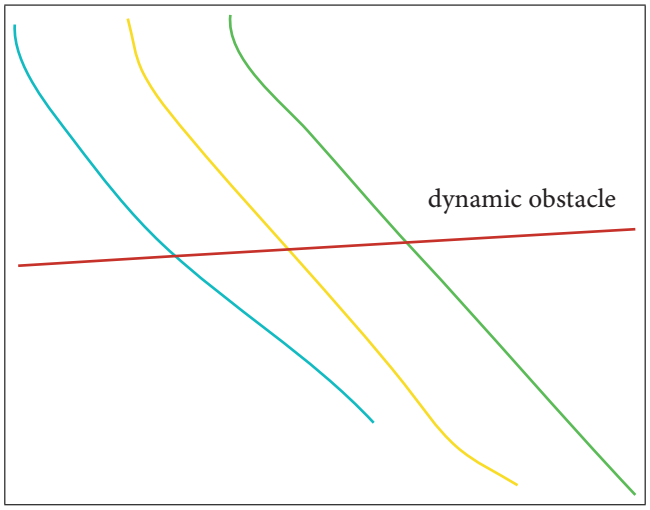


Figure 2.15: Traces of the trajectories of three mobile robots navigating simultaneously across the path of a dynamic obstacle (red, near-horizontal line) in the two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.2).

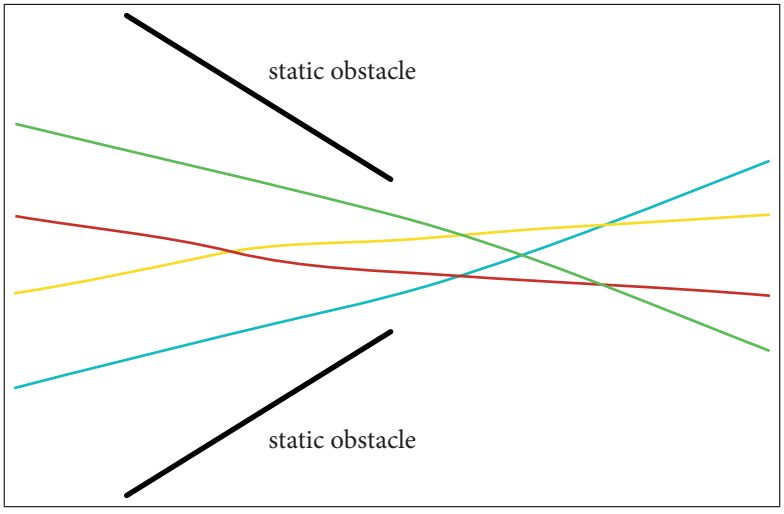


Figure 2.16: Traces of the trajectories of four virtual agents navigating simultaneously through a passage formed by two static obstacles in the two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.3).

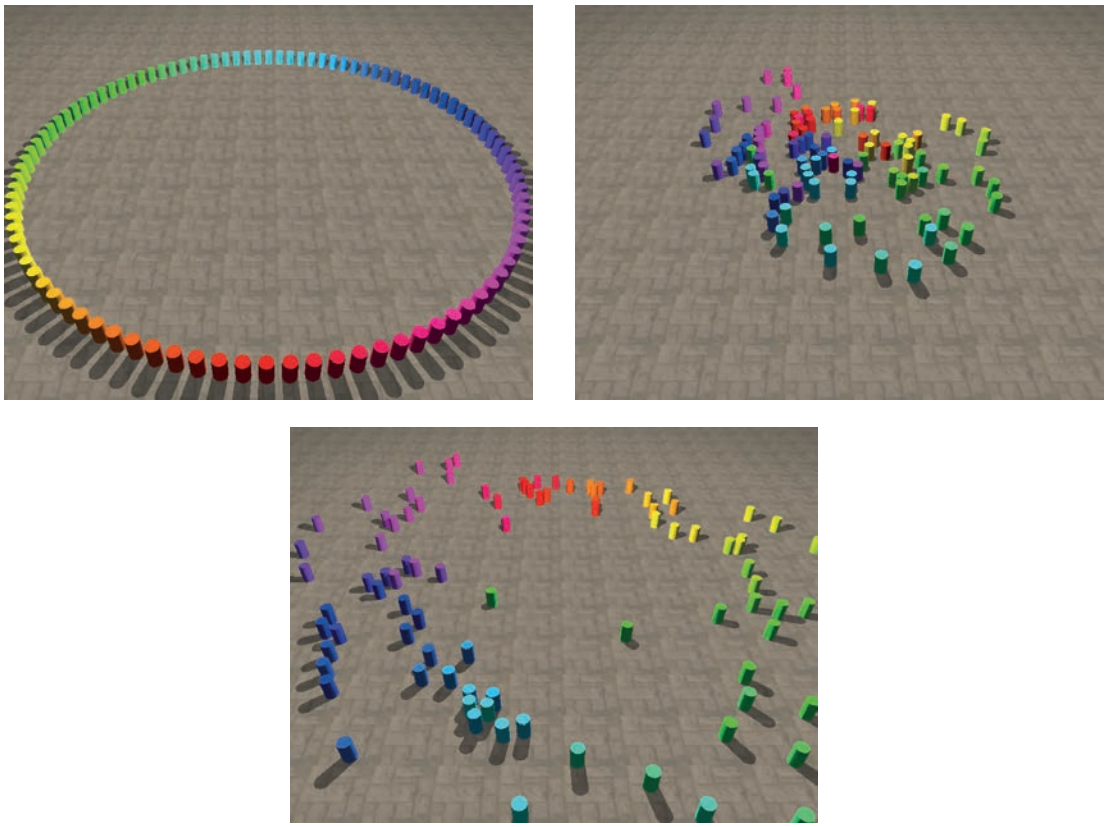


Figure 2.17: Screenshots of one hundred virtual agents navigating simultaneously across a circular environment in the two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.4).

Table 2.1: Average computation time at each time step on one core of a 2.53 GHz Intel Core 2 Duo processor, in milliseconds, for virtual agents navigating across a circular environment in the two-dimensional workspace using velocity obstacles, reciprocal velocity obstacles, and hybrid reciprocal velocity obstacles (Experiment 2.4).

	Computation time per time step (ms)
Velocity obstacle	0.8
Reciprocal velocity obstacle	0.8
Hybrid reciprocal velocity obstacle	1.2

Table 2.2: Average computation time at each time step on one core of a 2.53 GHz Intel Core 2 Duo processor, in milliseconds, for virtual agents navigating across a circular environment of increasing radius in the two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.4).

Number of virtual agents	Computation time per time step (ms)
10	0.2
100	1.2
200	2.8
300	4.6
400	6.8
500	9.1
1000	25.7

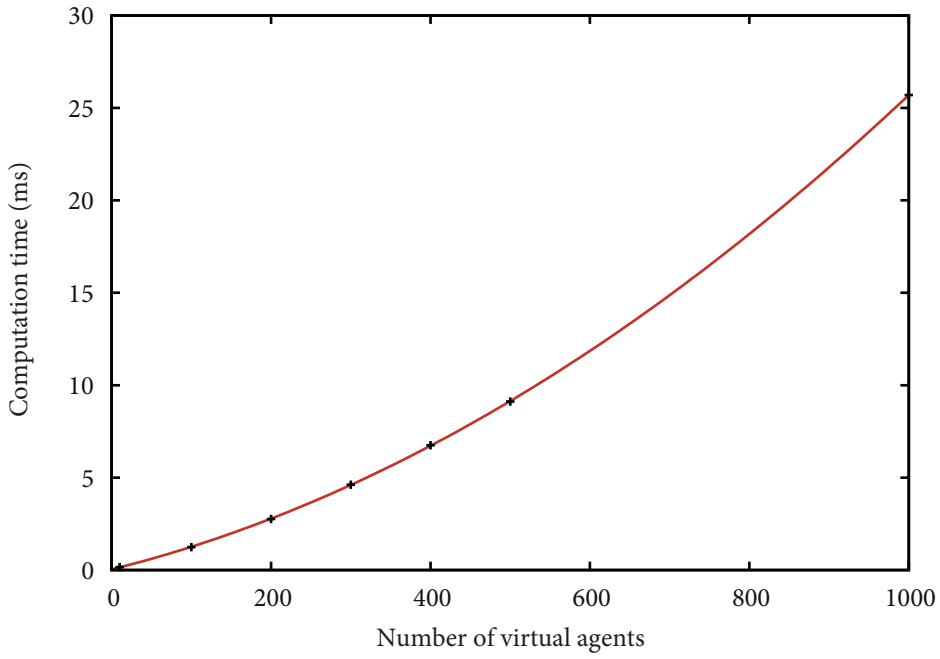


Figure 2.18: A plot of the average computation time at each time step on one core of a 2.53 GHz Intel Core 2 Duo processor, in milliseconds, for virtual agents navigating across a circular environment of increasing radius in the two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.4).

Table 2.3: Average number of collisions during each time step between virtual agents navigating across a circular environment of fixed radius in two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.4).

Number of virtual agents	Number of collisions per time step
10	0
100	0.2
200	0.9
300	1.9
400	3.1
500	4.4
1000	15.1

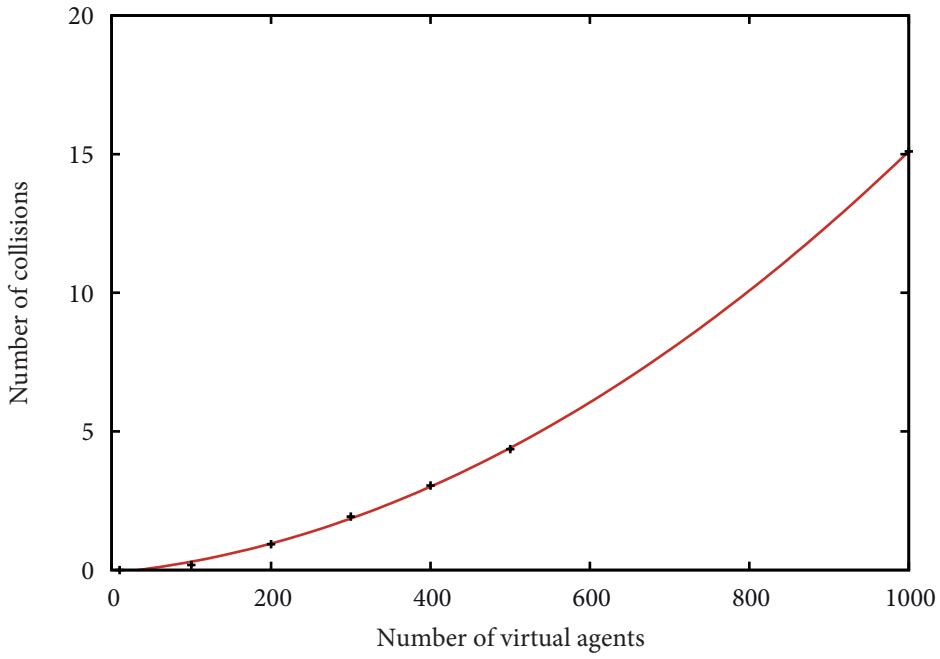


Figure 2.19: A plot of the average number of collisions during each time step between virtual agents navigating across a circular environment of fixed radius in the two-dimensional workspace using hybrid reciprocal velocity obstacles (Experiment 2.4).

Chapter 3

SMOOTH NAVIGATION OF MULTIPLE ROBOTS UNDER DIFFERENTIAL-DRIVE CONSTRAINTS

3.1 Introduction and Motivation

Most prior work on smooth and collision-free navigation has been limited to single robots moving among dynamic obstacles. There is extensive work on navigating multiple robots, including global methods based on centralized or decoupled approaches (Kant and Zucker, 1986) and local and reactive methods (Van den Berg, Lin, and Manocha, 2008; Chakravarthy and Ghose, 1998; Fiorini and Shiller, 1998; Fox, Burgard, and Thrun, 1997; Petti and Fraichard, 2005) for computing collision-free paths. However, most of these algorithms do not take into account kinematic constraints, nor do they provide guarantees of smoothness, and the resulting paths may have discontinuities.

We present a local and reactive method for navigating multiple independent robots with differential-drive constraints in the two-dimensional workspace using the optimal reciprocal collision avoidance algorithm (Van den Berg, Guy, Lin, *et al.*, 2011). Optimal reciprocal collision avoidance excludes an entire half-plane of velocities in the two-dimensional velocity space. While more conservative than the hybrid reciprocal velocity obstacle, this allows us to make several theoretical guarantees, including that of smooth trajectories for which we provide a proof. Our method incorporates the kinematic constraints of a differential-drive robot into the optimal reciprocal collision avoidance algorithm while preserving the guarantee of smooth trajectories by enlarging, in a precise way, the radius of a differential-drive robot to an "effective radius" that provides a reference point, which we call the "effective center," that can be maneuvered in any direction instantaneously.

We have implemented and applied our approach to a set of iRobot Create robots in an indoor environment using wireless remote control via Bluetooth and sensing from a ceiling-mounted digital video camera. Our experiments show that our approach achieves visibly smooth navigation without collisions in an environment containing multiple differential-drive robots. We have also demonstrated the low computational requirements and scalability of our approach in simulations of up to one thousand differential-drive robots. Moreover, to the best of our knowledge, this is the first algorithm that theoretically guarantees smooth and collision-free trajectories for multiple independent robots navigating in a shared environment that is local and reactive and takes into account the kinematic constraints of a differential-drive robot.

3.2 Prior Work

Some of the earliest work on planning for robots with kinematic constraints dates back to the Dubins car (Dubins, 1957), a simplified car model that was restricted to forward motions with a fixed speed and bounded turning radius. The Reeds-Shepp car (Reeds and Shepp, 1990) added a reverse gear to the Dubins car, and the simple car (Latombe, 1991; Laumond, Sekhavat, and Lamiraux, 1998; LaValle, 2006) extended the model further with variable speed in any direction.

Many works, as described in Section 2.2, have examined the issue of a single robot navigating through an environment containing dynamic obstacles, including local and reactive methods, such as the velocity obstacle (Fiorini and Shiller, 1998) and its extensions to navigating multiple robots or virtual agents (Van den Berg, Lin, and Manocha, 2008). These methods do not, however, provide any theoretical guarantees on the smoothness of the resulting trajectories.

Work on the navigation of multiple robots with kinematic constraints has previously considered problems such as follow-the-leader behaviors (Desai, Ostrowski, and Kumar, 2001) and time-optimal trajectories (Balkcom and Mason, 2002). Proposed methods have included a modified rapidly-exploring random tree planner (Bruce and Veloso, 2005) and mixed integer nonlinear programming (Peng and Akella, 2005). While centralized global planners (LaValle, 2006) generate a single composite system from many robots and apply traditional single robot navigation algorithms, prior single robot algorithms for smooth or continuous curvature paths have not necessarily been applicable to such a system in terms of generating individual trajectories of each robot that are smooth.

3.3 Optimal Reciprocal Collision Avoidance

In this section, we briefly review the concepts of the truncated velocity obstacle and optimal reciprocal collision avoidance that we use for navigating multiple differential-drive robots in the two-dimensional workspace and the two-dimensional velocity space. We also provide a proof that robots navigating using optimal reciprocal collision avoidance are theoretically guaranteed to have smooth trajectories in the two-dimensional workspace.

3.3.1 Truncated Velocity Obstacles

The velocity obstacle, as defined in Definition 2.1, is the set of all velocities of a robot in the velocity space $\mathbb{V}^2$ that will result in a collision between itself and another robot or dynamic obstacle in the plane $\mathbb{R}^2$ at some future moment in time, however distant the potential time of collision. Suppose instead we only consider collisions that will potentially occur within the short, finite window of time $[0, \tau]$. Then, we have a truncated velocity obstacle (Van den Berg, Guy, Lin, *et al.*, 2011; Guy, Chhugani, Kim, *et al.*, 2009; Tychonievich, Zaret, Mantegna, *et al.*, 1989), as follows.

Definition 3.1. The *truncated velocity obstacle* of robot A induced by robot or dynamic obstacle B within the window of time $[0, \tau]$ is defined as the set of all velocities of robot or dynamic obstacle B that may cause a collision between robot A and robot or dynamic obstacle B within the short, finite

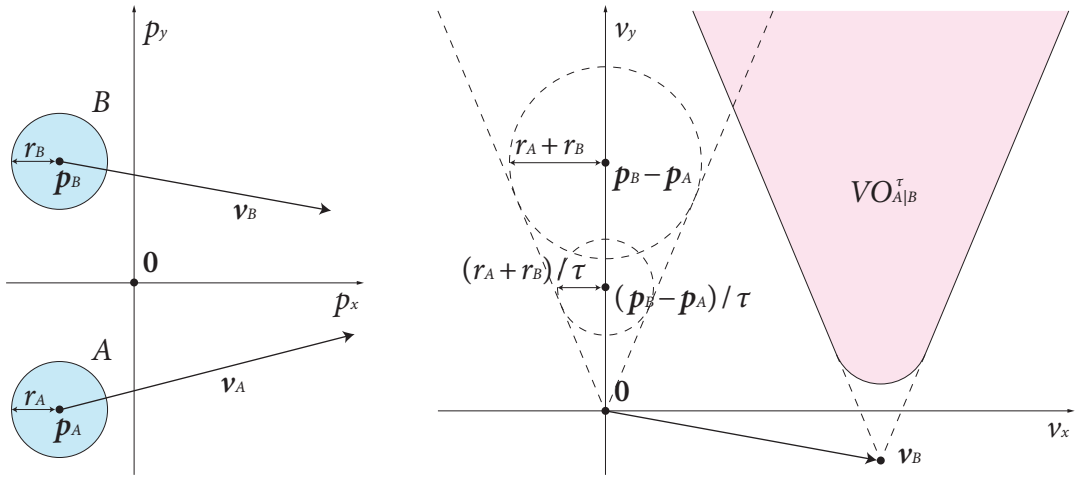


Figure 3.1: (a) Two disc-shaped robots A and B with positions $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$, velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$, and radii r_A and r_B, respectively, in the two-dimensional workspace. (b) The truncated velocity obstacle $VO^{\tau}_{A|B}$ of robot A induced by robot B within the finite window of time $[0, \tau = 2]$ (shaded pink) in the two-dimensional velocity space. The sides of the truncated velocity obstacle $VO^{\tau}_{A|B}$ are tangent to a disc of radius $r_A + r_B$ with center $\boldsymbol{p}_B - \boldsymbol{p}_A$ and the velocity obstacle is truncated by a disc of radius $(r_A + r_B)/\tau$ with center $(\boldsymbol{p}_B - \boldsymbol{p}_A)/\tau$.

window of time $[0, \tau]$, assuming that each robot or dynamic obstacle continues on the same trajectory during that window of time, *i.e.*,

$$VO^{\tau}_{A|B} := \{\boldsymbol{v} \mid \exists t \in [0, \tau] :: t(\boldsymbol{v} - \boldsymbol{v}_B) \in B \oplus -A\} \subseteq \mathbb{V}^2,$$

where $A \oplus B \subseteq \mathbb{R}^2$ denotes the Minkowski sum of robot A and robot or dynamic obstacle B, and $-A \subseteq \mathbb{R}^2$ denotes robot A reflected in its reference point, as described in Definition 2.1.

Figure 3.1 shows a configuration of two disc-shaped robots A and B, and the truncated velocity obstacle $VO^{\tau}_{A|B}$ of robot A induced by robot B. Compare this to Figure 2.1 and the non-truncated velocity obstacle. As in Definition 2.2, when both robots are disc-shaped, as shown in Figure 3.1, then the definition of the truncated velocity obstacle is simplified.

Definition 3.2. The *truncated velocity obstacle* of disc-shaped robot A with position $\boldsymbol{p}_A$ and radius r_A induced by disc-shaped robot or dynamic obstacle B with position $\boldsymbol{p}_B$, velocity $\boldsymbol{v}_B$, and radius r_B within the window of time $[0, \tau]$ is defined as

$$VO^{\tau}_{A|B} := \{\boldsymbol{v} \mid \exists t \in [0, \tau] :: t(\boldsymbol{v} - \boldsymbol{v}_B) \in \mathrm{D}(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B)\} \subseteq \mathbb{V}^2,$$

where $\mathrm{D}(\boldsymbol{p}, r) \subseteq \mathbb{R}^2$ is an open disc of radius $r \in \mathbb{R}^+$ centered at $\boldsymbol{p} \in \mathbb{R}$.

3.3.2 Optimal Reciprocal Collision Avoidance

As noted in Section 2.4.1 for non-truncated velocity obstacles, while choosing a velocity from outside the truncated velocity obstacle induced by another robot or dynamic obstacle ensures that

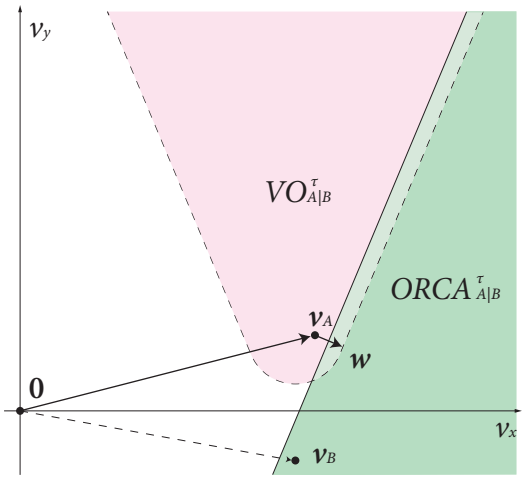


Figure 3.2: The optimal reciprocal collision avoidance half-plane $ORCA^{\tau}_{A|B}$ of robot A induced by robot B within the finite window of time $[0, \tau]$ (shaded green) in the two-dimensional velocity space. The half-plane $ORCA^{\tau}_{A|B}$ is bounded by a line perpendicular to $\boldsymbol{w}$ through $\boldsymbol{v}_A + \frac{1}{2}\boldsymbol{w}$, where $\boldsymbol{w}$ is from $\boldsymbol{v}_A - \boldsymbol{v}_B$ to the closest point on $\partial VO^{\tau}_{A|B}$.

a robot will not collide with the other robot or dynamic obstacle, their respective trajectories in the workspace $\mathbb{R}^2$ may not be smooth due to oscillations in velocity (Van den Berg, Lin, and Manocha, 2008). Optimal reciprocal collision avoidance (Van den Berg, Guy, Lin, *et al.*, 2011) resolves this issue, and, additionally, allows us to prove that the trajectories in the workspace $\mathbb{R}^2$ of robots navigating using the algorithm are theoretically guaranteed to be smooth.

The optimal reciprocal collision avoidance half-plane of robot A induced by robot B within the window of time $[0, \tau]$ is defined as follows. Referring to Figure 3.2, let $\boldsymbol{w} \in \mathbb{V}^2$ be the vector from the relative velocity $\boldsymbol{v}_A - \boldsymbol{v}_B \in \mathbb{V}^2$ to the closest point on the boundary $\partial VO^{\tau}_{A|B} \subseteq \mathbb{V}^2$ of the truncated velocity obstacle $VO^{\tau}_{A|B}$, with respect to Euclidean distance in the velocity space $\mathbb{V}^2$, *i.e.*,

$$\boldsymbol{w} := (\operatorname*{arg\,min}_{\boldsymbol{v} \in \partial VO^{\tau}_{A|B}} \|\boldsymbol{v} - (\boldsymbol{v}_A - \boldsymbol{v}_B)\|_2) - (\boldsymbol{v}_A - \boldsymbol{v}_B).$$

It follows that $\boldsymbol{w}$ is the minimum change of the relative velocity $\boldsymbol{v}_A - \boldsymbol{v}_B$ that will avoid a collision between the two robots.

Now, let $\boldsymbol{n} \in \mathbb{V}^2$ be the outward normal of $\partial VO^{\tau}_{A|B}$ at the point $\boldsymbol{v}_A - \boldsymbol{v}_B + \boldsymbol{w} \in \mathbb{V}^2$ and, as in Definition 2.3, assume that each robot will take half of the responsibility for avoiding a collision while assuming that the other robot involved reciprocates by taking care of the other half. Hence, both robots adapt their velocity by $\frac{1}{2}\boldsymbol{w}$ to avoid colliding with each other. It follows that the set of velocities of robot A that will avoid a collision with robot B is a half-plane in the direction of $\boldsymbol{n}$ starting at the point $\boldsymbol{v}_A + \frac{1}{2}\boldsymbol{w} \in \mathbb{V}^2$.

Definition 3.3. The *optimal reciprocal collision avoidance half-plane* of robot A induced by robot B within the window of time $[0, \tau]$ is defined as

$$ORCA^{\tau}_{A|B} := \left\{\boldsymbol{v} \;\middle|\; (\boldsymbol{v} - (\boldsymbol{v}_A + \tfrac{1}{2}\boldsymbol{w})) \cdot \boldsymbol{n} \geq 0\right\} \subseteq \mathbb{V}^2,$$

where $\boldsymbol{w}$ is the vector from $\boldsymbol{v}_A - \boldsymbol{v}_B$ to the closest point on the boundary of $VO^{\tau}_{A|B}$, and $\boldsymbol{n}$ is the outward normal of $\partial VO^{\tau}_{A|B}$ at the point $\boldsymbol{v}_A - \boldsymbol{v}_B + \boldsymbol{w}$.

This is the half-plane of velocities shown in Figure 3.2. In common with the truncated velocity obstacle, both robots can construct their optimal reciprocal collision avoidance half-planes without communication with each other, requiring knowledge of only the radius, position, and velocity of each other. It follows that if robot A chooses a velocity $\boldsymbol{v}_A \in ORCA^{\tau}_{A|B}$ and robot B chooses a velocity $\boldsymbol{v}_B \in ORCA^{\tau}_{B|A}$, then the trajectories of the robots will be smooth and collision-free within the window of time $[0, \tau]$.

3.3.3 Theoretical Guarantee of Smooth Trajectories

The key property of our formulation, which distinguishes it from other local and reactive methods, is that it theoretically guarantees that the motion of each robot in the workspace $\mathbb{R}^2$ is smooth.

Definition 3.4. A robot A has a *smooth trajectory* in the workspace $\mathbb{R}^2$ if the trajectory $\boldsymbol{v}_A(t)$ in the velocity space $\mathbb{V}^2$, generated by the sequence of velocities $\boldsymbol{v}_A$ at each time step, is continuous, for small time step $\delta t \to 0$, *i.e.*,

$$\boldsymbol{v}_A(t) \approx \boldsymbol{v}_A(t),$$

where $\approx$ denotes "arbitrarily close to" as $\delta t \to 0$.

Hence, in addition to the theoretical guarantee of collision-free paths (Van den Berg, Guy, Lin, *et al.*, 2011), we have the following new result for optimal reciprocal collision avoidance.

Theorem 3.1. *Given a small time step δt, the trajectory $\boldsymbol{v}_A(t)$ in the velocity space $\mathbb{V}^2$, generated by the sequence of velocities $\boldsymbol{v}_A$ that are computed using optimal reciprocal collision avoidance at each time step, is continuous,* i.e.,

$$\boldsymbol{v}_A(t) \approx \boldsymbol{v}_A(t + \delta t).$$

The proof of Theorem 3.1 is by induction on the time step δt. In the inductive step, we will prove that

$$\boldsymbol{v}_A(t + \delta t) \approx \boldsymbol{v}_A(t) \Rightarrow \boldsymbol{v}_A(t + 2\delta t) \approx \boldsymbol{v}_A(t + \delta t),$$

and in the base case we will prove that

$$\boldsymbol{v}_A(\delta t) \approx \boldsymbol{v}_A(0),$$

for a proper initialization of the simulation. The proof requires several additional results, which are presented first.

Lemma 3.1. *Given a small time step δt, the trajectory $\boldsymbol{v}^{\text{pref}}_A(t)$ in the velocity space $\mathbb{V}^2$, generated by the sequence of preferred velocities at each time step, is continuous,* i.e.,

$$\boldsymbol{v}^{\text{pref}}_A(t) \approx \boldsymbol{v}^{\text{pref}}_A(t + \delta t).$$

Proof. By Definition 2.4, the preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ of each robot is the difference of its current position and its goal position $\boldsymbol{p}_A^{\text{goal}}$. Since the trajectory $\boldsymbol{p}_A(t)$ of positions in the workspace $\mathbb{R}^2$ is clearly continuous and $\boldsymbol{p}_A^{\text{goal}}$ is fixed, it follows that $\boldsymbol{v}_A^{\text{pref}}(t)$ is continuous. □

Lemma 3.2. *Given a small time step δt, if the trajectory $\boldsymbol{v}_A(t)$ in the velocity space $\mathbb{V}^2$, is continuous,* i.e., *$\boldsymbol{v}_A(t) \approx \boldsymbol{v}_A(t + \delta t)$, then the trajectory $ORCA^{\tau}_{A|B}(t)$ is continuous,* i.e.,

$$ORCA^{\tau}_{A|B}(t) \approx ORCA^{\tau}_{A|B}(t + \delta t).$$

Corollary 3.1 (of Lemma 3.2). *Let $ORCA^{\tau}_{A}(t)$ be the intersection of half-planes $ORCA^{\tau}_{A|B}(t)$ of all other robots B, such that $A \neq B$, at time t. Given a small time step δt, the trajectory $ORCA^{\tau}_{A}(t)$ in the velocity space $\mathbb{V}^2$ is continuous,* i.e.,

$$ORCA^{\tau}_{A}(t) \approx ORCA^{\tau}_{A}(t + \delta t).$$

Proof. By Definition 3.3, the half-plane $ORCA^{\tau}_{A|B}$ is a tangent line to the truncated velocity obstacle $VO^{\tau}_{A|B}$. The trajectory $VO^{\tau}_{A|B}(t)$ is continuous because the trajectory $\boldsymbol{p}_A(t)$ of positions in the workspace $\mathbb{R}^2$ is continuous, the trajectory $\boldsymbol{v}_A(t)$ of velocities in the velocity space $\mathbb{V}^2$ is continuous by the inductive hypothesis, and the radius of robot A is fixed. Therefore, it follows from Definition 3.1 that $ORCA^{\tau}_{A|B}(t)$ is continuous. □

Lemma 3.3. *The optimal reciprocal collision avoidance algorithm selects the new velocity $\boldsymbol{v}_A$ based on the preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ and the intersection of half-planes $ORCA^{\tau}_{A}$,* i.e.,

$$\boldsymbol{v}_A(t + \delta t) = f\left(\boldsymbol{v}_A^{\text{pref}}(t), ORCA^{\tau}_{A}(t)\right),$$

for some function $f : \mathbb{V}^2 \to \mathbb{V}^2$, the linear programming function. If the trajectory $\boldsymbol{v}_A^{\text{pref}}(t)$ of preferred velocities in the velocity space $\mathbb{V}^2$ is continuous and the trajectory $ORCA^{\tau}_{A}(t)$ in the velocity space $\mathbb{V}^2$ is continuous, then the function f is continuous, i.e.,

$$f\left(\boldsymbol{v}_A^{\text{pref}}(t), ORCA^{\tau}_{A}(t)\right) \approx f\left(\boldsymbol{v}_A^{\text{pref}}(t + \delta t), ORCA^{\tau}_{A}(t + \delta t)\right).$$

Proof. The function f is a projection of $\boldsymbol{v}_A^{\text{pref}}(t)$ onto $ORCA^{\tau}_{A}(t)$ in the velocity space $\mathbb{V}^2$. Since the trajectories $\boldsymbol{v}_A^{\text{pref}}(t)$ and $ORCA^{\tau}_{A}(t)$ in the velocity space $\mathbb{V}^2$ are continuous by Lemma 3.1 and Corollary 3.1, respectively, it follows that the function f is continuous. □

Proof of Theorem 3.1. The inductive step proceeds as follows. From the optimal reciprocal collision avoidance algorithm,

$$\boldsymbol{v}_A(t + 2\delta t) = f\left(\boldsymbol{v}_A^{\text{pref}}(t + \delta t), ORCA^{\tau}_{A}(t + \delta t)\right).$$

Moreover, $ORCA_A^\tau(t+\delta t) \approx ORCA_A^\tau(t)$ from Corollary 3.1 and $\boldsymbol{v}_A^{\text{pref}}(t+\delta t) \approx \boldsymbol{v}_A^{\text{pref}}(t)$ from Lemma 3.1. The function f is continuous, *i.e.*,

$$f\left(\boldsymbol{v}_A^{\text{pref}}(t+\delta t), ORCA_A^\tau(t+\delta t)\right) \approx f\left(\boldsymbol{v}_A^{\text{pref}}(t), ORCA_A^\tau(t)\right),$$

and by Lemma 3.3,

$$\boldsymbol{v}_A(t+\delta t) = f\left(\boldsymbol{v}_A^{\text{pref}}(t), ORCA_A^\tau(t)\right).$$

Hence, $\boldsymbol{v}_A(t+2\delta t) \approx \boldsymbol{v}_A(t+\delta t)$. By the inductive hypothesis in Lemma 3.2, $\boldsymbol{v}_A(t)$ is continuous, *i.e.*, $\boldsymbol{v}_A(t+\delta t) \approx \boldsymbol{v}_A(t)$, so it follows that

$$\boldsymbol{v}_A(t+2\delta t) \approx \boldsymbol{v}_A(t+\delta t),$$

as required.

The base case of the inductive proof is $\boldsymbol{v}_A(\delta t) \approx \boldsymbol{v}_A(0)$. This occurs if each robot is initialized with $\boldsymbol{v}_A(0) = \boldsymbol{v}_A^{\text{pref}}(0)$ and its starting position is such that all other robots are sufficiently distant that $\boldsymbol{v}_A^{\text{pref}}(0) \in ORCA_A^\tau$. Then, the robot can keep moving at its preferred velocity $\boldsymbol{v}_A^{\text{pref}}$ after the first time step, *i.e.*, $\boldsymbol{v}_A(\delta t) = \boldsymbol{v}_A^{\text{pref}}(\delta t)$. Hence, $\boldsymbol{v}_A(\delta t) \approx \boldsymbol{v}_A(0)$ by Lemma 3.1, as required. □

3.4 Navigating Multiple Differential-Drive Robots

In this section, we show how we apply optimal reciprocal collision avoidance to navigating multiple differential-drive robots. We describe the overall approach and explain how to take into account obstacles in the environment, uncertainty in radius, position, and velocity, and outline a method for transforming collision-free velocities to the control inputs of the robot that maintains the theoretical guarantee that the trajectory of the robot will be smooth.

3.4.1 Overall Approach

We adopt the problem description and notation of Section 2.3 and suppose now that we have a set of differential-drive robots $\mathcal{R}$ sharing an environment in the workspace $\mathbb{R}^2$ with a set of dynamic and static obstacles $\mathcal{O}$. Each differential-drive robot A has a preferred velocity $\boldsymbol{v}_A^{\text{pref}}$, as defined by Definition 2.4, toward its point goal. It follows that the space of optimal reciprocal collision avoidance velocities of differential-drive robot A induced by all other robots B, such that $A \neq B$, within the window of time $[0, \tau]$ is the intersection of optimal reciprocal collision avoidance half-planes induced by each other robot B:

$$ORCA_A^\tau := \bigcap_{\substack{B \in \mathcal{R} \\ A \neq B}} ORCA_{A|B}^\tau \subseteq \mathbb{V}^2.$$

Each differential-drive robot A should, subject to its kinematic constraints, select as its new velocity $\boldsymbol{v}_A^{\text{new}}$ the velocity inside the intersection of optimal reciprocal collision avoidance half-planes

Algorithm 3.1: Our method for navigating multiple differential-drive robots in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius.

inputs
  List of differential-drive robots $\mathcal{R} \neq \varnothing$
  List of static and dynamic obstacles $\mathcal{O}$
loop
  for all $A \in \mathcal{R}$ **do**
    Sense $\boldsymbol{p}_A = (x, y)_A$ and $\boldsymbol{v}_A$
    Calculate $\boldsymbol{q}_A = (X, Y)_A$
    for all $B \in \mathcal{R}$ **such that** $A \neq B$ **do**
      Sense $\boldsymbol{p}_B = (x, y)_B$ and $\boldsymbol{v}_B$
      Calculate $\boldsymbol{q}_B = (X, Y)_B$
      Construct $VO^{\tau}_{A|B}$
      Expand $VO^{\tau}_{A|B}$ to $VO^{\tau*}_{A|B}$
      Construct $ORCA^{\tau*}_{A|B}$
    end for
    for all $O \in \mathcal{O}$ **do**
      Sense $\boldsymbol{p}_O$ and $\boldsymbol{v}_O$
      Construct $VO^{\tau}_{A|O}$
      Expand $VO^{\tau}_{A|O}$ to $VO^{\tau*}_{A|O}$
      Construct $ORCA^{\tau*}_{A|O}$
    end for
    Construct $ORCA^{\tau*}_{A}$ from all $ORCA^{\tau*}_{A|B}$ and $ORCA^{\tau*}_{A|O}$
    Compute preferred velocity $\boldsymbol{v}^{\text{pref}}_A$
    Compute new velocity $\boldsymbol{v}^{\text{new}}_A \in ORCA^{\tau*}_{A}$ closest to $\boldsymbol{v}^{\text{pref}}_A$ using linear programming
    Compute control inputs $\boldsymbol{u} = (v_{\text{left}}, v_{\text{right}})$ from $\boldsymbol{v}^{\text{new}}_A$ by solving $\boldsymbol{v}_A = M(\theta) \cdot \boldsymbol{u}$
    Apply control inputs to actuators of A
  end for
end loop

that is closest to its preferred velocity:

$$\boldsymbol{v}^{\text{new}}_A = \underset{\boldsymbol{v} \in ORCA^{\tau}_{A}}{\arg\min} \left\| \boldsymbol{v} - \boldsymbol{v}^{\text{pref}}_A \right\|_2 . \tag{3.1}$$

This may be calculated efficiently using a linear programming algorithm (De Berg, Cheong, Van Kreveld, *et al.*, 2008). The overall approach is summarized by Algorithm 3.1.

3.4.2 Dynamic and Static Obstacles

Dynamic and static obstacles in the workspace $\mathbb{R}^2$ are handled in a broadly similar way to other robots in the environment with the exception that since an obstacle does not reciprocate in making a collision-avoiding maneuver, the optimal reciprocal collision avoidance half-plane $ORCA^{\tau}_{A|O}$ of robot A induced by obstacle O within the window of time $[0, \tau]$ is the tangent line to $\partial VO^{\tau}_{A|O}$ at the closest point to the velocity $\boldsymbol{v}_A \in \mathbb{V}^2$ of the robot, with respect to Euclidean distance in the velocity space $\mathbb{V}^2$, as described in Van den Berg, Guy, Lin, *et al.* (2011). This follows the same reasoning as the use of velocity obstacles in place of hybrid reciprocal velocity obstacles in Section 2.5.2. Similarly, a covering roadmap (Canny, 1988) may be incorporated into the calculations of the preferred velocity

(Definition 2.4), with each robot navigating to the nearest visible roadmap node as a waypoint to their ultimate goal position.

3.4.3 Uncertainty in Radius, Position, and Velocity

Since each differential-drive robot uses sensors to measure the radius, position, and velocity of every robot and dynamic obstacle, and the position and physical extent of every static obstacle, this data may be uncertain. Hence, the position and orientation of the corresponding truncated velocity obstacles and optimal reciprocal collision avoidance half-planes may also be uncertain.

Assuming that each robot has onboard sensing and is able to measure the positions and derive the velocities of itself and every other robot or obstacle, and that it has prior knowledge or is able to sense the radius of itself and the other robots and dynamic obstacles and the position and physical extent of the static obstacles, we can adapt the idea of an uncertainty-adjusted velocity obstacle from Section 2.5.3 to an uncertainty-adjusted truncated velocity obstacle.

Definition 3.5. The *uncertainty-adjusted truncated velocity obstacle* of robot A induced by robot or dynamic obstacle B, written $VO^{\tau *}_{A|B} \subseteq \mathbb{V}^2$, is defined as the union of all truncated velocity obstacles $VO^{\tau}_{A|B} \subseteq \mathbb{V}^2$ over all radii and positions of robot A and robot or dynamic obstacle B in the workspace $\mathbb{R}^2$ and all velocities of robot or dynamic obstacle B in the velocity space $\mathbb{V}^2$ within one standard deviation of their respective means.

If we substitute the uncertainty-adjusted truncated velocity obstacle $VO^{\tau *}_{A|B}$ for $VO^{\tau}_{A|B}$ in Definition 3.3, then we have the uncertainty-adjusted optimal reciprocal collision avoidance half-plane.

Definition 3.6. The *uncertainty-adjusted optimal reciprocal collision avoidance half-plane* of robot A induced by robot B within the window of time $[0, \tau]$ is defined as

$$ORCA^{\tau *}_{A|B} := \left\{ \boldsymbol{v} \;\middle|\; (\boldsymbol{v} - (\boldsymbol{v}_A + \tfrac{1}{2}\boldsymbol{w}^*)) \cdot \boldsymbol{n}^* \geq 0 \right\} \subseteq \mathbb{V}^2,$$

where $\boldsymbol{w}^*$ is the vector from $\boldsymbol{v}_A - \boldsymbol{v}_B$ to the closest point on the boundary of $VO^{\tau *}_{A|B}$, and $\boldsymbol{n}^*$ is the outward normal of $\partial VO^{\tau *}_{A|B}$ at the point $\boldsymbol{v}_A - \boldsymbol{v}_B + \boldsymbol{w}^*$.

3.4.4 Kinematic Constraints

Recall from Definition 2.6 that the kinematic constraints of a differential-drive robot are described by

$$\dot{x} = \frac{v_{\text{left}} + v_{\text{right}}}{2} \cos\theta, \tag{3.2}$$

$$\dot{y} = \frac{v_{\text{left}} + v_{\text{right}}}{2} \sin\theta,$$

$$\dot{\theta} = \frac{v_{\text{right}} - v_{\text{left}}}{L}, \tag{3.3}$$

where $L \in \mathbb{R}^+$ is the wheel track of the robot and $v_{\text{left}}, v_{\text{right}} \in \mathbb{V}$ are the signed wheel speeds. Recall also that the wheel speeds are the control inputs of the robot and are bounded such that $v_{\text{left}}, v_{\text{right}} \in [-v_{\max}, v_{\max}] \subseteq \mathbb{V}$. The center of the robot is able to follow any continuous path within the environment, but not instantaneously as it may need to spin to change orientation.

As in Section 2.5.4, we must transform the computed new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ from (3.1) to wheel speeds v_{left} and v_{right}. However, since we would also like to preserve the theoretical guarantee that the trajectories of the robot will be smooth, we must take a different approach. Since the physical center $\boldsymbol{p}_A$ of a differential-drive robot A with radius $r_A \in \mathbb{R}^+$ cannot move in any direction in the workspace $\mathbb{R}^2$ instantaneously, we choose a different reference point $\boldsymbol{q}_A$ and radius $R_A \in \mathbb{R}^+$ for which to perform optimal reciprocal collision avoidance.

Definition 3.7. The *effective center* of a differential-drive robot A is defined as the reference point $\boldsymbol{q}_A = (X, Y)_A \in \mathbb{R}^2$ translated forward a distance $D \in \mathbb{R}^+$ from the physical center $\boldsymbol{p}_A = (x, y)_A \in \mathbb{R}^2$ of the robot in a direction parallel to the wheels of the robot, *i.e.*,

$$\begin{aligned} X &:= x + D\cos\theta, \\ Y &:= y + D\sin\theta. \end{aligned} \tag{3.4}$$

Definition 3.8. The *effective radius* of a differential-drive robot A is the radius

$$R_A := r_A + D \in \mathbb{R}^+$$

of the enlarged disc centered at the effective center $\boldsymbol{q}_A \in \mathbb{V}^2$ that bounds the entire physical extent of the robot.

The effective center and effective radius are shown in Figure 3.3. Unlike the physical center of the robot, the effective center may be translated in a direction orthogonal to the orientation of the wheels of the robot.

Now, substituting (3.2) into (3.4) and applying the chain rule, we have

$$\begin{aligned} \dot{X} &= \left(\frac{\cos\theta}{2} + \frac{D\sin\theta}{L}\right) v_{\text{left}} + \left(\frac{\cos\theta}{2} - \frac{D\sin\theta}{L}\right) v_{\text{right}}, \\ \dot{Y} &= \left(\frac{\sin\theta}{2} - \frac{D\cos\theta}{L}\right) v_{\text{left}} + \left(\frac{\sin\theta}{2} + \frac{D\cos\theta}{L}\right) v_{\text{right}}. \end{aligned} \tag{3.5}$$

This gives us a linear system of the form

$$\boldsymbol{v} = M(\theta) \cdot \boldsymbol{u}, \tag{3.6}$$

where $\boldsymbol{u} = (v_{\text{left}}, v_{\text{right}}) \in \mathbb{V} \times \mathbb{V}$ represents the control inputs, $\boldsymbol{v} = (\dot{X}, \dot{Y}) \in \mathbb{V}^2$ represents the velocity, and $M(\theta) : \mathbb{V} \times \mathbb{V} \to \mathbb{V}^2$ is a two-dimensional matrix. Hence, we can obtain control inputs and wheel speeds v_{left} and v_{right} from a velocity $\boldsymbol{v}$ by solving $\boldsymbol{u} = M^{-1}(\theta) \cdot \boldsymbol{v}$. Note that $D \neq 0$ and $L \neq 0$ by definition, so the matrix $M(\theta)$ is invertible for all $\theta \in \mathbb{S}^1$.

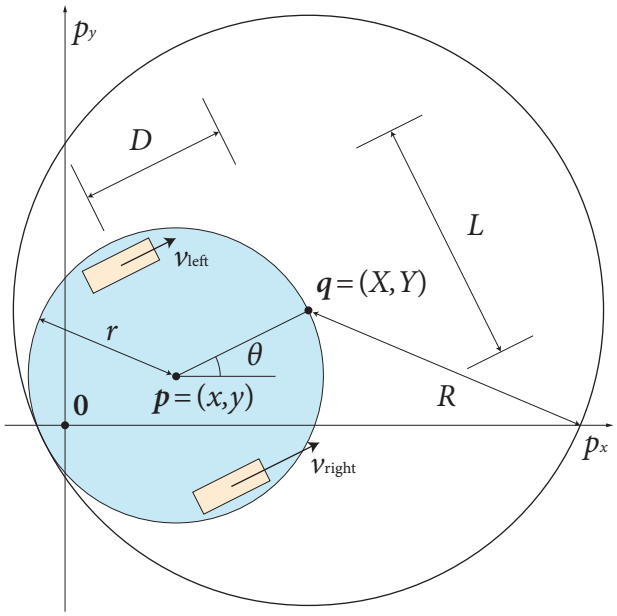


Figure 3.3: The kinematic model of a differential-drive robot with physical center $\boldsymbol{p} = (x, y)$, physical radius r, and wheel track L in the two-dimensional workspace. The effective center $\boldsymbol{q} = (X, Y)$ is located a distance D from $\boldsymbol{p}$ and the effective radius is R. The control inputs are the left and right signed wheel speeds v_{left} and v_{right}, respectively.

Since the control inputs of a differential-drive robot are bounded, $\boldsymbol{u}$ lies within an axis-aligned square S with lower left vertex $(-v_{\max}, -v_{\max})$ and upper right vertex $(v_{\max}, v_{\max})$. Hence, the set of velocities $\boldsymbol{v}$ that the robot can attain is given by the linear transformation $M(\theta) \cdot S$. It follows that if $D = r$, in which case the effective radius is $R = 2r$, then this set of velocities is a square T whose center lies at $\boldsymbol{v} = (0, 0)$ and whose orientation depends on θ. The incircle of T therefore contains the velocities that can be attained regardless of orientation $\theta \in \mathbb{S}^1$. Hence, using the notion of effective center and effective radius, we have following results.

Lemma 3.4. *Given a small time step δt, if the trajectory $\boldsymbol{v}_{\boldsymbol{q}\,A}(t)$ in the velocity space $\mathbb{V}^2$, generated by the sequence of velocities $\boldsymbol{v}_{\boldsymbol{q}\,A} = (\dot{X}, \dot{Y})_A$ at the effective center $\boldsymbol{q}_A = (X, Y)_A$ of a differential-drive robot A at each time step, is continuous,* i.e.,

$$\boldsymbol{v}_{\boldsymbol{q}\,A}(t) \approx \boldsymbol{v}_{\boldsymbol{q}\,A}(t + \delta t),$$

then the trajectory $\boldsymbol{v}_{\boldsymbol{p}\,A}(t)$ in the velocity space $\mathbb{V}^2$, generated by the sequence of velocities $\boldsymbol{v}_{\boldsymbol{p}\,A} = (\dot{x}, \dot{y})_A$ at the physical center $\boldsymbol{p}_A = (x, y)_A$ of the robot, is continuous, i.e.,

$$\boldsymbol{v}_{\boldsymbol{p}\,A}(t) \approx \boldsymbol{v}_{\boldsymbol{p}\,A}(t + \delta t).$$

Proof. If $\boldsymbol{v}_{\boldsymbol{q}\,A}(t)$ is continuous, then by (3.6), the trajectory $M(\theta) \cdot \boldsymbol{u}(t)$ is continuous. Referring to (3.5), it is easy to show by induction on the time step δt that θ is continuous when the robot is initialized as

in the base case of the proof of Theorem 3.1. Therefore, $\boldsymbol{u}(t) = (v_{\text{left}}, v_{\text{right}})(t)$ is continuous. By (3.2), it follows that $\boldsymbol{v}_{p\,A}(t) = (\dot{x}, \dot{y})_A(t)$ is continuous, as required. □

Corollary 3.2 (of Theorem 3.1)**.** *Given a small time step δt, the trajectory $\boldsymbol{v}_{p\,A}(t)$ in the velocity space $\mathbb{V}^2$, generated by the sequence of velocities $\boldsymbol{v}_{p\,A} = (\dot{x}, \dot{y})_A$ at the physical center $\boldsymbol{p}_A = (x, y)_A$ of the robot that are computed using optimal reciprocal collision avoidance with effective center and effective radius at each time step, is continuous,* i.e.*, $\boldsymbol{v}_{p\,A}(t) \approx \boldsymbol{v}_{p\,A}(t + \delta t)$.*

Proof. Follows directly from Theorem 3.1 and Lemma 3.4. □

The guarantee given by Van den Berg, Guy, Lin, *et al.* (2011) that optimal reciprocal collision avoidance generates locally collision-free paths also holds for differential-drive robots navigating using the notion of effective center and effective radius.

Corollary 3.3 (of Van den Berg, Guy, Lin, *et al.*)**.** *Given a small time step δt, the path $\boldsymbol{p}_A(t)$ in the workspace $\mathbb{R}^2$, generated by the sequence of positions $\boldsymbol{p}_A$ of the physical center of a differential-drive robot using optimal reciprocal collision avoidance with effective center and effective radius at each time step, is free of collisions with the paths $\boldsymbol{p}_B(t)$ of every other robot B.*

Proof. Choosing a velocity $\boldsymbol{v}_A$ within the intersection of half-planes $ORCA^{\tau}_A$ ensures that the disc $\mathrm{D}(\boldsymbol{q}_A, R_A)$ of effective center $\boldsymbol{q}_A$ and effective radius R_A is collision free by Van den Berg, Guy, Lin, *et al.* Robot A is always completely contained by this disc, so its path $\boldsymbol{p}_A(t)$ must be free of collisions, as required. □

We can only guarantee a collision-free velocity will be found when $ORCA^{\tau}_A \neq \varnothing$. Potentially, the enlarged, effective radius of each differential-drive robot could make the situation $ORCA^{\tau}_A = \varnothing$ more likely than when using the physical radius conventionally, but this has not been an issue that we have observed in practice.

3.5 Experimentation and Performance

In this section, we describe the implementation of our algorithm and present the results of our experiments with differential-drive robots.

3.5.1 Implementation

We again applied our approach to a set of iRobot Create differential-drive robots, and our experimental setup was similar to that described in Section 2.6.1. The robots were tracked centrally using fiducial markers within a 6 m² indoor space using a Point Grey Flea2 digital video camera connected via FireWire 800 to a notebook computer, and processed using the ARToolKit augmented reality library (Kato and Billinghurst, 1999) to determine the position and orientation of each robot. The velocity of each robot was inferred from the position and orientation measurements using a Kalman filter (Kalman, 1960).

We implemented our approach in C++, and the code was compiled using the Intel C++ Compiler, version 11.1. All calculations were performed on a dual-core 2.53 GHz Intel Core 2 Duo processor within a standard notebook computer containing 4 GB of memory and running Microsoft Windows Vista Ultimate, 64-bit version. Calculations for each robot were carried out in separate and independent threads and in parallel, where possible, using OpenMP, version 3.1. The computed wheel speeds were sent to the robots over a Bluetooth 2.1 virtual serial connection. The velocities of the differential-drive robots were updated at a rate of 30 Hz, limited by the refresh rate of the tracking camera.

In the simulated experiments, only a subset of all other robots within a fixed radius of each differential-drive robot, with respect to Euclidean distance in the workspace $\mathbb{R}^2$, were considered for computation of optimal reciprocal collision avoidance half-planes. These robots were chosen using an algorithm based on k-D trees (De Berg, Cheong, Van Kreveld, *et al.*, 2008).

3.5.2 Experiments

Using the iRobot Create robots and simulated differential-drive robots, we tested our approach in the following experiments.

3.1. Four differential-drive robots are placed at the corners of a rectangular environment. Their goals are to navigate to the corners diagonally opposite. The robots will meet and have to negotiate around each other in the center.

3.2. The center of the environment is blocked by a robot that has malfunctioned and is unable to move. Experiment 3.1 is repeated, but the remaining three differential-drive robots must avoid each other and the malfunctioning robot.

3.3. From ten to one thousand simulated differential-drive robots are spaced evenly on the perimeter of a circular environment. Their goals are to navigate to the antipodal positions on the circle. The robots will meet and have to negotiate around each other in the center.

3.5.3 Discussion

Traces of the differential-drive robots are shown in Figure 3.4 for Experiment 3.1. The paths generated by our algorithm are free of collisions and do not exhibit any oscillations in velocity. Each of the four differential-drive robots makes just enough room for the other robots, resulting in direct paths from the starting position to the goal position with a minimal amount of deviation.

In Experiment 3.2, we demonstrate that our algorithm generates paths that are visibly smooth and free of collisions, as shown in Figure 3.5, even when the direct path from the starting position to the goal position of each differential-drive robot is blocked by the malfunctioning robot.

Table 3.1 and Figure 3.6 show the timing of Experiment 3.3 with an increasing number of simulated differential-drive robots moving across a circle with a radius that has been increased proportionally to the number of robots. Computation time, per time step, increases linearly with

respect to the number of simulated differential-drive robots. Even with as many as one thousand robots, the simulation updates at a rate of more than 300 Hz. This is more than ten times faster than when using the hybrid reciprocal velocity obstacles. All timings are for one thread running on a single processor core.

Table 3.2 and Figure 3.7 show the collisions in Experiment 3.3 with an increasing number of simulated differential-drive robots moving across a circular environment of fixed radius. A small number of collisions are observed as the number of robots exceeds one hundred, around double those of when using hybrid reciprocal velocity obstacles.

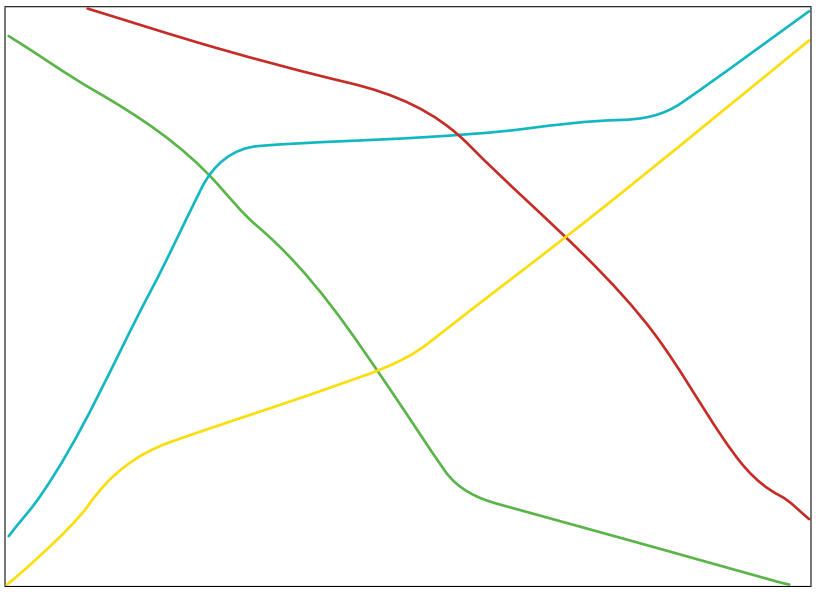


Figure 3.4: Traces of the trajectories of four differential-drive robots navigating simultaneously across a rectangular environment in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius (Experiment 3.1).

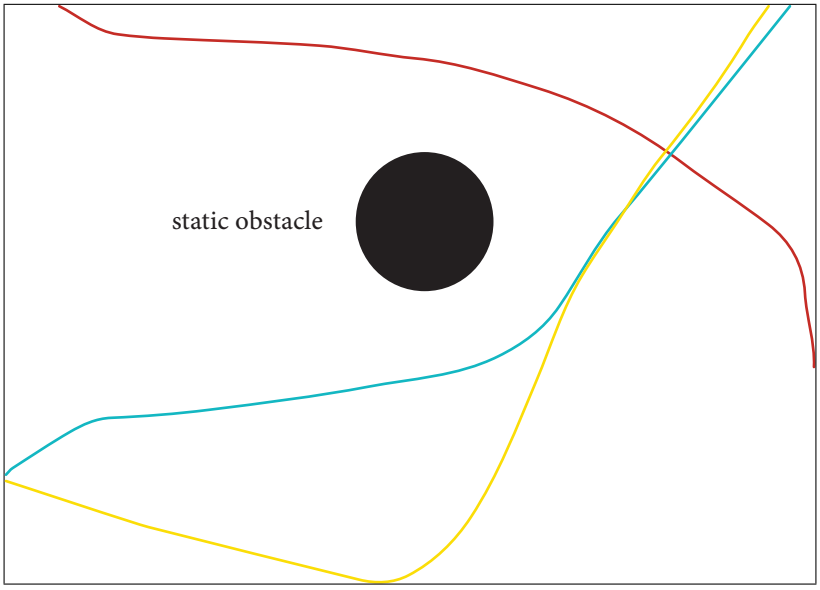


Figure 3.5: Traces of the trajectories of three differential-drive robots navigating simultaneously across a rectangular environment containing a static, malfunctioning robot (black disc) in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius (Experiment 3.2).

Table 3.1: Average computation time at each time step on one core of a 2.53 GHz Intel Core 2 Duo processor, in milliseconds, for differential-drive robots navigating across a circular environment of increasing radius in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius (Experiment 3.3).

Number of differential-drive robots	Computation time per time step (ms)
10	0.1
100	0.4
200	0.6
300	1.0
400	1.2
500	1.5
1000	3.0

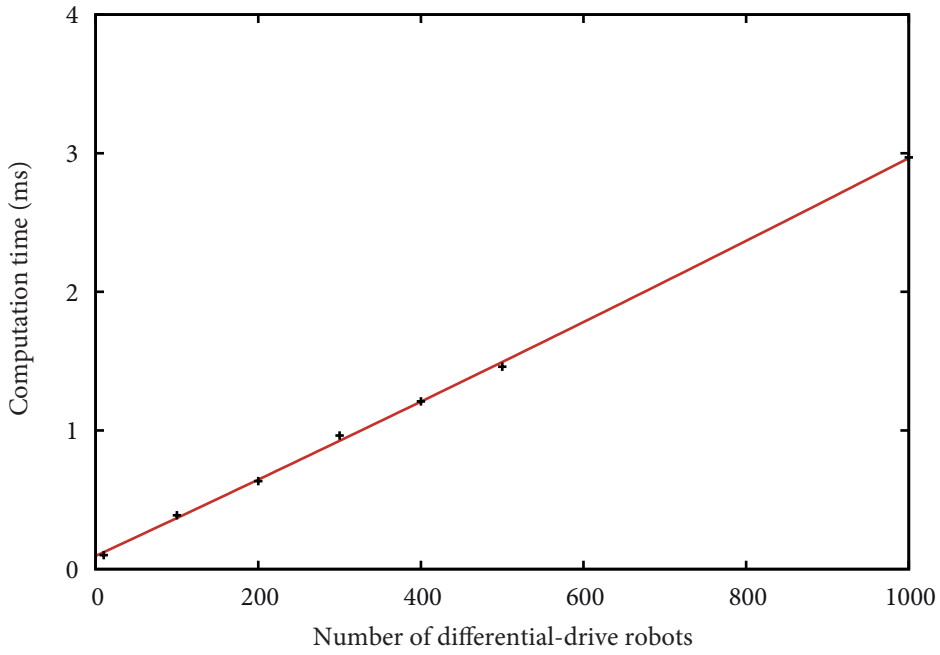


Figure 3.6: A plot of the average computation time at each time step on one core of a 2.53 GHz Intel Core 2 Duo processor, in milliseconds, for differential-drive robots navigating across a circular environment of increasing radius in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius (Experiment 3.3).

Table 3.2: Average number of collisions during each time step between differential-drive robots navigating across a circular environment of fixed radius in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius (Experiment 3.3).

Number of differential-drive robots	Number of collisions per time step
10	0
100	0.3
200	1.4
300	3.3
400	5.6
500	9.4
1000	28.4

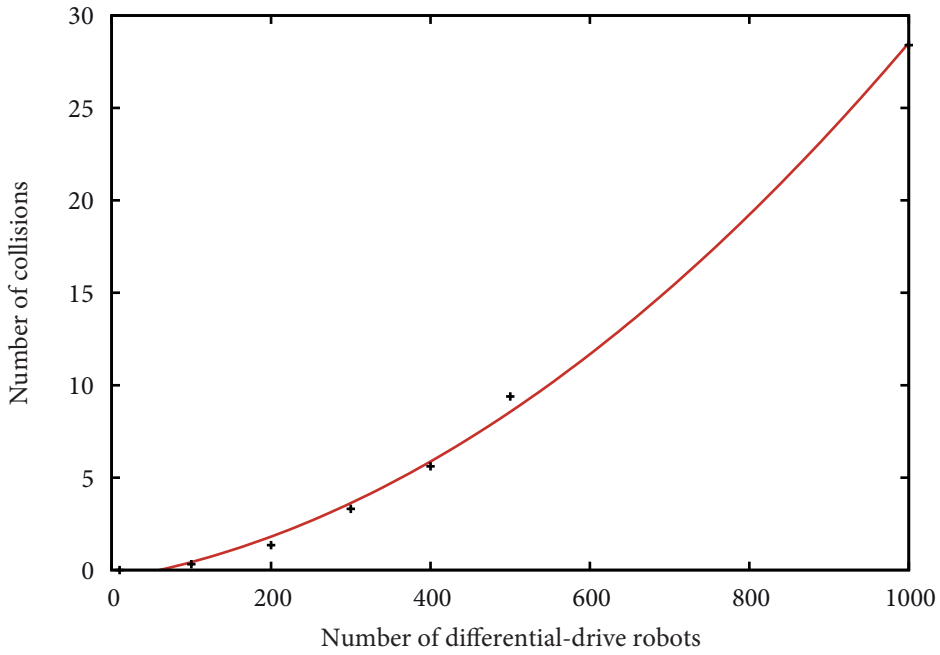


Figure 3.7: A plot of the average number of collisions during each time step between differential-drive robots navigating across a circular environment of fixed radius in the two-dimensional workspace using optimal reciprocal collision avoidance with effective center and effective radius (Experiment 3.3).

Chapter 4

NAVIGATING MULTIPLE SIMPLE-AIRPLANES IN THREE-DIMENSIONAL WORKSPACE

4.1 Introduction and Motivation

While there is extensive work on the navigation and coordination of multiple mobile robots, most prior methods are either limited to robots moving in the two-dimensional workspace or do not take into account the kinematic and dynamic constraints on the motion of the robots. Moreover, many methods focus only on task allocation or precompute the entire path of each robot, rather than perform dynamic collision avoidance and navigation.

We present a method for navigating multiple airplanes with kinematic and dynamic constraints in the three-dimensional workspace among dynamic obstacles. We use a simplified kinematic and dynamic model for each airplane based on the two-dimensional simple car model (Latombe, 1991; Laumond, Sekhavat, and Lamiraux, 1998; LaValle, 2006). Instead of fixing either the speed (Chitsaz and LaValle, 2007) or altitude (Richards and How, 2002) of the airplane, we allow both quantities to vary continuously and refer to the resulting kinematic model as a "simple-airplane." By extending the optimal reciprocal collision avoidance algorithm (Van den Berg, Guy, Lin, *et al.*, 2011) to the three-dimensional workspace, we compute collision-free and oscillation-free motion for each simple-airplane among dynamic obstacles and each other. Moreover, we compute the trajectory of each simple-airplane independently and assume no centralized coordination of the simple-airplanes.

Instead of distributing the load equally among the simple-airplanes for collision avoidance, fifty percent for each, we introduce the notion of "variable reciprocity" between pairs of simple-airplanes. Informally, variable reciprocity assigns a higher responsibility for avoiding collisions to a simple-airplane that has fewer constraints in terms of choosing its velocity. We deal with the kinematic and dynamic constraints of each simple-airplane by sampling velocities from a reduced set of kinematic and dynamic constraints, and then satisfy the remaining kinematic constraints by enumerating a set of precomputed curves that correspond to solutions of the equations that define the kinematic model of the simple-airplane.

We have implemented our method and performed experiments that simulate up to sixteen simple-airplanes sharing an environment. Our experimental results show that our method computes trajectories that are observed to be both free of collisions and free of oscillations while satisfying the kinematic and dynamic constraints of each simple-airplane even in the presence of dynamic obstacles.

4.2 Prior Work

The seminal work Dubins (1957) describes the kinematic constraints of a simplified car that is restricted to forward motions with a fixed constant speed within a bounded turning radius. Later work has generalized the Dubins car model into the model of the simple car (LaValle, 2006) by adding a reverse gear (Boissonnat, Cérézo, and Leblond, 1994; Reeds and Shepp, 1990; Sussmann and Tang, 1991) and variable speed in any direction subject to a maximum steering angle (Latombe, 1991; Laumond, Sekhavat, and Lamiraux, 1998).

The Dubins airplane (Chitsaz and LaValle, 2007) extends the Dubins car into three dimensions with the addition of a varying altitude. While its airspeed remains constant, the rate of change of altitude of a Dubins airplane may vary continuously. Other simplified models for airplanes, *e.g.*, Richards and How (2002), choose to fix the altitude and allow the airspeed of the airplane to vary. It is far less common to allow the altitude as well as the airspeed to vary, particularly when navigating multiple airplanes in a shared environment.

Research in the coordination and navigation of multiple airplanes has mainly concentrated on efficient task allocation, *e.g.*, Richards, Bellingham, Tillerson, *et al.* (2002), or management of air traffic control (Arkin, Mitchell, and Polishchuk, 2010; Chiang, Klosowski, Lee, *et al.*, 1997) rather than local collision avoidance. However, there exists a large amount of work on collision-free navigation in the two-dimensional workspace in the context of multiple robots or virtual agents, as described in Sections 2.2 and 3.2.

4.3 Navigating Multiple Simple-Airplanes

In this section, we describe the kinematic and dynamic constraints of the simple-airplane model and outline our approach for navigating multiple simple-airplanes using optimal reciprocal collision avoidance with variable reciprocity.

4.3.1 Overall Approach

We transfer the problem description and notation of Section 2.3 from mobile robots or virtual agents in the two-dimensional workspace $\mathbb{R}^2$ to simplified airplanes in the three-dimensional workspace $\mathbb{R}^3$. We have a nonempty set $\mathcal{S}$ of sphere-bounded airplanes and, possibly, a set $\mathcal{O}$ of dynamic obstacles sharing an environment in the three-dimensional workspace $\mathbb{R}^3$ and the three-dimensional velocity space $\mathbb{V}^3$. Each simplified airplane $A \in \mathcal{S}$ has a fixed bounding radius $r_A \in \mathbb{R}^+$, a current position $\boldsymbol{p}_A \in \mathbb{R}^3$, and a current velocity $\boldsymbol{v}_A \in \mathbb{V}^3$, as well as a point goal located at $\boldsymbol{p}_A^{\text{goal}} \in \mathbb{R}^3$ and a preferred speed $v_A^{\text{pref}} \in \mathbb{V}^+$. Each airplane has "simple-airplane" kinematic and dynamic constraints on its motion, including a minimum speed that it must exceed at all times.

Algorithm 4.1 outlines our overall approach. Initially, a simple-airplane acquires its own position and velocity, and those of surrounding simple-airplanes. It also estimates the kinematic and dynamic constraints of the other simple-airplanes based on their prior motion over several time steps. The set of velocities that correspond to a reduced set of kinematic and dynamic constraints are

Algorithm 4.1: Our method for navigating multiple simple-airplanes in the three-dimensional workspace using optimal reciprocal collision avoidance with variable reciprocity.

inputs
  List of simple-airplanes $\mathcal{S} \neq \varnothing$
  List of dynamic obstacles $\mathcal{O}$
for all $A \in \mathcal{S}$ **do**
  Sample full constraint space U_A and precompute curves $\gamma_A \in \Gamma_A$
end for
loop
  for all $A \in \mathcal{S}$ **do**
    Sense $\boldsymbol{p}_A$ and $\boldsymbol{v}_A$
    Estimate reduced constraint space U_A^* and velocities V_A^*
    for all $B \in \mathcal{S}$ **such that** $A \neq B$ **do**
      Sense $\boldsymbol{p}_B$ and $\boldsymbol{v}_B$
      Construct $VO_{A|B}^{\tau}$
      Estimate reduced constraint space U_B^* and velocities V_B^*
      Compute reciprocity factor $\rho_{A|B}$ from $\mu(V_A^*)$ and $\mu(V_B^*)$
      Construct $ORCA_{A|B}^{\tau,\rho}$
    end for
    for all $O \in \mathcal{O}$ **do**
      Sense $\boldsymbol{p}_O$ and $\boldsymbol{v}_O$
      Construct $VO_{A|O}^{\tau}$
      Construct $ORCA_{A|O}^{\tau,\rho=1}$
    end for
    Construct $ORCA_A^{\tau}$ from all $ORCA_{A|B}^{\tau,\rho}$ and $ORCA_{A|O}^{\tau,\rho=1}$
    Compute preferred velocity $\boldsymbol{v}_A^{\text{pref}}$
    Sample velocities $\boldsymbol{v}_A^* \in V_A^*$ and rank by minimum distance from $\boldsymbol{v}_A^{\text{pref}}$ in $\mathbb{V}^3$
    Enumerate $\gamma_A \in \Gamma_A$ to find curve γ_A^{new} corresponding to highest ranked $\boldsymbol{v}_A^*$ that satisfies full constraint space U_A
    Take path given by γ_A^{new}
  end for
end loop

then computed for each simple-airplane based on these estimates. Meanwhile, each simple-airplane computes its preferred velocity and constructs the truncated velocity obstacles induced by the other simple-airplanes within the window of time $[0, \tau]$.

The reciprocity factor of the simple-airplane with respect to each other simple-airplane is calculated based on their estimated constraints, followed by the optimal reciprocal collision avoidance half-spaces. The collision-free velocities are then selected from the half-spaces by sampling, and they are ranked by the shortest distance from the preferred velocity, with respect to Euclidean distance in the velocity space $\mathbb{V}^3$.

Finally, the remaining kinematic constraints that are not used in the calculation of reduced set of kinematic and dynamic constraints are satisfied by enumerating a set of precomputed curves that are a subset of all valid paths defined by those constraints. If the highest ranked velocity, or one within a small threshold, can be attained by taking a path defined by one of the precomputed

curves, then that velocity is valid, and the simple-airplane takes that path. Otherwise, the lower ranked velocities are tested in turn against the curves until a match is found.

4.3.2 Kinematic and Dynamic Constraints

We base the kinematic constraints of a simple-airplane in the workspace $\mathbb{R}^3$ on those of a simple car (Latombe, 1991; Laumond, Sekhavat, and Lamiraux, 1998; LaValle, 2006) in the workspace $\mathbb{R}^2$. The simple car is characterized by four wheels arranged in pairs on two common axes, front and rear, separated by a nonzero wheelbase. The front wheels steer to adjust the orientation of the simple car, and the rear wheels are fixed, and drive the simple car forward and in reverse. The control inputs of the simple car are its signed rear wheel speed, positive for forward and negative for reverse, and steering angle, both of which may vary continuously within some specified bounds.

Definition 4.1. The *kinematic constraints of a simple car* with position $(x, y) \in \mathbb{R}^2$, orientation $\theta \in \mathbb{S}^1$, and unit wheelbase are described by

$$\begin{aligned} \dot{x} &= s \cos\theta, \\ \dot{y} &= s \sin\theta, \\ \dot{\theta} &= s \tan\phi, \end{aligned}$$

where s is the signed rear wheel speed, and ϕ is the steering angle. The control inputs s and ϕ are bounded, such that

$$\begin{aligned} -s^{\max} \leq s \leq s^{\max}, \\ -\phi^{\max} \leq \phi \leq -\phi^{\max}, \end{aligned}$$

for $s^{\max} > 0$ and $\phi^{\max} < \frac{1}{2}\pi$.

In contrast to the differential-drive robot of Section 2.5.4, the individual wheels of a simple car cannot be driven at different speeds independently. The simple car is also unable to follow any continuous path within the workspace $\mathbb{R}^2$, instantaneously or otherwise.

In an analogous manner to Chitsaz and LaValle (2007) for the Dubins airplane, we extend the simple car into the workspace $\mathbb{R}^3$ by adding varying altitude in the direction of the z-axis. For simplicity, we ignore pitch and roll rotations and other disturbances, so we assume that the simple-airplane remains parallel to the xy-plane at all times, as indicated in Figure 4.1. Therefore, the airspeed refers the speed of the simple-airplane relative to the x-axis and y-axis, and yaw and steering angle refer to angles relative to the x-axis and y-axis only. The rate of climb refers to the speed of the simple-airplane parallel to xz-plane and yz-plane.

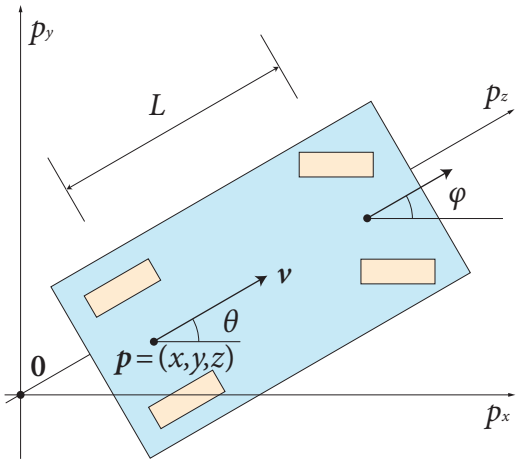


Figure 4.1: The kinematic model of a simple-airplane with unit wheelbase in the three-dimensional workspace. The control inputs are signed speeds s_{xy} and s_z, and steering angle ϕ.

Definition 4.2. The *kinematic constraints of a simple-airplane* with position $(x, y, z) \in \mathbb{R}^3$ and yaw $\theta \in \mathbb{S}^1$ are described by

$$\begin{aligned} \dot{x} &= s_{xy} \cos\theta, \\ \dot{y} &= s_{xy} \sin\theta, \\ \dot{z} &= s_z, \\ \dot{\theta} &= s_{xy} \tan\phi, \end{aligned} \tag{4.1}$$

where $(\dot{x}, \dot{y})$ is the airspeed, $\dot{y}$ is the signed rate of climb, and $\dot{z}$ is the yaw rate. The control inputs are $\boldsymbol{u} = (s_{xy}, s_z, \phi)$, and are bounded, such that

$$\begin{aligned} s_{xy}^{\min} \leq s_{xy} \leq s_{xy}^{\max}, \\ -s_z^{\max} \leq s_z \leq s_z^{\max}, \\ -\phi^{\max} \leq \phi \leq \phi^{\max}, \end{aligned}$$

for $s_{xy}^{\max} > s_{xy}^{\min} > 0$, $s_z^{\max} > 0$, and $\phi^{\max} < \pi/2$.

Clearly, the system may control the airspeed and yaw rate independently from the rate of climb. Note also that a simple-airplane cannot stop, or hover, or travel in reverse. We denote the three-dimensional space of all permissible control inputs by U and denote the space of all velocities that may be attained by choosing a control input in U by V. In a similar manner to Scheuer and Laugier (1998), we also define the following constraints.

Definition 4.3. The *dynamic constraints of a simple-airplane* with control inputs $\boldsymbol{u} = (s_{xy}, s_z, \phi)$ are described by

$$-a_{xy} \leq \dot{s}_{xy} \leq a_{xy},$$
$$-a_z \leq \dot{s}_z \leq a_z,$$
$$-a_\phi \leq \dot{\phi} \leq a_\phi,$$

where $a_{xy}, a_z, a_\phi > 0$.

In other words, a simple-airplane cannot increase or decrease its airspeed or rate of climb arbitrarily fast; neither can it adjust its steering angle discontinuously. We denote the three-dimensional space of permissible changes in control inputs under these constraints by $\dot{U}$.

4.3.3 Reduced Constraints and Precomputed Curves

We use a two-stage approach to compute a velocity that satisfies the kinematic and dynamic constraints of a simple-airplane A. First, we satisfy a reduced set of constraints U_A^* that are easier to compute, and then fit a set of curves Γ_A that satisfy the full constraints, and can be precomputed before navigation or simulation commences.

For the reduced constraints, we consider a window of time $[0, \tau]$ and define a set of control inputs that are valid at time $t + \tau$ based on the control inputs $\boldsymbol{u}_A(t)$ of the simple-airplane at time t by

$$U_A^*(t + \tau) := \left\{ \boldsymbol{u}_A(t) + \dot{\boldsymbol{u}}\tau \;\middle|\; \dot{\boldsymbol{u}} \in \dot{U}_A \right\}.$$

Shown in Figure 4.2, we denote by V_A^* the space of attainable velocities corresponding to U_A^*.

While the space U_A^* depends on time t, and must be calculated at each time step, the set of curves Γ_A needs to be computed only once. Each curve $\gamma_A \in \Gamma_A$, where

$$\gamma_A : [0, 1] \to \mathbb{R}^3,$$

corresponds to the path taken when a simple-airplane performs action $\boldsymbol{u}_A$ for time τ, and we populate the set by uniformly sampling the range of valid values for each of s_{xy}, s_z, and ϕ. For each combination of the three control inputs, we calculate the values $\dot{x}$, $\dot{y}$, $\dot{z}$, and $\dot{\theta}$ by substituting directly into (4.1). The change in velocity of the simple-airplane between the start and the end of each curve, *i.e.*,

$$\delta \boldsymbol{v}_A^\gamma = \boldsymbol{v}^{\gamma=1} - \boldsymbol{v}^{\gamma=0},$$

is precomputed and used in the last stage of our algorithm at each time step. We precompute approximately 10^5 curves.

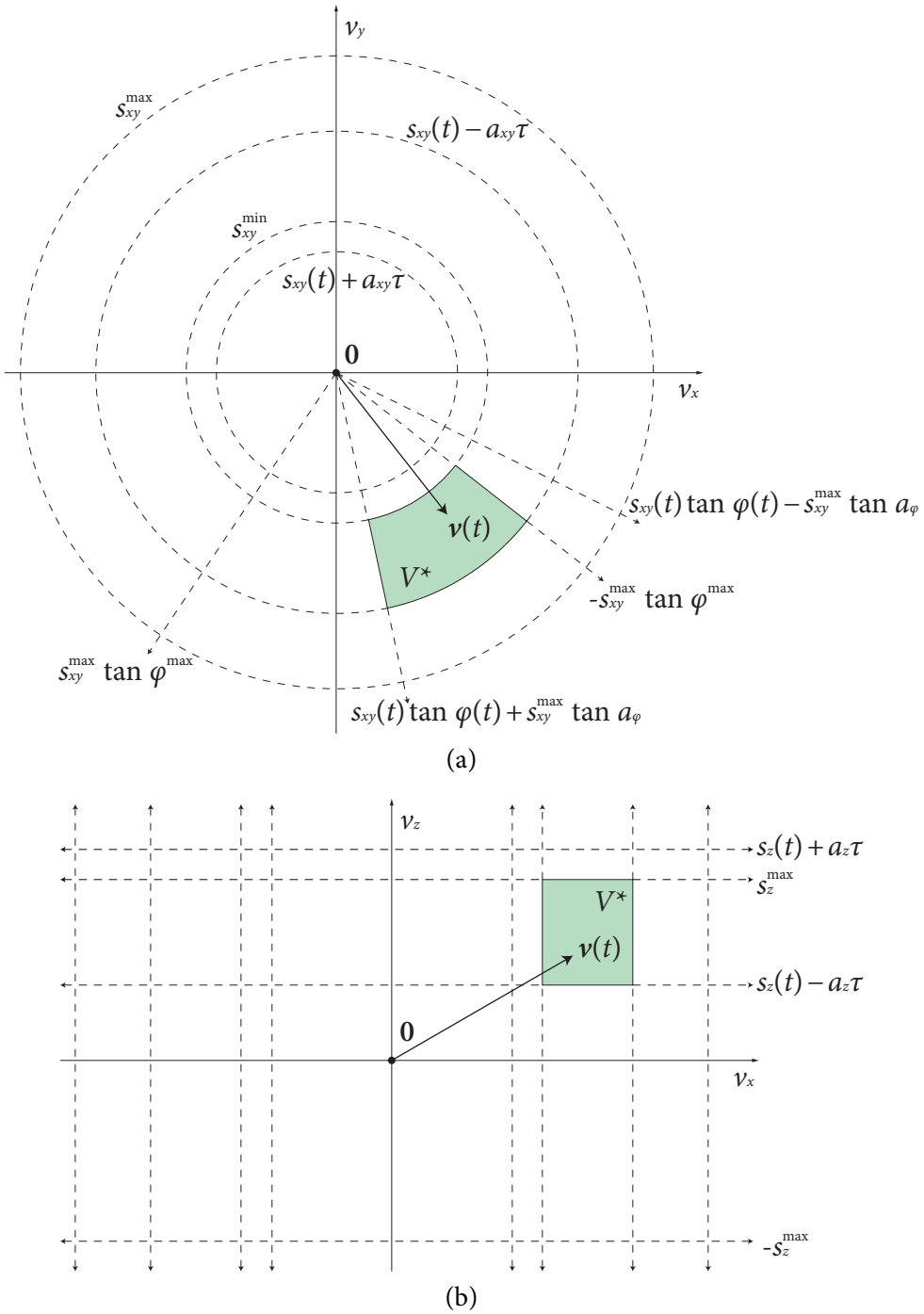


Figure 4.2: Space of velocities V^* (shaded green) corresponding to the reduced kinematic and dynamic constraints U^* of a simple-airplane from which the potential new velocities are sampled, in (a) the xy-plane and (b) the xz-plane in the three-dimensional velocity space.

4.3.4 Optimal Reciprocal Collision Avoidance with Variable Reciprocity

Intuitively, a simple-airplane that has a larger space of attainable velocities V, or for ease of calculation, a larger reduced space of attainable velocities V^*, is less constrained in terms of deviating from its current path and plays a larger role in terms of taking more responsibility for avoiding a collision than a simple-airplane that has a small choice of attainable velocities. Recall Definition 3.3 of the optimal reciprocal collision avoidance half-plane in the velocity space $\mathbb{V}^2$. In the velocity space $\mathbb{V}^3$, we have half-spaces instead of half-planes.

Definition 4.4. The *optimal reciprocal collision avoidance half-space* of simple-airplane A induced by simple-airplane B within the window of time $[0, \tau]$ is defined as

$$ORCA^{\tau}_{A|B} := \left\{ \boldsymbol{v} \;\middle|\; (\boldsymbol{v} - (\boldsymbol{v}_A + \tfrac{1}{2}\boldsymbol{w})) \cdot \boldsymbol{n} \geq 0 \right\} \subseteq \mathbb{V}^3, \tag{4.2}$$

where $\boldsymbol{w}$ is the vector from $\boldsymbol{v}_A - \boldsymbol{v}_B$ to the closest point on the boundary of $VO^{\tau}_{A|B} \subseteq \mathbb{V}^3$, and $\boldsymbol{n}$ is the outward normal of $\partial VO^{\tau}_{A|B}$ at the point $\boldsymbol{v}_A - \boldsymbol{v}_B + \boldsymbol{w}$.

Note the term $\frac{1}{2}\boldsymbol{w}$ in (4.2). If we replace this by $\rho_{A|B}\,\boldsymbol{w}$ for a suitable $\rho_{A|B} \in [0, 1]$, we can express the notion of "variable reciprocity."

Definition 4.5. The *optimal reciprocal collision avoidance half-space* of simple-airplane A induced by simple-airplane B within the window of time $[0, \tau]$ with *variable reciprocity* is defined as

$$ORCA^{\tau,\rho}_{A|B} := \left\{ \boldsymbol{v} \;\middle|\; (\boldsymbol{v} - (\boldsymbol{v}_A + \rho_{A|B}\,\boldsymbol{w})) \cdot \boldsymbol{n} \geq 0 \right\} \subseteq \mathbb{V}^3,$$

where $\rho_{A|B}$ is the reciprocity factor.

The value of the reciprocity factor is based on the relative volumes $\mu(V^*_A)$ and $\mu(V^*_B)$ in the velocity space $\mathbb{V}^3$ of the spaces of velocities V^*_A and V^*_B, respectively.

Definition 4.6. The *reciprocity factor* of simple-airplane A with respect to simple-airplane B is defined as

$$\rho_{A|B} := \frac{\mu(V^*_A)}{\mu(V^*_A) + \mu(V^*_B)},$$

where $\mu(V^*)$ denotes the volume of V^* in the velocity space $\mathbb{V}^3$.

The reciprocity factor $\rho_{A|B}$ is illustrated in Figure 4.3, and is, informally, a measure of the amount of responsibility that a simple-airplane A will take to avoid another simple-airplane B. A high value of $\rho_{A|B}$ denotes that simple-airplane A is less constrained than simple-airplane B and is able to take a large amount of responsibility for avoiding a collision; a low value represents the opposite. The sum of reciprocity factors for any two simple-airplanes always should be equal to one, *i.e.*, $\rho_{A|B} + \rho_{B|A} = 1$.

Apart from this change, we construct the optimal reciprocal collision avoidance half-spaces in an analogous way to the formulation in the two-dimensional velocity space $\mathbb{V}^2$. However, since the velocities reachable during a window of time $[0, \tau]$ are restricted by the kinematic and dynamic

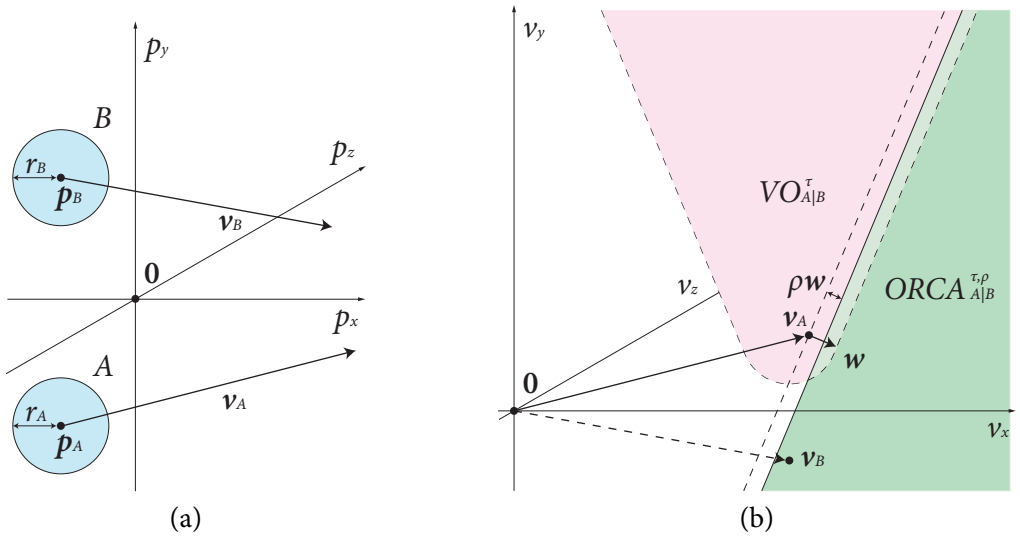


Figure 4.3: (a) Two disc-bounded simple-airplanes A and B with positions $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$, velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$, and bounding radii r_A and r_B, respectively, in the three-dimensional workspace. (b) The optimal reciprocal collision avoidance half-space $ORCA^{\tau,\rho}_{A|B}$ of simple-airplane A induced by simple-airplane B within the finite window of time $[0, \tau]$ with variable reciprocity ρ (shaded green) in the three-dimensional velocity space. Constant ρ is the reciprocity factor.

constraints of a simple-airplane, we search for collision-free velocities in the velocity space $\mathbb{V}^3$ by sampling the intersection $V^*_A \cap ORCA^{\tau,\rho}_A$. After finding these velocities, we rank them by the shortest distance from the preferred velocity, with respect to Euclidean distance in the velocity space $\mathbb{V}^3$.

The final task is to satisfy the remaining kinematic constraints that do not correspond to velocities in the reduced space of velocities V^*_A. We enumerate each precomputed curve $\gamma_A \in \Gamma_A$, comparing the velocity $\boldsymbol{v}_A + \delta\boldsymbol{v}^{\gamma}_A$ obtained at the end of the curve with our highest ranking velocity $\boldsymbol{v}^*_A$. If there exists a curve $\gamma_A = \gamma^{\text{new}}_A \in \Gamma_A$ that allows $\boldsymbol{v}^*_A$, or a velocity within some acceptable small threshold of $\boldsymbol{v}^*_A$, to be attained, then the velocity $\boldsymbol{v}^*_A$ is valid, and the simple-airplane chooses the path γ^{new}_A to make collision-free progress towards its goal. If not, we select a lower ranked velocity and repeat the process, continuing until we find a valid combination of velocity $\boldsymbol{v}^*_A$ and path γ_A.

4.4 Experimentation and Performance

In this section, we describe the implementation of our algorithm and present the results of our simulated experiments.

4.4.1 Implementation

We implemented our approach in C++, and the code was compiled using the GCC compiler, version 4.2. All calculations were performed on a dual-core 2.53 GHz Intel Core 2 Duo processor within a standard notebook computer containing 4 GB of memory and running Mac OS X Snow Leopard, version 10.6.1. Calculations for each simple-airplane were carried out in separate and independent threads, and in parallel, where possible, using the Grand Central Dispatch library. For efficiency

reasons, only a subset of all other simple-airplanes within a fixed radius of each simple-airplane, with respect to Euclidean distance in the workspace $\mathbb{R}^3$, were considered for collision avoidance, and these simple-airplane were selected at the beginning of every time step using an algorithm based on k-D trees (De Berg, Cheong, Van Kreveld, *et al.*, 2008).

4.4.2 Experiments

We applied our approach to experiments containing up to sixteen simple-airplanes as follows.

4.1. Four to sixteen simple-airplanes are spaced evenly around the equator of a spherical environment. Their goals are to navigate to the antipodal positions on the sphere. The simple-airplanes will meet and have to negotiate around each other in the center.

4.2. One to four dynamic obstacles move through a rectangular environment at a constant velocity. Three to twelve simple-airplanes have to cross the path of the dynamic obstacles to navigate to their goals.

We assume that each simple-airplane has full knowledge of the kinematic and dynamic constraints of the other simple-airplanes and can easily identify a dynamic obstacle from a cooperating simple-airplane. Videos of these experiments are available online at `http://gamma.cs.unc.edu/S-AIRPLANE/`.

4.4.3 Discussion

Traces of four simple-airplanes in Experiment 4.1 are shown in Figure 4.4. The paths computed by the simple-airplanes are collision-free and contain no oscillations. They are smooth and direct, with no sudden changes in the direction of the simple-airplanes that would indicate that their kinematic constraints have been violated. The simple-airplanes are not restricted to a fixed two-dimensional plane and change altitude when and where necessary, clearly using the positions and velocities of the other simple-airplanes to plan their path.

In Figure 4.6, Experiment 4.2 demonstrates that our method can deal with an entity that is entirely constrained and unable, or unwilling, to adjust its motion due to the proximity of simple-airplanes. Here, the three simple-airplanes detect that the region of attainable velocities of the single dynamic obstacle is a point and of zero volume, naturally obtaining a reciprocity factor $\rho = 1$ that corresponds to taking full responsibility for avoiding a collision without requiring the addition of a special case to our formulation. The dynamic obstacle nominally has a reciprocity factor $\rho = 0$, indicating that it takes no responsibility for avoiding each simple-airplane.

Expanded versions of Experiment 4.1, navigating sixteen simple-airplanes, and Experiment 4.2, navigating twelve simple-airplanes around four dynamic obstacles, are shown in Figures 4.5 and 4.7, respectively, and in the online video.

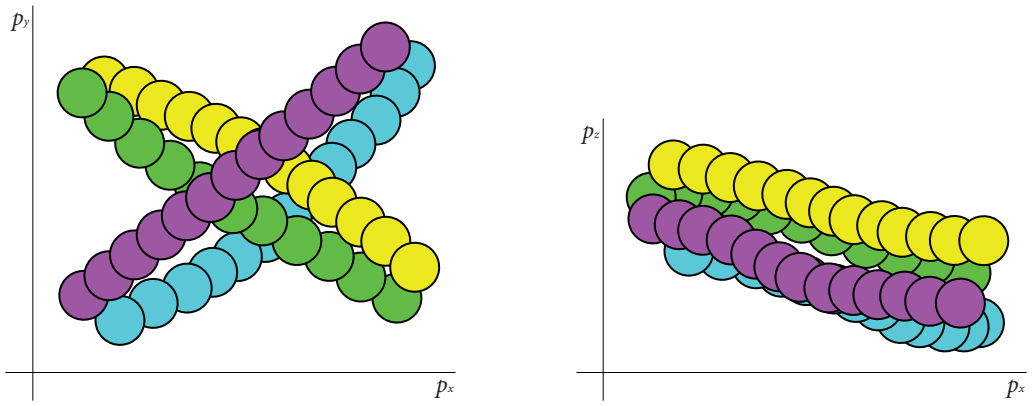


Figure 4.4: Traces of the trajectories of four simple-airplanes navigating simultaneously across a spherical environment in the three-dimensional workspace using optimal reciprocal collision avoidance with variable reciprocity (Experiment 4.1) in (a) the xy-plane and (b) the xz-plane. Positions of simple-airplanes every ten time steps are shown with a disc, later positions drawn on top of earlier positions. The trajectory of each simple-airplane is a different color.

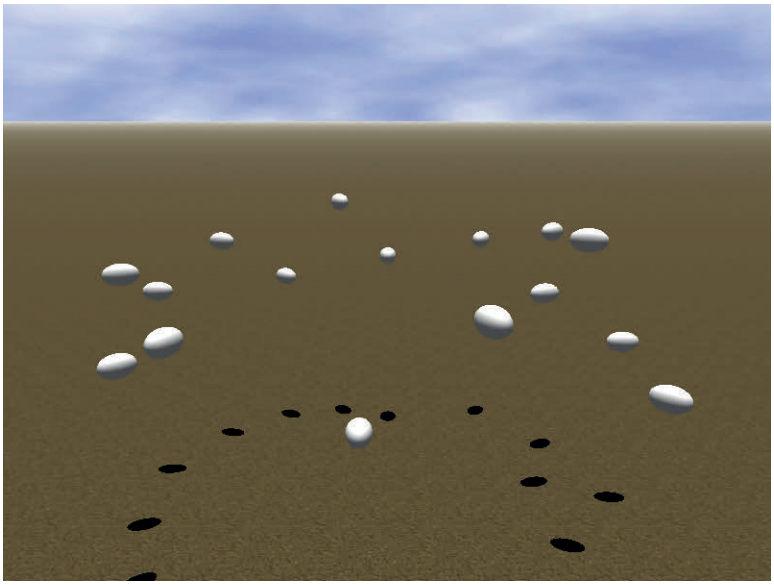


Figure 4.5: A screenshot of sixteen simple-airplanes (small white ellipses) navigating simultaneously across a spherical environment in the three-dimensional workspace using optimal reciprocal collision avoidance with variable reciprocity (Experiment 4.1).

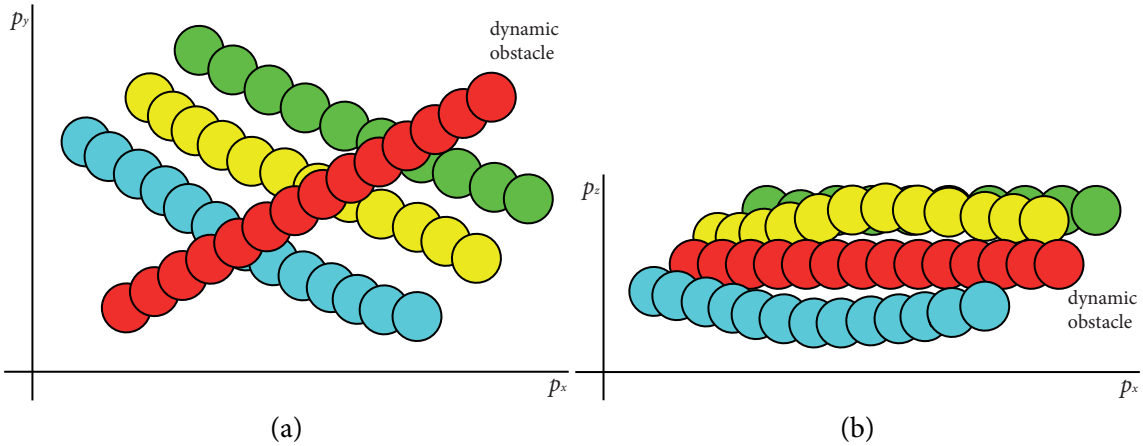


Figure 4.6: Traces of the trajectories of three simple-airplanes navigating simultaneously across a rectangular environment containing a dynamic obstacle in the three-dimensional workspace using optimal reciprocal collision avoidance with variable reciprocity (Experiment 4.2) in (a) the xy-plane and (b) the xz-plane. Positions of simple-airplanes every ten time steps are shown with a disc, later positions drawn on top of earlier positions. The trajectory of each simple-airplane is a different color, with red discs corresponding to the trajectory of the dynamic obstacle.

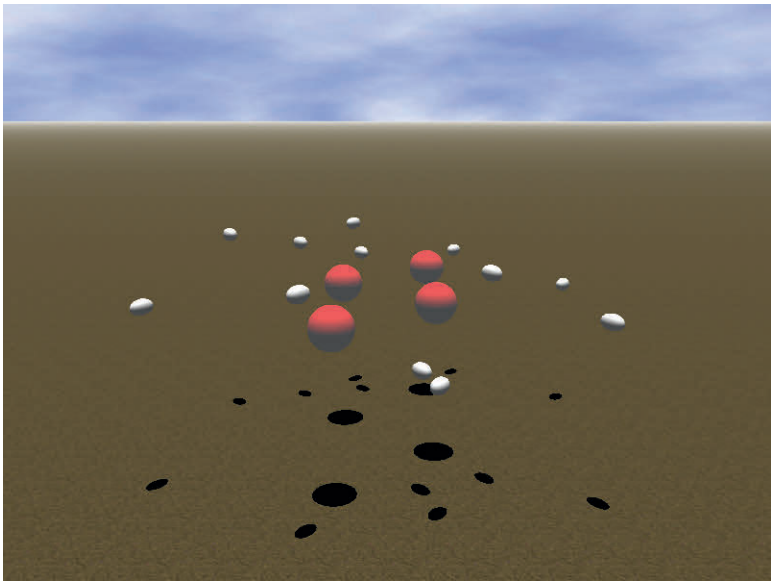


Figure 4.7: A screenshot of twelve simple-airplanes (smaller white ellipses) navigating simultaneously across a spherical environment containing four dynamic obstacles (larger red spheres) in the three-dimensional workspace using optimal reciprocal collision avoidance with variable reciprocity (Experiment 4.2).

Chapter 5

GOAL VELOCITY OBSTACLES FOR SPATIAL NAVIGATION OF MULTIPLE VIRTUAL AGENTS

5.1 Introduction and Motivation

The spatial navigation of groups of virtual agents to specified goal locations is an important component of many video games and simulated environments. Large numbers of virtual agents may be incorporated into game levels and simulated environments, and, often, these must interact with agents controlled by a player. Efficient collision avoidance algorithms that can react to a dynamic environment are particularly important in these circumstances since the virtual agents must adapt their motion to the unpredictable actions of the human. A human-controlled agent may also represent the goal location of a group of pursuing virtual agents. In this case, the group of virtual agents must be able to converge on a possibly moving, planar goal region that is the footprint of the human-controlled agent.

In previous work, such as Van den Berg, Lin, and Manocha (2008); Fiorini and Shiller (1998), each virtual agent chooses an avoiding new velocity based on some optimization to make progress toward its goal. Commonly, this optimization is on the preferred velocity (Definition 2.4), which may be directed to a roadmap node (Canny, 1988) or a fixed point in the center of a navigation mesh edge or face (Snook, 2000). However, often these points approximate planar goal regions, and this contraction of the goal region to a point can cause artifacts, such as collisions when several virtual agents converge on a single point. Behavior would be improved if the virtual agent could navigate to any point in a goal region. However, with limited exceptions (Curtis, Snape, and Manocha, 2012), most velocity-based methods are simply coupled with a cluster of point goals and a goal allocation algorithm that chooses the point goal of a virtual agent based on some heuristic. When goal regions are moving, optimizing on a preferred velocity ignores that the position of the goal region may have changed significantly by the time the virtual agent has computed a new velocity. Hence, the trajectory of the virtual agent will not necessarily be directed toward its goal region, and the lengths of paths to the goal region will be increased. If the velocity of the goal region were considered during the optimization of the velocity, then the motion of the virtual agent toward its goal region would again be improved.

We introduce the "goal velocity obstacle" for navigating multiple virtual agents to planar, spatial goal regions that counters the above-described disadvantages of formulations that optimize on preferred velocity. The basic idea is that instead of only using the notion of truncated velocity obstacles

(Definition 3.1) to compute collision-avoiding velocities, we also use them to define the goal regions of the virtual agent within velocity space. We call the truncated velocity obstacle of a virtual agent induced by its goal region a "goal velocity obstacle," and if the virtual agent chooses a velocity that is inside the goal velocity obstacle at each time step, then it will eventually reach its goal region.

The goal velocity obstacle provides a unified formulation that allows for goals specified as points, line segments, and bounded, planar regions in two dimensions that may be static or moving. A virtual agent navigating using goal velocity obstacles may have multiple goal regions without requiring an explicit goal allocation algorithm that would choose a particular goal region to navigate toward in advance. Goal regions may also have a time dependency, such that the goal region is only available to a virtual agent during a specific window of time, without requiring an explicit scheduling algorithm.

In experiments with hundreds of virtual agents, those navigating using goal velocity obstacles toward static, moving, and multiple goal regions have shorter path lengths from their starting positions to their goal regions and fewer collisions with other virtual agents, than when using velocity-based methods that optimize on a single preferred velocity toward the goal of each virtual agent. The additional computational overhead is just a few microseconds, per virtual agent, per time step, compared to previous velocity-based methods.

5.2 Prior Work

The prevalent approach to navigation in video games, mobile robotics, and simulated environments has been to use roadmaps (Canny, 1988). Increasingly, navigation meshes (Kallmann, 2010; Snook, 2000; Van Toll, Cook, and Geraerts, 2011) and similar methods (Geraerts, Kamphuis, Karamouzas, *et al.*, 2008; Pettré, Laumond, and Thalmann, 2005) are superseding that approach. Randomized methods (Kavraki, Švestka, Latombe, *et al.*, 1996; LaValle and Kuffner, 2001) may be used for roadmap generation, and the Hertel-Mehlhorn algorithm (Hertel and Mehlhorn, 1985) and space-filling volumes (Tozour, 2003) allow for automatic navigation mesh generation.

In static environments, derivatives of the A* algorithm (Hart, Nilsson, and Raphael, 1968) are usually used to search the roadmap or navigation mesh for a path to the goal. In dynamic environments, the D* algorithm (Stentz, 1995) may be used to repair a previously planned path instead of re-planning from scratch. Planners based on roadmaps have also been adapted to accommodate dynamic environments by reusing information that was previously computed (Ferguson, Kalra, and Stentz, 2006; Jaillet and Simeon, 2004; Kallmann and Mataric, 2004; Zucker, Kuffner, and Branicky, 2007) or by integrating dynamic obstacle movement directly into the planner (Hsu, Kindel, Latombe, *et al.*, 2002).

Historically, video games have used force-based methods (Reynolds, 1987), in combination with roadmap and navigation mesh approaches, to provide local collision avoidance for groups of virtual agents moving through the environment. Many other methods from mobile robotics (Fox, Burgard, and Thrun, 1997; Petti and Fraichard, 2005) and rule-based or social-force models from crowd simulation (Guy, Curtis, Lin, *et al.*, 2012; Helbing and Molnár, 1995; Karamouzas and Overmars,

2010; Kluge and Prassler, 2006; Van Welbergen, Van Basten, Egges, *et al.*, 2010) are equally suited to the navigation of virtual agents in video games and simulated environments.

Generally, current collision avoidance approaches are limited to using some form of point goal (Van den Berg, Patil, Sewall, *et al.*, 2008) or line segment goal (Curtis, Snape, and Manocha, 2012) in connection with the global planner.

5.3 Goal Velocity Obstacles

In this section, we introduce the new concept of using truncated velocity obstacles (Van den Berg, Guy, Lin, *et al.*, 2011; Guy, Chhugani, Kim, *et al.*, 2009; Tychonievich, Zaret, Mantegna, *et al.*, 1989) to specify the goal regions of virtual agents in the two-dimensional velocity space from which to choose collision-free velocities toward their goal regions in the two-dimensional workspace.

5.3.1 Overall Approach

We adopt the problem description and notation of Section 2.3 with one important change; instead of a point goal $\boldsymbol{p}_A^{\text{goal}} \in \mathbb{R}^2$, let each virtual agent A instead have a bounded goal region $G \subseteq \mathbb{R}^2$, which is not necessarily known to the other virtual agents. The goal region may be of any shape, need not simply be a point, and may have a nonzero linear velocity $\boldsymbol{v}_G \in \mathbb{V}^2$. For simplicity, we assume that the goal region does not rotate, *i.e.*, its angular velocity is zero. The objective of each virtual agent A is now to choose, independently and simultaneously, a new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ at each time step to compute a trajectory toward any point in its goal region G without collisions with other virtual agents, static obstacles, or dynamic obstacles. A virtual agent A has reached its goal region G if $A \cap G \neq \emptyset$.

Recall the definitions of the truncated velocity obstacle (Definition 3.1) and preferred velocity (Definition 2.4) of a virtual agent A that is navigating to a point goal $\boldsymbol{p}_G \in \mathbb{R}^2$ with a preferred speed $v_A^{\text{pref}} \in \mathbb{V}$ while avoiding a virtual agent B:

$$VO_{A|B}^{\tau} := \{\boldsymbol{v} \mid \exists s \in [0,\tau] :: s(\boldsymbol{v} - \boldsymbol{v}_B) \in B \oplus -A\},$$
$$\boldsymbol{v}_A^{\text{pref}} := v_A^{\text{pref}} \frac{\boldsymbol{p}_A - \boldsymbol{p}_G}{\|\boldsymbol{p}_A - \boldsymbol{p}_G\|_2}.$$

Instead of using truncated velocity obstacles purely for excluding velocities that may cause collisions with other virtual agents or dynamic obstacles, then optimizing with respect to a preferred velocity for navigation to a goal in the workspace $\mathbb{R}^2$, we propose the additional use of velocity obstacles to define the goal regions of a virtual agent within the velocity space $\mathbb{V}^2$.

Definition 5.1. The *goal velocity obstacle* of virtual agent A toward the goal region G is defined as

$$GVO_{A|G} := VO_{A|G}^{\tau} := \{\boldsymbol{v} \mid \exists s \in [0,\tau] :: s(\boldsymbol{v} - \boldsymbol{v}_B) \in G \oplus -A\} \subseteq \mathbb{V}^2.$$

We then choose a new velocity $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2$ of virtual agent A such that $\boldsymbol{v}_A^{\text{new}}$ lies not only outside the truncated velocity obstacles induced by other virtual agents, but also inside the goal velocity obstacle

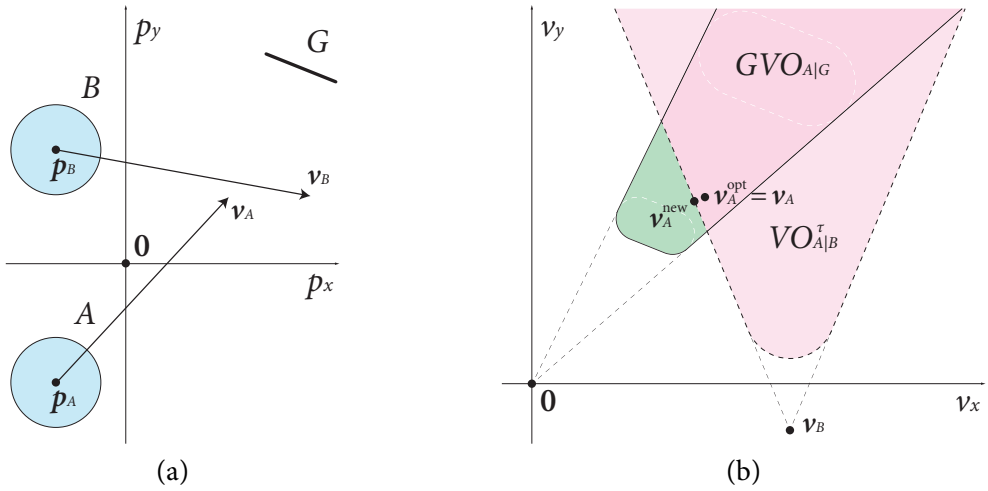


Figure 5.1: (a) Virtual agent A navigating toward a static line segment goal region G in the two-dimensional workspace while avoiding virtual agent B. (b) The goal velocity obstacle $GVO_{A|G}$ of virtual agent A toward the goal region G (shaded green) in the two-dimensional velocity space, and the truncated velocity obstacle $VO^{\tau}_{A|B}$ of virtual agent A induced by virtual agent B (shaded pink).

toward the goal region G, *i.e.*, $\boldsymbol{v}_A^{\text{new}} \in GVO_{A|G} \setminus VO^{\tau}_A$. This is illustrated in Figure 5.1, and the overall approach is summarized by Algorithm 5.1.

5.3.2 Choice of Velocities

In general, there will be a choice of collision-free velocities $\boldsymbol{v}_A^{\text{new}}$ that will navigate the virtual agent A to some point in its goal region. Assuming that there is no preference as to which point in the goal region a virtual agent A ultimately reaches, we choose a velocity $\boldsymbol{v}_A^{\text{opt}} \in \mathbb{V}^2$, which we call the "optimization velocity," with respect to which we must optimize from those velocities that are collision-free and inside the goal velocity obstacle, *i.e.*,

$$\boldsymbol{v}_A^{\text{new}} = \underset{\boldsymbol{v} \in GVO_{A|G} \setminus VO^{\tau}_A}{\arg\min} \left\| \boldsymbol{v} - \boldsymbol{v}_A^{\text{opt}} \right\|_2 .$$

Motivated by a desire for virtual agents to make as minimal change in velocity as possible at each time step, *c.f.*, Guy, Curtis, Lin, *et al.* (2012), we choose the optimization velocity $\boldsymbol{v}_A^{\text{opt}}$ of a virtual agent A as follows.

1. If the current velocity $\boldsymbol{v}_A$ is inside the goal velocity obstacle $GVO_{A|G}$, we choose the current velocity as the optimization velocity, whether or not that velocity is collision-free, *i.e.*, $\boldsymbol{v}_A^{\text{opt}} := \boldsymbol{v}_A$.

2. If the current velocity is outside the goal velocity obstacle, so the virtual agent is moving away from its goal region, we choose the closest velocity to the current velocity $\boldsymbol{v}_A$, with respect to Euclidean distance in the velocity space $\mathbb{V}^2$, that lies inside the goal velocity obstacle, *i.e.*,

$$\boldsymbol{v}_A^{\text{opt}} := \underset{\boldsymbol{v} \in GVO_{A|G}}{\arg\min} \left\| \boldsymbol{v} - \boldsymbol{v}_A \right\|_2 .$$

Algorithm 5.1: Our method for navigating multiple virtual agents in the two-dimensional workspace using goal velocity obstacles.

inputs
  List of virtual agents $\mathcal{A} \neq \emptyset$ each with list of goal regions $\mathcal{G} \neq \emptyset$
  List of static and dynamic obstacles $\mathcal{O}$
loop
  for all $A \in \mathcal{A}$ **do**
    for all $B \in \mathcal{A}$ **such that** $A \neq B$ **do**
      Construct $VO_{A|B}$
      Construct $RVO_{A|B}$
      Construct $HRVO_{A|B}$ from $VO_{A|B}$ and $RVO_{A|B}$
    end for
    for all $O \in \mathcal{O}$ **do**
      Construct $VO_{A|O}$
    end for
    Construct $HRVO_A$ from all $HRVO_{A|B}$ and all $VO_{A|O}$
    for all $G \in \mathcal{G}$ **do**
      Construct $GVO_{A|G} := VO^{\tau}_{A|G}$
    end for
    Construct GVO_A from all $GVO_{A|G}$
    Compute optimization velocity $\boldsymbol{v}_A^{\text{opt}}$ from current velocity $\boldsymbol{v}_A$ and GVO_A
    Compute $GVO_A \setminus HRVO_A$
    Compute new velocity $\boldsymbol{v}_A^{\text{new}} \in GVO_A \setminus HRVO_A$ closest to $\boldsymbol{v}_A^{\text{opt}}$
  end for
end loop

The optimization velocity $\boldsymbol{v}_A^{\text{opt}}$ is distinct from the notion of preferred velocity $\boldsymbol{v}_A^{\text{pref}}$, and, in general, much less influences the path taken by the virtual agent A.

5.3.3 High Densities of Virtual Agents

By definition, if a virtual agent chooses a velocity inside the goal velocity obstacle at every time step, it will reach its goal region at some future moment in time, assuming that such a velocity exists. If, however, due to a high density of virtual agents, there is no such velocity, *i.e.*,

$$GVO_{A|G} \subseteq VO^{\tau}_A := \bigcup_{\substack{B \in \mathcal{A} \\ A \neq B}} VO^{\tau}_{A|B},$$

this means that either the goal region G is moving away from the virtual agent A at a faster speed than the virtual agent can attain, or else the path to the goal region is blocked by other virtual agents or dynamic obstacles. In the first case, it is not possible for the virtual agent to reach its goal region. However, in the second case, we choose a new velocity by relaxing some of the constraints in velocity space. In relaxing constraints, we must balance the possibly conflicting objectives of avoiding collisions and reaching the goal region:

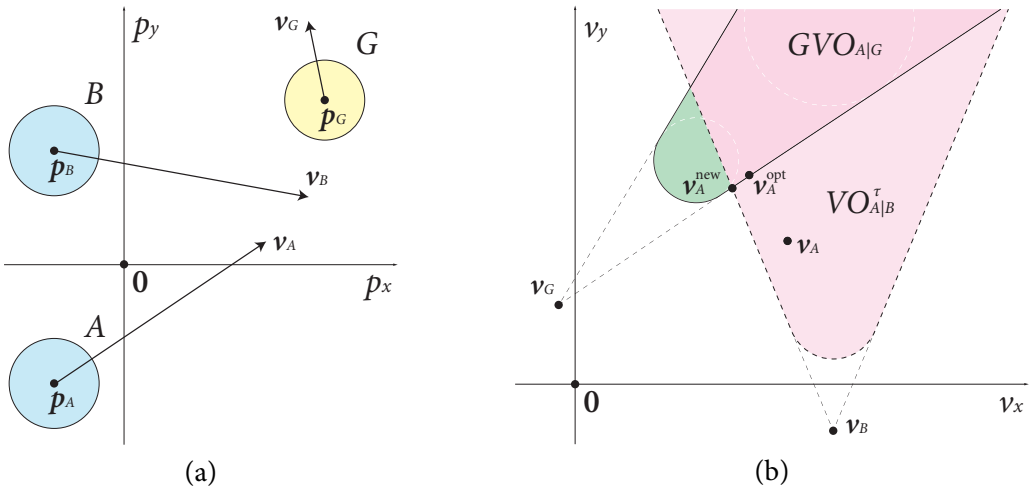


Figure 5.2: (a) Virtual agent A navigating toward a moving disc-shaped goal region G with velocity $\boldsymbol{v}_G$ (yellow disc) in the two-dimensional workspace while avoiding virtual agent B. (b) The goal velocity obstacle $GVO_{A|G}$ of virtual agent A toward the goal region G (shaded green) in the two-dimensional velocity space, and the truncated velocity obstacle $VO^{\tau}_{A|B}$ of virtual agent A induced by virtual agent B (shaded pink).

1. If the priority is to avoid collisions, above all else, we can replace the goal velocity obstacle $GVO_{A|G}$ with the whole of the velocity space $\mathbb{V}^2$ at each time step until a velocity within the goal velocity obstacle becomes available, *i.e.*, $\boldsymbol{v}_A^{\text{new}} \in \mathbb{V}^2 \setminus VO^{\tau}_A \neq \varnothing$.

2. If an increased possibility of collision is allowable, we remove, in turn, the truncated velocity obstacle induced by the most distant virtual agent, with respect to Euclidean distance in the workspace $\mathbb{R}^2$, until a velocity within the goal velocity obstacle becomes free, *i.e.*,

$$\boldsymbol{v}_A^{\text{new}} \in GVO_{A|G} \setminus (VO^{\tau}_{A_1} \cup \cdots \cup VO^{\tau}_{A_m}) \neq \varnothing,$$

where $\left\|\boldsymbol{p}_A - \boldsymbol{p}_{A_i}\right\|_2 \leq \left\|\boldsymbol{p}_A - \boldsymbol{p}_{A_{i+1}}\right\|_2$, and $GVO_{A|G} \subseteq VO^{\tau}_{A_1} \cup \cdots \cup VO^{\tau}_{A_{m+1}}$ for $m < n$.

In our experiments, we favored the second of these options.

5.3.4 Moving Goal Regions

Consider now a goal region moving in a near-perpendicular direction to a virtual agent (see Figure 5.2). When navigating using preferred velocities, the future trajectory $\boldsymbol{p}_G + s\boldsymbol{v}_G$ of the goal region G is not taken into account, only the instantaneous position of its center $\boldsymbol{p}_G$. Therefore, the virtual agent is always changing its preferred velocity, and, hence, current velocity, at each time step, navigating toward the position of the goal region at the previous time step. This will occur even though it is apparent that the goal region will cross the path of the virtual agent if the virtual agent continues on its original trajectory. Contrast this with the goal velocity obstacle, which, by definition, considers the future trajectory of the goal region, coupled with an optimization velocity that favors minimal changes in current velocity.

5.3.5 Multiple Goal Regions

When navigating using preferred velocities, if a goal region consists of the union of all elements of a set $\mathcal{G}$ of multiple goal sub-regions (see Figure 5.3), virtual agent A would be required to choose in advance one goal sub-region $G \in \mathcal{G}$ toward which to navigate explicitly and set $\boldsymbol{v}_A^{\text{pref}}$ in the direction of $\boldsymbol{p}_G$. However, with the formulation of goal velocity obstacles, we can construct goal velocity obstacles of each individual goal sub-region and then define the goal velocity obstacle of the entire goal region as the union of these individual goal velocity obstacles, *i.e.*,

$$GVO_{A|\mathcal{G}} := \bigcup_{G \in \mathcal{G}} GVO_{A|G}.$$

This has the advantage that if the path to one of the goal sub-regions G is blocked by other virtual agents, then the navigating virtual agent will automatically divert to another goal sub-region since the goal velocity obstacle $GVO_{A|G}$ corresponding to the blocked goal sub-region will be completely covered with truncated velocity obstacles, *i.e.*, $GVO_{A|G} \subseteq VO_A^{\tau}$.

We choose the optimization velocity to be such that it lies inside the goal velocity obstacle of the goal sub-region toward which the virtual agent moved at the previous time step, which reduces the possibility that the velocity of the virtual agent will oscillate between goal velocity obstacles of different goal sub-regions. While the optimization velocity is chosen relative to a particular goal velocity obstacle, this does not preclude a velocity being chosen if that goal sub-region is later found to be blocked, and no explicit changes to the formulation are required to accommodate this situation.

5.3.6 Goal Regions with Time Windows

Suppose now that a goal region is only available to a virtual agent during some time window $[\tau_1, \tau_2]$ relative to the current time. We can express this time dependency by truncating the goal velocity obstacle both at the apex and toward the base (see Figure 5.4). For this doubly truncated goal velocity obstacle, the values of τ_1 and τ_2 decrease at each successive time step, and the definition of the goal velocity obstacle $GVO_{A|G}^{\tau_1,\tau_2}$ of virtual agent A toward the goal region G with time window $[\tau_1, \tau_2]$ becomes

$$GVO_{A|G}^{\tau_1,\tau_2} = \left\{\boldsymbol{v} \;\middle|\; \exists s \in [\tau_1, \tau_2] :: s(\boldsymbol{v} - \boldsymbol{v}_B) \in G \oplus -A\right\}.$$

5.3.7 Point Goals and Point Virtual Agents

While, by construction, the goal velocity obstacle most benefits scenarios that do not treat the goal regions and virtual agents simply as points, we can accommodate their occurrence as follows. The goal velocity obstacle of point virtual agent $\boldsymbol{p}_A$ toward point goal $\boldsymbol{p}_G$ will consist of a single ray,

$$GVO_{A|G} = \left\{\boldsymbol{v}_G + s(\boldsymbol{p}_G - \boldsymbol{p}_A) \;\middle|\; s > 0\right\},$$

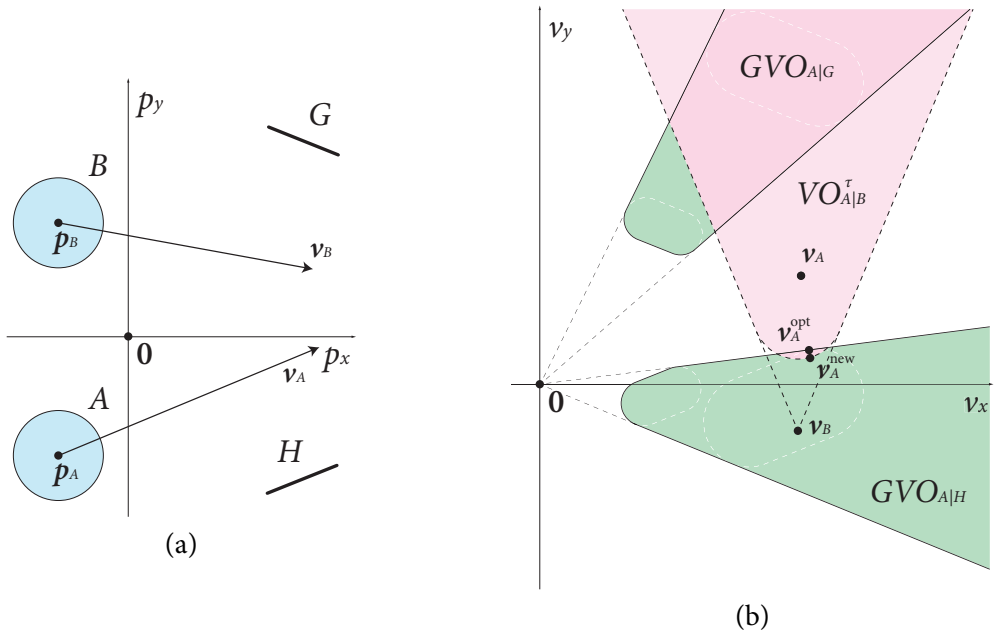


Figure 5.3: (a) Virtual agent A navigating toward a choice of two static line segment goal sub-regions G and H in the two-dimensional workspace while avoiding virtual agent B. (b) The goal velocity obstacles $GVO_{A|G}$ of virtual agent A toward goal sub-region G and $GVO_{A|H}$ of virtual agent A toward goal sub-region H (shaded green) in the two-dimensional velocity space, and the truncated velocity obstacle $VO^{\tau}_{A|B}$ of virtual agent A induced by virtual agent B (shaded pink).

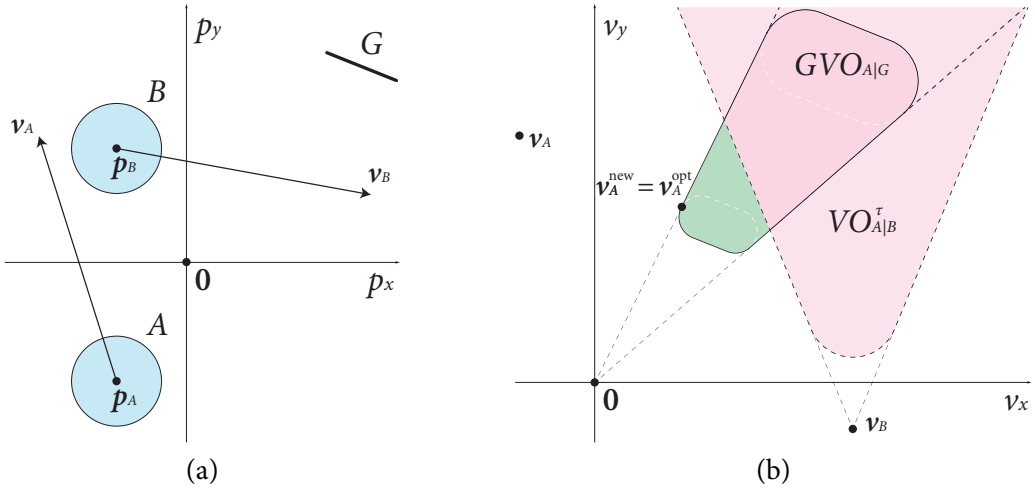


Figure 5.4: (a) Virtual agent A navigating toward a static line segment goal region G with limited time window $[\tau_1, \tau_2]$ in the two-dimensional workspace while avoiding virtual agent B. (b) The goal velocity obstacle $GVO_{A|G}$ of virtual agent A toward the goal region G with limited time window $[\tau_1, \tau_2]$ (shaded green) in the two-dimensional velocity space, and the truncated velocity obstacle $VO^{\tau}_{A|B}$ of virtual agent A induced by virtual agent B (shaded pink).

so the optimization velocity will lie on that ray. If the point goal is static, *i.e.*, $\boldsymbol{v}_G = 0$, then the optimization velocity will be toward the point goal, although possibly with a different magnitude to that of a preferred velocity, depending on the current velocity of the virtual agent. If the new velocity cannot lie on the ray, due to other virtual agents or dynamic obstacles blocking the path to the point goal, then we must replace the ray that is the goal velocity obstacle with the whole of the velocity space $\mathbb{V}^2$ until a velocity on the ray becomes available in the future. While the formulation is effectively reduced to that of previous methods, *i.e.*, using a single, preferred velocity, we do not need to handle it as a special case.

5.4 Experimentation and Performance

In this section, we describe the implementation of our approach and discuss the results of our experiments involving multiple virtual agents.

5.4.1 Implementation

We implemented our approach in C++ using hybrid reciprocal velocity obstacles (Chapter 2) for collision avoidance between pairs of virtual agents. Our algorithm to choose the new velocity of each virtual agent at every time step was based on the ClearPath efficient geometric algorithm (Guy, Chhugani, Kim, *et al.*, 2009). Calculations for each virtual agent were carried out in separate and independent threads, and in parallel, where possible, using Intel Threading Building Blocks, version 4.1. The code was compiled using the Intel C++ Compiler XE, version 13.

For efficiency reasons, only a subset of all other virtual agents within a fixed radius of each virtual agent, with respect to Euclidean distance in the workspace $\mathbb{R}^2$, were considered for collision avoidance, and these virtual agents were selected at the beginning of every time step using an algorithm based on k-D trees (De Berg, Cheong, Van Kreveld, *et al.*, 2008).

5.4.2 Experiments

We applied our approach to experiments containing twenty-five to two hundred virtual agents as follows.

5.1. The virtual agents are positioned evenly along one side of a rectangular environment in two dimensions. The virtual agents must navigate to one static line segment goal region located midway along the opposite side of the environment to the starting positions of the virtual agents.

5.2. The virtual agents must navigate across the environment toward one moving line segment goal region. The goal region moves at a constant velocity along the opposite side of the environment to the starting positions of the virtual agents, perpendicular to the direct paths of the virtual agents to the goal region.

5.3. The virtual agents must navigate across the environment toward two static line segment goal regions located at each end of the opposite side of the environment to the starting positions of the virtual agents.

Each experiment was performed twice; first using goal velocity obstacles and hybrid reciprocal velocity obstacles, and then using preferred velocities and hybrid reciprocal velocity obstacles. Each virtual agent had a radius of 1 m, an initial or preferred speed of 1.4 m/s, as appropriate, and a maximum speed of 2.5 m/s.

Tables 5.1 to 5.3 list the total number of collisions between virtual agents during the entirety of each experiment, the average path length from the starting position to the goal region of each virtual agent in each experiment, and the average computation time at each time step. All timings are for one core of a quad-core 2.8 GHz Intel Core i5 processor within a standard desktop computer containing 8 GB memory and running OS X Mountain Lion, version 10.8.2. Videos of these experiments are available online at `http://gamma.cs.unc.edu/GVO/`.

5.4.3 Discussion

Figures 5.5 to 5.7 show that, for twenty-five virtual agents, almost the entire goal region is utilized by the virtual agents in each experiment using goal velocity obstacles. In Figure 5.6, the virtual agents take into account the motion of the goal region, and, in particular, virtual agents farthest from the starting position of the goal region maintain a direct path to intercept the goal region as it passes close by, later in the experiment. Figure 5.7 demonstrates the virtual agents splitting into two groups to move toward the closest goal region to their starting position.

From Table 5.1, it is clear that in all experiments, for twenty-five to two hundred virtual agents, the number of collisions between virtual agents is significantly less when using goal velocity obstacles, rather than preferred velocities. Specifically, there are at least fifty-five percent fewer collisions in the experiments with one static goal region, at least ninety-seven percent fewer collisions in the experiments with one moving goal region, and at least ninety percent fewer collisions in the experiments with two static goal regions.

The length of the path that each virtual agent takes to a goal region, shown in Table 5.2, is also less when using goal velocity obstacles. In the experiments with one or two static goal regions, the paths are at least five percent shorter, and in the experiments with one moving goal region, the paths are always at least ten percent shorter, and more than twenty-five percent shorter in most cases.

Computationally, Table 5.3 reports that it takes between two percent and twenty-one percent longer, at each time step, to compute new velocities using goal velocity obstacles, rather than preferred velocities. Mostly, the difference is less than ten percent, however, and, overall, using goal velocity obstacles adds only a few microseconds of computation, per agent, at each time step.

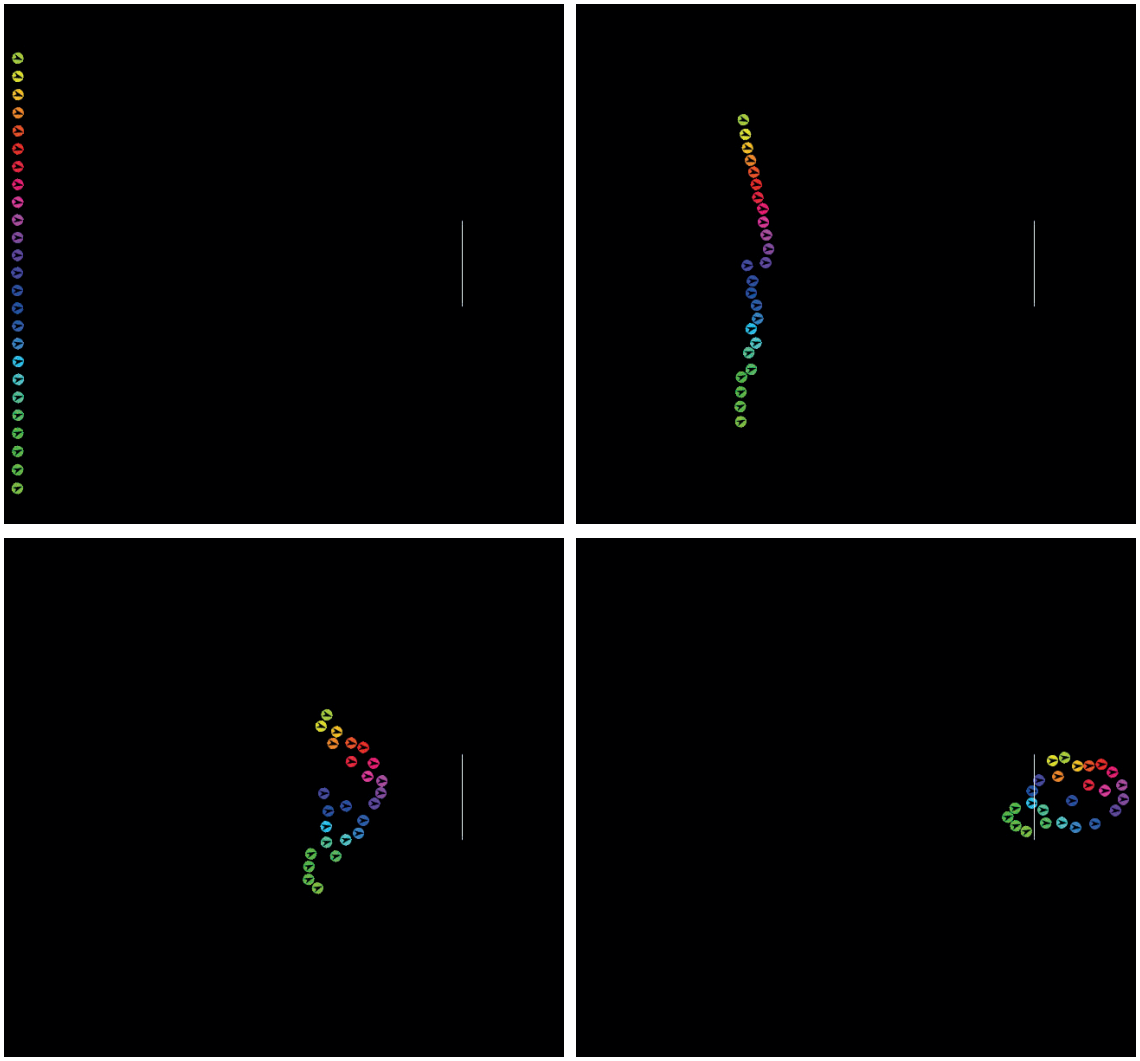


Figure 5.5: Screenshots of twenty-five virtual agents (colored discs) navigating simultaneously toward one static line segment goal region (white line) in the two-dimensional workspace using goal velocity obstacles (Experiment 5.1).

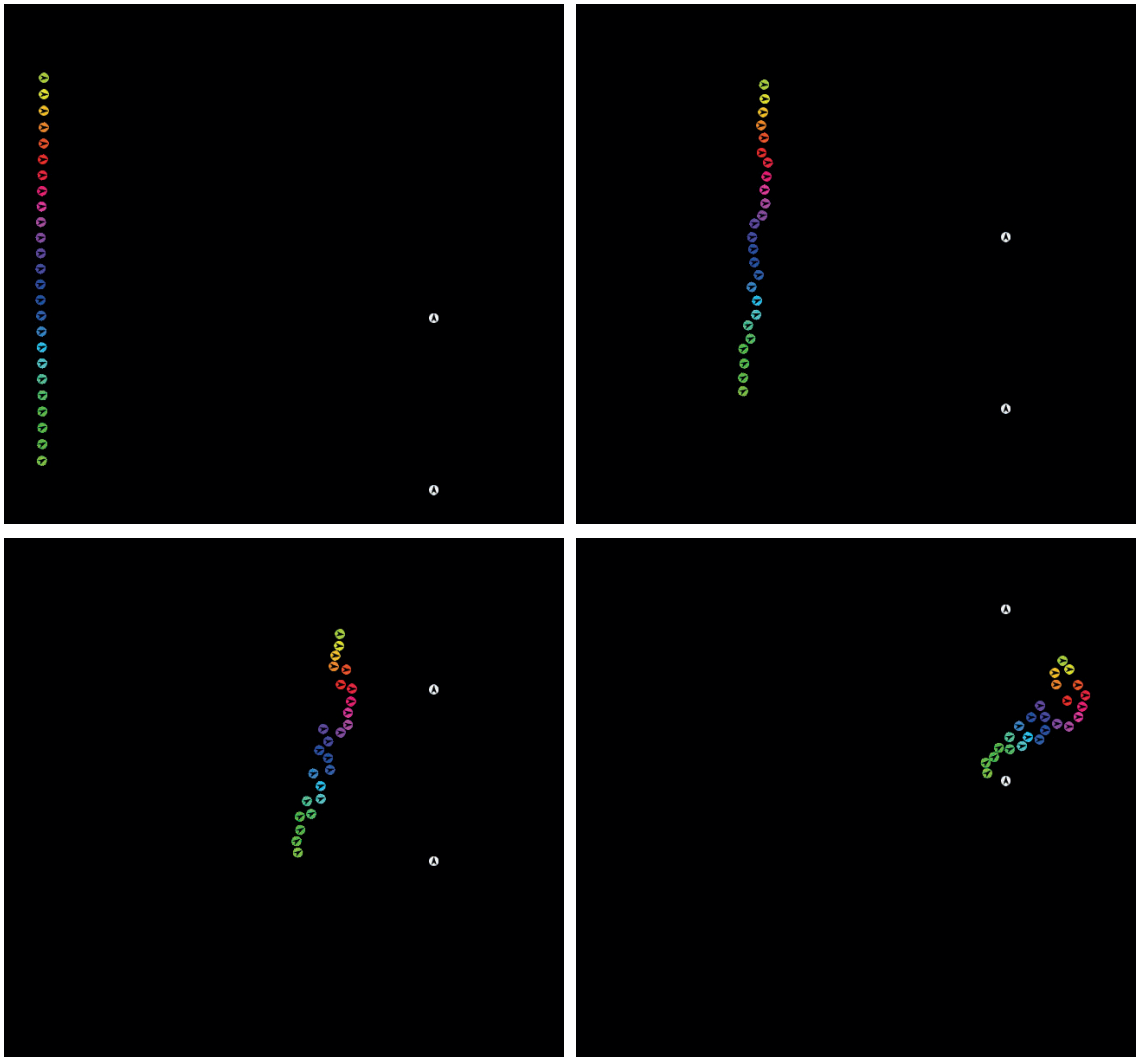


Figure 5.6: Screenshots of twenty-five virtual agents (colored discs) navigating simultaneously toward one moving line segment goal region (between the two white discs) in the two-dimensional workspace using goal velocity obstacles (Experiment 5.2).

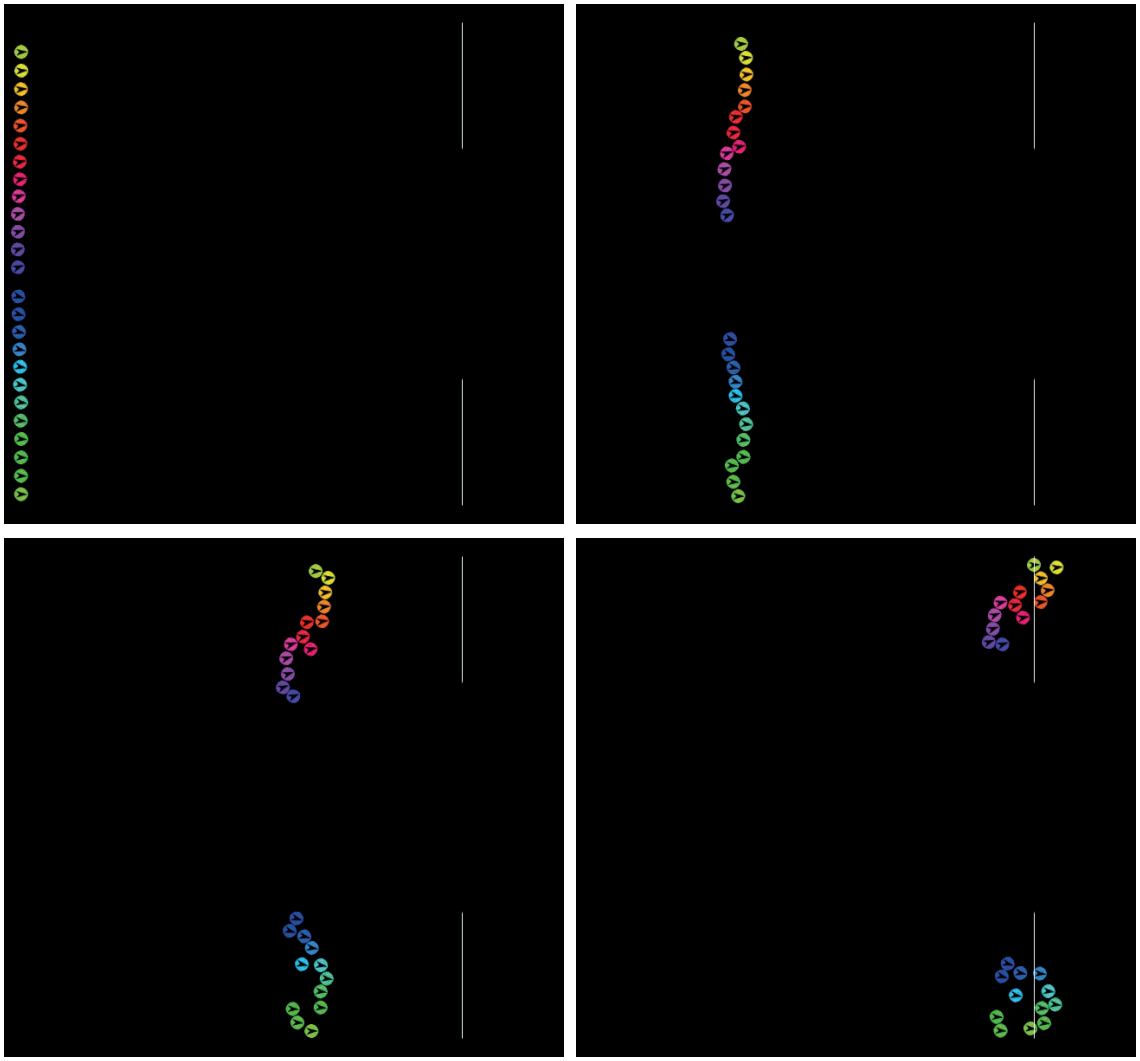


Figure 5.7: Screenshots of twenty-five virtual agents (colored discs) navigating simultaneously toward two static line segment goal regions (white lines) in the two-dimensional workspace using goal velocity obstacles (Experiment 5.3).

Table 5.1: Total number of collisions between virtual agents during the entirety of each experiment using hybrid reciprocal velocity obstacles with preferred velocities (PV) and hybrid reciprocal velocity obstacles with goal velocity obstacles (GVO).

	Number of virtual agents	Number of collisions per experiment	
		PV	GVO
Experiment 5.1	25	56	0
	50	62	10
	100	80	24
	200	106	46
Experiment 5.2	25	132	4
	50	144	2
	100	166	2
	200	218	0
Experiment 5.3	25	348	4
	50	498	10
	100	778	28
	200	996	94

Table 5.2: Average path length from the starting position to the goal region of each virtual agent, in meters, using hybrid reciprocal velocity obstacles with preferred velocities (PV) and hybrid reciprocal velocity obstacles with goal velocity obstacles (GVO).

	Number of virtual agents	Path length per virtual agent (m)	
		PV	GVO
Experiment 5.1	25	243	229
	50	266	246
	100	340	308
	200	441	404
Experiment 5.2	25	323	239
	50	341	241
	100	407	301
	200	455	405
Experiment 5.3	25	217	206
	50	241	227
	100	315	296
	200	428	407

Table 5.3: Average computation time at each time step on one core of a 2.8 GHz Intel Core i5 processor, in milliseconds, for virtual agents using hybrid reciprocal velocity obstacles with preferred velocities (PV) and hybrid reciprocal velocity obstacles with goal velocity obstacles (GVO).

	Number of virtual agents	Computation time per time step (ms)	
		PV	GVO
Experiment 5.1	25	0.5	0.6
	50	1.4	1.5
	100	3.1	3.3
	200	6.4	6.6
Experiment 5.2	25	2.4	2.9
	50	4.8	4.9
	100	6.9	7.3
	200	9.1	9.7
Experiment 5.3	25	0.3	0.4
	50	1.0	1.1
	100	2.0	2.3
	200	4.0	4.1

Chapter 6

CONCLUSION

6.1 Thesis Statement and Conclusions

We conclude by recalling our thesis statement from Section 1.6.

Multiple mobile robots or virtual agents with kinematic and dynamic constraints may navigate through shared environments in two-dimensional or three-dimensional workspaces containing static and dynamic obstacles without collisions or oscillations and without central coordination using formulations based on velocity obstacles.

We have presented the following contributions in support of the thesis statement.

Independent Navigation of Multiple Mobile Robots or Virtual Agents with Hybrid Reciprocal Velocity Obstacles. We have introduced the "hybrid reciprocal velocity obstacle" for navigating multiple mobile robots or virtual agents sharing an environment in the two-dimensional workspace.

We take into account obstacles in the environment and uncertainty in radius, position, and velocity. We also consider the dynamics and kinematics of the robots, allowing us to apply our approach to iRobot Create mobile robots. Our formulation explicitly considers reciprocity, such that each robot can assume that other robots are cooperating to avoid collisions, but each of the robots acts completely independently without central coordination, and does not communicate with other robots. We show direct and collision-free navigation that is free of oscillations in velocity.

We have shown the effectiveness of approach in experiments with up to five mobile robots or one thousand virtual agents sharing an environment.

Smooth Navigation of Multiple Robots under Differential-Drive Constraints. We have described a method for obtaining control inputs, the two wheel speeds, from a given velocity of a robot with differential-drive constraints using the notion of effective center and effective radius to overcome the inherent limitation that the center of a differential-drive robot may not be moved instantaneously in a direction orthogonal to its wheels. We have combined this formulation with optimal reciprocal collision avoidance to derive our algorithm and proved that it theoretically guarantees smooth and locally collision-free motion for multiple differential-drive robots navigating in a shared environment in the two-dimensional workspace. Each differential-drive robot is independent and is able to react

to the other robots without explicit communication by simply observing their current positions and velocities.

While other approaches (Kluge, Bank, Prassler, *et al.*, 2004) exhibit empirically smooth trajectories in limited examples, they provide no mathematical guarantees that the trajectories will be smooth in other circumstances. Furthermore, our algorithm is not constrained to a finite set of behaviors (Pallottino, Scordio, Bicchi, *et al.*, 2007), potentially allowing any maneuver permitted by the kinematics of each robot. Unlike other approaches that require explicit communication between every robot (Bekris, Tsianos, and Kavraki, 2007), robots using our algorithm can be fully independent, making all decisions based only on their own observations.

We have implemented and applied our method to iRobot Create robots and shown its effectiveness in experiments with up to four robots sharing an environment, as well as in simulations of up to one thousand differential-drive robots.

Navigating Multiple Simple-Airplanes in Three-Dimensional Workspace. We have introduced the "simple-airplane" and presented an extension of the optimal reciprocal collision avoidance algorithm in three dimensions to allow multiple simple-airplanes to navigate among each other. In practice, our approach is able to generate collision-free and oscillation-free paths that satisfy the underlying kinematic and dynamic constraints. We consider most kinematic and dynamic constraints of the simple-airplane when choosing a velocity from those permitted by the optimal reciprocal collision avoidance algorithm, and then enumerate a set of precomputed curves to confirm that the new velocity meets all remaining kinematic constraints.

We use the notion of reciprocity to prevent undesirable oscillations, and, by incorporating kinematic and dynamic constraints, introduce the idea of "variable reciprocity" to ensure that simple-airplanes that are less constrained take more responsibility for avoiding collisions. Moreover, the simple-airplanes are restricted to neither constant speed nor fixed altitude, as is the case in many approaches.

We have implemented our method and performed experiments that simulate up to sixteen simple-airplanes. We compute trajectories that are observed to be both free of collisions and free of oscillations, and which satisfy the kinematic and dynamic constraints of each simple-airplane.

Goal Velocity Obstacles for Spatial Navigation of Multiple Virtual Agents. We have presented the "goal velocity obstacle" for the spatial navigation of multiple virtual agents to arbitrary-shaped, planar goal regions in the two-dimensional plane. Our approach uses truncated velocity obstacles not only to compute velocities that may cause collisions with other virtual agents, but also to define the goal velocity obstacle, which specifies velocities in the two-dimensional velocity space that will direct a virtual agent toward its goal region in the workspace.

Our goal velocity obstacle formulation is general, allowing for planar goal regions of any shape without the need to approximate the goal region as a point or line segment as is required by most previous collision avoidance methods. Goal regions may be static, or they may be moving with a

nonzero linear velocity. We may specify multiple goal regions of each virtual agent without requiring an explicit goal allocation algorithm to choose a particular goal region of each virtual agent in advance of each time step. Our approach also allows for goal regions that are available for a limited time window.

We have applied our approach to multiple challenging experiments by integrating with the hybrid reciprocal velocity obstacle formulation for collision avoidance. On average, the virtual agents traverse shorter path lengths and have fewer collisions than when simply using preferred velocities directed to a single point in their goal region instead of goal velocity obstacles.

6.2 Limitations and Future Work

Our work has some limitations that could be addressed by future work.

Independent Navigation of Multiple Mobile Robots or Virtual Agents with Hybrid Reciprocal Velocity Obstacles and Smooth Navigation of Multiple Robots under Differential-Drive Constraints. In the future, we would like to develop a more sophisticated and less conservative model of uncertainty that takes into account more than simply uncertainty in position and velocity originating from the sensors of the robot and apply it to both the hybrid reciprocal velocity formulation and optimal reciprocal collision avoidance with effective center and effective radius.

In both of our implementations, the robots currently receive their sensor readings from an overhead digital video camera and computation is performed on a single notebook computer. As a next step, we would like to equip each robot with purely localized sensing and distributed computing, using odometry, orientation sensors, and relative positions to estimate global positions (Roumeliotis and Rekleitis, 2004). Our approaches can be applied without adaptation if data is gathered locally, and the hybrid reciprocal velocity obstacles and optimal reciprocal collision avoidance half-planes are defined just as well using only the relative positions and velocities of the mobile robots.

At present, we assume in general that a velocity outside all hybrid reciprocal velocity obstacles or inside an optimal reciprocal collision avoidance half-plane exists. We would like to relax this assumption in the future to accommodate very dense situations without observing any collisions or deadlocks when the space is entirely covered with hybrid reciprocal velocity obstacles or optimal reciprocal collision avoidance half-planes.

Our method for incorporating static obstacles into the hybrid reciprocal velocity obstacle formulation does not allow for navigation through some narrow passages, and the enlarged effective radius of each differential-drive robot makes maneuvering through dense situations and narrow passages difficult when using optimal reciprocal collision avoidance. Also, our formulations consider kinematic constraints, but, at most, very simple dynamic constraints. We have addressed this, in part, with the acceleration-velocity obstacle formulation (Van den Berg, Snape, Guy, *et al.*, 2011; Snape, Guy, Van den Berg, *et al.*, 2013).

Navigating Multiple Simple-Airplanes in Three-Dimensional Workspace. Our current implementation ignores both the roll and the pitch of airplanes, but in principle, we can easily extend it to add extra kinematic constraints. We also ignore some external factors, such as wind and drag, that may influence the paths chosen by a simple-airplane. Furthermore, we currently assume that each simple-airplane has perfect sensing. We would like to address all of these limitations in the future.

While our formulation is capable of handling dynamic obstacles, we do not consider static obstacles and the resulting complex environments, containing, for instance, buildings or higher terrain. We would also need to add a global planner to our approach to direct the simple-airplanes to their goals.

Given that each simple-airplane essentially plans its path independently by only observing other simple-airplanes and dynamic obstacles, there is a significant opportunity to exploit the parallel nature of this approach further. Another possibility would be a decentralized approach for the navigation of physical robots, such as quad-rotor helicopters.

Goal Velocity Obstacles for Spatial Navigation of Multiple Virtual Agents. We have focused on virtual agents, such as in video games or simulated environments, but, in principle, the goal velocity obstacle could be adapted for the navigation of multiple mobile robots.

While we have only considered planar goal regions in the two-dimensional workspace and goal velocity obstacles in the two-dimensional velocity space, in the future, we could extend our formulation to three dimensions for the navigation of virtual or real-world flying agents to goal regions in the three-dimensional workspace and goal velocity obstacles in the three-dimensional velocity space.

BIBLIOGRAPHY

Y. Abe and Y. Matsuo, "Collision avoidance method for multiple autonomous mobile agents by implicit cooperation", in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Maui, Hawaii, Oct. 29–Nov. 3, 2001), vol. 3, New York, N.Y.: IEEE Press, 2001, pp. 1207–1212, DOI: 10/fvpkh5 (cit. on pp. 8, 11).

N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning", in *Proc. IEEE Int. Conf. Robotics and Automation* (Minneapolis, Minn., Apr. 22–28, 1996), vol. 1, New York, N.Y.: IEEE Press, 1996, pp. 113–120, DOI: 10/d5cdn4 (cit. on p. 2).

E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk, "Maximum thick paths in static and dynamic environments", in *Comput. Geom.* 43.3 (Apr. 2010): Special issue on 24th Annu. Symp. Computational Geometry, pp. 279–294, DOI: 10/b27xst (cit. on p. 49).

D. J. Balkcom and M. T. Mason, "Time optimal trajectories for bounded velocity differential drive vehicles", in *Int. J. Robot. Res.* 21.3 (Mar. 2002): Special issue on algorithmic foundations of robotics, pp. 199–217, DOI: 10/fn43mc (cit. on p. 32).

K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online", in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (San Diego, Calif., Oct. 29–Nov. 2, 2007), New York, N.Y.: IEEE Press, 2007, pp. 3784–3790, DOI: 10/cpkhdn (cit. on pp. 8, 76).

J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal *n*-body collision avoidance", in *Robotics Research, Proc. 14th Int. Symp. Robotics Research* (Lucerne, Switzerland, Aug. 31–Sep. 3, 2009), C. Pradalier, R. Siegwart, and G. Hirzinger, eds., Springer Tracts Advanced Robotics 70, Heidelberg, Germany: Springer, 2011, pp. 3–19, DOI: 10/bbzghd (cit. on pp. 5, 6, 8, 31, 32, 34, 35, 38, 42, 48, 62).

J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation", in *Proc. IEEE Int. Conf. Robotics and Automation* (Pasadena, Calif., May 19–23, 2008), New York, N.Y.: IEEE Press, 2008, pp. 1928–1935, DOI: 10/b8nt78 (cit. on pp. 1, 4–8, 11, 12, 31, 32, 34, 60).

J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environments", in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (Redwood City, Calif., Feb. 15–17, 2008), E. Haines and M. McGuire, eds., New York, N.Y.: ACM Press, 2008, pp. 139–147, DOI: 10/dp9xx8 (cit. on p. 62).

J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles", in *Proc. IEEE Int. Conf. Robotics and Automation* (Shanghai, China, May 9–13, 2011), New York, N.Y.: IEEE Press, 2011, pp. 3475–3482, DOI: 10/bjpqxq (cit. on p. 77).

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed., Heidelberg, Germany: Springer, 2008, ISBN: 978-3-540-77973-5 (cit. on pp. 24, 38, 43, 57, 68).

J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal, "Extending obstacle avoidance methods through multiple parameter-space transformations", in *Auton. Robot.* 24.1 (Jan. 2008), pp. 29–48, DOI: 10/cc28t8 (cit. on p. 8).

J.-D. Boissonnat, A. Cérézo, and J. Leblond, "Shortest paths of bounded curvature in the plane", in *J. Intell. Robot. Syst.* 11.1–2 (Mar. 1994): Special issue on mathematical methods of robotics, pp. 5–20, DOI: 10/b4zsgh (cit. on p. 49).

R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation", in *IEEE Trans. Syst., Man, Cybern.* 15.2 (Mar.–Apr. 1985), pp. 224–233, DOI: 10/jqw (cit. on p. 2).

J. Bruce and M. Veloso, "Real-time multi-robot motion planning with safe dynamics", in *Multi-Robot Systems: from Swarms to Intelligent Automata, Proc. Int. Workshop Multi-Robot Systems* (Washington, D.C., Mar. 21–23, 2005), L. E. Parker, F. E. Schneider, and A. C. Schultz, eds., vol. 3, Heidelberg, Germany: Springer, 2005, pp. 159–170, DOI: 10/djh2qt (cit. on p. 32).

J. F. Canny, *The Complexity of Robot Motion Planning*, ACM Doctoral Dissertation Award 1987, Cambridge, Mass.: MIT Press, 1988, ISBN: 978-0-262-03136-3 (cit. on pp. 2, 38, 60, 61).

D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. McLain, S.-M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles", in *Int. J. Syst. Sci.* 37.6 (May 2006): Special issue on cooperative control approaches for multiple autonomous vehicles, pp. 351–360, DOI: 10/b6q2mv (cit. on p. 1).

A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: a collision cone approach", in *IEEE Trans. Syst., Man, Cybern. A—Syst. Hum.* 28.5 (Sep. 1998), pp. 562–574, DOI: 10/c2sfq9 (cit. on pp. 4, 7, 8, 31).

P. Cheng, V. Kumar, R. Arkin, M. Steinberg, and K. Hedrick, "Cooperative control of multiple heterogeneous unmanned vehicles for coverage and surveillance", in *IEEE Robot. Autom. Mag.* 16.2 (Jun. 2009), p. 12, DOI: 10/b28c8r (cit. on p. 1).

Y.-J. Chiang, J. T. Klosowski, C. Lee, and J. S. B. Mitchell, "Geometric algorithms for conflict detection/resolution in air traffic management", in *Proc. IEEE Conf. Decision and Control* (San Diego, Calif., Dec. 10–12, 1997), T. E. Djaferis, ed., vol. 2, New York, N.Y.: IEEE Press, 1997, pp. 1835–1840, DOI: 10/bgr4wz (cit. on p. 49).

H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane", in *Proc. IEEE Conf. Decision and Control* (New Orleans, La., Dec. 12–14, 2007), J. C. Spall, ed., New York, N.Y.: IEEE Press, 2007, pp. 2379–2384, DOI: 10/dxt66j (cit. on pp. 4, 48, 49, 51).

S. Curtis, J. Snape, and D. Manocha, "Way portals: efficient multi-agent navigation with line-segment goals", in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games* (Costa Mesa, Calif., Mar. 9–11, 2012), M. Garland, R. Wang, S. N. Spencer, M. Gopi, and S.-E. Yoon, eds., New York, N.Y.: ACM Press, 2012, pp. 15–22, DOI: 10/htq (cit. on pp. 60, 62).

J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots", in *IEEE Trans. Robot. Autom.* 17.6 (Dec. 2001), pp. 905–908, DOI: 10/dh874d (cit. on p. 32).

L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", in *Amer. J. Math.* 79.3 (Jul. 1957), pp. 497–516, DOI: 10/dtf2g6 (cit. on pp. 4, 32, 49).

D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs", in *Proc. IEEE Int. Conf. Robotics and Automation* (Orlando, Fla., May 15–19, 2006), New York, N.Y.: IEEE Press, 2006, pp. 1243–1248, DOI: 10/bxkbmw (cit. on p. 61).

F. Feurtey, "Simulating the collision avoidance behavior of pedestrians", Master's thesis, Dept. Electronic Engineering, Univ. Tokyo, Tokyo, Japan, Feb. 2000 (cit. on pp. 7, 12).

P. Fiorini and D. Botturi, "Introducing service robotics to the pharmaceutical industry", in *Intell. Serv. Robot.* 1.4 (Oct. 2008), pp. 267–280, DOI: 10/ft8njb (cit. on pp. 1, 8).

P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles", in *Int. J. Robot. Res.* 17.7 (Jul. 1998), pp. 760–772, DOI: 10/ckv86z (cit. on pp. 4, 6–9, 11, 31, 32, 60).

D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance", in *IEEE Robot. Autom. Mag.* 4.1 (Mar. 1997), pp. 23–33, DOI: 10/dmgkzx (cit. on pp. 7, 8, 31, 61).

C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid", in *Proc. IEEE Int. Conf. Robotics and Automation* (Rome, Italy, Apr. 10–14, 2007), New York, N.Y.: IEEE Press, 2007, pp. 1610–1616, DOI: 10/df8zkb (cit. on pp. 8, 11).

O. Gal, Z. Shiller, and E. Rimon, "Efficient and safe on-line motion planning in dynamic environments", in *Proc. IEEE Int. Conf. Robotics and Automation* (Kobe, Japan, May 12–17, 2009), New York, N.Y.: IEEE Press, 2009, pp. 88–93, DOI: 10/c76fgb (cit. on p. 8).

R. Geraerts, A. Kamphuis, I. Karamouzas, and M. Overmars, "Using the corridor map method for path planning for a large number of characters", in *Motion in Games, Proc. 1st Int. Workshop Motion in Games* (Utrecht, Netherlands, Jun. 14–17, 2008), A. Egges, A. Kamphuis, and M. Overmars, eds., Lecture Notes Computer Science 5277, Heidelberg, Germany: Springer, 2008, pp. 11–22, DOI: 10/c9jhdm (cit. on pp. 2, 61).

A. R. Girard, A. S. Howell, and J. K. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles", in *Proc. IEEE Conf. Decision and Control* (Nassau, Bahamas, Dec. 14–17, 2004), W. Gong, ed., vol. 1, New York, N.Y.: IEEE Press, 2004, pp. 620–625, DOI: 10/fw76xg (cit. on p. 1).

S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying", in *Proc. IEEE Int. Conf. Robotics and Automation* (Kobe, Japan, May 12–17, 2009), New York, N.Y.: IEEE Press, 2009, pp. 2878–2883, DOI: 10/fwknj6 (cit. on p. 1).

S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "ClearPath: highly parallel collision avoidance for multi-agent simulation", in *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (New Orleans, La., Aug. 1–2, 2009), D. W. Fellner and S. N. Spencer, eds., New York, N.Y.: ACM Press, 2009, pp. 177–187, DOI: 10/cmk8rv (cit. on pp. 8, 15, 16, 32, 62, 68).

S. J. Guy, S. Curtis, M. C. Lin, and D. Manocha, "Least-effort trajectories lead to emergent crowd behaviors", in *Phys. Rev. E* 85.1, 016110 (Jan. 2012), DOI: 10/fzbkx7 (cit. on pp. 61, 63).

P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", in *IEEE Trans. Syst. Sci. Cybern.* 4.2 (Jul. 1968), pp. 100–107, DOI: 10/c9zgc4 (cit. on pp. 2, 61).

D. Helbing and P. Molnár, "Social force model for pedestrian dynamics", in *Phys. Rev. E* 51.5 (May 1995), pp. 4282–4286, DOI: 10/fn55b2 (cit. on pp. 3, 61).

S. Hertel and K. Mehlhorn, “Fast triangulation of the plane with respect to simple polygons”, in *Inform. Control* 64.1–3 (Jan.–Mar. 1985): Special issue on Int. Conf. Foundations of Computation Theory, pp. 52–76, DOI: 10/fkttg5 (cit. on p. 61).

D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles”, in *Int. J. Robot. Res.* 21.3 (Mar. 2002): Special issue on algorithmic foundations of robotics, pp. 233–255, DOI: 10/bbxcfg (cit. on pp. 7, 8, 61).

L. Jaillet and T. Simeon, “A PRM-based motion planner for dynamically changing environments”, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Sendai, Japan, Sep. 28–Oct. 2, 2004), vol. 2, New York, N.Y.: IEEE Press, 2004, pp. 1606–1611, DOI: 10/d4rqzq (cit. on p. 61).

J. L. Jones, N. E. Mack, D. M. Nugent, and P. E. Sandin, “Autonomous floor-cleaning robot”, U.S. pat. 6883201, Apr. 26, 2005 (cit. on pp. 1, 23).

M. Kallmann, “Shortest paths with arbitrary clearance from navigation meshes”, in *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation* (Madrid, Spain, Jul. 2–4, 2010), Z. Popovic and M. A. Otaduy, eds., Aire-la-Ville, Switzerland: Eurographics Assoc., 2010, pp. 159–168, ACM: 1921451 (cit. on p. 61).

M. Kallmann and M. Mataric, “Motion planning using dynamic roadmaps”, in *Proc. IEEE Int. Conf. Robotics and Automation* (Barcelona, Spain, Apr.–May 2004), vol. 5, New York, N.Y.: IEEE Press, 2004, pp. 4399–4404, DOI: 10/bsgsst (cit. on p. 61).

R. E. Kalman, “A new approach to linear filtering and prediction problems”, in *Trans. ASME—J. Basic Eng.* 82.1 (Mar. 1960), pp. 35–45, DOI: 10/dmftj3 (cit. on pp. 18, 42).

K. Kant and S. W. Zucker, “Towards efficient trajectory planning: the path-velocity decomposition”, in *Int. J. Robot. Res.* 5.3 (Sep. 1986), pp. 72–89, DOI: 10/bxh4m9 (cit. on p. 31).

I. Karamouzas and M. Overmars, “Simulating the local behaviour of small pedestrian groups”, in *Proc. ACM Symp. Virtual Reality Software and Technology* (Hong Kong, China, Nov. 22–24, 2010), G. Baciu, R. W. H. Lau, M. Lin, T. Komura, and Q. Peng, eds., New York, N.Y.: ACM Press, 2010, pp. 183–190, DOI: 10/b6hd7x (cit. on p. 61).

H. Kato and M. Billinghurst, “Marker tracking and HMD calibration for a video-based augmented reality conferencing system”, in *Proc. IEEE/ACM Int. Workshop Augmented Reality* (San Francisco, Calif., Oct. 20–21, 1999), New York, N.Y.: IEEE Press, 1999, pp. 85–94, DOI: 10/bxz7t4 (cit. on pp. 23, 42).

L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, in *IEEE Trans. Robot. Autom.* 12.4 (Aug. 1996), pp. 566–580, DOI: 10/fsgth3 (cit. on pp. 2, 8, 17, 61).

O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots”, in *Int. J. Robot. Res.* 5.1 (Mar. 1986), pp. 90–98, DOI: 10/d6f495 (cit. on p. 2).

B. Kluge, D. Bank, E. Prassler, and M. Strobel, “Coordinating the motion of a human and a robot in a crowded, natural environment”, in *Advances in Human-Robot Interaction*, E. Prassler, G. Lawitzky, A. Stopp, G. Grunwald, M. Hägele, R. Dillmann, and I. Iossifidis, eds., Springer Tracts Advanced Robotics 14, Heidelberg, Germany: Springer, 2004, pp. 207–219 (cit. on p. 76).

B. Kluge and E. Prassler, "Recursive probabilistic velocity obstacles for reflective navigation", in *Field and Service Robotics: Recent Advances in Research and Applications, Proc. 4th Int. Conf. Field and Service Robotics* (Lake Yamanaka, Japan, Jul. 14–16, 2003), S. Yuta, H. Asama, S. Thrun, E. Prassler, and T. Tsubouchi, eds., Springer Tracts Advanced Robotics 24, Heidelberg, Germany: Springer, 2006, pp. 71–79, DOI: 10/d68qs7 (cit. on pp. 8, 62).

F. Large, S. Sckhavat, Z. Shiller, and C. Laugier, "Using non-linear velocity obstacles to plan motions in a dynamic environment", in *Proc. Int. Conf. Control, Automation, Robotics, and Vision* (Singapore, Dec. 2–5, 2002), M. J. Er, ed., vol. 2, New York, N.Y.: IEEE Press, 2002, pp. 734–739, DOI: 10/cjqd39 (cit. on p. 8).

J.-C. Latombe, "A fast path planner for a car-like indoor mobile robot", in *Proc. AAAI Nat. Conf. Artificial Intelligence* (Anaheim, Calif., Jul. 14–19, 1991), T. L. Dean and K. McKeown, eds., vol. 2, Menlo Park, Calif.: AAAI/MIT Press, 1991, pp. 659–665 (cit. on pp. 4, 5, 32, 48, 49, 51).

J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots", in *Robot Motion Planning and Control*, J.-P. Laumond, ed., Lecture Notes Control and Information Sciences 229, Heidelberg, Germany: Springer, 1998, pp. 1–53, DOI: 10/c97pmn (cit. on pp. 4, 5, 32, 48, 49, 51).

S. M. LaValle, *Planning Algorithms*, Cambridge, United Kingdom: Cambridge Univ. Press, 2006, ISBN: 978-0-521-86205-9 (cit. on pp. 4, 5, 22, 32, 48, 49, 51).

S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning", in *Int. J. Robot. Res.* 20.5 (May 2001), pp. 378–400, DOI: 10/bnxbnz (cit. on pp. 2, 8, 17, 61).

D. Longo and G. Muscato, "The Alicia³ climbing robot: a three-module robot for automatic wall inspection", in *IEEE Robot. Autom. Mag.* 13.1 (Mar. 2006), pp. 42–50, DOI: 10/cmnnvq (cit. on p. 1).

T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles", in *Commun. ACM* 22.10 (Oct. 1979), pp. 560–570, DOI: 10/b9vhdb (cit. on p. 2).

N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multirobot teams", in *IEEE Robot. Autom. Mag.* 15.1 (Mar. 2008), pp. 53–61, DOI: 10/ft33g3 (cit. on p. 1).

C. Ó'Dúnlaing and C. K. Yap, "A 'retraction' method for planning the motion of a disc", in *J. Algorithm* 6.1 (Mar. 1985), pp. 104–111, DOI: 10/fpmk22 (cit. on p. 2).

L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems", in *IEEE Trans. Robot.* 23.6 (Dec. 2007), pp. 1170–1183, DOI: 10/d4m5qq (cit. on pp. 8, 76).

S. Pedduri, K. M. Krishna, and H. Hexmoor, "Cooperative navigation function based navigation of multiple mobile robots", in *Proc. Int. Conf. Integration of Knowledge Intensive Multi-Agent Systems* (Waltham, Mass., Apr. 30–May 3, 2007), J. Si and J. Fontanari, eds., New York, N.Y.: IEEE Press, 2007, pp. 277–282, DOI: 10/ctrkzq (cit. on p. 8).

J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths", in *Int. J. Robot. Res.* 24.4 (Apr. 2005), pp. 295–310, DOI: 10/cgm5bn (cit. on p. 32).

S. Petti and T. Fraichard, "Safe motion planning in dynamic environments", in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Edmonton, Alberta, Canada, Aug. 2–6, 2005), New York, N.Y.: IEEE Press, 2005, pp. 2210–2215, DOI: 10/bvjrqn (cit. on pp. 7, 8, 31, 61).

J. Pettré, J.-P. Laumond, and D. Thalmann, "A navigation graph for real-time crowd animation on multilayered and uneven terrain", in *Proc. Int. Workshop Crowd Simulation* (Lausanne, Switzerland, Nov. 24–25, 2005), S. R. Musse and D. Thalmann, eds., Lausanne, Switzerland: ÉPFL, 2005, pp. 81–89, ISBN: 978-2-8399-0118-5 (cit. on p. 61).

R. Philippsen and R. Siegwart, "Smooth and efficient obstacle avoidance for a tour guide robot", in *Proc. IEEE Int. Conf. Robotics and Automation* (Taipei, Taiwan, Sep. 14–19, 2003), vol. 1, New York, N.Y.: IEEE Press, 2003, pp. 446–451, DOI: 10/dxv3qb (cit. on p. 1).

E. Prassler, J. Scholz, and P. Fiorini, "Navigating a robotic wheelchair in a railway station during rush hour", in *Int. J. Robot. Res.* 18.7 (Jul. 1999): Special issue on field and service robotics, pp. 711–727, DOI: 10/crn85s (cit. on pp. 1, 8, 11).

J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards", in *Pac. J. Math.* 145.2 (Oct. 1990), pp. 367–393 (cit. on pp. 32, 49).

J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles", in *Proc. IEEE Annu. Symp. Foundations of Computer Science* (Portland, Ore., Oct. 21–23, 1985), New York, N.Y.: IEEE Press, 1985, pp. 144–154, DOI: 10/c6n34x (cit. on p. 3).

C. Reynolds, "Flocks, herds and schools: a distributed behavioral model", in *ACM SIGGRAPH Comput. Graph.* 21.4 (Jul. 1987): Special issue on 14th Annu. Conf. Computer Graphics and Interactive Techniques, pp. 25–34, DOI: 10/djp7nr (cit. on pp. 4, 61).

C. Reynolds, "Big fast crowds on PS3", in *Proc. ACM SIGGRAPH Video Game Symp.* (Boston, Mass., Jul. 29–30, 2006), A. Heirich and D. Thomas, eds., New York, N.Y.: ACM Press, 2006, pp. 113–121, DOI: 10/fnrdnh (cit. on p. 1).

A. Richards, J. Bellingham, M. Tillerson, and J. P. How, "Coordination and control of multiple UAVs", in *Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit* (Monterey, Calif., Aug. 5–8, 2002), Reston, Va.: AIAA, 2002, pp. 1936–1941 (cit. on p. 49).

A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming", in *Proc. American Control Conf.* (Anchorage, Alaska, May 8–10, 2002), E. Misawa, ed., vol. 3, New York, N.Y.: IEEE Press, 2002, pp. 1936–1941, DOI: 10/bk55mn (cit. on pp. 48, 49).

S. I. Roumeliotis and I. M. Rekleitis, "Propagation of uncertainty in cooperative multirobot localization: analysis and experimental results", in *Auton. Robot.* 17.1 (Jul. 2004): Special issue on analysis and experiments in distributed multi-robot systems, pp. 41–54, DOI: 10/bgmx7w (cit. on p. 77).

A. Scheuer and C. Laugier, "Planning sub-optimal and continuous-curvature paths for car-like robots", in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Victoria, British Columbia, Canada, Oct. 10–17, 1998), vol. 1, New York, N.Y.: IEEE Press, 1998, pp. 25–31, DOI: 10/fs5w4k (cit. on p. 52).

J. T. Schwartz and M. Sharir, "On the 'piano movers' problem II: general techniques for computing topological properties of real algebraic manifolds", in *Adv. Appl. Math.* 4.3 (Sep. 1983), pp. 298–351, DOI: 10/fpkkbp (cit. on p. 2).

Z. Shiller, R. Prasanna, and J. Salinger, "A unified approach to forward and lane-change collision warning for driver assistance and situational awareness", in *Proc. SAE World Congr. and Exhibition* (Detroit, Mich., Apr. 14–17, 2008), 2008-01-0204, Warrendale, Pa.: SAE Int., 2008, DOI: 10/cbggrz (cit. on p. 8).

J. Snape, S. J. Guy, J. van den Berg, and D. Manocha, "Smooth coordination and navigation for multiple differential-drive robots", in *Experimental Robotics, Proc. 12th Int. Symp. Experimental Robotics* (New Delhi, India, Dec. 18–21, 2010), O. Khatib, V. Kumar, and G. Sukhatme, eds., Springer Tracts Advanced Robotics 79, Heidelberg, Germany: Springer, 2013, pp. 601–613 (cit. on p. 77).

G. Snook, "Simplified 3D movement and pathfinding using navigation meshes", in *Game Programming Gems*, M. DeLoura, ed., Hingham, Mass.: Charles River, 2000, ch. 3, pp. 288–304, ISBN: 978-1-58450-049-0 (cit. on pp. 2, 60, 61).

A. Stentz, "The focussed D* algorithm for real-time replanning", in *IJCAI-95, Proc. 14th Int. Joint Conf. Artificial Intelligence* (Montréal, Québec, Canada, Aug. 20–25, 1995), C. Mellish, ed., vol. 2, Burlington, Mass.: Morgan Kaufmann, 1995, pp. 1652–1659, ACM: 1643113 (cit. on p. 61).

H. J. Sussmann and G. Tang, "Shortest paths for the Reeds-Shepp car: a worked out example of the use of geometric techniques in nonlinear optimal control", Tech. rep. SYNCON 91-10, Rutgers Univ., Piscataway, N.J., 1991 (cit. on p. 49).

W. van Toll, A. Cook IV, and R. Geraerts, "Navigation meshes for realistic multi-layered environments", in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (San Francisco, Calif., Sep. 25–30, 2011), New York, N.Y.: IEEE Press, 2011, pp. 3526–3532, DOI: 10/cw7xgp (cit. on p. 61).

P. Tozour, "Search space representations", in *AI Game Programming Wisdom 2*, S. Rabin, ed., Hingham, Mass.: Charles River, 2003, ch. 2, pp. 85–102, ISBN: 978-1-58450-289-0 (cit. on p. 61).

L. Tychonievich, D. Zaret, J. Mantegna, R. Evans, E. Muehle, and S. Martin, "A maneuvering-board approach to path planning with moving obstacles", in *IJCAI-89, Proc. 11th Int. Joint Conf. Artificial Intelligence* (Detroit, Mich., Aug. 20–25, 1989), N. S. Sridharan, ed., vol. 2, Burlington, Mass.: Morgan Kaufmann, 1989, pp. 1017–1021, ACM: 1623918 (cit. on pp. 8, 32, 62).

H. van Welbergen, B. van Basten, A. Egges, Z. Ruttkay, and M. Overmars, "Real time animation of virtual humans: a trade-off between naturalness and control", in *Comput. Graph. Forum* 29.8 (Dec. 2010), pp. 2530–2554, DOI: 10/fr88jz (cit. on p. 62).

M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments", in *Proc. IEEE Int. Conf. Robotics and Automation* (Rome, Italy, Apr. 10–14, 2007), New York, N.Y.: IEEE Press, 2007, pp. 1603–1609, DOI: 10/ftxddq (cit. on p. 61).