2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

# Generating Formal Safety Assurances for High-Dimensional Reachability

Albert Lin[1] and Somil Bansal[2]

*Abstract*— **Providing formal safety and performance guarantees for autonomous systems is becoming increasingly important. Hamilton-Jacobi (HJ) reachability analysis is a popular formal verification tool for providing these guarantees, since it can handle general nonlinear system dynamics, bounded adversarial system disturbances, and state and input constraints. However, it involves solving a PDE, whose computational and memory complexity scales exponentially with respect to the state dimensionality, making its direct use on large-scale systems intractable. A recently proposed method called DeepReach overcomes this challenge by leveraging a sinusoidal neural PDE solver for high-dimensional reachability problems, whose computational requirements scale with the complexity of the underlying reachable tube rather than the state space dimension. Unfortunately, neural networks can make errors and thus the computed solution may not be safe, which falls short of achieving our overarching goal to provide formal safety assurances. In this work, we propose a method to compute an error bound for the DeepReach solution. This error bound can then be used for reachable tube correction, resulting in a safe approximation of the true reachable tube. We also propose a scenario-based optimization approach to compute a probabilistic bound on this error correction for general nonlinear dynamical systems. We demonstrate the efficacy of the proposed approach in obtaining probabilistically safe reachable tubes for high-dimensional rocket-landing and multi-vehicle collision-avoidance problems.**

## I. Introduction

It is becoming increasingly important that we can design provably safe controllers for autonomous systems. Hamilton-Jacobi (HJ) reachability analysis provides a powerful framework to design such controllers for general nonlinear dynamical systems [1], [2]. In reachability analysis, the safe states for the system are characterized through the *Backward Reachable Tube (BRT)* of the system. This is the set of states from which trajectories will eventually reach some given target set despite the best control effort. Thus, if the target set represents the set of undesirable states, the BRT represents unsafe states for the system and should be avoided. Along with the BRT, reachability analysis also provides a safety controller to keep the system outside the BRT.

Traditionally, the BRT computation in HJ reachability is formulated as an optimal control problem. The BRT can then be obtained as a sub-zero level solution of the corresponding value function. Obtaining the value function requires solving a partial differential equation (PDE) over a state-space grid, resulting in an exponentially scaling computation complexity with the number of states [3]. To overcome this challenge, a variety of solutions have been proposed that trade off between the class of dynamics they can handle, the approximation quality of the BRT, and the required computation. These include specialized methods for linear and affine dynamics [4]–[12], polynominal dynamics [13]–[16], monotonic dynamics [17], and convex dynamics [18] (see [3], [19] for a survey). Owing to the success of deep learning, there has also been a surge of interest in approximating high-dimensional BRTs [20]–[24] and optimal controllers [25] through deep neural networks (DNN). Building upon this line of work, the authors in [19] have proposed DeepReach – a toolbox that leverages recent advances in neural implicit representations and neural PDE solvers to compute a value function and a safety controller for high-dimensional systems. Compared to the aforementioned methods, DeepReach can handle general nonlinear dynamics, the presence of exogenous disturbances, as well as state and input constraints during the BRT computation. However, even if we can now obtain an approximate BRT for high-dimensional systems, it is limited in usefulness by its approximate nature. In particular, the learned BRT can be overly optimistic; thus, no formal safety guarantees can be provided on the obtained BRT and controller.

In this work, our goal is to compute a BRT and a safety controller for high-dimensional systems with provable safety guarantees. Specifically, we build upon DeepReach and propose a verification method to provide safety assurances on the DeepReach solution. The key insight of our work is to rely on the consistency between the learned value function and the implicit safety controller induced by this value function to compute a *uniform correction bound* for the value function. Essentially, if a particular state is outside the BRT (i.e., "safe" as per the learned value function), then starting from this state, the corresponding safety controller should keep the system trajectory outside the target set at all times (i.e., the state should also be "safe" under the prescribed controller). Otherwise, we cannot ensure safety of this state, and the value function should be corrected such that this state is inside the BRT. Once all such states have been added to the learned BRT, we have obtained a provably safe approximation of the true BRT.

We show that the computation of the correction bound can be posed as an optimization problem. However, in general, it is challenging to tractably compute this bound, since the learned value function can be highly nonlinear and hard to optimize. We propose a scenario-based optimization method to compute this correction. Scenario optimization is a sampling-based method to solve semi-infinite optimization problems, and has been widely used for system and control design [26]–[29]. The proposed method is not restricted

[1]Author is with CS at Princeton: {aklin}@princeton.edu. [2]Author is with ECE at University of Southern California: {somilban}@usc.edu. Project Website: https://sia-lab-git.github.io/DeepReach_Verification/

to a specific class of system dynamics or value functions. Given that the value function correction is obtained via randomization and, hence, is a random quantity, we provide probabilistic guarantees on the safety of the recovered BRT. However, this confidence is a design parameter and can be chosen as close to 1 as desired (within a simulation budget).

To summarize, the key contributions of this paper are:

- an error correction mechanism for DeepReach that results in a provably safe BRT and safety controller for general dynamical systems;
- a practical method to compute a probabilistic bound on this error correction that is not restricted to a specific class of systems, resulting in a tractable computation of probabilistically safe reachable tubes; and
- a demonstration of the proposed approach for various dynamical systems, inspired by rocket landing and multi-vehicle collision avoidance problems.

## II. Problem Setup

Consider a dynamical system with state $x \in X \subseteq \mathbb{R}^n$, control $u \in \mathcal{U}$, and dynamics $\dot{x} = f(x, u)$ governing how $x$ evolves over time until a final time $T$. Let $\xi^u_{x,t}(\tau)$ denote the state achieved at time $\tau \in [t, T]$ by starting at initial state $x$ and time $t$ and applying control $u(\cdot)$ over $[t, \tau]$. Let $\mathcal{L}$ represent a target set of states that the agent wants to either reach (e.g. goal states) or entirely avoid (e.g. obstacles).

***Running example: Dubins3D Avoid.*** As a running example, consider a simple low-dimensional system in the literature known as Dubins3D. It involves a car with position $(p_x, p_y)$, heading $\theta$, velocity $v$, and steering control $u_1 \in [u_{\min}, u_{\max}]$. The car's state $x$ evolves according to

$$\dot{p_x} = v\cos\theta, \quad \dot{p_y} = v\sin\theta, \quad \dot{\theta} = u_1$$

It wants to avoid a circle with radius $R$ centered at the origin. Thus, we define the target set $\mathcal{L}$ of this system to be:

$$\mathcal{L} = \{x : \sqrt{p_x^2 + p_y^2} \leq R\}$$

We use Dubins3D as a benchmark because it is tractable for traditional reachability methods to compute a solution for, and we can thus compare our results with a ground truth. In Section V, we present high-dimensional problems which traditional methods struggle with.

In this setting, we are interested in computing the system's initial-time Backward Reachable Tube, which we denote as BRT. We define BRT as the set of all initial states in $X$ from which the agent will eventually reach $\mathcal{L}$ within the time horizon $[0, T]$, despite best control efforts:

$$\text{BRT} = \{x : x \in X, \forall u(\cdot), \exists \tau \in [0, T], \xi^u_{x,0}(\tau) \in \mathcal{L}\} \quad (1)$$

When $\mathcal{L}$ represents unsafe states for the system, staying outside of BRT is desirable. When $\mathcal{L}$ instead represents the states that the agent wants to reach (e.g., a goal set), BRT is defined as the set of all initial states in $X$ from which the agent, acting optimally, can eventually reach $\mathcal{L}$ within $[0, T]$. Thus, staying within BRT is desirable.

Our goal in this work is to compute a provable approximation of the safe set. Specifically, we want to compute an approximation $\hat{\mathcal{S}}$ such that $\hat{\mathcal{S}} \subseteq \text{BRT}^C$ ($\hat{\mathcal{S}} \subseteq \text{BRT}$ when $\mathcal{L}$ represents goal states). We are especially interested in settings where the system is high-dimensional, with which current state-of-the-art reachability methods struggle.

## III. Background: Hamilton-Jacobi (HJ) Reachability and DeepReach

In this work, we build upon Hamilton-Jacobi reachability analysis to compute an approximation of the safe set. Here, we provide a quick overview of Hamilton-Jacobi reachability analysis and a specific toolbox to compute high-dimensional reachable sets, DeepReach.

### A. Hamilton-Jacobi (HJ) Reachability.

In HJ reachabilty, the computation of BRT is formulated as an optimal control problem. We will explain it in the context of $\mathcal{L}$ being a set of undesirable states. In the end, we will comment on the case where $\mathcal{L}$ represents a set of desirable states.

To compute BRT, we first define a target function $l(x)$ such that the sub-zero level of $l(x)$ yields $\mathcal{L}$:

$$\mathcal{L} = \{x : l(x) \leq 0\} \quad (2)$$

$l(x)$ is commonly a signed distance function to $\mathcal{L}$. For example, we can choose $l(x) = \sqrt{p_x^2 + p_y^2} - R$ for our Dubins3D Avoid running example.

Next, we define the cost function of a state corresponding to some policy $u(\cdot)$ to be the minimum of $l(x)$ over its trajectory:

$$J_{u(\cdot)}(x, t) = \min_{\tau \in [t,T]} l(\xi^u_{x,t}(\tau)). \quad (3)$$

Since the system wants to avoid $\mathcal{L}$, our goal is to maximize $J_{u(\cdot)}(x, t)$. Thus, the value function corresponding to this optimal control problem is:

$$V(x, t) = \sup_{u(\cdot)} J_{u(\cdot)}(x, t). \quad (4)$$

By defining our optimal control problem in this way, we can recover BRT using the value function. In particular, the value function being sub-zero implies that the target function is sub-zero somewhere along the optimal trajectory, or in other words, that the system has reached $\mathcal{L}$. Thus, BRT is given as the sub-zero level set of the value function at the initial time:

$$\text{BRT} = \{x : x \in X, V(x, 0) \leq 0\} \quad (5)$$

The value function in Equation (4) can be computed using dynamic programming, resulting in the following final value Hamilton-Jacobi-Bellman Variational Inequality (HJB-VI):

$$\min\Big\{D_t V(x, t) + H(x, t), l(x) - V(x, t)\Big\} = 0, \quad (6)$$

with the terminal value function $V(x, T) = l(x)$. $D_t$ and $\nabla$ represent the time and spatial gradients of the value function. $H$ is the Hamiltonian that encodes the role of dynamics and the optimal control.

$$H(x, t) = \max_u \langle \nabla V(x, t), f(x, u)\rangle. \quad (7)$$

The value function in Equation (4) induces the optimal safety controller:

$$u^*(x, t) = \arg\max_u \langle \nabla V(x, t), f(x, u)\rangle. \quad (8)$$

Intuitively, the safety controller aligns the system dynamics in the direction of the value function's gradient, thus steering the system towards higher-value states.

We have just explained the case where $\mathcal{L}$ represents a set of undesirable states. When the system instead wants to reach $\mathcal{L}$, an infimum is used instead of a supremum in Equation (4). The control wants to reach $\mathcal{L}$, hence there is a minimum instead of a maximum in Equation (7) and Equation (8).

Traditionally, the value function is computed by solving the HJB-VI over a discretized grid in the state space. Unfortunately, doing so involves computation whose memory and time complexity scales exponentially with respect to the system dimensionality, making these methods practically intractable for high-dimensional systems, such as those beyond 5D. Fortunately, a recent deep learning approach, DeepReach, has been proposed to enable Hamilton-Jacobi reachability for high-dimensional systems.

### B. DeepReach Approximate Solutions.

Instead of solving the HJB-VI over a grid, DeepReach represents the value function as a sinusoidal deep neural network (DNN) to learn a parameterized approximation of the value function [19]. Thus, memory and complexity requirements for training scale with the value function complexity rather than the grid resolution. To train the DNN, DeepReach uses self-supervision on the HJB-VI itself. Ultimately, it takes as input a state $x$ and time $t$, and it outputs a learned value $\tilde{V}(x,t)$. $\tilde{V}(x,t)$ also induces a corresponding policy $\tilde{\pi}(x,t)$. We refer interested readers to [19] for further details.

The learned $\tilde{V}(x,t)$ from DeepReach can be used to obtain a BRT as in Equation (5), but it will only be as accurate as $\tilde{V}(x,t)$. Unfortunately, like any learning method, $\tilde{V}(x,t)$ can deviate substantially from the true $V(x,t)$.

Our goal is to recover the biggest safe set $\hat{\mathcal{S}}$ using $\tilde{V}(x,t)$ that we can (probabilistically) guarantee to be fully contained within the true safe set. Although we work with DeepReach solutions in particular for this problem setup, our proposed approach in Section IV can verify any general $\tilde{V}(x,t)$ and $\tilde{\pi}(x,t)$, regardless of whether DeepReach, a level-set method, or some other tool is used to obtain them.

## IV. Approach

Our approach is to apply a minimal error correction to $\tilde{V}(x,t)$ such that the corrected value function can be used to extract a safe set. We will explain our approach in the context of $\mathcal{L}$ being a set of undesirable states. In the end of each subsection, we will comment on the case where $\mathcal{L}$ represents a set of desirable states. In Section IV-A, we propose an error metric for correcting the value function. In Section IV-B, we propose a practical method to compute a probabilistic bound on this error metric through scenario optimization.

### A. Error Metric

We propose a new error metric $\delta_{\tilde{V},\tilde{\pi}}$ for the value function correction. It is defined as the maximum learned value of an empirically unsafe initial state under the induced policy $\tilde{\pi}$:

$$\delta_{\tilde{V},\tilde{\pi}} \coloneqq \max_{x \in X} \{\tilde{V}(x,0) : J_{\tilde{\pi}}(x,0) \leq 0\}, \tag{9}$$

where $J_{\tilde{\pi}}(x,0)$ is the cost function associated with the trajectory obtained by using the policy $\tilde{\pi}(x,t)$ from an initial state $x$ and initial time $t=0$ until $T$ (see Equation (3)). Here on, we use $\delta$ as a shorthand for $\delta_{\tilde{V},\tilde{\pi}}$ for brevity purposes.

Intuitively, $\delta$ finds the tightest level of the learned value function that separates the states that are safe under the induced policy $\tilde{\pi}(x,t)$ from the ones that are not. Thus, any initial state within the super-$\delta$ level set of $\tilde{V}(x,0)$ is guaranteed to be safe under the (possibly sub-optimal) policy $\tilde{\pi}(x,t)$. Lemma 1 formalizes this claim.

*Lemma 1:* If we compute $\tilde{\mathcal{S}}$ as:

$$\tilde{\mathcal{S}} = \{x \in X : \tilde{V}(x,0) > \delta\} \tag{10}$$

then $\tilde{\mathcal{S}} \subseteq \text{BRT}^C$.

The proof for Lemma 1 is in Appendix-A of the extended version of this article [30]. Intuitively, Lemma 1 states that if we could exactly compute $\delta$, then we can recover a $\tilde{\mathcal{S}}$ that is guaranteed to be a subset of the true safe set. Furthermore, it is clear that, by definition, $\delta$ is the smallest (uniform) value adjustment required on $\tilde{V}(x,0)$ to guarantee its safety. Thus, $\tilde{\mathcal{S}}$ is the largest safe set we can recover from the learned value function by such a uniform error correction procedure.

Unfortunately, computing $\delta$ is a challenging optimization problem, since both the value function and the cost function in the metric definition (9) are typically non-convex functions of $x$. In the next section, we propose a scenario-based optimization approach to compute a high-confidence bound on $\delta$ and thereby generate a high-confidence safe set.

*Remark 1:* We assume there exists an unsafe state for the metric definition (9). Otherwise, define $\delta_{\tilde{V},\tilde{\pi}}$ trivially as $-\infty$.

So far, we have discussed an error correction metric for the case where $\mathcal{L}$ represents a set of undesirable states. When $\mathcal{L}$ represents a set of desirable states, we use a minimum instead of a maximum and flip the cost inequality in the metric definition (9). To recover the safe set, we extract the sub-$\delta$ level set of the value function in Equation (10).

### B. Scenario Optimization Verification Method

We will compute a high-confidence bound on $\delta$ by utilizing a random sampling procedure referred to as *scenario optimization* in the systems and control design literature [29]. The Scenario Optimization Verification Method to compute an approximation $\hat{\delta}$ is summarized in Algorithm 1.

At a high-level, Algorithm 1 computes a converging sequence of $\delta_i$ that approximates $\delta$ via random sampling until we no longer find any safety violations or reach a maximum number of iterations, $M$. Specifically, at each iteration $i$, we randomly sample $N$ initial states within the super-$\delta_i$ level set of $\tilde{V}$ (Line 3) using rejection sampling and compute the costs of associated trajectories $J_{\tilde{\pi}}(x,0)$ under $\tilde{\pi}(x,t)$ as the controller (Equation (3)). We next compute the maximum learned value among the states that violate safety and use that as the new estimate of $\delta$ (Line 5)), essentially discarding a level region of $\tilde{V}(x,0)$ that is empirically unsafe under $\tilde{\pi}$. Thus, with each iteration, we obtain a tighter approximation of $\delta$ and terminate when no more safety violations are found or a maximum number of iterations is achieved.

**Algorithm 1:** Scenario Optimization Verification

**Require:** $X$, $N$, $M$, $\tilde{V}(x,0)$, $J_{\tilde{\pi}}(x,0)$

$\delta_0 \leftarrow -\infty$
**for** $i = 0, 1, \ldots, M-1$ **do**
  $\mathcal{D}_i \leftarrow$ Sample $N$ states IID from $\{x : x \in X, \tilde{V}(x,0) > \delta_i\}$
  **if** $\exists x \in \mathcal{D}_i : J_{\tilde{\pi}}(x,0) \leq 0$ **then**
    $\delta_i \leftarrow \max_{x \in \mathcal{D}_i} \{\tilde{V}(x,0) : J_{\tilde{\pi}}(x,0) \leq 0\}$
  **else**
    **break**
  **end if**
**end for**
**return** $\hat{\delta} := \delta_i$

While Algorithm 1 is simple to understand and execute, scenario optimization provides a formal guarantee for bounding $\delta$ with $\hat{\delta}$ when the algorithm terminates prior to reaching the maximum number of iterations. Thus, we can use the approximated $\hat{\delta}$ instead of $\delta$ to compute an approximate safe set $\hat{\mathcal{S}}$ similar to the one defined in Lemma 1:

$$\hat{\mathcal{S}} = \{x : x \in X, \tilde{V}(x,0) > \hat{\delta}\} \tag{11}$$

Crucially, we can make a formal probabilistic safety guarantee for $\hat{\mathcal{S}}$. This is summarized in Theorem 1 and proven in Appendix-C of the extended version of this article [30].

*Theorem 1 (Scenario Optimization Verification Theorem):* Select a violation parameter $\epsilon \in (0,1)$ and a confidence parameter $\beta \in (0,1)$. Pick $N$ such that

$$N \geq \frac{2}{\epsilon}\left(\ln\frac{1}{\beta} + 1\right) \tag{12}$$

Suppose Algorithm 1 converges, then with probability at least $1-\beta$, the recovered safe set $\hat{\mathcal{S}}$ in Equation (11) satisfies:

$$\mathbb{P}_{x \in \hat{\mathcal{S}}}(V(x,0) \leq 0) \leq \epsilon \qquad \square \tag{13}$$

In practice, we find that the algorithm converges within 3-4 iterations. Intuitively, the higher the quality of the value function approximation, the faster the convergence of $\delta_i$.

Disregarding the confidence parameter $\beta$ for a moment, Theorem 1 states that the volume of the unsafe states within $\hat{\mathcal{S}}$ is smaller than or equal to the prescribed $\epsilon$ value, as long as we sample enough states to satisfy Inequality (12) during the computation of $\hat{\delta}$. As $\epsilon$ approaches 0, the number of safety violations in the recovered safe set also approaches 0. In turn, the simulation effort grows unbounded since $N$ is inversely proportional to $\epsilon$.

To interpret the confidence parameter $\beta$, note that $\hat{\mathcal{S}}$ is a random variable that depends on a randomly sampled set of initial states. It may be the case that we just happen to draw a poorly representative sample, in which case the $\epsilon$ bound does not hold. $\beta$ controls the probability of this adverse event happening, which regards the correctness of the final guarantee in Equation (13). Fortunately, $N$ only grows logarithmically with $\frac{1}{\beta}$, so $\beta$ can be chosen to be an extremely small value such as $10^{-12}$. $1-\beta$ should then be so close to 1 that it does not have any practical importance.

When $\mathcal{L}$ represents a set of desirable states, we initialize $\delta_0$ to be $\infty$ instead of $-\infty$, flip the inequalities, and take a minimum instead of a maximum in Algorithm 1. We flip the value inequalities in Equation (11) and Equation (13).

*Remark 2:* Note that when Algorithm 1 does not converge, scenario optimization can still be used to bound $\delta$. However, the safety guarantees are more involved. We defer a detailed investigation of these guarantees to future work.

***Running example: Dubins3D Avoid.*** We now demonstrate the Scenario Optimization Verification method on the Dubins3D Avoid running example introduced in Section II. We choose system parameters $v = 0.6m/s, u_{\min} = -1.1rad/s, u_{\max} = 1.1rad/s, R = 0.25m$, and use DeepReach to learn the system's value function. The used neural network parameters and architecture are the same as in the DeepReach paper [19], i.e., a DNN with 3 hidden layers and 512 neurons in each layer.

For this first example, we purposefully sabotage the training of DeepReach by permitting it to train for only 8,000 epochs, far less than 100K epochs used in the DeepReach paper. We do this to illustrate our method's utility even with low-quality learned solutions. We apply our method to a well-trained DeepReach solution in Section V-A.

The overall training took less than 30 minutes on an NVIDIA GeForce RTX 3090Ti. We execute our verification approach choosing $\beta = 10^{-16}, \epsilon = 10^{-3}$, resulting in an $N = 75683$ as per Theorem 1. Thus, we will be $1 - 10^{-16}$ confident that at least $1 - 10^{-3}$ of our recovered set will be provably safe. Algorithm 1 converges in 4 iterations in this case, resulting in a $\hat{\delta} = 0.3728$. Slices of the trained BRT (yellow), the recovered BRT (blue), and the ground truth BRT boundary (black) are shown in Figure 1. The ground truth is computed by a state-of-the-art PDE solver, *Level Set Toolbox (LST)* [31], which computes the value function over a discrete grid of size $101 \times 101 \times 101$.

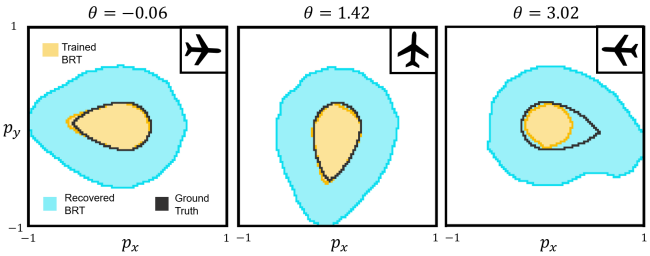


Fig. 1: Dubins3D Avoid: Slices of the trained, recovered, and ground truth border BRTs for three values of $\theta$ from a sabotaged DeepReach solution. The recovered BRT completely encompasses the ground truth BRT and provides a probabilistically safe approximation of the safe set.

As evident from the figure, the recovered BRT slices completely encompass the true unsafe states (the blue region encompasses the states within the black boundary). Hence, the complement of the recovered BRT is safe, as we expect based on Theorem 1. In the rightmost slice, observe that the trained BRT is smaller than the ground truth BRT and is thus unsafe as it is. Our verification approach correctly expands the trained BRT past the ground truth boundary, ensuring safety. Validation by sampling 1M states within the recovered safe set reveals a violation rate of $3 \times 10^{-6} << \epsilon$.

Note that the trained solution's safety violations are concentrated near $\theta = \pi$. It is likely these errors which are responsible for the large expansion of the recovered BRT even for slices that are almost accurate (left and middle slices in Figure 1). This is because our method expands the BRT uniformly. However, too much expansion is undesirable, since we want to recover as much of the safe set as possible.

One way to overcome this challenge is to apply refined error correction to $\tilde{V}(x,t)$ by independently verifying separate regions of the states-space (i.e., separately consider each region as our $X$). This allows us to recover sets with the same safety guarantee but expanded differently depending on their own region's error. However, this method's efficacy depends heavily on the choice of sub-regions, which are hard to prespecify, especially for high-dimensional systems. Instead, we can resort to a data-driven approach.

Specifically, we trained a simple multilayer perceptron (MLP) on 1M randomly sampled initial states and their empirical safety violation costs (the training took an hour on a standard GPU). Even though the predictor may be inaccurate (after all, predicting the safe set is the original challenging problem), we hypothesize that it will have learned something useful enough about the general distribution of errors throughout the state space. We divided the output of the MLP in 10 bins, each corresponding to a different range of empirical cost. Thus, we expect the last bin to correspond to the states with maximal safety violations (in this case, that will correspond to states with $\theta = \pi$). We next ran Algorithm 1 independently for each bin to compute an error correction. The corresponding results are shown in Figure 2. As evident, we get a much tighter approximation of the safe set near the slices that have small errors, allowing us to recover a much bigger safe set overall. In this case, the recovered safe set volume increased by 35%.

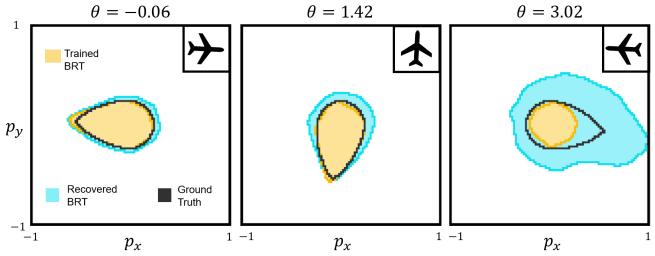


Fig. 2: Dubins3D Avoid: Slices of the trained, recovered, and ground truth border BRTs for three values of $\theta$ from a sabotaged DeepReach solution when recovery is refined by MLP binning. The recovered BRT completely encompasses the ground truth BRT and provides a probabilistically safe approximation of the safe set that is larger than when binning is not used.

As discussed earlier, the performance improvement of this approach depends heavily on the selection of subregions. We defer this to future work, and for the rest of this paper, we only focus on uniform value function correction.

## V. Case Studies

We will evaluate our approach on various reachability problems. First, we show results for the avoid version and the reach version of the low-dimensional Dubins3D system for which we have a ground truth BRT available for comparison. Next, we recover safe sets for a 9D multivehicle collision avoidance problem and for a 6D rocket landing problem, with which traditional methods struggle. Like our running example, we use $\beta = 10^{-16}, \epsilon = 10^{-3}, N = 75683$.

### A. Dubins3D Avoid

In Figure 3, we show the results of applying our method to a well-trained DeepReach solution for the same Dubins3D Avoid system as in the running example in Section IV. The solution is trained to consider the periodicity of $\theta$ and for 100K, instead of 8K, epochs. The proposed algorithm results in a very small $\hat{\delta} = -0.0016$. Interestingly, the fact that $\hat{\delta} < 0$ indicates that the learned value function is conservative. Our method shrinks the trained BRT to recover a larger safe set. Validation by sampling 1M states within the recovered safe set reveals a violation rate of $3.2 \times 10^{-5} << \epsilon$.

The fact that a much smaller error correction is found for the trained solution shown in Figure 3 than the one in Figure 1 indicates it is of a much higher quality. This illustrates how the error correction metric $\delta$ can also be used to evaluate the quality of different approximate value function solutions. For example, we can evaluate the relative performance of different DNN hyperparameters, which is especially helpful when the ground truth BRT is not available for comparison.

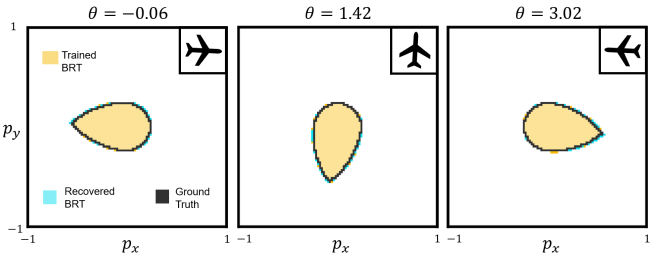


Fig. 3: Dubins3D Avoid: Slices of the trained, recovered, and ground truth border BRTs for three values of $\theta$ from a well-trained DeepReach solution. The trained BRT completely encompasses the recovered BRT, so we plot the recovered BRT border on top for visualization purposes.

### B. Dubins3D Reach

In Figure 4, we show the results of applying our method to a DeepReach solution trained with the same scheme as for the Dubins3D Avoid solution in Section V-A, but for the reach version. That is, $\mathcal{L}$ is now a set of desirable states, so a safe approximate BRT should be fully contained by the true BRT. The trained BRT is just barely crossing the ground truth boundary, and the proposed method cuts it down to just behind the boundary, ensuring system safety.

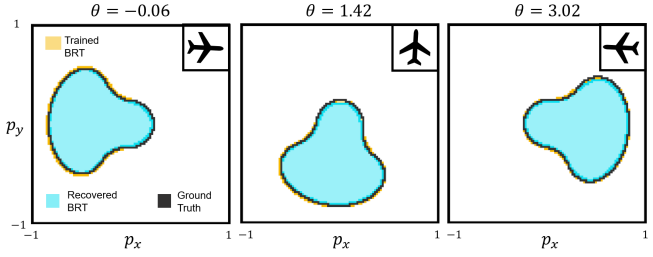


Fig. 4: Dubins3D Reach: Slices of the trained, recovered, and ground truth border BRTs for three values of $\theta$ from a well-trained DeepReach solution. The recovered BRT is correctly a subset of the ground truth BRT.

### C. Multivehicle Collision Avoidance

We now consider a 9D collision avoidance system involving 3 independent Dubins3D cars. The $i$th car has position $(p_{xi}, p_{yi})$, heading $\theta_i$, velocity $v$, and steering control $u_i \in [u_{\min}, u_{\max}]$. The dynamics of vehicle $i$ are given as:

$$\dot{p}_{xi} = v\cos\theta_i, \quad \dot{p}_{yi} = v\sin\theta_i, \quad \dot{\theta}_i = u_i$$

The (undesirable) target set is given by the states where any of the vehicle pairs is in collision:

$$\mathcal{L} = \{x : \min\{d(Q_1, Q_2), d(Q_1, Q_3), d(Q_2, Q_3)\} \leq R\}$$

where $d(Q_i, Q_j)$ is the distance between cars $i, j$. We choose $v = 0.6, u_{\min} = -1.1, u_{\max} = 1.1, R = 0.25$ for our case study. The results of applying our method to a DeepReach solution are shown in Figure 5.

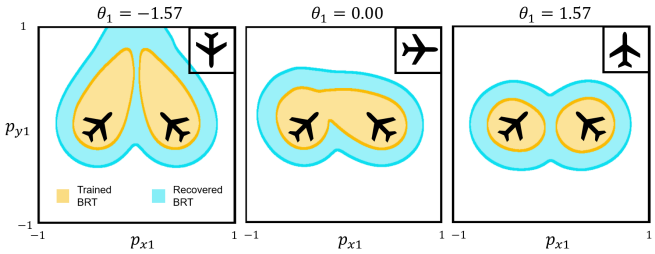


Fig. 5: Multivehicle Collision Avoidance: Slices of the trained and recovered BRTs for three values of $\theta_1$ from a DeepReach solution.

This high-dimensional example is typically difficult to compute with traditional methods, yet here we demonstrate successful recovery of a safe set with formal safety guarantees. While the recovery preserves much of the safe set that is learned, a significant amount is pruned off. The percent reduction in safe set size (calculated from 1M samples) is much larger in this example (26%) than for the Dubins3D solutions in Section V-A (0%) and Section V-B (10%), suggesting that the learned value function is more inaccurate for higher dimensional systems.

To validate safety of the recovered BRT, in Figure 6, we plot the minimum pairwise distance between vehicles along trajectories spawning from states sampled within the trained and recovered safe sets. Note that the trained safe set contains states which result in a collision, as shown by the mass to the left of the dotted line representing the collision distance. After verification, the recovered safe set shows no such mass. Validation by sampling 1M samples in the recovered safe set reveals a violation rate of $5 \times 10^{-6} << \epsilon$.

### D. Rocket Landing

Consider a 6D rocket landing system with position $(p_x, p_y)$, heading $\theta$, velocity $(v_x, v_y)$, angular velocity $\omega$, and torque controls $\tau_1, \tau_2 \in [-250, 250]$.
The dynamics are:

$$\dot{p_x} = v_x, \ \dot{p_y} = v_y, \ \dot{\theta} = \omega, \ \dot{\omega} = 0.3\tau_1,$$
$$\dot{v_x} = \tau_1\cos\theta - \tau_2\sin\theta, \ \dot{v_y} = \tau_1\sin\theta + \tau_2\cos\theta - g,$$

where $g = 9.81$ is acceleration due to gravity. The target set is the set of states where the rocket reaches a rectangular

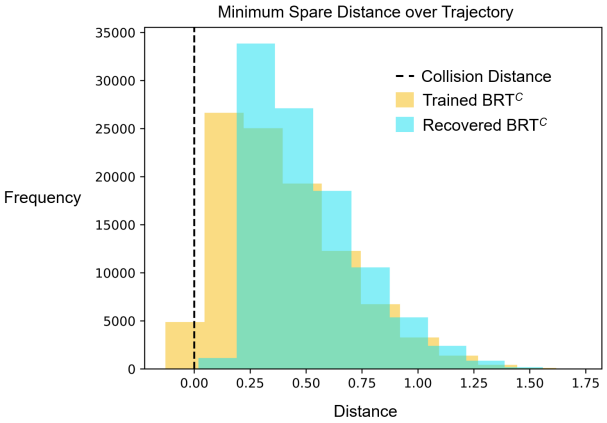


Fig. 6: Multivehicle Collision Avoidance: Histogram of the minimum pairwise spare distance between vehicles along trajectories which spawn from states sampled within the trained and recovered safe sets. The collision distance is indicated by the dotted line.

landing zone of side length 20 centered at the origin:

$$\mathcal{L} = \{x : |p_x| < 20.0, p_y < 20.0\}$$

In Figure 7, we show the results of applying our method to a DeepReach solution. We also plot the trajectories emanating from 20 randomly sampled initial states within the recovered safe set, demonstrating that they do indeed reach the target set (the green region) by following the induced policy. Validation by sampling 1M states within the recovered safe set reveals a violation rate of $5 \times 10^{-6} << \epsilon$.

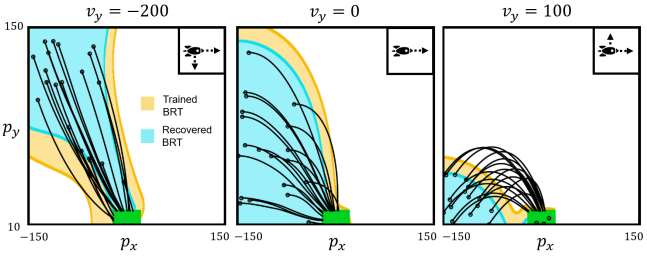


Fig. 7: Rocket Landing: Slices of the trained and recovered BRTs for three values of $v_y$ from a DeepReach solution. The system trajectories safely reach the target set (green) from the initial states within the recovered BRT.

## VI. Discussion and Future Work

In this work, we present an approach to compute an error bound for DeepReach solutions to recover a provably safe approximation of the true reachable tube. We also propose a practical method to compute a probabilistic bound on this error correction that is not restricted to a specific class of systems. This allows us to utilize the power of learning-based reachability methods to provide probabilistic safety assurances for high-dimensional dynamical systems. We apply our method to obtain probabilistically safe reachable tubes for high-dimensional rocket-landing and multi-vehicle collision-avoidance problems which traditional methods struggle with.

In the future, we will explore a more refined approach to error correction (as opposed to a uniform correction), as discussed briefly in the end of Section IV. Other directions include considering worst-case disturbances in the system dynamics and using the verification approaches proposed here for a targeted refinement of DeepReach solutions.

## REFERENCES


[1] J. Lygeros, "On reachability and minimum cost optimal control," *Automatica*, vol. 40, no. 6, pp. 917–927, 2004.

[2] I. Mitchell, A. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control (TAC)*, vol. 50, no. 7, pp. 947–957, 2005.

[3] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi Reachability: A brief overview and recent advances," in *IEEE Conference on Decision and Control (CDC)*, 2017.

[4] M. R. Greenstreet and I. Mitchell, "Integrating projections," in *Hybrid Systems: Computation and Control*, T. A. Henzinger and S. Sastry, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 159–174.

[5] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *International Conference Computer Aided Verification*, 2011.

[6] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis: internal approximation," *Systems & Control Letters*, 2000.

[7] A. Kurzhanski and P. Varaiya, "On ellipsoidal techniques for reachability analysis. part ii: Internal approximations box-valued constraints," *Optimization Methods and Software*, vol. 17, pp. 207–237, 01 2002.

[8] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. Oishi, and G. A. Dumont, "Lagrangian methods for approximating the viability kernel in high-dimensional systems," *Automatica*, 2013.

[9] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*, 2005, pp. 291–305.

[10] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear analysis: hybrid systems*, vol. 4, no. 2, pp. 233–249, 2010.

[11] S. Bak, H.-D. Tran, and T. T. Johnson, "Numerical verification of affine systems with up to a billion dimensions," in *International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 23–32.

[12] P. Nilsson and N. Ozay, "Synthesis of separable controlled invariant sets for modular local control design," in *American Control Conference*, 2016, pp. 5656–5663.

[13] A. Majumdar, R. Vasudevan, M. M. Tobenkin, and R. Tedrake, "Convex optimization of nonlinear feedback controllers via occupation measures," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1209–1230, 2014. [Online]. Available: https://doi.org/10.1177/0278364914528059

[14] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[15] T. Dreossi, T. Dang, and C. Piazza, "Parallelotope bundles for polynomial reachability," in *International Conference on Hybrid Systems: Computation and Control*, 2016.

[16] D. Henrion and M. Korda, "Convex computation of the region of attraction of polynomial control systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 297–312, 2014.

[17] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, 2015, pp. 58–67.

[18] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, "Algorithm for overcoming the curse of dimensionality for time-dependent non-convex hamilton–jacobi equations arising from optimal control and differential games problems," *Journal of Scientific Computing*, vol. 73, no. 2-3, pp. 617–643, 2017.

[19] S. Bansal and C. J. Tomlin, "DeepReach: A deep learning approach to high-dimensional reachability," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[20] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin, "A classification-based approach for approximate reachability," in *International Conference on Robotics and Automation*. IEEE, 2019, pp. 7697–7704.

[21] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning," *International Conference on Robotics and Automation*, 2019.

[22] B. Djeridane and J. Lygeros, "Neural approximation of pde solutions: An application to reachability computations," in *Conference on Decision and Control*, 2006, pp. 3034–3039.

[23] K. Niarchos and J. Lygeros, "A neural approximation to continuous time reachability computations," in *Conference on Decision and Control*, 2006, pp. 6313–6318.

[24] J. Darbon, G. P. Langlois, and T. Meng, "Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures," *Research in the Mathematical Sciences*, vol. 7, no. 3, pp. 1–50, 2020.

[25] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, "A neural network approach for high-dimensional optimal control applied to multiagent path finding," *IEEE Transactions on Control Systems Technology*, 2022.

[26] S. Ghosh, S. Bansal, A. Sangiovanni-Vincentelli, S. A. Seshia, and C. Tomlin, "A new simulation metric to determine safe environments and controllers for systems with unknown dynamics," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 185–196.

[27] M. C. Campi and S. Garatti, "A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality," *Journal of Optimization Theory and Applications*, 2011.

[28] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Transactions on Automatic Control*, 2006.

[29] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, 2009.

[30] A. Lin and S. Bansal, "Generating formal safety assurances for high-dimensional reachability," *arXiv preprint arXiv:2209.12336*, 2022.

[31] I. Mitchell, *A Toolbox of Level Set Methods*, 2009, http://people.cs.ubc.ca/~mitchell/ToolboxLS/index.html.