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

Tara Toufighi TTOUFIGH@SFU.CA
Minh Bui MINH_BUI_3@SFU.CA
Rakesh Shrestha RAKESHS@SFU.CA
Mo Chen MOCHEN@SFU.CA
School of Computing Science, Simon Fraser University, Burnaby BC, Canada, V5A 1S6

Editors: A. Abate, M. Cannon, K. Margellos, A. Papachristodoulou

Abstract

We develop a self-supervised learning method that predicts 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 the decision boundary via a soft-margin Support Vector Machine loss function. We experimentally show that our method is generalizable to the real world and generates safety decision boundaries in unseen indoor environments without knowledge of the obstacle map. Our method's advantages are its explainability and accurate safety prediction, which are important as an aid to human operators in semi-autonomous systems. Finally, we demonstrate our method's ability to transfer to the real-world without additional training.

1. Introduction

Navigating robots through intricate and congested 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 Murphy (2004), most notably in challenging environments with clutter, poor illumination, and narrow paths Bruemmer et al. (2004). Consequently, human involvement remains indispensable for the operation of such vehicles. Semi-autonomous robots have found applications in myriad scenarios, from search and rescue Doroodgar et al. (2014) to assistive robots Yousefi et al. (2022) and flight control Eraslan et al. (2020). 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 Casper and Murphy (2003); Burke and Murphy (2004). 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.

In the wake of the success of deep learning, visual navigation has gained prominence, even in safety-critical robotic scenarios. Visual navigation boasts advantages such as affordability, lightweight hardware, and ubiquity. Within the research community focused on learning, robot

navigation is typically investigated in the context of an agent exploring an environment that is also deemed unknown. In this context, a system is designed to acquire policies that directly correlate onboard sensor readings with control commands through an end-to-end approach. Such methodologies offer various advantages, enabling the learning of policies without a prior understanding of the specific system or environment the robot will navigate. The fundamental concept involves training a model based on convolutional neural networks (CNNs) utilizing high-level policies. These policies leverage current RGB image observations to generate a sequence of intermediate states, referred to as "waypoints". 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. In this paper, we propose a deep learning-based method to identify safe sets of waypoints for robot navigation. 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 Li et al. (2020).

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; 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, which improves interpretability in classifying safe and unsafe waypoints; and 3) a hardware demonstration of our approach, showcasing zero-shot transfer of the learned safe decision boundary estimation based on monocular RGB images and current linear speed.

2. Related Works

Navigation generally involves metric maps and classical path planning. Obtaining such maps can be challenging in many scenarios. Numerous research endeavors have explored deep learning-based monocular vision solutions to tackle autonomous robot navigation without using metric maps Li et al. (2020); Bansal et al. (2019); Kang et al. (2019); Gandhi et al. (2017); Kahn et al. (2017); Sadeghi and Levine (2016). The direct perception paradigm, as outlined in Chen et al. (2019), 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 approaches, exemplified by Rill et al. (Rill and Faragó (2021)), directly map input images to actuator actions.

Despite recent advances in autonomous navigation through monocular vision, limitations remain. 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 (Aichinger et al. (2016); Phillips et al. (2019); Wulfe et al. (2018)), 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 (Aichinger et al. (2016)), 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 (Rill and Faragó (2021)). 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 (Wabersich et al. (2023); Li et al. (2020); Richter et al. (2018); Borquez et al. (2023); Jung et al. (2018); Loquercio et al. (2018); Müller et al. (2018); Meng et al. (2019); Bansal et al. (2019); Bajcsy et al. (2019)) also seek to address these challenges.

3. Problem Setup

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. 1. At each time step t, our learning architecture takes an RGB image I_t representing the first-person view of the robot, along with the robot velocity v_t, as input, and 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 $(x(t), y(t), \theta(t), v(t))$ 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. 1.

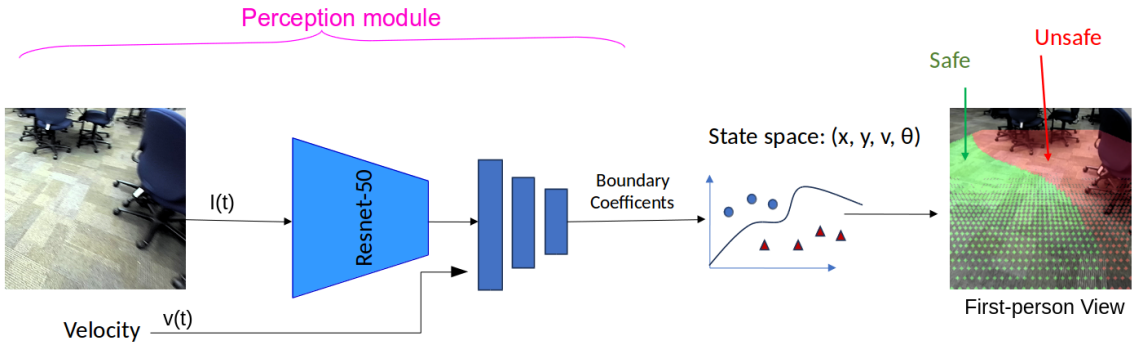


Figure 1: Overview of the system components

3.1. Dynamics Modeling

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

$$\dot{x} = v\cos(\theta) + d_x \quad \dot{y} = v\sin(\theta) + d_y \quad \dot{\theta} = \omega + d_\theta \quad \dot{v} = a \tag{1}$$

$v \in [0, \bar{v}]$ is the linear velocity, a is the linear acceleration, and ω is the angular velocity. $z := (x, y, \theta, v)$ is the system's state, and the system input (control) u is represented as $u := (a, \omega)$. Let $d := (d_x, d_y, d_\theta)$ be the disturbance that accounts for the error in the system modeling. Note that we assume that the disturbance follows an non-anticipative strategy that react to the control input based on the state as described in Chen and Tomlin (2018). In addition, we impose constraints on our control inputs and disturbances as follows:

$$a \in [-\bar{a}, \bar{a}], \omega \in [-\bar{\omega}, \bar{\omega}], \quad d_x^2 + d_y^2 \leq \bar{d}_{xy}^2, d_\theta \in [-\bar{d}_\theta, \bar{d}_\theta] \tag{2}$$

3.2. Differential Flatness

The dynamics in Eq. (1) are differentially flat, so trajectory path planning can be computed efficiently as trajectories can be represented in simple functional forms such as splines. This simplification facilitates real-time feasibility in trajectory planning Koo and Sastry (1999); Mellinger and Kumar (2011). In this section, we present the differential flatness of the extended Dubins Car system.

Using Eq. 1, we have $\theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right)$ and $v = \sqrt{\dot{x}^2 + \dot{y}^2}$. From there we can express the control input as follows:

$$\omega = \dot{\theta} = \frac{d}{dt} \arctan\left(\frac{\dot{y}}{\dot{x}}\right) = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{y}^2 + \dot{x}^2} \text{ and } a = \dot{v} = \frac{d}{dt}\sqrt{\dot{x}^2 + \dot{y}^2} = \frac{1}{2v}(2\ddot{y}\dot{y} + 2\ddot{x}\dot{x}) \tag{3}$$

Based on Eq. (3), one can follow Walambe et al. (2016), which is fully explained in the Appendix [1] section, to compute a trajectory $z(\cdot)$ given initial state z_{init}, final state z_{final}, and trajectory duration T'. Abstracting away this process into a function $\mathcal{S}$, we write $z(\cdot) = \mathcal{S}(z_{\text{init}}, z_{\text{final}}, T')$.

3.3. 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 minimal Backward Reachable Tube (BRT) which is defined as follows:

$$\bar{\mathcal{A}} = \{z : \exists \gamma \in \Gamma(t), \forall u(\cdot) \in \mathbb{U}, \exists s \in [t, 0], \zeta(s; z, t, u(\cdot), \gamma[u](\cdot)) \in \mathcal{T}\} \tag{4}$$

where $\zeta(s; z, t, u(\cdot), \gamma[u](\cdot))$ is the system trajectory over time, $\gamma[u](\cdot)$ is the disturbance, with $d(\cdot) = \gamma[u](\cdot)$ lying in $\Gamma(t)$, the set of non-anticipative disturbances that satisfy disturbance constraints. The minimal BRT is the set of states where no matter what the control function is there exists a disturbance function that leads the system to a target set $\mathcal{T}$ representing 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:

$$V(z, t) = \inf_{\gamma[u](\cdot) \in \Gamma(t)} \sup_{u(\cdot) \in \mathbb{U}} \min_{s \in [t, 0]} V_0(\zeta(s; z, t, u(\cdot), \gamma[u](\cdot))) \tag{5}$$

1. Appendix link

Given the system dynamics and target set $\mathcal{T}$ representing an obstacle map in the system's state space, we can obtain the BRT through computing value functions $V(z,t)$ as the viscosity solution to the following HJI PDE (Chen and Tomlin (2018)):

$$\begin{aligned} \min\{D_s V(z,s) + H\left(z, \frac{\partial V(z,s)}{\partial z}\right), V(z,0) - V(z,s)\} = 0 \\ V(z,0) = V_0(z), s \in [t,0] \\ H\left(z, \frac{\partial V(z,s)}{\partial z}\right) = \max_u \min_d \frac{\partial V(z,s)}{\partial z}^\top f(z,u,d) \end{aligned} \tag{6}$$

where $f(z,u,d)$ is the system dynamics described by Eq. 1. In addition, we also define Forward Reachable Tubes (FRTs) as the set of dynamically feasible states the robot can arrive within time T seconds starting at a state z_0; we don't consider disturbance in this case.

$$\mathcal{F}(T, z_0) = \{\mathbf{z} : \exists u(\cdot), \text{ such that } z(\cdot) \text{ satisfies } f(z,u), z(0) = z_0; z(\tau) = \mathbf{z} \text{ for } \tau \in [0,T]\} \tag{7}$$

We compute BRTs to convergence, resulting in the infinite horizon BRT represented by the converged value function $V_\infty(z) := \lim_{t\to\infty} V(z,t)$. All BRTs and FRTs are computed using the OptimizedDP toolbox Bui et al. (2022), which implements dynamic programming-based algorithms in optimal control on a multidimensional grids for obtaining numerically convergent (globally) optimal solutions. The toolbox parallelizes the computation for fast training data generation.

4. Method

We employ a HJ Reachability-based framework, briefly introduced in Section 3.3, to generate data (Section 4.1). HJ value functions accurately quantify the safety of waypoints by taking into account the worst-case disturbances; we invite readers to refer to Chen and Tomlin (2018); Bansal et al. (2017) for a more in-depth discussion of HJ reachability. A waypoint is an intermediate state representing a short-term navigational goal within our framework, which serve as a link between perception and control. In section 4.2, we explain how we train our model to learn a decision boundary between safe and unsafe waypoints using an RGB image and current robot velocity. Differing from the work of Bui et al. (2021), the safe/unsafe boundary is predicted using only RGB images as input and can be computed efficiently onboard.

4.1. Data Generation

As shown in Fig. 2, we spawn a robot at a random state in simulation to collect training data. We follow Alg. 1 to generate data by sampling sets of waypoints from the FRT for the state (Line 2-4). For each waypoint, the system's dynamics, as outlined in Eq. (1) is used to compute dynamically feasible trajectories as third-order spline using $z(\cdot) = \mathcal{S}(z_{\text{init}}, z_{\text{final}}, T')$. 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 (Lines 6-8).

Note that we consider the values over the trajectory instead of just the value on the waypoint because a robot may enter an unsafe region on the way to a safe waypoint; this consideration is especially important 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

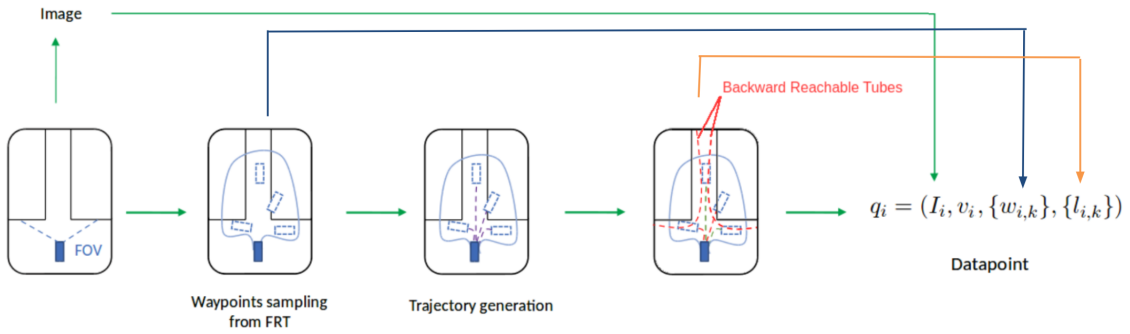


Figure 2: Overview of training data generation pipeline. Given a map and simulator, we first capture an RGB image at the camera pose, which is determined by the robot's position and heading (pose). Then, using our precomputed FRT, we sample possible waypoints the robot can arrive at the next T seconds. 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.

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

Require: System dynamics: Eq. (1), FRT and value function representing BRT corresponding to given map, t current time and T' time horizon

for i = 1 to N **do**
  Sample an initial state $z_i : (x_i, y_i, \theta_i, v_i)$
  Render current image I_i
  Sample K waypoints using relative FRT $\hat{W}_i := (x^r_{i,k}, y^r_{i,k}, \theta^r_{i,k}, v^r_{i,k})_{\{k=1:K\}}$
  Convert the relative waypoints $\hat{W}_i$ into absolute states
  for $\hat{w}_{i,k} \in \hat{W}_i$ **do**
    $\{z, u\}_{t:t+T} = \mathcal{S}(z_i, \hat{w}_k, T)$
    $V_{\text{min}} \leftarrow \min\left(V^{\text{BRT}}_{\infty}(z(t)), \ldots, V^{\text{BRT}}_{\infty}(z(t+T'))\right)$
    if $V_{\text{min}} > \epsilon$ **then**
      $l_{i,k} \leftarrow 1$
    else
      $l_{i,k} \leftarrow -1$
    end if
    $D \leftarrow D \cup \{(I_i, v_{i,k}, \hat{w}_{i,k}), l_{i,k}\}$
  end for
end for

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 (Lines 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 $q_i = (I_i, v_{i,k}, \{w_{i,k}\}, \{l_{i,k}\})$, where $k \in \{1, \dots, K\}$ for each i.

4.2. Training

As shown in Fig. 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 training time. At the last convolution layer, the image features obtained are concatenated with the current linear speed before passing them to the fully connected layers, which generates the weights for the decision boundary, denoted by $\hat{y}_\psi$. The product $\hat{y}_\psi^\top \phi(w_k)$, where $\phi(w_k)$ is a feature of w_k (chosen to be a degree 3 polynomial), gives the logit scores of the waypoint being unsafe[2]. These are used to calculate the soft-margin SVM-based hinge loss on a set of waypoints. Applying sigmoid on the logit scores gives us the probabilities in range $[0, 1]$, and by applying a threshold on the probabilities, we can classify waypoints to safe and unsafe classes.

For every set of waypoints, we also compute the optimal soft-margin SVM parameters y_{svm_i} based on ground truth labels; these parameters are incorporated into our overall loss function in Eq. (8), which combines three terms (with relative weights λ_1, λ_2): hinge loss, which maximizes the margin between the labels and the predicted decision boundary; cosine distance loss to minimize the angle difference between $\hat{y}_{\psi_i}$ and y_{svm_i}; and the SVM weights regularization term to alleviate overfitting. Once trained, the neural network can robustly transfer to novel and unknown environments without finetuning, as we demonstrate in our hardware experiments.

$$\mathcal{L} = L_{\text{hinge}} + \lambda_1 L_{\text{reg}} + \lambda_2 L_{\text{cosine distance}}, \text{ where} \tag{8}$$

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

5. Summary of Simulation Results

Dataset: We use the photorealistic Stanford's large-scale 3D Indoor Spaces Dataset as described in Armeni et al. (2016). As described in Sec. 4.1, 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). We choose $\bar{d}_{xy} = 0.05$ m/s and $\bar{d}_\theta = 0.15$ rad/s. The FoV of the camera is set to $62.1°$ and the time horizon T 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 our model using dataset available in simulation and test in an arbitrary real-world environment without a *a priori* map.

Implementation details: Using a pre-trained ResNet-50 He et al. (2016) model, we further train the network with $N = 175,000$ data points from the reachability expert. To optimize the loss

2. $\phi(\cdot)$ represents the waypoint feature, as opposed to θ, which represents the heading of the robot.

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.

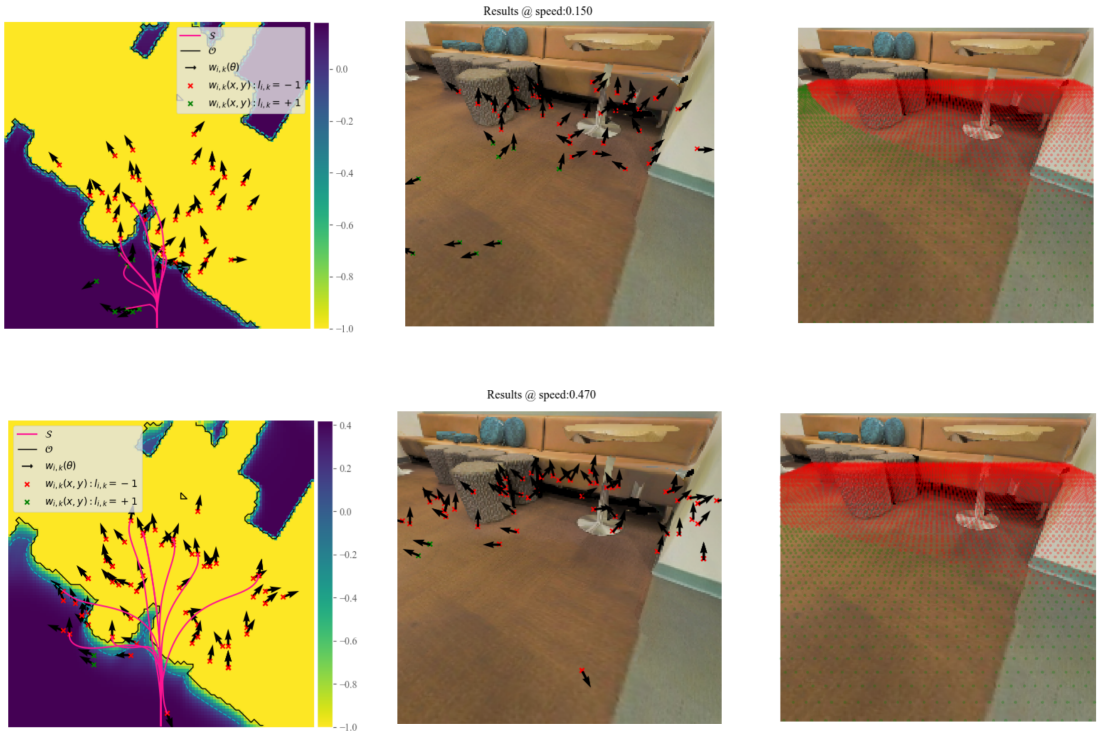


Figure 3: Plots for different speeds. From left to right: top-down view, first-person view, and decision boundary from our model

In Fig. 3, the plot on the top left 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. A slice of the sub-zero level set of the value function V representing the BRT sliced 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. This validates our choice to assess trajectories using the value function derived from BRT. Note that the set of waypoints are different in the two rows because they are sampled from FRT for each initial speed. 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 plot delineates the decision boundary generated by our model, highlighting safe and unsafe areas. Our model reveals larger unsafe regions at higher speeds (bottom row), aligning with our expectations based on system dynamics: it is harder to avoid nearby obstacles at a higher speed. Fig. 4 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.

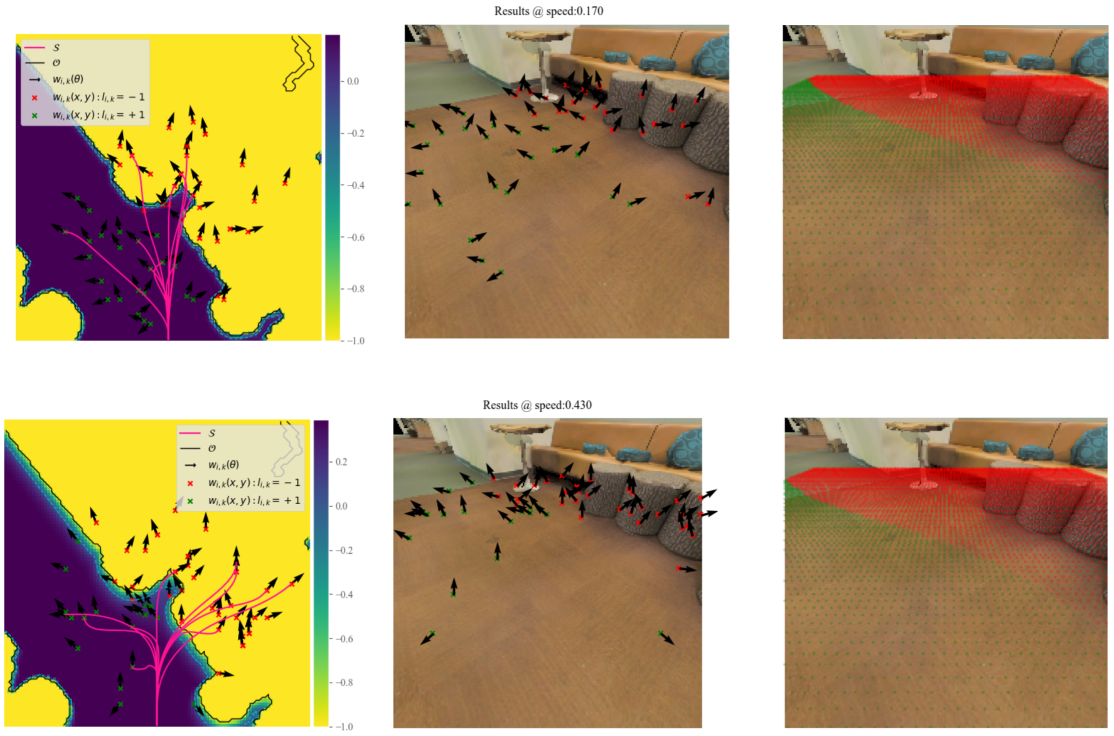


Figure 4: Plots for different speeds. From left to right: top-down view, first-person view, and decision boundary from our model

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.

Table 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 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.

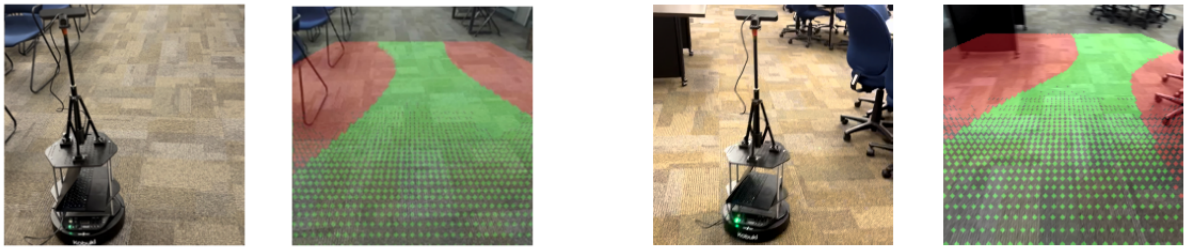
5.1. Testing

We evaluated the model on the Turtlebot 2 without further training or fine-tuning at various indoor locations at Simon Fraser University, which did not appear in the training dataset.

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 1: Comparison of classification metrics for our method and baseline in simulation

6. Hardware Experiments



a b

Figure 5: 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 conduct tests on our framework using a Turtlebot 2 robot hardware testbed, as shown in the left images of Fig. 5a and Fig. 5b. The tests utilize an onboard StereoLabs ZED2 camera to capture RGB images and wheel encoders to obtain robot velocity to support navigation. For these experiments, we tele-operate the robot and compute decision boundaries for each image frame, shown in the right images of Fig. 5a and Fig. 5b. The decision boundary is expressed in a closed form polynomial, which makes computation faster than evaluating safety for each waypoint individually. The presented 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

7. Conclusion

In conclusion, our research has addressed safe robot navigation in novel environments by evaluating safety boundary of the system's state space via Hamilton-Jacobi reachability analysis and support vector machine. Other techniques, such as control barrier functions, can also be used; however, we chose to use grid-based HJ reachability computations as they are especially convenient when applied to large, arbitrary obstacle maps. Even though HJ reachability analysis is computationally expensive, recent advances in decomposition methods Chen et al. (2018) make safety analysis of more high-dimensional systems feasible. Since our method is deep-learning based, safety guaranteed of our method will be further investigated. In future work, we can consider concrete downstream applications of this work, such as a semi-autonomous navigation system.

Acknowledgements

This work was supported by the NSERC Discovery Program and the CIFAR AI Chairs Program. Besides, T. Toufighi received support from SFU Graduate Dean Entrance Scholarship.

References

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.

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.

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.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. 2017.

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.

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.

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.

Minh Bui, Michael Lu, Reza Hojabr, Mo Chen, and Arrvindh Shriraman. Real-time hamilton-jacobi reachability analysis of autonomous system with an fpga. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1666–1673, 2021. doi: 10.1109/IROS51168.2021.9636410.

Minh Bui, George Giovanis, Mo Chen, and Arrvindh Shriraman. Optimizeddp: An efficient, user-friendly library for optimal control and dynamic programming, 2022. URL https://arxiv.org/abs/2204.05520.

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. doi: 10.1109/ROMAN.2004.1374778.

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. doi: 10.1109/TSMCB.2003.811794.

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. doi: 10.1146/annurev-control-060117-104941.

Mo Chen, Sylvia L. Herbert, Mahesh S. Vashishtha, Somil Bansal, and Claire J. Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018. doi: 10.1109/TAC.2018.2797194.

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.

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.

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.

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

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.

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.

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

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.

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.

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.

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.

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.

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

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

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.

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.

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.

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

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

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.

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.

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.

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.