2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments

Andrea Bajcsy*, Somil Bansal*, Eli Bronstein, Varun Tolani, Claire J. Tomlin

***Abstract*— Real-world autonomous vehicles often operate in *a priori* unknown environments. Since most of these systems are safety-critical, it is important to ensure they operate safely even when faced with environmental uncertainty. Current safety analysis tools enable autonomous systems to reason about safety given full information about the state of the environment *a priori*. However, these tools do not scale well to scenarios where the environment is being sensed in real time, such as during navigation tasks. In this work, we propose a novel, real-time safety analysis method based on Hamilton-Jacobi reachability that provides strong safety guarantees despite the unknown parts of the environment. Our safety method is planner-agnostic and provides guarantees for a variety of mapping sensors. We demonstrate our approach in simulation and in hardware to provide safety guarantees around a state-of-the-art vision-based, learning-based planner. Videos of our approach and experiments are available on the project website[1].**

I. Introduction

Autonomous vehicles operating in the real world must navigate through *a priori* unknown environments using on-board, limited-range sensors. As a vehicle makes progress towards a goal and receives new sensor information about the environment, rigorous safety analysis is critical to ensure that the system's behavior does not lead to dangerous collisions. Such an analysis should take into account multiple sources of uncertainty, such as modelling error, external disturbances, and unknown parts of the environment.

A variety of mechanisms have been proposed to ensure robustness to modeling error and external disturbances [25], [17], [35]. Additionally, safety guarantees for systems using limited-range sensors in unknown environments have been investigated [22], [23], [33], [20]. However, the safety guarantees are typically provided by imposing specific assumptions on the sensor and/or the planner that are rather restrictive for real-world autonomous systems and sensors used for navigational purposes. In contrast, we aim to design a safety framework that is compatible with a broad class of sensors, planners, and dynamics.

There are two main challenges with providing such a framework. The first challenge relates to ensuring safety with respect to unknown parts of the environment and external disturbances while minimally interfering with goal-driven behavior. Second, real-time safety assurances need to be provided as new environment information is acquired, which requires approximations that are both computationally efficient and not overly conservative. Moreover, this safety analysis should be applicable to a wide variety of real-world sensors, planners, and vehicles.

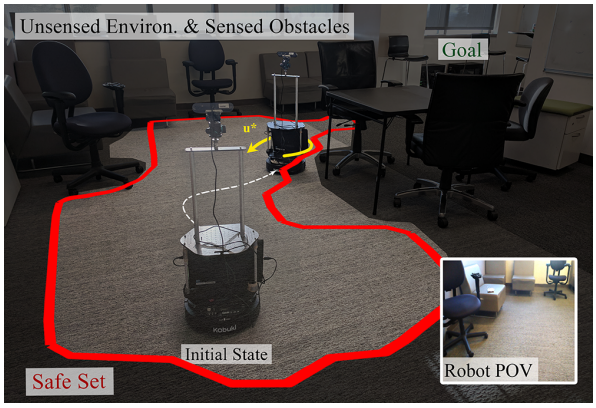


Fig. 1. **Overview:** We consider the problem of safe navigation from an initial state to a goal state in an *a priori* unknown environment. Our approach treats the unsensed environment as an obstacle, and uses a HJ reachability framework to compute a safe controller for the vehicle, which is updated in real-time as the vehicle explores the environment. We show an application of our approach on a Turtlebot using a vision-based planner. When the robot is at risk of colliding, the safe controller (u^*) keep the system safe.

In this paper, we propose a safety framework that can overcome these challenges for autonomous vehicles operating in *a priori* unknown static environments under the assumption that the sensors work perfectly within their ranges. Our framework is based on Hamilton Jacobi (HJ) reachability analysis [26], [28]. In particular, we treat the unknown environment at any given time as an obstacle and use HJ reachability to compute the *backward reachable set (BRS)*, i.e. the set of states from which the vehicle can enter the unknown and potentially unsafe part of the environment, despite the best control effort. The complement of the BRS therefore represents the safe set for the vehicle. With this computation, we also obtain the corresponding least restrictive safety controller, which does not interfere with the planner unless the safety of the vehicle is at risk. Use of HJ reachability analysis in our framework thus allows us to overcome the first challenge—our framework can be applied to general nonlinear vehicles, sensors, and planners.

However, due to the computationally expensive nature of HJ reachability, this approach is generally not leveraged in settings that require the BRS computation at run-time. To overcome this challenge, we propose a novel, real-time algorithm to compute the BRS. Our algorithm only locally updates the BRS in light of new environment information, which significantly alleviates the computational burden of HJ reachability while still maintaining the safety guarantees at all times. To summarize, our key contributions are:

*Equal contribution. All authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.
[1]Project website: https://abajcsy.github.io/safe_navigation/

1) a provably-safe framework for navigation in static unknown environments that is applicable to a broad class of sensors, planners, and dynamics,
2) an algorithm for online safe set updates from new sensor measurements as the robot navigates,
3) demonstration of our approach on different sensors and planners on a vehicle with nonlinear dynamics in the presence of external disturbances,
4) a hardware demonstration of our approach to provide safety around a state-of-the-art learning-based planner which uses only monocular RGB images for planning.

II. Related Work

A. Safe motion planning

Methods that ensure safety despite modeling error and disturbances are largely motivated by the trade-off between safety and efficiency during real-time planning. A popular approach is to perform *offline* computations that quantify disturbances and modeling error which can be used *online* to determine collision-free trajectories [25], [17], [35]. Alternatively, [2], [3] use control barrier functions to design provably stable controllers while satisfying given state-space constraints. However, these methods assume that a recursively feasible collision-free path can be obtained despite the unknown environment. Several works address this problem for single-agent scenarios within a model predictive control framework [30], [32], as well as for multiple vehicles using sequential trajectory planning [34], [6]. However, these works assume *a priori* knowledge of all obstacles, whereas our framework guarantees safety in an *a priori* unknown environments for potentially nonlinear dynamics.

Ensuring safety with respect to both modeling error and limited sensing horizons have been studied using sum-of-squares [22], linear temporal logic [23], reactive synthesis approaches [33], graph-based kinodynamic planner [20] among others. These works typically impose restrictions on sensors or planners to ensure safety with respect to the unknown environment. In contrast, the proposed framework is sensor and planner agnostic, provided that the sensor can accurately identify the obstacles within its sensing region.

B. Safe exploration

The problem of finding feasible trajectories to a specified goal in an unknown environment has also been studied in the robotic exploration literature for simplified kinematic motion models using frontier-exploration methods [37] and D* [21]. Other works include sampling-based motion planners for drift-less dynamics [9] and dynamic exploration methods for vehicles with a finite stopping time [18]. Robotic exploration has been also studied within the context of fully and partially observable Markov decision processes [31], [29] and reinforcement learning [19] to reduce collision probabilities; however, no theoretical safety guarantees are typically provided. Safe exploration has also been studied in terms of Lyapunov stability [10], [12]. Even though stability is often desirable, it is insufficient to guarantee collision avoidance. In contrast, our formulation uses a stronger definition of safety, and is more in line with [16], [14], which characterize safety using reachable sets.

III. Problem Statement

We study the problem of autonomous navigation in *a-priori unknown static* environments. Consider a stable, deterministic, nonlinear dynamics model of the vehicle $\dot{x} = f(x, u, d)$, where $x \in \mathbb{R}^n$, $u \in \mathcal{U}$, and $d \in \mathcal{D}$ represent the state, the control, and the disturbance experienced by the vehicle. Here, d can include the effect of both the external disturbances or dynamics mismatch. We assume that the flow field f satisfies the standard assumptions for the existence and uniqueness of trajectories [13]. We also assume that the vehicle state x can be accurately sensed at all times.

Let x_0 and x^* denote the start and the goal state of the vehicle. The vehicle aims to navigate from x_0 to x^* in an *a priori* unknown environment, $\mathcal{E}$, whose map or topology is not available to the robot. At any time t and state $x(t)$, the vehicle has a planner $\Pi(x(t), x^*, \mathcal{E})$, which outputs the control command $u(t)$ to be applied at time t. The vehicle also has a sensor which at any given time exposes a region of the state space[2] $\mathcal{S}_t \subset \mathbb{R}^n$, and provides a conservative estimate of the occupancy within $\mathcal{S}_t$. For example, if the vehicle has a camera sensor, $\mathcal{S}_t$ would be a triangular region (prismatic in 3D) representing the field-of-view of the camera. We assume perfect perception within this limited sensor range. Dealing with erroneous perception, sensor noise, and dynamic environments are problems in their own right, and we defer them to future work. Finally, we assume that there is a known initial obstacle-free region around x_0 given by $\mathcal{X}_{\text{init}} \subset \mathbb{R}^n$; e.g. this is the case when the vehicle is starting at rest and its initial state is collision-free.

Given x_0, x^*, $\mathcal{X}_{\text{init}}$, the planner Π, and the sensor measurements $\mathcal{S}$, the goal of this paper is to design a least restrictive control mechanism to navigate the vehicle to the goal state while remaining safe, which means avoiding obstacles at all times. Since the environment $\mathcal{E}$ is unknown, the safety needs to be ensured given the partial observations of the environment obtained through the sensor, which in general is challenging. We use the HJ reachability-based framework to ensure safety despite only partial knowledge of the environment.

IV. Running Example

To illustrate our approach, we introduce a simple running example: a 3D Dubins' car system with disturbances added to the velocity. The dynamics of the system are given by:

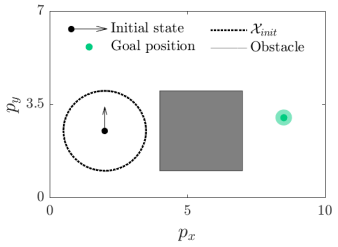
$$\begin{aligned} \dot{p}_x = v\cos\phi + d_x, \quad \dot{p}_y = v\sin\phi + d_y, \quad \dot{\phi} = \omega\,, \\ \underline{v} \le v \le \bar{v}, \quad |\omega| \le \bar{\omega}, \quad |d_x|, |d_y| \le d_r \end{aligned} \tag{1}$$

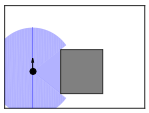
where $x := (p_x, p_y, \phi)$ is the state, $p = (p_x, p_y)$ is the position, ϕ is the heading, and $d = (d_x, d_y)$ is the disturbance experienced by the vehicle. The control of the vehicle is $u := (v, \omega)$, where v is the speed and ω is the turn rate.

[2]Typically, a sensor only exposes a part of the position space which is projected up in the entire state space.

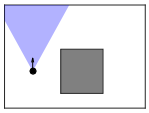
Both controls have a lower and upper bound, which for this example are chosen to be $\underline{v} = 0.1m/s$, $\bar{v} = 1m/s$, $\bar{\omega} = 1rad/s$. The disturbance bound is chosen as $d_r = 0.1m/s$.

The environment setup for the vehicle is shown in Fig. 2. The vehicle start and the goal state are given by $x_0 = [2, 2.5, \pi/2]$ (shown in black) and $x^* = [8.5, 3, -\pi/2]$ (the center of the green area). The goal is to reach within $0.3m$ of x^* (the light green area). However, there is an obstacle in the environment which is not known to the vehicle beforehand (shown in grey). We assume that there is no obstacle within $1.5m$ of x_0, and obtain the initial obstacle-free region $\mathcal{X}_{\text{init}} := \{x : \|p - p_0\| \leq 1.5\}$ (the area inside the dashed black line).





(a) LiDAR sensing region



(b) Camera sensing region

Fig. 2. The initial setup for the running example. The goal is to safely reach the goal (center of the green area) from the initial position (black marker) in the presence of an unknown obstacle (the grey square). We also show the initial sensing region for the LiDAR and camera sensors.

To demonstrate the sensor-agnostic nature of our approach, we simulate the Dubins' car with two different sensors: a LiDAR and a camera. For a LiDAR, the sensing region $\mathcal{S}_t$ is given by a circle of radius R centered around the current position $p(t)$, where $R = 3m$ in this simulation (shown in Fig. 2a). For a camera, the sensing region $\mathcal{S}_t$ is determined by a triangular region with solid angle F (also called the field-of-view) and apex at the current vehicle heading, and a maximum extent of R. We use $F = \pi/3$ and $R = 20m$ for our simulations (shown in Fig. 2b). However, part of the regions of $\mathcal{S}_t$ can be occluded by the obstacles, as would be the case for any real-world sensors.

Additionally, for each sensor, we demonstrate our approach on two different planners Π: a sampling-based RRT planner [24] and a model-based spline planner [36]. Our goal is to navigate to the goal despite the unknown obstacles.

V. Hamilton-Jacobi (HJ) Reachability

We use HJ reachability analysis to compute a backward reachable set (BRS) $\mathcal{V}(\tau)$ given a set of unsafe states $\mathcal{L}$ [26], [28], [7]. Intuitively, $\mathcal{V}(\tau)$ is the set of states such that the system trajectories that start from this set can enter $\mathcal{L}$ within a time horizon of τ for some disturbance despite the best control efforts. Conversely, for any trajectory that starts from $\mathcal{V}^c(\tau)$, there exists a control such that the system trajectory will *never* enter $\mathcal{L}$, despite the worst-case disturbance. Here, $\mathcal{V}^c(\tau)$ represents the complement of the set $\mathcal{V}(\tau)$.

The computation of the BRS can be formulated as a differential game between the control and disturbance, which can be solved using dynamic programming. Ultimately, a BRS can be computed by solving for the value function $V(\tau, x)$ in the following final value Hamilton Jacobi-Isaacs Variational Inequality (HJI-VI) [15], [26]:

$$\begin{aligned} &\min\{D_\tau V(\tau, x) + H(\tau, x, \nabla V(\tau, x)), l(x) - V(\tau, x)\} = 0 \\ &\quad V(0, x) = l(x), \quad \tau \leq 0. \end{aligned} \tag{2}$$

Here, $D_\tau V(\tau, x)$ and $\nabla V(\tau, x)$ denote the time and space derivatives of the value function respectively. The function $l(x)$ is the implicit surface function representing the unsafe set $\mathcal{L} = \{x : l(x) \leq 0\}$. The Hamiltonian, $H(\tau, x, \nabla V(\tau, x))$, encodes the role of system dynamics, control, and disturbance, and is given by

$$H(\tau, x, \nabla V(\tau, x)) = \max_{u \in \mathcal{U}_i} \min_{d \in \mathcal{D}} \nabla V(\tau, x) \cdot f(x, u, d). \tag{3}$$

Once the value function $V(\tau, x)$ is computed, the BRS, and consequently, the set of safe states are given by

$$\mathcal{V}(\tau) = \{x : V(\tau, x) \leq 0\}, \tag{4}$$

$$\mathcal{W}(\tau) = \mathcal{V}^c(\tau) = \{x : V(\tau, x) > 0\}. \tag{5}$$

HJI reachability also provides the optimal control to keep the system in the safe set and is given by

$$u^*(\tau, x) = \arg\max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \nabla V(\tau, x) \cdot f(x, u, d). \tag{6}$$

In fact, the system can safely apply any control within the safe region. If the system reaches the boundary of $\mathcal{V}(\tau)$, the control in (6) steers the system away from the unsafe states. This least restrictive controller provided by HJI reachability is also the basis for ensuring safety for any planner in a least restrictive fashion in our framework.

VI. Our Approach

We first describe our HJ-reachability-based framework to ensure safety in an *a priori* unknown environment. We then present a novel, real-time computation of a conservative approximation of the safe set based on new observations of the environment as the vehicle is navigating.

A. Ensuring safety in unknown environments

Our framework treats the unsensed environment at any given time as an obstacle. The unsensed environment along with the sensed obstacles are used to compute the vehicle's safe region using HJI reachability. This ensures that the vehicle never enters the unknown and potentially unsafe part of the environment, despite the worst case disturbance.

Let $\mathcal{F}_t$ denote the sensed obstacle-free region of the environment at time t. Given the initial obstacle-free region $\mathcal{F}_0 = \mathcal{X}_{\text{init}}$, we compute the safe set $\mathcal{W}_0$ by solving the HJI-VI in (2), assuming everything outside $\mathcal{F}_0$ is an obstacle. For this computation, the initial value function in (2) is given by $V_0(0, x) := l_0(x)$, where $l_0(x)$ is positive inside $\mathcal{F}_0$ and negative outside. One such function is given by the signed distance to $\mathcal{F}_0^c$. Starting from $l_0(x)$, the HJI-VI is solved to obtain the value function $V_0(x) := \lim_{\tau \to -\infty} V_0(\tau, x)$. Here, τ is the dummy computation variable in (2). $V_0(x)$ is then used to compute the safe region $\mathcal{W}_0$ (see (5)). As long as the

vehicle is inside $\mathcal{W}_0$, a controller exists to prevent it from colliding with known or unknown obstacles.

We next execute a controller on the system for the time horizon $t \in [0, T]$ as per the following control law:

$$u(t) = \begin{cases} \Pi(x(t), x^*, \mathcal{E}), & \text{if } x(t) \in \mathcal{W}_t \\ u^*(t, x(t)), & \text{otherwise} \end{cases} \tag{7}$$

where $u^*(t, x(t))$ is the optimal safe controller corresponding to the safe set $\mathcal{W}_t$ and is given by (6). Also, until the safe set is updated, we use the last computed safe set for finding the optimal safe controller, i.e., $\mathcal{W}_t = \mathcal{W}_0 \ \forall t \in [0, T]$. The control mechanism in (7) is least restrictive in the sense that it lets the planner execute the desirable control on the system, except when the system is at the risk of violating safety. Note that the control horizon T in our framework can be arbitrarily chosen by the system designer while still ensuring safety.

While the system is executing the control law in (7), it will obtain new sensor measurements $\mathcal{S}_t$ at each time t, which is used to obtain $\mathcal{F}_t$, the free space sensed at that time. If the sensor is completely occluded by an obstacle at any time, the corresponding free space is an empty set. Thus, the overall known free space at time t is given by: $\mathcal{M}_t = \bigcup_{s \in [0,t]} \mathcal{F}_\tau$. At the end of the control time horizon, we compute another safe region $\mathcal{W}_T$ assuming everything outside $\mathcal{M}_T$ is an obstacle. This safe region is obtained by solving HJI-VI until convergence. We then execute a control law similar to in (7), except that the safety controller intervenes only when the system is at the boundary of $\mathcal{W}_T$. The procedure is repeated until the system reaches the goal.

Since the safety controller does not allow the system trajectory to leave the known free space, the proposed framework is guaranteed to avoid collision at all times. However, the safe set can be rather conservative especially early on when most of the environment is still unexplored, which is a trade-off we make to ensure safety against all unexpected obstacles. If additional information about the obstacles in the environment is known, it can be incorporated and will only reduce the conservativeness of the safe set.

B. Efficient update of the BRS

Our framework requires the computation of a safe set in real-time as the vehicle is navigating through the environment. In general, this is challenging due to the exponentially scaling computational complexity of HJI reachability with respect to the state dimension [7]. To mitigate some of the computational challenges, we introduce two novel approaches to computing the BRS: warm-starting and local value function updates.

*1) **Warm-start approach**:* At any given time, the vehicle senses only a small additional part of the environment. Consequently, the free space map $\mathcal{M}_t$ only changes by a small amount in a small time horizon. Intuitively, this should only cause a small change in the safe region. We leverage this intuition to propose a novel, faster way to update the reachable set. For brevity, we explain our approach assuming that the safe set is updated every T seconds, but the same results hold when the safe set is updated at a non-fixed rate.

Given the last computed safe set at time t_{last}, and the maps at t_{last} and the current time t, we "warm-start" the value function in (2) for the BRS computation at time t as follows:

$$V_t(0, x) = \begin{cases} l_t(x), & \text{if } x \in \mathcal{M}_t \cap \mathcal{M}_{t_{\text{last}}}^c \\ V_{t_{\text{last}}}(x), & \text{otherwise} \end{cases} \tag{8}$$

where $l_t(x)$ as before is defined such that it is positive inside $\mathcal{M}_t$ and negative outside. Intuitively, instead of initializing the value function with $l_t(x)$ everywhere in the state space, (8) initializes it with the last computed value function for the states where no new information has been obtained since the last computation, and with $l_t(x)$ only at the states which were previously assumed to be occupied but are actually obstacle-free. This leads to a much faster computation of BRS because the value function needs to be updated only for a much smaller number of states that are newly found out to be free. At all the other states, the value function is already almost accurate and only small refinements are required. Interestingly, this procedure also maintains the conservativeness of the safe region, which is sufficient to ensure collision avoidance at all times.

Lemma 1: Let $\tilde{V}_t(\tau, x)$ be the solution of the following warm-started HJI-VI:

$$\min\{D_\tau \tilde{V}_t(\tau, x) + H(\tau, x, \nabla \tilde{V}_t(\tau, x)), l_t(x) - \tilde{V}_t(\tau, x)\} = 0,$$

where $\tilde{V}_t(0, x)$ is defined as in (8). Let $V_t(\tau, x)$ be the solution of the HJI-VI in (2) with $V_t(0, x) = l_t(x)$. Then $\tilde{V}_t(\tau, x) \leq V_t(\tau, x)$ for all $\tau \leq 0$. In particular, $\tilde{\mathcal{V}}_t(-\infty) \supseteq \mathcal{V}_t(-\infty)$ and $\tilde{\mathcal{W}}_t(-\infty) \subseteq \mathcal{W}_t(-\infty)$.

The proof of Lemma 1 can be found in [5]. Intuitively, Lemma 1 states that the safe set obtained by the warm-start approach is an under-approximation of the actual safe set obtained by solving full HJI-VI. Thus, it can be used to ensure safety for the vehicle while being computationally efficient. In practice, for the sensors and navigation problems in this paper, the amount of conservatism incurred by warm-starting is very small, as we demonstrate in Sec. VII. Our overall approach with warm-starting to update the safe set is summarized in Algorithm 1.

We start with the initial known free space $\mathcal{X}_{\text{init}}$ and compute the initial safe set $\mathcal{W}_0$ using HJI-VI (Line 6). The value function for this computation is initialized by the signed distance to $\mathcal{X}_{\text{init}}$. We also maintain the last computed BRS $\mathcal{V}_{\text{last}}$, the safe set $\mathcal{W}_{\text{last}}$, and the corresponding time t_{last} (Line 7). At every state, the vehicle obtains the current sensor observation and extracts the sensed free space (Line 9). Next, a control command is applied to the vehicle (Line 10). If the vehicle is inside $\mathcal{W}_{\text{last}}$, the planner is used to obtain the control command; otherwise, the safety controller is applied. Every T seconds, the safe set and controller are updated based on the free space sensed by the vehicle so far using HJI-VI (Line 14). The value function for this computation is warm-started with V_{last} except at the states which are discovered to be obstacle free since t_{last} as described in (8) (Line 13). The whole procedure is repeated until the vehicle reaches its goal.

Algorithm 1: Safe navigation using HJ reachability

1 x_0, x^*: Start and the goal states
2 $\mathcal{F}_0 \longleftarrow \mathcal{X}_{\text{init}}$: The initial obstacle-free region
3 $\mathcal{E}$: The unknown environment
4 $\Pi(\cdot, x^*, \mathcal{E})$: The planner for the vehicle
5 T: The control horizon
6 $\mathcal{W}_0$: The initial safe region obtained by solving HJI-VI in (2)
7 $\mathcal{W}_{\text{last}} \longleftarrow \mathcal{W}_0$; $\mathcal{V}_{\text{last}} \longleftarrow \mathcal{W}_0^c$; $t_{\text{last}} \longleftarrow 0$: The last computed safe set, BRS, and the corresponding time
8 **while** *the vehicle is not at the goal* **do**
9 Obtain the current sensor observation $\mathcal{S}_t$ and free space $\mathcal{F}_t$
10 Apply the least restrictive control $u(t)$ given by (7)
11 **for** ***every*** T *seconds* **do**
12 Obtain the current map $\mathcal{M}_t = \bigcup_{s\in[0,t]} \mathcal{F}_\tau$
13 Warm-start the BRS computation using $\mathcal{V}_{\text{last}}$, $\mathcal{M}_t$, and (8)
14 Obtain the new safe region, $\mathcal{W}_t$, by solving HJI-VI with the warm-started value function
15 $\mathcal{W}_{\text{last}} \longleftarrow \mathcal{W}_t$; $\mathcal{V}_{\text{last}} \longleftarrow \mathcal{W}_t^c$; $t_{\text{last}} \longleftarrow t$

Algorithm 2: Local update of the BRS

1 $\mathcal{Q} \longleftarrow \mathcal{M}_t \cap \mathcal{M}^c_{t_{\text{last}}}$: Initialize list of states for which the value function should be updated
2 $\mathcal{Q} \longleftarrow \mathcal{Q} \cup \mathcal{N}(\mathcal{Q})$: Add neighboring states to $\mathcal{Q}$
3 Warm-start the value for states in $\mathcal{Q}$, $V_t(0, \mathcal{Q})$, using (8)
4 $V_{\text{old}} \longleftarrow V_t(0, \mathcal{Q})$: The last computed value function for states in $\mathcal{Q}$
5 **while** $\mathcal{Q}$ *is not empty* **do**
6 $V_{\text{updated}} \longleftarrow$ Update the value function V_{old} for a time step ΔT
7 $\Delta V = \|V_{\text{updated}} - V_{\text{old}}\|$: Change in the value function
8 $\mathcal{Q}_{\text{remove}} \longleftarrow \{x \in \mathcal{Q} : \Delta V = 0\}$: States with unchanged value
9 $\mathcal{Q} \longleftarrow \mathcal{Q} - \mathcal{Q}_{\text{remove}}$: Remove states with unchanged value
10 $\mathcal{Q} \longleftarrow \mathcal{Q} \cup \mathcal{N}(\mathcal{Q})$: Add neighboring states to $\mathcal{Q}$
11 $V_{\text{old}} \longleftarrow V_{\text{updated}}$

*2) **Local update of the BRS**:* In the last section, we discussed how warm-starting the value function computation might lead to a faster convergence of the value function; however, the value function is still computed over the entire state space. In this section, we present a more practical algorithm that leverages the advantages of warm-starting by computing and updating the value function only locally at the states for which new information has been obtained since the last value function computation. Our safety framework is still same as what described in Algorithm 1—only the computational procedure for the safe set computation (Line 14 in Algorithm 1) is being modified to update the value function locally. We outline this procedure in Algorithm 2.

In Algorithm 2, we maintain a list of states $\mathcal{Q}$ at which the value function needs to be updated in light of the new environment observations. $\mathcal{Q}$ is initialized to be the set of states that are newly discovered to be free since t_{last}, i.e., $\mathcal{Q} = \mathcal{M}_t \cap \mathcal{M}^c_{t_{\text{last}}}$ (Line 1). Since the change in the value of the states in $\mathcal{Q}$ (compared to $V_{t_{\text{last}}}(x)$) would also cause a change in the value of the neighboring states, $\mathcal{N}(\mathcal{Q})$, we also add them to $\mathcal{Q}$ (Line 2). Thus, $\mathcal{Q} = \mathcal{Q} \cup \mathcal{N}(\mathcal{Q})$. Typically, the value function in HJI-VI is computed by discretizing the state-space into a grid and solving the VI over that grid. In such cases, the spatial derivative of the value function (required to compute the Hamiltonian in the HJI-VI in (2)) is computed numerically using the neighboring grid points. This spatial derivative is precisely responsible for the propagation of the change in the value function at a state to its neighboring states. In such cases, $\mathcal{N}(\mathcal{Q})$ might represent the neighboring grid points used to compute the spatial derivative of the value function for the states in $\mathcal{Q}$; however, other neighboring criteria can be used.

Once the neighbors are added to $\mathcal{Q}$, the value for all the states in $\mathcal{Q}$ is initialized as per (8) (Line 3), and their value is updated using HJI-VI in (2) for some time step ΔT (Line 6). This computation is much faster than classical HJI-VI computation since it is typically performed for many fewer states. Next, we remove all those states from $\mathcal{Q}$ whose value function hasn't changed significantly over this ΔT (Line 8 and 9), as these states won't cause any change in the value function for any other state. The neighbors of the remaining states are next added to $\mathcal{Q}$ (Line 10) and the entire procedure is repeated until the value function is converged for all states. Note that Algorithm 2 still maintains the conservatism of the safe set since it is just a different computational procedure for computing the warm-started value function, which is still used within the safety framework in Algorithm 1.

VII. Simulations

A. Running example revisited

We now return to our running example and demonstrate the proposed approach in simulation (described in Sec. IV). We implement our safety framework with three different methods to update the BRS: using the full HJI-VI, the warm-start approach (Sec. VI-B.1), and the local update approach (Sec. VI-B.2). The corresponding system trajectories for different planners and sensors for all the three methods are shown in Figure 3. For all combinations of planners and sensors, the proposed framework is able to safely navigate the vehicle to its goal position despite the external disturbances and no *a priori* knowledge of the obstacle (none of the trajectories go through the obstacle). As the vehicle navigates through the environment, the planner makes optimistic decisions at several states that might lead to a collision; however, the safety controller intervenes to ensure safety. States where the safety controller is applied are marked in red. Note that the safety controller intervenes more frequently for the camera sensor as compared to the LiDAR. This is because the field-of-view (FoV) of a camera is typically much narrower than a LiDAR (which senses the obstacles in all directions). Given this limited FoV, the safety controller needs to account for a much larger unexplored environment, which leads to more cautious control.

We compare the computation time required for each of the three methods to compute the BRS for the camera and LiDAR sensors in Table I. All computations were done on a MATLAB implementation on a desktop computer with a Core i7 5820K processor using the Level Set toolbox [27]. As expected, across all scenarios, warm-starting the value function for the BRS computation leads to a significant improvement in computation time compared to full HJI-VI; however, the computation time might still not be practical for most real-world applications. Only locally updating the value function in addition to warm-starting leads to a significant further improvement in the computation time, and the BRS

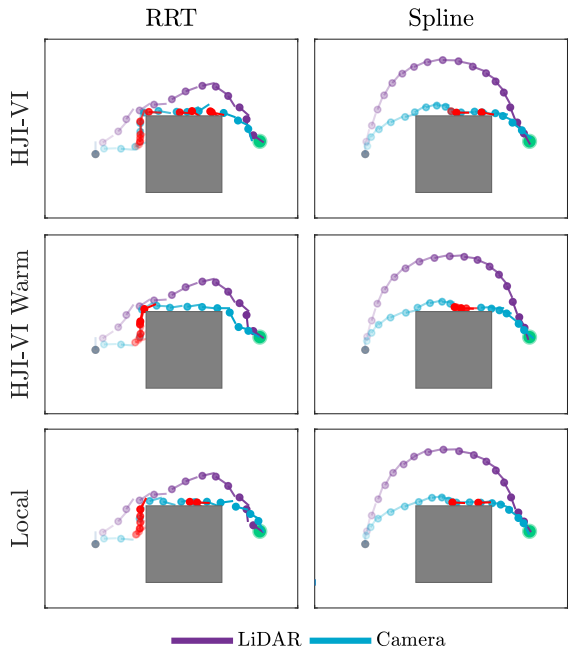


Fig. 3. The vehicle trajectories for the problem setting in Figure 2 for both planners (RRT and Spline planners) and both sensors (LiDAR and Camera sensors) with the safety controller computed from each of the three candidate safety approaches. The proposed framework is able to safely navigate the vehicle to the goal in all cases. When the planner makes unsafe decisions, the safety controller intervenes (the states marked in red) to ensure safety.

is updated in approximately 1s on average for all sensors and planners. This improvement is impressive considering that the computation was done in MATLAB without any parallelization which is known to decrease the computation time by a factor of 100 [11].

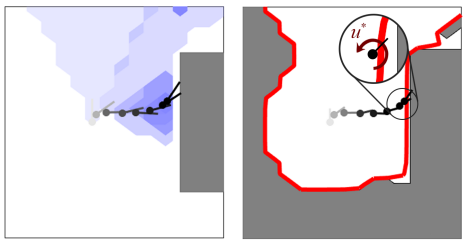
Lemma 1 indicates that the safe set obtained by warm-starting the value function is conservative compared to the one obtained by full HJI-VI. Therefore, we also compare the percentage volume of the states at which the safe set is conservative. This over-conservative volume is typically limited to 0.5% which indicates that the warm-starting approach is able to approximate the true value function quite well.

Finally, we take a closer look at how the safe control comes into play when the system is operating with a range-limited sensor. Fig. 4a showcases a Dubins' car with a camera sensor and an RRT planner, where the current robot state is shown in black, the corresponding sensed region is in dark blue, and the trajectory and corresponding sensed regions are shown in grey and light blue respectively. Since the camera's FoV is occluded by an obstacle at the current state, it cannot sense the environment past the obstacle. Figure 4b illustrates the corresponding current belief map $\mathcal{M}_t$ of the environment which is the union of the free space sensed by the vehicle so far (shown in white). Since the current sensed region is contained within the sensed region at the previous state, no new environment information is obtained and hence the BRS is not updated. The slice of the safe set at the current vehicle heading is shown in Fig. 4b (the area within the red boundary). Since the vehicle is at the safe set boundary, the safety controller intervenes and applies a control u^* that leads the system towards the interior

Simulated Camera Results				
Metric	Planner	HJI-VI	Warm	Local
Average Compute Time (s)	RRT	45.688	26.290	0.596
	Spline	51.723	12.489	0.898
% Over-conservative States	RRT	0.0	1.112	0.517
	Spline	0.0	0.474	0.506

Simulated LiDAR Results				
Metric	Planner	HJI-VI	Warm	Local
Average Compute Time (s)	RRT	21.145	6.075	1.108
	Spline	25.318	3.789	1.158
% Over-conservative States	RRT	0.0	0.032	0.290
	Spline	0.0	0.024	0.240

TABLE I. Numerical comparison of average compute time and relative volume of over-conservative states for each planner and sensor across different BRS update methods. Local updates compute an almost exact BRS in ≈1 second, and are significantly faster than both HJI-VI and warm-start.



(a) (b)

Fig. 4. (a) The sensed region by the vehicle at different states in time for the camera sensor. (b) The overall free space sensed by the vehicle and the corresponding safe set (interior of the red boundary). Since the vehicle is at the boundary of the safe set, the safety controller u^* is applied to steer the robot inside the safe set and ensure collision avoidance.

of the safe set (the red arrow) to ensure collision avoidance.

B. *Safety for a learning-based planner*

Since the proposed safety framework is planner-agnostic, we can use it to ensure safe navigation even in the presence of a learning-based planner. In particular, we use the vision-based planner proposed in [8], which takes an RGB camera image and the goal position as input, and uses a Convolutional Neural Network-based perception module to produce a desired next state that moves the robot towards its goal while trying to avoid obstacles on its way. This desired next state is used by a model-based low-level planner to produce a smooth trajectory from the vehicle's current state to the desired state. The authors demonstrate that the proposed planner can leverage robot's prior experience to navigate efficiently in novel indoor cluttered environments; however, it still leads to collisions in several real-world scenarios, like when the vehicle needs to go through narrow spaces. We use the proposed safety framework to ensure both safe and efficient planning in such difficult navigation scenarios.

The task setup for our simulation is shown in Fig. 5a. The robot needs to go through a very narrow hallway, followed by a door into the room to reach its goal (the green circle) starting from the initial state (black arrow). Initially, the robot

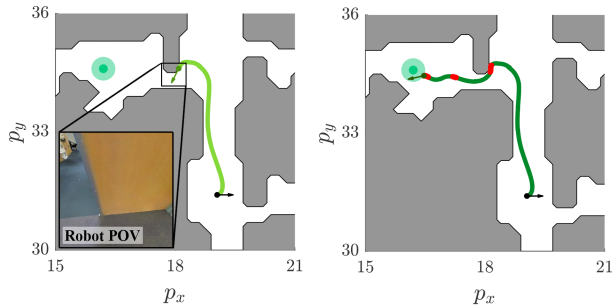


Fig. 5. The proposed framework can be exploited to provide safety guarantees around vision-based planners that incorporate learning in the loop. The vision-based planner plans a path through the doorway. Without safety control (a) this results in collision, however with safety (b) the robot avoids collision and reaches the goal.

has no knowledge about the obstacles (shown in dark grey). We simulate this scenario using the S3DIS dataset which contains mesh scans of several Stanford buildings [4]. By rendering this mesh at any state, we can obtain the image observed by the camera (used by the planner) as well as the occupancy information within the robot's FoV (used for the safety computation). For the robot dynamics, we use the 4D Dubins' car model:

$$\dot{p}_x = v\cos\phi, \quad \dot{p}_y = v\sin\phi, \quad \dot{v} = a, \quad \dot{\phi} = \omega \tag{9}$$

where $p = (p_x, p_y)$ is the position, ϕ is the heading, and v is the speed of the vehicle. The control is $u := (a, \omega)$, where $|a| \leq 0.4$ is the acceleration and $|\omega| \leq 1.1$ is the turn rate.

The trajectory taken by the learning-based planner in the absence of the safety module is shown in Fig. 5a. Even though the vehicle is able to go through the narrow hallway, it collides with the door eventually. The trajectory taken by the vehicle when the planner is combined with the proposed safety framework is shown in Fig. 5b. When the planner takes an unsafe action near the door, the safety controller intervenes (marked in red) and guides the robot to safely go through the doorway. We also illustrate the image observed by the robot near the doorway in Fig. 5a. Even though most of the robot's vision is blocked by the door, the planner makes a rather optimistic decision of moving forward and leads to a collision. In contrast, the safety controller makes a conservative decision of rotating in place to explore the environment more before moving forward, and eventually goes through the doorway to reach the goal.

VIII. Experiments

We test the proposed approach in hardware using a TurtleBot 2 with a mounted stereo RGB camera. For the vehicle state measurement, we use the on-board odometry sensors on the TurtleBot. In our experiment, the vehicle needs to navigate through an unknown cluttered indoor environment to reach its goal (shown in Fig. 1). For the BRS computation, we use the dynamics model in (9). We pre-map the environment using an open-source Simultaneous Localization and Mapping (SLAM) algorithm and the on-board stereo camera. This pre-mapping step is used to avoid the significant delay and inaccuracies in the real-time SLAM map update.

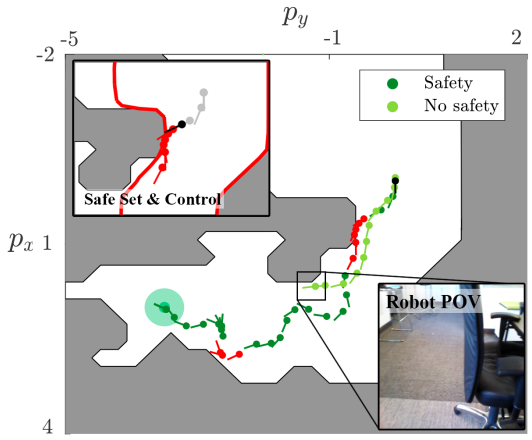


Fig. 6. We show the top-view of our experiment setting, and the corresponding system trajectories with and without the proposed safety framework. Without the safety framework, the robot collides into the chair. In contrast, our safety framework is able to safely navigate the robot to its goal by intervening when the vehicle is too close to the obstacles.

However, the full map is not provided to the robot during deployment. Instead, for the safe set computation at any given time, the current FoV of the camera is projected on the SLAM map and only the information within the FoV is used to update the safe set. This emulates the limited sensor range of the Turtlebot's camera. Regardless, this alludes to one of the important and interesting future research directions of ensuring safety despite sensor noise.

For planning, we use the learning-based planner described in Sec. VII-B that uses the current RGB image to determine a candidate next state. A top-view of our experiment setting is shown in Fig. 6. The vehicle starting position and heading are shown in black, the goal region is shown in green, and the obstacles (unknown to the vehicle beforehand) are shown in grey. We ran the experiment with and without the safety controller and show the corresponding trajectories in Fig. 6. Without the safety controller, the learning-based planner struggles with making sharp turns near the corner, and eventually collides into the obstacle (the chair, in this case). For context, we also show the RGB observation received by the planner near the corner. Even though the robot is very close to the chair, the planner makes the unsafe decision of continuing to move forward. However, when the learning-based planner is used within the proposed safety framework, the safety controller is able to account for this unsafe situation and safely steer the vehicle away from the obstacle. We show the corresponding safe set when the vehicle is at the obstacle boundary and the corresponding vehicle trajectory obtained using the safety controller. Afterwards, the planner takes over and steers the vehicle to the goal.

IX. Practical Considerations

We now discuss some practical considerations we encountered while using the proposed framework. Since the value function is computed over a discretized state space, it might incur some numerical inaccuracies. We found that 3rd or higher order approximations schemes work well for the spatial derivative. Due to the complicated geometry of real-world obstacles, the sensed map could be highly

irregular, which can significantly hamper the value function computation. Thus, it might be desirable to convert the occupancy map into a regular, well-behaved function, such as a signed distance map, before the value function computation. Theoretically speaking, the safety controller only needs to be applied at the safe set boundary. However, due to numerical inaccuracies, it might be desirable to apply the safety controller at a positive level of the value function.

Further computational saving can be obtained by leveraging narrow-band level-set methods [1], which update the value function only around the zero level set boundary. One can use the local update method only around the zero-level, thus combining the advantages of both methods.

X. Conclusion And Future Work

We propose an HJ reachability-based safety framework for provably safe navigation in *a priori* unknown environments. Our framework is applicable to a wide variety of planners and sensors. To overcome the computation complexity of classical HJ reachability analysis, a novel, real-time algorithm is proposed to update the reachable set as the vehicle traverses the environment. We demonstrate our approach on multiple sensors and planners, including a learning-based planner, both in simulation and on a hardware testbed.

Several future directions emerge from this work, such as overcoming perfect state estimate and sensor assumptions and extending to dynamic, multi-agent environments.

Acknowledgments

This research is supported in part by the DARPA Assured Autonomy program under agreement number FA8750-18-C-0101, by NSF under the CPS Frontier project VeHICaL project (1545126), and by SRC under the CONIX Center, and by Berkeley Deep Drive. The authors would also like to thank Kene Akametalu, Ellis Ratner, and Anca Dragan for their helpful advice.

References

[1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of computational physics*, 118(2):269–277, 1995.

[2] A. Agrawal and K. Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *RSS*, 2017.

[3] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[4] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *arXiv preprint*, 2017.

[5] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. *arXiv preprint*, 2019.

[6] S. Bansal, M. Chen, J. Fisac, and C. J. Tomlin. Safe sequential path planning under disturbances and imperfect information. In *ACC*, 2017.

[7] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *CDC*, 2017.

[8] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining optimal control and learning for visual navigation in novel environments. *arXiv preprint*, 2019.

[9] K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *ICRA*, 2007.

[10] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. *NIPS '17*.

[11] M. Chen, S. Bansal, K. Tanabe, and C. J. Tomlin. Provably safe and robust drone routing via sequential path planning: A case study in san francisco and the bay area. *arXiv preprint*, 2017.

[12] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *NIPS '18*.

[13] E. A. Coddington and N. Levinson. *Theory of ordinary differential equations*. McGraw-Hill, New York, 1955. pp. 1–42.

[14] J. Fisac, A. Akametalu, M. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Trans. on Automatic Control*, 2018.

[15] J. Fisac, M. Chen, C. J. Tomlin, and S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *HSCC*, 2015.

[16] T. Fraichard and H. Asama. Inevitable collision states-a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[17] S. Herbert, M. Chen, S. Han, S. Bansal, J. Fisac, and C. J. Tomlin. Fastrack: a modular framework for fast and guaranteed safe motion planning. In *CDC*, 2017.

[18] L. Janson, T. Hu, and M. Pavone. Safe motion planning in unknown environments: Optimality benchmarks and tractable policies. *arXiv preprint*, 2018.

[19] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint*, 2017.

[20] D.F. Keil, J. Fisac, and C. J. Tomlin. Safe and complete real-time planning and exploration in unknown environments. *arXiv preprint*, 2018.

[21] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Trans. on Robotics*, 21(3):354–363, 2005.

[22] S. Kousik, S. Vaskov, F. Bu, M. J. Roberson, and R. Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *arXiv preprint*, 2018.

[23] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Vardi. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Trans. on Robotics*, 32:583–599, 2016.

[24] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[25] A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[26] K. Margellos and J. Lygeros. Hamilton-Jacobi Formulation for ReachAvoid Differential Games. *IEEE Trans. on Automatic Control*, 56(8):1849–1861, 2011.

[27] I. Mitchell. A toolbox of level set methods. *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004.

[28] I. Mitchell, A. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. on automatic control*, 50(7):947–957, 2005.

[29] T. Moldovan and P. Abbeel. Safe exploration in markov decision processes. *arXiv preprint*, 2012.

[30] A. Richards and J. How. Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. *ACC '03*.

[31] C. Richter, W. Vega-Brown, and N. Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Robotics Research*, pages 325–341. 2018.

[32] U. Rosolia and F. Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Trans. on Automatic Control*, 63(7):1883–1896, 2018.

[33] S. Sarid, Bingxin X., and H. Kress-gazit. Guaranteeing high-level behaviors while exploring partially known maps. In *RSS*, 2012.

[34] T. Schouwenaars, J. How, and E. Feron. Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In *AIAA Guidance, Navigation, and Control Conference*, 2004.

[35] S. Singh, A. Majumdar, J. Slotine, and M. Pavone. Robust online motion planning via contraction theory and convex optimization. In *ICRA '17*.

[36] R. Walambe, N. Agarwal, S. Kale, and V. Joshi. Optimal trajectory generation for car-type mobile robot using spline interpolation. *IFAC-PapersOnLine*, 49(1):601–606, 2016.

[37] L. Yoder and S. Scherer. Autonomous exploration for infrastructure modeling with a micro aerial vehicle. In *Field and service robotics*, 2016.