Safety Methods for Robotic Systems

by

Chia-Yin Shih

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Laurent El Ghaoui, Chair
Professor Ruzena Bajcsy
Professor Koushil Sreenath

Fall 2021

Safety Methods for Robotic Systems

Abstract

Safety Methods for Robotic Systems

by

Chia-Yin Shih

Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Laurent El Ghaoui, Chair

Recently there have been vast interests in introducing robotic systems such as autonomous cars and UAVs into the real world. Ensuring the safety of these systems when they are deployed is thus a highly crucial and urgent problem. Safety problems can arise from various different settings such as when there are multiple vehicles or human-operated vehicles in the environment. Different safety-critical settings often require different approaches for addressing the safety of vehicles. In this dissertation, we contribute novel methods for safety problems that arise from three different scenarios.

First, we have seen a surge of interests in deploying autonomous vehicles into the everyday lives of people. Developing accurate and generalizable algorithms for modeling and predicting human behavior thus becomes important. We present a method for generating the probabilistic forward reachable set of a human-controlled vehicle in an environment where a robot is operating in close proximity to the human-controlled vehicle.

Second, motivated by the recent advances in deploying unmanned aerial vehicles into the airspace, we tackle the problem of multi-vehicle safety. We first contribute a planning and control strategy for guaranteeing safety of multiple vehicles while vehicles complete their objectives. We also present an initialization strategy based on machine learning to enhance the safety of multi-vehicle systems when they adopt least-restrictive safety-aware algorithms.

Finally, machine learning has emerged as a promising tool to enable robots to accomplish challenging tasks under uncertainty in the dynamics of the robots or the environment. However, the safety of the robot while it's learning online is often not taken into account, which could lead to unsafe behavior of the robot. We present an online learning framework that enables a robot to learn about its dynamics, accomplish a task, and update its safe set simultaneously online.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Prof. Laurent El Ghaoui for his advice. I am grateful for Prof. Geoff Gordon for showing me that one can be technically capable, decent as a human being, and genuinely interested in knowledge for the sake of knowledge at the same time in academia when I most needed such an example. Thanks also to several other professors who made a positive influence on me. Thanks to Akshara and Franzi for the collaboration on a few of the projects presented in this dissertation.

One of the most valuable things I learned in the past few years is: it is good to know young that righteousness, authenticity, and depth of a person are independent of anything else (such as titles, fame, accolades, gender, or how educated a person is etc). I have truly learned the essence of the saying, “When someone shows you who they truly are, believe them the first time.”

Serving as a TA was a big part of my graduate school life and I am thankful for the numerous professors I TA-ed for. I would particularly like to thank Prof. Matt Gormley, Prof. Ani Adhikari, Prof. Joey Gonzalez, and Prof. Andrew Bray for their mentorship and trust. I developed skills in teaching, mentoring, and managing a team through these experiences.

I am grateful for the funding I received throughout my PhD. It supported my basic living and my travels to many places. I visited around 40 US states and 135 different cities around the world and in the US during my PhD. The funding also enabled me to pursue quite a few adventurous activities.

Thanks to my many teachers in the earlier stage of my life. They showed me what it means to be a great mentor since young. I have a deep respect for what they are as human beings and their sincere dedication in developing young minds. They were not narcissistic, not fake, not manipulative, not cliquey, not ruthless, not vain, not shows, and showed no affinity for obsequiousness etc.

Thanks to all my friends for all the fun we had together. It has truly been a pleasure to have wonderful friends.

Thanks also to the many humanities-related activities I took part in during my PhD. They were also big parts of my PhD life and enriched my life immensely.

Finally, I owe my deepest gratitude to my parents for their unconditional love and support.

Chapter 1

Introduction

Many things we take for granted everyday rely on safe robotic systems. From planes, cars, ships, to trains, it is of utmost importance to have functional mechanical and electrical systems in these vehicles so they can be effectively controlled by human operators. Furthermore, the human operators of these vehicles are generally trained and certified to operate these vehicles safely. These robotic systems have advanced our society and enabled easy continental travels and convenient commuting. In the modern era, these technologies are generally considered reliable and people operating or riding these vehicles trust the safety of these systems.

Robots have further been introduced to tackle wider problems. Many factories use robots in their manufacturing process, particularly on procedures that are highly repetitive like packaging, assembly, and polishing etc [22]. Robots have also been adopted in helping farmers more effectively manage their crops and farms in agricultural settings [13]. Farmers have used drones to monitor their crops [76] and small robots on the ground to capture close-up images of plants [39]. In medicine, robots have been used for wide-ranging purposes such as assisting in surgery to increase precision, transporting medicine, and helping patients get out of bed [64]. In homes, robots have been used to perform household works like vacuuming and mopping [53]. The safety of these robots and the objects they interact with are of high importance when they are deployed.

Given that it seems that robots are widely deployed in a rather safe way in the above examples, what are the safety problems that arise in robotics in recent years? If we look at the type of robotic systems that have already been mature for use in our daily lives, they generally exhibit one or more of the following properties that make it easy to reason about safety or make safety problems less likely to occur: First, the dynamics of the systems that are being controlled is fully known except for some limited degree of noise. Second, they are controlled by humans who use their prior training and experience to navigate around vehicles that are also controlled by humans. Third, how the robots are affecting their environment or the objects they're operating on is carefully monitored by humans. Fourth, the robots are traveling at a very slow speed or operating in a sparse environment with no or very few other robots in close proximity. Fifth, the cost of the potential damage caused by the robot is low.

The next generation of challenge in robotics is to design robots that not only function mechanically and can be controlled by humans but can themselves make intelligent decisions from learning about the environment they're acting in, reasoning about other agents in the environment, and determining how to act safely themselves without human intervention. The goal is to enable fully autonomous robotic systems that can accomplish interesting and challenging tasks while relieving humans from having to supervise or operate them. This is a much more challenging problem as it often doesn't possess the aforementioned properties that make existing robotic systems safe. Next we discuss different facets of the safety problems that arise as we endeavor to enable robots to act more intelligently and autonomously.

Recently, we have seen a lot of interests in introducing self-driving vehicles into the roads [82, 33, 8, 65]. For the vehicles to navigate around other vehicles safely, they need to first be able to make sense of the environment, which involves using perception systems to obtain information such as where objects are in their surrounding environment and the lanes the cars should travel in etc. Based on this information, the robots should generate actions such that they do not collide with static or moving objects. Key to this is the ability to make good predictions on how non-static objects such as cars or pedestrians are going to move in the next few time steps in a myriad of scenarios. This can be challenging as there is generally no explicit communication among the robotic vehicles, the humans operating the other vehicles, and the pedestrians. Hence the inference of future trajectories of humans or other objects are solely based on observations gathered by the robotic vehicles and situational context. Based on this prediction, the robotic vehicles then determine actions that will likely avoid collisions. This decision making process can be made an even more complicated problem when the robots reason about how their actions might instead affect the behavior of other vehicles when they determine what actions to take next, making the problem game-theoretic.

Drone technology has been picking up lots of momentum in the past few years. There have been many commercial, military, and medical applications that aim to tap into the agility and the relatively low cost of fully autonomous drone technology for transport and delivery [5, 47, 78, 86, 79, 19, 38]. There may be many drones operating in close proximity in the airspace in the near future, making it challenging to maintain safety for all the vehicles. In addition, compared to typical vehicles traveling on the ground in specified lanes, drones in the air generally travel in a more unstructured fashion. The safety of these drones are crucial, not only for the hardware itself but also for the properties and humans on the ground that the drones may crash on if they malfunction or collide in the air. The problem of detecting potential conflicts and determining when and how to avoid collisions for any agent in the multi-vehicle system is thus of high importance.

1.1 Outline and contributions of this thesis

In Chapter 2 Background, we give a high level introduction to technical ideas that our works in the subsequent chapters are built on. We first give an overview of Hamilton-Jacobi reachability, a control theoretic framework that can be used to compute safe sets. Then we offer some background on machine learning knowledge that our works rely on.

In Part I Safety in Human-operated Space, we tackle safety problems that arise in scenarios where humans and robots operate vehicles in close proximity. In Chapter 3, we first demonstrate how to incorporate information derived from HJ reachability into a machine learning problem which predicts human behavior in a simulated collision avoidance context, and show that this yields a higher prediction accuracy than learning without this information. Then we propose a framework to generate probabilistic forward reachable sets that flexibly provides the probability of whether a human-controlled vehicle will be in a given region for different regions and generalizes to novel scenarios.

In Part II Safety for Multi-Vehicle Systems, we develop approaches for addressing the safety of multiple vehicles. In Chapter 4, we propose a novel reachability-based approach that guarantees safety for *any* number of vehicles while vehicles complete their objectives of visiting multiple targets efficiently, given any K-vehicle collision avoidance algorithm where K can in general be a small number. Our proposed method is scalable to large number of vehicles with little computation overhead.

In Chapter 5 of Part II, we propose a novel approach using machine learning to enhance the safety of vehicles by proposing new initial states in very *close* neighborhood of the original initial states of vehicles while vehicles use *any* least-restrictive safety-aware algorithm to get to their goals.

In Part III Safety under Uncertain Dynamics, we tackle problems in the realm of safety of robots under uncertain dynamics. In Chapter 6, we propose a novel framework to learn a safe control policy in simulation, and use it to generate online safe sets under uncertain dynamics. We also show an application of our framework to a model-based reinforcement learning problem, proposing a safe model-based RL setup. Our framework enables robots to simultaneously learn about their dynamics, accomplish tasks, and update their safe sets online.

In Chapter 7, we summarize our contributions and provide possible future directions in the context of safety for robots.

Chapter 2

Background

In this chapter, we describe mathematical background that our works build on. First, we give a brief overview of Hamilton-Jacobi (HJ) reachability, a mathematical framework that addresses the safety of dynamical systems. Then we give a high level introduction to machine learning concepts central to some of our works.

2.1 Hamilton-Jacobi Reachability

Hamilton-Jacobi reachability is a control theoretic framework that can be used to characterize and compute the set of all states a vehicle could be at or the set of all states such that a vehicle can remain safe from another vehicle or obstacles in the environment with optimal control. While reachability can also be used to compute sets such as reachable tubes or reach-avoid sets, we focus on reachability concepts and background that are directly relevant to our works in this chapter. We can broadly characterize reachability problems into single-player problems and two-player problems.

2.1.1 Forward reachable sets for single-player systems

Given a single-player system with state $x \in \mathcal{X}$, action $u \in \mathcal{U}$, and dynamics $\dot{x} = f(x, u)$, we define the forward reachable set (FRS) with horizon T as the set of all states that the vehicle can possibly be in within the time interval $[0, T]$ if the vehicle starts in some initial set $\mathcal{L}$ at time $t = 0$. Mathematically, the FRS with time horizon T and initial set $\mathcal{L}$ can be expressed as

$$\mathcal{F}(T) = \{x : \exists u(t) \in \mathcal{U} \text{ s.t. } t \in [0, T], x(\cdot) \text{ satisfies } \dot{x} = f(x, u), x(0) \in \mathcal{L}, x(t) = x\}. \quad (2.1)$$

To compute the forward reachable set, one can formulate the problem into an HJ PDE and use techniques described in [63] to solve for the solution.

2.1.2 Safe sets for single-player systems

Consider a single-player system with state $x \in \mathcal{X}$, action $u \in \mathcal{U}$, and dynamics $\dot{x} = f(x, u)$. We assume that f is uniformly continuous, bounded, and Lipschitz continuous in arguments x for fixed u.

Let $\mathcal{Z}$ represent the danger zone, which is the set of states that the robot should avoid. We can define $\mathcal{Z}$ by using a level set function $l(x)$ such that $l(x) \leq 0$ if and only if $x \in \mathcal{Z}$.

Let $\xi_x^{\boldsymbol{u}}(\cdot)$ be the trajectory resulting from executing $\boldsymbol{u}(\cdot)$ from state x. Then the value function V at a state x is defined as

$$V(x) = \sup_{\boldsymbol{u}(\cdot)} \inf_{t \geq 0} l(\xi_x^{\boldsymbol{u}}(t)).$$

Intuitively, $V(x)$ is the closest the trajectory gets to the danger zone $\mathcal{Z}$ given the *best possible control* to avoid the danger zone. The safe set $\mathcal{K}$ is defined as $\mathcal{K} = \{x : V(x) > 0\}$ and the unsafe set is then the complement of $\mathcal{K}$. Intuitively, the safe set $\mathcal{K}$ is the set of states such that there exists at least one control strategy for the system to avoid the danger zone $\mathcal{Z}$. For a finite time horizon $t \in [0, T]$, this value function can be computed by solving the Hamilton-Jacobi-Bellman variational inequality described in [63] for continuous systems.

We can also formulate the discrete-time version of the value function $V(x)$ for the infinite-horizon case as follows

$$V(x) = \min \left\{ l(x), \max_{u \in \mathcal{U}} V(x + f(x, u)\Delta t) \right\}.$$

In practice, due to errors introduced by discretization, it is common to introduce a safety level $\epsilon > 0$ such that the safe set is defined as $\mathcal{K} = \{x : V(x) > \epsilon\}$.

2.1.3 Backward reachable sets for two-player systems

HJ reachability can provide safety certificates by characterizing the set of states that could, under the worst case behavior of the unknown but bounded disturbance, lead to danger.

We give a brief overview of how to apply HJ reachability to solve a pairwise collision avoidance problem such as the one in [63]. Consider two vehicles Q_1, Q_2 described by the following ordinary differential equations (ODE):

$$\dot{x}_i = f_i(x_i, u_i), \quad u_i \in \mathcal{U}_i, \quad i = 1, 2. \tag{2.2}$$

Given the dynamics of the two vehicles (5.1), we can derive the relative dynamics in the form of (2.3):

$$\begin{aligned} &\dot{x}_{ij} = g_{ij}(x_{ij}, u_i, u_j) \\ &u_i \in \mathcal{U}_i, u_j \in \mathcal{U}_j \quad i, j = 1, 2, \quad i \neq j \end{aligned} \tag{2.3}$$

where x_{ij} is a relative state representation between vehicles Q_i and Q_j.

We assume the functions f_i and g_{ij} are uniformly continuous, bounded, and Lipschitz continuous in arguments x_i and x_{ij} respectively for fixed u_i and (u_i, u_j) respectively. In addition, the control functions $u_i(\cdot) \in \mathbb{U}_i$ are drawn from the set of measurable functions[1].

The set of states that represents a collision is denoted as $\mathcal{Z}_{ij}$, and we compute the following backward reachable set (BRS), which is the set of states from which a collision could occur over $[0, t]$ based on the worst case action of Q_j:

$$\begin{aligned}&\mathcal{V}_{ij}(t) = \{x_{ij} : \forall u_i \in \mathbb{U}_i, \exists u_j \in \mathbb{U}_j,\\ &x_{ij}(\cdot) \text{ satisfies } (2.3), \exists s \in [0, t], x_{ij}(s) \in \mathcal{Z}_{ij}\}.\end{aligned} \tag{2.4}$$

Reachability theory is valid for any time horizon t; however, for clarity, we will let $t \to \infty$ in this dissertation. $\mathcal{V}_{ij}$ can be obtained as the sub-zero level set of the viscosity solution $V_{ij}(t, x)$ of a terminal value HJ PDE. For details on obtaining V_{ij}, please see [63]. The BRS can thus be denoted as $\mathcal{V}_{ij} = \{x_{ij} \in \mathbb{R}^n : \lim_{t\to\infty} V_{ij}(t, x_{ij}) \leq 0\}$. We will also use a slight abuse of notation and write $V_{ij}(x_{ij}) = \lim_{t\to\infty} V_{ij}(t, x_{ij})$. The interpretation is that Q_i is guaranteed to be able to avoid collision with Q_j over an infinite time horizon as long as the optimal control

$$u^*_{ij} = \arg\max_{u_i \in \mathcal{U}} \min_{u_j \in \mathcal{U}} D_{x_{ij}} V(x_{ij}) \cdot g_{ij}(x_{ij}, u_i, u_j) \tag{2.5}$$

is applied as soon as the potential conflict occurs, represented by the sub-zero level set of $V_{ij}(x_{ij})$.

2.1.3.1 Least restrictive safe control

As we've seen above on HJ reachability, as long as the optimal safe control in Equation (2.5) is applied by Q_i at the boundary of the BRS $\mathcal{V}_{ij}$, Q_i will remain safe from Q_j for all time. A similar least-restrictive safe control strategy based on reachability can be adopted for single agent systems that aim to avoid dangerous regions in the environment. This enables a control strategy where an agent gets to execute any type of controller such as a goal controller that gets the vehicle to its target [24], [35] or a learning-based controller [2], [69] when the agent is not at the boundary of a backward reachable set. We summarize the control of each vehicle at each time step when it adopts a least-restrictive safe controller below:

Algorithm 2.1: Least-restrictive safety strategy

```
if Vehicle is at the boundary of an unsafe set then
    Safe control is applied.
else
    Any control such as a control to get the vehicle to its target or a machine learned
    control can be applied.
end
```

[1] A function $f : X \to Y$ between two measurable spaces (X, Σ_X) and (Y, Σ_Y) is said to be measurable if the preimage of a measurable set in Y is a measurable set in X, that is: $\forall V \in \Sigma_Y, f^{-1}(V) \in \Sigma_X$, with Σ_X, Σ_Y σ-algebras on X,Y.

This can be a highly desirable property in a safety strategy because it offers high flexibility for agents to execute whatever control they would like when they're deemed safe and decouples the reasoning of safety-oriented controllers and task-oriented controllers. On the other hand, this means that the system performs a zero-step look-ahead at each time step online and does not reason how its current action affect the future trajectories of the vehicles in the environment. Theoretically we can incorporate safety derived from reachability into a trajectory optimization problem and reason about future trajectories online. However, it would be very difficult to perform this optimization online efficiently due to the nature that the safety value function $V_{ij}(x_{ij})$ is a discrete look-up table pre-computed offline. Thus incorporating the safety values prevents the use of typical efficient optimization methods that work on continuous optimization problems. One could theoretically perform the optimization with a sampling-based method, however, this scales poorly with the time horizon of the trajectory and the number of vehicles.

2.2 Machine Learning

Machine learning (ML) has emerged as a promising tool for many application domains such as vision, speech, and robotics. ML methods have shown high potential in tackling problems when the dimension of the task is high or when direct modeling is not feasible due to the complexity of the system and the substantial computation required.

Machine learning is a field that aims to figure out how to learn from data to best predict on new data or make decisions for systems. It encompasses subfields such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. We will give a brief introduction on supervised learning and reinforcement learning below.

2.2.1 Supervised learning

In supervised learning, we are given both the input and the output of each training example where the outputs are what we call supervisions. The goal of a supervised learning task is to learn to best predict the outputs given the inputs from unseen input-output pairs.

Supervised learning can be further categorized into regression and classification tasks. In regression, the outputs take on continuous values; in classification, the outputs are discrete classes and we typically call the outputs "labels" in this scenario. In this dissertation, we are mainly concerned with classification tasks so we focus our attention on classification problems in this section.

Given a data set $\{(x_j, y_j)\}_{j=1}^n$ where x_j is a vector representing the input data point and y_j is the corresponding label for the input data point, a classification problem aims to learn to predict the labels given the inputs through training. In general, we often transform, with a function $\phi(\cdot)$, the original representation of each input data point x to a feature vector $\phi(x)$ and use the transformed feature for learning. How to determine this mapping $\phi(\cdot)$ can be nontrivial and is often referred to as "feature engineering". There are various types of models that have been proposed for classification tasks. We will go over a few of them on a high level and delve deeper into models that use probabilistic modeling as understanding them more deeply is required for our work.

Support vector machine (SVM) learns the maximum margin separating hyperplane to separate data with different labels and uses this learned hyperplane for prediction on unseen data. The hard-margin SVM does not permit any training data point to be misclassified and hence it can only be used on data that is linearly separable. On the other hand, the soft-margin SVM allows the plane to misclassify data points and uses a hyper-parameter to determine the trade-off between the maximum margin and the degree of the misclassification errors allowed.

A decision tree is another common model used for classification. Its goal is to build a tree-based structure such that given a data point, starting from the root node, a decision rule is applied to the data point to determine whether it should go to the left subtree or the right subtree of the root node next; then recursively each subsequent node the data point passes through has itself a decision rule to determine the subtree the data point should go to until a leaf node is reached. Once the data point arrives at the leaf node, a label is predicted for the data point based on the learned classification rule at the leaf. The goal of the learning is then to learn to build a tree and determine the decision rule at each node of the tree. Here we won't go into the details of how such a model is trained, the typical decision rules, and practical design decisions for learning a good decision tree model.

2.2.1.1 Probabilistic modeling

In this subsection, we go deeper into probabilistic modeling for classification tasks in ML. To simplify things, we focus here on the scenario where the labels y are binary and take on values of either 0s or 1s.

In particular, we model the probability that feature h is associated with label $y = 1$ as $f_\theta(h)$ where $f_\theta(h)$ is some function parameterized by θ, i.e., $p(y = 1|h) = f_\theta(h)$ and, equivalently, $p(y = 0|h) = 1 - f_\theta(h)$. The goal is to learn the parameters θ from data. $f_\theta(h)$ can in general be any function such as a neural network.

By assuming all observations are independent and identically distributed, a common assumption made in probabilistically-motivated ML modeling, we have that the likelihood of observing the data points $\{(h_j, y_j)\}_{h=1}^n$ given θ is

$$L(\theta) = \prod_{j=1}^{n} f_\theta(h_j)^{y_j} (1 - f_\theta(h_j))^{1-y_j}. \tag{2.6}$$

The goal of learning is then to maximize the above likelihood with respect to θ. This is equivalent to minimizing the negative log likelihood on the data set with respect to θ:

$$\sum_{j=1}^{n} -y_j \log f_\theta(h_j) - (1 - y_j) \log (1 - f_\theta(h_j)) \, . \tag{2.7}$$

2.2.2 Reinforcement learning

Reinforcement learning (RL) is the study on how to enable agents to make decisions that maximize the cumulative sum of rewards in a given environment. On a high level, reinforcement learning can be categorized into model-free RL and model-based RL. A model-based RL algorithm explicitly builds a model for the dynamics of the world using data gathered from the agent's interactions with the world and updates the model as more data is gathered. Using this model, the agent selects an action to take at each time step. On the other hand, model-free RL does not explicitly build a model of the dynamics. There have also been RL works that mix both model-free and model-based RL, but we will not go into the details of this.

Since our work includes an extension of model-based RL, a more sample-efficient learning framework, here we give a slightly deeper overview of the model-based RL framework. It has shown recent success at complex robotics tasks [31], even on hardware [14].

Model-based reinforcement learning (MBRL) iteratively tries to optimize a policy to accomplish a task, and learns the dynamics of the robot. Similar to [31], we use model-predictive control (MPC) to optimize the policy in Chapter 6. Given $c(x_h, x_h)$, the cost of an agent taking action u_h from state x_h, the objective of model-predictive control (MPC) is to minimize the total cost $J = \sum_{h=t}^{t+H-1} c(\mathbf{x}_h, \mathbf{u}_h)$ with respect to the actions $u_{t:t+H-1} \equiv \{u_t, \ldots, u_{t+H-1}\}$ over a horizon H from current the time step t subject to dynamics constraint. After the MPC problem is solved, the first action u_t is applied to the system, and the process repeats, starting with the new current state.

Part I

Safety in Human-operated Space

Chapter 3

Predicting Probabilistic Human Forward Reachable Sets Based on Learned Human Behavior

The research presented in this chapter was originally published in the paper Predicting Stochastic Human Forward Reachable Sets Based on Learned Human Behavior in American Control Conference 2019 [70].

3.1 Introduction and Related Work

In recent years, there has been much excitement in introducing intelligent systems that can navigate autonomously in environments with humans. For example, many self-driving car companies [82], [33], [8] have emerged in the past few years. Furthermore, projects like Amazon Prime Air [6] and Google's Project Wing [20] aim to tap into the airspace for package delivery. There has also been immense interest in using UAVs for disaster response [9]. While roads provide structure, in airspace the interaction of vehicles occurs in a more unstructured setting, presenting the challenge of making predictions based on more limited information.

Critical to introducing autonomous vehicles into the workspace of humans is safety. The authors in [63] have focused on safety from a differential game perspective by characterizing the set of states from which one vehicle is guaranteed to be safe from a second assuming the second can take worst case actions within a bounded set. In [2], the authors demonstrate learning these bounds online in an uncertain environment. The authors in [51] characterize the notion of safety using torque limits on the robot. However, these works don't consider the complexity of having humans in the environment. Taking humans into account is challenging because unlike dynamics model governing the robotic systems, we don't generally have models for how humans make decisions in unstructured scenarios.

Key to having safe human-robot interaction is the ability for robots to navigate safely around human-operated vehicles. To enable this, the autonomous systems need to make predictions of human behaviors to avoid collisions. Past work has used various methods to perform behavior prediction of vehicles operated by humans. Many works have modeled humans as dynamical systems optimizing their own cost functions [1], [85]. Inverse reinforcement learning aims to learn these cost functions by observing past trajectories of the humans. There has also been work on using recurrent neural networks to make predictions on future trajectories [83], [3]. However, the methods presented in [1]-[3] only generate a single trajectory and do not provide probabilistic information of future trajectories. Since human behavior is noisy, having probabilistic information on future trajectories is important.

In the control theory literature, HJ reachability theory models the worst case scenario [63] and hence is often overly conservative when applied to real world scenarios. There has been a growing body of work in using probabilistic reachable sets to reduce conservatism and model uncertainty. For example, [59] derives a probabilistic reachable set and uses it for motion planning by making assumptions on the behavior of moving obstacles *a priori*. However, humans often don't satisfy these assumptions and human behavior is also often influenced by the behavior of other vehicles in the environment.

In [61], the authors develop the notion of a human safe set for a human supervisor, to model when the supervisor would start to intervene with robot teams operating on their own in order to avoid static obstacles. However, this does not provide information on a mapping from *any* joint configuration of the human and robot to the human's action when we aim to also model how the human would avoid, for example, predicting the direction of avoidance, and the entire avoidance process. In this paper, we also aim to model the situation in which humans are avoiding moving robots and comprehensively evaluate the effectiveness of incorporating varying degrees of information derived from HJ reachability.

In [48], the authors learn a forward reachable set by optimizing the disturbance bound, learning from data that is repeatedly gathered from similar initial configurations and generating a reachable set that satisfies an accuracy threshold. However, if we aim to apply the learned reachable set to a scenario in which the two vehicles are approaching from an angle different from what's seen in the training data set, the reachable set generated will not be suitable. In [37], the authors assume that there are different modes that humans are in and learn a reachable set for each of these distinct modes. In this paper, we are interested in prediction methods that generalize to novel scenarios without the need to train a reachable set for each scenario.

In this paper, we first demonstrate how to incorporate information derived from HJ reachability in learning human behavior in a specific example of a simulated two vehicle unstructured setting. We illustrate that the use of HJ reachability yields considerable improvement in predicting human behavior when compared to not using it, for the humans that participated in our study. Furthermore, we propose a framework for learning probabilistic human forward reachable sets (PHFRS) that flexibly captures regions with varying levels of safety. The proposed framework generalizes to prediction in scenarios not trained on. We validate our approaches on data gathered from human experiments with 8 participants.

The paper is organized as follows: Section 3.2 presents our problem statement, how we incorporate information derived from HJ reachability into the learning problem, and the probabilistic reachable set framework. Section 3.3 presents our experiment setup, evaluation metrics, experimental results, and an implementation of the probabilistic forward reachable set method described in 3.2. Section 3.4 includes conclusion and future work.

3.2 Methodology

We aim to predict the behavior of a human controlling a vehicle (which we will call $Q_{\mathcal{H}}$ for human vehicle) in a shared space with an autonomous vehicle (called $Q_{\mathcal{R}}$ for robot vehicle), in an unstructured setting. The unstructured setting presents the challenge of using a limited amount of information to correctly model humans. The methodology section consists of two parts. In the first, we frame the learning problem and propose a way to use information derived from HJ reachability in making better predictions. In the second, we propose a framework to generate probabilistic human forward reachable sets (PHFRS) using the behavior prediction obtained from the result of the learning problem in the first part, although the proposed framework can work with any learning model that outputs probabilistic prediction over actions of humans.

3.2.1 Predicting human actions

3.2.1.1 Problem Statement

In an environment with a human vehicle $Q_{\mathcal{H}}$ and a robot vehicle $Q_{\mathcal{R}}$ with dynamics in the form of (5.1), the human operator controls $Q_{\mathcal{H}}$ to avoid colliding with $Q_{\mathcal{R}}$. The human can provide control from a discrete set of M control inputs $\{u_1, \dots, u_M\}$ where $u_{min} \leq u_m \leq u_{max}, m \in \{1, \dots, M\}$. For example, these could be one-dimensional real numbers corresponding to turning clockwise, turning counter-clockwise, or going straight. Let the states of the human and robot be denoted as $x_{\mathcal{H}}$ and $x_{\mathcal{R}}$ respectively. We aim to learn to predict the human action $\hat{u}$ within $\{u_1, \dots, u_M\}$ at any joint configuration of $Q_{\mathcal{H}}$ and $Q_{\mathcal{R}}$. Our first contribution is to show through a set of human experiments that incorporating information derived from HJ reachability into the proposed learning problem can improve accuracy in prediction.

3.2.1.2 Proposed method

In machine learning, feature vectors are representations of data points that can potentially improve the predictive performance. Since we're working in an unstructured environment, we avoid direct dependency on absolute states and use the relative state $x_{\mathcal{HR}} = x_{\mathcal{H}} - x_{\mathcal{R}}$.

The construction of good features is in general a challenging but important problem in machine learning [36]. In this project, we choose from a set of ”standard” geometric features, and augment with features derived from the safety value functions. The standard features we can incorporate are the translational and rotational components of the relative state. We also include cosine and sine of the rotational components in the relative states to provide nonlinear angular information. We denote features relevant to translation properties as $\vec{g}_t$, features relevant to rotational properties as $\vec{g}_r$, and features relevant to the trigonometric properties of the angles as $\vec{g}_{trig}$. For example, if the state of the vehicle is described by $x_{\mathcal{H}} = \begin{bmatrix} x_{\mathcal{H},1} & x_{\mathcal{H},2} & x_{\mathcal{H},3} \end{bmatrix}$ and $x_{\mathcal{R}} = \begin{bmatrix} x_{\mathcal{R},1} & x_{\mathcal{R},2} & x_{\mathcal{R},3} \end{bmatrix}$ where the first, second, and third components correspond to the x coordinates, y coordinates, and rotational angles, the relative state is $x_{\mathcal{HR}} = \begin{bmatrix} x_{\mathcal{HR},1} & x_{\mathcal{HR},2} & x_{\mathcal{HR},3} \end{bmatrix} = \begin{bmatrix} x_{\mathcal{H},1} - x_{\mathcal{R},1} & x_{\mathcal{H},2} - x_{\mathcal{R},2} & x_{\mathcal{H},3} - x_{\mathcal{R},3} \end{bmatrix}$. Then based on our feature construction method, $\vec{g}_t := \begin{bmatrix} |x_{\mathcal{HR},1}| & |x_{\mathcal{HR},2}| \end{bmatrix}^T$, $\vec{g}_r := \begin{bmatrix} x_{\mathcal{HR},3} \end{bmatrix}^T$, and $\vec{g}_{trig} := \begin{bmatrix} cos\left(x_{\mathcal{HR},3}\right) & sin\left(x_{\mathcal{HR},3}\right) \end{bmatrix}^T$. Combining everything, we have that the standard feature vector has the form $\vec{g}_{std} := \begin{bmatrix} \vec{g}_t^T & \vec{g}_r^T & \vec{g}_{trig}^T \end{bmatrix}^T$.

In safety critical scenarios, past works predominantly incorporate distance, measured by the l-2 norm, $\|\cdot\|_2$, of the translational properties in relative states, as a feature in learning human behavior. For example, using the previous example, the distance feature is $g_d := \left\| \begin{bmatrix} x_{\mathcal{HR},1} & x_{\mathcal{HR},2} \end{bmatrix} \right\|_2$.

We hypothesize that safety levels derived from HJ reachability can potentially provide crucial information in predicting human action and the safety levels viewed from different agents may both provide valuable information. Let the safety levels of the human with respect to the robot be denoted as $V_{\mathcal{HR}}(x_{\mathcal{HR}})$ and the safety levels of the robot with respect to the human be denoted as $V_{\mathcal{RH}}(x_{\mathcal{RH}})$. If we include all the features proposed so far, we have the feature vector $\vec{g} = \begin{bmatrix} \vec{g}_{std}^T & g_d & V_{\mathcal{HR}}(x_{\mathcal{HR}}) & V_{\mathcal{RH}}(x_{\mathcal{RH}}) \end{bmatrix}^T$.

Now we present how we learn the human actions given the construction of feature vector $\vec{g}$. Let the data set be represented as $\{x_{\mathcal{HR},n}, u_n\}_{n=1}^N$, where N represents the number of data points, $x_{\mathcal{HR},n}$ represents the relative state of data point n, and u_n represents the action the human took at this relative state. We call u_n the label for datapoint $x_{\mathcal{HR},n}$. Let $\vec{g}_n$ be the feature vector for $x_{\mathcal{HR},n}$.

To model the problem probabilistically, we denote the probability of label u_n of a datapoint $x_{\mathcal{HR},n}$ as the function $P(u_n|x_{\mathcal{HR},n};\theta)$ that is parameterized by θ, with the goal to learn the parameter θ. We assume that the data points are independent and identically distributed (iid) and we learn the parameter θ by maximizing the probability

$$\underset{\theta}{max} \prod_{n=1}^{N} P(u_n|x_{\mathcal{HR},n};\theta). \tag{3.1}$$

This is referred to as the maximum likelihood method. Here we refer to $P(u_n|x_{\mathcal{HR},n};\theta)$ as the function approximator.

Alternatively, we could make no probabilistic assumption on the data. For example, a Support Vector Machine (SVM) finds a hyperplane that separates data points with different labels. Another example is a decision tree, which uses a tree-like structure to separate the data by recursively grouping it based on the decision rule at each node of the tree.

3.2.2 A framework for generating probabilistic human forward reachable sets (PHFRS) based on learned human behavior

3.2.2.1 Problem statement

We aim to provide a framework to generate probabilistic human forward reachable sets (PHFRS), with varying levels of safety, that capture future trajectories of Q_H. Mathematically, the problem is defined as follows: Let T_c denote the current time step. Given the past and current states of the trajectory, $\{x_{\mathcal{H}}^{(i)}, x_{\mathcal{R}}^{(i)}\}_{i=0}^{T_c}$, develop a framework to generate a PHFRS $\mathcal{S}$ composed of F mutually disjoint regions $\mathcal{S}_j$ such that each region $\mathcal{S}_j$ is associated with a probability $p_{\mathcal{S}_j}$. Furthermore, the probabilities $p_{\mathcal{S}_j}$'s should satisfy $\sum_{j=1}^{F} p_{\mathcal{S}_j} = 1$.

We say that a region provides a higher *safety probability* if the region captures future trajectories with a higher probability. Our second main contribution is thus a framework that provides varying levels of safety probabilities through probabilistic reachable sets in novel scenarios.

3.2.2.2 Proposed method

To generate a PHFRS satisfying the properties described earlier, we learn F forward reachable sets, $\mathcal{F}_1, \ldots, \mathcal{F}_F$ that satisfy the following properties:

- $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \cdots \subseteq \mathcal{F}_{F-1} \subseteq \mathcal{F}_F$.
- We associate a probability $p_{\mathcal{F}_j}$ with each reachable set $\mathcal{F}_j$. This probability specifies the likelihood that the trajectory in the next T time steps will be within $\mathcal{F}_j$. $p_{\mathcal{F}_1}, \ldots, p_{\mathcal{F}_F}$ should satisfy $p_{\mathcal{F}_1} \geq \underline{p}$, $p_{\mathcal{F}_F} = 1$, and $p_{\mathcal{F}_{j+1}} \geq p_{\mathcal{F}_j}, \forall j \in \{1, \ldots, F-1\}$. Note that $\underline{p}$ represents the desired minimum probability with which the smallest reachable set $\mathcal{F}_1$ should capture the human trajectories.

Note that we define $\mathcal{F}_0 = \emptyset$. With this definition , $p_{\mathcal{F}_0} = 0$. The set of constraints $p_{\mathcal{F}_{j+1}} \geq p_{\mathcal{F}_j}, \forall j \in \{1, \ldots, F-1\}$ encourages the set of reachable sets to provide increasing levels of safety. The constraint $p_{\mathcal{F}_F} = 1$ enables us to provide a safety certificate in the worst case scenario. Observe that if we let $\mathcal{S}_j = \mathcal{F}_j \setminus \mathcal{F}_{j-1}$ and let $p_{\mathcal{S}_j} = p_{\mathcal{F}_j} - p_{\mathcal{F}_{j-1}}$, we will have $\sum_{j=1}^{F} p_{\mathcal{S}_j} = 1$, as desired.

We propose to generate these reachable sets $\mathcal{F}_j$'s by learning time-varying control input bounds for each. For simplicity, we consider the case in which the human input is one dimensional, though our method generalizes to the multi-dimensional case.

Definition 1. ***Time-varying bounds on control inputs:*** *Let $\underline{u}_j^{(i)}$ and $\bar{u}_j^{(i)}$ denote the lower and upper bounds on the control inputs for reachable set $\mathcal{F}_j$, between time t_{T_c+i} and t_{T_c+i+1}. This is defined for all $\forall i \in \{0, \ldots, T-1\}$, $\forall j \in \{1, \ldots, F\}$. Note that t_{T_c+i} corresponds to the real-valued time at time step $T_c + i$.*

Given $\underline{u}_j^{(i)}$ and $\bar{u}_j^{(i)}$ for all $i \in \{0, \ldots, T-1\}$, we can construct the forward reachable set $\mathcal{F}_j$ using the following definition.

Definition 2. ***Forward reachable set with time-varying constraints on inputs****: Suppose we have time varying constraints for the control inputs, i.e., there exists $t_0, \dots, t_T$ such that $0 = t_0 < t_1 < \dots < t_{T-1} < t_T$ and the control input between time t_i and time t_{i+1} is within the set $U^{(i)} = [\underline{u}^{(i)}, \bar{u}^{(i)}]$. Denote $\mathcal{L}$ as the initial set of states to grow the reachable set from. To make this well defined on the boundary time points, we define the reachable set to be $\mathcal{F}(t_0) = \mathcal{L}$, $\mathcal{F}(t_{i+1}) = \{x : \exists u(t) \in U^{(i)}$ s.t. $t \in [t_i, t_{i+1}], x(\cdot)$ satisfies $\dot{x} = f(x, u), x(t_i) \in \mathcal{F}(t_i), x(t) = x\}$.* [1]

Algorithm 3.1 presents how we determine the time-varying bounds $\underline{u}_j^{(i)}, \bar{u}_j^{(i)}$. The algorithm takes in the following arguments: $\{x^{(i)}\}_{i=0}^{T_c}$, G, $\{\epsilon_j\}_{j=1}^{F}$, and $\{k^{(i)}\}_{i=0}^{T-1}$. We use $x^{(i)}$ as a shorthand notation for $(x_{\mathcal{H}}^{(i)}, x_{\mathcal{R}}^{(i)})$. For generality, the algorithm takes in joint states from all past time steps and the current time step T_c. The argument G represents any learned function that can take in the past and current joint configurations of the vehicles, and outputs a probabilistic distribution over the set of possible actions. Note that we allow G to take in the past states for prediction here, however, it could also be the case that G makes predictions solely based on the current joint configuration like the function approximators learned in 3.2.1. As we illustrate in the next paragraph as we describe our Algorithm, the scalar ϵ_j determines how much to grow the input bounds for $\mathcal{F}_j$, and the positive integer $k^{(i)}$ indicates the number of likely actions we use to generate these bounds at time step i.

Algorithm 3.1, which learns time varying bounds for the human inputs, is described below. On line 1, "low" and "high" predicted joint states are initialized with the values of the current joint states; here $\hat{x}_l^{(T_c+i)}, i > 0$, refers to the predicted joint states at time step $T_c + i$, using the low control input value, the subscript h is used for the high control input value. Inside the loop, lines 3-4 obtain the $k^{(i)}$ top likely actions based on the learned function G, for both the low and high cases; and lines 5-6 obtain the corresponding lowest and highest value inputs, denoted $\underline{u}^{(i)}, \bar{u}^{(i)}$ respectively. In lines 7-10, we obtain the corresponding desired $\underline{u}_j^{(i)}, \bar{u}_j^{(i)}$ for each reachable set $\mathcal{F}_j$ by expanding $\underline{u}^{(i)}, \bar{u}^{(i)}$ with ϵ_j as shown. Lines 11-12 obtain the robot's actions for both the l and h cases using the function getRobotAction, which is defined using the robot's model. On line 13, the predicted actions of the human for the case l and h at time step $T_c + i$ are set to be the lower and upper bounds, respectively, of the control input for the smallest reachable set $\mathcal{F}_1$. Lines 14-18 simulate the predicted states of the human and robot for the next time step $T_c + i + 1$. This process repeats as i increments.

We now present the conditions ϵ_j's should satisfy to enable $\mathcal{F}_j \subseteq \mathcal{F}_{j+1}, \forall j \in \{1, \dots, F-1\}$.

Theorem 1. *Based on this reachable set generation scheme, if for any $j \in \{1, \dots, F-1\}$, $\epsilon_j \leq \epsilon_{j+1}$ and $\forall j \in \{1, \dots, F\}, \epsilon_j \geq 0$, for any $j \in \{1, \dots, F-1\}$, $\mathcal{F}_j \subseteq \mathcal{F}_{j+1}$.*

[1] In this definition, we adopt the convention used in hybrid systems, in which the input bound switch corresponds to a mode switch, which allows us to directly use the Level Set Toolbox [62] from past work.

Algorithm 3.1: Generation of time-varying input bounds for reachable sets $\mathcal{F}_j$'s

```
input  : $\{x\}_{i=0}^{T_c}$, $G$, $\{\epsilon_j\}_{j=1}^{F}$, $\{k^{(i)}\}_{i=0}^{T-1}$
output: $\underline{u}_j^{(i)}, \bar{u}_j^{(i)}$ for $i \in \{0, \ldots, T_c - 1\}$, $j \in \{1, \ldots, F\}$
Assign $\hat{x}_l^{(T_c)} = x^{(T_c)}$, $\hat{x}_h^{(T_c)} = x^{(T_c)}$;
for $i \leftarrow 0$ to $T - 1$ do
    set_$\hat{u}_{\mathcal{H},l}$ = getTopKPredictedHumanActions(G, $\{x\}_{i=0}^{T_c-1}$, $\{\hat{x}_l\}_{i=T_c}^{T_c+i}$, $k^{(i)}$ ) ;
    set_$\hat{u}_{\mathcal{H},h}$ = getTopKPredictedHumanActions(G, $\{x\}_{i=0}^{T_c-1}$, $\{\hat{x}_h\}_{i=T_c}^{T_c+i}$, $k^{(i)}$ ) ;
    $\underline{u}^{(i)} =$ getMin (set_$\hat{u}_{\mathcal{H},l}$, set_$\hat{u}_{\mathcal{H},h}$) ;
    $\bar{u}^{(i)} =$ getMax (set_$\hat{u}_{\mathcal{H},l}$, set_$\hat{u}_{\mathcal{H},h}$) ;
    for $j \leftarrow 1$ to $F$ do
        $\underline{u}_j^{(i)} = \max(\underline{u}^{(i)} - \epsilon_j, u_{min})$ ;
        $\bar{u}_j^{(i)} = \min(\bar{u}^{(i)} + \epsilon_j, u_{max})$ ;
    end
    $\hat{u}_{\mathcal{R},l}^{(T_c+i)}$ = getRobotAction( $\{x\}_{i=0}^{T_c-1}$, $\{\hat{x}_l\}_{i=T_c}^{T_c+i}$ ) ;
    $\hat{u}_{\mathcal{R},h}^{(T_c+i)}$ = getRobotAction( $\{x\}_{i=0}^{T_c-1}$, $\{\hat{x}_h\}_{i=T_c}^{T_c+i}$ );
    $\hat{u}_{\mathcal{H},l}^{(T_c+i)}, \hat{u}_{\mathcal{H},h}^{(T_c+i)} = \underline{u}_1^{(i)}, \bar{u}_1^{(i)}$ ;
    for boundType $\in \{l, h\}$ do
        for agentType $\in \{\mathcal{H}, \mathcal{R}\}$ do
            $\hat{x}_{\text{agentType,boundType}}^{(T_c+i+1)}$ =ForwardDynamics( $\hat{x}_{\text{agentType,boundType}}^{(T_c+i)}, \hat{u}_{\text{agentType,boundType}}^{(i)}$) ;
        end
    end
end
```

Proof. First, observe that if $\forall j \in \{1, \ldots, F-1\}$, $\epsilon_j \leq \epsilon_{j+1}$, then at each time step $i \in \{0, \ldots, T-1\}$, $\underline{u}_{j+1}^{(i)} \leq \underline{u}_j^{(i)}$ and $\bar{u}_j^{(i)} \leq \bar{u}_{j+1}^{(i)}$. Equivalently, the control input sets at each time step satisfy $U_j^{(i)} = [\underline{u}_j^{(i)} \bar{u}_j^{(i)}] \subseteq [\underline{u}_{j+1}^{(i)} \bar{u}_{j+1}^{(i)}] = U_{j+1}^{(i)}$.

Then consider any two initial sets and input bounds for reachable sets $\mathcal{F}_a$, $\mathcal{F}_b$ that satisfy $\mathcal{L}_a \subseteq \mathcal{L}_b$ and $U_a \subseteq U_b$ between time $[t_s, t_e]$ where t_s and t_e indicate the start and end time respectively. If $x_a \in \mathcal{F}_a$, then by definition $\exists u(t) \in U_a$, s.t. $t \in [t_s, t_e]$, $x(\cdot)$ satisfies $\dot{x} = f(x, u), x(t_s) \in \mathcal{L}_a, x(t) = x_a$. Since $U_a \subseteq U_b$ and $\mathcal{L}_a \subseteq \mathcal{L}_b$, this means that $u(t) \in U_b$, $t \in [t_s, t_e]$, and $x(t_s) \in \mathcal{L}_b$. Hence by definition, $x_a \in \mathcal{F}_b$. This proves that $\mathcal{F}_a \subseteq \mathcal{F}_b$.

We let $\mathcal{F}_j(t_i)$ denote the reachable set j grown so far from time t_0 to time t_i. Using the above result, we have $\mathcal{F}_j(t_1) \subseteq \mathcal{F}_{j+1}(t_1)$ because $\mathcal{F}_j(t_0) = \mathcal{F}_{j+1}(t_0) = \mathcal{L}$ and $U_j^{(0)} \subseteq U_{j+1}^{(0)}$. Then applying this recursively, we have $\mathcal{F}_j(t_i) \subseteq \mathcal{F}_{j+1}(t_i)$ for any $i \in \{2, \ldots, T\}$. Hence $\mathcal{F}_j \subseteq \mathcal{F}_{j+1}$, $\forall j \in \{1, \ldots, F-1\}$. □

We now present how we compute $p_j, j \in \{1, \ldots, F\}$.

We compute p_j associated with $\mathcal{F}_j$ by letting it be the percentage of the human trajectories that fall entirely within the predicted $\mathcal{F}_j$.

Corollary 1. *Using our algorithm,* $p_{\mathcal{F}_{j+1}} \geq p_{\mathcal{F}_j}, \forall j \in \{1, \ldots, F-1\}$.

Proof. According to Theorem 1, we have that for any $j \in \{1, \ldots, F-1\}$, $\mathcal{F}_j \subseteq \mathcal{F}_{j+1}$. This suggests that if $\zeta_{k,l} \in \mathcal{F}_{j,k,l}$, then $\zeta_{k,l} \in \mathcal{F}_{j+1,k,l}$. Hence, $\forall k \in \{1, \ldots, K\}, \forall l \in \{0, \ldots, L_k - T - 1\}, \mathbb{1}\{\zeta_{k,l} \in \mathcal{F}_{j,k,l}\} \leq \mathbb{1}\{\zeta_{k,l} \in \mathcal{F}_{j+1,k,l}\}$. Hence, $p_{\mathcal{F}_j} \leq p_{\mathcal{F}_{j+1}}, \forall j \in \{1, \ldots, F-1\}$. □

Corollary 2. *If we set* ϵ_F *such that* $\underline{u}_F^{(i)} = u_{min}, \bar{u}_F^{(i)} = u_{max}, \forall i \in \{0, \ldots, T-1\}$, *then* $p_{\mathcal{F}_F} = 1$.

Proof. Since human inputs are constrained within $[u_{min}, u_{max}]$, any human trajectory must fall inside the forward reachable set where the bounds on the inputs are always $[u_{min}, u_{max}]$. Hence if $\underline{u}_F^{(i)} = u_{min}, \bar{u}_F^{(i)} = u_{max}$, then $\zeta_{k,l} \in \mathcal{F}_{F,k,l}, \forall k \in \{1, \ldots, K\}, \forall l \in \{0, \ldots, L_k - T - 1\}$. Thus, $p_{\mathcal{F}_F} = 1$. □

Having $\sum_{j=1}^{F} p_{\mathcal{S}_j} = p_{\mathcal{F}_F} = 1$ allows us to capture all possible future trajectories and provides us with the ability to use $\mathcal{F}_F$ as the most conservative safety standard.

3.3 Experiments

We conduct experiments to understand how incorporating information derived from HJ reachability affects the prediction of human behavior with our proposed method. Furthermore, we present the learned probabilistic human forward reachable sets (PHFRS) generated based on our proposed framework with the same experimental data.

3.3.1 Experimental design

We recruited 8 participants between the age 20-27 from the university campus to participate in the experiment. We gather trajectory data by having each human subject control a human vehicle in a simulated environment with another robot vehicle. The robot vehicle has a goal which is displayed to the human, and the robot is automatically controlled to reach its designated goal in exactly 10 seconds. The human subject is informed that the robot is heading straight to the goal and will not actively avoid the human vehicle.

In our experiment, we use vehicles with the following dynamics:

$$\begin{aligned} \dot{p}_{x,i} &= v \cos \psi_i \\ \dot{p}_{y,i} &= v \sin \psi_i \\ \dot{\psi}_i &= \omega_i, \quad \omega_{min} \leq \omega_i \leq \omega_{max} \end{aligned} \tag{3.2}$$

where the state variables $p_{x,i}, p_{x,i}, \psi_i$ represent the x position, y position, and heading of vehicle Q_i, $i \in \{\mathcal{H}, \mathcal{R}\}$. Each vehicle travels at a constant speed of $v = 2$, and its turn rate

ω_i is constrained by $\omega_{min} = -0.5, \omega_{max} = 0.5$. Using a computer keyboard, the participants can provide control inputs corresponding to $\{\omega_{min}, 0, \omega_{max}\}$, which are control inputs for turning right, going straight, and turning left respectively. We think that these controls are sufficient for the human participants to avoid the robot vehicle while not burdening the cognitive load of the humans by giving them an abundance of possible controls, which may confound the experimental results.

Two example scenes from our data collection web application are presented in the top two plots in Figure 3.1. The robot and human vehicle are colored in red and blue respectively. The tails and the directions of the arrows indicate the exact locations and orientations of the vehicles respectively. The goal of the robot is presented as a red square. The human operator is asked to provide control inputs so that the human vehicle avoids the danger zone of the robot vehicle, represented as $\mathcal{L}_{ij} = \{p_{x,ij} : (p_{x,i} - p_{x,j})^2 + (p_{y,i} - p_{y,j})^2 \leq R_c^2\}$, where $R_c = 3$ in our experiment. Each scene is also designed so that if the human subject doesn't avoid the robot, the human and robot vehicles will collide eventually. Hence the human subjects are instructed to continuously provide control inputs for avoidance starting from when they feel danger until they no longer think that the human vehicle and the robot vehicle will collide if they don't provide any more avoidance inputs. The human does not need to repeatedly press the key to avoid but can just hold down the key to avoid continuously. The scene ends when the robot reaches the goal.

The experiment is divided into three phases. In the first phase, we provide instructions and the participants are given 1 minute to familiarize themselves with the interface we developed and the dynamics of the vehicle. In the second phase, participants are given three practice scenes to further familiarize themselves with the setup of the study. In the third phase, participants are given 50 vehicle avoidance tasks. The initial states of the human and robot vehicles are randomized for each scene. We record the states of the human and robot vehicles, along with the action of the human, every 0.2 seconds.

3.3.2 Analysis of prediction performance

In this section, we present the experimental results of incorporating information derived from HJ reachability into the learning problem.

We perform a five fold cross validation to tune the hyperparameters of the models and evaluate the result based on predictive performance on the test data. We model that each human has a different pattern in avoidance behavior, hence, we trained a classifier for each subject. We perform an extensive comparison of incorporating different subsets of information derived from HJ reachability. To rigorously compare the sets of features, we conduct statistical significance tests to see if the differences in performance are statistically significant. We apply the Mann-Whitney test on pairs of feature sets and compute the p-values.

As described in 3.2.1, the standard features are $\mathcal{B} = \{|p_{x,\mathcal{HR}}|, |p_{y,\mathcal{HR}}|, \psi_{\mathcal{HR}}, \cos\psi_{\mathcal{HR}}, \sin\psi_{\mathcal{HR}}\}$. We augment these with features, $d_{\mathcal{HR}} = \sqrt{p_{x,\mathcal{HR}}^2 + p_{y,\mathcal{HR}}^2}$, $v_{\mathcal{H}} = V_{\mathcal{HR}}(x_{\mathcal{HR}})$, and $v_{\mathcal{R}} = V_{\mathcal{RH}}(x_{\mathcal{RH}})$ to the feature set, which represent the distance between the two vehicles, and the safety levels of the human relative to robot and robot relative to human, respectively. To obtain safety levels, we compute the BRS (2.4) with the relative dynamics of the two vehicles derived from the vehicle dynamics (3.2). To describe the sets of features, we use subscripts d, h, and r to represent the addition of the features to the standard feature set $d_{\mathcal{HR}}$, $v_{\mathcal{R}}$, and $v_{\mathcal{H}}$ respectively. For example, $\mathcal{B}_{hrd} = \mathcal{B} \cup \{v_{\mathcal{R}}, v_{\mathcal{H}}, d_{\mathcal{HR}}\}$ and $\mathcal{B}_{hr} = \mathcal{B} \cup \{v_{\mathcal{H}}, d_{\mathcal{HR}}\}$.

We conduct two sets of experiments: in experiment (I), we perform prediction on the exact control the human inputs, i.e., we predict the control as one of the three possible control inputs, $\{\omega_{min}, 0, \omega_{max}\}$. In experiment (II), the goal is to predict whether the human will input control to avoid at any joint state of the $Q_{\mathcal{H}}$ and $Q_{\mathcal{R}}$, which is equivalent to predicting whether the input is $\omega = 0$ or if the input falls in $\{\omega_{min}, \omega_{max}\}$ in our setup. We consider the following metric for both sets of experiments:

- Accuracy: $\frac{1}{\sum_{k=1}^{K} L_k} \sum_{k=1}^{K} \sum_{l=1}^{L_k} \mathbb{1}\{\hat{y}_k^{(l)} = y_k^{(l)}\} \times 100\%$ where $\hat{y}_k^{(l)}$ and $y_k^{(l)}$ represent the predicted action and the ground truth action of the human at time step l in trajectory k respectively. Here K represents the number of trajectories and L_k represents the number of time steps in trajectory k.

For experiment II, we further evaluate the following metrics:

- D_{start}: Let $\hat{T}_{k,f}$ be the first time step in trajectory k that our algorithm predicts the vehicle avoids and $T_{k,f}$ be the first time step the human avoids in the experiment. This metric is defined as: $\frac{1}{K} \sum_{k=1}^{K} \left| \hat{T}_{k,f} - T_{k,f} \right|$.

- D_{end}: Let $\hat{T}_{k,e}$ be the last time step in trajectory k that our algorithm predicts the vehicle avoids and $T_{k,e}$ be the last time step the human avoids in the experiment. This metric is defined as: $\frac{1}{K} \sum_{k=1}^{K} \left| \hat{T}_{k,e} - T_{k,e} \right|$.

We consider support vector machine (SVM), decision tree (DT), and logistic regression (LR) machine learning models. SVM and DT models don't make assumptions about the probabilistic distribution of the data. LR assumes that each data point is iid. Despite the fact that the data we gather are temporally correlated, we are interested in investigating how well LR performs on the data.

Our experimental results indicate that incorporating the safety levels derived from HJ reachability can yield considerable improvement in predictive performance. Table 3.1 illustrates the accuracies obtained by applying the SVM, DT, and LR models on the two sets of experiments, (I) and (II). We can see that for all models on both tasks, using the feature set $\mathcal{B}_{hrd}$ yields the highest performance, considerably higher than just incorporating distance, $\mathcal{B}_d$ ($p < 0.05$). It is also notable that for all three algorithms on both tasks, incorporating exactly one of the two safety levels $v_{\mathcal{H}}, v_{\mathcal{R}}$ outperforms incorporating just the distance $d_{\mathcal{HR}}$ ($p < 0.05$). We see that incorporating the robot's safety level with respect to the human, $v_{\mathcal{R}}$, generally, but not always, yields higher performance than if not including it. This suggests that taking into account the safety from the perspective of the robot vehicle is also important. With the feature set and model fixed, the accuracies in experiment (II) is in general higher than those in experiment (I). This is expected as in experiment (II), we need to also predict the direction of the avoidance. However, the difference is not big, suggesting that the algorithms did reasonably well in predicting avoidance direction.

Table 3.2 illustrates the performance of the models on predicting the first and last time steps of avoidance. We can see from the table that using DT with feature set $\mathcal{B}_{hrd}$ yields the best performance in metric D_s and is on average 1.48 time steps off from the ground truth, which is equivalent to $1.48 \times 0.2 = 0.296$ seconds off since every time step is 0.2 second apart. Similar to the accuracy metric, incorporating just one of the safety levels yields a more accurate prediction than just considering $d_{\mathcal{HR}}$ ($p < 0.05$). We can also see that including the feature $v_{\mathcal{R}}$ generally result in better prediction than including the feature $v_{\mathcal{H}}$. It is also interesting to see that the models did a much better job predicting when a human would start avoiding than when a human would end avoiding. We hypothesize that this is because humans are generally more noisy in determining when to stop avoiding after there is no longer immediate danger as once the danger is cleared, when to stop avoiding is less important.

	$\mathcal{B}_{hrd}$	$\mathcal{B}_{hr}$	$\mathcal{B}_{hd}$	$\mathcal{B}_{rd}$	$\mathcal{B}_{h}$	$\mathcal{B}_{r}$	$\mathcal{B}_{d}$
SVM (I)	**78.67**	78.46	76.56	76.74	75.11	76.33	69.75
DT (I)	**75.77**	74.24	73.65	75.27	71.37	73.65	68.68
LR (I)	**77.73**	77.4	74.71	77.38	73.31	76.61	69.15
SVM (II)	**83.89**	83.15	81.12	82.85	79.62	82.15	70.72
DT (II)	**81.29**	78.58	79.26	80.55	79.15	78.5	68
LR (II)	**81.70**	81.61	78.79	81.62	77.54	80.88	71.35

Table 3.1: This table shows the accuracies of SVM, DT, and LR models using different feature sets. We can see that including the information derived from HJ reachability yields improvement in predictive performance than just including distance as a feature.

Intuitively, we think the higher performance of including safety levels derived from HJ reachability is that inherently the safety levels encode some information about the dynamics and how dangerous the configuration is based on worst case analysis. The geometric distance of the two vehicles can also encode useful information about how dangerous the configuration might be, however, it's not as informative as the safety levels.

3.3.3 Probabilistic human forward reachable set (PHFRS) implementation

In this section, we demonstrate an implementation of our proposed forward reachable set prediction framework in 3.2.2 by applying our framework on the experimental data gathered.

	$\mathcal{B}_{hrd}$	$\mathcal{B}_{hr}$	$\mathcal{B}_{hd}$	$\mathcal{B}_{rd}$	$\mathcal{B}_{h}$	$\mathcal{B}_{r}$	$\mathcal{B}_{d}$
SVM, D_s	2.02	2.06	2.23	**1.88**	2.3	2.03	4.22
DT, D_s	**1.48**	2.02	1.98	1.56	2.44	1.68	2.92
LR, D_s	2.43	2.84	2.6	**2.29**	2.93	2.83	7.3
SVM, D_e	**6.7**	7.38	8.67	7.03	8.78	8.07	15.94
DT, D_e	8.77	9.09	8.43	**8.02**	9.45	9.25	14.5
LR, D_e	7.67	**7.35**	9.17	7.72	9.96	8.09	17.57

Table 3.2: This table shows results for the metrics D_{start} (D_s) and D_{end} (D_e). Similarly to the accuracy metric, we see that both safety levels derived from HJ reachability improve prediction performance.

For demonstration purpose, we use the logistic regression predictor learned in experiment I in 3.3.2 for prediction, although it is possible to use *any* predictor that gives probabilistic information on the predicted actions. Note that the constraint $p_{\mathcal{F}_1} \geq \underline{p}$ can always be satisfied by tuning ϵ_1's and $k^{(i)}$'s. The intuition is that the better the predictor we use in Algorithm 3.1, the smaller ϵ_1 and $k^{(i)}$'s are needed to achieve $p_{\mathcal{F}_1} \geq \underline{p}$. An example of the PHFRS using an implementation of this framework is illustrated in Figure 3.2. We can see that the probabilistic reachable set we produce gives varying degrees of safety probabilities.

To provide intuition on the effect of tuning ϵ_j's and $k^{(i)}$'s during the optimization, we demonstrate the effect of changing these parameters. The leftmost figure of Figure 3.2 shows the probabilistic reachable set generated with $\epsilon_1 = 0, \epsilon_2 = 0.15, \epsilon_3 = 0.25, \epsilon_4 = 0.4, \epsilon_5 = 1.0$ and $k^{(i)} = 2, i \in \{0, 1\}$ and $k^{(i)} = 1, i \in \{2, \dots, 9\}$. The PHFRS in the middle figure is generated by fixing the $k^{(i)}$'s used in the left figure and varying the ϵ_j's. We let $\epsilon_1 = 0.2$ $\epsilon_2 = 0.3, \epsilon_3 = 0.35, \epsilon_4 = 0.45$ and leave ϵ_5 unchanged. Increasing ϵ_j makes the region $\mathcal{F}_j$ larger. On the other hand, the PHFRS in rightmost figure is generated by fixing the ϵ_j's used in generating the PHFRS in the leftmost figure and varying the $k^{(i)}$'s. We let $k^{(i)} = 2, i = \{0\}$ and $k^{(i)} = 1, i \in \{1, \dots, 9\}$ in the rightmost figure. Making $k^{(i)}$ smaller decreases the area of $\mathcal{F}_j$'s, except for $\mathcal{F}_F$, which aims to capture the worst case scenario.

Algorithm 3.1 can be computed efficiently online. The generation of PHFRS based on the output of Algorithm 3.1 can be computed online if the mapping from any $\{[\underline{u}_j^{(i)}, \bar{u}_j^{(i)}]\}_{i=0}^{T-1}$ to $\mathcal{F}_j$ has been computed offline.

3.4 Conclusion and Future Work

We first demonstrate how to incorporate information derived from HJ reachability into a machine learning problem and show that this can yield considerable improvement in prediction performance of human behavior compared with using one of the most typical features, the distance feature, in safety critical scenarios. We then propose a framework to generate probabilistic human forward reachable set (PHFRS) that flexibly offers different levels of safety probabilities and generalizes to unseen scenarios. In future work, it would be interesting to develop methodologies to evaluate how useful safety levels from HJ reachability are for continuous action prediction or under situations where temporal information are considered. Other directions include further imposing metrics on the PHFRS framework and generalizing the framework to work with continuous action prediction.

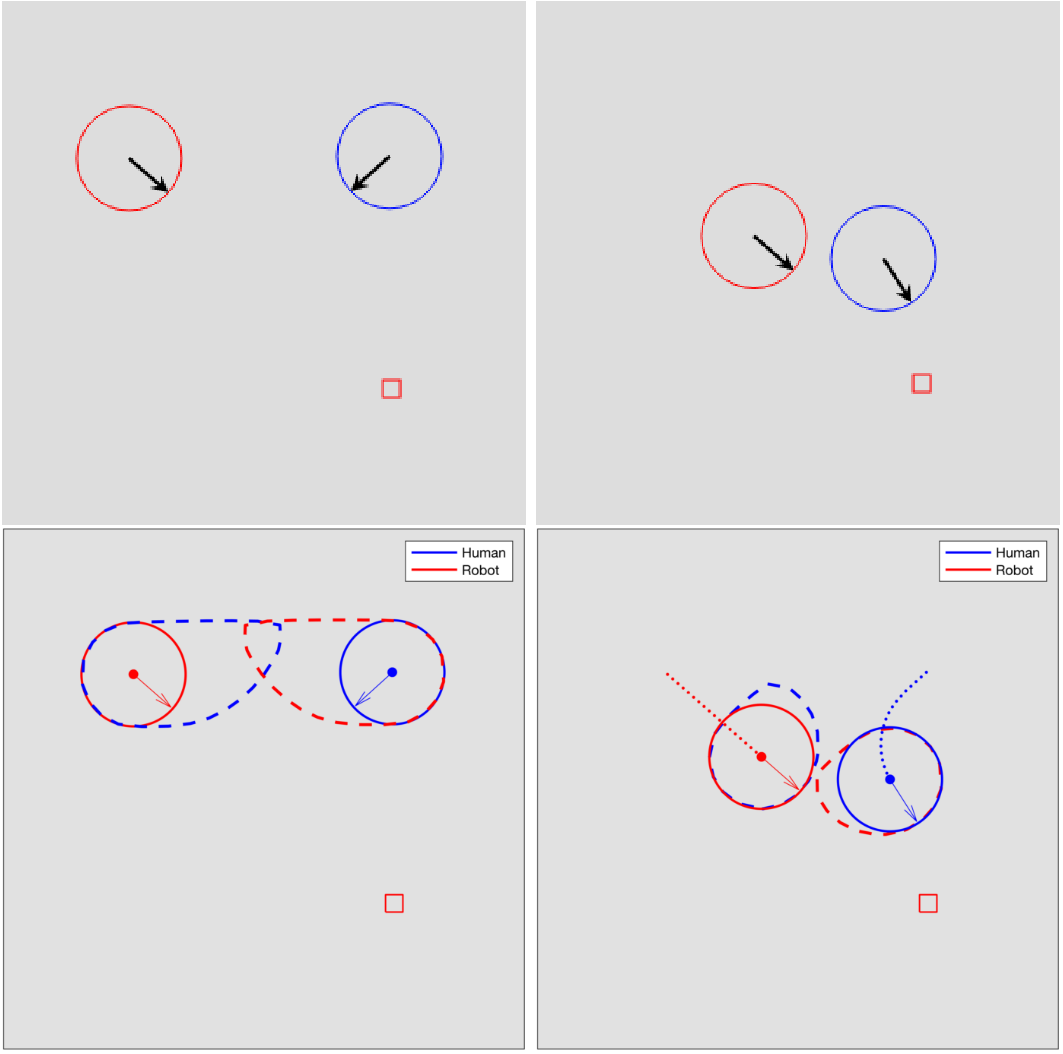


Figure 3.1: The top two figures illustrate the interface we designed to collect data from participants. The top left figure shows the initial configuration and the top right figure shows the configuration after the human has inputted controls to avoid the robot. The bottom figures illustrate the sub-zero level set for the value functions $V_{\mathcal{HR}}$ and $V_{\mathcal{RH}}$ for the configurations in the top figures, computed using [62]. Neither of the safety value functions are provided to the human subjects during the experiment: the subjects only see the scenes in the top figures.

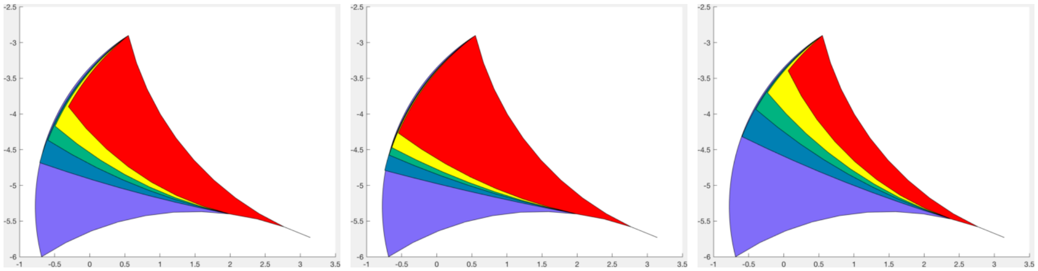


Figure 3.2: These are probabilistic human forward reachable sets (PHFRS) generated using our framework. The algorithm predicts that the human will likely turn right within the next $T = 10$ time steps. The regions in red, yellow, green, blue, purple correspond to $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5$ respectively. The middle figure is generated with the same $k^{(i)}$'s as the leftmost figure but the ϵ_j's used are larger or equal to those used in the leftmost figure. Increasing ϵ_j increases the area of $\mathcal{F}_j$. The rightmost figure is generated with the same ϵ_j's as the leftmost figure, but with $k^{(i)}$'s smaller or equal to those in the leftmost figure. We can see that decreasing $k^{(i)}$'s makes the areas of $\mathcal{F}_j$'s smaller. For the PHFRS in the middle figure, the probability for each of the five regions is: $p_{\mathcal{F}_1} = p_{\mathcal{S}_1} = 0.752, p_{\mathcal{F}_2} = p_{\mathcal{S}_1 \cup \mathcal{S}_2} = 0.771, p_{\mathcal{F}_3} = p_{\cup_{i=1}^{3} \mathcal{S}_i} = 0.773, p_{\mathcal{F}_4} = p_{\cup_{i=1}^{4} \mathcal{S}_i} = 0.775, p_{\mathcal{F}_5} = p_{\cup_{i=1}^{5} \mathcal{S}_i} = 1.0$.

Part II

Safety for Multi-vehicle Systems

Chapter 4

Reachability-based Safe Planning for Multi-Vehicle Systems with Multiple Targets

The research presented in this chapter was originally published in the paper Reachability-based Safe Planning for Multi-Vehicle Systems with Multiple Targets in American Control Conference 2021 [71].

4.1 Introduction and Related Work

In recent years, there have been vast interests from commercial companies to government agencies in introducing unmanned aerial vehicles (UAVs) into the airspace. For example, Google X [47], Amazon [5], and UPS [78] have all been developing drone technology for goods transport and delivery. Companies such as Zipline Inc. [86] and Vayu Inc. [79] utilize drones for delivery of critical medical supplies. The government is also tapping into UAVs for disaster response [38], [52], [9] and military operations [19]. With the burgeoning enthusiasm for this emerging technology, the Federal Aviation Administration recently devised guidelines specifically for UAVs [40]. Ensuring the safety of UAVs is thus an imminent and highly impactful problem. A central problem in UAVs is to have them visit multiple targets for purposes such as delivery of supplies or inspection at different locations. Thus the problem of efficiently enabling all vehicles to accomplish their objectives of visiting multiple targets while maintaining safety at all times is of paramount importance.

The problem of collision avoidance among multi-agent systems has been studied through various methods. For example, [41, 15] assume that vehicles employ specific simple control strategies to induce velocity obstacles that must be avoided by other vehicles to maintain safety. There have also been approaches that use potential functions to tackle safety while multiple agents travel along pre-determined trajectories [66, 32]. While these approaches offer insights into tackling multi-agent problems, they do not offer the safety guarantees that are highly desirable for safety-critical systems with general dynamical systems.

Differential game concerns the model and analysis of conflicts in dynamical systems and is a promising tool for safety-critical problems for multi-vehicle systems due to the strong theoretical guarantees it can provide. One such technique is Hamilton-Jacobi (HJ) reachability [63]. HJ reachability has been successfully used to guarantee safety for small-scale problems that concern one or two vehicles [45, 63]. Despite its favorable theoretical guarantees and applicability to systems with general dynamics, it suffers from the curse of dimensionality because the computation of reachable sets grows exponentially with the number of states in the system and hence the number of vehicles, making its direct application to systems of more than two vehicles intractable.

There have been many attempts in using differential games to analyze three-player differential games with varying-degree of assumptions on each agent in non-cooperative settings [75, 73, 44]. [23] is the first work built on reachability that guarantees safety for three vehicles while vehicles are allowed to execute *any* control when the safe controller does not need to be applied, which endows vehicles more flexibility and is thus preferable in certain scenarios. [35] further builds on [23] to guarantee safety for four vehicles in unstructured settings. However, [35] assumes that vehicles can remove themselves from the environment when conflicts cannot be resolved for all vehicles, which is not always possible and could be undesirable in some situations. In contrast, we propose a control strategy to guarantee safety for four and more number of vehicles without assuming the ability to remove *any* vehicle during conflict resolution.

Works such as [26, 25] have proposed controllers that guarantee safety for larger number of vehicles by imposing varying degrees of structure on the vehicles, including strong assumptions such as vehicles traveling in a single line of platoon [25] or vehicles determining their trajectories *a priori* [26]. In general, there is a trade-off between the number of vehicles safety can be guaranteed for and how strong the assumption on the structure of the multi-vehicle system is. In this paper, we provide a novel approach based on reachability that guarantees safety for *any* number of vehicles by using less structure than those of [26, 25] for a class of dynamical systems. Although our proposed method adopts more structure than that of [23] and [35], our approach can guarantee safety for *any* number of vehicles while avoiding having to remove vehicles from the environment when conflict cannot be resolved and retaining some level of unstructuredness.

Our main contribution is a novel approach to guarantee safety while *any* number of vehicles are tasked with visiting multiple targets for a class of dynamical systems. We first propose a method that assigns vehicles into "teams" and induces the behavior that vehicles with similar objectives are assigned to the same team for efficiency. We then propose a control strategy to guarantee safety for any pair of vehicles within a team and across different teams, effectively guaranteeing safety for all vehicles.

The paper is organized as follows:

- In Section 4.2, we formulate the multi-vehicle collision avoidance and multi-target satisfaction problem.

- In Section 4.3, we present our approach of assigning vehicles into different teams based on their objectives for efficient task completion and our strategy for ensuring safety for all vehicles.

- In Section 4.4, we demonstrate our proposed method in a four-vehicle and a fifteen-vehicle simulation, illustrating the effectiveness of both our proposed team assignment algorithm and proposed safety strategy.

4.2 Problem Formulation

Consider N vehicles, denoted $Q_i, i = 1, 2, \dots, N$, with identical dynamics described by the following ordinary differential equation (ODE)

$$\dot{x}_i = f(x_i, u_i), \quad u_i \in \mathcal{U}, \quad i = 1, \dots, N \tag{4.1}$$

where $x_i \in \mathbb{R}^n$ is the state of the ith vehicle Q_i, and u_i is the control of Q_i. In this paper, we work with a class of dynamical systems such that the dynamics f can be described completely by a subset of the state and the control input, i.e., we can write $x_i = [x_{i,a}\ x_{i,b}]$ where $x_{i,a} \in \mathbb{R}^{n_a}, x_{i,b} \in \mathbb{R}^{n_b}$, $n_a \geq 1, n_b \geq 0$, such that

$$\dot{x}_i = f(x_i, u_i) = f_b(x_{i,b}, u_i), \quad u_i \in \mathcal{U}, \quad i = 1, \dots, N \tag{4.2}$$

for some function f_b. Note that we will use the subscript ”a” or ”b” to denote the components of a given state based on the definition above throughout the paper.

Each of the N vehicles is tasked with visiting a set of targets $\mathcal{G}_i$, in no particular order, out of a set of M targets $\{\mathcal{T}_1, \dots, \mathcal{T}_M\}$, i.e., $\mathcal{G}_i \subseteq \{\mathcal{T}_1, \dots, \mathcal{T}_M\}$. Note that the exact location of each target need not to be known *a priori*. Each vehicle Q_i must reach all of its targets while at all times avoid the *danger zone* $\mathcal{Z}_{ij}$ with respect to any other vehicle $Q_j, j = 1, \dots, N, j \neq i$. The danger zone $\mathcal{Z}_{ij}$ represents relative configuration between Q_i and Q_j that are considered undesirable, such as collision. In this work, we assume the danger zone $\mathcal{Z}_{ij}$ for each pair of vehicles can be identically defined by a *norm* function on the $x_{ij,a}$ component of the relative state x_{ij}, $d(x_{ij,a}) : \mathbb{R}^{n_a} \to \mathcal{R}^+$ where $x_{ij,a} \equiv x_{i,a} - x_{j,a}$. In particular, the danger zone $\mathcal{Z}_{ij}$ is defined such that $x_{ij} \in \mathcal{Z}_{ij} \Leftrightarrow d(x_{ij,a}) \leq R_{ij}$ where R_{ij} is some positive real number. Note that in this work, we assume $R_{ij} = R_{ji}$ for any pair of vehicles Q_i, Q_j.

Here we note that, in this work, we use a slight abuse of notation to represent $x_i - x_j$ as x_{ij}. This is in general different from the x_{ij} used in defining relative dynamics and the danger zone in subsection 2.1.3 on backward reachable sets in the Background chapter. We make this distinction because sometimes the relative dynamics between two vehicles cannot be directly defined in closed form with the relative state when the relative state is defined directly as $x_i - x_j$. We instead use the notation $\bar{x}_{ij}$ to represent a relative state representation between x_i and x_j that doesn't necessarily have to be $x_i - x_j$. We assume there is a bijection between x_{ij} and $\bar{x}_{ij}$. In this work specifically, similarly, we use the notation $\mathcal{Z}$ to denote the danger zone defined using relative state x_{ij} and the notation $\bar{\mathcal{Z}}$ to denote danger zone defined based on $\bar{x}_{ij}$. The reachability computation is done based on danger zone $\bar{\mathcal{Z}}$ and relative dynamics defined based on $\bar{x}_{ij}$.

Remark 1. *Many practical and common dynamical systems have dynamics structures outlined in Equation (4.2), such as the 2D point system [67], 3D Dubins Car [23], 6D Quadrotor [67], 6D Acrobatic Quadrotor [46], 7D Quadrotor [67], and 10D near-hover quadrotor [18]. In addition, for all these dynamical systems, defining the danger zone based on the x_a component of the state x makes intuitive sense as the x_a components represent the x, y, z translational coordinates of these systems, which is what we generally use to define collisions among vehicles.*

Given the vehicle dynamics in (5.1), the derived relative dynamics in (2.3), the danger zones $\mathcal{Z}_{ij}, i, j = 1, \dots, N, i \neq j$, and the sets of targets each vehicle Q_i needs to go through $\mathcal{G}_i, i = 1, \dots, N$, we propose a cooperative planning and control strategy that:

1. assigns vehicles to clusters (teams) based on their objectives;

2. determines the initial states of all vehicles;

3. guarantees safety for all vehicles for all time.

Remark 2. *In this work, we will use the terms "cluster" and "team" interchangeably.*

Our proposed method guarantees that all vehicles will be able to stay out of the danger zone with respect to any other vehicle regardless of the number of vehicles N in the environment. Additionally our method guarantees safety for all vehicles without vehicles having to remove themselves from the environment when conflicts cannot be resolved, as assumed in [35]. For all initial configurations, target locations, and objectives of each vehicle in our simulations, all vehicles also complete their objectives of visiting all their targets successfully.

4.3 Methodology

Our proposed method consists of two phases: first, we develop the notion of *teams (clusters)* of vehicles and present a method to assign vehicles to teams based on their targets, with the goal of minimizing the time it takes for all vehicles to complete their objectives. Second, we propose the idea of *augmented* danger zone for each pair of teams. Based on this, we propose a control strategy to ensure safety for any pair vehicles on the same team and across different teams, which in combination guarantees safety for all vehicles.

4.3.1 Assignment of vehicles to clusters

We first propose an optimization problem that assigns the N vehicles to K teams, $\mathcal{H}_1, \dots, \mathcal{H}_K$. Each vehicle should be assigned to exactly one cluster and the objective of each cluster is then to visit, in no particular order, the union of the sets of targets of the vehicles in this cluster. Since we aim to have our approach be applicable to scenarios where the location of each target is not known *a priori*, we assume that the amount of time a cluster takes to complete its objective is proportional to the number of targets each cluster needs to visit and we don't consider the order in which each cluster visits its targets during the planning process in this paper. With this in mind, we formulate the objective function of the proposed optimization problem to minimize the maximum number of targets each cluster needs to visit, which load-balances the number of targets each cluster should visit by grouping vehicles with similar objectives into the same cluster. Furthermore, we show that the proposed optimization problem can be converted into an integer linear program and thus solved efficiently with standard integer program solvers.

Recall that each vehicle Q_i's objective is to visit a set of targets $\mathcal{G}_i$ where $\mathcal{G}_i \subseteq \{\mathcal{T}_1, \dots, \mathcal{T}_M\}$. Based on this, we define binary variables e_{ij}, $i \in \{1, \dots, N\}, j \in \{1, \dots, M\}$, such that $e_{ij} = \mathbb{1}\{\mathcal{T}_j \in \mathcal{G}_i\}$ [1]. Next we define optimization variables y_{ik}, $i \in \{1, \dots, N\}, k \in \{1, \dots, K\}$, which are also binary variables. $y_{ik} = 1$ means that vehicle Q_i is assigned to cluster $\mathcal{H}_k$, and $y_{ik} = 0$ otherwise. Based on the goal of minimizing the maximum number of targets each cluster needs to visit as described in the previous paragraph, we propose the following optimization problem to solve for y_{ik}'s:

$$\begin{aligned} \min_{y_{ik}} \quad & \max_k \left(\sum_{j=1}^{M} \underset{i}{max} \{e_{ij} y_{ik}\} \right) \\ \text{subject to} \quad & \sum_{k=1}^{K} y_{ik} = 1, \forall i \in \{1, \dots, N\} \\ & y_{ik} \in \{0, 1\}, \forall i \in \{1, \dots, N\}, \forall k \in \{1, \dots, K\} \end{aligned} \tag{4.3}$$

Note that due to space constraints under the min, max notations in the objective, we omit that we're optimizing over $y_{ik}, \forall i \in \{1, \dots, N\}, \forall k \in \{1, \dots, K\}$ for the minimization and $k, \forall k \in \{1, \dots, K\}$ and $i, \forall i \in \{1, \dots, N\}$ for the maximization in the above optimization problem.

The summation $\sum_{j=1}^{M} \underset{i}{max}\{e_{ij} y_{ik}\}$ is equivalent to the total number of targets that cluster $\mathcal{H}_k$ needs to visit. To see this, for a given target $\mathcal{T}_j$, the term $e_{ij} y_{ik}$ in the summation equals to 1 if vehicle Q_i needs to visit target $\mathcal{T}_j$ *and* Q_i is assigned to cluster $\mathcal{H}_k$. $e_{ij} y_{ik} = 0$ otherwise. Hence $\underset{i}{max}\{e_{ij} y_{ik}\}$ equals to 1 if at least one vehicle assigned to cluster $\mathcal{H}_k$ needs to visit target $\mathcal{T}_j$. $\underset{i}{max}\{e_{ij} y_{ik}\} = 0$ otherwise. Summing $\underset{i}{max}\{e_{ij} y_{ik}\}$ over all targets gives the total number of targets cluster $\mathcal{H}_k$ needs to visit.

[1] $\mathbb{1}(\mathcal{A})$ is an indicator function on event $\mathcal{A}$ such that $\mathbb{1}(\mathcal{A}) = 1$ if $\mathcal{A}$ is true and $\mathbb{1}(\mathcal{A}) = 0$ otherwise.

Next we show that the optimization problem (4.3) can be converted into a standard integer linear program by introducing a slack variable and an inequality constraint for each of the maximization operations in the objective.

$$\begin{aligned}
\min_{y_{ik}, o_{kj}, O} \quad & O \\
\text{subject to} \quad & \sum_{k=1}^{K} y_{ik} = 1, \forall i \in \{1, \dots, N\} \\
& y_{ik} \in \{0, 1\}, \forall i \in \{1, \dots, N\}, \forall k \in \{1, \dots, K\} \\
& e_{ij} y_{ik} \leq o_{kj}, \forall i \in \{1, \dots, N\}, \forall k \in \{1, \dots, K\}, \\
& \forall j \in \{1, \dots, M\} \\
& \sum_{j=1}^{M} o_{kj} \leq O, \forall k \in \{1, \dots, K\}.
\end{aligned}$$

The above integer linear problem can be solved efficiently by off-the-shelf integer program solvers. Once solved, the values of y_{ik}'s are the solution to the team assignment problem. This completes the first step of the planning process.

4.3.2 Collision Avoidance Protocol Design

In this section, we present our proposed control strategy that ensure all vehicles remain safe when completing their objectives after the vehicles have been assigned to teams. Specifically, given any K-vehicle collision avoidance algorithm that guarantees safety when resolving potential conflicts among K vehicles, we propose a general way to initialize vehicle locations and a safe control strategy such that the following always hold for N vehicles where N can be much larger than K:

- Any vehicle is safe from any other vehicle within the same cluster.
- Any vehicle in a cluster is safe from any vehicle in any other cluster.

4.3.2.1 Guaranteed safety for all vehicles within the same cluster

We first prove a theorem that motivates the control strategy that enables any pair of vehicles in the same cluster to remain safe from each other.

Theorem 2. *Give the structure of the dynamics and the danger zone defined in Section 4.2, for any two vehicles Q_i and Q_j, if the initial states $x_i(t_0), x_j(t_0)$ of the two vehicles satisfy $d(x_{ij,a}(t_0)) > R_{ij}$ and $x_{i,b}(t_0) = x_{j,b}(t_0)$ and the controls of the vehicles satisfy $u_i(t) = u_j(t)$ $\forall t \geq t_0$, then vehicles Q_i and Q_j will remain safe from each other for all $t > t_0$.*

Proof. Given that $x_{i,b}(t_0) = x_{j,b}(t_0)$ and $u_i(t) = u_j(t)\ \forall t \geq t_0$, we have that at any time $t \geq t_0$, $\dot{x}_i(t) = f(x_i(t), u_i(t)) = f_b(x_{i,b}(t), u_i(t)) = f_b(x_{j,b}(t), u_j(t)) = f(x_j(t), u_j(t)) = \dot{x}_j(t)$. Because $\dot{x}_{ij,a}(t) = 0\ \forall t \geq t_0$, $x_{ij,a}(t) = x_{ij,b}(t_0)\ \forall t > t_0$. Thus, $d(x_{ij,a}(t)) = d(x_{ij,a}(t_0)) > R_{ij}\ \forall t > t_0$. Since $R_{ij} = R_{ji}$ and $x_{ij}(t) = -x_{ji}(t)$, we have $d(x_{ij,a}(t)) = d(x_{ji,a}(t)) > R_{ij} = R_{ji}\ \forall t \geq t_0$, which proves that Q_i and Q_j will remain safe from each other for all $t > t_0$. □

The above shows that if we initialize any pair of vehicles Q_i, Q_j in the same cluster such that $x_{i,b}(t_0) = x_{j,b}(t_0)$, vehicles Q_i, Q_j start out safe from each other, and that they employ the same control at any time, the two vehicles will continue to remain outside of each other's danger zone for all time. We can directly use this insight to initialize all vehicles in the same cluster such that any pair of vehicles in the same cluster satisfies the above conditions and have all vehicles in the same cluster employ the same control to guarantee safety for all vehicles in the same cluster for all time.

4.3.2.2 Guaranteed safety of any vehicle with respect to any other vehicle in a *different* cluster

The key idea of our proposed method is that we can think of each cluster $\mathcal{H}_k$ as an *imaginary* vehicle with state $x_{\mathcal{H}_k}$ and dynamics identical to that of the individual vehicle's dynamics. We propose the concept of *augmented* danger zone between any pair of clusters, which allows us to guarantee that any vehicle in a cluster will remain safe from any vehicle in any other cluster.

Before we proceed to describe our approach, we first define a few essential terms:

Definition 3. ***Maximum vehicle distance to cluster center*** *for cluster* $\mathcal{H}_k$ *is defined as* $R_{\mathcal{H}_k} \equiv \underset{i:Q_i \in \mathcal{H}_k}{max}\ d(x_{\mathcal{H}_k,a} - x_{i,a})$ *where* $x_{\mathcal{H}_k}$ *is the state of the imaginary vehicle representing cluster* $\mathcal{H}_k$*.*

Definition 4. ***Augmented danger zone*** $\mathcal{Z}_{\mathcal{H}_k\mathcal{H}_l}$ *of cluster* $\mathcal{H}_k$ *with respect to* $\mathcal{H}_l$ *is defined as* $x_{\mathcal{H}_k\mathcal{H}_l} \in \mathcal{Z}_{\mathcal{H}_k\mathcal{H}_l} \Leftrightarrow d(x_{\mathcal{H}_k\mathcal{H}_l,a}) \leq R_{\mathcal{H}_k\mathcal{H}_l}$ *where* $x_{\mathcal{H}_k\mathcal{H}_l} = x_{\mathcal{H}_k} - x_{\mathcal{H}_l}$ *and* $R_{\mathcal{H}_k\mathcal{H}_l} = R_{\mathcal{H}_k} + R_{\mathcal{H}_l} + \underset{Q_i \in \mathcal{H}_k, Q_j \in \mathcal{H}_l}{max} R_{ij}$*.*

Definition 5. ***Safety level*** *of cluster* $\mathcal{H}_k$ *with respect to* $\mathcal{H}_l$ *is defined as* $s_{\mathcal{H}_k\mathcal{H}_l} \equiv V_{\mathcal{H}_k\mathcal{H}_l}(\bar{x}_{\mathcal{H}_k\mathcal{H}_l})$ *where* $V_{\mathcal{H}_k\mathcal{H}_l}(\bar{x}_{\mathcal{H}_k\mathcal{H}_l})$ *is computed based on reachability computation described in subsection 2.1.3 with dynamics identical to that of the vehicle dynamics and danger zone* $\bar{\mathcal{Z}}_{\mathcal{H}_k\mathcal{H}_l}$*.*

Now we prove a result that relates the danger zone of the *imaginary* vehicles representing the clusters and the danger zone of the actual vehicles.

Theorem 3. *If* $x_{\mathcal{H}_k\mathcal{H}_l} \notin \mathcal{Z}_{\mathcal{H}_k\mathcal{H}_l}$*, then* $x_{ij} \notin \mathcal{Z}_{ij}$ *for any pair of vehicles* Q_i, Q_j *such that* $Q_i \in \mathcal{H}_k$ *and* $Q_j \in \mathcal{H}_l$*.*

Proof. Let $r_{\mathcal{H}_k} = x_{\mathcal{H}_k} - x_i$ and $r_{\mathcal{H}_l} = x_{\mathcal{H}_l} - x_j$. Based on the definition of the augmented danger zone $\mathcal{Z}_{\mathcal{H}_k\mathcal{H}_l}$, we have $x_{\mathcal{H}_k\mathcal{H}_l} \notin \mathcal{Z}_{\mathcal{H}_k,\mathcal{H}_l} \Leftrightarrow d(x_{\mathcal{H}_k\mathcal{H}_l,a}) > R_{\mathcal{H}_k\mathcal{H}_l}$. With this in mind, we have

$$\begin{aligned} d(x_{\mathcal{H}_k\mathcal{H}_l,a}) &= d(x_{\mathcal{H}_k,a} - x_{\mathcal{H}_l,a}) \\ &= d(x_{i,a} + r_{\mathcal{H}_k,a} - x_{j,a} - r_{\mathcal{H}_l,a}) \\ &\le d(x_{i,a} - x_{j,a}) + d(r_{\mathcal{H}_k,a}) + d(r_{\mathcal{H}_l,a}) \\ &\le d(x_{ij,a}) + R_{\mathcal{H}_k} + R_{\mathcal{H}_l} \end{aligned}$$

where the first inequality follows from the triangle inequality on norms and the second inequality follows from the definitions of $R_{\mathcal{H}_k}$ and $R_{\mathcal{H}_l}$. Hence we have

$$\begin{aligned} d(x_{ij,a}) + R_{\mathcal{H}_k} + R_{\mathcal{H}_l} &\ge d(x_{\mathcal{H}_k\mathcal{H}_l,a}) \\ &> R_{\mathcal{H}_k\mathcal{H}_l} \\ &= R_{\mathcal{H}_k} + R_{\mathcal{H}_l} + \underset{Q_i \in \mathcal{H}_k, Q_j \in \mathcal{H}_l}{max} R_{ij} \\ &\ge R_{\mathcal{H}_k} + R_{\mathcal{H}_l} + R_{ij}. \end{aligned}$$

Subtracting $R_{\mathcal{H}_k} + R_{\mathcal{H}_l}$ from both sides results in $d(x_{ij,a}) > R_{ij}$, which implies $x_{ij} \notin \mathcal{Z}_{ij}$, as desired. □

Corollary 3. *Suppose at time $t = t_0$, for any cluster $\mathcal{H}_k$, $x_{\mathcal{H}_k,b}(t_0) = x_{i,b}(t_0)$ for all i such that $Q_i \in \mathcal{H}_k$. We apply the K-vehicle collision avoidance strategy that guarantees safety on the K imaginary vehicles representing the K clusters when resolving potential conflicts. If the strategy suggests to apply $u^\star_{\mathcal{H}_k}$ to the imaginary vehicle representing cluster $\mathcal{H}_k$, then in addition to applying this control on the imaginary vehicle, we also apply this control to all vehicles in this cluster. Given the aforementioned assumptions and the control strategy, if at time $t = t_0$, any pair of imaginary vehicles representing two distinct clusters $\mathcal{H}_k, \mathcal{H}_l$ are not in potential conflict with each other, for any pair of vehicles $Q_i \in \mathcal{H}_k, Q_j \in \mathcal{H}_l$, Q_i will remain safe from Q_j for all time $t \ge t_0$.*

Proof. First we note that it is only possible to have the same or less number of vehicles in a cluster as time proceeds because a vehicle is allowed to stay at its final target once it completes visiting all its targets. In addition, the same control is applied to all vehicles in any cluster $\mathcal{H}_k$ and the imaginary vehicle representing $\mathcal{H}_k$. Thus the maximum vehicle distance to cluster center $R_{\mathcal{H}_k}$ for each cluster $\mathcal{H}_k$ is non-increasing throughout execution, which means that the radius $R_{\mathcal{H}_k\mathcal{H}_l}$ defining the *augmented* danger zone between any two distinct clusters $\mathcal{H}_k, \mathcal{H}_l$ is non-increasing. By applying the K-vehicle collision avoidance control strategy on the *imaginary* vehicles representing the K clusters, we know that for any distinct clusters $\mathcal{H}_k, \mathcal{H}_l$, we have $x_{\mathcal{H}_k\mathcal{H}_l} \notin \mathcal{Z}_{\mathcal{H}_k\mathcal{H}_l}$ for all $t \ge t_0$ under the assumption that they are not in potential conflict initially. Applying Theorem 3, we have that $x_{ij} \notin \mathcal{Z}_{ij}$ for any vehicle $Q_i \in \mathcal{H}_k, Q_j \in \mathcal{H}_l$, which implies that any vehicle with respect to any vehicle in another cluster will remain safe from each other for all $t \ge t_0$. □

With the above in mind, we summarize our proposed overall initialization and cooperative control strategy for all vehicles to visit all their targets safely for all time:

- (1) Initialize all vehicles such that for any pair of vehicles Q_i, Q_j in the same cluster $\mathcal{H}_k$, $x_{ij} \notin \mathcal{Z}_{ij}$ and $x_{\mathcal{H}_k,b}(t_0) = x_{i,b}(t_0) = x_{j,b}(t_0)$. Additionally, any two distinct clusters $\mathcal{H}_k$ and $\mathcal{H}_l$ are initialized so that the imaginary vehicles representing them are not in potential conflict with each other.

- (2) At any time t, for any cluster $\mathcal{H}_k$, if the K-vehicle collision avoidance algorithm determines it's necessary to apply the optimal safety controller, then all vehicles in $\mathcal{H}_k$ apply this safe control; if the K-vehicle collision avoidance algorithm determines that no safety control is needed at this time step, all vehicles in $\mathcal{H}_k$ apply the target controller that gets the cluster to its next target.

The target controller is obtained by first computing the optimal control, up to discretization accuracy, to reach the goal for any relative state of a vehicle and the goal within a finite grid using reachability offline. Online, all is needed to get the current target control is to look up the optimal control using the current relative state of the cluster and its next goal location. Hence the target locations need not to be known *a priori*.

Corollary 4. *Give the control strategy outlined above, all vehicles will remain safe from each other for all time.*

Proof. The above initialization and control strategy satisfy the assumptions of both Theorem 2 and Corollary 3. Since the union of any pair of vehicles within the same cluster and across different clusters is exactly all pairs of vehicles, any pair of vehicles will remain safe from each other for all time. □

4.4 Numerical Simulations

We demonstrate our proposed approach on safe planning and control for multiple vehicles, each with an objective of visiting multiple targets, in simulation. We show that our approach enables guaranteed safety for $N = 4$ vehicles without the need to remove any vehicle in the environment like it is assumed in [35]. In addition, we also demonstrate that our approach scales easily to large number of vehicles by demonstrating it on $N = 15$ vehicles. In all our simulations, we divide vehicles into $K = 3$ clusters and build on the 3-vehicle collision avoidance algorithm in [23].

For illustration purposes, we assumed that the dynamics of each vehicle Q_i is given by

$$\dot{p}_{x,i} = v \cos\theta_i, \; \dot{p}_{y,i} = v \sin\theta_i, \; \dot{\theta}_i = \omega_i, \quad |\omega_i| \leq \bar{\omega} \tag{4.4}$$

where the state variables $p_{x,i}, p_{y,i}, \theta_i$ represent the x position, y position, and heading of vehicle Q_i. Each vehicle travels at a constant speed of $v = 5$, and chooses its turn rate ω_i, constrained by maximum $\bar{\omega} = 1$. The danger zone for HJ computation between Q_i and Q_j is defined as

$$\mathcal{L}_{ij} = \{x_{ij} : (p_{x,i} - p_{x,j})^2 + (p_{y,i} - p_{y,j})^2 \leq R_c^2\}, \tag{4.5}$$

whose interpretation is that Q_i and Q_j are considered to be in each other's danger zone if their positions are within R_c of each other. Here, $x_{ij} = [p_{x,ij}, p_{y,ij}, \theta_{ij}] = [p_{x,i} - p_{x,j}, p_{y,i} - p_{y,j}, \theta_i - \theta_j]$. The danger zone can be equivalently defined by the L-2 norm of the x and y components of the states, i.e., $x_{ij} \in \mathcal{Z}_{ij}$ if and only if $d(x_{ij,a}) = \|x_{ij,a}\|_2 \leq R_c$.

To obtain safety levels and the optimal pairwise safety controller, we compute the BRS (2.4) with the relative dynamics

$$\begin{aligned}
\dot{q}_{x,ij} &= -v + v\cos q_{\theta,ij} + \omega_i q_{y,ij} \\
\dot{q}_{y,ij} &= v \sin q_{\theta,ij} - \omega_i q_{x,ij} \\
\dot{q}_{\theta,ij} &= \omega_j - \omega_i, \quad |\omega_i|, |\omega_j| \leq \bar{\omega}
\end{aligned} \tag{4.6}$$

where $[q_{x,ij}, q_{y,ij}]$ is $[-p_{x,ij}, -p_{y,ij}]$ rotated clockwise by θ_i around the origin on the 2D plane and $q_{\theta,ij} = -\theta_{ij}$. Note that the L-2 norm on $[q_{x,ij},\ q_{y,ij}]$ is the same as the L-2 norm on $[p_{x,ij}, p_{y,ij}]$ because changing the sign and rotating do not change the value of the norm so we could have similarly defined the danger zone as $\bar{\mathcal{Z}}_{ij} = \{\bar{x}_{ij} : \|[q_{x,ij},\ q_{y,ij}]\|_2 \leq R_c\}$ where $\bar{x}_{ij} = [q_{x,ij}, q_{y,ij}, q_{\theta,ij}]$.

For all simulation, we initialize all vehicles and states of the clusters such that any pair of vehicles in the same cluster is of distance greater than R_c of each other and the pairwise safety levels of any two distinct clusters based on the *augmented danger zones* between them are all above the safety threshold $K = 1.5$.

In Figure 4.1, we provide snapshots of the simulation of our proposed approach on 4 vehicles in an environment with 4 targets. In this simulation, the set of targets each vehicle needs to visit is $Q_1 : [A, D]$, $Q_2 : [B]$, $Q_3 : [C]$, $Q_4 : [D]$. By using our proposed team assignment algorithm presented in Section 4.3.1, the three clusters $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$ have the following vehicles assigned to them, $\mathcal{H}_1 : Q_1, Q_4$, $\mathcal{H}_2 : Q_2$, $\mathcal{H}_3 : Q_3$. Recall that the set of targets for each cluster is the union of the targets of all vehicles in the cluster. Hence $\mathcal{H}_1$ should visit targets $[A, D]$, $\mathcal{H}_2$ should visit target $[B]$, and $\mathcal{H}_3$ should visit target $[C]$. We see that the team assignment algorithm offers a solution such that no clusters have to visit more than 2 targets to encourage efficient completion of the objectives of all vehicles. If Q_1 was paired with either Q_2 or Q_3 instead, one cluster would have to visit 3 targets.

In this simulation, the danger zone radius is $R_c = 3$. For cluster $\mathcal{H}_1$, we choose the state x_1 of vehicle Q_1 to be identical to $x_{\mathcal{H}_1}$, the state of the *imaginary* vehicle representing the cluster, and choose Q_4 to be at a distance of $R_c + \epsilon$ from the cluster center where ϵ is a small positive real number. Hence $R_{\mathcal{H}_1} = \underset{i \in \{1,4\}}{max}\ d(x_{\mathcal{H}_1} - x_i) = 3 + \epsilon$. For clusters $\mathcal{H}_2$ and $\mathcal{H}_3$, the state of the *imaginary* vehicle is the state of the only vehicle in each cluster, i.e., $x_{\mathcal{H}_2} = x_2, x_{\mathcal{H}_3} = x_3$. Hence $R_{\mathcal{H}_2} = R_{\mathcal{H}_3} = 0$. For each cluster $\mathcal{H}_k$, a circle with radius $R_{\mathcal{H}_k}$ centered at $x_{\mathcal{H}_k}$ is plotted if $R_{\mathcal{H}_k} > 0$. We also plot the 0-safety level reachable sets derived from the *augmented danger zones* of the clusters around the cluster centers. We can see from the top two subplots in Figure 4.1 that the 0-safety level sets corresponding to $V_{\mathcal{H}_1\mathcal{H}_2}$ and $V_{\mathcal{H}_3\mathcal{H}_1}$ are greater than that of $V_{\mathcal{H}_2\mathcal{H}_3}$ because the radii $R_{\mathcal{H}_1\mathcal{H}_2}, R_{\mathcal{H}_3\mathcal{H}_1}$ that define their *augmented danger zones* are $R_{\mathcal{H}_1\mathcal{H}_2} = R_{\mathcal{H}_3\mathcal{H}_1} = R_{\mathcal{H}_1} + R_{\mathcal{H}_2} + R_c = R_{\mathcal{H}_3} + R_{\mathcal{H}_1} + R_c = 6 + \epsilon$ while the radius $R_{\mathcal{H}_2\mathcal{H}_3}$ defining the *augmented danger zone* between $\mathcal{H}_2$ and $\mathcal{H}_3$ is $R_{\mathcal{H}_2\mathcal{H}_3} = R_{\mathcal{H}_2} + R_{\mathcal{H}_3} + R_c = 3$.

In Figure 4.1, as the clusters move towards their first targets, they get into potential conflicts with each other. Hence the safety control kicks in. After each cluster successfully resolves the conflict, $\mathcal{H}_2$ heads to target B, $\mathcal{H}_3$ heads to C, and $\mathcal{H}_1$ first goes to target D, followed by target A. At time $t = 14.5$s, we see that all vehicles have completed their objectives. Note that once a vehicle has visited all its targets, it remains at its last visited target and is no longer considered for collision avoidance.

Vehicle	Vehicle Targets	Cluster	Cluster Targets
Q_6	[F, G, H]	$\mathcal{H}_1(red)$	[F, G, H, I, J, M]
Q_7	[H, I]		
Q_8	[H, I, J]		
Q_{10}	[I, M]		
Q_{14}	[J]		
Q_1	[A, C, E]	$\mathcal{H}_2(green)$	[A, B, C, D, E, G]
Q_2	[A, C]		
Q_4	[B, C, D]		
Q_5	[B, E]		
Q_9	[B, D, G]		
Q_{15}	[C, E]		
Q_3	[P, K, O]	$\mathcal{H}_3(blue)$	[P, A, F, K, O, N]
Q_{11}	[P]		
Q_{12}	[A, F]		
Q_{13}	[O, N]		

Table 4.1: This table summarizes the targets for each vehicle, the cluster each vehicle is assigned to based on the proposed cluster assignment algorithm, and the targets that each cluster should visit for the 15-vehicle collision avoidance problem. We see that the solution to the cluster assignment successfully minimized the maximum number of targets each cluster needs to visit and load balances it so that each cluster needs to visit the same number of targets.

We demonstrate the scalability and effectiveness of our proposed method with a simulation on getting 15 vehicles to complete their objectives where there are 16 targets in the environment. In this simulation, the danger zone radius is $R_c = 2$. The targets of each vehicle and the cluster assignments from running our proposed team assignment algorithm are summarized in Table 4.1. We see that our proposed assignment algorithm successfully divides the vehicles into three clusters such that the number of targets each cluster needs to visit is well-balanced. Each cluster visits the targets in the order under the column "Cluster Targets" in Table 4.1. The top left graph in Figure 4.2 shows the starting configuration of the vehicles where the initialization scheme is similar to that explained for the four-vehicle simulation: cluster $\mathcal{H}_1$ (red) has its center at Q_{10}, i.e., $x_{\mathcal{H}_1} = x_{Q_{10}}$ and the rest of the vehicles in the cluster are located at equal distance to each other on a circle of radius $R_c = 2 + \epsilon$ centered at the cluster center. Similarly, for cluster $\mathcal{H}_2$ (green) and $\mathcal{H}_3$ (blue), the cluster center is located at where vehicles Q_9 and Q_{13} are at respectively, and the rest of the vehicles in each cluster are located at equal distance to each other on a circle of radius $R_c = 2 + \epsilon$. In general, we make the state of the imaginary vehicle representing the cluster identical to the state of the vehicle that completes its objective last in the cluster. We see that our proposed method resolves all conflicts and all 15 vehicles complete their objectives of visiting their targets while maintaining safety successfully.

For the 15-vehicle simulation, it takes on average 0.018 seconds to perform computation at each time step. All computations were done on a MacBookPro 15.1 laptop with an Intel Core i7 processor.

4.5 Conclusion and Future Work

In this paper, we proposed a novel method for *any* number of vehicles to complete their objectives of visiting multiple targets with guaranteed safety for a class of dynamical systems. We demonstrate the effectiveness and scalability of our approach through a 15-vehicle simulation. Our work is a promising step towards making HJ reachability more applicable to real world applications by guaranteeing safety for *any* number of vehicles when they complete their objectives while avoiding the need to have highly structured formations such as a single platoon [25] or having to know trajectories of other vehicles in advance [26]. Future work includes optimizing the order in which the targets are visited if target locations are known *a priori* and developing guaranteed safe control strategies that require less synchronous actions among groups of vehicles for *any* number of vehicles.

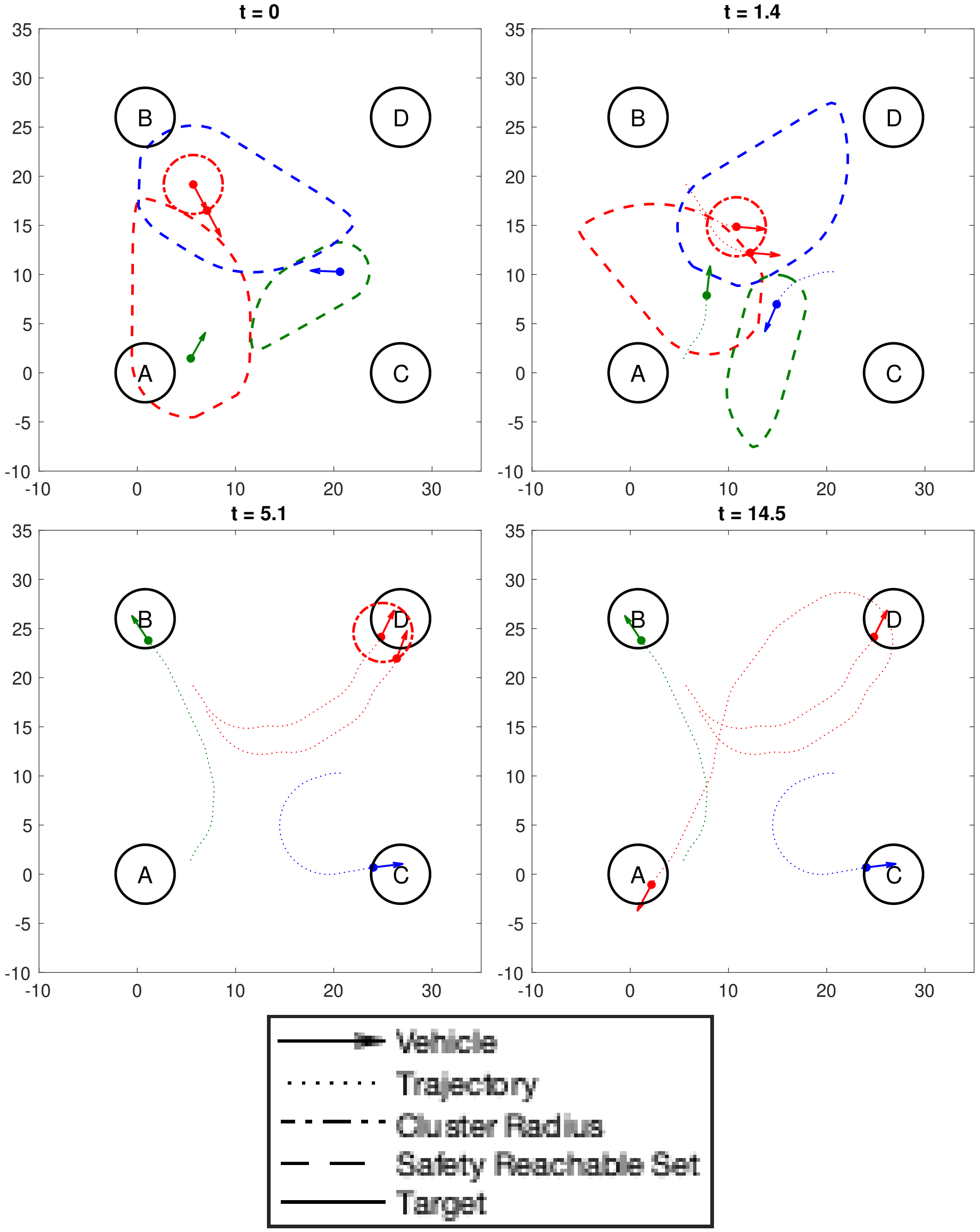


Figure 4.1: Four vehicles Q_1, Q_2, Q_3, Q_4 are tasked with visiting their targets. Based on their targets, the team assignment optimization problem described in Section 4.3.1 assigns Q_1 and Q_4 to cluster $\mathcal{H}_1$ (red), Q_2 to cluster $\mathcal{H}_2$ (green), and Q_3 (blue) to cluster $\mathcal{H}_3$. At $t = 1.4$s, the clusters get into potential conflicts with each other and the safety control strategy kicks in to make sure each vehicle remains safe. At $t = 14.5$s, we see that each vehicle completes visiting all their targets successfully without any collisions.

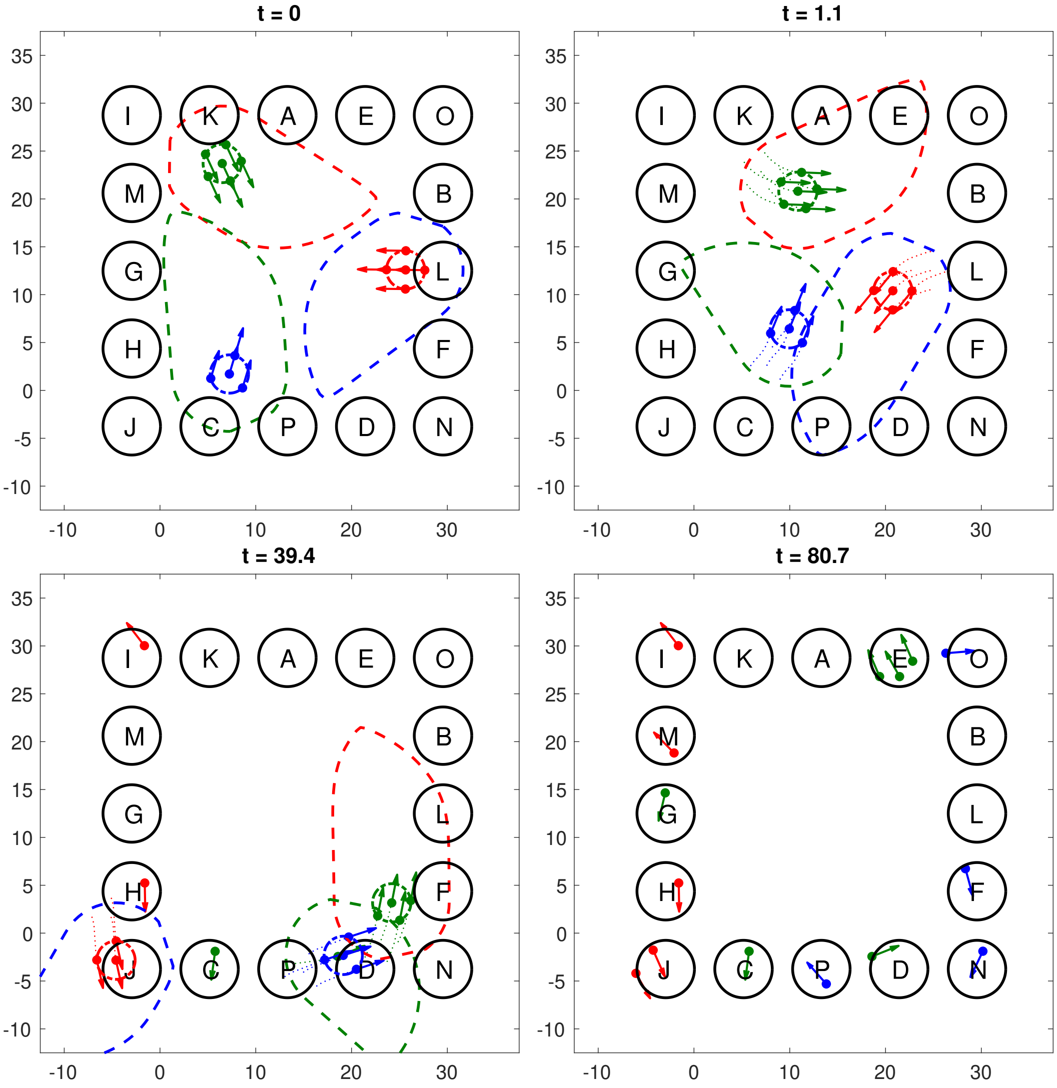


Figure 4.2: In this figure, we demonstrate our approach on 15 vehicles. The vehicles are assigned into three cluster, with cluster $\mathcal{H}_1$ (red) having 5 vehicles, cluster $\mathcal{H}_2$ (green) having 6 vehicles, and $\mathcal{H}_3$ having 4 vehicles. We can see that the clusters resolve conflicts with each other successfully while they are en route to their targets. At the end, we see that all vehicles safely visited all their targets.

Chapter 5

Learning-based Initialization Strategy for Safety of Multi-Vehicle Systems

The research presented in this chapter originally appeared in the paper Learning-based Initialization Strategy for Safety of Multi-Vehicle Systems [72].

5.1 Introduction and Related Work

The safety of multi-vehicle systems has emerged as an essential and important problem as new technologies such as unmanned aerial vehicles (UAVs) develop quickly. We have seen vast interests and growth in the domain of UAVs in industry or for government purposes. For example, Google X [47], Amazon [5], and UPS [78] aim to use drones to accomplish their business goals of delivery of goods. Drones have also been proposed for use in transport of critical medical supplies [86], [79]. There have also been many efforts in using UAVs for disaster responses and military operations [38], [52], [9], [19]. Due to the substantial growth in utilizing drones for a wide range of domains, the Federal Aviation Administration created guidelines specifically targeting UAVs in recent years [40]. It is thus of high urgency to develop effective approaches for multiple UAVs to achieve their goals in the same environment safely.

The problem of safety in multi-agent systems has been studied through various approaches. Some methods used potential functions to address safety while vehicles travel along pre-specified trajectories [66, 32]. There have also been works that introduce the idea of velocity obstacles, induced by control inputs of the vehicles, for collision avoidance [4, 41, 15]. Authors in [60] used control strategies derived with Lyapunov-type analysis for safe control of multiple vehicles. However, these approaches do not flexibly offer the safety guarantee for general dynamical systems that reachability offers. They also don't offer the the desirable property of a "least-restrictive" safe control strategy that reachability-based strategies permit.

A promising class of methods for addressing safety in the context of multi-vehicle systems is differential games. In particular, Hamilton-Jacobi (HJ) reachability [63] is a framework that uses differential games to model conflicts of more than one agent. However, although HJ reachability offers safety guarantees for general dynamical systems, its computation scales exponentially with the number of states in the systems, limiting reachabilty to be directly applicable to systems with only two vehicles [45, 63]. While attempts have been made to use reachability-based methods to guarantee safety for a larger number of vehicles [26, 25], these works either make strong assumptions on the formation of the vehicles or require that the vehicles know other vehicles' trajectories *a priori*. In contrast, in this paper, we tackle *unstructured* collision avoidance where vehicles don't have to follow specific structures or require knowledge of future trajectories of other agents.

The recent work [24] is the first work that enables guaranteed safety for three vehicles in unstructured settings using reachability via a higher level control logic. [35] further investigates the problem of guaranteed safety for four vehicles, however, it requires the assumption that vehicles can be removed from the environment when conflicts cannot be resolved, which is not always possible. Guaranteed-safe collision avoidance methods for four or more vehicles in unstructured settings without needing to remove vehicles in certain situation using reachability do not yet exist. However, reachability-based methods enable the desirable least-restrictive safe control algorithms such that agents can perform *any* action while they're deemed safe. Inspired by this, we tackle the problem of improving safety performance of systems with at least four vehicles when the vehicles adopt least-restrictive safe control strategies. While our goal is not to offer safety guarantees, we demonstrate that our proposed learning-based approach can effectively improve safety performance just by learning good initialization of the vehicle states while using the same least-restrictive safe control strategy.

Machine learning approaches for tackling collision avoidance for multi-vehicle systems have been investigated in prior works. For example, [58] uses an end-to-end learning approach to generate reactive safe policies. However, it only considers local collision avoidance and assumes the system is holonomic. Another line of work uses reinforcement learning (RL) to learn control policies of multi-vehicle systems [21], [49]. However, RL-based methods require substantial number of experiences of interactions among the vehicles to learn good policies and can take hours and, often, days to train. Furthermore, they do not result in least-restrictive safe controllers. In contrast, our proposed method is not aim at learning a policy but directed towards tackling the problem of improving safety performance through learning better initialization for all agents given *any* least-restrictive safety-aware collision avoidance algorithm.

Our main contribution is a novel learning-based approach to effectively enhance the safety of multi-vehicle systems by learning good initialization of vehicles. We formulate the problem such that each vehicle is tasked with visiting a goal and each also proposes a state it will start closely at. These agents use a least-restrictive safety-aware algorithm to get to their goals while taking safety into account. Motivated by the fact that safety cannot be guaranteed for larger multi-vehicle systems and the difficulty of reasoning about long-horizon trajectories for least-restrictive safety-aware algorithms, we show that it is possible to figure out, without human intervention, a fast and effective strategy that makes only *minor* modification to each agent's original proposed initial state and run the same safety-aware algorithm while improving the safety performance of the system. We demonstrate through extensive experiments on four to six vehicles that our proposed learning-based method consistently and reliably improves the safety performance of multi-vehicle systems.

The paper is organized as follows:

- In Section 5.2, we formulate the multi-vehicle collision avoidance problem and state the main goal of devising an approach to improve safety performance of any least-restrictive safety algorithm.
- In Section 5.3, we describe our proposed learning based methods including data collection, data processing, formulation of the learning problem, and how to utilize the learned model to improve safety performance.
- In Section 5.4, we demonstrate our proposed learning-based method in large scale randomized experiments on four to six vehicles and demonstrate its effectiveness in improving safety performance of multi-vehicle systems.

5.2 Problem Formulation

Consider N vehicles, denoted $Q_i, i = 1, 2, \ldots, N$, with identical dynamics described by the following ordinary differential equation (ODE)

$$\dot{x}_i = f(x_i, u_i), \quad u_i \in \mathcal{U}, \quad i = 1, \ldots, N \tag{5.1}$$

where $x_i \in \mathbb{R}^n$ is the state of the ith vehicle Q_i, and u_i is the control of Q_i. Each of the N vehicles is tasked with visiting a target whose location $g_i \in \mathbb{R}^{n_g}$ is known before all vehicles begin their journey.

We assume the vehicles adopt a least-restrictive safety-aware algorithm $\mathcal{A}$ that explicitly optimizes for safety of the vehicles while they get to their targets. Given the vehicle dynamics in Equation (5.1), the initial states $x_i(t_0)$, and the target location g_i of each vehicle Q_i, the algorithm $\mathcal{A}$ should determine the control u_i for each vehicle Q_i based on the joint configuration of all vehicles at each time step. In addition, the safety-aware algorithm $\mathcal{A}$ should be primarily designed with safe control of multiple vehicles in mind and should not be naive when concerning safety. For example, the algorithm introduced in [23] satisfies this criteria. In this paper, we focus on multi-vehicle systems where there is no guarantee that the safety-aware algorithm $\mathcal{A}$ is able to get all vehicles to their targets without any safety violation and our goal is to improve the safety performance of the system when safety cannot be guaranteed.

We allow each vehicle the flexibility in determining approximately where their starting states are. Instead of having full freedom of placing vehicles wherever we wish, we make the problem more challenging by only allowing our proposed method to place each vehicle within a *close* neighborhood of the original proposed state of the vehicle. We also enact the constraint that we are not allowed to modify the proposed initial states of N_{fixed} vehicles of the N vehicles. Mathematically, let the original proposed initial state of vehicle Q_i be $x_{i,o}(t_0) = [p_{1,o}, p_{2,o}, \dots, p_{K,o}]$ where $p_{k,o}$'s, $k \in \{1, \dots, K\}$, are disjoint blocks of the state and how the state of the system is divided into blocks can be freely determined by users of our proposed approach. For each agent Q_i such that we are allowed to modify the initial state for, the new initial state $x_i(t_0) = [p_1, p_2, \dots, p_K]$ based on our proposed method should satisfy constraints $\|p_k - p_{k,o}\| \leq \epsilon_k$ for some small real $\epsilon_k > 0$. The norm can be any norm that makes sense for measuring distance, which typically we use the L-1 or L-2 norm. On the other hand, if we are restricted from modifying a vehicle Q_j's initial state, then the new initial state $x_j(t_0) = x_{j,o}(t_0)$.

Given the vehicle dynamics in Equation (5.1), the original proposed initial state of each vehicle $x_{i,o}(t_0)$, the set of N_{fixed} vehicles that we cannot modify initial states for, the danger zones $\mathcal{Z}_{ij}$, the target location g_i for each vehicle Q_i, and the least-restrictive safety-aware algorithm $\mathcal{A}$, we propose an effective learning-based method to improve the safety performance of the multi-vehicle system while adopting the same algorithm $\mathcal{A}$. We demonstrate the effectiveness of our proposed learning-based approach by comparing it with randomly selecting *close* neighboring states of the original proposed initial states as new initial states with experiments and show that our approach results in better overall success rate of zero safety violation throughout the execution. Our proposed method also achieves lower number of total safety violations on average.

5.3 Methodology

In this section, we describe in detail our proposed learning-based method for improving safety performance of *any* least-restrictive safety-aware algorithm while incurring very little computation cost online. In particular, our proposed approach encompasses how we frame this problem as a machine learning problem, which includes gathering data, modeling the problem, learning the model, and using the learned model to obtain better initialization for the vehicles to enhance the safety performance of the multi-vehicle system.

5.3.1 Data gathering and preparation

To gather training data for a N-vehicle system for our proposed approach, we ran M simulations such that the initial states of all vehicles are randomly generated as follows. First, we determine N distinct initial states that will likely make collision avoidance a challenging problem. In each simulation, we then randomly assign each vehicle to a distinct initial state it should start close to. For each vehicle, we further randomly sample a state around the initial state it is assigned to such that the new initial state is in close proximity to its original initial state as illustrated in Section 5.2. There are N distinct fixed goal locations, one for each of the N vehicles. In each simulation, we also randomize the goal location each vehicle is assigned to. The reason we determine in advance a set of original initial states the vehicles should start close to and the target locations instead of just randomly sample initial states and target locations throughout the entire space is that the vehicles will rarely even come close to being in danger of each other in the latter initialization method. We want to focus on challenging scenarios where we have high confidence that the agents will come into close contact with each other as they head to their targets, enabling the safety-critical control from algorithm $\mathcal{A}$ to play a large role in the safety performance and making it meaningful to apply our proposed method.

For each simulation $j \in \{1, \dots, M\}$, we keep track of the following information: the initial states of all vehicles $x_i(t_0)$'s, the goal locations of all vehicles g_i's, and an indicator variable y_j that represents whether the least-restrictive safety-aware algorithm $\mathcal{A}$ was able to get all vehicles to their goals without *any* vehicle getting into each other's danger zone in this trial. We let $y_j = 1$ if all vehicles reach their goals without *any* safety violation and $y_j = 0$ otherwise. Note that for each trial j, the simulation continues even when vehicles get into each other's danger zone and only ends when all vehicles have reached their goals.

To illustrate how we propose to construct the features for training, first let the concatenated vector of all initial states $x_i(t_0)$ and target locations g_i in trial j be

$$p_j = [x_1(t_0), \dots, x_N(t_0), g_1, \dots, g_N] \in \mathbb{R}^{N\times(n+n_g)}. \tag{5.2}$$

We construct the feature map $\phi(p)$ as follows: first we determine the order these initial states are in counter-clockwise starting from a particular reference direction such as the twelve o'clock direction. This gives a bijective map whose domain and range are both $\{1, \dots, N\}$ and maps each vehicle to its position based on the ordering logic. We use $x^{(i)}(t_0)$ and $g^{(i)}$ to denote the initial state and the target location of the vehicle in the ith position based on the ordering logic mentioned above. The feature map ϕ is then

$$h = \phi(p) = [x^{(1)}(t_0), \dots, x^{(N)}(t_0), g^{(1)}, \dots, g^{(N)}]. \tag{5.3}$$

After applying this feature map to the data gathered from all M trials, we obtain the data set $\{h_j, y_j\}_{j=1}^{M}$ for training.

5.3.2 Learning a model with machine learning

To achieve our desired goal of determining good initialization for vehicles, one intermediate step is to determine the likelihood of algorithm $\mathcal{A}$ succeeding in getting all vehicles to their targets without any vehicle getting into each other's danger zone given the initial states and target locations of all vehicles. To achieve this, we use supervised learning to make predictions on this likelihood.

During training, we model the probability that algorithm $\mathcal{A}$ will succeed in getting all vehicles to their targets without any safety violations given the feature vector h as $f_\theta(h)$ and aim to learn the parameter θ where $f_\theta(h)$ can in general be any function approximator such as a neural network. As described in the probabilistic modeling part 2.2.1.1 in the

Background chapter, given data set $\{h_j, y_j\}_{j=1}^{M}$ where y_j's are binary variables, we minimize the following negative log likelihood with respect to θ to solve for the optimal θ:

$$\sum_{j=1}^{n} -y_j \log f_\theta(h_j) - (1-y_j)\log\left(1 - f_\theta(h_j)\right). \tag{5.4}$$

We use stochastic gradient descent to find a local minimizer $\theta^\star$ of this loss function. Once we obtain the minimizer $\theta^\star$, given any new feature vector h representing the configuration of the initial states and target locations of the vehicles in the environment not seen during training, we predict the probability that algorithm $\mathcal{A}$ will get all vehicles to their goals without any safety violations as $f_{\theta^\star}(h)$.

5.3.3 Evaluation on novel test scene online

Recall that our goal is to design a strategy to propose a new initial state *close* to the original proposed initial state of each vehicle that will result in a higher success rate of getting all vehicles to their targets without any danger zone violations. We also aim to have less number of total danger zone violations with a better initialization.

Given a novel scene online where the original proposed states of each vehicle Q_i is $x_{i,o}(t_0) = [p_{1,o}, p_{2,o}, \dots, p_{K,o}]$ where as described in Section 5.2, $p_{k,o}$'s are disjoint blocks of the state. Suppose each vehicle's target location is g_i. To find a good initialization, first we uniformly sample L sets of N initial states in the close neighborhood of $x_{i,o}(t_0)$'s by setting the constraint that each sampled state $x_i(t_0)$ should be in the set $\{x_i(t_0) = [p_1, p_2, \dots, p_K] | \|p_k - p_{k,o}\| \le \epsilon_k, k = 1, \dots, K\}$. For the N_{fixed} vehicles that we are restricted from modifying their original proposed initial states of, we set $\epsilon_k = 0, \forall k \in \{1, \dots, K\}$. Otherwise we set ϵ_k to a small positive real number. Note that we cannot modify the target locations of the vehicles.

After obtaining L candidate sets of initial states for all vehicles and constructing data points p_l's, $l \in \{1, \dots, L\}$, based on Section 5.3.1, we select the set of initial states with the highest likelihood of succeeding in getting all vehicles to their targets without any danger zone violations using the learned function approximator $f_{\theta^\star}(h)$ as the new set of initial states. Mathematically, the selected set of initial states is the the set of initial states $p^\star$ correspond to where

$$p^\star = \underset{l \in \{1,\dots,L\}}{max} f_{\theta^\star}(\phi(p_l)). \tag{5.5}$$

5.4 Experiments

In this section, we present extensive experimental results which demonstrate that with our proposed learning-based initialization strategy, the safety performance of multi-vehicle systems are effectively and reliably better compared with the baseline initialization strategy that randomly picks a set of initial states from all candidate initial sets in the vicinity of the original proposed set of initial states of all vehicles.

For all experiments, we use the least-restrictive safety-aware algorithm proposed in [24]. This reachability-based algorithm guarantees safety for three-vehicle systems but does not guarantee safety when the number of vehicles N is greater than 3. This algorithm has been demonstrated to be substantially better in safety performance already compared to a baseline safety algorithm in the paper when $N > 3$. Thus the safety algorithm we use in this paper is not a naive collision avoidance algorithm and is an ideal algorithm for use in the evaluation on our proposed learning-based initialization strategy. We conduct experiments using this algorithm on multi-vehicle systems where the number of vehicles N are equal to four, five, or six.

In our experiments, the dynamics of each vehicle Q_i is given by the Dubins Car dynamics

$$\dot{q}_{x,i} = v\cos\phi_i,\ \dot{q}_{y,i} = v\sin\phi_i,\ \dot{\phi}_i = \omega_i, \quad |\omega_i| \leq \bar{\omega}$$

where the state variables $q_{x,i}, q_{y,i}, \phi_i$ represent the x position, y position, and heading of vehicle Q_i. Each vehicle travels at a constant speed of $v = 5$, and chooses its turn rate ω_i, constrained by maximum $\bar{\omega} = 1$. The danger zone for HJ computation between Q_i and Q_j is defined as

$$\mathcal{Z}_{ij} = \{x_{ij} : (q_{x,j} - q_{x,i})^2 + (q_{y,j} - q_{y,i})^2 \leq R_c^2\}, \tag{5.6}$$

whose interpretation is that Q_i and Q_j are considered to be in each other's danger zone if their positions are within R_c of each other. Here, x_{ij} represents their joint state, $x_{ij} = [q_{x,j} - q_{x,i}, q_{y,j} - q_{y,i}, \phi_j - \phi_i]$.

To obtain safety levels and the optimal pairwise safety controller, we compute the BRS (2.4) with the relative dynamics

$$\begin{aligned} \dot{s}_{x,ij} &= -v + v\cos\phi_{ij} + \omega_i s_{y,ij} \\ \dot{s}_{y,ij} &= v\sin\phi_{ij} - \omega_i s_{x,ij} \\ \dot{\phi}_{ij} &= \omega_j - \omega_i, \quad |\omega_i|, |\omega_j| \leq \bar{\omega} \end{aligned} \tag{5.7}$$

where $[s_{x,ij},\ s_{y,ij}]$ is $[q_{x,ij},\ q_{y,ij}]$ rotated clockwise by ϕ_i around the origin on the 2D plane. Note that we can similarly define the danger zone as $\{[s_{x,ij}, s_{y,ij}, \phi_{ij}] : (s_{x,j} - s_{x,i})^2 + (s_{y,j} - s_{y,i})^2 \leq R_c^2\}$ because the norm of a vector is invariant under rotations around the origin.

To evaluate the effectiveness of our proposed learning-based initialization strategy, we perform large scale experiments on two settings of the speed v in the dynamics and the danger zone radius R_c, $(v = 5, R_c = 5)$ and $(v = 6, R_c = 4)$, for number of vehicles $N = 4, 5, 6$. For each set of experiments, we evaluate extensively the safety performance of the multi-vehicle systems by varying N_{fixed}, the number of vehicles such that we cannot modify the original proposed initial states for. Note that it only makes sense to run experiments to evaluate the effective of our approach when we can modify at least one of vehicles' proposed initial states. For each run, we initialize each vehicle by placing them symmetrically on a circle of radius $10 + 2 \times (N - 3)$ facing the center of the circle, and then add random perturbations to these states in each run. This gives us the original proposed initial states of all vehicles. This initialization ensures challenging collision avoidance scenarios as all vehicles will likely come in close contact with each other as they head to their targets. For any vehicle Q_i that we are allow to modify its initial state for, suppose its proposed initial state is $x_{i,o}(t_0) = [q_{x,i,o}, q_{y,i,o}, \phi_{i,o}]$. We constrain the new initial state $x_i(t_0) = [q_{x,i}, q_{y,i}, \phi_i]$ to be in close proximity to the original proposed state such that $x_i(t_0)$ should satisfy

$$|q_{x,i,o} - q_{x,i}| \leq 3, |q_{y,i,o} - q_{y,i}| \leq 3, |\phi_i - \phi_{i,o}| \leq \pi/5. \tag{5.8}$$

We train a model for each $N \in \{4, 5, 6\}$ for each of the two settings ($v = 5, R_c = 5$) or ($v = 6, R_c = 4$). To train each model, we gather 5000 data points when $N = 4$ and 10000 data points when $N = 5, 6$ with the data collection technique described in Section 5.3.1. For all models, we use a three-layer fully connected neural network with ReLU activation on the hidden layer and Sigmoid activation on the output layer. The number of nodes in the hidden layer of the network is $10, 15, 20$ for $N = 4, 5, 6$ vehicles respectively. As described in Section 5.3.2, we use the cross entropy loss as the loss function. For optimization, we the Adam optimizer [54] to train the network.

To demonstrate the safety benefits with our proposed approach, we compare our proposed initialization strategy with a random initialization strategy. For each N-vehicle system, we ran $N_{runs} = 200$ randomized runs for each possible combination of v, R_c, N_{fixed}. For each individual comparison run, we randomly sample $L = 10$ set of initial states in close proximity to the original proposed states such that each set satisfies the constraints (5.8). With our proposed learning-based approach, we use the learned parameters of ML model to select the best set of proposed initial states out of all candidate sets; for the baseline random initialization strategy, we uniformly select one of the L candidate sets of initial states as the initial states for the vehicles. We report the results for the settings ($v = 6, R_c = 4$) and ($v = 5, R_c = 5$) in Table 5.1 and Table 5.2 respectively. We consider the following two safety metrics:

- Success rate p_s: the percentage of runs such that *all* vehicles get to their goals without *any* safety violation.
- Average number of collisions N_{col}: the total number of safety violations throughout the *entire* execution for all N_{runs} runs divided by the product of the number of vehicles N and the number of runs N_{runs}. One safety violation is defined as a pair of vehicle being within a distance R_c of each other in a time step. There can be multiple safety violations at a given time because multiple pairs of vehicles might be in each other's danger zones at once.

From Table 5.1 and Table 5.2, we can see that our proposed learning-based initialization strategy effectively and reliably improves the success rate p_s and reduces the average number of collisions N_{col} across all experiments, For some scenarios, our approach substantially outperforms the baseline. In particular, when $v = 5, R_c = 5, N = 4, N_{fixed} = 1$, we see that our proposed approach has 90% success rate whereas the baseline approach has only 66.5% success rate. In the same setting, the average number of collisions with our proposed approach is only 25% of the average number of collisions with the baseline initialization strategy. In general, we can see that our proposed approach outperforms the baseline approach more substantially when N_{fixed} is smaller, which intuitively makes sense because we get to optimize the states of more vehicles when N_{fixed} is small. In addition, we can also observe that our proposed approach is more likely to considerably outperform the baseline when the number of vehicles N is smaller, which also makes intuitive sense as there are likely more interactions among vehicles that are more difficult to be inferred by the initial states alone when N is large. However, even when the number of vehicles N is 6 and $N_{fixed} = 5$, we still get around an 18% reduction in the average number of collisions for both settings of (v, R_c) with our proposed learning-based initialization method.

In Figure 5.1, we plotted the initial states selected with our proposed learning-based method (solid arrows) versus those selected via the baseline randomized selection (dash-dot arrows) from the L candidate sets of initial states sampled around the original proposed initial states for each scenario. For all depicted scenarios, the least-restrictive safety-aware algorithm was able to get all agents successfully to the goal locations without *any* safety violation with our proposed learning-based initialization strategy while the randomized selection failed and resulted in safety violations. The goal location of each vehicle is plotted with the same color as the vehicle. When the initial state of a vehicle is not allowed to be modified, initial states are identical for both strategies and the solid and dash-dot arrows are overlaid on top of each other.

Although it is not possible for humans to always pinpoint why the initial states learned by our proposed method are more effective for safety than the baseline just by looking at the initial states given that they are generally very close to each other, we can observe patterns by running the least-restrictive safety-aware algorithm. Our approach tends to effectively identify initial states such that vehicles are less likely to run into the situation where many vehicles are on the boundary of the unsafe sets of each other simultaneously or the situation where conflicts of multiple vehicles are less likely to be resolved based on the algorithm used. This shows that our proposed method is able to reason about safety based on the geometry of initial states of the vehicles and their goals by identifying the strength of the safety-aware algorithm used.

Figure 5.2 and Figure 5.3 further illustrate the proposed learned v.s. baseline initialization in the scenario depicted in the top right figure of Figure 5.1. We plot the danger zone around each vehicle with a dash-dot circle in the same color as the vehicle; if the base of the arrow representing a vehicle is in a circle of a different color, safety has been violated. We see that in Figure 5.2, our proposed initialization strategy enables that only three (red, blue, purple) of the four vehicles get close to the unsafe sets of each other, which our safety-aware algorithm is able to resolve with guaranteed success. At the end, all vehicles get to their goals without *any* safety violations. On the other hand, in Figure 5.3, the initial states selected by the baseline strategy results in all four vehicles getting very close to each other and the algorithm isn't able to maintain safety while resolving the conflicts. We can see that

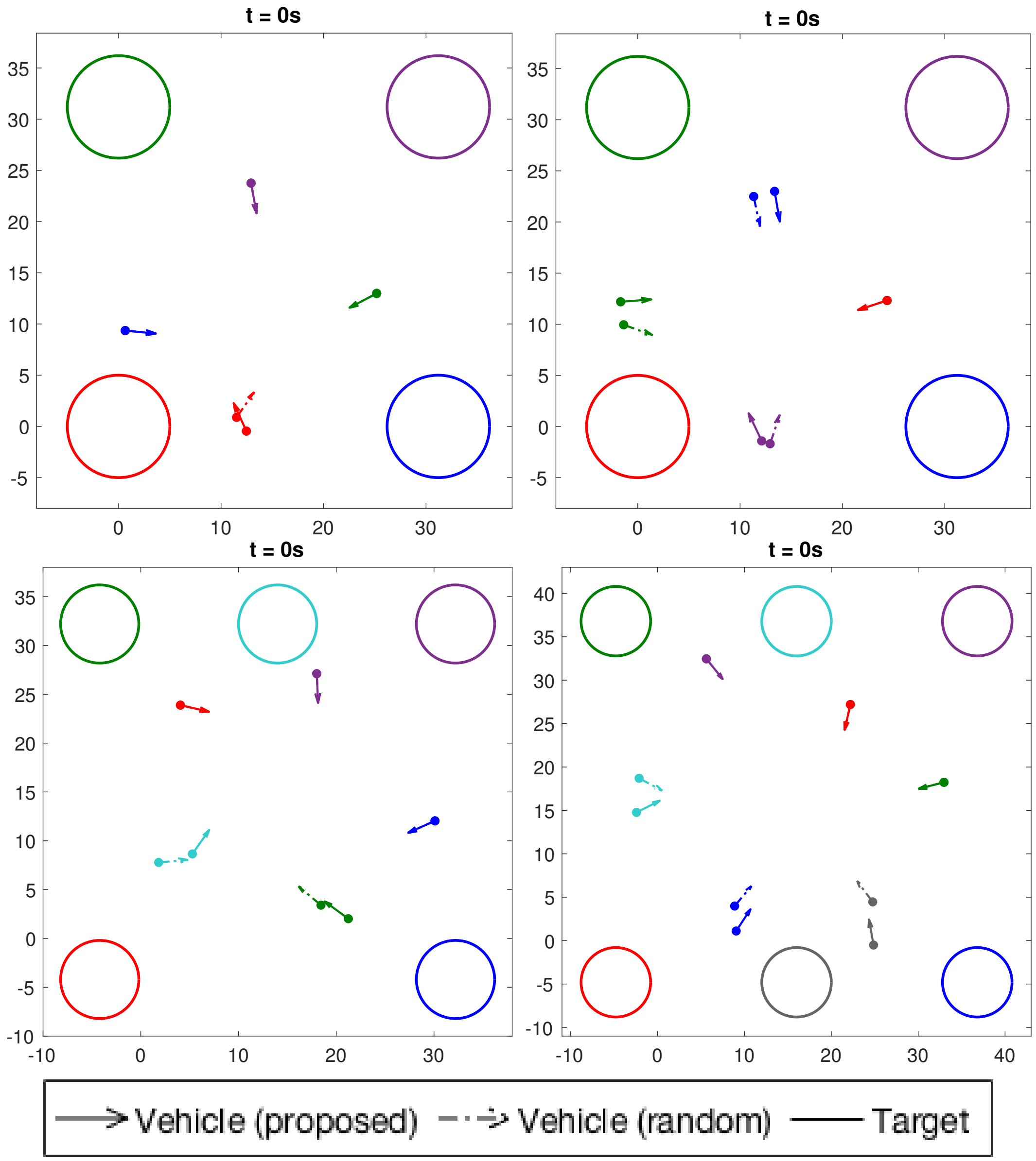


Figure 5.1: In this figure, we illustrate the initial states selected by our proposed learning-based strategy (solid arrows) versus those chosen with the baseline random selection method (dash-dot arrows) for four different scenarios where our proposed method succeeded in getting all vehicles to their goals successfully without any safety violation while the baseline method resulted in safety violations even though the initial states selected from the two methods are very close to each other. Figure 2 and 3 further illustrate the simulation based on the initial states selected by the two different strategies in the scenario depicted in the top right figure.

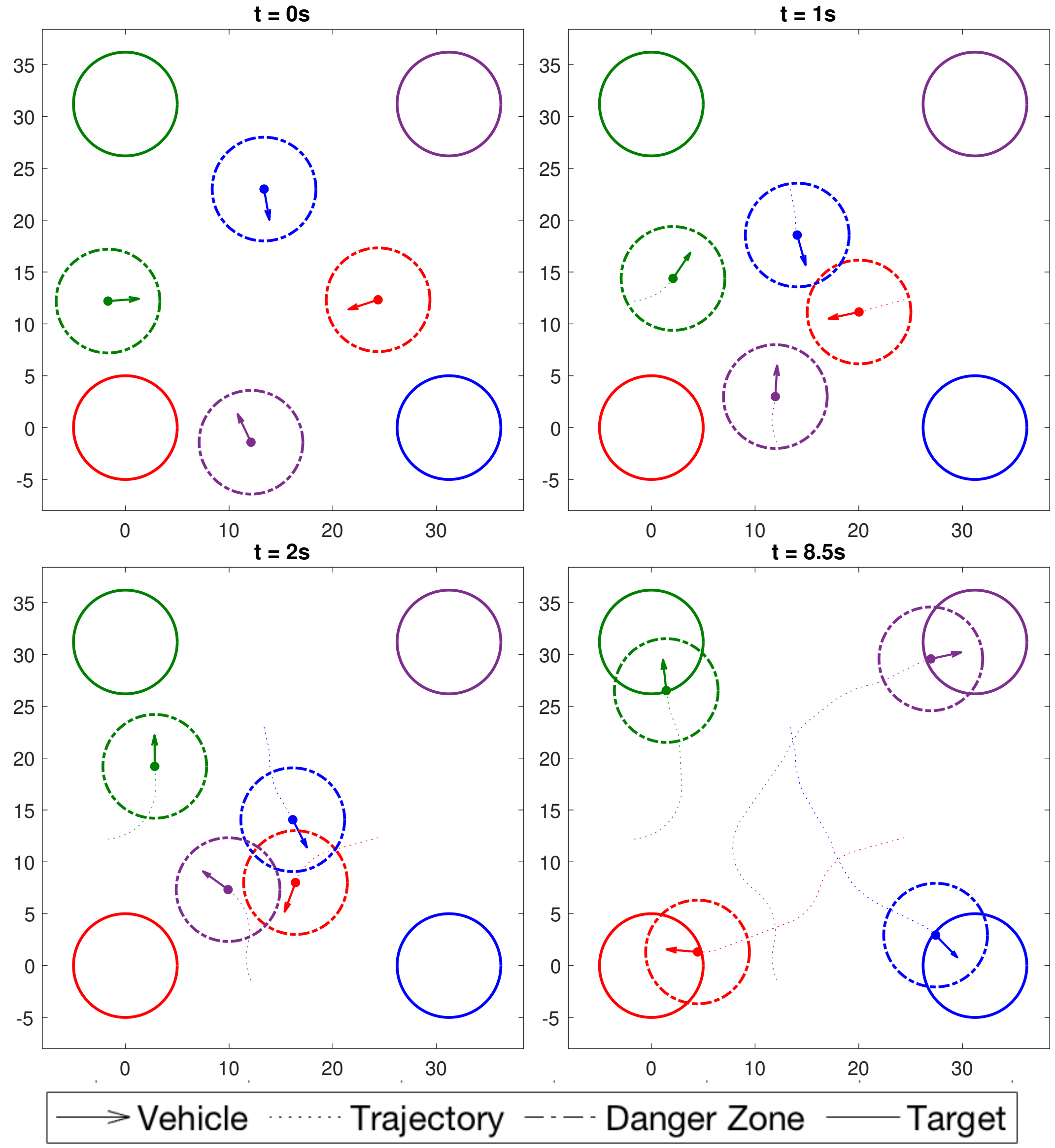


Figure 5.2: This figure illustrates different time points of the simulation when we use our proposed learning-based initialization strategy to select the initial states of the vehicles. This scenario is identical to that in the top right figure of Figure 5.1. We observe that our proposed approach learns to identify the strength of the safety-aware algorithm in guaranteeing safety for three vehicles and initializes vehicles such that only three vehicles end up coming into close contact with each other. All vehicles successfully reach their goals without *any* safety violations.

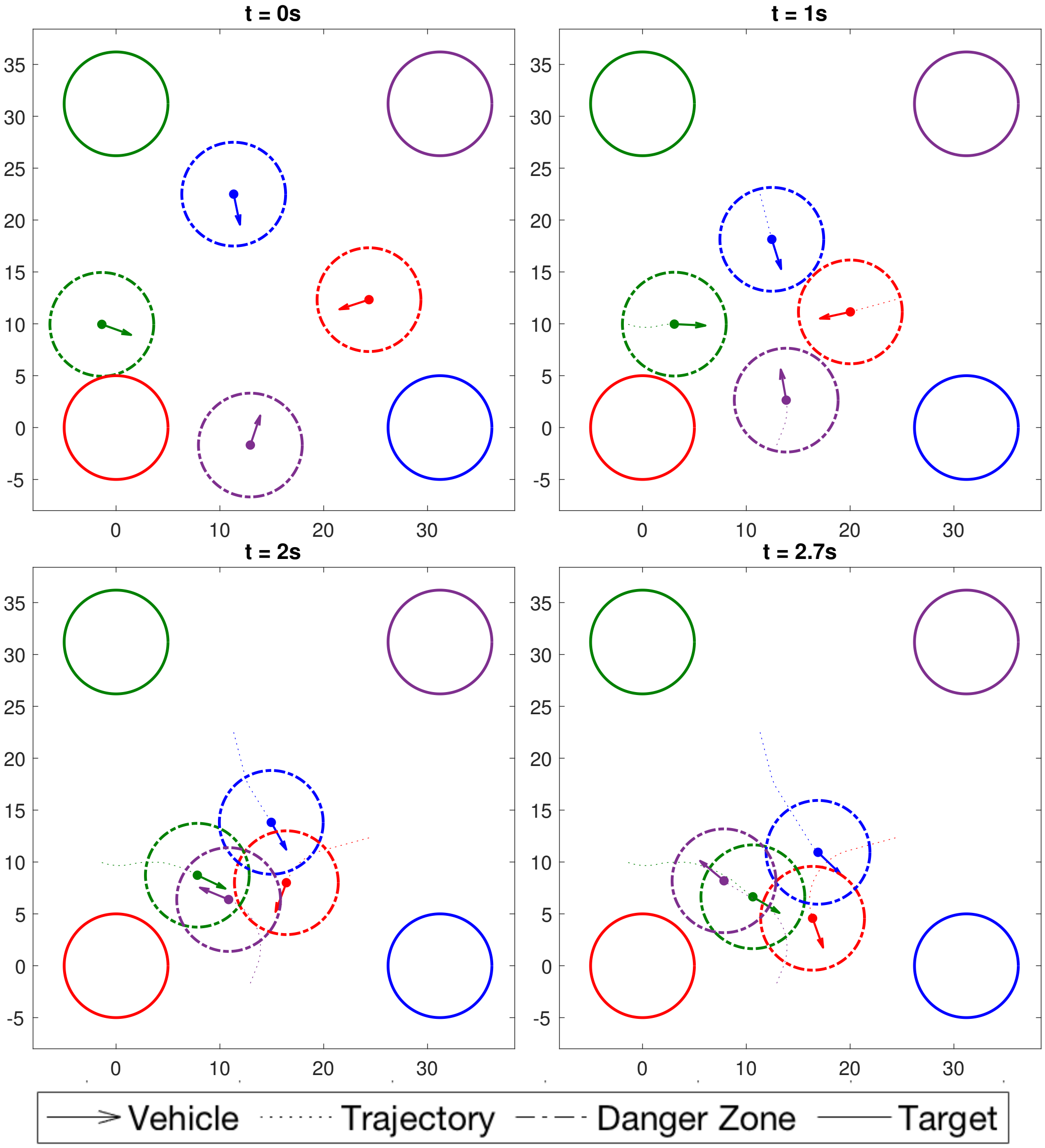


Figure 5.3: This figure illustrates the simulation when the baseline random initialization strategy is used in the scenario identical to that in the top right figure of Figure 5.1. This is meant to contrast Figure 5.2 that with the randomized strategy, even though the initial states are very close to those selected by our proposed method, it results in the undesirable event that the green and the purple vehicles get into each other's danger zone at time $t = 2.0$s and $t = 2.7$s.

the green and purple vehicles violated safety at time $t = 2.0$s and $t = 2.7$s. We see that our learning-based proposed method is able to effectively pick up on the advantages of the least-restrictive safety-aware algorithm assign a higher probability of success for initialization that is favorable with the algorithm used, effectively improving the safety performance of the multi-vehicle system.

		$N_{fixed} = 0$		$N_{fixed} = 1$		$N_{fixed} = 2$		$N_{fixed} = 3$		$N_{fixed} = 4$		$N_{fixed} = 5$	
		p_s	N_{col}	p_s	N_{col}	p_s	N_{col}	p_s	N_{col}	p_s	N_{col}	p_s	N_{col}
Four vehicles	Learned	**92**	**0.425**	**88.5**	**0.915**	**89.5**	**0.59**	**81.5**	**1.175**	-	-	-	-
	Random	75.5	1.58	77.5	1.69	80	1.255	76	1.75	-	-	-	-
Five vehicles	Learned	**88**	**1.11**	**79.5**	**1.87**	**80.5**	**1.44**	**79.5**	**1.925**	**69.5**	**2.61**	-	-
	Random	66.5	3.32	64.5	3.415	67.0	2.925	73.5	2.265	57.5	4.35	-	-
Six vehicles	Learned	**74**	**2.54**	**67.5**	**3.145**	**70.5**	**2.46**	**66.5**	**3.5**	**63**	**3.635**	**62**	**3.575**
	Random	66	3.77	56	4.58	62	4.335	61.5	4.285	55	4.575	58	4.425

Table 5.1: In this table, we summarize the success rate p_s and the average number of collisions N_{col} where speed $v = 6$ and danger zone radius $R_c = 4$ when using the learned initialization strategy versus using the baseline randomized initialization strategy. Our method outperforms the baseline in safety performance for both metrics across all scenarios.

		$N_{fixed} = 0$		$N_{fixed} = 1$		$N_{fixed} = 2$		$N_{fixed} = 3$		$N_{fixed} = 4$		$N_{fixed} = 5$	
		p_s	N_{col}	p_s	N_{col}	p_s	N_{col}	p_s	N_{col}	p_s	N_{col}	p_s	N_{col}
Four vehicles	Learned	**91**	**0.785**	**90**	**0.875**	**80.5**	**2.715**	**80**	**1.68**	-	-	-	-
	Random	68	3.305	66.5	3.56	73	2.88	71	2.81	-	-	-	-
Five vehicles	Learned	**80**	**2.315**	**77.5**	**2.79**	**71**	**4.01**	**74**	**3.39**	**71**	**3.51**	-	-
	Random	65	4.55	62	5.39	53.5	8.2	58.5	5.77	61	4.94	-	-
Six vehicles	Learned	**65.5**	**4.065**	**64**	**6.375**	**61.5**	**5.36**	**64**	**5.11**	**59**	**5.805**	**57.5**	**6.31**
	Random	57.5	6.075	46.5	8.69	52	8.27	52	7.30	49	7.25	53.5	7.70

Table 5.2: This table summarizes the success rate p_s and the average number of collisions N_{col} where speed $v = 5$ and danger zone radius $R_c = 5$. Similarly, we see that our proposed learning-based strategy outperforms the baseline in terms of safety performance across all scenarios.

During the training phase, all models take less than 30 seconds to complete training. During the online phase where we figure out the new initial states based on the learned model and the original proposed initial states, it takes on average 0.30 seconds total to compute the optimal sets of initial states for all $N_{runs} = 200$ runs in parallel for a given (v, R_c, N, N_{fixed}) setting with our proposed strategy. Thus our proposed method can be very efficiently applied as we encounter new scenes online.

5.5 Conclusion

In this paper, we proposed a novel approach for enhancing the safety performance of least-restrictive safety-aware algorithms for multiple vehicles in *unstructured* settings and showed that it is possible to use machine learning to make *minor* modifications to initial states of vehicles in the environment and improve, sometimes quite substantially, the safety of the system compared to a randomized initialization approach. This is a promising step towards making least-restrictive safety algorithms such as those enabled by reachability more practically useful in unstructured scenarios for multi-vehicle systems that safety cannot be guaranteed for with the algorithms.

Part III

Safety under Uncertain Dynamics

Chapter 6

A Framework for Online Updates to Safe Sets for Uncertain Dynamics

The research presented in this chapter was originally published in the paper A Framework for Online Updates to Safe Sets for Uncertain Dynamics in IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2020 [69].

6.1 Introduction

Machine learning can help robots adapt to unseen scenarios, but the fear of damaging the robots or their environment hinders its deployment in the real world. Combining learning algorithms with control theoretic tools, developed to ensure the safety of dynamical systems, can help with this. In our work, we build on Hamilton-Jacobi (HJ) reachability [63], a control-theoretic framework that offers safety guarantees for dynamical systems. Intuitively, HJ reachability computes safe sets and an action policy that can together ensure that the system does not enter a danger zone. Once safe sets are computed, they can be used in combination with learning algorithms to build frameworks that can be guaranteed to be safe, while achieving a given task.

However, computation of safe sets suffers from the curse of dimensionality, due to discretization of the state space. [68] reported taking 3 days for safe set computation on a 4-dimensional system and exact computation of reachability is intractable for systems with more than 5 dimensions. As a result, the optimal safe policy and safe sets are often computed offline for low-dimensional systems, assuming perfect knowledge of the dynamics. In the presence of uncertainties, however, the pre-computed safe sets might not be valid, and can lead to dangerous situations on the robot. This makes it important to update safe sets based on the dynamics of the robot online. [42] addresses this for low-dimensional systems.

In this work, we present a *least-restrictive* safety framework that can be used in combination with any type of controllers under uncertain dynamics by updating safe sets online for complex high-dimensional dynamical systems. Specifically, we focus on scenarios where the

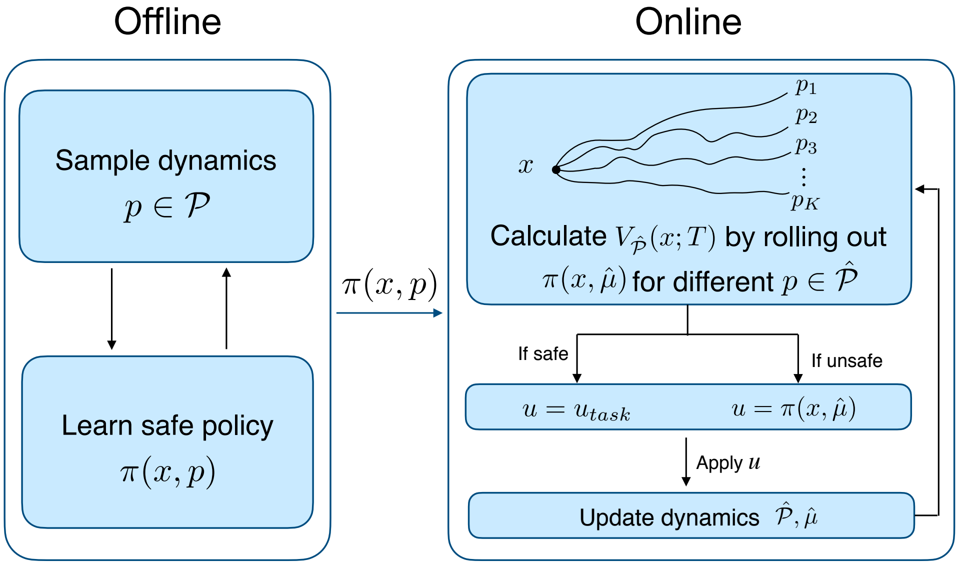


Figure 6.1: An outline of our approach. Offline, we learn a safe policy $\pi(x, p)$, which depends on state x and dynamics parameters p, by randomly sampling different dynamics parameters p. Online, we roll out $\pi(x, \hat{\mu})$ using the current estimate of the dynamics, and use it to determine the safety value at current state x. If the state is deemed safe, a task policy is applied, and otherwise a safe policy is applied. Finally, our estimate of the dynamics parameters $\hat{\mathcal{P}}$ and $\hat{\mu}$ are updated based on the new data.

dynamics are inaccurate due to uncertainty in rigid body parameters such as mass, inertia, and center of mass of links. This is a common source of uncertainty in robots, such as manipulators, but the process of identifying the uncertain parameters with learning typically does not take safety into account. In our proposed framework, we learn a safe policy offline by considering a distribution of dynamics for high-dimensional systems using reinforcement learning (RL) by building on [43]. Online, we start with an initial belief of the distribution of the dynamics parameters, update it as we gather more data, and re-compute the safe set by forward simulating the safe policy based on the new belief of the dynamics parameters. As a result, the robot is not overly conservative during online execution, while maintaining a high level of safety. Figure 6.1 provides a high level overview of our approach.

The central ideas of our framework can also be directly used with learning approaches with optimization components, like model-based RL [74]. We demonstrate incorporating safety derived from our framework in a model-based RL setting, by adding a safety constraint to the optimization problem, thereby proposing a safe model-based RL setup where the task policy takes safety into account.

Our work is a step towards online updates and sim-to-real transfer of safe sets for high-dimensional systems. We test the efficacy of our proposed framework at avoiding obstacles during random exploration on an 8-dimensional quadrotor and a 3-link manipulator (6-dimensional state space). In our experiments, our approach is able to avoid obstacles reliably, as compared to using a nominal safe set, especially for complex dynamics such as the 3-link manipulator in challenging scenarios. While our work is shown only in simulation, we emulate sim-to-real differences by perturbing dynamics parameters and adding unmodeled noise. We also test incorporating safe sets learned from our framework in a model-based RL setting on a 2-link manipulator, and show that this results in safer learning when completing a task, compared with standard model-based RL. Our results showcase the robustness of our framework and open promising applications to robotic systems in the future.

6.2 Related Work

Safety of dynamical systems has been widely studied in control literature, and used in combination with learning approaches to ensure or encourage safe learning. Constrained optimization, for example with barrier functions, can be used to certify probabilistic safety [81] or guide exploration to safe regions [28]. However, barrier function methods are generally limited to control-affine systems and the uncertainty considered in these papers are input-independent. MPC approaches have also been proposed to address safety [80], [56]. However, safety computations in these works are either applied to linear systems or local linear approximations of nonlinear systems. In comparison, our proposed approach can be applied to general nonlinear dynamical systems directly and the uncertainty considered in our proposed framework can be input-dependent.

Works such as [16, 29, 30] use Lyapunov functions to guarantee or encourage safety during learning. However, [16] requires a Lyapunov function for a given system, [29] is limited to discrete action spaces, and [30] uses approximations in its proposed Lyapunov constraints when addressing safety. This can limit their applicability to the safety of complex high-dimensional complex robots.

Our paper focuses on addressing safety with HJ reachability [63], which is a control-theoretic framework that provides safety guarantees for general dynamical systems. There has been a lot of work on guaranteeing safety of single-agent and multi-agent systems using reachability under varying assumptions on the dynamics and agent formation, such as [27, 34, 24, 12]. [11] provides an overview.

In this work, we build on [43] for approximating the *best possible control* with reinforcement learning (RL), which typically adopts a discrete-time formulation. Hence in this paper, we work with the discrete-time formulation of reachability where in the infinite-horizon scenario, the value function $V(x)$ is defined as

$$V(x) = \min \left\{ l(x), \max_{u \in \mathcal{U}} V(x + f(x,u)\Delta t) \right\}.$$

In practice, due to errors introduced by discretization, it is common to introduce a safety level $\epsilon > 0$ such that the safe set is defined as $\mathcal{K} = \{x : V(x) > \epsilon\}$ in order to ensure safety. We adopt this convention in this paper.

6.2.1 Learning to approximate safe sets from HJ reachability

It is intractable to compute exact safe sets for dynamical systems with more than five states, due to discretization of the state space during computation [62], and the resulting curse of dimensionality. As a result, there have been some works on approximating safe sets for high-dimensional problems.

[68] proposes to use function approximators to represent the optimal safe policy for control-affine systems, which in turn generates safe sets. [43] proposes a time-discounted Safety Bellman Equation that adapts the standard dynamic programming backup to induce a contraction mapping in the space of value functions. As a result, RL algorithms designed for temporal difference learning can now be used to learn safe sets for single-player systems. Note that there exists a 2-player formulation of reachability [63] where bounded disturbance is considered. However, no effective approach has been proposed for approximating two-player reachability for general high dimensional systems.

In practice, we found that the method proposed by [43] learns policies for complicated or high dimensional systems more reliably. However, it assumes perfect knowledge of the dynamics and hence does not account for discrepancies between offline training and online environment, which can cause the robot to think it's safe while it's not. We adapt this approach to learn a safe policy under dynamics uncertainty. Next, we use the learned safe policy to compute safe sets online, while updating the estimate of the dynamics from data.

6.2.2 Online updates to safe sets from HJ reachability in robotics problems

In this section, we give an overview of papers that address online updates to safe sets in the presence of uncertainties in dynamics or the environment. [10] proposes to use local updates and warm-start to generate safe sets online, as static obstacles in the environment are detected. However, the proposed methods still rely on discretization of the state space and is hence intractable for high-dimensional systems.

[42] uses a Gaussian process (GP) to model uncertainty in the dynamics of the system, followed by applying standard HJ reachability to compute safe sets based on the estimated dynamics. However, this also relies on online re-computation of safe sets through discretization of the state space, making it inapplicable to high-dimensional systems. Moreover, the uncertainty considered in this work is input-independent, which is easily violated for common platforms such as robot manipulators.

In contrast, our online update framework can be used on high-dimensional problems, and does not assume input-independent uncertainty. We assume inaccuracies in dynamics parameters of a robot and aim to identify these parameters online, while maintaining safety. This is relevant for tasks such as lifting heavy objects, where the added mass can affect a manipulator's dynamics.

Algorithm 6.1: Online loop of proposed framework

```
Given : safe policy $\pi(x, p)$, $\hat{\mathcal{P}}_0$, $\hat{\mu}_0$, $x_0$, $T$, $\epsilon$, $t = 0$
while $t < T$ do
    $s_t = V_{\hat{\mathcal{P}}_t}(x_t; T)$; // safety value based on $\hat{\mathcal{P}}_t$, $\hat{\mu}_t$
    if $s_t > \epsilon$ then
        $u_t$ = any action (can be an action from a learning controller or a random action)
    else
        $u_t = \pi(x_t, \hat{\mu}_t)$
    end
    $x_{t+1} = f(x_t, u_t; p_{true})$ ;
    Update belief of dynamics parameters with $(x_t, u_t, x_{t+1})$ and compute $\hat{\mathcal{P}}_{t+1}, \hat{\mu}_{t+1}$ ;
    $t \leftarrow t + 1$ ;
end
```

6.3 Framework for Safe Set Computations for Uncertain Dynamics

In this section, we present our framework for offline training and online updates to safe sets, outlined in Figure 6.1. During the offline phase, we train a safe policy $\pi(x, p)$, which takes the state x and dynamics parameters p as input. During the online phase, we forward simulate the safe policy $\pi(x, \hat{\mu})$ from the current state x, using our current best estimate of the dynamics parameters $\hat{\mu}$. This computes an approximate safety value $V(x)$ based on our belief about the dynamics distribution. If $V(x) > \epsilon$, the robot is in the safe set, and can apply any task-specific actions. Otherwise, the robot applies the safe action derived from the safe policy to avoid entering the danger zone. The resulting state transition on the robot is then used to update the belief about dynamics parameters p.

6.3.1 Offline computation of safe policy

During offline computation, we use reinforcement learning to train a safe policy $\pi(x, p)$ which is a function of both the state x and the dynamics parameters p. We assume that we know where the danger zone $\mathcal{Z}$ is during offline training. Every N episodes, we sample new dynamics parameters p and use them to collect data to train $\pi(x, p)$. This data is generated by repeatedly sampling the "true" dynamics parameters from the set $\mathcal{P} = [p - dp, p + dp]$ for some positive dp for the dynamics simulator. For a fixed p, the safe policy $\pi(x, p)$ is thus trained on data from a distribution of dynamics, with dynamics parameters drawn from $\mathcal{P}$, making it robust to slight variance in the estimated dynamics. The value of dp is determined based on the predicted uncertainty on the dynamics parameters during test time.

To train $\pi(x, p)$, we adapt the update rule from [43] to suit our purposes and use RL algorithms such as Soft Actor-Critic [50] or Q-learning to train the safe policy $\pi(x, p)$. The update rule of the Q-function $Q(x, p, u)$ we use during RL training is

$$\begin{aligned} Q(x, p, u) \leftarrow & (1-\gamma) l(x) + \\ & \gamma \min \left\{ l(x), \max_{u' \in \mathcal{U}} Q(x + f(x, u; p)\Delta t, p, u') \right\}, \end{aligned}$$

where $f(x, u; p)$ is the dynamics of the robot with dynamics parameters p and γ is the discount factor.

6.3.2 Online updates to safe set

Given the safe policy $\pi(x, p)$ learned offline, we can generate safe sets online based on our estimate of the distribution of the dynamics parameters p. We iteratively perform the following steps at every time step t: (1) Compute the safety value $V_{\hat{\mathcal{P}}_t}(x_t)$ of the current state x_t based on the current estimate of the dynamics parameters set $\hat{\mathcal{P}}_t$. (2) Execute an action on the robot based on whether the safety value is above the safety threshold ϵ. (3) Update the estimate of the distribution of p using the new data gathered.

Algorithm 6.1 summarizes our framework for the online phase. We describe the above three steps during the online phase in detail below.

6.3.2.1 Computing the safe set online

At any given time step t, we maintain an estimate of $\hat{\mathcal{P}}_t$, a set that dynamics parameters p fall in with some high probability c. We also maintain a current best estimate $\hat{\mu}_t$ of the parameters. For example, for a one-dimensional p, if we estimate our belief of p with a Gaussian distribution $p \sim \mathcal{N}(\hat{\mu}_t, \hat{\sigma}_t^2)$ and $c = 0.95$, then $\hat{\mathcal{P}}_t = [\hat{\mu}_t - 1.96\hat{\sigma}_t, \hat{\mu}_t + 1.96\hat{\sigma}_t]$. $\hat{\mathcal{P}}_t$ can also be a discrete set if the dynamics parameters are drawn from a discrete distribution.

The safety value at any state x at time t for a fixed time horizon T is computed as follows:

$$V_{\hat{\mathcal{P}}_t}(x; T) = \min_{p \in \hat{\mathcal{P}}_t} V(x; p, T) \tag{6.1}$$

where $V(x; p, T)$ is the estimated safety value for dynamics parameter p for time horizon T. To compute $V(x; p, T)$, we roll out the safe policy $\pi(x, \hat{\mu}_t)$ from state x for horizon T. By denoting the resulting trajectory from the roll-out as $\xi_{x,p}^{\pi_{\hat{\mu}_t}, T}(\cdot)$, we compute $V(x; p, T)$ as follows

$$V(x; p, T) = \min_{t' \in \{t, t+1, \ldots, t+T\}} l\left(\xi_{x,p}^{\pi_{\hat{\mu}_t}, T}(t')\right). \tag{6.2}$$

Intuitively, $V(x; p, T)$ is the minimum distance between the robot and the danger zone, when executing the policy $\pi(x, \hat{\mu}_t)$, if the true dynamics parameters were p, for a horizon of T. At any time t, we can compute the safety values at all states x in the space and form a safe set based on this. However, in practice, we only need the safety value at the current state x_t to determine the safety of the robot. By taking the minimum of $V(x_t; p, T)$ over all possible dynamics parameters in $\hat{\mathcal{P}}_t$, the robot uses the safe policy if *any* dynamics parameter in $\hat{\mathcal{P}}_t$ results in $V(x_t; p, T)$ less than or equal to the safety threshold ϵ.

As described in Equation 6.1, to compute the safety value $V_{\hat{\mathcal{P}}_t}(x_t; T)$ at x_t, we need to forward simulate the dynamics for all $p \in \hat{\mathcal{P}}_t$. When $\hat{\mathcal{P}}_t$ is a continuous set, in practice, we discretize finely over $\hat{\mathcal{P}}_t$ and simulate the dynamics with each of the discretized dynamics parameters in $\hat{\mathcal{P}}_t$ in parallel.

Given our proposed approach, we now formally present a proof showing that the safe sets computed using our proposed framework are conservative under specific conditions.

Theorem 4. *Assume $\hat{\mathcal{P}}_t$ is a discrete set. For any time t, state x and horizon T, if the true dynamics parameter $p_{true} \in \hat{\mathcal{P}}_t$, then $V_{\hat{\mathcal{P}}_t}(x; T) \leq V^\star(x; p_{true}, T)$ where $V^\star(x; p_{true}, T)$ is the true safety value with p_{true} as the true dynamics parameters.*

Proof. First, we remind the reader that $V(x; p, T)$ is the safety value at x derived from using policy $\pi(x, \hat{\mu}_t)$ as presented in Equation 6.2. Now we know that for any p, $V(x; p, T) \leq V^\star(x; p, T)$ because $\pi(x, \hat{\mu})$ is at most as good as the true optimal policy $\pi^\star(x, p)$. Hence, $V_{\hat{\mathcal{P}}_t}(x; T) = \min\limits_{p \in \hat{\mathcal{P}}_t} V(x; p, T) \leq \min\limits_{p \in \hat{\mathcal{P}}_t} V^\star(x; p, T) \leq V^\star(x; p_{true}, T)$, where the second inequality holds because $p_{true} \in \hat{\mathcal{P}}_t$ under our assumption. □

This implies that our framework results in a conservative estimate of the true safe set under the aforementioned assumptions. For cases where dynamics parameters are drawn from a continuous set, we discretize the dynamics parameters set $\hat{\mathcal{P}}_t$ during forward roll-out. As a result, we lose guaranteed conservatism, but we still demonstrate empirically that using our proposed approach is safer than using the nominal safe sets.

6.3.2.2 Determining the action to take

In the case of HJ reachability with continuous dynamics [63], as long as the optimal safe control policy is applied immediately when the safety value is smaller or equal ϵ, safety of the robot is guaranteed. Although we are working with discrete dynamics, this decision rule still provides a good criterion for selecting whether or not to execute the safe policy. When the safety value is above ϵ, i.e., $V_{\hat{\mathcal{P}}_t}(x_t; T) > \epsilon$, the robot is determined safe and it can apply any action, such as an action determined by any learning controller. When $V_{\hat{\mathcal{P}}_t}(x_t; T) \leq \epsilon$, the robot is deemed unsafe and it applies the action determined by the safe policy. In this sense, we have a *least-restrictive* safety framework such that the robot is free to perform any action until it is close to the unsafe set, at which point, the safety controller takes over.

6.3.2.3 Updating dynamics from data

To update our belief about the dynamics parameters p, we can use any system identification approach that gives us a probabilistic estimate of p. Identifying dynamics parameters for robots is widely studied in robotics, such as in [7], [77]. In the experiments for this paper, we model the uncertain dynamics parameters as a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ where μ and Σ are the mean and covariance matrix of the Gaussian respectively. We use Bayesian Linear Regression (BLR) to update our belief of μ and Σ, and in turn use them to update $\hat{\mu}_t$ and $\hat{\mathcal{P}}_t$. Interested readers can consult Chapter 3 in [17] for details about Bayesian Linear Regression. This gives us an online and continual way of learning dynamics parameters that can easily generalize to tasks such as picking and placing heavy objects. In cases where the uncertain parameters are not linear in the dynamics, we can use more advanced techniques for parameter identification, such as [77].

6.3.3 Safe Model-based Reinforcement Learning

Our approach can easily be used in combination with model-based learning approaches, like model-based RL. We add a safety constraint to the model-based RL optimization, which renders the search for the optimal policy to be biased towards safety. Given current state x_t, the current best estimate $\hat{\mu}_t$ of the parameters and the set $\hat{\mathcal{P}}_t$ that dynamics parameters fall in with high probability, the safe model-based RL optimization problem becomes

$$u_{t:t+H-1} = \arg\min_{u_{t:t+H-1}} \sum_{h=t}^{t+H-1} c(x_h, u_h) \tag{6.3}$$

$$\text{s.t.} \quad x_{h+1} = f(x_h, u_h; \hat{\mu}_t) \tag{6.4}$$

$$V_{\hat{\mathcal{P}}_t}(x_{h+1}; T) > \epsilon \quad \forall h \in \{t, \ldots, t+H-1\} \tag{6.5}$$

where $f(x_h, u_h; \hat{\mu}_t)$ is the discrete dynamics function assuming the dynamics paramters are $\hat{\mu}_t$ and cost $c(x_h, u_h)$ is determined based on the task, such as getting the robot to a goal. After u_t is applied on the robot, the (state, action, next state) transition is used to update our belief about the dynamics. The current state is then updated to the new state, and the process repeats. If no feasible action sequence can be found, the action $\pi(x_t, \hat{\mu}_t)$ from the safe policy is applied. [55] proposes a related setup for model-predictive control, with ellipsoidal safe sets computed with linearized dynamics. In contrast, our approach incorporates safety directly using the original dynamics.

Note that the planning horizon H is typically chosen to be much shorter than the safety horizon T. This is because increasing H increases the dimensionality of the optimization variables, actions $u_{t:t+H-1}$. This can lead to poorer solutions or high computation time when solving the optimization problem for a large H. On the other hand, T does not affect the optimization dimension, and can be much larger than H. This ensures that even though our model-based RL algorithm has limited foresight for task planning, the safety constraint enforces a longer horizon plan for safety. In our experiments, we use random sampling in the space of actions to solve the optimization problem.

6.4 Experiments

In this section, we present experimental results on 2-link and 3-link manipulators (4D and 6D problems), and an 8D quadrotor system. We perform extensive experiments to compare performance of our framework against applying safe sets computed with inaccurate nominal dynamics. Furthermore we present results on the benefit of using MBRL with safety constraints derived from our framework, versus using MBRL with no safety constraints.

6.4.1 Comparison between our proposed framework and using nominal safe sets

We experiment with two different scenarios, random and challenging. For the random scenario, we initialize the robot randomly at states that are safe; for the challenging scenario, we initialize the robot at safe states closer to the obstacles. We observe that the robot rarely gets close to the unsafe states with random initialization. The initialization in the challenging scenario increases the frequency the robot gets close to the unsafe states. In all trials across different scenarios and methods, the nominal dynamics parameter is always $\hat{p}_0 = 2$. For the random scenario, the true dynamics is sampled from the set $p_{true} \sim \text{uniform}[\hat{p}_0 - 2\sigma, \hat{p}_0 + 2\sigma]$ and for the difficult scenario, the true dynamics is sampled from $p_{true} \sim \text{uniform}[\hat{p}_0 - 2\sigma, \hat{p}_0]$, for $\sigma = 0.1, 0.3$. In general, our method shines in conditions that are tough. For safety, it is important to avoid obstacles in all scenarios, and the challenging scenario showcases the robustness of our approach versus using the nominal safe set.

We simulate 200 trials for various dynamical systems. Each trial lasts for $T = 100$ time steps. A trial is successful if the robot does not hit any obstacle throughout the entire trial and unsuccessful otherwise. In each trial, we apply a uniform random action, if it is determined that the robot is safe, and apply the safe policy otherwise. We compare the success rates of our framework with that of using nominal safe sets with inaccurate dynamics. We use Soft Actor-Critic [50] with implementation from [84] to train the safe policy $\pi(x, p)$ for all dynamical systems.

Both methods start from the same initial safe condition. To emulate sim-to-real differences, we add a random Gaussian noise with zero mean and standard deviation of 0.1 to the control inputs for all systems. Note that the noise added to the control input does not satisfy the assumptions of BLR (Section 6.3.2.3), but our proposed method consistently outperforms the baseline in challenging scenarios. In addition, for the 3-link manipulator, we also perform an extensive experiment where our framework assumes a damping coefficient different from the true damping coefficient, without updating our belief about this coefficient. This further shows the robustness of our approach in situations that violate the assumptions of our proposed framework.

The complete experimental results are summarized in Table 6.1.

6.4.1.1 2-link manipulator

We consider the task of safely controlling a 2-link manipulator in the x-y plane. In general, the dynamics of a manipulator are:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = u. \tag{6.6}$$

For a 2-link manipulator, $q = [\theta_1, \theta_2]$ are the joint angles of the two links, $M(q)$ is the inertia matrix, and $C(q, \dot{q})$ is the Coriolis matrix. The full state of the system is 4-dimensional, $x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$. $u = [\tau_1, \tau_2]$ is the 2-dimensional control input, which is constrained to $|\tau_1|, |\tau_2| \leq 1$. While we did not consider gravity in the dynamics in Equation 6.6, these dynamics generalize to manipulators with manufacturer-provided gravity compensation, such as Kuka LBR [57]. We use simplified dynamics with masses at the end of each link (each of length 1.0 m) with the mass at the end of the first link being m_1 =1.0 kg. The uncertainty in the dynamics comes from the uncertainty in the mass at the end effector $p = m_2$. In all experiments for manipulators, the danger zone in the environment is a square obstacle centered at $(x, y) = (0, 1.5)$m with edge length 1.0m.

The initialization of the states is described as follows: where each variable is sampled

	$[\theta_1, \theta_2]$	$[\dot{\theta}_1, \dot{\theta}_2]$
Random	$[0, 0] + x_{rand}$	$[0, 0] + dx_{rand}$
Challenging	$[\frac{\pi}{7}, \frac{\pi}{6}] + x_{chal}$	$[0.3, 0.4] + dx_{chal}$

from uniform distribution within the range: x_{rand} : $[-\pi, \pi]$ rad, dx_{rand} : $[-0.5, 0.5]$rad/s, x_{chal} : $[-0.5, 0.5]$ rad, and dx_{chal} : $[-0.5, 0.5]$rad/s. Figure 6.2 visualizes the state that we sample around for the challenging scenario. Note that if the sampled initial state is unsafe, we re-sample until the initial state is inside the safe set.

As summarized in Table 6.1, in the random scenario, the success rates for our proposed framework and using the nominal safe sets are similar. However, in the challenging scenario with $\sigma = 0.3$, our approach successfully avoids the obstacle with a 99.5% success rate, while using the nominal safe set only succeeds 85% of the time. This shows that even on a 4-dimensional system, inaccurate dynamics can adversely affect the performance of safety approaches, especially in challenging scenarios. In such a case, we see that online updates to safe sets can considerably improve the rate of success for avoiding obstacles.

6.4.1.2 3-link manipulator

The dynamics of a 3-link manipulator can also be described by Equation 6.6. This makes the state space 6-dimensional and action space 3-dimensional, adding computational complexity. The full state is $x = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$. $u = [\tau_1, \tau_2, \tau_3]$ is the torque, which is constrained by $|\tau_1| |\tau_2|, |\tau_3| \leq 1$. Note that computing the safe sets for this 6-dimensional system is intractable with standard reachability [63]. Similar to the 2-link manipulator, we consider a

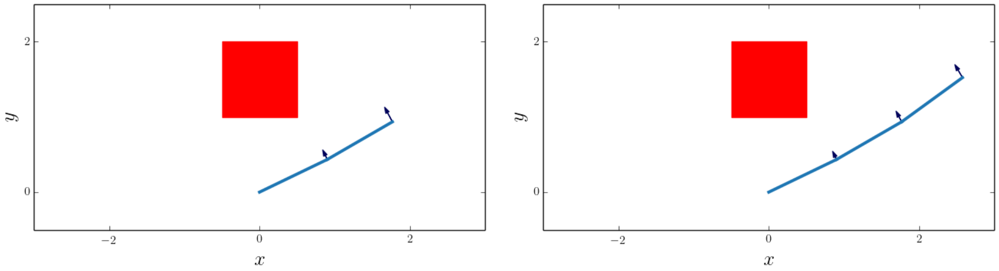


Figure 6.2: Environments and the challenging initial conditions that we randomize around for the 2-link and 3-link manipulators experiments. The arrows represent the velocities at the joints and the end effectors. The red squares represent the obstacles.

3-link manipulator with mass at the end of each link where the length of each link is 1.0 m. We denote the mass at the end of first, second, and third link as m_1, m_2, and m_3 where m_1 =1.0 kg, m_2 =2.0 kg. The uncertainty in the dynamics comes from the uncertainty in the mass at the end effector $p = m_3$.

The initialization of the states is described as follows: where each variable is sampled

	$[\theta_1, \theta_2, \theta_3]$	$[\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$
Random	$[0, 0, 0] + x_{rand}$	$[0, 0, 0]+_{rand}$
Challenging	$[\frac{\pi}{7}, \frac{\pi}{6}, \frac{\pi}{5}] + x_{chal}$	$[0.2, 0.2, 0.2] + dx_{chal}$

from the uniform distribution within the range: $x_{rand} : [-\pi, \pi]$ rad, $dx_{rand} : [-0.5, 0.5]$ rad/s, $x_{chal} : [-0.2, 0.2]$ rad, and $dx_{chal} : [-0.2, 0.2]$ rad/s. Figure 6.2 visualizes the state that we sample around for the challenging scenario

In the random scenario, we observe comparable performance between our framework and the baseline, with ours succeeding 96.5% of the time and the baseline succeeding 91.5% of the time for $\sigma = 0.3$. Our framework shines in this complicated dynamical systems in difficult scenarios, with the success rate being 53.0% with our framework and 35.5% with the baseline for $\sigma = 0.1$. The performance of both approaches gets worse as the complexity of the dynamics increase, but the nominal safe sets are more brittle than our framework. This highlights the need for robust, reliable, and online updated safe sets, especially for complex high-dimensional systems.

6.4.1.3 3-link damped manipulator

We also consider a damped variant of a 3-link manipulator, whose dynamics are given by

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + B\dot{q} = u. \tag{6.7}$$

Here, B is the damping coefficient and the rest of the notations are identical to those of Equation 6.6. The experimental settings are identical to those of the non-damped version described previously, except for the added damping. The uncertainty in the dynamics arises from the mass $p = m_3$ at the end effector, as well as the damping coefficient B. To emulate unmodelled sim-to-real differences, we do not update our belief about the damping coefficient B online. We only update our belief about m_3 online.

For all experiments, we fix the inaccurate damping coefficient to be 0.4. The true damping coefficients are sampled from a uniform distribution within the ranges $[0.3, 0.5]$ and $[0.3, 0.4]$ for the random and challenging scenarios respectively. Our experiments show that even with such unmodelled disturbances, with our approach, the robot avoids obstacles 87% of the time with $\sigma = 0.3$ in the challenging scenario. On the other hand, with the nominal safe set, the robot only avoids the obstacle 64.5% of the time. This shows that our framework generates robust safe sets that generalize to unmodelled disturbances, enabling the robot to avoid the danger zone reliably. Note that the success rates are higher than those from 3-link manipulator without damping in the previous section because adding damping makes it easier to learn a reliable safe policy offline.

6.4.1.4 Quadrotor (8-dimensional system)

We also consider an 8-dimensional quadrotor dynamical system for our experiments. The states of the quadrotor are $[x, y, z, v_x, v_y, v_z, \theta, \phi]$ where x, y, z are the positions in the x-y-z space, v_x, v_y, v_z are the velocities, and θ, ϕ are the roll and yaw angles. Denoting gravity as g, the dynamics are:

$$\dot{q_x} = v_x, \quad \dot{q_y} = v_y, \quad \dot{q_z} = v_z \tag{6.8}$$

$$\dot{v}_x = gtan\theta, \quad \dot{v}_y = -gtan\phi \tag{6.9}$$

$$\dot{v}_z = \frac{u_z}{m} - g, \quad \dot{\theta} = \frac{u_\theta}{m}, \quad \dot{\phi} = \frac{u_\phi}{m}. \tag{6.10}$$

The input to the systems are forces u_θ, u_ϕ, u_z that directly affect the vertical and angular accelerations of the quadrotor. The uncertainty in the dynamics arises from uncertainty in the mass m. The bounds on the control are: $|u_\theta|, |u_\phi| \leq 0.1$ and $u_z \in [g - 2.0, g + 2.0]$. The obstacle is a cube centered at $[q_x, q_y, q_z] = [0, 0, 0]$ with edge length 1.0 m.

The initialization of the states is described as follows: where x_{rand} and dx_{rand} are sampled

	$[q_x, q_y, q_z]$	$[v_x, v_y, v_z]$
Random	$[0, 0, 0] + x_{rand}$	$[0, 0, 0] + dx_{rand}$
Challenging	$q_x = 0, q_y = 0, q_z = x_{chal}$	$[0, 0, 0] + dx_{chal}$

from the uniform distribution within the range: $x_{rand} : [-2, 2]$m, $dx_{rand} : [-0.2, 0.2]$m/s, $x_{chal} : [0.55, 0.6]$m, and $dx_{chal} : [-0.1, 0]$m/s. For both random and challenging scenarios, the initial $[\theta, \phi]$ is always set to $[0, 0]$rad.

Even though the quadrotor has higher dimensions than the manipulators, it has simpler dynamics, making it easier to learn a good safe policy. Both using the nominal safe sets and

		2-link manipulator (4D)		3-link manipulator (6D)		3-link-damped (6D)		Quadrotor (8D)	
		$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$
Random	Nominal (Baseline)	99.5	97	93.5	91.5	97	98.5	100	99.5
	Our framework	100	98.5	93.5	96.5	98.5	97	100	98.5
Challenging	Nominal (Baseline)	93	85	35.5	27.5	80.5	64.5	89.5	85
	Our framework	99	99.5	53	39.5	93.5	87	93.5	95

Table 6.1: In this table, we show comparisons of success rates between using our framework and the nominal safe sets. For the random scenario, our framework performs slightly better than the baseline. This is due to the fact that with the initialization scheme in the random scenario, the robot rarely gets to a situation where it's close to being unsafe. However, to test the robustness of our approach and the baseline, we consider initialization that is challenging. We can clearly see the performance benefit of our framework in the challenging scenario, especially for systems with complicated dynamics such as the 3-link robot arm. Here σ determines the range of values we sample the true dynamics parameter p_{true} from and is explained in detail in the text (Section 6.4.1).

	2-link	3-link	Quadrotor
Compute time	0.17 s	0.25 s	0.10 s

Table 6.2: Average computation time for updating the dynamics and re-computing safety values at each time step. The compute time for the quadrotor is smaller because even though the state of the quadrotor has a larger dimension, its dynamics are much simpler than those of the manipulators.

our framework result in close to 100% success rates for avoiding the obstacle for the random scenario. For the challenging scenario, we amplified the difficulty by applying a downward u_z when safe, instead of a random action. In this setting with $\sigma = 0.3$, our framework has a success rate of 95% while the baseline has a success rate of 85%, again demonstrating the robustness of our framework compared to the nominal safe sets.

The compute time online for all dynamical systems considered is shown in Table 6.2, demonstrating that our framework is fast at updating dynamics and safe sets.

6.4.2 Experiments with safe model-based RL

In this section, we present experimental results on applying our proposed framework for enhancing safety under uncertain dynamics to model-based RL as described in Section 6.3.3. We consider a 2-link manipulator identical to that used in the previous experiment 6.4.1.1 and the uncertainty similarly comes from the mass m_2 at the end effector. The objective is to get the end-effector of the manipulator to some target location while avoid hitting the obstacle.

We compare safe MBRL with standard MBRL without safety. Both approaches start at the same initial configuration, $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2] = [-\frac{\pi}{12}, \frac{13\pi}{20}, 0, 0]$. The obstacle is a square

	With safety	Without safety
Completion rate	93	77
Collision rate	4	21

Table 6.3: This table summarizes our large scale experiment results for incorporating safety into MBRL. Completion rate indicates the percentage of trials the robot reaches the goal without hitting the obstacle within the maximum time steps allowed for task completion. Collision rate refers to the percentage of trials the robot hits the obstacle. We can see that incorporating safety in MBRL increases the success rate and decreases the collision rate considerably.

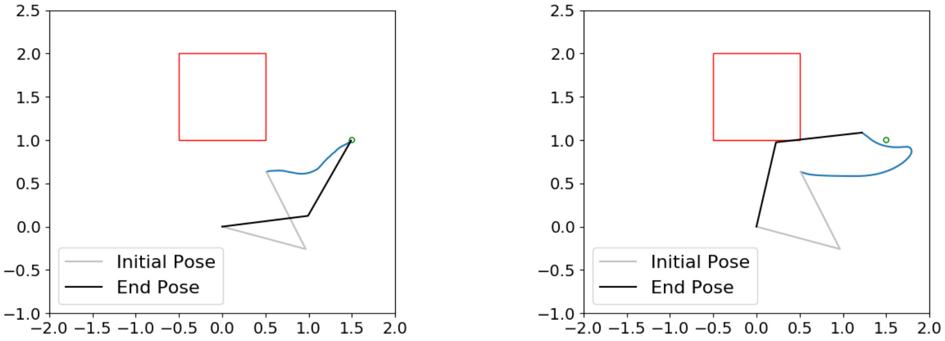
centered at $[x, y] = [0, 1.5]$ with side length 1.0 and the goal is a circle centered at $[x, y] = [1.5, 1.0]$ with radius 0.03. Figure 6.3 demonstrates qualitatively the benefit of incorporating safety in MBRL. When considering safety, the 2-link manipulator learns to apply actions that avoid the obstacle (red) while reaching the goal (green). On the other hand, with standard MBRL without safety, the robot hits the obstacle while trying to get to the goal.

To compare safe MBRL with standard MBRL quantitatively, we also ran an experiment with 100 randomized runs to evaluate the effect of incorporating safety into MBRL. For each trial, the obstacle location and the initial state are identical to those presented in Figure 6.3 and the goal location is randomized for each run. We set the maximum time steps allowed to reach the goal for both methods to be 300, with integration time-step $\Delta t = 0.1$. For each run, both methods (with and without safety constraints) start with the same initial condition and have the same goal. A trial terminates early when the robot hits the obstacle or reaches the goal. The true mass at the end effector is $p = m_2 = 2.4$. Note that both approaches update the dynamics parameter using BLR, described in Section 6.3.2.3, but MBRL with safety explicitly reasons about safety using our framework when solving the optimization problem 6.3. For both approaches, we start with an initial belief of the uncertain dynamics parameter p as a Gaussian distribution with mean 2.2 and standard deviation 0.1 and update the belief over time as MBRL runs.

The results are summarized in table 6.3. Completion rate indicates the percentage of trials in which the robot reaches the goal without hitting the obstacle, and collision rate refers to the percentage of trials where the robot hits the obstacle. We can see that by incorporating safety in MBRL, the collision rate is 4% compared with 21% when not incorporating safety. Hence using MBRL with safety derived from our proposed framework leads to a much safer learning process.

6.5 Conclusion and Future Work

In this work, we present a framework for offline training and online updates to safe sets in the context of Hamilton-Jacobi reachability. We start by learning a robust safe policy by considering a distribution over dynamics. This is then used online to generate safe sets by rolling out the safe policy from states and current estimate of the dynamics. Simultaneously,



(a) MBRL with safety

(b) MBRL without safety

Figure 6.3: With the same initial configuration (grey), MBRL with safety learns to reach the goal (green) without hitting the obstacle (red) while MBRL without safety hits the obstacle. The blue curves illustrate the trajectories of the end effector. Without safety, the robot starts out speeding towards the goal greedily, turns around, and speeds to the goal again. Due to torque saturation, it misses the goal then ends up hitting the obstacle. On the other hand, with safe MBRL, the robot moves slowly and safely towards the goal.

we collect dynamics data to update our belief about the dynamics parameters. This gives rise to a safe learning framework that allows robots to learn about its dynamics and achieve a task with reliable safety. Our framework generalizes to high-dimensional systems, such as 3-link manipulators and quadrotors, and reliably avoids obstacles in challenging scenarios where using nominal safe sets might fail. In addition, we demonstrate that the central idea of our framework can be used in combination with MBRL to have robots learn to accomplish tasks safely.

While our experimental results demonstrate that our framework is robust to uncertainties in dynamics, our experiments are conducted in simulation with added noise and unmodeled inaccuracy in the dynamics parameters. The next step is to study this approach on a real high-dimensional robot arm. In such cases, some of the modeling inaccuracy arises from inertial parameters, but there may also be other sources of inaccuracy, such as state-dependent friction coefficient, which are not captured in our current setup. In the future, we aim to develop safety approaches that consider more general dynamics uncertainty and require less computation online.

Chapter 7

Conclusion

In this dissertation, we first proposed an approach to generate a probabilistic forward reachable set that characterizes the probabilities that the human vehicle will be in different regions at each time step and, in the worst case, captures all possible states that the human can be at in the next few time steps. Then we contributed two methods for multi-vehicle collision avoidance, focusing on different aspects of the problem. We first proposed a cooperative planning and control algorithm that guarantees safety for all vehicles while they complete their tasks and this safety guarantee can be applied to any number of vehicles. Then we investigated enhancing the safety performance of least-restrictive safety-aware algorithms that don't have safety guarantees for the multi-vehicle systems being considered. To this end, we proposed a machine learning based approach that enhances the safety of the overall multi-vehicle system by making *minor* modification to the initial states of vehicles. Finally, we proposed an online framework that enables a high dimensional robot to learn about its dynamics, accomplish a task, and update its safe sets online under uncertain dynamics. We next outline a few future directions for robot safety in the pursuit of making robots more autonomous and capable of making complex decisions by themselves.

Making accurate predictions of humans or human-controlled vehicles is a challenging task but it will undoubtedly have a huge impact to the future of technology as we envision that robots will be used to work around and assist humans. In Chapter 3, we introduced a way to predict where the human may be in the next few time steps. In this work, the human operator in the experiment is aware that the robot only focuses on getting towards its goal and will not act differently based on how the human vehicle avoids it. This algorithm is primarily suitable for a second robot that aims to not collide with the human vehicle while it's avoiding the first robot as this eliminates considering the interaction effect between the first robot and the human vehicle, which may not reflect practical scenarios. As humans, we sometimes make decisions base on how our decisions will influence others. Hence one direction is to develop prediction strategies that take into account of the interaction effect between the two agents and construct probabilistic forward reachable sets based on game-theoretic reasoning. Furthermore, there may be many robotic and human vehicles operating close to each other at the same time. The heterogeneous multi-agent problem increases the complexity of the modeling and prediction problem quickly and is an important direction to realize the potential of robotic vehicles navigating safely in complicated urban scenarios. It would also be interesting to explore whether tools in reachability can be useful in prediction of this type of setting.

How to ensure the safety when multiple robotic vehicles are in close proximity will remain a highly important problem. Various assumptions are often made to make analyzing the safety of multi-vehicle systems more manageable such as assuming all vehicles have a specific form of dynamics, vehicles follow a certain formation, or vehicles have prior knowledge of how other vehicles will act etc. Generally, there is a trade-off between how few assumptions we make towards the vehicle dynamics or their formation and how strong the safety guarantees are. In Chapter 4, we proposed a way to guarantee safety for any number of vehicles by dividing vehicles into clusters. Although the assumption we made is generally weaker than other reachability-based methods for multi-agent safety, it would be beneficial to explore ways that can relax the requirement of synchronous actions among vehicles in the same cluster while still guaranteeing safety. Another potential direction is to investigate devising collision avoidance algorithms based on reachability that can guarantee safety for a larger number of vehicles in an completely unstructured setting compared to the current state-of-the-art algorithm built on reachability. This can increase the efficiency of our proposed algorithm as there will be on average less number of vehicles in a given cluster, making the time it takes for all vehicles to complete their objectives on average shorter.

Machine learning also has the potential to improve the safety of multi-vehicle systems. In Chapter 5, we proposed a learning-based approach to improve the safety of least-restrictive safety-aware algorithms by making minor adaptations to the original proposed initial states of the vehicles. This works well for the Dubins car dynamics, and we hypothesize this strategy should work well for most practical dynamical systems that this method aims to be applied to as systems such as drones or cars are intuitive to control given the geometry of where the robot is, its heading, and its target. However, it would be interesting to see whether the proposed strategy can work well on more atypical dynamical systems that do not exhibit such properties. Another possible direction is to explore ways to instead use machine learning for parts of the decision making process, such as the cutoff threshold of when safe control should be applied, of a least-restrictive safety-aware algorithm to improve safety.

The interplay between machine learning and safety will become increasingly important as the capability of using machine learning methods for robotic control enhances. Recently, machine learning has enabled agents to perform impressive tasks such as achieving super-human performance in simulated Atari games, navigating in cluttered environments through visual inputs of the environment directly, and grasping objects of various shapes and forms in an situation where the objects are cluttered in a cramped space. In Chapter 6, the proposed online framework is a first step towards making robots safer by building safe sets online during robot learning. The uncertainty considered in this work is limited compared to the uncertainty that the current state-of-the-art machine learning-based methods for task-oriented controllers can accomplish under. However, these methods typically do not take safety into account. Potential directions in this realm include generating safe sets online when the uncertainty in the dynamics is more complicated such as when there are state-dependent uncertainty such as frictions. Another line of work is to tackle safety problems when the inputs are high-dimensional such as images, when inputs include measurement from force sensors, or situations where the robotic tasks are contact-rich.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the Twenty-first International Conference on Machine Learning.* ICML '04. New York, NY, USA: ACM, 2004.

[2] A. K. Akametalu et al. "Reachability-based safe learning with Gaussian processes". In: *53rd IEEE Conference on Decision and Control.* Dec. 2014, pp. 1424–1431. DOI: 10.1109/CDC.2014.7039601.

[3] A. Alahi et al. "Social LSTM: Human Trajectory Prediction in Crowded Spaces". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2016, pp. 961–971.

[4] Javier Alonso-Mora et al. "Collision Avoidance for Aerial Vehicles in Multi-Agent Scenarios". In: *Auton. Robots* 39.1 (June 2015), 101–121. ISSN: 0929-5593. DOI: 10.1007/s10514-015-9429-0. URL: https://doi.org/10.1007/s10514-015-9429-0.

[5] Amazon.com, Inc. *Amazon Prime Air.* https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011, accessed 2020-05-15.

[6] Amazon.com, Inc. *Amazon Prime Air.* http://www.amazon.com/b?node=8037720011, accessed 2018-09-09.

[7] Chae H An, Christopher G Atkeson, and John M Hollerbach. "Estimation of inertial parameters of rigid body links of manipulators". In: *1985 24th IEEE Conference on Decision and Control.* IEEE. 1985, pp. 990–995.

[8] Aurora, Inc. *Aurora.* https://aurora.tech/, accessed 2018-09-09.

[9] AUVSI News. *UAS Aid in South Carolina Tornado Investigation.* http://www.auvsi.org/blogs/auvsi-news/2016/01/29/tornado, accessed 2016-02-16.

[10] Andrea Bajcsy et al. "An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments". In: *arXiv preprint arXiv:1905.00532* (2019).

[11] Somil Bansal et al. "Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances". In: (2017).

[12] S. Bansal et al. “Safe sequential path planning under disturbances and imperfect information”. In: *2017 American Control Conference (ACC)*. May 2017, pp. 5550–5555. DOI: `10.23919/ACC.2017.7963818`.

[13] BBC. *'Farms are going to need different kinds of robots'*. `https://www.bbc.com/news/business-56195288`, accessed 2021-07-22.

[14] Sarah Bechtle et al. “Curious ilqr: Resolving uncertainty in model-based rl”. In: *Conference on Robot Learning*. 2020, pp. 162–171.

[15] Jur van den Berg, Ming C. Lin, and Dinesh Manocha. “Reciprocal Velocity Obstacles for real-time multi-agent navigation”. In: *IEEE International Conference on Robotics and Automation*. May 2008, pp. 1928–1935. DOI: `10.1109/ROBOT.2008.4543489`.

[16] Felix Berkenkamp et al. “Safe Model-based Reinforcement Learning with Stability Guarantees”. In: *Proc. of Neural Information Processing Systems (NeurIPS)*. 2017. URL: `https://arxiv.org/abs/1705.08551`.

[17] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2007. ISBN: 978-0-387-31073-2.

[18] Patrick Bouffard. “On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments”. In: *Master's thesis,University of California, Berkeley*. 2012.

[19] Business Insider. *Drone Technology for Military*. `https://www.businessinsider.com/drone-technology-uses-applications`, accessed 2020-06-05.

[20] Business Insider. *Google's drone delivery project just shared some big news about its future*. `https://www.businessinsider.com/project-wing-update-future-google-drone-delivery-project-2017-6` accessed 2017-06-07.

[21] L. Busoniu, R. Babuska, and B. De Schutter. “A Comprehensive Survey of Multiagent Reinforcement Learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2 (2008), pp. 156–172. DOI: `10.1109/TSMCC.2007.913919`.

[22] California's Manufacturing Network. *Ready or Not, Robotics in Manufacturing Is On the Rise*. `https://www.cmtc.com/blog/overview-of-robotics-in-manufacturing`, accessed 2021-07-22.

[23] Mo Chen, Jennifer C. Shih, and Claire J. Tomlin. “Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming”. In: *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*. 2016, pp. 1695–1700.

[24] Mo Chen et al. “Guaranteeing Safety and Liveness of Unmanned Aerial Vehicle Platoons on Air Highways”. In: *arXiv:1602.08150* (2016).

[25] Mo Chen et al. “Safe Platooning of Unmanned Aerial Vehicles via Reachability”. In: *IEEE Conference on Decision and Control*. 2015.

[26] Mo Chen et al. "Safe Sequential Path Planning of Multi-Vehicle Systems via Double-Obstacle Hamilton-Jacobi-Isaacs Variational Inequality". In: *European Control Conference*. 2015.

[27] M. Chen et al. "Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems". In: *IEEE Transactions on Automatic Control* 63.11 (Nov. 2018), pp. 3675–3688. ISSN: 0018-9286. DOI: `10.1109/TAC.2018.2797194`.

[28] Richard Cheng et al. "End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks". In: *CoRR* abs/1903.08792 (2019). arXiv: `1903.08792`. URL: `http://arxiv.org/abs/1903.08792`.

[29] Yinlam Chow et al. "A lyapunov-based approach to safe reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8092–8101.

[30] Yinlam Chow et al. "Lyapunov-based Safe Policy Optimization for Continuous Control". In: *arXiv preprint arXiv:1901.10031* (2019).

[31] Kurtland Chua et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *Advances in Neural Information Processing Systems*. 2018, pp. 4754–4765.

[32] Yao-Li Chuang et al. "Multi-Vehicle Flocking: Scalability of Cooperative Control Algorithms using Pairwise Potentials". In: *IEEE International Conference onRobotics and Automation*. Apr. 2007, pp. 2292–2299. DOI: `10.1109/ROBOT.2007.363661`.

[33] Cruise Automation, Inc. *Cruise*. `https://getcruise.com/`, accessed 2018-09-09.

[34] Aparna Dhinakaran et al. "A hybrid framework for multi-vehicle collision avoidance". In: *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*. 2017, pp. 2979–2984. DOI: `10.1109/CDC.2017.8264092`. URL: `https://doi.org/10.1109/CDC.2017.8264092`.

[35] A. Dhinakaran et al. "A hybrid framework for multi-vehicle collision avoidance". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 2979–2984.

[36] Pedro Domingos. "A Few Useful Things to Know About Machine Learning". In: *Commun. ACM* 55.10 (Oct. 2012), pp. 78–87.

[37] Katherine Rose Driggs-Campbell, Roy Dong, and Ruzena Bajcsy. "Robust, Informative Human-in-the-Loop Predictions via Empirical Reachable Sets". In: *IEEE Trans. Intelligent Vehicles* 3.3 (2018), pp. 300–309. DOI: `10.1109/TIV.2018.2843125`. URL: `https://doi.org/10.1109/TIV.2018.2843125`.

[38] DSLRPros Inc. *DSLRPros Disaster response drone*. `https://www.dslrpros.com/disaster-response-drones.html`, accessed 2020-06-05.

[39] Earthsense Systems. *Earthsense*. `https://www.earthsense.co/`, accessed 2021-07-22.

[40] Federal Administration Regulation. *FAA Drone Zone*. `https://www.faa.gov/uas/`, accessed 2020-06-05.

[41] Paolo Fiorini and Zvi Shillert. “Motion Planning in Dynamic Environments using Velocity Obstacles”. In: *International Journal of Robotics Research* 17 (1998), pp. 760–772.

[42] J. F. Fisac et al. “A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems”. In: *IEEE Transactions on Automatic Control* 64.7 (July 2019), pp. 2737–2752. ISSN: 0018-9286. DOI: 10.1109/TAC.2018.2876389.

[43] J. F. Fisac et al. “Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 8550–8556. DOI: 10.1109/ICRA.2019.8794107.

[44] Jaime F. Fisac and S. Shankar Sastry. “The Pursuit-Evasion-Defense Differential Game in Dynamic Constrained Environments”. In: *IEEE Conference on Decision and Control*. 2015.

[45] Jaime F. Fisac et al. “Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints”. In: *18th International Conference on Hybrid Systems: Computation and Controls*. 2015.

[46] Jeremy H. Gillula et al. “Applications of hybrid reachability analysis to robotic aerial vehicles”. In: *The International Journal of Robotics Research*. Vol. 30. 3. 2011, pp. 335–354.

[47] Google, Inc. *X Project Wing*. https://x.company/projects/wing/, accessed 2020-06-05.

[48] V. Govindarajan, K. Driggs-Campbell, and R. Bajcsy. “Data-driven reachability analysis for human-in-the-loop systems”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 2617–2622. DOI: 10.1109/CDC.2017.8264039.

[49] J. Gupta, M. Egorov, and Mykel J. Kochenderfer. “Cooperative Multi-agent Control Using Deep Reinforcement Learning”. In: *Autonomous Agents and Multiagent Systems*. 2017.

[50] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: (2017).

[51] David Held et al. “Probabilistically safe policy transfer”. In: *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*. 2017, pp. 5798–5805.

[52] irevolutions.org. *Disaster response drone*. https://irevolutions.org/2014/06/25/humanitarians-in-the-sky/, accessed 2020-06-05.

[53] iRobot Inc. *iRobot*. https://www.irobot.com/, accessed 2021-07-22.

[54] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[55] Torsten Koller et al. *Learning-based Model Predictive Control for Safe Exploration and Reinforcement Learning*. 2019. arXiv: `1906.12189 [eess.SY]`.

[56] T. Koller et al. "Learning-Based Model Predictive Control for Safe Exploration". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 6059–6066.

[57] Kuka Inc. *Kuka LBR iiwa*. `https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa`, access 2019-09-01.

[58] Pinxin Long, Wenxi Liu, and Jia Pan. "Deep-Learned Collision Avoidance Policy for Distributed Multiagent Navigation". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 656–663.

[59] Nick Malone et al. "Stochastic Reachability Based Motion Planning for Multiple Moving Obstacle Avoidance". In: *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*. HSCC '14. Berlin, Germany: ACM, 2014, pp. 51–60. ISBN: 978-1-4503-2732-9. DOI: `10.1145/2562059.2562127`. URL: `http://doi.acm.org/10.1145/2562059.2562127`.

[60] Silvia Mastellone et al. "Formation Control and Collision Avoidance for Multi-agent Non-holonomic Systems: Theory and Experiments". In: *The International Journal of Robotics Research* 27.1 (2008), pp. 107–126. DOI: `10.1177/0278364907084441`. eprint: `https://doi.org/10.1177/0278364907084441`. URL: `https://doi.org/10.1177/0278364907084441`.

[61] David L. McPherson et al. "Modeling Supervisor Safe Sets for Improving Collaboration in Human-Robot Teams". In: *CoRR* abs/1805.03328 (2018). arXiv: `1805.03328`. URL: `http://arxiv.org/abs/1805.03328`.

[62] Ian M. Mitchell. "A Toolbox of Level Set Methods." In: *UBC Department of Computer Science Technical Report TR-2007-11 (June 2007)* ().

[63] I.M. Mitchell, A.M. Bayen, and C.J. Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *IEEE Transactions on Automatic Control* 50.7 (2005), pp. 947–957. DOI: `10.1109/TAC.2005.851439`.

[64] NS Healthcare. *Four types of medical robot leading the way in healthcare... they are taking over, but they're here to help*. `https://www.ns-healthcare.com/analysis/medical-robots/`, accessed 2021-07-22.

[65] Nuro Inc. *Nuro*. `nuro`, accessed 2021-07-22.

[66] Reza Olfati-Saber and Richard M. Murray. "Distributed Cooperative Control of Multiple Vehicle Formations Using Structural Potential Functions". In: *IFAC World Congress*. 2002.

[67] V. Rubies-Royo et al. "A Classification-based Approach for Approximate Reachability". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7697–7704.

[68] V. Rubies-Royo et al. "A Classification-based Approach for Approximate Reachability". In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 7697–7704. DOI: 10.1109/ICRA.2019.8793919.

[69] J. C. Shih, F. Meier, and A. Rai. "A Framework for Online Updates to Safe Sets for Uncertain Dynamics". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5994–6001. DOI: 10.1109/IROS45743.2020.9341606.

[70] Jennifer C. Shih. "Predicting Stochastic Human Forward Reachable Sets Based on Learned Human Behavior". In: *2019 American Control Conference (ACC)*. 2019, pp. 5247–5253. DOI: 10.23919/ACC.2019.8814396.

[71] Jennifer C. Shih and Laurent El Ghaoui. "Reachability-based Safe Planning for Multi-Vehicle Systems with Multiple Targets". In: *2021 American Control Conference (ACC)*. 2021, pp. 3288–3295. DOI: 10.23919/ACC50511.2021.9482978.

[72] Jennifer C. Shih, Akshara Rai, and Laurent El Ghaoui. *Learning-based Initialization Strategy for Safety of Multi-Vehicle Systems*. 2021.

[73] Mao Su, Yong-ji Wang, and Lei Liu. "Bounded guidance law based on differential game for three-player conflict". In: *IEEE Conference on Modeling, Identification, and Control*. 2014.

[74] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[75] S. Tanimoto. "On a class of three-player differential games". In: *Journal of Optimization Theory and Applications* 25.3 (1978), 469?473. ISSN: 1573-2878.

[76] Technoserve.org. *Drone-Assisted Mapping for Climate-Smart Cashew Production*. https://www.technoserve.org/our-work/projects/drone-assisted-mapping-for-climate-smart-cashew-production/, accessed 2021-07-22.

[77] Jo-Anne Ting, Aaron D'Souza, and Stefan Schaal. "Bayesian robot system identification with input and output noise". In: *Neural Networks* 24.1 (2011), pp. 99–108.

[78] UPS. *UPS Drone Delivery Service*. https://www.ups.com/us/en/services/shipping-services/flight-forward-drones.page, accessed 2020-06-05.

[79] Vayu Inc. *Vayu Medical Supply Drone Delivery*. https://www.vayu.us/, accessed 2020-06-05.

[80] Kim Peter Wabersich and Melanie N. Zeilinger. "Linear model predictive safety certification for learning-based control". In: *CoRR* abs/1803.08552 (2018). arXiv: 1803.08552. URL: http://arxiv.org/abs/1803.08552.

[81] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. "Safe learning of quadrotor dynamics using barrier certificates". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2460–2465.

[82] Waymo, Inc. *Waymo*. `https://waymo.com`, accessed 2018-09-09.

[83] Hao Wu et al. "Modeling Trajectories with Recurrent Neural Networks". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 3083–3090. DOI: `10.24963/ijcai.2017/430`. URL: `https://doi.org/10.24963/ijcai.2017/430`.

[84] Denis Yarats et al. "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images". In: *arXiv preprint arXiv:1910.01741* (2019).

[85] Brian D. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. 2008, pp. 1433–1438. URL: `http://www.aaai.org/Library/AAAI/2008/aaai08-227.php`.

[86] Zipline Inc. *Zipline Medical Supply Drone Delivery*. `https://flyzipline.com/`, accessed 2020-06-05.