Decision Boundary Learning For Safe Vision-based Navigation via Hamilton-Jacobi Reachability Analysis and Support Vector Machine

by

Tara Toufighi

M.Sc., Tehran Polytechnics, 2016
B.Sc., Tehran Polytechnics, 2013

Thesis Submitted in Partial Fulfillment o f the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2024

Declaration of Committee

Name:	**Tara Toufighi**
Degree:	**Master of Science**
Thesis title:	**Decision Boundary Learning For Safe Vision-based Navigation via Hamilton-Jacobi Reachability Analysis and Support Vector Machine**
Committee:	**Chair:** Lawrence Kim Assistant Professor, Computing Science
	Mo Chen supervisor Assistant Professor, Computing Science
	Yasutaka Furukawa Committee Member Associate Professor, Computing Science
	Angelica Lim Examiner Assistant Professor, Computing Science

Abstract

We develop a self-supervised learning method that can predict the decision boundary between safe and unsafe high-level waypoints for robot navigation given the first-person view in the form of an RGB image, and the current speed of the robot, without knowledge of the map of the environment. To provide the theoretical basis for such predictions, we use Hamilton-Jacobi reachability analysis, a formal verification method, as the oracle for labeling training datasets. Given the labeled data, our neural network learns the coefficients of a decision boundary via a soft-margin Support Vector Machine loss function to classify safe and unsafe system states. We experimentally show that our method is generalizable to the real world and generates safety decision boundaries in unseen indoor environments. Our method's advantages are its explainability, robustness, data efficiency, and accurate safety prediction. Finally, we demonstrate our method via real-world experiments.

Keywords: Self-supervised learning, Decision boundary, Robot navigation, Hamilton-Jacobi reachability analysis, Soft-margin Support Vector Machine, Explainability

Dedication

To my family, huge thanks for being my biggest supporters and always cheering me on to reach for the stars.

Acknowledgements

I deeply appreciate the continuous guidance, support, and encouragement from my advisor Dr. Mo Chen. He's been an amazing supervisor, always there for me, guiding me through my studies and being a great role model. I've learned so much from him, not just technical stuff but also valuable life lessons. Thanks to that I have improved and became better at conducting research as well as writing papers and giving presentations. I also would like to say special thanks to Rakesh and Minh for their assistance in my work presented here, without you guys this thesis would not be possible.

I would like to also say thanks to friends I have met in the past years and everyone in MARS lab; your friendship has made a big difference in my life, and I'm grateful for it every day.

Last but not least, I would like to express my gratitude to my family, thank you for always supporting me, especially during the tough times. Your love mean everything to me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Navigating robots through intricate and confined environments poses considerable challenges. Robots can be classified into three main categories concerning navigation: fully autonomous, semi-autonomous, and tele-operated. Despite advancements, fully autonomous vehicles have not yet attained the required sophistication to navigate our complex world independently while ensuring the safety of both the vehicle and humans [25], most notably in challenging environments with clutter, poor illumination, and narrow paths [6]. Consequently, human involvement remains indispensable for the operation of such vehicles. Semi-autonomous robots have found applications in myriad scenarios, from search and rescue [12] to assistive robots [33] and flight control [13]. Such robots allow humans and robots to cooperate to achieve a desired goal within a shared autonomy framework. Alternatively, tele-operated robots require one or more human operators to pilot them [9, 8]. This requires the operators to react swiftly to audio-visual or haptic feedback and can be physically and mentally taxing, especially in mission-critical scenarios, increasing the possibility of failure.

Visual navigation boasts advantages such as affordability, lightweight hardware, and ubiquity. Within the research community focused on learning, robot navigation is typically studied in the context of an unknown agent exploring an environment that is also deemed unknown. End-to-end (E2E) learning approach a system is designed to avoid explicit map estimation and acquire policies that directly correlate onboard sensor readings with control commands. Such methodologies offer various advantages, enabling the learning of policies without a prior understanding of the specific system or environment the robot will navigate but they lack data efficiency, robustness, safety, and explainability. The hybrid approach combines both trajectory planning and learning, our method belongs to this category.

The fundamental concept involves training convolutional neural networks (CNNs) using high-level policies. These policies leverage current RGB image observations to generate a sequence of intermediate states, referred to as "waypoints". A waypoint is a set of intermediate state variables. It serves as a link between perception and control. Ultimately, these waypoints guide the robot along a collision-free path to its desired destination in environments that were previously unexplored. Waypoints define the path that a robotic system follows on

a map, marking each step of its trajectory. Using waypoints offers better stability and intuitiveness compared to traditional controls.

In this thesis, we propose a deep learning-based method to identify sets of safe and unsafe waypoints, instead of just outputting where the robot should go next. Our system provides the users with a safe set of waypoints to aid their decision-making process in achieving their goals. This approach offers flexibility in determining the robot's next destination. Our work can act as a safety layer for any controller, waypoint-based navigation systems, or human controllers. More specifically, we can envision our approach benefits in three scenarios: enhancing mobile robot operation for novice robot operators in crowded environments through co-navigation; facilitating teleoperation in remote environments with limited human peripheral vision, where an interface aids in collision avoidance and provides guidance; and filtering for more generalizable waypoint-based policies such as [20].

Our model is trained using data generated through optimal control and a support vector machine (SVM) loss function. Our primary contributions are as follows:

1) A reachability-based framework for aiding navigation in unknown static environments and, 2) An explainable algorithm for computing an explicit decision boundary in the robot‘s state space to obtain a safe set of waypoints online based only on sensor measurements without an *a priori* map as the robot navigates, learned through minimizing the soft-margin support vector machine loss during training, this approach aims to improve interpretability in classifying safe and unsafe waypoints, and 3) A hardware demonstration of our approach, showcasing a learning-based safe decision boundary estimation by employing monocular RGB images and current linear speed.

Chapter 2

Background

Safely navigating robots using visual navigation in cluttered areas is complex. The map-based approach involves metric maps and classical path planning. The modular approach uses perception information to construct a geometric map, enabling the planning of collision-free trajectories. Creating a detailed and accurate map in environments with challenging elements like transparent objects, or with strong ambient light is difficult, even with a depth camera.

Numerous research endeavors have explored deep learning-based monocular vision-based solutions to tackle autonomous robot navigation without using metric maps [20, 4, 18, 14, 17, 29]. The direct perception approach, as outlined by Chen et al. ([11]), involves translating input images into key indicators, such as the robot's angle relative to the route, distance from lane markings, and proximity to surrounding robots. After direct perception, end-to-end learning (E2E) approaches, exemplified by Rill et al. ([28]), directly map input images to actuator actions.

Despite significant achievements in autonomous navigation through monocular vision, these approaches suffer from data efficiency and robustness. The focus often lies on training networks to learn steering angles, with the occasional inclusion of speed as an input, and evaluations commonly take place in simulated environments, neglecting real-world driving complexities. This raises concerns, especially for collision avoidance in everyday traffic scenarios ([1, 26, 32]), making the application of learning-based approaches impractical in such situations.

While some systems can identify critical situations using GPS/motion sensor data along with *a priori* maps ([1]), an effective safety system should engage in real-time environmental monitoring, issuing warnings or taking preemptive actions. Hence, a subset of research leverages monocular vision for collision avoidance. Studies focusing on collision avoidance through single-camera images often estimate time-to-collision (TTC) as a risk metric ([28]). While previous studies focus on implementing collision avoidance policies through lower-level control, our research examines learning higher-level actions (waypoints). Our approach introduces a novel perspective, considering a Hamilton-Jacobi (HJ) reachability-based value function that directly defines safety.

While end-to-end deep learning-based approaches have achieved impressive results in limited scenarios, they lack data efficiency and robustness in wide-ranging conditions. Similar to our method, a hybrid of deep learning and optimal control/path planning ([30, 20, 27, 5, 16, 21, 24, 23, 4, 3]) also seek to address these challenges.

Chapter 3

Problem Setup

3.1 Problem Setup

Our goal is to create a perception module that can predict decision boundaries that using first-person view images. We aim to learn a holistic decision boundary for the entire area rather than just labeling waypoints. The hardware configuration of our robot includes a monocular RGB camera mounted at a fixed height, with a fixed pitch and forward-facing orientation. The overview of our system is shown in Fig. 3.1. At each time step t, our learning architecture takes in an RGB image I_t representing the first-person view of the robot, along with the robot velocity v_t as input, generates a decision boundary that separates safe and unsafe waypoints in the robot's ego frame. The robot state and decision boundary are in a 4-dimensional (4D) space $(x(t), y(t), \phi(t), v(t))$ where $(x(t), y(t))$ is the robot position and $\theta(t)$ is the robot orientation at time t. A projection of this decision boundary overlaid on the first-person view of a robot is shown in the right photo of Fig. 3.1.

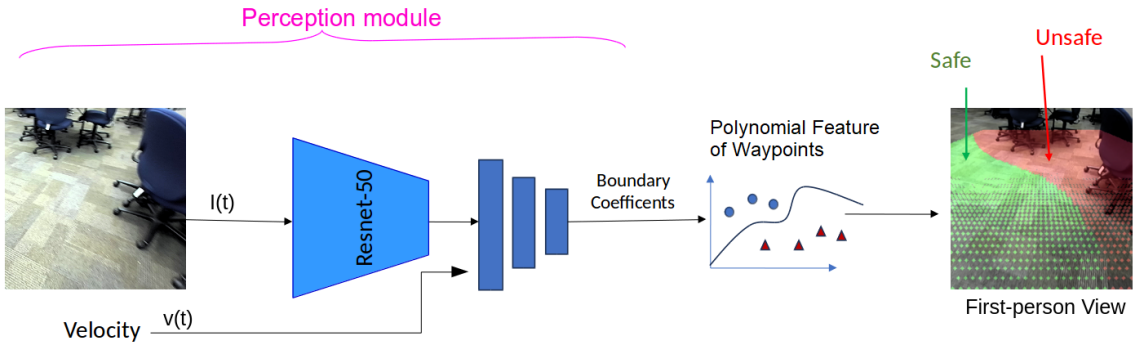


Figure 3.1: Overview of the system components

3.1.1 Dynamics Modeling

We model our ground robot as a 4D extended Dubins car system with the following dynamics that can approximately describe many systems that must travel in the direction of their heading in a curvature-constrained motion, such as differential drive robots, autonomous cars, fixed-wing drones, and aircraft.

$$\dot{z} = f(z, u, d) \tag{3.1}$$

$$\begin{aligned} \dot{x} &= v\cos(\phi) + d_x \\ \dot{y} &= v\sin(\phi) + d_y \\ \dot{\phi} &= \omega + d_\phi \\ \dot{v} &= a \end{aligned} \tag{3.2}$$

$v \in [0, \bar{v}]$ is the linear velocity, a is the linear acceleration, and ω is the angular velocity. $z := (x, y, \phi, v)$ is the system's state, and the system input (control) u is represented as $u := (a, \omega)$. Let $d := (d_x, d_y, d_\phi)$ be the disturbance that accounts for the error in the system modeling. In addition, we impose constraints on our control inputs and disturbances as follows:

$$\begin{aligned} a \in [-\bar{a}, \bar{a}], \omega \in [-\bar{\omega}, \bar{\omega}], \\ d_x^2 + d_y^2 \leq \bar{d}_{xy}^2, d_\phi \in [-\bar{d}_\phi, \bar{d}_\phi] \end{aligned} \tag{3.3}$$

3.1.2 Differential Flatness

Assume we are interested in moving between two states s_{init} and s_{final} in time T for our robot as an extended Dubins 4D car. Dynamics display a property called differential flatness which simplifies real-time feasibility analysis in trajectory planning [19, 22, 31]. Based on Eq. (3.6)-(3.16), one can follow to compute a trajectory $s(\cdot)$ given initial state s_{init}, final state s_{final}, and trajectory duration T.

In general form, we can write the solutions of the nonlinear flat system as functions of z and their q derivatives which completely determine the whole state and the inputs without the need to integrate the system. More formally, a system with state $s \in \mathbb{R}^n$ and inputs in $u \in \mathbb{R}^m$ is differentially flat if one can find outputs $z \in \mathbb{R}^m$ of the form

$$z = \alpha\left(s, u, \dot{u}, \ldots, u^{(p)}\right), \tag{3.4}$$

such that

$$\begin{aligned} s &= \beta\left(z, \dot{z}, \ldots, z^{(q)}\right), \\ u &= \gamma\left(z, \dot{z}, \ldots, z^{(q)}\right). \end{aligned} \tag{3.5}$$

The coordinates z are called flat outputs.
For extended Dubins car model, we know $\theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right)$ and $v = \sqrt{\dot{x}^2 + \dot{y}^2}$, thus:

$$\begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \\ \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \\ \sqrt{\dot{x}^2 + \dot{y}^2} \end{bmatrix} \tag{3.6}$$

From our dynamics, we have

$$\omega = \dot{\theta} = \frac{d}{dt}\arctan\left(\frac{\dot{y}}{\dot{x}}\right) = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{y}^2 + \dot{x}^2} \tag{3.7}$$

and

$$a = \dot{v} = \frac{d}{dt}\sqrt{\dot{x}^2 + \dot{y}^2} = \frac{1}{2v}(2\ddot{y}\dot{y} + 2\ddot{x}\dot{x}) \tag{3.8}$$

here flat output z is

$$z = (x, y) \tag{3.9}$$

as a result $q = 2$ and we have (v is not expanded for readability):

$$\begin{bmatrix} \omega \\ a \end{bmatrix} = \gamma(z, \dot{z}, \ddot{z}) = \begin{bmatrix} \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{y}^2 + \dot{x}^2} \\ \frac{1}{2v}(2\ddot{y}\dot{y} + 2\ddot{x}\dot{x}) \end{bmatrix} \tag{3.10}$$

The boundary conditions of differentially flat systems are expressed as:

$$\begin{aligned} s(0) &= \beta\left(z(0), \dot{z}(0), \ldots, z^{(q)}(0)\right) = s_{\text{init}}, \\ s(T) &= \beta\left(z(T), \dot{z}(T), \ldots, z^{(q)}(T)\right) = s_{\text{final}} \end{aligned} \tag{3.11}$$

they are specified entirely using the flat outputs. Thus, it is possible to perform trajectory generation in the flat output space.

Trajectories can be represented in simple functional forms such as splines. The general strategy is to assume a parametric form. Let

$$\begin{aligned} z(t) &= \sum_{i=1}^{N} b_i \psi_i(t) \Rightarrow \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t) \\ &\vdots \\ z^{(q)}(t) &= \sum_{i=1}^{N} b_i \psi_i^{(q)}(t) \end{aligned} \tag{3.12}$$

where $\psi(t) \in \mathbb{R}^N$ are basis functions.

Now from 3.12, we have:

$$\begin{bmatrix} \psi_1(0) & \psi_2(0) & \cdots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \cdots & \dot{\psi}_N(0) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(0) & \psi_2(0)^{(q)} & \cdots & \psi_N^{(q)}(0) \\ \psi_1(T) & \psi_2(T) & \cdots & \psi_N(T) \\ \dot{\psi}_1(T) & \dot{\psi}_2(T) & \cdots & \dot{\psi}_N(T) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(T) & \psi_2^{(q)}(T) & \cdots & \psi_N^{(q)}(T) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \vdots \\ z_1^{(q)}(0) \\ z_1(T) \\ \dot{z}_1(T) \\ \vdots \\ z_1^{(q)}(T) \end{bmatrix} \tag{3.13}$$

In compact form:

$$Z = B\Lambda \tag{3.14}$$

where

$$B = [b_1, b_2, ..., b_N]^T \tag{3.15}$$

To satisfy these boundary conditions, it is necessary that $N \geq 2(q+1)$. In particular, for $N = 2(q+1)$, the matrix is full-rank and one can directly compute B. As a result, we choose basis functions as follows to have a polynomial equation of the fifth-degree (quintic) spline trajectories:

$$\psi(t) = \left[t^5, t^4, t^3, t^2, t^1, 1\right]^T \tag{3.16}$$

Given B, in each time step $z(t)$ is specified and we can find $x(t)$ and $u(t)$ as described by Eq. (3.6)-(3.9), so the states and controls over the trajectory can be computed. Abstracting away this process into a function $\mathcal{S}$, we write $s(\cdot) = \mathcal{S}(s_{\text{init}}, s_{\text{final}}, T)$.

3.1.3 Hamilton-Jacobi Reachability Analysis

A natural question to ask is how to decide when the planning is safe and what learning objective can be used so that safe learning of high-level planning is encouraged during training. To answer those, we utilize a powerful theoretical tool used in the formal verification of dynamic systems, Hamilton-Jacobi (HJ) reachability analysis.

Given the system dynamics and a target set $\mathcal{T} \subseteq \mathbb{R}^4$, we compute a set of states where collision is inevitable, called Backward Reachable Tube (BRT) which is defined as follows:

$$\bar{\mathcal{A}} = \{z : \exists d(\cdot) \in \mathbb{D}, \forall u(\cdot) \in \mathbb{U}, \exists s \in [t, 0], \zeta(s; z, t, u(\cdot), d(\cdot)) \in \mathcal{T}\} \tag{3.17}$$

where $\zeta(s; z, t, u(\cdot), d(\cdot))$ is the system trajectory over time. Set of states where no matter what the control function is there exists a disturbance function that leads the system to collision. Typically, the target set can be represented by an implicit surface function $V_0(z)$ as $\mathcal{T} = \{z : V_0(z) \leq 0\}$. Then the BRT is the zero sublevel set of a value function $V(z, t)$

defined as follows:

$$V(z,t) = \max_{u(\cdot)\in\mathbb{U}} \min_{d(\cdot)\in\mathbb{D}} \min_{s\in[t,0]} V_0(\zeta(s;z,t,u(\cdot),d(\cdot))) \tag{3.18}$$

Given the system dynamics and target set $\mathcal{T}$ representing an obstacle map in the system's state space, computes BRTs and value functions $V(z,t)$. As the viscosity solution to the following HJ PDE ([10]):

$$\begin{gathered} \min\{D_s V(z,s) + H(z, \frac{\partial V(z,s)}{\partial z}), V(z,0) - V(z,s)\} = 0 \\ V(z,0) = V_0(z), s \in [t,0] \\ H(z, \frac{\partial V(z,s)}{\partial z}) = \min_d \max_u \frac{\partial V(z,s)}{\partial z}^\top f(z,u,d) \end{gathered} \tag{3.19}$$

where $f(z,u,d)$ is the system dynamics. In addition, we also define Forward Reachable Tubes (FRTs) as the set of dynamically feasible states the robot can arrive within time H seconds starting at a state z_0 under zero disturbance.

$$\mathcal{F}(T,z_0) = \{z : \exists u(\cdot), \text{ such that } z(\cdot) \text{ satisfies } f(z,u,d), z(0) = z_0; z(T) = z\} \tag{3.20}$$

Chapter 4

Method

4.1 Method

We employ a HJ Reachability-based framework, described in Section 4.1.2, to generate data. To train our model, described in section 4.1.3, the inputs are RGB image and current robot velocity, and outputs are decision boundary coefficients to classify safe and unsafe waypoints.

For data generation, we begin by rendering first-person view images based on the current state of the robot. Feasible waypoints are then identified, and trajectories are computed accordingly. We have a method based on HJ value functions to evaluate these trajectories and label the waypoints accordingly. We will delve into the specifics of each step in the subsequent sections.

4.1.1 BRT and FRT Computation

Forward reachability methods are employed to project potential future states. The HJ backward reachability analysis is utilized to determine the optimal safety value function and its corresponding safety controller for the robot. Drawing inspiration from prior research [20], the value function can be leveraged to assess the quality of waypoints and facilitate the learning of a safety layer for the agent. We will be using reachability analysis to compute both a backward reachable tube (BRT) and a forward reachable tube (FRT), given their target set $\mathcal{T}$, and the Hamiltonian H which captures the system dynamics as well as the roles of the control and disturbance.

The BRT is computed backward in time over an interval $[-T, 0]$. By setting a sufficiently large value for T, we can ensure the value function converges at the end. The initial condition for Eq. 3.19 is initialized by the obstacle set $\mathcal{O}$ of a map in the environment as follow:

$$V^{BRT}(z, t = 0) = \begin{cases} \text{-1} & \text{if } z \in \mathcal{O} \\ 1 & \text{if } z \notin \mathcal{O} \end{cases} \tag{4.1}$$

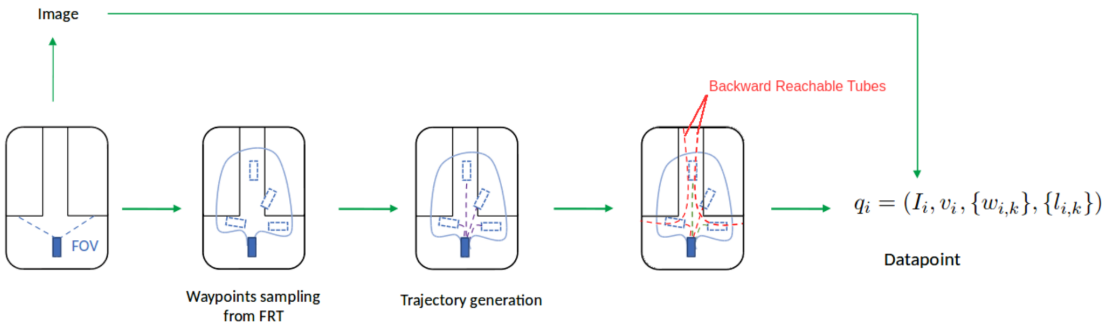


Figure 4.1: Overview of training data generation pipeline. Given a map and simulator, we first capture an RGB image at the camera pose, which is determined by the robot's position and heading (pose). Then, using our precomputed FRT, we sample possible waypoints the robot can arrive at the next H sections. For each waypoint, we can compute the trajectory toward that waypoint. Any trajectories that pass by the precomputed BRT will be labeled as unsafe and safe otherwise.

Computing the BRT requires access to the obstacle map. Also, since we only need one map to generate the training data, BRT only needs to be computed once.

The FRT is computed over a time interval $[0, H]$. Since the speed of the system is not included in the car's ego coordinate frame, we have to compute FRT for different starting speed. Due to the infinite number of speeds, pre-computing the FRT for each initial speed is impractical. Instead we discretize the speed range into bins v_i and set each of the bins as the initial state. To identify dynamically-feasible waypoints, FRT_i must be calculated for initial speed offline, v_i, considering a horizon H, 6 seconds in our case. The sub-zero level set consists of reachable states.

Initial value function input to FRT computation is defined as follows:

$$V^{FRT}(z, t = 0) \begin{cases} \leq 0 & \text{if } z = (0, 0, 0, v_i) \\ > 0 & \text{otherwise} \end{cases} \tag{4.2}$$

A small size 4D grid where is negative at the state of robot otherwise is positive. Notably, the dynamics remaining invariant under translation and rotation allow for a convenient transformation of the value function from the robot starting at the origin to a robot starting at any position and heading. More conveniently, we can assume the robot begins at the coordinates $(0, 0, 0, v_i)$. Initially, the value function is negative only at $(0, 0, 0, v_i)$ when t = 0. Subsequently, we calculate the FRTs and map them to the original pose and heading of the robot. The time horizon is set sufficiently long (10 seconds); we compute long enough for the BRT to converge so that we get the infinite time horizon BRT, $T = \infty$. BRT is computed for a whole map once and a portion of that is cropped and loaded based on the current state of the robot.

Both BRT and FRT are computed using the OptimizedDP toolbox [7]. The OptimizedDP toolbox encompasses implementations of dynamic programming-based algorithms in optimal control. Implemented on a multidimensional grid, computational complexities increase exponentially as the dimensionality increases. The toolbox facilitates the creation of a Cartesian grid, implemented as a Python object, through specifications such as the number of grid nodes, upper and lower bounds for each dimension, and periodic dimension.

4.1.2 Data Generation

As shown in Fig. 4.1, we spawn a robot at a random state in simulation and collect training data. We follow Alg. 1 to generate data, where we randomly choose the starting state and sample sets of waypoints from the forward reachable tube (FRT) for the state (Line 2-4). For each waypoint, the system's dynamics, as outlined in Eq. (3.2) is used to compute dynamically feasible trajectories as third-order spline using $z(\cdot) = \mathcal{S}(z_{\text{init}}, z_{\text{final}}, H)$. This leads to a smooth, dynamically feasible, and computationally efficient trajectory to the waypoint. We evaluate each of these trajectories on the value function representing the BRT, by taking the minimum over time of its value function (Line 6-8).

Algorithm 1 Data Generation for Learning Vision-based Decision Boundary Estimation

Require: System dynamics: Eq. (3.2)
Require: FRT and value function representing BRT corresponding to given map
Require: t current time and H time horizon

1: **for** i = 1 to N **do**
2: Sample an initial state $z_i : (x_i, y_i, \theta_i, v_i)$
3: Render current image I_i
4: Sample K waypoints using relative FRT $\hat{W}_i := (x^r_{i,k}, y^r_{i,k}, \theta^r_{i,k}, v^r_{i,k})_{\{k=1:K\}}$
5: Convert the relative waypoints $\hat{W}_i$ into absolute states
6: **for** $\hat{w}_{i,k} \in \hat{W}_i$ **do**
7: $\{z, u\}_{t:t+H} = \mathcal{S}(z_i, \hat{w}_k, H)$
8: $V_{\min} \leftarrow \min\Big(V(z(t)), \ldots, V(z(t+H))\Big)$
9: **if** $V_{\min} > \epsilon$ **then**
10: $l_{i,k} \leftarrow 1$
11: **else**
12: $l_{i,k} \leftarrow -1$
13: **end if**
14: $D \leftarrow D \cup \{(I_i, v_{i,k}, \hat{w}_{i,k}), l_{i,k}\}$
15: **end for**
16: **end for**

We consider the values over the trajectory instead of just the waypoint's value because it is safer to plan short-term with a lower time horizon (H) as we have a limited camera Field of View (FoV) and do not have depth information. A positive minimum value signifies a safe waypoint ($l_k = 1$), indicating that the trajectory from the present state will avoid collisions.

Conversely, a minimum value that drops below zero implies an unsafe waypoint ($l_k = -1$), suggesting that the trajectory might lead to collisions ((Line 9-12). We repeat the above procedure for different initial states until sufficient data-label pairs are obtained for the training dataset represented as D, where K is the number of waypoints per image/initial state (Line 14). The entire data set is written as: $\{q_i\}_{i\in\{1,\dots,M\}}$ and $q_i = (I_i, v_i, \{w_{i,k}\}, \{l_{i,k}\})$, where $k \in \{1, \dots, K\}$ for each i. Thus, each data point q_i, q_i, includes for each image i, some corresponding speed, and a collection of K waypoints with their corresponding labels.

4.1.3 Training

As shown in Fig. 3.1, we use ResNet-50 as our CNN backbone for the perception module. Our system is agnostic to the choice of the backbone network. We choose ResNet-50 because of its reasonable size and faster training time. The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, which reduces the number of parameters and matrix multiplications to enable faster training of each layer. The image features obtained at the last convolution layer are concatenated with the current linear speed before passing them to the fully connected layers, which generates the weights for degree 3 polynomial kernel $\hat{y}_\psi$. The dot product of $\hat{y}_\psi$ and polynomial mapping of a waypoint $\phi(w_k)$ where ϕ is the polynomial kernel function. This gives the logit (log-odds) scores of the waypoint being unsafe, which are used to calculate the SVM-based hinge loss on a set of waypoints. Here $k \in \{1 \dots K\}$ is the waypoint index. Applying sigmoid on the logit scores gives us the probabilities in the range $[0, 1]$, and by applying a threshold on the probabilities, we can classify waypoints into safe and unsafe classes.

For every set of waypoints, the decision boundary calculated by SVM based on ground truth labels (without considering the image itself) is represented by y_{svm_i}. The loss function Eq. (4.3), combines three terms: hinge loss, which maximizes the margin between decision boundary and waypoint classes and is less sensitive to outliers; cosine distance loss to maximize the proximity between $\hat{y}_{\psi_i}$ and y_{svm_i}; and the SVM weights regularization term to prevent overfitting. Note that cosine distance loss is a number between 0 and 1, where values closer to 0 indicate greater similarity. Once trained, the neural network can robustly transfer to novel and unknown environments.

$$\mathcal{L} = L_{\text{hinge}} + \lambda_1 L_{\text{reg}} + \lambda_2 L_{\text{cosine distance}} \tag{4.3}$$

Where $L_{\text{hinge}} = \sum_{k=1}^{K} \max(0, 1 - l_k \hat{y}_\psi^\top \phi(w_k))$, $L_{\text{reg}} = \frac{1}{2}(\hat{y}_\psi)^2$, and $L_{\text{cosine distance}} = (1 - \frac{\hat{y}_\psi \cdot y_{\text{svm}}}{\|\hat{y}_\psi\|_2 \|y_{\text{svm}}\|_2})$

Chapter 5

Simulation Results

Dataset: We use the photorealistic Stanford's large-scale 3D Indoor Spaces Dataset as described in [2]. This dataset consists of a large-scale indoor environment including six indoor areas with 271 rooms for a total of 695 million points. These rooms cover office areas, educational and exhibition spaces, conference rooms, personal offices, restrooms, open spaces, lobbies, stairways, and hallways. S3DIS dataset contains mesh scans of several Stanford buildings. By rendering this mesh at any state, we can obtain the image observed by the camera as well as the occupancy information within the robot's FoV. As described in Sec. 4.1.2, we spawn a robot at many different random states, whereby the robot's onboard camera captures a 224×224 pixel RGB image, denoted as I_t. We set $\bar{v} = 0.6$ m/s, $\bar{\omega} = 1.1$ rad/s, and $\bar{a} = 0.4$ m/s^2 to align with the specifications of the Turtlebot 2 employed in the hardware experiments (Sec. 6.1). We choose $\bar{d}_{xy} = 0.05$ m/s and $\bar{d}_{\phi} = 0.15$ rad/s. The FoV of the camera is set to 62.1° and the time horizon H for trajectory generation is 6 seconds. The perception module takes I_t and the linear speed v_t as input and outputs the parameters for the decision boundary. We train on the generated dataset from Area 3 and Area 5 and test our model on Area 4 in simulation and real-world without an *a priori* known map.

Implementation details: Our implementation utilizes a pre-trained ResNet-50 model, specifically trained for ImageNet Classification, to encode the input image. We remove the final fully-connected layer and instead add six fully-connected layers to regress to the parameters of the decision boundary. We train the ResNet-50 network [15] with $N = 175,000$ data points from the reachability expert. The inference time for one image is, on average, 0.1 seconds. To optimize the loss function, we employ the Adaptive Moment Estimation (ADAM) algorithm with a learning rate of 10^{-5} and loss weights $\lambda_1 = \lambda_2 = 10^{-2}$ for cosine distance loss and SVM regularization loss.

Experimental results: In Fig. 5.1, the initial plot on the top left plot illustrates a top-down view of the map, with obstacles and free areas depicted in yellow and blue, respectively. The robot is positioned at the bottom center, with the forward direction pointing upwards (↑). A subset of trajectories leading to waypoints is then overlaid onto the scene. Note that the sets of waypoints are different in the two rows because they are sampled

from FRT for each initial speed. A slice of the sub-zero level set of the value function V representing the BRT $\bar{\mathcal{A}}$ taking at the current robot heading and speed are displayed as the dashed cyan line. The BRT is computed using the obstacles, shown as the solid black lines as the target set $\mathcal{T}$. The waypoints depicted as red crosses are unsafe, while green ones are safe, plus headings are shown with the arrows. We can discern differences between obstacles and the sub-zero level set, especially at higher speeds (bottom left), because the BRT considers both robot dynamics and disturbance to determine the waypoints' labels. By comparing these trajectories with the BRT, especially at higher speeds, we highlight how our model accounts for dynamics and disturbances.

The second column of plots shows the projection of viewpoints onto first-person view images at the ground plane. Safe and unsafe waypoints are again differentiated by green and red colors, respectively. The third column of the plot delineates the decision boundary generated by our model, highlighting safe and unsafe areas. Our model reveals larger unsafe regions at higher speeds which is depicted in the bottom row, aligning with our expectations based on system dynamics: it is harder to avoid nearby obstacles at a higher speed. Fig. 5.2 depicts a similar concept at a different position. This shows our model accounts for the robot velocity, unlike object segmentation methods. The labels of waypoints stem from trajectories rooted in dynamic equations, an attribute unattainable by image segmentation models.

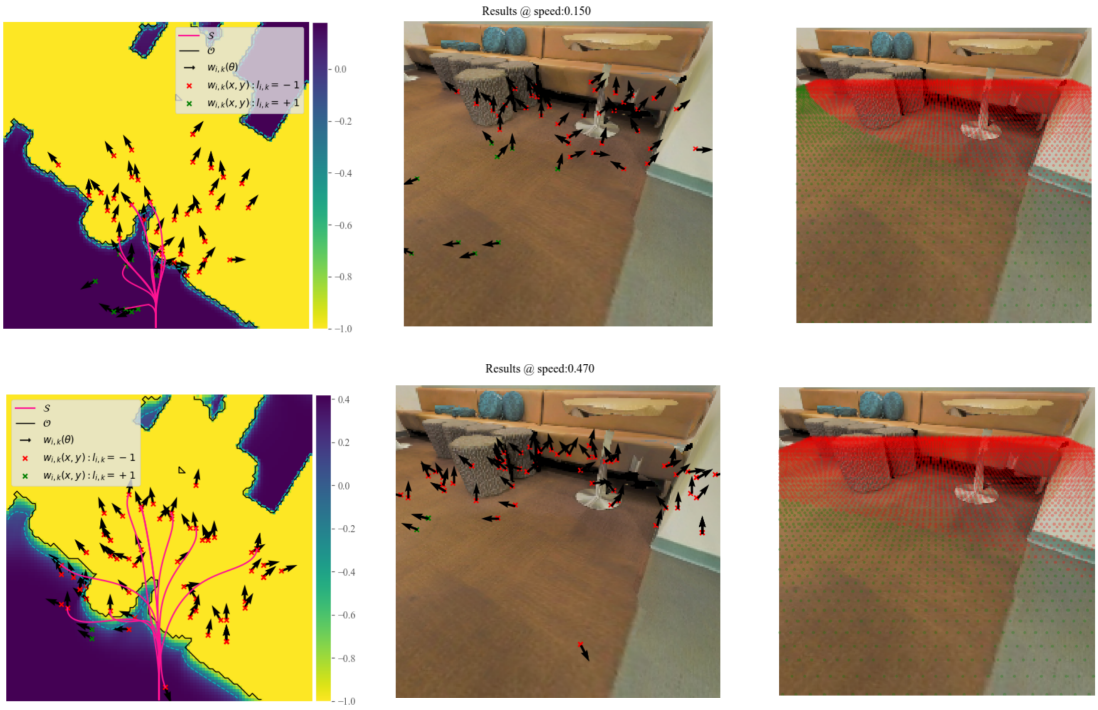


Figure 5.1: Plots for different speeds. From left to right: top-down view, first-person view, and decision boundary from our model

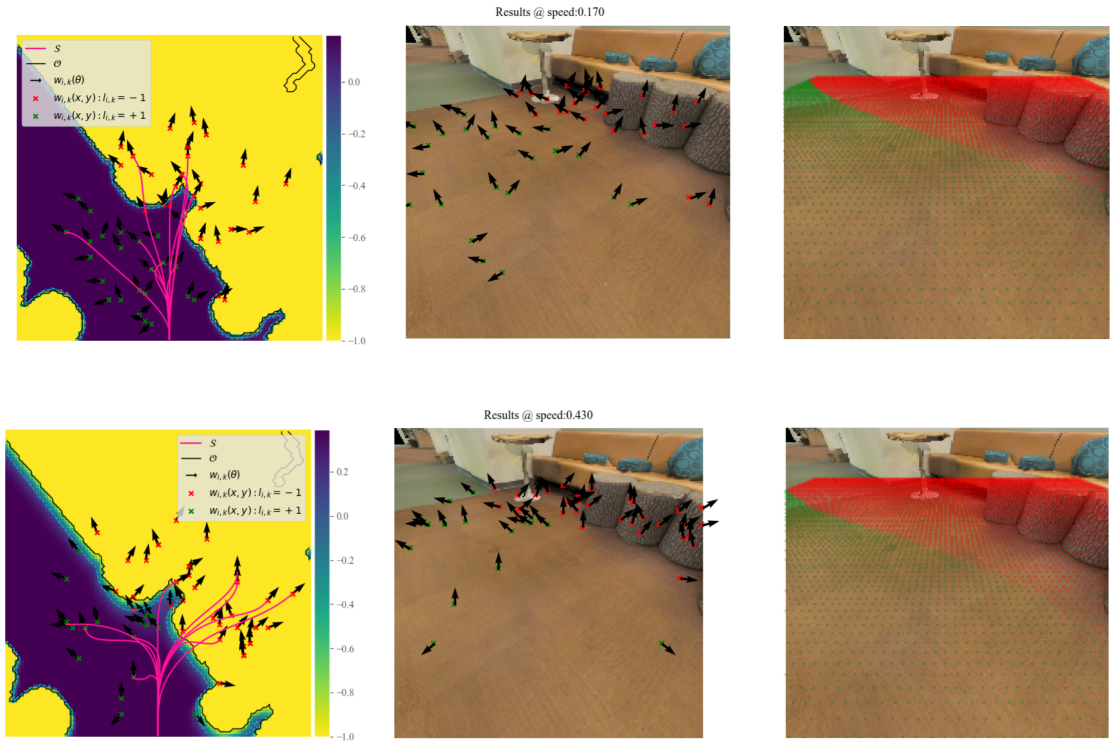


Figure 5.2: Plots for different speeds. From left to right: top-down view, first-person view, and decision boundary from our model

Metrics: For quantitative evaluation, we use accuracy, precision, recall, and F1-score as our metrics, with the unsafe label being the positive class.

Baseline: We compare our approach against a straightforward classification model with hinge loss. The model takes image, velocity, and waypoint as input and predicts corresponding labels as output. A ResNet-50 is used to extract a feature vector from the image, which is concatenated with the velocity and waypoint and fed to an MLP to obtain the label for the waypoint. This baseline does not predict the explicit decision boundary.

Outputting decision boundary parameters offers advantages over solely focusing on the safety of individual waypoints: having the parameters makes the decision boundary easier to visualize, and decision boundaries span the entire feature space of waypoints, enabling the model to predict values for any input combination. Comprehending these boundaries yields valuable insights into the decision-making process and facilitates diagnostic analysis. Visualizing decision boundaries showcases the model's sensitivity to data points. SVMs typically exhibit smooth boundaries and demonstrate lower sensitivity to dataset variations compared to other models.

In order to deal with an imbalanced dataset we use various classification metrics, focusing on F-1 scores. Our goal is to accurately classify unsafe labels as the positive class. Table 5.1 illustrates a comparative analysis of metrics between learning the decision boundary through

our model (with and without the cosine distance loss as our ablation study) and the more straightforward classification model as our baseline. The lack of improvement in performance with cosine distance loss can be attributed to the fact that the SVM decision boundary is based on a limited number of waypoints and does not take into account the characteristics of the images, making it prone to misclassification. Our model demonstrates comparable metrics, with fewer parameters but with enhanced user interpretability compared to the baseline.

Metrics				
Model	Accuracy	precision	recall	**F1-score**
Ours	85.7	86.7	91.2	**88.9**
Ours w/o cosine distance loss	85.5	87.6	89.4	**88.5**
Baseline	86.0	85.8	93.8	**89.6**

Table 5.1: Quantitative Comparison of Classification Metrics for Our Method and Baseline in Simulation

Chapter 6

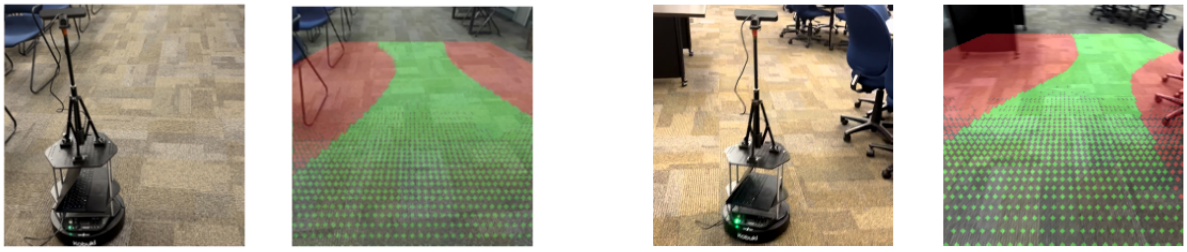
Hardware Experiments

We evaluated the model on the Turtlebot 2 robot without undergoing additional training or fine-tuning. The experiments were conducted in various locations within Simon Fraser University's Technology Science Complex 1 (TASC1) building, specifically in room 9204. None of these locations were part of the original training set. The threshold is set higher than 0.5, considering the importance of accurately detecting unsafe labels rather than potentially mislabeling safe waypoints, reflecting a cautious approach.

6.1 Hardware Experiments

We conducted tests on our framework using a Turtlebot 2 hardware testbed, as shown in the left images of Fig. 6.1a and Fig. 6.1b. The tests utilize an onboard StereoLabs ZED2 camera to capture RGB images and wheel encoders to obtain robot velocity to support navigation. We crop 224x224 images with a FoV of 62.1°, consistent with the simulated images.

For these experiments, we teleoperate the robot and compute decision boundaries for each image frame, shown in the right images of Fig. 6.1a and Fig. 6.1b. The decision boundary is expressed in a closed form, therefore determining the sign value of the dot product between decision boundary parameters and a data point, allows us to identify the positive and negative sides of the contour. This visualization is done using SVM decision boundary which is faster compared to the baseline method because we evaluate against the polynomial instead of running neural network inference on each point.



a b

Figure 6.1: We present two scenarios: a and b. In each pair, left image is the snapshots of the real robot in third-person view and the right is the decision boundary in first-person view

We choose the threshold for classification of 0.5 on the probabilities to classify safe waypoints from unsafe. Depending on our safety criteria, we have the flexibility to modify the threshold for classification, allowing us to control the proximity of the decision boundary to obstacles. The images showcase tests conducted in both simulation and real-world environments, providing quantitative evaluations for the former and qualitative assessments for the latter. Video footage of all experiments can be found at: `https://www.youtube.com/watch?v=3ySt0V79FYE&list=PLUBop1d3Zm2sdaiYb0Gme9PxJGqpKvVPB`

Chapter 7

Conclusion

In conclusion, our research tries to address the robot safety challenge as they traverse through unfamiliar environments. The cornerstone of our approach lies in the development of a novel perception model designed to improve the safety of navigation tasks. Our study harnesses the capabilities of visual sensors to proactively predict safe and unsafe high-level actions via Hamilton-Jacobi reachability analysis. Notably, the model is trained using data derived from optimal control techniques, incorporating a loss function based on support vector machines.

This work involves the development of a self-supervised learning method aimed at predicting the decision boundary between safe and unsafe waypoints. By leveraging HJ reachability analysis, a theoretical basis is established to label training datasets. Neural networks are then empowered to learn the coefficients of the decision boundary through the application of a soft-margin SVM loss function. Notably, the method's versatility and effectiveness are demonstrated through its successful application to real-world scenarios, showcasing its generalizability.

In future endeavors, we will explore concrete downstream applications stemming from this work. For instance, showcasing real-world applications to illustrate the practicality of this method, such as a semi-autonomous navigation system featuring a graphical user interface (GUI) that suggests safe waypoint commands for controlling ground robots. Further, integrating this method as a safety layer for waypoint-based policies to improve their metrics and delving into more advanced architectures like transformers to enhance the results can serve as promising next steps.

Bibliography

[1] Claus Aichinger, Philippe Nitsche, Rainer Stütz, and Marko Harnisch. Using low-cost smartphone sensor data for locating crash risk spots in a road network. *Transportation research procedia*, 14:2015–2024, 2016.

[2] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1534–1543, 2016.

[3] Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1758–1765. IEEE, 2019.

[4] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. *arXiv preprint arXiv:1903.02531*, 2019.

[5] Javier Borquez, Kensuke Nakamura, and Somil Bansal. Parameter-conditioned reachable sets for updating safety assurances online. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10553–10559. IEEE, 2023.

[6] David J Bruemmer, Ronald L Boring, Douglas A Few, Julie L Marble, and Miles C Walton. " i call shotgun!": an evaluation of mixed-initiative control for novice users of a search and rescue robot. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 3, pages 2847–2852. IEEE, 2004.

[7] Minh Bui, George Giovanis, Mo Chen, and Arrvindh Shriraman. Optimizeddp: An efficient, user-friendly library for optimal control and dynamic programming, 2022.

[8] J.L. Burke and R.R. Murphy. Human-robot interaction in usar technical search: two heads are better than one. In *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, pages 307–312, 2004.

[9] J. Casper and R.R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385, 2003.

[10] Mo Chen and Claire J. Tomlin. Hamilton–jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):333–358, 2018.

[11] Yilun Chen, Palanisamy Praveen, Mudalige Priyantha, Katherina Muelling, and John Dolan. Learning on-road visual control for self-driving vehicles with auxiliary tasks. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 331–338. IEEE, 2019.

[12] Barzin Doroodgar, Yugang Liu, and Goldie Nejat. A learning-based semi-autonomous controller for robotic exploration of unknown disaster scenes while searching for victims. *IEEE Transactions on Cybernetics*, 44(12):2719–2732, 2014.

[13] Emre Eraslan, Yildiray Yildiz, and Anuradha M Annaswamy. Shared control between pilots and autopilots: An illustration of a cyberphysical human system. *IEEE Control Systems Magazine*, 40(6):77–97, 2020.

[14] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *IROS*, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 2018.

[17] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[18] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 international conference on robotics and automation (ICRA)*, pages 6008–6014. IEEE, 2019.

[19] T John Koo and Shankar Sastry. Differential flatness based full authority helicopter control design. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 2, pages 1982–1987. IEEE, 1999.

[20] Anjian Li, Somil Bansal, Georgios Giovanis, Varun Tolani, Claire Tomlin, and Mo Chen. Generating robust supervision for learning-based visual navigation using hamilton-jacobi reachability. In *Learning for Dynamics and Control*, pages 500–510. PMLR, 2020.

[21] Antonio Loquercio, Ana I Maqueda, Carlos R del Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[22] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[23] Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Neural autonomous navigation with Riemannian motion policy. *arXiv preprint arXiv:1904.01762*, 2019.

[24] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladen Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

[25] Robin R Murphy. Activities of the rescue robots at the world trade center from 11-21 september 2001. *IEEE Robotics & Automation Magazine*, 11(3):50–61, 2004.

[26] Derek J Phillips, Juan Carlos Aragon, Anjali Roychowdhury, Regina Madigan, Sunil Chintakindi, and Mykel J Kochenderfer. Real-time prediction of automotive collision risk from monocular video. *arXiv preprint arXiv:1902.01293*, 2019.

[27] Charles Richter, William Vega-Brown, and Nicholas Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Robotics Research*, pages 325–341. Springer, 2018.

[28] Róbert-Adrian Rill and Kinga Bettina Faragó. Collision avoidance using deep learning-based monocular vision. *SN Computer Science*, 2(5):375, 2021.

[29] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[30] Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie N Zeilinger. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

[31] Rahee Walambe, Nipun Agarwal, Swagatu Kale, and Vrunda Joshi. Optimal trajectory generation for car-type mobile robot using spline interpolation. *IFAC-PapersOnLine*, 49(1):601–606, 2016.

[32] Blake Wulfe, Sunil Chintakindi, Sou-Cheng T Choi, Rory Hartong-Redden, Anuradha Kodali, and Mykel J Kochenderfer. Real-time prediction of intermediate-horizon automotive collision risk. *arXiv preprint arXiv:1802.01532*, 2018.

[33] Ehsan Yousefi, Dylan P Losey, and Inna Sharf. Assisting operators of articulated machinery with optimal planning and goal inference. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2832–2838. IEEE, 2022.

Appendix A

Code

Codes of all experiments can be found at this repository:

`https://github.com/SFU-MARS/Safety-reachability`