Probabilistic Inverse Velocity Obstacles for Navigation under Uncertainty

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Electronics and Communication Engineering
by Research

by

SriSai Naga Jyotish Poonganam
20161217
srisai.poonganam@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2020

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled '**Probabilistic Inverse Velocity Obstacles for Navigation under Uncertainty** " by **SriSai Naga Jyotish Poonganam**, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. K. Madhava Krishna

To my mother

Acknowledgments

I would like to thank Prof. K. Madhava Krishna, Professor, and Head of Robotics Research Center (RRC), IIIT Hyderabad, for allowing me to work at RRC. I also thank him for his support and insightful guidance throughout this journey. His invaluable experience has helped me a lot while making important decisions.

I would extend my heartfelt gratitude to Bharath Gopalakrishnan for his mentorship and support during all the ups and downs of my journey. I'm deeply grateful for his invaluable efforts, encouragement in materializing this thesis. I'm indebted to Yash Goel and Bhargav for their unconditional support. I will always be grateful to Krishna Chaitanya, Yashwanth, Harsha, Vasisht, Shanmukh, Siddhartha, Josyula Gopala Krishna, Gokul, Unnikrishnan, and many more invaluable friends who were always there when I needed help. My life at IIIT Hyderabad would not be this wonderful without their company. Special thanks to S. P. Mohanty for his enormous support and meticulous reviews on my thesis.

Above all, I would like to thank my parents and brother for their unconditional love and moral support.

Abstract

Autonomous navigation gained a lot of attention in the past decade. Motion planning and collision avoidance are the core things that drive these systems. Several advances resulted in different algorithms to tackle the problem of motion planning and collision avoidance. The crucial question at this stage is how reliable and robust are these systems in a real-life scenario. Recent works in the field have proposed algorithms that account for the uncertainties that arise in a real-life scenario. But the work is sparse. In an attempt to address the reliability and robustness of the navigation frameworks in real-life scenarios, we present two approaches that incorporate the uncertainty in the sensor measurements.

First, we present a clever way, Inverse Velocity Obstacle (IVO), to achieve better collision detection based on the popularly used velocity obstacles. The proposed method stems from the concept of velocity obstacle and can be applied for both single-agent and multi-agent systems. It focuses on computing collision-free maneuvers without any knowledge or assumption on the position and the velocity of the robot. We achieve this by reformulating the velocity obstacle in an ego-centric frame. This is a significant step towards improving real-time implementations of collision avoidance in dynamic environments as there is no dependency on state estimation techniques to infer the robot's position and velocity. We evaluate IVO for both single-agent and multi-agent cases in different scenarios and show it's efficacy over the existing formulations. We also show the real-time scalability of the proposed methodology.

Next, we present a probabilistic variant of the same to handle the state estimation and motion uncertainties that arise due to the other participants of the environment. We present an algorithmic framework that computes collision-free velocities for the robot dynamic and uncertain environments. We make no assumptions on the nature of the uncertainties and model them as non-parametric probability distributions. In our Probabilistic Inverse Velocity Obstacle (PIVO), we pose the collision-free navigation problem as an optimization problem by reformulating the velocity conditions of IVO as chance constraints that take the uncertainty into account. We also define a lower bound on the confidence of collision avoidance maneuvers that are a result of the optimization. We demonstrate the algorithm's ability to generate safe trajectories under highly uncertain environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Let's consider a hypothetical multi-tasking robot. We ask the robot to fetch a glass of water. What are the steps that the robot has to perform?

1	**Move** to the location of the water dispenser.
2	**Move** the arm of the robot to the glass utensil.
3	Pickup the glass utensil.
4	**Move** the arm with the glass utensil to the water dispensing area.
5	**Move** the arm to the water dispensing switch.
6	Hold the water dispensing switch till the glass is full.
7	**Move** the arm to the glass utensil filled with water.
8	Pickup the glass utensil.
9	**Move** to the initial location (home location).

Table 1.1 Subtasks for the hypothetical robot to fetch a glass of water

There are nine subtasks that the robot has to fulfill to complete its original task of fetching a glass of water. Out of these nine subtasks, six subtasks involve motion planning. Not only this, imagine any task that this hypothetical robot has to do and you will realize that motion planning lies at the heart of all these tasks. Now that we established that motion planning is crucial, what exactly does it mean to do this? We essentially want to compute a set of controls following which the robot starts from an initial position and reaches a final position while

- Avoiding collisions with other objects in the environment.
- Following a set of kinematic constraints specific to the robot *(not covered)*.
- Maintaining smooth state transitions.
- Minimizing the wear and tear of the hardware.

Collision avoidance of dynamic obstacles by planar robots is a well-studied problem. Numerous approaches solve this problem using geometric methods. They work well when the conditions are ideal. The conditions here are the perfect state estimates of the robot and the obstacles in the environment. Most of these approaches assume that they have access to the ideal state estimates and solve the problem. But the sensors that capture these state estimates are not so ideal and perfect in real life. They are noisy. There is always some uncertainty involved in the measurements made using these sensors. This can be seen in the figure 1.1.

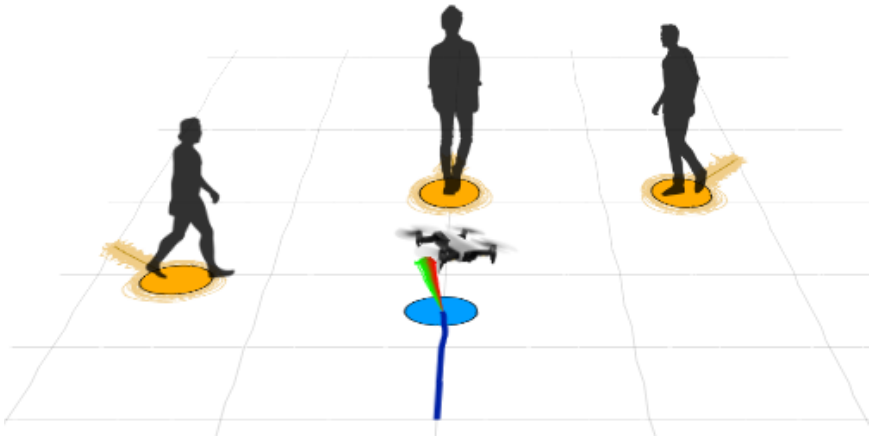


Figure 1.1 An illustration of a quadrotor navigating in a dynamic environment. The quadrotor and the obstacles are approximated using circular discs. The noise in the position estimates of the obstacles is shown using translucent orange discs. The perturbation on the control as a result of actuation noise is shown using different quivers. The green quivers represent the set of controls that avoid the collision while the red quivers represent the set of controls that lead to a colliding maneuver.

Unfortunately, most of the classical approaches do not work well when the state estimates are noisy. A commonly adopted method to deal with the noise is increasing the volume of the bounding box used to enclose the obstacles and the robot proportional to the magnitude of noise in the measurement. Though this works in most of the cases, it often leads to very conservative maneuvers. Another commonly adopted approach is taking a Gaussian approximation of the original noise. Though this method tends to work better than the bounding box based methods, this method leads to conservative trajectories as well. This is graphically represented in figure 1.2.

1.2 Related Work

In this section, we present an overview of the approaches to collision avoidance and navigation in dynamic environments. Quite a few approaches [2], [5], [7], [18] assume that the obstacles are static and plan for the control to avoid the collision. In the case of moving obstacles, these approaches replan based on the updated positions of the obstacles. But these fail to generate safe trajectories around fast-

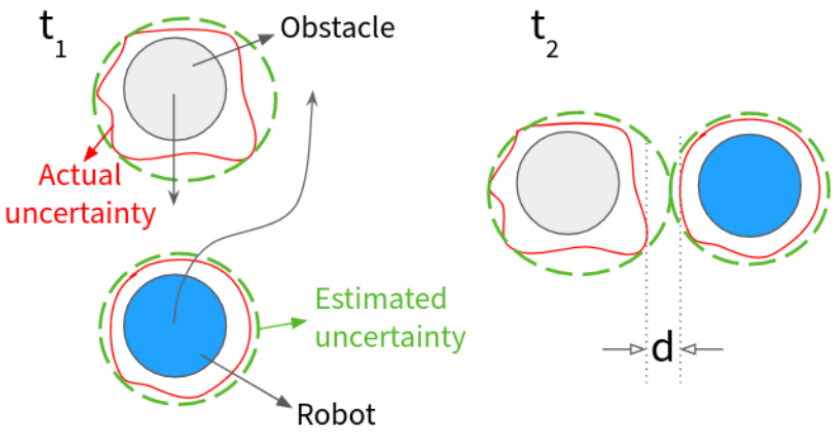


Figure 1.2 The figure shows how a poor approximation of noise leads to more conservative maneuvers. The robot and obstacle are represented using blue and gray discs respectively. The red line around the robot and obstacle is the noise in their positions. The green dotted line represents an approximation using known parametric distributions. At time t_2, when the robot and obstacle avoid each other with respect to the approximate noise estimations, they leave a margin of d that could be minimized with a better noise approximation.

moving obstacles. In [8], [4], [16], [27] the future position of obstacles are computed by extrapolating with the current velocity to handle high velocities. But these approaches cannot handle the reactive nature of other agents. Many works like [29], [36], [34], [10] focused on crowd simulation in which each agent considers the other agents as obstacles and navigates independently.

Centralized planning scheme on a given configuration space in the case of multiple agents is presented in [24], [31]. These works majorly focus on optimal coordination and are difficult to implement in real-time. Works like Velocity Obstacle [6] are quite popular in the literature of collision avoidance for dynamic obstacle collision avoidance. A variant called Recursive Velocity Obstacles [20] is proposed, which considers the reactive behavior of the other participants. However, this approach leads to the oscillations of the agents that sometimes may not converge. To address the issue, the velocity obstacle is extended as Reciprocal Velocity Obstacle (RVO) in [38], where both the agents that are on the course of the collision, select the velocities that bring them out of collision assuming that the other agent implements the same algorithm. But this requires the knowledge of current pose and velocity of the obstacle which might bottleneck the update rates during real-time implementation. It is also worth mentioning that the state estimations of the robot are often noisy. They are several other extensions of Velocity Obstacle like [32] [23].

To address this issue, we present an ego-centric framework called Inverse Velocity Obstacles (IVO), that does not require the knowledge of the robot's pose and velocity. This eliminates the state estimation layer reducing the computational time (for state estimation) and false collision detection due to noisy estimates.

Quite a few works like [32], [12], [23] have branched out from the concept of velocity obstacle as an elegant extension in different applications. Our work Inverse Velocity Obstacle [17] is one such extension that removes the dependency on the self-state estimations of the robot in an autonomous system thereby improving the computational complexity significantly. Works like [37] [1], that are extensions of Velocity Obstacles introduced ways to incorporate car-like obstacles. All the above methods fail to generate collision-free velocities in an uncertain dynamic environment.

In [9] the uncertainty associated with the obstacle is handled by modeling it as a Gaussian Process. [19], [33], [3] model the uncertainty as Gaussian random variables parameterized by their mean and covariance. But these works simply consider the uncertainty that arises in the state estimation of other participants and assume a deterministic model for the robot. Being an ego-centric framework our approach eliminates the concept of self-state estimation, which is a shortcoming in the previous methods. In [26] the presented algorithm considers both the perception and motion uncertainty but is computationally heavy for calculating tight probability bound at each step.

We also cite the works like [15] that defines the human interaction social norms to develop models. In some approaches like [21], they consider humans as static obstacles or to cooperate with the robots that the robot shares the environment with. The motion planner proposed in [25] generates the trajectory by considering the scenario where all the humans perform their worst possible motion. Learning techniques like [22], [35] with filter tracking and motion models integrated have been implemented that take into account the past error while making future predictions.

Most of the above methods do not handle uncertainty and even if they do they resort to parametric (Gaussian) models of state, observation, and control noise. In contrast, the proposed formulation assumes non parametric noise models for the uncertainty and bypasses the need to handle state uncertainty. As a consequence, it shows less conservative trajectories without compromising the safety of the interacting agents.

1.3 Contributions

This thesis presents two motion planning frameworks that are capable of performing well under uncertain state estimations. In the first approach, we present a way to cleverly deal with the noisy state estimates of the robot. This aids in better collision detection and also eliminates the need to transform the sensor measurements to a frame that can be used by the motion planning algorithms. In the second approach, we present a novel probabilistic optimization framework that takes into account the perception and state noise while having a computation efficiency close to a deterministic solution. The key contributions of this work are given below.

- The principal contribution of the work is the construction of efficient collision avoidance schemes for autonomous navigation called Inverse Velocity Obstacles (IVO) and its probabilistic extension, Probabilistic Inverse Velocity Obstacles (PIVO).

- We show that IVO is amenable to a probabilistic setting as we cast it into a chance-constrained formulation that solves for control actions accounting for and respecting chance constraints. Chance constraints are probabilistic constraints of the form $\Pr(f(\cdot) > 0) > \eta$ which are typically intractable. However, they were shown effective in collision avoidance settings by solving for surrogate constraints that are tractable and provide closed-form solutions [13], [11], [39].
- We show ablation studies that portray the above advantages vis a vis chance-constrained formulations that take into account the state and velocity noise of the drone/ego-vehicle.

IVO is a variant of the widely used velocity obstacle [6] and can be used as a drop-in replacement in any framework that is based on the velocity obstacle. We show how bypassing the need to estimate the drones ego state results in a significant decrease in trajectory lengths without compromising on safety. Further in PIVO, we bypass the need to model uncertainty in a parametric setting even as we model the observation/measurement uncertainties non-parametrically and show the efficacy of our formulation in the non-parametric setting.

1.4 Layout of the thesis

Chapter 2 presents a deterministic navigation framework, IVO, with results for single and multi-agent scenarios. Chapter 3 presents a probabilistic extension of IVO called PIVO that accounts for the perception and actuation uncertainties in a computationally efficient way using chance constraints. In chapter 4, we briefly discuss how the proposed methods can be extended to robots with different motion models.

1.5 Notations ans symbols

In this thesis, we represent the vectors in bold letters, $\mathbf{x}$, matrices in capital, M, and sets in mathcal, $\mathcal{A}$. $\|\mathbf{x}\|$ denotes the Euclidean norm of $\mathbf{x}$. For a random variable x, $\hat{\text{x}}$ denotes the mean. $\Pr(\cdot)$ denotes the probability of an event while $P(\cdot)$ denotes the probability density function. The subscript $\cdot_t$ indicates the value at time t while the superscript $\cdot^j$ indicates the j^{th} robot or obstacle. Some of the commonly used symbols and notations are summarized in the table 1.2. We also define some notations in the first place of their use.

Table 1.2 Common notations and symbols

Symbol	Description
$({}^{r}\mathbf{x}_t, {}^{r}\,\dot{\mathbf{x}}_t)$	(Position, Velocity) of the robot at time t in global frame
$({}^{o}\mathbf{x}_t^j, {}^{o}\,\dot{\mathbf{x}}_t^j)$	(Position, Velocity) of the j^{th} obstacle at time t in global frame
$({}^{o}_{r}\mathbf{x}_t^j, {}^{o}_{r}\,\dot{\mathbf{x}}_t^j)$	(Position, Velocity) of the j^{th} obstacle at time t in robot's frame
$\mathbf{u}_t$	Control input to the robot at time t
$f_t^j(\cdot) \leq 0$	Deterministic collision avoidance constraint
$P(f_t^j(\cdot))$	Distribution of $f_t^j(\cdot)$ under uncertainty
η	Probability of satisfaction of the chance constraint

Chapter 2

Inverse Velocity Obstacle

2.1 Contributions

- We propose an algorithm, IVO, for robust and efficient collision detection. We also show that an ego-centric approach leads to better collision detection.
- We propose an optimization based, computationally efficient collision avoidance navigation framework using IVO to compute the robot's controls.
- Further, we extend the proposed navigation framework to a multi-agent setting.

2.2 Preliminaries

In this chapter, we present a deterministic collision avoidance and navigation framework using the concept of Inverse Velocity Obstacles (IVO). We discuss the formulation of the collision avoidance constraints in detail and how it compares with the popularly used velocity obstacle.

2.2.1 Velocity Obstacle

In this section, we briefly review the original concept of Velocity Obstacle and analyze its behavior in the presence of the state, actuation, and perception uncertainties.

2.2.1.1 Definition

Consider a mobile robot (our agent) and an obstacle, both taking the shape of a disc of radius r_A and r_B respectively, be denoted by A and B. The velocity obstacle for robot A induced by obstacle B, denoted by $VO_{A|B}$, is the set of velocities of A which can result in a collision with B at some point in the future. Let C_A and C_B represent the centers of A and B respectively. The robot and obstacle are geometrically modified such that the robot takes the form of a point object and the obstacle grows

its radius to $r_A + r_B$. If B is a static obstacle, a cone can be constructed with the vertex on A and the edges touching B as shown in figure 2.1. This cone represents the set of velocities of A which leads to a collision. In case the obstacle is in motion, it is assumed to be static by taking a relative velocity of A.

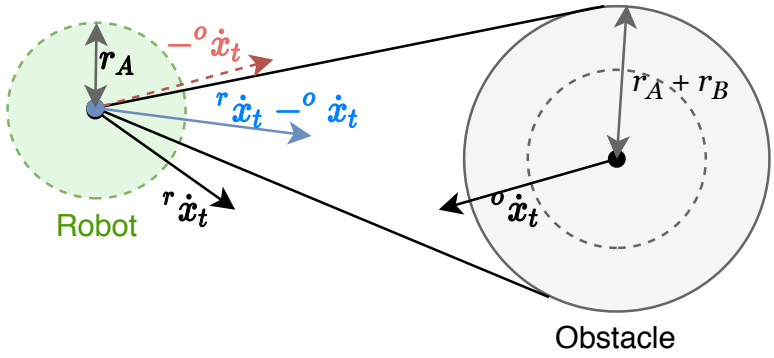


Figure 2.1 Velocity obstacle for agent A (shown in green) induced by obstacle B (shown in gray).

2.2.1.2 Implementation problems

The obvious assumption from the definition of the velocity obstacle is that we need to track the velocity of the robot along with the position and velocity of the obstacle. In the case of planning trajectories on a global frame, we also need to track the positions of robot and obstacle with respect to a global frame. Though we can plan trajectories in the robot's frame, this still needs us to have an estimation of the velocity of the robot. Generally, we take the instantaneous velocity from a sensor. This accounts for an additional noise in estimation of the velocity of the robot apart from the noise we end up having in the states of the obstacle. Other prominent methods include state estimation using SLAM which is not as reliable as the feed from the sensor since SLAM methods tend to break when complex maneuvers are involved.

2.3 Inverse Velocity Obstacle

In this section, we propose a new concept of "Inverse Velocity Obstacle" to minimize the uncertainty in collision detection during the planning phase. This integrates into our optimization framework which provides controls leading to collision-free and smooth trajectories.

2.3.1 Concept

The idea is simple - Instead of assuming that the obstacle is stationary, we assume that the robot is stationary and get a relative velocity vector for the obstacle. At this point, our robot is stationary at the origin (since we are in an ego-frame). We also make the obstacles point objects and grow the radius of

the robot to $r_A + r_B$. Now, we find a relative velocity for our robot (which is stationary in the relative frame) which is outside the collision cone. A simple case is demonstrated in figure 2.2. We show that we can calculate the relative velocity of the obstacle as seen by the agent using the ego-centric observations of the obstacle by the agent at two consecutive time instances, here t and $t+\tau$, as shown in the equation (2.1).

$$ {}^{o}_{r}\dot{\mathbf{x}}^{j}_{t+\tau} = \frac{{}^{o}_{r}\mathbf{x}^{j}_{t+\tau} - {}^{o}_{r}\mathbf{x}^{j}_{t}}{\tau} \tag{2.1} $$

At any time, t suppose the global position of the obstacle moving with velocity ${}^{o}\dot{\mathbf{x}}^{j}_{t}$ and agent moving with velocity ${}^{r}\dot{\mathbf{x}}_t$ be ${}^{o}\mathbf{x}^{j}_{t}$ and ${}^{r}\mathbf{x}_t$ respectively. At the next time instance, the global positions of the obstacle and agent will be ${}^{o}\mathbf{x}^{j}_{t+\tau}$ and ${}^{r}\mathbf{x}_{t+\tau}$ respectively. The ego-centric observations of the obstacle by the agent for these instances are ${}^{o}_{r}x^{j}_{t}$ and ${}^{o}_{r}x^{j}_{t+\tau}$ for agent frame $\mathbf{F}_t$ and $\mathbf{F}_{t+\tau}$ respectively.

So, the global position of the obstacle at first instance is

$$ {}^{o}\mathbf{x}_t = T^{g}_{t}\ {}^{o}_{r}\mathbf{x}_t = {}^{o}_{r}\mathbf{x}_t + {}^{r}\mathbf{x}_t $$

where T^{g}_{t} is the transformation matrix. Similarly for the second instance we have

$$ \begin{aligned} {}^{o}\mathbf{x}_{t+\tau} &= T^{g}_{t+\tau}\ {}^{o}_{r}\mathbf{x}_{t+\tau} \\ &= {}^{o}_{r}\mathbf{x}_{t+\tau} + {}^{r}\mathbf{x}_{t+\tau} \\ &= {}^{o}_{r}\mathbf{x}_{t+\tau} + {}^{r}\mathbf{x}_t + {}^{r}\dot{\mathbf{x}}_t\tau \end{aligned} $$

Therefore the obstacle velocity in the global frame is

$$ \begin{aligned} {}^{o}\dot{\mathbf{x}}_{t+\tau} &= \frac{{}^{o}\mathbf{x}_{t+\tau} - {}^{o}\mathbf{x}_t}{\tau} \\ &= \frac{{}^{o}_{r}\mathbf{x}_{t+\tau} - {}^{o}_{r}\mathbf{x}_t}{\tau} + {}^{r}\dot{\mathbf{x}}_t \end{aligned} $$

And hence the relative velocity of the obstacle with respect to the agent is

$$ \begin{aligned} {}^{o}_{r}\dot{\mathbf{x}}_{t+\tau} &= {}^{o}\dot{\mathbf{x}}_{t+\tau} - {}^{r}\dot{\mathbf{x}}_{t+\tau} \\ &= \frac{{}^{o}_{r}\mathbf{x}_{t+\tau} - {}^{o}_{r}\mathbf{x}_t}{\tau} \end{aligned} $$

Now, we can write the collision avoidance constraint using inverse velocity obstacles,

$$ f^{j}_{t}(\cdot) \leq 0 : \frac{(\mathbf{r}^{T}_{j}\mathbf{v}_j)^2}{\|\mathbf{v}_j\|^2} - \|\mathbf{r}_j\|^2 + ({}^{r}r + {}^{o}r^{j})^2 \leq 0, \forall j \tag{2.2a} $$

$$ \mathbf{r}_j = {}^{o}_{r}\mathbf{x}^{j}_{t}, \mathbf{v}_j = {}^{o}_{r}\dot{\mathbf{x}}^{j}_{t} \tag{2.2b} $$

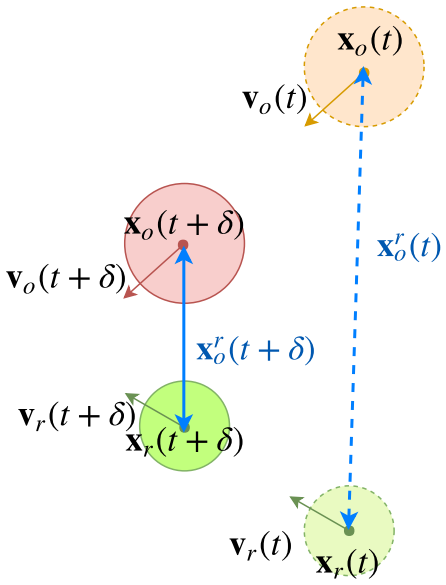
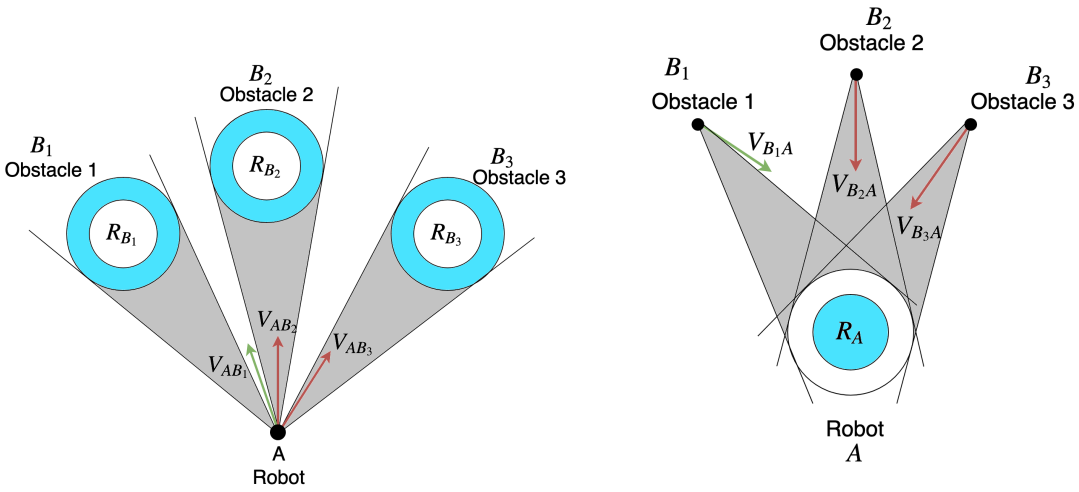


Figure 2.2 $\mathbf{x}_r$, $\mathbf{v}_r$ denote the position and velocity of the agent while $\mathbf{x}_o$ and $\mathbf{v}_o$ denote the position and velocity of the obstacle in global frame. $\mathbf{x}_o^r$ and $\mathbf{v}_o^r$ denote the position and velocity of the obstacle as seen from the agent's frame (agent is at origin and stationary in this frame).

2.3.2 Definition

Consider a mobile robot, denoted by A, and an obstacle, denoted by B, both taking the shape of a disc of radius ${}^{r}r_A$ and ${}^{o}r_B$ respectively. The traditional velocity obstacle for robot A induced by obstacle B is the range of velocities of A that can result in a collision with B at some point in the future. Here, the robot is reduced to a point while the radius of the obstacle is increased to ${}^{r}r_A + {}^{o}r_B$. The collision cone is formed with a point robot at its vertex. The inverse velocity obstacle reduces the obstacle to a point object and grows the radius of the robot to ${}^{r}r_A + {}^{o}r_B$. The collision cone is formed with the point obstacle at its vertex. It is straight forward from this that the inverse velocity obstacle is the range of velocities of B that can result in a collision with A at some point in the future. A graphical representation of collision cones formed with velocity obstacle and inverse velocity obstacle can be found in figure 2.3. We try to find a new relative velocity of the obstacle which would take the obstacle on a collision-free maneuver. We assume that the velocity of obstacle and the robot are instantaneously constant during the time interval, τ. This essentially means that the change relative velocity of the obstacle comes from the change in the velocity of the robot. The inverse velocity obstacle is given by equation (2.2).

The relative position of the obstacles can be directly retrieved from sensor data. Since the collision cone is free from the robot's position and velocity estimates, which are usually prone to high magnitudes of noise, one can completely bypass the robot's state estimations for collision detection and avoidance.



(a) Velocity obstacles for three obstacles

(b) Inverse Velocity obstacles for three obstacles

Figure 2.3 Collision cones formed with VO and IVO. In IVO, the robot is assumed to be at rest while the obstacles move towards it with their respective relative velocities. The green relative velocity lies outside the collision cone and hence not in collision whereas the red ones are in a collision.

2.4 Navigating agents

2.4.1 Single Agent

Let us start with the case of a single agent that follows a holonomic motion model and obstacles that do not have a complex behavior but move with some constant velocity. Now, consider the following minimization problem with optimization variable $\mathbf{u}_t$ which represents the controls to the agent at a time t. The goal position in the agent's frame is denoted by ${}_r\mathbf{g}$ and $\mathbf{u}_t$ is the control given to the agent, which in this case is the change in the velocity of the agent. $\mathbf{r}$ and $\mathbf{v}$ represent the position and velocity of the obstacle as seen by the agent. The smoothing factor λ can be adjusted based on the requirement. We assume that the maximum attainable velocity of the agent is $\mathbf{v}_{max}$.

$$\arg\min_{\mathbf{u}_t} J(\mathbf{u}_t) = \left\| {}^r\dot{\mathbf{x}}_t - {}^d\dot{\mathbf{x}}_t \right\|^2 + \lambda \left\| \mathbf{u}_t \right\|^2 \tag{2.3a}$$

$$\text{subject to} \quad f_t^j(\cdot) \leq 0 : \frac{(\mathbf{r}_j^T \mathbf{v}_j)^2}{\left\| \mathbf{v}_j \right\|^2} - \left\| \mathbf{r}_j \right\|^2 + ({}^r r + {}^o r^j)^2 \leq 0, \forall j \tag{2.3b}$$

$${}^d\dot{\mathbf{x}}_t = \frac{{}_r\mathbf{g}}{\left\| {}_r\mathbf{g} \right\|} \mathbf{v}_{max} \tag{2.3c}$$

$${}^r\dot{\mathbf{x}}_t = {}^r\dot{\mathbf{x}}_{t-\tau} + \mathbf{u}_t \tag{2.3d}$$

$$\mathbf{r}_j = {}^o_r\mathbf{x}_t^j, \mathbf{v}_j = {}^o_r\dot{\mathbf{x}}_t^j \tag{2.3e}$$

Desired velocity, ${}^d\dot{\mathbf{x}}_t$, is a vector towards the goal position with magnitude equal to the maximum velocity of the agent. The equation (2.3a) tries to find a control $\mathbf{u}_t$ that makes the current velocity of the

agent, ${}^{r}\dot{\mathbf{x}}_t$, as closes as possible to the desired velocity. The collision avoidance constraint, $f_t^j(\cdot)$, exists for every possible pair of agents and obstacles within the sensor range of the agent. In section 2.5.1, we experimentally show that this formulation is valid and the agent successfully avoids the obstacles and reaches the goal.

2.4.2 Multiple Agents

Let us consider n agents that use the optimization routine described in equation (2.3). In this case, the obstacles may not necessarily move with constant velocity. For the sake of simplicity, we assume that every agent moves with some instantaneous velocity during the time τ. Now, we scale the single-agent problem to n agents by considering every other agent to be an obstacle. Following this idea, a navigation algorithm for a multi-agent scenario is described in Algorithm 1.

Algorithm 1 Controls for agents in multi-agent setup

for $i = 1$ to n **do**
  for obstacle, j, in obstacles in sensor range **do**
    ${}^{o}_{r}\mathbf{x}_t^j \leftarrow$ Position of j^{th} obstacle in agent's frame
    ${}^{o}_{r}\dot{\mathbf{x}}_t^j \leftarrow ({}^{o}_{r}\mathbf{x}_t^j - {}^{o}_{r}\mathbf{x}_{t-\tau}^j) \cdot \tau$
    ${}^{o}r_j \leftarrow$ Radius of the j^{th} obstacle
    Evaluate the value of $f_t^j(\cdot)$ from equation (2.2)
  end for
  $\mathbf{u}_t^i \leftarrow \arg\min_{\mathbf{u}_t^i} J(\mathbf{u}_t)$ from equation (2.3a)
end for
return $\mathcal{U} = [\mathbf{u}_t^1\ \mathbf{u}_t^2 \dots \mathbf{u}_t^n]^T$

In section 2.5.2, we experimentally show that the algorithm works for multiple agents with large values of n.

2.5 Experimental Results

To evaluate the performance of the presented methodology we have tested both the single agent and the multi-agent scenarios. All the simulations are performed on the Intel i7 processor @ 3.2 GHz clock speed. The methodology is also validated on a real quadrotor. For this, we used Parrot Bebop2. The detailed videos of all the simulations and real-time implementations are available at [1]. All the results are obtained from a Matlab based code available at [2].

[1] https://sites.google.com/view/inverse-velocity-obstacle
[2] https://github.com/RoboticsIIITH/inverse-velocity-obstacle

2.5.1 Single agent

First, we validate our formulation in a single agent case. Figure (2.4) shows the scenario where a single agent is among five dynamic obstacles. All the participants in the environment are of the same radius and have the same speed limits. As can be seen, the agent executes safe trajectories to avoid all the obstacles and reaches the goal. The computation time for each cycle in this scenario is around 10ms making it achieve an update rate of 100Hz.

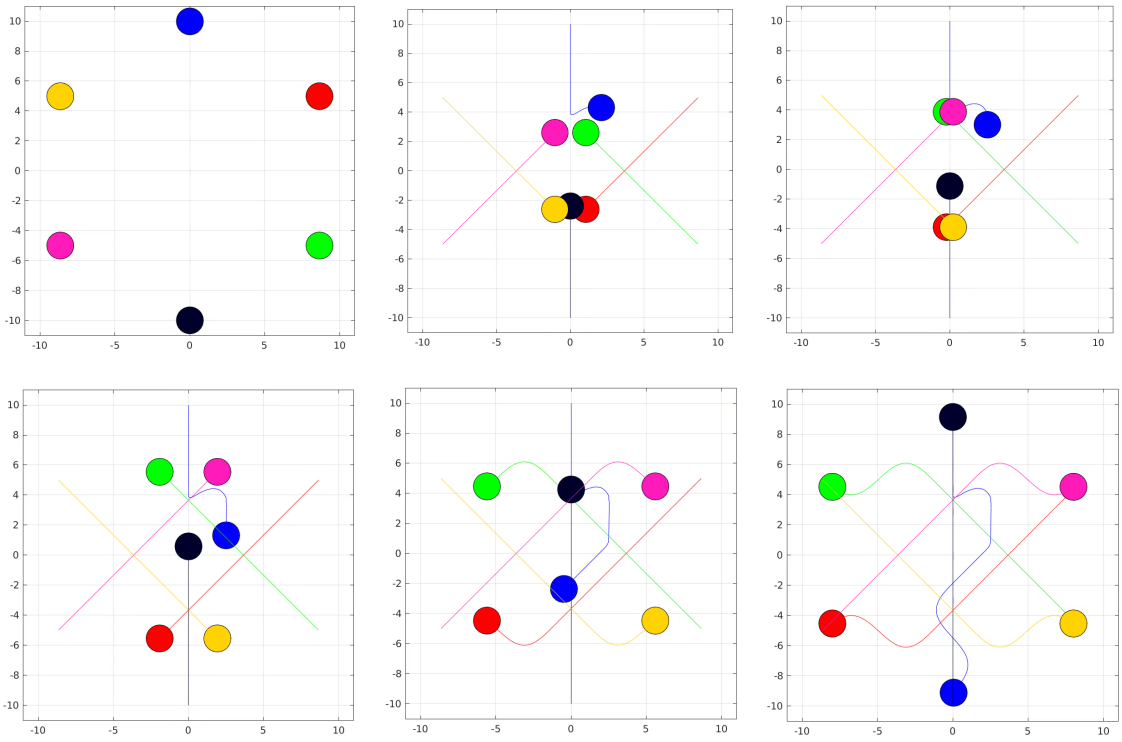


Figure 2.4 The blue disc represents the agent. Other discs are obstacles that follow a constant velocity. The figures show the trajectory taken by the agent as time progresses.

2.5.2 Multiple agents

In this section, we evaluate the performance of our Inverse Velocity Obstacle in a multi-agent collision avoidance scenario. We first evaluate for six agents scenario in an antipodal case. All the agents are of the same radius and have the same speed and acceleration limits. Figure 2.5 shows the scenario where six agents take avoidance maneuvers. All the agents plan independently considering all the other participants as potential obstacles. As shown in figure 2.5, all the agents follow safe, non-colliding trajectories avoiding each other and reach the goal. The computational time for each cycle in this scenario is 15ms with update rates of 66Hz.

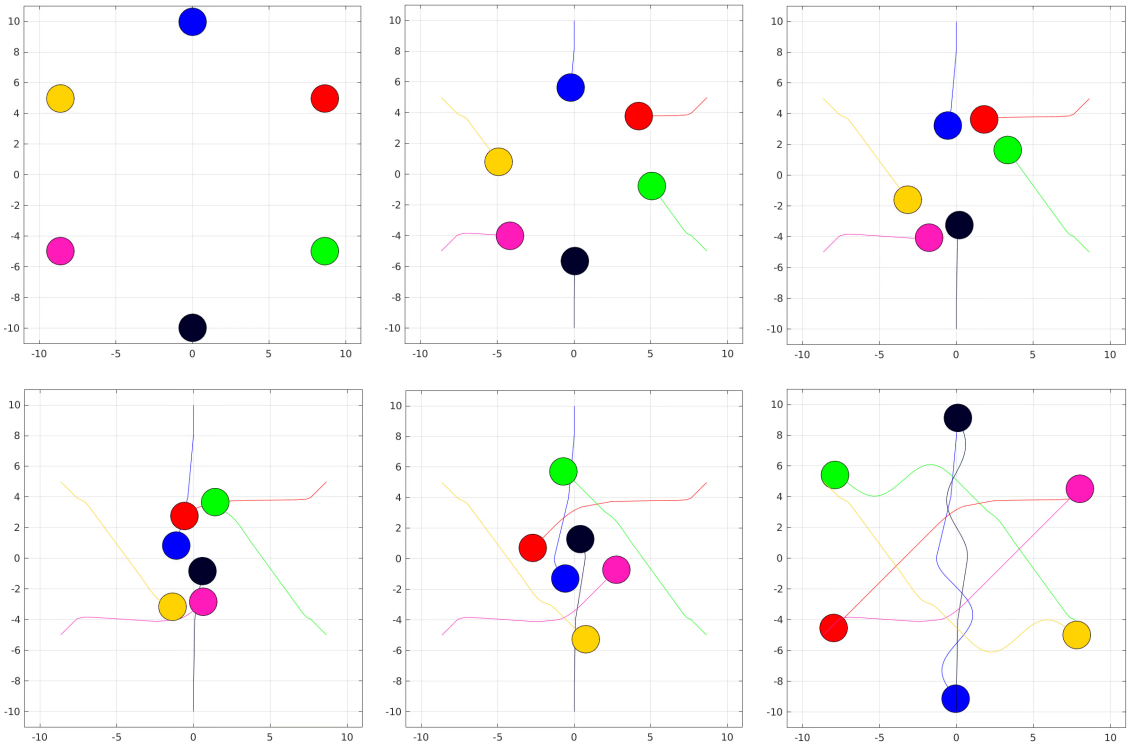


Figure 2.5 The figure shows the trajectories taken by six agents that independently execute the proposed algorithm.

Next, we increase the number of agents with a similar configuration to validate the scaling of IVO to more number of agents. Figure 2.6 presents the scenario with 10 agents. The computational time increases with the increase in the number of agents and for this scenario it is around 20ms for each cycle and has the update rates close to 50Hz. Even though the computational time is increasing with the increase in the number of agents, the update rates are high enough for aiding in a easy real time implementation. It is possible to get much better performance from a C++ based implementation of the same algorithm.

Additional simulation results are available at [3].

2.5.3 Real time Experiments

In this section, we evaluate the performance of Inverse Velocity Obstacles in real-time. For this, we use the Parrot Bebop2 quadrotor which accepts the yaw, pitch, roll angles as the control input. We developed a PID controller on top of the on-board controller to track the desired velocity. This is done as our algorithm operates in velocity control space. This lets us pass velocity as a control command to the drone. We used April Tags [28] of the family Tag36h11 for better state estimation of the other participants in the environment. We have completely bypassed the self-state estimation module as our

[3] https://sites.google.com/view/inverse-velocity-obstacle

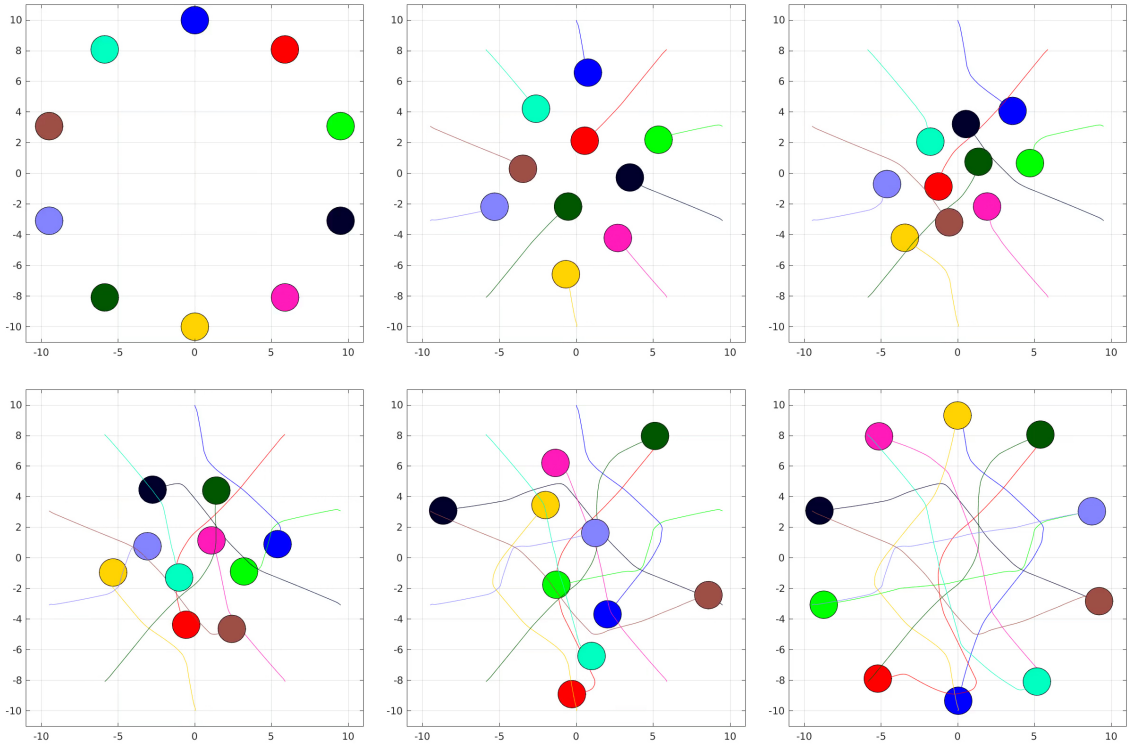


Figure 2.6 The figure shows the collision avoidance maneuvers adopted by ten different agents. These ten agents execute the proposed algorithm to obtain the controls.

framework does not need the agent's self-state for collision detection and avoidance. Figure 2.7 shows the snapshots of the real-time implementation of the proposed method on a quadrotor in a dynamic unknown environment.

2.5.4 Comparisons with Velocity Obstacle for Collision Detection

In this section, we compare the presented approach with Velocity Obstacle and show that the collision detection for IVO is more reliable compared to the traditional Velocity Obstacle in the presence of sensor noise. We consider noise in the position and velocities of both agents and obstacles. Here, we are essentially considering the state variables, $\mathbf{x}_t$ and $(\dot{\mathbf{x}}_t$, as random variables following some Gaussian distribution using Monte-Carlo based sampling methods. For every combination of the noise samples, we compute the collision avoidance constraint described in equation (2.2) for IVO and VO. We then plot the density functions in both the cases and analyze the nature of the resulting density functions.

The advantage with IVO is that the random variables need not depend on $\mathbf{x}_t$. In figure 2.8, we compare the probability distributions of the error in collision avoidance constraint for velocity obstacle as well as inverse velocity obstacle. The error function is defined in the equation (2.4).

$$e = f_t^j({}_r^o\mathbf{x}_t^j, {}_r^o\dot{\mathbf{x}}_t^j, \mathbf{u}_t) - f_t^j({}_r^o\mathbf{w}_t^j, {}_r^o\dot{\mathbf{w}}_t^j, \mathbf{z}_t) \tag{2.4}$$

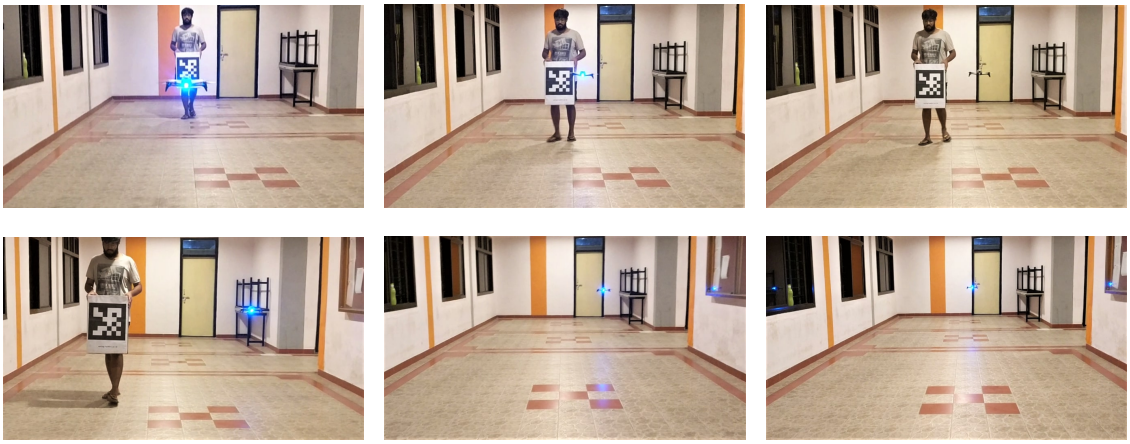
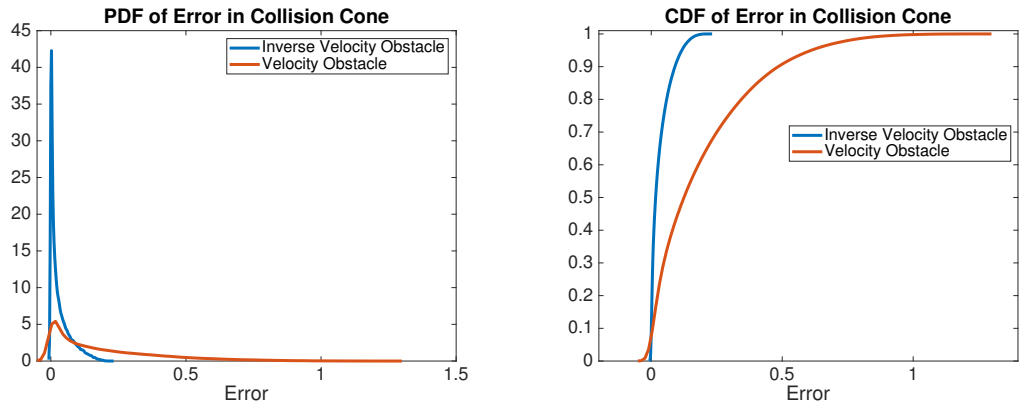


Figure 2.7 Real-time implementation with one dynamic obstacle. We added a safety margin to avoid collisions with the quadrotor. The quadrotor avoids the person (obstacle) by deviating from its original trajectory and merges back once the obstacle is avoided.

Here, f_t^j is the inverse velocity obstacle, ${}^o_r\mathbf{x}_t^j$ and ${}^o_r\dot{\mathbf{x}}_t^j$ are the true values of the relative position and velocity vectors of the obstacle respectively while ${}^o_r\mathbf{w}_t^j$ and ${}^o_r\dot{\mathbf{w}}_t^j$ are the noisy observations of the same. $\mathbf{u}_t$ is the ideal control given to the robot while $\mathbf{z}_t$ is the control that is executed in reality.

The noise in agent and obstacle states were assumed to be Gaussian distributions with zero mean. The distributions clearly show a reduction in the noise. The 99% confidence region for the inverse velocity obstacle is between 0 and 0.14 error range while it is between -0.03 and 0.56 error range for velocity obstacle. This provides a better scope for dealing with the noise just by increasing the radius of the obstacle.



(a) (b)

Figure 2.8 Distributions for error in collision cone from equation (2.4)

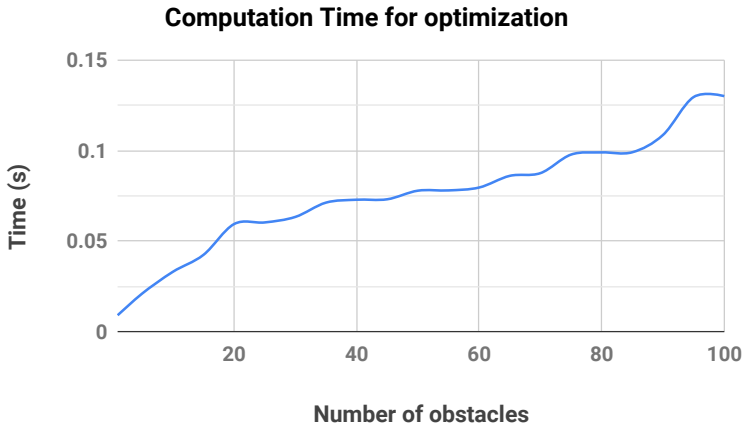


Figure 2.9 Computational time for different number of obstacles

2.6 Conclusion

In this chapter, we presented a new concept called Inverse Velocity Obstacles, for the safe navigation of autonomous agents in dynamic environments. In contrast to the previous works, we developed an ego-centric framework that eliminates the reliance on the robot's state for collision detection. This also decreases the computational complexity improving real-time implementation. The formulation presented is a natural extension of Velocity Obstacle and is easy to implement. We also extended this to a multi-agent navigation scenario and we show its efficacy to generate natural paths for systems as high as 50 agents in a very tight environment.

Chapter 3

Probabilistic Inverse Velocity Obstacle

3.1 Contributions

- We show that IVO is amenable to a probabilistic setting as we cast it into a chance constrained formulation that solves for control actions accounting for and respecting chance constraints.
- We show how bypassing the need to estimate the drones ego state results in significant decrease in trajectory lengths without compromising on the safety.
- We show ablation studies that portray the above advantages vis a vis chance constrained formulations that take into account state and velocity noise of the drone.
- Further, we bypass the need to model uncertainty in a parametric setting even as we model the observation/measurement uncertainties non parametric and show the efficacy of our formulation in non parametric setting.

3.2 Preliminaries

In the previous chapter, we presented a deterministic collision avoidance and navigation framework using the concept of IVO. In this chapter, we extend it to a probabilistic setting where we give a probabilistic guarantee or a lower bound on the risk involved during the collision avoidance in a dynamic uncertain environment. We pose the problem as a chance-constrained optimization and present a tractable and computationally efficient solution to the problem. The computational complexity of the presented method is on par with that of the deterministic collision avoidance problem from chapter 1.

3.2.1 Inverse Velocity Obstacle

Refer chapter 2.3.

3.2.2 Obstacle State Estimation

We assume that we have access to the state estimates ${}^{o}_{r}\mathbf{x}^{j}_{t}$ of the obstacles via a sensor like a camera or a LiDAR. The sensors are prone to noise and hence ${}^{o}_{r}\mathbf{x}^{j}_{t}$ is a random variable described by an unknown non-parametric probability distribution. We estimate ${}^{o}_{r}\dot{\mathbf{x}}^{j}_{t} = D({}^{o}\mathbf{x}^{j}_{t})$ using another particle filter to track the obstacles. Refer equation (2.1) for $D(\cdot)$.

3.2.3 Collision avoidance under uncertainty

In section 2.5.4, we showed how the error function behaves in presence of noise for IVO and VO. Though IVO improves the collision detection by a reasonable factor, it is not completely foolproof. We formulate the collision avoidance constraints in a way that will allow us to tune the risk involved in the avoidance maneuvers. According to the state estimations described in section 3.2.2, the position, and velocity of the obstacles are random variables described by unknown probability distributions. A typical approach would be to obtain a constraint for every possible value of the combination of random variables via Monte-Carlo sampling and try to make sure that all the constraints are satisfied. This will ensure that the robot always avoids all the noise samples of the obstacle. But this will not give us any control over the risk involved in the maneuvers. To this end, we can intuitively say that we want only η portion of the constraints to be satisfied instead of satisfying all the constraints. This will give a lower bound on the risk involved. This motivates us to pose the collision avoidance constraints under uncertainty as chance constraints.

3.2.4 Chance Constraints

We have a function of random variables $f^{t}_{j}(\cdot)$.

$$f^{j}_{t}(\cdot) = \frac{(\mathbf{r}^{T}_{j}\mathbf{v}_{j})^{2}}{\|\mathbf{v}_{j}\|^{2}} - \|\mathbf{r}_{j}\|^{2} + ({}^{r}r + {}^{o}r^{j})^{2} \leq 0, \forall j \tag{3.1a}$$

$$\mathbf{r}_{j} = {}^{o}_{r}\mathbf{x}^{j}_{t}, \mathbf{v}_{j} = {}^{o}_{r}\dot{\mathbf{x}}^{j}_{t} \tag{3.1b}$$

where, ${}^{o}_{r}\mathbf{x}^{j}_{t}$ and ${}^{o}_{r}\dot{\mathbf{x}}^{j}_{t}$ are random variables. The deterministic collision avoidance constraint is given by $f^{j}_{t}(\cdot) \leq 0$. For a probabilistic case with n samples of ${}^{o}_{r}\mathbf{x}^{j}_{t}$ and m samples of ${}^{o}_{r}\dot{\mathbf{x}}^{j}_{t}$, we will have $n \times m$ samples of $f^{j}_{t}(\cdot)$. We want at least η portion of the values of $f^{j}_{t}(\cdot)$ to be less than zero. Hence, the collision avoidance constraints should be satisfied in a probabilistic manner given by (3.2).

$$\Pr(f^{j}_{t}(\cdot) \leq 0) > \eta, \forall j \tag{3.2}$$

In simple terms, a control $\mathbf{u}_{t}$ is required that moves a considerable mass of the distribution to the left of 0. Figure 3.1 shows how the shape of the distribution can be manipulated by $\mathbf{u}_{t}$ to obtain a desired distributional shift. Note that the constraint $f(\cdot)$ defined above is same as $\mathcal{C}$ in the figure 3.1. Here, η is the minimum confidence with which we want to avoid the collision.

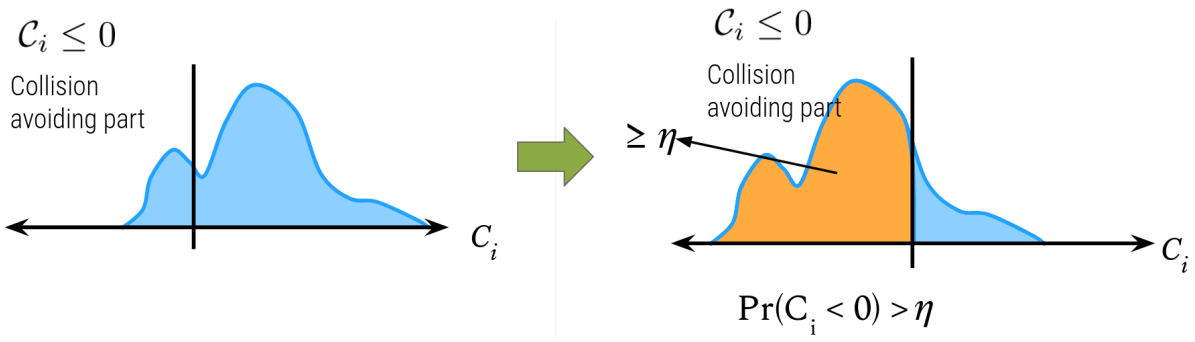


Figure 3.1 The figure shows the effect of a chance constrained optimization. $\mathcal{C}_i \leq 0$ is the deterministic constraint and η represents the percentage of the times the deterministic constraint is satisfied which is also proportional to the area under the density function from $-\inf$ to 0. Here, $\mathcal{C}$ is same as $f(\cdot)$ defined in 3.2.

Though conceptually simple, chance constraints of this form are often computationally intractable. We present a tractable solution for chance constraints in section 3.3.

3.2.5 Uncertainty assumptions

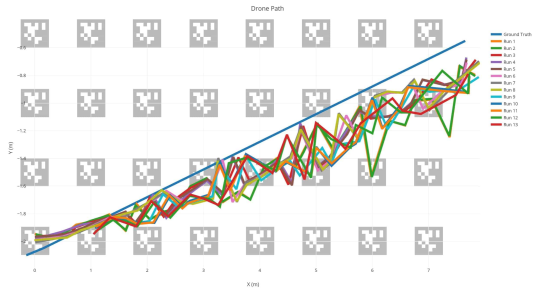
We try to take all uncertainties possible in the formulation. Specifically, in the case of IVO, collision detection and avoidance is invariant to the uncertainty in the robot's position and velocity. So, we do not consider the uncertainty in state estimates in our formulation. From equations (2.2) and (2.1), it is evident that the collision detection relies on obstacles' position which is prone to sensor noise and the new relative velocity of the obstacle which is effected by the control noise. We assume that the aforementioned uncertainties come from noise models following non-parametric probability distribution. For the sake of simplicity, we assume that the noise models are independent and their means are zero centered.

3.2.6 Modelling the noise

3.2.6.1 Data collection

Here, we are looking for positional uncertainties in the drone's odometry and obstacle detection. We are using Bebop 2 for the data collection. The position estimation for the drone depends on the feedback from IMU and also the optical flow retrieved from the downward-facing camera. IMU data is prone to the magnetic interference of the motors and the optical flow highly depends on the ambient light. We assume that the light is constant (made constant using external light sources). For obtaining the ground truth, we arranged April Tags of the family Tag36h11 for every 0.5 meters in Y-direction and 1 meter in X-direction to form an 8x3 grid as shown in the figure 3.2(a). Along with these markers, there are three more relatively larger markers to ensure that they get detected throughout the flight. We start the drone from a known position with respect to the grid that we formed using markers and record

(a) Experimental setup with Apriltags arranged as a grid for ground truth.



(b) Trajectories obtained for various runs.

Figure 3.2 Data collection setup for modelling the sensor noise.

the drone's odometry along with the tag detections from the drone's front-facing camera. Since it is difficult to make the drone fly in the same way for every run or record the data at specific locations, we bin the ground truth data that we recorded for every 0.01 meters. This means, all the corresponding odometry data for the range $x - 0.5$ to $x + 0.5$ meters will be considered as odometry data for x meters. This completes our positional ground truth vs positional odometry data. We have also recorded the timestamps for tag detections and drone's odometry data from which we can compute the ground truth velocities and accelerations, along with corresponding odometry values. The same binning procedure is repeated for velocity and acceleration data as well. This gives us the uncertainty in the drone's position, velocity, and acceleration. Obstacles position estimation depends both on the accuracy of the camera calibration and the drone's the then position (since we need to add drone's position to get the position of the obstacle in the global frame). Obstacles in our case are markers themselves. We choose one marker and exclude it from the ground truth calculations and treat this marker as an obstacle. Now, we compute the position of this obstacle with respect the the global frame by adding it's tag detection value and the drone's position. The tag is stationary in the global frame and we already know it's position (measured using tape). Now we apply a linear transformation considering the tag to be in motion relative to the drone (here, drone is stationary with respect to itself) and repeat the data extraction process mentioned above for position and velocity of the obstacle.

3.2.6.2 Generating noise samples

Once we have the error samples from the data collection step, we calculate the mean, variance, skewness, and kurtosis for the positions, velocities of the robot, and obstacle. We also repeat this step for getting the moments for the state estimation errors in the ego-frame. Then we use a Pearson random system of numbers to generate samples that satisfy the computed moments. These moments act as a

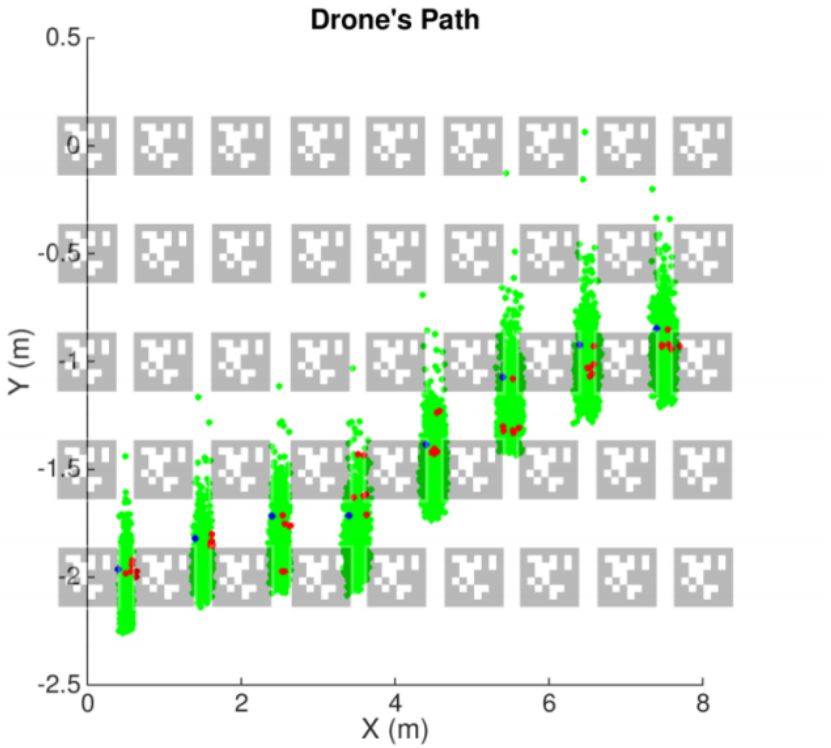


Figure 3.3 This figure shows the samples generated from the modeled parent distribution. Green position samples are from the parent distribution while the red ones are the actual samples. The blue samples represent the ground truth.

parent distribution of the actual noise distribution. The samples generated using this method are shown in figure 3.3.

3.3 Probabilistic IVO

3.3.1 Solving the collision chance constraint

The chance constraints of the form described in equation (3.2) are typically intractable. To address this issue, we can replace the original chance constraints with solvable surrogate constraints.

According to Cantelli's inequality,

$$\Pr(\mathrm{x} - \hat{x} \geq \beta) \quad \begin{cases} \leq \frac{\sigma^2}{\sigma^2+\beta^2} & \text{if } \beta > 0 \\ \geq 1 - \frac{\sigma^2}{\sigma^2+\beta^2} & \text{if } \beta < 0 \end{cases} \tag{3.3}$$

where, $\hat{x}$ and σ are the mean and standard deviation of the random variable x, and $\beta \in \mathbb{R}$.

In our case, we can re-write the inequality as

$$\Pr(f_t^j(\cdot) - \hat{f}_t^j(\cdot) \geq \lambda \sigma_{f_t^j(\cdot)}) \quad \begin{cases} \leq \frac{1}{1+\lambda^2} & \text{if } \lambda > 0 \\ \geq 1 - \frac{1}{1+\lambda^2} & \text{if } \lambda < 0 \end{cases} \tag{3.4a}$$

$$\hat{f}_t^j(\cdot) = \frac{1}{n \times m} \sum_{i=0}^{n} \sum_{j=0}^{m} f_t^j(p_i, q_j, \cdot) \tag{3.4b}$$

$$\sigma_{f_t^j(\cdot)} = \sqrt{\frac{1}{n \times m} \sum_{i=0}^{n} \sum_{j=0}^{m} (f_t^j(p_i, q_j, \cdot) - \hat{f}_t^j(\cdot))^2} \tag{3.4c}$$

Where, p_i are the uncertain samples of the observations, ${}^{o}_{r}\mathbf{x}_t^j$. q_j are the uncertain samples of the relative velocity of the obstacle as a result of actuation noise, $q_j = {}^{o}_{r}\dot{\mathbf{x}}_{t-1}^j + \mathbf{u}_t$ and $\mathbf{u}_t$ is the change in velocity control given to the robot at time t and acts as a random variable.

Using equation (3.4a), the chance constraint in equation (3.2) can be written as,

$$\Pr(f_t^j(\cdot) \leq 0 \mid \hat{f}_t^j(\cdot) + \lambda \sigma_{f_t^j(\cdot)} \leq 0) \geq \frac{\lambda^2}{1+\lambda^2} \tag{3.5}$$

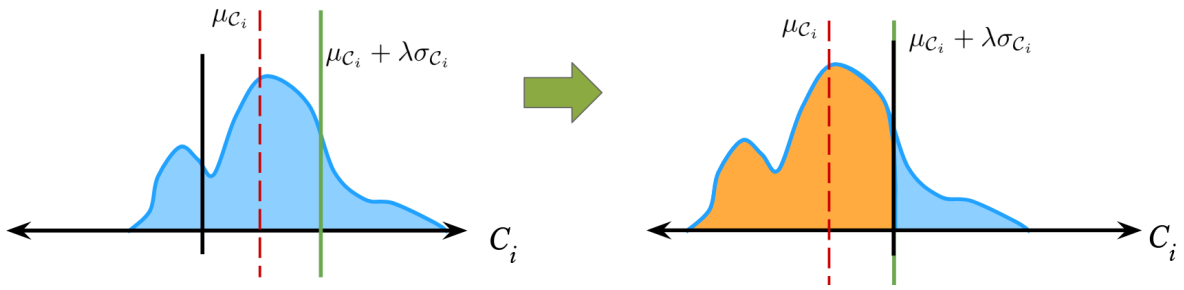


Figure 3.4 The left figure shows the state of the distribution before the chance-constrained optimization and the right figure shows the state of distribution after the optimization. The blue shaded region represents the area under the PDF of a constraint $\mathcal{C}_i$. The orange shaded region represents the mass of the distribution that satisfies the constraint. The red dotted line represents the sample mean of the constraint, $\mu_{\mathcal{C}_i}$. $\sigma_{\mathcal{C}_i}$ is the standard deviation of $\mathcal{C}_i$. If the surrogate constraint from equation (3.5) is satisfied, the $\mu + \lambda\sigma$ line will minimally touch the $\mathcal{C}_i = 0$ line.

Equation (3.5) basically says that if $\hat{f}_t^j(\cdot) + \lambda \sigma_{f_t^j(\cdot)} \leq 0$ is satisfied, then $\Pr(f_t^j(\cdot) \leq 0) > \eta$ is also satisfied where $\eta = \frac{\lambda^2}{1+\lambda^2}$ in this case. Intuitively speaking, this is equivalent to shifting the mean, $\hat{f}_t^j(\cdot)$, and a certain margin from the mean, $\lambda\sigma$, of the distribution to the left side of zero. This is graphically represented in the figure 3.4.

This method is adopted from [14] and comparisons with other methods that deal with chance constraints can be found in the same. Unlike [14], we do not formulate the solution in a closed form. We

make no assumption on the nature of the distributions and consider sample mean and standard deviation. We later show that using sample level data can still lead to reasonable collision avoidance under uncertainty in our experiments. Hence this can be considered a more generalized form.

3.4 Navigation Under Uncertainty

3.4.1 Path Optimization

Consider a robot at ${}^{r}\mathbf{x}_t$ moving with a velocity ${}^{r}\dot{\mathbf{x}}_t$ towards a goal, $\mathbf{g}$. Since the setting of IVO happens to be in an ego-centric frame, let us define a relative goal ${}_{r}\mathbf{g}$ given by

$$ {}_{r}\mathbf{g}_t = \mathbf{g} -^{r} \mathbf{x}_t $$

In a deterministic case, the new velocity that needs to be taken to reach the goal is given by equation (2.3). This can be rewritten as shown in the equation (3.6a).

$$ \arg\min_{\mathbf{u}_t} J(\mathbf{u}_t) = \left\| {}^{d}\dot{\mathbf{x}}_t + {}_{r}\dot{\mathbf{g}}_t - \mathbf{u}_t \right\| \tag{3.6a} $$

$$ {}^{d}\dot{\mathbf{x}}_t = \frac{{}_{r}\mathbf{g}_t}{\|{}_{r}\mathbf{g}_t\|}\mathbf{v}_{max} \tag{3.6b} $$

$$ {}_{r}\dot{\mathbf{g}}_t = \frac{{}_{r}\mathbf{g}_t - {}_{r}\mathbf{g}_{t-\tau}}{\tau} \tag{3.6c} $$

Where, ${}^{d}\dot{\mathbf{x}}_t$ is a desired velocity which takes the robot towards the goal, $\mathbf{u}_t$ is the change in velocity control given to the robot and ${}_{r}\dot{\mathbf{g}}_t$ is the velocity estimate of the goal with respect to the robot. Let us take a case where the goal is stationary in the global frame. Then the velocity of the robot in the global frame is given by ${}^{r}\dot{\mathbf{x}}_t = -{}_{r}\dot{\mathbf{g}}_t$. So, for this case, the equation (3.6a) boils down to

$$ \arg\min_{\mathbf{u}_t} J(\mathbf{u}_t) = \left\| {}^{d}\dot{\mathbf{x}}_t - {}^{r}\dot{\mathbf{x}}_{t-\tau} - \mathbf{u}_t \right\| $$

Where ${}^{r}\dot{\mathbf{x}}_t$ is the velocity of the robot in the global frame. The relative positions of the goal in the robot's frame can be obtained from a sensor. Since the sensor data can be prone to noise, we alter the cost in equation (3.6a) a little bit to make it robust even in the presence of noise. We perform Monte-Carlo sampling on the modelled parent distribution of sensor noise and define a new cost function based on the sum of squared error between the relative velocity of the goal as seen from the robot, ${}_{r}\dot{\mathbf{g}}_t^i$ (for i^{th} sample), and ${}^{d}\dot{\mathbf{x}}_t$. The path optimization cost for probabilistic case is given by equation (3.7)

$$ \arg\min_{\mathbf{u}_t} J(\mathbf{u}_t) = \sum_{j=1}^{M}\sum_{i=1}^{N} \left\| {}^{d}\dot{\mathbf{x}}_t + {}_{r}\dot{\mathbf{g}}_{t-\tau}^{i} - \mathbf{u}_t^j \right\| \tag{3.7} $$

Where, ${}_{r}\dot{\mathbf{g}}_t^i$ and $\mathbf{u}_t^j$ are the i^{th} sample of relative velocity of the goal and j^{th} sample of control obtained through Monte-Carlo sampling from their respective noise models.

An alternate approach is to obtain an approximate value of ${}_r\dot{\mathbf{g}}_t$ and $\mathbf{u}_t$ by taking their respective sample means. In this case, the goal-reaching cost is defined using the equation (3.6a) replacing the random variables with their sample means.

3.4.2 Navigation framework

The navigation of the robot can be now posed as an optimization problem. Consider the following optimization problem with variable $\mathbf{u}_t$ which denotes the controls given to the robot at time step t. The goal in robot's frame at time t is given by ${}_r\mathbf{g}_t$ and the maximum attainable velocity of the robot is given by v_{max}.

$$\arg\min_{\mathbf{u}_t} J(\mathbf{u}_t) = \left\| {}^d\dot{\mathbf{x}}_t + {}_r\hat{\dot{\mathbf{g}}}_t - \hat{\mathbf{u}}_t \right\| \tag{3.8a}$$

$$\text{subject to} \quad \hat{f}_t^j(\cdot) + \lambda\sigma_{f_t^j(\cdot)} \leq 0, \;\; \forall j = 1, \ldots, N \tag{3.8b}$$

The terms ${}_r\hat{\dot{\mathbf{g}}}_t$ and $\hat{\mathbf{u}}_t$ in (3.8a) are the sample means of relative goal velocity in the robot's frame and controls given to the robot respectively. ${}^d\dot{\mathbf{x}}_t$ is obtained using equation (3.6b). N is the number of noise samples obtained using Monte-Carlo method. We obtain better sample complexity as the value of N increases. The constraints given by equation (3.8b) make sure that $\Pr(f_t^j(\mathbf{u}_t^i)) \geq \frac{\lambda^2}{1+\lambda^2} \; \forall i = 1, \ldots, N$ are satisfied. The larger the value of λ is, the higher is the lower bound on confidence of collision avoidance.

3.5 Results

We implemented the framework in Matlab and the codes are available at [1].

3.5.1 Navigation Under Uncertainty

We first compare the trajectories taken by the robot using the deterministic constraints and the probabilistic constraints. Figure 3.5 shows the trajectories for both the cases with the same start and goal configuration. For the deterministic case, the robot overlaps with the noise samples. Since these noise samples are one of the positions the obstacle could have been, this is the same as a possible collision in a real-life scenario. On the other hand, for the probabilistic case, the robot avoids almost all of the noise samples. The percentage of avoidance of the samples can be controlled using λ.

Figure 3.6 shows the avoidance maneuvers for different values of λ. We experimentally show that an increase in the value of λ leads to higher confidence in the avoidance. This can be seen as the number of samples of the obstacle decrease as the value of λ increases. It should also be noted that, increasing λ will not only lead to higher confidence in collision avoidance but also more conservative trajectories.

[1] https://github.com/RoboticsIIITH/probabilistic-inverse-velocity-obstacle

Figure 3.7 shows the complete trajectory obtained using the proposed method with two obstacles. The moving obstacles are shown in a shade of orange with their uncertainty samples giving a noisy visualization. The robot is shown in blue. We show colliding velocity samples of the robot in various shades of red and green. More red vectors imply a large number of velocity samples colliding with a large number of obstacle samples. More green represents a very small fraction, possibly even zero, samples of velocities colliding with very small numbers of obstacle samples. It is to be noted we only compute the change in velocity as the control and not the absolute velocity of the robot. It is purely for illustrative purposes we show velocity samples that are in a collision. Figure 3.7(a) depicts the situation when the robot has just avoided the obstacle coming from the right and only 34.3% of velocity samples are in collision at that moment, all of which eventually come out as the robot keeps moving towards the left.

This soon turns to deep red in figure 3.7(b) wherein all velocity samples collide with most obstacle samples. Due to the computed controls this situation progressively improves from figures 3.7(c)-3.7(e) as green shades progressively dominate. Eventually no velocity in collision in figure 3.7(h). The robot then safely navigates through the two obstacles ensuring no collision to finally reach the goal in figure 3.7(i) through our chance-constrained formulation for handling uncertainty.

3.5.2 IVO vs VO

In this section, we compare the performance of IVO vs VO. We first compare how they both work in the presence of robot state noise in a deterministic setting. By deterministic setting, we mean that we do not invoke the chance-constrained formulation to compute the robot controls but they are computed through the deterministic IVO formulation. In subsection (3.5.2.1) we show while robot state noise alone is present (no observation noise) IVO is superior as it does not need any estimates of its state for computing the avoidance maneuver. Then in subsection (3.5.2.2) we compare PIVO vs PVO in a probabilistic framework where robot state noise, obstacle perception noise, and control noise is taken into account.

3.5.2.1 In presence of Robot State Noise

The uncertainty due to robot state noise is characterized through a Gaussian distribution with a standard deviation of 0.05 meters with a mean as zero. This uncertainty propagates to the state of the moving obstacle mediated by the measurement. Eventually, this propagates to the velocity of the moving obstacle when computed as differentiation of states in the classical VO formulation. This uncertainty leads to collisions and longer trajectories of the VO agent. Whereas the IVO agent computes the relative velocity of the obstacle with respect to the goal directly from observations bypassing the need to estimate the robot state. This results in collision-free trajectories. The VO agent apart from colliding takes a longer path of 25.6859 meters in 29.1 seconds while the IVO agent takes a collision-free maneuver

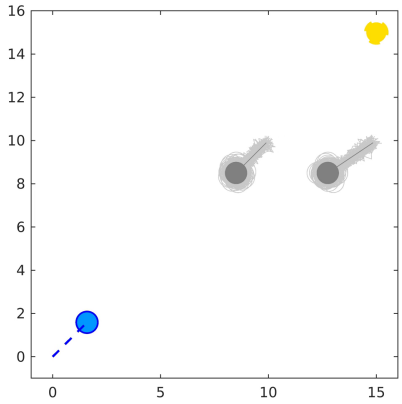
towards the goal covering a shorter distance of 22.4833 meters in 23.1 seconds. Figure 3.8 shows the different trajectories taken in both cases.

3.5.2.2 Trajectory Comparison for PIVO vs PVO

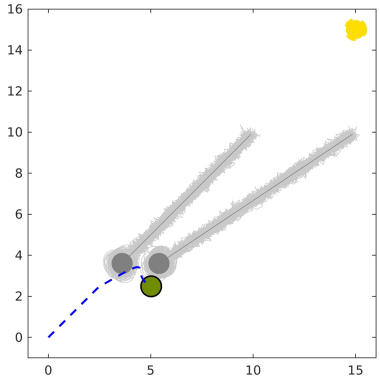
Here we consider a probabilistic framework where uncertainty related to robot state, perception, and control is taken. The figure 3.9 shows the comparison of the trajectories taken by PIVO and PVO for different configurations 3.9(a)-3.9(c) whereas the table 3.1 quantifies how well the PIVO performs in comparison to PVO for those cases. The PIVO trajectories are shown with orange solid lines while the PVO ones are shown with blue dashed lines. It is to be noted that the PIVO trajectories are less deviant because they are handling lower values of variance in their chance constraints due to their ability to not account for robot states which are noisy to produce collision avoidance maneuver. The PIVO trajectories take a smaller amount of time to reach the goal while covering a smaller distance as well. For the cases taken here, there is an average reduction of 10.73% and 14.06% in the distance traveled and time taken for PIVO in comparison to PVO.

3.6 Conclusion

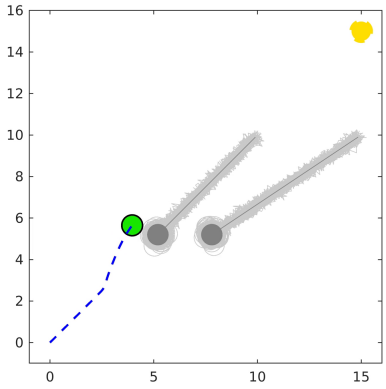
In this chapter, we presented PIVO, an algorithmic framework to handle the uncertain dynamic environment shared by humans using the concept of chance constraints. The presented framework can handle any non-parametric uncertainty and doesn't approximate it to any parametric distribution. Being an ego-centric framework it removes the dependency on the state estimation techniques for inferring the self-state thereby reducing the computational complexity and aiding for real-time implementation. We have evaluated our framework through numerical simulations in different conditions. We show performance gain of PIVO over PVO in terms of trajectory deviation and time while continuing to maintain safety. Our future work includes extending the framework for non-holonomic systems.



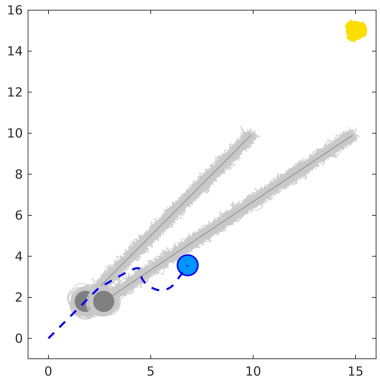
(a) Initial position for both probabilistic and deterministic cases.



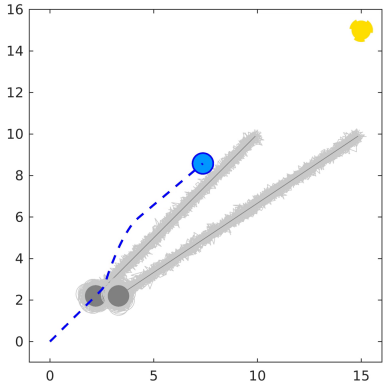
(b) Avoidance maneuver taken due to the deterministic constraints. It is evident that there is a huge overlap of the noise samples of the obstacle and the robot.



(c) Avoidance maneuver taken due to the probabilistic constraints. The percentage of avoidance, in this case, is much higher compared to the deterministic case.

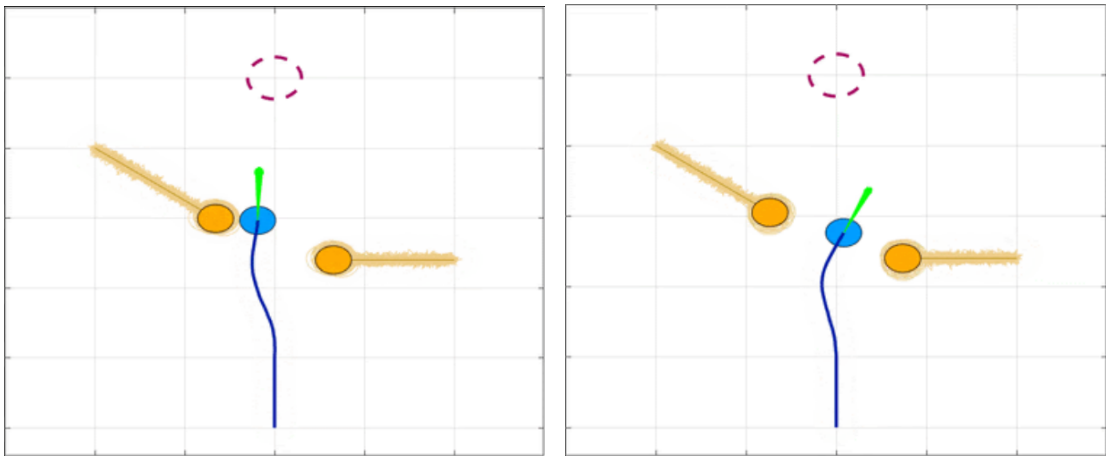


(d) Trajectory with deterministic constraints.



(e) Trajectory probabilistic constraints.

Figure 3.5 The blue circle represents the robot while the gray circles represent the obstacles. The translucent gray patches around the obstacles are the noise samples. The yellow patch at the top right corner is the goal position. The color of the agent turns green if avoidance has started. Darker green indicates lower confidence of avoidance.



(a) $\lambda = 0.7$ (b) $\lambda = 1.5$

Figure 3.6 Comparisons with different values of λ

Table 3.1 Trajectory Comparison for λ = 1.2

Configuration	**Distance travelled (m)**		**Time taken (s)**	
	PIVO	PVO	PIVO	PVO
Figure 3.9(a)	22.6365	24.8578	13.2	14.6
Figure 3.9(b)	22.0204	25.1388	11.9	14.1
Figure 3.9(c)	15.8772	17.8072	8.3	10.0

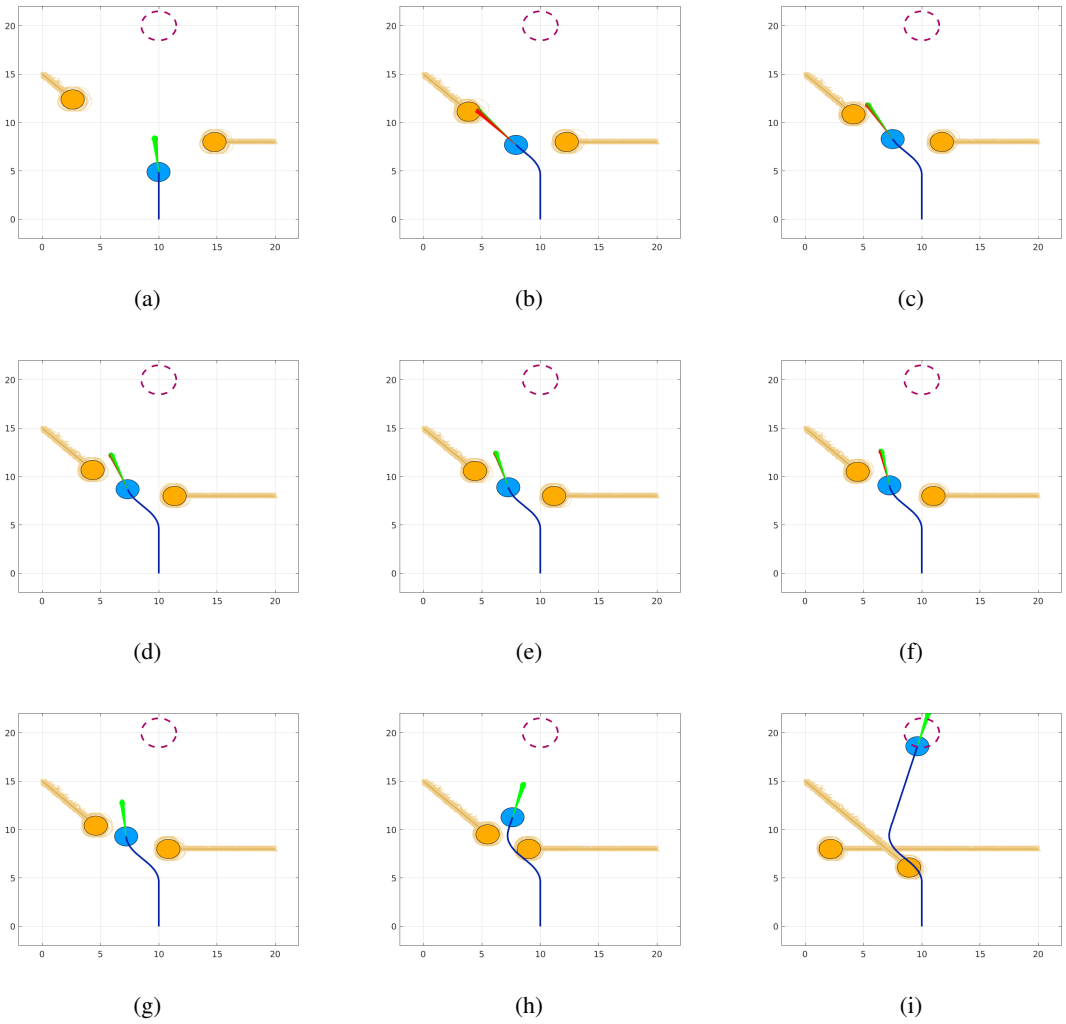


Figure 3.7 Collision avoidance sequence considering non-parametric uncertainty. The moving obstacles are shown in a shade of orange with their uncertainty samples while the robot is shown in blue. The velocity samples are shown in shades of green and red depending on how many velocity samples are in a collision with the obstacle samples. x and y axis in the plots denote position (m) in x and y direction.

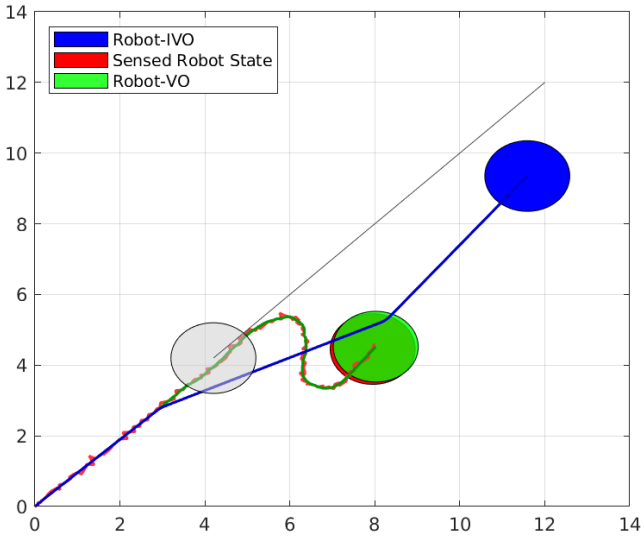


Figure 3.8 Deterministic collision avoidance comparison between IVO and VO in the presence of robot state noise. The figure demonstrates how IVO is invariant to the noise and manages to avoid a collision. x and y axis in the plots denote position (m) in x and y direction.

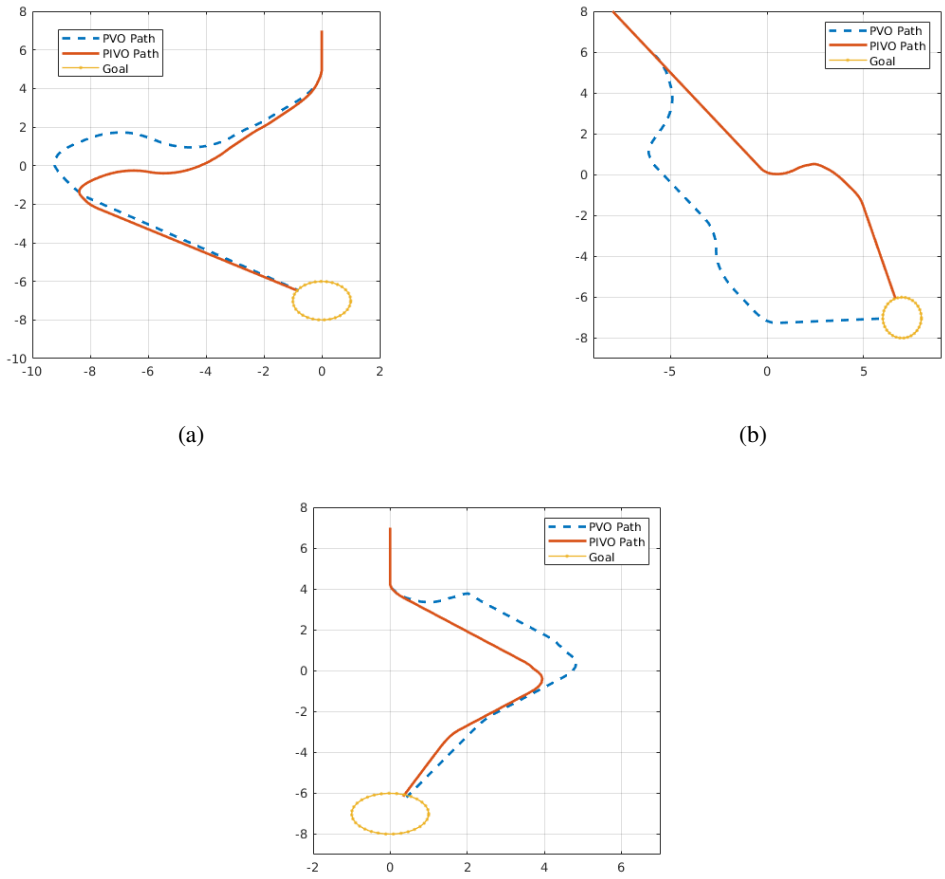


Figure 3.9 Trajectory Comparison of VO and IVO. The blue and orange paths are a result of the velocity obstacle while the red path is the one taken by the IVO agent. Obstacle trajectories are not shown in these figures for clarity and ease of viewing. x and y axis in the plots denote position (m) in x and y direction.

Chapter 4

Extending to other motion models

In the previous chapter, we presented a way to solve the chance constraints. In this chapter, we briefly touch upon possible extensions to the previously presented approach. We assumed a linear motion model so far for simplicity. We present the approaches to solve the collision avoidance problem for differential drive robots but it should be straight forward to extend these approaches to other motion models as well.

4.1 Preliminaries

4.1.1 Differential drive motion model

Consider a differential drive non-holonomic mobile robot as shown in figure 4.1. We assume that there is no slip in the lateral direction. The non-holonomic constraints for such system are given by equation (4.1).

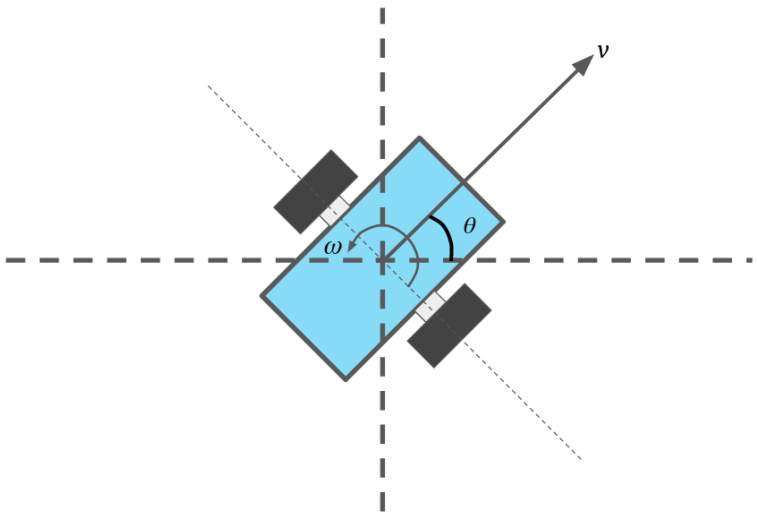


Figure 4.1 Differential drive wheeled robot

$$\begin{bmatrix} {}^r\dot{\mathbf{x}}_t \\ \dot{\theta}_t \end{bmatrix} = \begin{bmatrix} \cos\theta_t & 0 \\ \sin\theta_t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \tag{4.1}$$

where θ_t is the heading of the robot at time t. (v_t, ω_t) are the magnitude of the velocity and angular velocity of the robot at time t respectively.

4.2 Motion planning under uncertainty for differential drive robots

4.2.1 Linearizing the constraints

The idea here is simple. We rewrite the collision avoidance constraints presented in equation (2.2) as described in the equation (4.2).

$$f_t^j(\cdot) = \frac{(\mathbf{r}_j^T \mathbf{v}_j)^2}{\|\mathbf{v}_j\|^2} - \|\mathbf{r}_j\|^2 + ({}^r r + {}^o r^j)^2 \leq 0, \forall j \tag{4.2}$$

$$\mathbf{r}_j = {}^r\mathbf{x}_t - {}^o\mathbf{x}_t^j \tag{4.3}$$

$$\mathbf{v}_j = {}^r\dot{\mathbf{x}}_t - {}^o\dot{\mathbf{x}}_t^j \tag{4.4}$$

$$= \begin{bmatrix} v\cos(\theta_t + \omega_t\tau) \\ v\sin(\theta_t + \omega_t\tau) \end{bmatrix} - {}^o\dot{\mathbf{x}}_t^j \tag{4.5}$$

$$\tag{4.6}$$

where v_t and ω_t are the controls given to the robot and τ is the time horizon for which these instantaneous velocities are valid. Equation (4.2), when used as a constraint in the optimization, turns out to be non-linear and non-convex and can't be solved by most solvers that are available. To address this issue, we will use a first-order Taylor series approximation of equation (4.2) which will make it a linear constraint.

Figure 4.2 shows the trajectories obtained using linearization of the constraints. Then we compare the deterministic and probabilistic cases for a differential drive/unicycle motion model. Figure 4.3 shows the initial configuration for both the cases. Figure 4.4 compares the avoidance maneuvers adopted in both the cases. Figure 4.5 shows the final trajectories that are a result of collision avoidance in both the cases.

4.2.2 Tracking a linear motion model

While the linearization based approach gives decent results, we also explore an alternate way to deal with non-holonomic motion models. We assume that we have a tracking function $\zeta(\mathbf{p}_t, \mathbf{u}_t, \cdot)$ that takes the current state estimates and the control given to the robot. Here, $\mathbf{p}_t$ denotes the state estimates of the

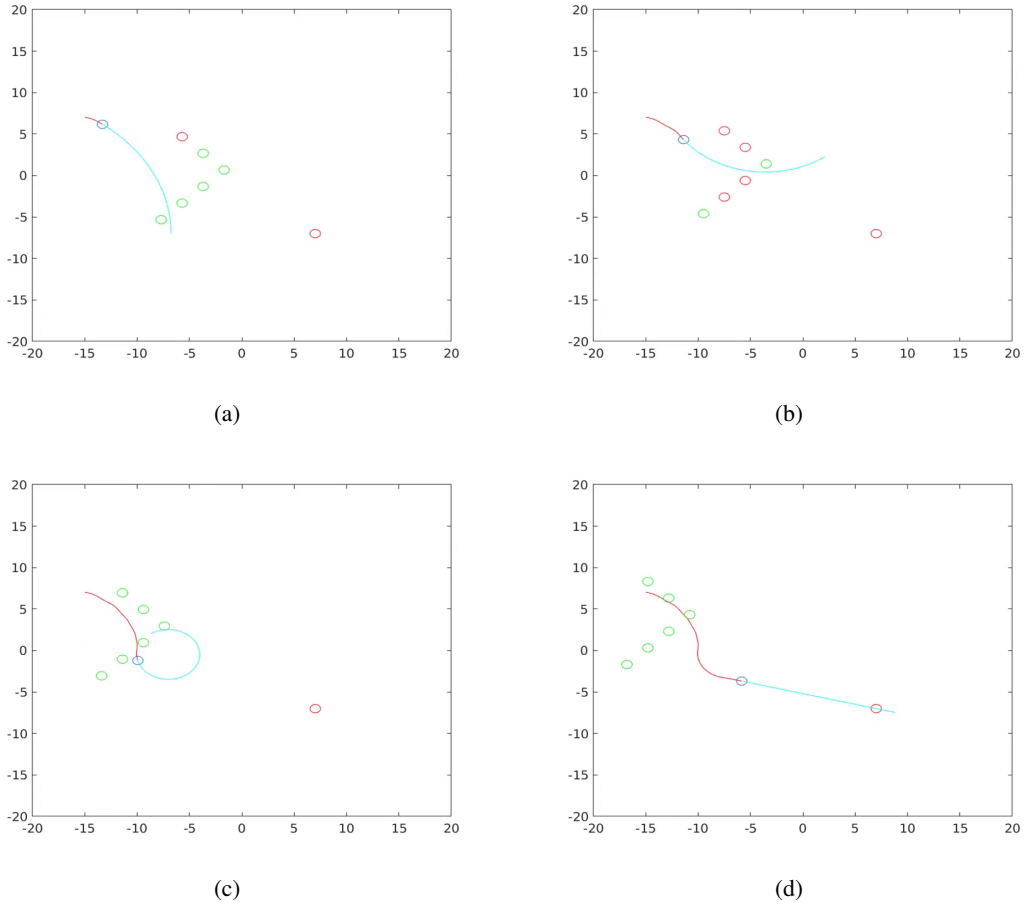


Figure 4.2 The red curve is the trajectory followed by the agent and the blue curve is the possible future trajectory obtained from optimization. The red circles represent the obstacles that are being considered for the collision while the green circles represent arbitrary obstacles in the environment.

robot. Every control value, $\mathbf{u}_t$ leads to a new trajectory at time t. In the navigation problem described in 3.4.2, we try to move towards a goal position while avoiding obstacles. In this case, we replace the desired velocity, ${}^{d}\mathbf{x}_t$, with a slightly different function. In the previous section, this desired velocity is defined as velocity with a magnitude equivalent to the maximum velocity attainable by the robot towards the goal position. Now, we define it as velocity along a feasible trajectory and this trajectory is kinematically constrained. The feasible trajectory is obtained using the tracking function, $\zeta(\cdot)$. A graphical representation of the outputs of the tracking function is shown in figure 4.6.

$$ {}^{d}\mathbf{x}_t = \zeta(\mathbf{p}_t, {}^{g}\mathbf{u}_t, \cdot) \tag{4.7} $$

where ${}^{g}\mathbf{u}_t$ represents the control that takes to the robot towards the desired goal assuming a linear motion model.

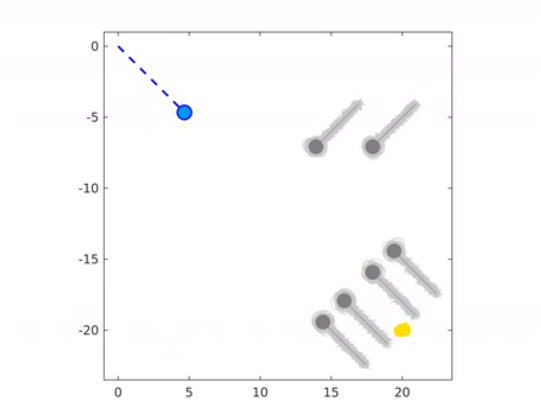
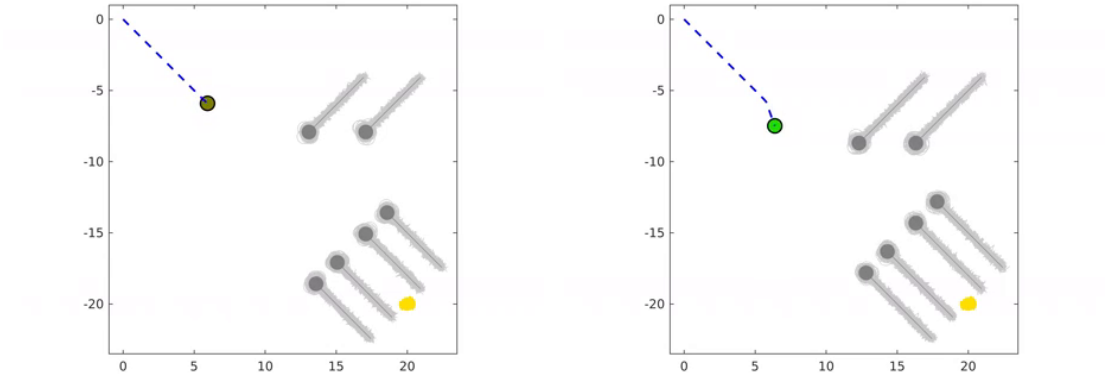


Figure 4.3 Initial configuration for comparing the probabilistic and deterministic non-holonomic scenarios. The blue circle represents the robot while the gray circles represent the obstacles. The translucent gray patches around the obstacles are the noise samples. The yellow patch at the top right corner is the goal position.

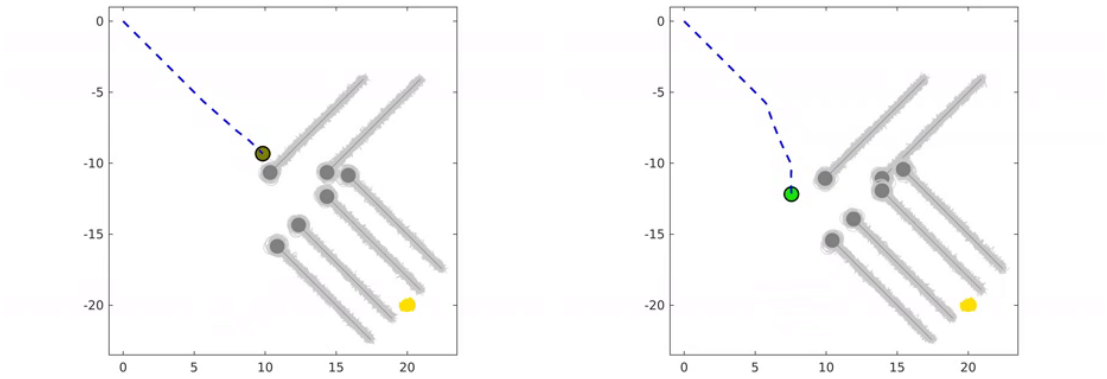
With the newly defined desired velocity, we obtain the controls from the optimization routine described in section 3.4.2 assuming a linear motion model. The controls obtained from this step may not be feasible to be executed bu the robot. So, we pass these controls to our tracking function and get new controls that are kinematically feasible. When we do this, the trajectory taken by the robot differs from the trajectory that the optimization controls were resulting in, and may the new trajectory may lead to collisions. To address this, we find a maximum tracking error, say β, and enlarge the radius of the robot with this value. So the deterministic collision avoidance constraint can be rewritten as shown in equation (4.8).

$$f_t^j(\cdot) \leq 0 : \frac{(\mathbf{r}_j^T \mathbf{v}_j)^2}{\|\mathbf{v}_j\|^2} - \|\mathbf{r}_j\|^2 + ({}^r r + {}^o r^j + \beta)^2 \leq 0, \forall j \tag{4.8}$$

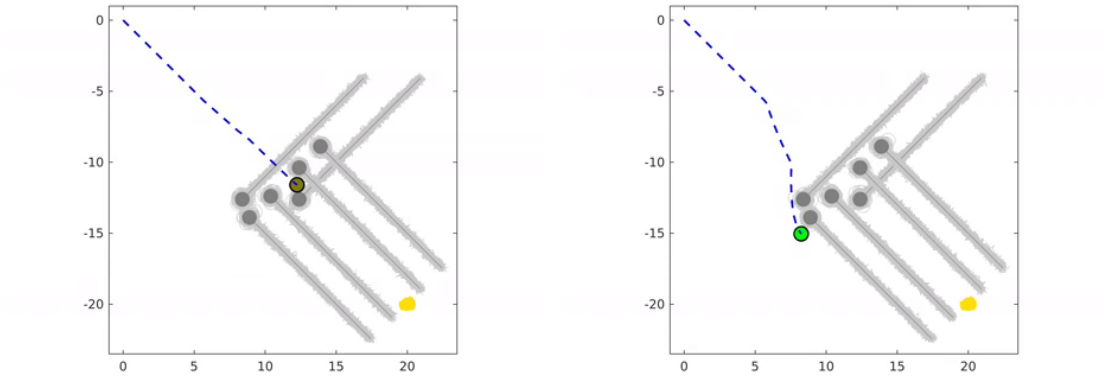
This will ensure that the robot does not collide due to the deviations from the trajectory.



(a) Deterministic (b) Probabilistic

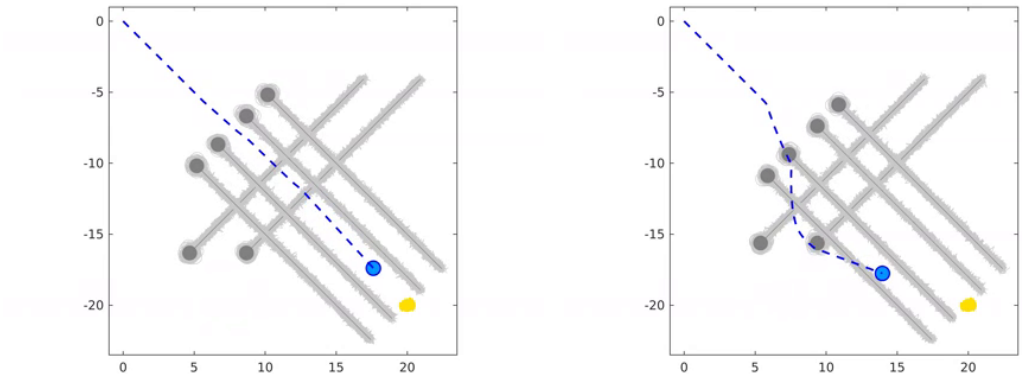


(c) Deterministic (d) Probabilistic



(e) Deterministic (f) Probabilistic

Figure 4.4 Avoidance maneuvers adopted by the probabilistic and deterministic cases. The blue circle represents the robot while the gray circles represent the obstacles. The translucent gray patches around the obstacles are the noise samples. The yellow patch at the top right corner is the goal position. The color of the agent turns green if avoidance has started. Red or darker green indicate lower confidence of avoidance.



(a) Deterministic (b) Probabilistic

Figure 4.5 Final trajectory comparison for deterministic and probabilistic cases. The blue circle represents the robot while the gray circles represent the obstacles. The translucent gray patches around the obstacles are the noise samples. The yellow patch at the top right corner is the goal position.

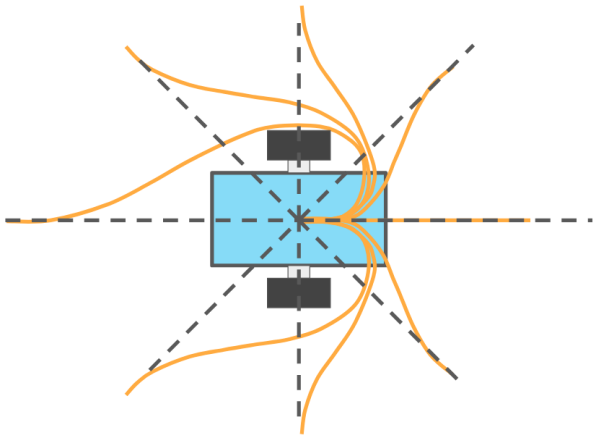


Figure 4.6 The dotted lines represent the trajectories that are formed from the controls obtained from a linear motion model. The orange trajectories are the approximations obtained for respective linear trajectories using the $\zeta(\cdot)$. The orange trajectories are kinematically feasible.

Chapter 5

Conclusions and Future work

In this thesis, we presented two motion planning frameworks that are capable of performing well under perception and actuation uncertainties. The first framework cleverly bypasses the need to compute state estimates of the robot. We showed that our method helps in better collision detection and also leads to less conservative trajectories. Next, we showed how the first framework can be extended to a probabilistic setting. We presented a novel approach to solve the chance constraints in a computationally efficient manner. Our methods focus on the real-time implementation aspects and show that they are easy and efficient to implement. Since our methods are based on the concept of velocity obstacle, they reap all the benefits and guarantees provided by velocity obstacles. The probabilistic navigation framework in the thesis was based on the IVO. But it is straight forward to extend it to other variants of velocity obstacle with minimal effort. We also show how the proposed navigation framework can be extended to other complex motion models and solve them in a convex optimization setting.

Though not a part of the thesis, we explored how the probabilistic collision avoidance problem can be solved differently [30]. To this end, we formulated a robust Model Predictive Control or the aforementioned chance constrained optimization as a problem of distribution matching. We illustrated two approaches that give tractable solutions to optimization problem with complex chance constraints like PVO under non-parametric uncertainty. We first proposed a baseline method that approximates the distribution of the chance constraints with a Gaussian Mixture Model (GMM) and then proceed to perform distribution matching using Kullback Leibler (KL) divergence. Our second method is built on the possibility of embedding distributions in the Reproducing Kernel Hilbert Space (RKHS) and using Maximum Mean Discrepancy as a distribution similarity metric. We evaluated both the GMM-KLD and the RKHS based approaches for quality of maneuvers and computational time.

Our future works include exploring new methods to solve the chance constraints in a computationally efficient way. We also plan to explore the possibility of projecting the chance constraints into a latent space using deep learning methods like variational autoencoders and perform optimization over the latent space for the controls. We also plan to explore the possibility of replacing the probabilistic framework with a reinforcement learning-based model where we can try to make sure that the model understands the need to take the state, perception and actuation uncertainties into account.

Appendix A

Solvers used for optimization

A.1 Choosing an optimal solver

While most of the gradient based solvers should be able to solve the controls for the proposed approaches, we summarize our observations with some of the popular ones. We experimented with solvers based on interior point method (IP), sequential quadratic programming (SQP), and active set methods. While SQP based solvers give us the best results, active set method based solver showed the poorest performance.

SQP and IP based solvers usually give us similar results. When solving for PIVO constraints, we noticed better results with SQP in terms of optimality as well as computational efficiency. We believe that the lower optimality of IP is due to the way it works. Consider the following optimization routine,

$$\begin{aligned} &\min_{u} J(u) \\ &\text{subject to} f(u,\cdot) \geq 0 \end{aligned}$$

For an optimal solution we can write

$$\begin{aligned} \nabla J(u) - \lambda f'(u,\cdot) &= 0 \\ \lambda f(u,\cdot) &= 0 \\ f(u,\cdot) &\geq 0 \\ \lambda &\geq 0 \end{aligned}$$

For interior point, we modify the above inequalities to strict inequalities by introducing slack variables.

$$\begin{aligned} \nabla J(u) - \lambda f'(u, \cdot) &= 0 \\ \lambda f(u, \cdot) &= \mu \\ f(u, \cdot) &> 0 \\ \lambda &> 0 \end{aligned}$$

Where μ is called a barrier parameter. We use Newton's method to solve this and reduce μ to zero. Intuitively speaking, this can be considered as solving for a perturbed variant of the original problem using Newton's method. Our navigation framework runs in loop where it gives a control that the robot needs to take to avoid collision and reach a desired goal for every time step. Here, we are repeatedly trying to solve an optimization problem for every time step. The optimization problems for two consecutive time steps can be treated as perturbed variants of each other. This leads to a lower optimality when using IP based solvers.

A.2 Tuning the solvers for runtime performance

The computational times reported were using the solvers that compute the needed gradient or Hessian automatically. This is usually computationally expensive. Passing the gradients or Hessian function to the solver generally improves the computational efficiency of the solver. One can also cap the maximum iterations and increase the tolerance thresholds for the solver to converge at an acceptable solution much faster.

Related Publications

1. **P. S. Naga Jyotish***, Yash Goel*, A. V. S. Sai Bhargav Kumar, and K. Madhava Krishna. 2019. IVO: Inverse Velocity Obstacles for Real Time Navigation. In *Proceedings of the Advances in Robotics 2019 (AIR 2019)*. Association for Computing Machinery, New York, NY, USA, Article 16, 16. DOI:https://doi.org/10.1145/3352593.3352610

2. **P. S. Naga Jyotish***, Y. Goel*, A. V. S. Sai Bhargav Kumar and K. M. Krishna, "PIVO: Probabilistic Inverse Velocity Obstacle for Navigation under Uncertainty," 2019 *28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019, pp. 1-6, doi: 10.1109/RO-MAN46459.2019.8956406.

Additional Publications

1. **SriSai Naga Jyotish Poonganam***, Bharath Gopalakrishnan*, Venkata Seetharama Sai Bhargav Kumar Avula, Arun Kumar Singh, K. Madhava Krishna and Dinesh Manocha, "Reactive Navigation Under Non-Parametric Uncertainty Through Hilbert Space Embedding of Probabilistic Velocity Obstacles," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2690-2697, April 2020, doi: 10.1109/LRA.2020.2972840.

2. Anish Gupta*, Unni Krishnan R Nair*, **SriSai Naga Jyotish Poonganam**, Arun Kumar Singh, and K. Madhava Krishna, "Non Holonomic Collision Avoidance of Dynamic Obstacles under Non-Parametric Uncertainty: A Hilbert Space Approach", *In review*.

Bibliography

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

[2] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[3] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen. Collision avoidance under bounded localization uncertainty. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1192–1198. IEEE, 2012.

[4] J. G. de Lamadrid. Avoidance of obstacles with unknown trajectories: Locally optimal paths and periodic sensor readings. *The International journal of robotics research*, 13(6):496–507, 1994.

[5] B. Faverjon and P. Tournassoud. *A local based approach for path planning of manipulators with a high number of degrees of freedom*. PhD thesis, INRIA, 1987.

[6] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

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

[8] C. Fulgenzi, A. Spalanzani, and C. Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1610–1616. IEEE, 2007.

[9] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1056–1062. IEEE, 2008.

[10] R. Gayle, A. Sud, M. C. Lin, and D. Manocha. Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3777–3783. IEEE, 2007.

[11] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha. Chance constraint based multi agent navigation under uncertainty. In *Proceedings of the Advances in Robotics*, page 53. ACM, 2017.

[12] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna. Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4169–4176. IEEE, 2014.

[13] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna. Closed form characterization of collision free velocities and confidence bounds for non-holonomic robots in uncertain dynamic environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4961–4968, Sep. 2015.

[14] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna. Closed form characterization of collision free velocities and confidence bounds for non-holonomic robots in uncertain dynamic environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4961–4968, 2015.

[15] E. T. Hall. The hidden dimension, doubleday & co. *Garden City: New York*, 1966.

[16] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[17] P. S. N. Jyotish, Y. Goel, A. V. S. S. B. Kumar, and K. M. Krishna. Ivo: Inverse velocity obstacles for real time navigation. In *Proceedings of the Advances in Robotics 2019*, AIR 2019, New York, NY, USA, 2019. Association for Computing Machinery.

[18] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond. A local collision avoidance method for non-strictly convex polyhedra. *Proceedings of robotics: science and systems IV*, 2008.

[19] B. Kluge and E. Prassler. Recursive probabilistic velocity obstacles for reflective navigation. In *Field and Service Robotics*, pages 71–79. Springer, 2003.

[20] B. Kluge and E. Prassler. Reflective navigation: Individual behaviors and group behaviors. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 4172–4177. IEEE, 2004.

[21] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[22] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*, 2012.

[23] A. S. B. Kumar, A. Modh, M. Babu, B. Gopalakrishnan, and K. M. Krishna. A novel lane merging framework with probabilistic risk based lane selection using time scaled collision cone. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1406–1411. IEEE, 2018.

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

[25] H. Li, O. A. I. Ramirez, and M. Chetouani. Potential human reaction aware mobile robot motion planner: Potential cost minimization framework. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 441–448. IEEE, 2014.

[26] B. Luders, M. Kothari, and J. How. Chance constrained rrt for probabilistic robustness to environmental uncertainty. In *AIAA guidance, navigation, and control conference*, page 8160, 2010.

[27] L. Martinez-Gomez and T. Fraichard. Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation. In *2009 IEEE International Conference on Robotics and Automation*, pages 100–105. IEEE, 2009.

[28] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[29] J. Pettré, P. d. H. Ciechomski, J. Maïm, B. Yersin, J.-P. Laumond, and D. Thalmann. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds*, 17(3-4):445–455, 2006.

[30] S. N. J. Poonganam, B. Gopalakrishnan, V. S. S. B. K. Avula, A. K. Singh, K. M. Krishna, and D. Manocha. Reactive navigation under non-parametric uncertainty through hilbert space embedding of probabilistic velocity obstacles. *IEEE Robotics and Automation Letters*, 5(2):2690–2697, 2020.

[31] G. Sanchez and J.-C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 2112–2119. IEEE, 2002.

[32] A. K. Singh and K. M. Krishna. Reactive collision avoidance for multiple robots by non linear time scaling. In *52nd IEEE Conference on Decision and Control*, pages 952–958. IEEE, 2013.

[33] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011.

[34] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha. Real-time path planning for virtual agents in dynamic environments. In *ACM SIGGRAPH 2008 classes*, page 55. ACM, 2008.

[35] P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[36] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3):1160–1168, 2006.

[37] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[38] J. Van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE, 2008.

[39] H. Zhu and J. Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.