CS 229: Final Project Report
Machine Learning Techniques for Optimal Sampling-Based Motion Planning

Ross E. Allen, Ashley A. Clark, and Joseph A. Starek

***Abstract*— A key question arising in optimal control and motion planning of dynamically-constrained systems is how to address the difficulty of (often non-convex) collision avoidance constraints in a real-time fashion. One promising approach is sampling-based motion planning algorithms, which work by drawing samples from the continuous set of configurations available to the robot and attempting to link them together in a graph to form a solution to the global problem. Graph construction requires that intermediate sample points be joined together in an optimal fashion; this is achieved by solving a two-point boundary value problem (2PBVP), or *steering problem*, for each graph edge. In practice, accomplishing this task in real time is computationally intractable (on the order of thousands or millions of edges, each requiring up to roughly tens-of-seconds per solution for non-convex problems). Computational efficiency can be gained by considering connections to only those nodes within the local vicinity that are physically reachable by the robot, *i.e.*, nodes within its *reachability set*, before attempting to steer between them. Unfortunately, for the majority of systems, computation of these sets is as difficult as the original motion planning problem. Once reachable nodes are identified, however, further efficiency can be gained by using an approximate cost function to estimate the best candidate node to steer towards, and only then solving the full 2PBVP to that node. We propose the use of supervised machine learning as a heuristic strategy for addressing these two facets of the motion planning problem: approximating the reachable set of a dynamic system, and approximating the optimal cost-to-go function between any pair of states.**

I. Introduction

With the recent rise in interest in autonomous vehicles, including Google's Driverless Car and the recently-proposed Amazon Prime Air, the need has grown for rapid decision-making with hard constraints on safety and guaranteed collision avoidance. The underlying question is: how does a dynamic system select a collision-free trajectory from points A to B through an obstructed environment? Not only is feasible path-planning required to meet mission goals, but quite often it is desirable to find an *optimal* path in *real time* that optimizes some objective, such as time, energy, or fuel use. Sampling-based motion planning has emerged as a promising approach to solving this problem. Sampling-based algorithms first discretize the problem by drawing samples from the continuous set of configurations available to the robot and then attempt to link them together into a connectivity graph that is amenable to artificial intelligence search techniques. This simplifies calculations by allowing obstacle constraints to be ignored, initially, and then performing collision-checking a posteriori. They rely on the assumption that configurations near each other are more likely to be connected without passing through an obstacle; their correctness hinges on the concept that once sufficiently many intermediate connections have been explored, a global path from start to goal will be discovered if one exists. See Fig. 1 for an illustration. Most planners fall into one of two categories: roadmap-based planners that construct a roadmap interconnecting all nodes, or tree-based planners that construct a spanning tree between all nodes.

Stanford University, Department of Aeronautics and Astronautics
Contact: {rallen10,aaclark,jstarek}@stanford.edu

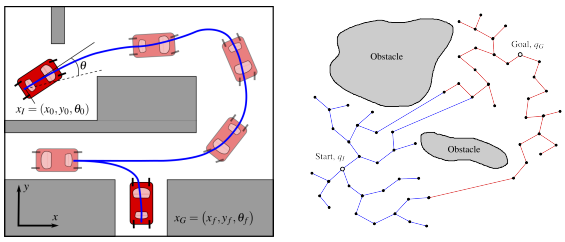

Fig. 1: Illustrations of the motion planning problem and sampling-based motion planning.

For optimal motion planning, graph construction requires that intermediate sample points be joined together in an optimal fashion, achieved by solving an optimal two-point boundary value problem (2PBVP), also called a steering problem, for each graph edge. The optimal 2PBVP, subject to obstacle constraints, can be mathematically stated as:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & J(\mathbf{x}, \mathbf{u}, t) \\ \text{subject to} \quad & \mathbf{x}(t) \in \mathcal{X}_{\text{free}} \quad \text{for all } t \in [t_0, t_f] \\ & \mathbf{u}(t) \in \mathcal{U} \quad \text{for all } t \in [t_0, t_f] \\ & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \\ & \mathbf{x}(t_0) = \mathbf{x}_{\text{init}}, \mathbf{x}(t_f) = \mathbf{x}_{\text{goal}} \end{aligned} \tag{1}$$

where J is the cost function to be minimized, $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the control vector, $\mathcal{X}_{\text{free}}$ is the obstacle-free state space, and $\mathcal{U}$ is the set of admissable controls.

In practice, for heavily cluttered environments, discovering a path could require on the order of thousands or millions of edge computations and collision-checks. Solving such a number of steering problems in real time is computationally intractable. In order to reduce the number of these calculations, we may consider only connecting to nodes within the local vicinity of the current state. For optimal motion planning, the "vicinity" of a state is the set of states reachable by the robot within a certain cost α, called a reachability (or its α-reachable) set. Connections need only be considered to nodes within this set, and the order in which connection attempts are made can further improve running time if the cost to each of them is known or otherwise approximated without solving each steering problem. To this end, we propose the use of supervised machine learning to both approximate these reachability sets and additionally estimate optimal connection costs between node pairs.

The idea of applying machine learning techniques to improve the running time of motion planning algorithms is not a new one. Burns [1] proposes an approximate, machine learning based approach for efficient roadmap construction that

attempts to reduce collision-checking time. The approach utilizes locally-weighted Gaussian regression to determine the likelihood that a new sample or candidate edge is in collision with the environment, based on the safety of its neighboring nodes. The technique is useful for roadmap-based approaches with many edges, but likely yields limited benefit for many tree-based approaches, which only check one edge at a time per iteration.

Our proposed technique, on the other hand, would be applicable to all flavors of optimal sampling-based motion planning for most realistic scenarios[1]. As reachability and cost computations are integral to essentially every motion planning algorithm, the approach may lead (within an acceptable error tolerance) to an effective, general way to improve convergence rates of motion planning queries. In this paper, we will show that with a training set of pre-computed 2PBVP costs, we can accurately use (1) a locally-weighted regression algorithm for predicting cost function values and (2) a support vector machine learning algorithm for nonlinear binary classification of the system's reachable sets. To the best of the authors' knowledge, this is the first application of machine learning to reachability classification. We apply these learning algorithms to two systems of interest: the Dubins car model (see Fig. 2) and a deep-space spacecraft model (see Fig. 6).

II. Approach

Here we outline the methodology behind our techniques.

A. Cost Prediction

For a given system, each motion planning query (represented by Eqn 1) differs only in the initial and final conditions of the 2PBVP. As a result, the optimal cost-to-go J^* can be thought of as a function of the endpoints $\mathbf{x}(t_0)$ and $\mathbf{x}(t_f)$ parameterized by system dynamics. Since motion planning algorithms seek to connect only those nodes in close proximity (in terms of cost) to one other, under mild assumptions[2] the solution value to a similar query must be closely related and hence can be interpolated from neighboring queries. This suggests the use of local regression to estimate the optimal cost-to-go for new motion plans – a simple, robust, and well-developed form of which is locally-weighted linear regression, which we employ here. In order to more effectively apply linear regression techniques to general cost functions (likely nonlinear in the initial and final state variables), we compiled a set of nonlinear combinations of state variables that we hypothesized might be relevant to estimating the optimal cost-to-go function, such as component norms, component products, energy values, etc. We then ran a feature selection algorithm to identify those that actually had a significant impact on the approximation accuracy; this allows not only a view of the trade-off in approximation error and feature vector size, but also an intuition for the control designer in terms of choosing a feature vector for his or her relevant application.

[1]Though not proven here, it is believed the techniques of this paper require Lipschitz continuity of the objective function and the system dynamics with respect to the control vector

[2]objective function and system dynamics regularity conditions, *e.g.*, Lipshitz continuity

Model Selection: Locally-weighted linear regression uses optimal cost values of nearby feature vectors to interpolate the value for a new query point. Due to the large differences in scale between feature vector components, the weights used for regression were normalized to bring each of the components into the same unit scaling and improve interpolation performance. We defined weights as:

$$w^{(i)} = \exp\left(\frac{\left|\left|\mathbf{v}^{(i)}\right|\right|^2}{2\tau^2}\right)$$

$$\text{with } \mathbf{v}_j^{(i)} = \frac{\phi\left(\mathbf{x}^{(i)}\right)_j - \phi(\mathbf{x})_j}{\text{range}\Big(\phi(\mathbf{x})_j\Big)}, j = 1, \ldots, n$$

where τ is the bandwidth parameter, $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is the i^{th} training point, $\phi : \mathbb{R}^d \to \mathbb{R}^n$ is the feature vector mapping, and $\text{range}\Big(\phi(\mathbf{x})_j\Big)$ is the maximum extent of the j^{th} feature.

With locally-weighted linear regression, there is a tradeoff between overfitting the data with a very small bandwidth parameter, and underfitting the data with a very large bandwidth parameter. Thus, we ran model selection over a range of bandwidth parameters to determine the most appropriate one for prediction. Each parameter's model was passed through k-fold cross-validation. The bandwidth parameter that yielded the lowest average percent error over all of the k training sets was selected as the best bandwidth parameter to use for cost prediction.

Training Method: For a fixed regression model (bandwidth parameter), training for optimal cost estimation was performed using k-fold cross-validation, with $k = 10$. Error was defined as the average percent error over all training examples between the predicted cost and the actual cost as returned by an optimal 2PBVP solver.

Feature Selection: A backwards feature selection search was run to obtain a trade-off between feature count and average cost prediction error, where the initial feature vector was chosen to be an over-approximation of the feature set relevant to the prediction problem. For a given feature set, once the best bandwidth parameter was determined via model selection, the model was subsequently re-fit to the entire training set to yield parameter vector $\boldsymbol{\theta}$. The feature vector component corresponding to the smallest magnitude component of $\boldsymbol{\theta}$ was selected as the least relevant feature, and removed from the feature set. This procedure was iterated until one feature remained, yielding a priority ranking for each feature in terms of its effect on cost estimation from least to most relevant.

B. Reachability Classification

In the geometric case (without dynamic constraints), the freedom of the robot to maneuver on straight lines in any direction makes reachability easy to assess for shortest-path objective functions in unobstructed configuration spaces: cost-reachable sets form n-spheres about the current node $\mathbf{x} \in \mathbb{R}^n$. For dynamically-constrained vehicles, on the other hand, boundaries demarcating cost-reachable sets are generally much more difficult to assess geometrically. Analytical descriptions often only exist for over- or underapproximations. Numerical approximations become necessary, which

are known to be as difficult to compute (i.e. exponential-time) as the original motion planning problem [2]. The motivation for a machine learning approach follows from the following observation: by continuity of the system dynamics, reachable sets are guaranteed to be simply connected, and therefore reachable states must be *separable* from non-reachable ones. This suggests we can approximate the reachability boundary using a non-linear classifier, separating training examples into reachable and non-reachable sets. The classifier can then be used to estimate whether new query point pairs $(\mathbf{x}_{\text{init}}, \mathbf{x}_{\text{goal}})$ are reachable within the cost threshold used to define the trained reachability boundary. For the applications in this paper, a Support Vector Machine (SVM) algorithm with a non-linear kernel function was employed.

C. Data Generation

In order to train the locally-weighted regression and SVM algorithms previously discussed, it was necessary to generate "true" examples with known optimal cost-to-go. This was accomplished by solving a large number of 2PBVP problems for our chosen systems and recording the inputs/attributes (initial and final states) and outputs (optimal cost). Any 2PBVPs with infinite cost, *i.e.*, those with no feasible connection between initial and final states, were neglected from training. To ensure a well-distributed sampling of the inputs/attributes, training data was generated using the Halton sequence, assuming a hypercubical feature space. Each 2PBVP was solved by first discretizing the continuous optimal control problem using the Chebyshev Pseudospectral Method, transforming the optimal control problem into a nonlinear programming problem (NLP). The existing software package "fmincon" in MATLAB was then employed as the NLP-solver. 1000 training examples were generated for each of our numerical demonstrations.

III. Application and Results

To evaluate the effectiveness of machine learning for motion planning problems, we selected two example models. The first, a Dubins car, is a canonical model that has a well-known analytical solution useful for reachability set comparison. The second, a spacecraft in deep-space, is more complex and more relevant to practical autonomous vehicle control.

A. Dubins Car

The Dubins car models a simple non-holonomic car-like land vehicle constrained by a maximum turning angle $\phi_{\max}$ with a fixed forward speed $v \in \mathbb{R}_{++}$. This maximum steering angle imposes a minimum turning radius $\rho_{\min}$ on the vehicle. The dynamics of Dubins car [3] is given as:

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u = \frac{v}{L}\tan\phi\end{aligned}$$

where the set of admissible controls is $\mathcal{U} = \left[-\frac{v}{L}\tan\phi_{\max}, \frac{v}{L}\tan\phi_{\max}\right]$; hence the state is represented as $\mathbf{x} = [x, y, \theta]$. The control task is concerned with minimizing the total path length, which is equivalent to minimizing the traversal time $J = t_f - t_0$ since $v(t) = v$ is constant. The scenario is depicted visually in Figure 2.

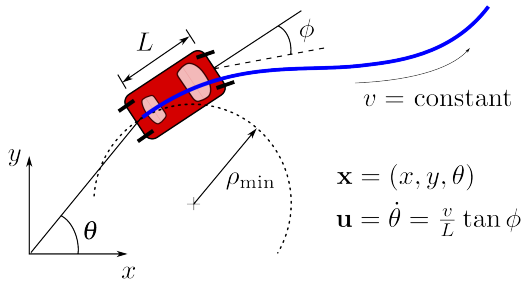

Fig. 2: Dynamics for the Dubins car model.

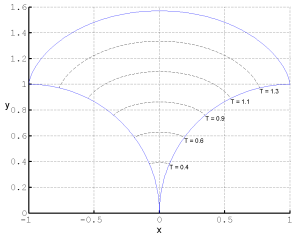

Fig. 3: Reachability sets for an instance of the Dubins car vehicle ($\rho_{\min} = 1, v = 1, L = 1$) for cost threshold horizon times $T \leq \frac{pi}{2}$.

The cost-limited reachable set for a given Dubins car is known analytically, and the projection of this set into the x-y plane is depicted in Figure 3. Heading constraints are also accounted for in the analytical solution, but are not displayed for the sake of simplicity.

The dynamics of the Dubins car are invariant with respect to its state. Therefore, without loss of generality, we may assume the initial state $\mathbf{x}_{\text{init}}$ is always the origin, and set the feature vector to consist of combinations of the target state only. In a real motion planning problem, the motion planning query can always be transformed to this case.

In simulation, we began with a 27-element feature vector, as shown in Table I. The ordering of the elements in Table I was determined by backwards feature selection, from most relevant feature to least relevant feature, as described in Section II-A. In Figure 4, each data point represents the average cost estimation error obtained with the n most relevant features, with n being the value along the x-axis. Results show that one can achieve 10% error from the true cost with just eleven features.

Given the same 2PBVP training examples as for cost prediction, reachability was assessed using a nonlinear SVM

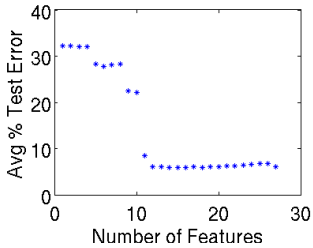

Fig. 4: Dubins car average percent error for cost prediction, per feature set size.

TABLE I: Dubins car features from most important to least important.

1-9	10-18	19-27
$\lvert\theta\rvert$	y	xy
$\cos\theta$	$x\cos\theta$	y^2
θ^2	$x\theta$	x^2
$\sin^2\theta$	$\theta\sqrt{x^2+y^2}$	$\tan\theta$
$\sqrt{x^2+y^2}$	$y\theta$	$x\theta$
$\sqrt{x^2+y^2+\theta^2}$	$x\sin\theta$	$\tan^2\theta$
$\sin\theta$	$y\cos\theta$	$y\tan\theta$
θ	x	$\cos^2\theta$
$y\sin\theta$	$\lvert x\rvert$	$\lvert y\rvert$

classifier with a 4th-order polynomial kernel[3]. The classification results on a test set of 100 new query points are shown in Figure 5. Several kernel functions were attempted; as the trials have shown, the 4-th order kernel seems sufficiently accurate (on the order of 10%). Errors were 4.17%, 7.29%, and 3.13% respectively for each of three cost thresholds (used to establish the cost-reachable boundary): $J_1 = \mu - \sigma, J_2 = \mu,$ and $J_3 = \mu + \sigma$, where μ was the training set mean cost and σ the standard deviation. Note that these results are not shown directly superimposed on the reachability set in Figure 3 due to the complex three-dimensional nature of the state space.

B. Deep-Space Spacecraft

Here we explore a more challenging dynamic system, a three degree-of-freedom deep-space spacecraft, for which reachability sets cannot be described analytically. "Deep-space" refers to the fact that the spacecraft operates in an assumed gravitationally-free environment. For simplicity and clarity of exposition, we omit coupled attitude dynamics and model the vehicle as a point mass that can be accelerated by a thrust vector $\mathbf{T}$ that can point freely in any direction. We consider motions sufficiently small such that propellant use is negligible in comparison to the spacecraft structural mass, and therefore its mass m can be considered approximately constant. This allows system dynamics to be linear, with each coordinate direction acting as classical double integrator:

$$\begin{aligned} \dot{x} &= v_x & \dot{v_x} &= T_x \,/\, m \\ \dot{y} &= v_y & \dot{v_y} &= T_x \,/\, m \\ \dot{z} &= v_z & \dot{v_z} &= T_x \,/\, m \end{aligned}$$

The vehicle is controlled by the thrust vector direction $\hat{\mathbf{n}} \in \mathcal{S}^2$ (the 2-sphere) and throttle $\eta \in [0,1]$; hence $\mathcal{U} = \mathcal{S}^2 \times [0,1]$. The control task is concerned with minimizing the transfer time, $J = t_f - t_0$. The scenario is illustrated in Figure 6.

The features examined for the spacecraft are shown in Table II, listed in order of importance as determined by our backwards-search feature selection algorithm. The average weighted-linear regression estimation error obtained from increasingly large sets of the top-priority features is depicted in Figure 7. Interestingly, it appears that only the top 5 features out of the original 26 in our feature vector are required for the average cost-estimation error to fall below 10%, for an unseen 2PBVP problem.

[3]The SVM classifier was implemented using the MATLAB Statistics toolbox.

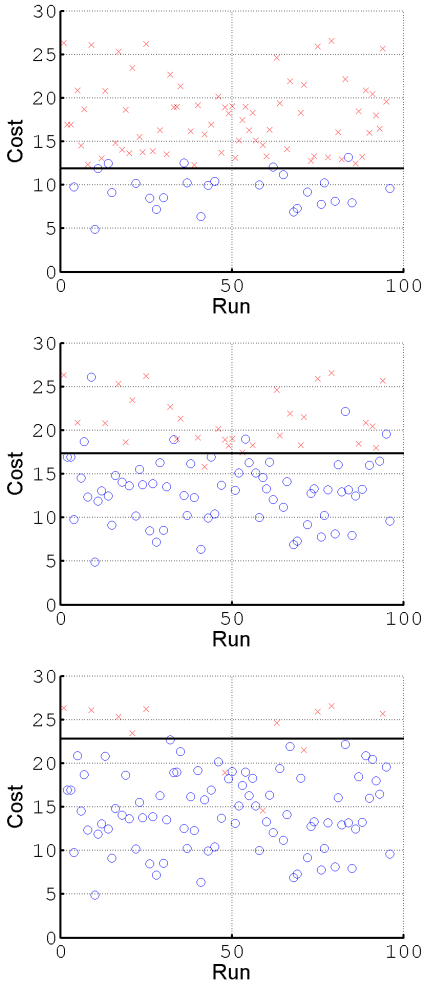

Fig. 5: Trained reachability set for the Dubins Car example. Any blue circles above the cost threshold (shown as a solid black line) represent false positives, while any red crosses below the threshold are true negatives.

For the full feature set, reachability was assessed using a 4th-order polynomial kernel, similar to the Dubins car. In both cases, the 4th-order polynomial kernel seemed to balance the trade-off between bias and variance for the classifier. The classification of a 100-pt test set of new 2PBVP query points can be seen in Figure 8, yielding errors 8.08%, 11.11%, and 7.07%, respectively, again for cost thresholds $J_1 = \mu - \sigma, J_2 = \mu,$ and $J_3 = \mu + \sigma$.

C. Execution Time

The main purpose of this research is to reduce the amount of time required by a motion planning algorithm to determine a reachable set of states, along with the cost to connect to those states. Table III confirms that by using machine learning techniques, we are able to drastically reduce the computation time necessary for these operations by as much as 3 orders of magnitude.

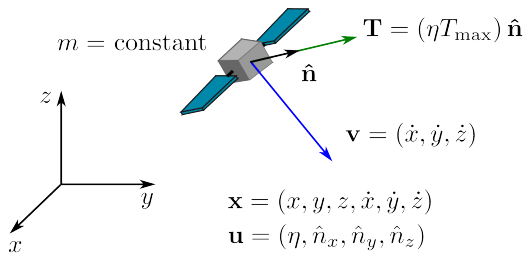

Fig. 6: Dynamics for the deep space spacecraft.

TABLE II: Deep space spacecraft features, from most important to least important

1-9	10-18	19-26
$\sqrt{\|\|\mathbf{x}_f\|\|^2 + \|\|\mathbf{v}_0\|\|^2}$	$\|\dot{y}_0\|$	y_f^2
$\|\|\mathbf{x}_f\|\|$	y_f	$y_f/\dot{y}_0$
$\dot{y}_0$	$\|\dot{x}_0\|$	$x_f/\dot{x}_0$
$\dot{z}_0$	$\|\dot{z}_0\|$	$z_f/\dot{z}_0$
$\dot{x}_0$	x_f	$\|\|\mathbf{s}_0\|\|$
$\|\|\mathbf{v}_0\|\|$	$\|\|\mathbf{x}_f\|\|/\|\|\mathbf{v}_0\|\|$	$(y_f/\dot{y}_0)^2$
$\dot{y}_0^2$	z_f	$(x_f/\dot{x}_0)^2$
$\dot{z}_0^2$	x_f^2	$(z_f/\dot{z}_0)^2$
$\dot{x}_0^2$	z_f^2	
$\mathbf{x}_f = [x_f, y_f, z_f]^{\mathrm{T}}$	$\mathbf{v}_0 = [\dot{x}_0, \dot{y}_0, \dot{z}_0]^{\mathrm{T}}$	$\mathbf{s} = [x/\dot{x}_0, y/\dot{y}_0, z/\dot{z}_0]^{\mathrm{T}}$

IV. Conclusion

We have succesfully shown that machine learning algorithms can be used as heuristics for estimating the steering costs and reachability sets for dynamically-constrained systems. By training local regression algorithms and nonlinear classifiers with numerical solutions to 2PBVPs, we can radically reduce the computation time of the steering function – by up to three orders of magnitude in our numerical experiments – for an online, sampling-based motion planning algorithm. The effectiveness of our approach was demonstrated consistently on two representative systems: the Dubins car and a deep-space spacecraft. For the Dubins car, we achieved an average error of 4.86% for reachability set classification and 6.14% error for cost estimation. For the deep-space spacecraft, we achieved an error of 8.75% for classification and 4.76% for cost estimation. We are hence optimistic that these techniques will help enable real-time optimal sampling-based motion planning for real-world applications.

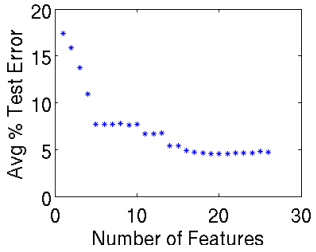

Fig. 7: Deep-space spacecraft average percent error for cost prediction, per feature set size.

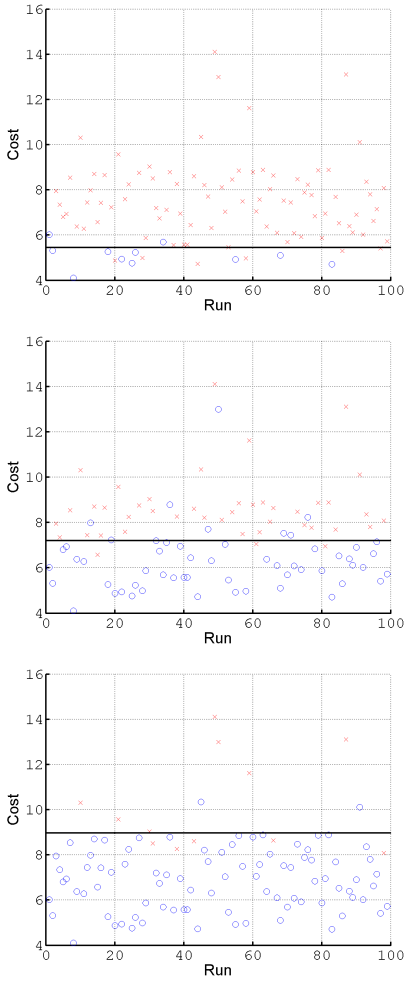

Fig. 8: Trained reachability set for the Deep Space spacecraft example. Blue circles above the cost threshold (shown as a solid black line) represent false positives and red crosses below the threshold are true negatives.

TABLE III: Average computation time comparison for the two-point boundary value problem solver, the machine learning cost approximation (best fit model using all features), and the machine learning reachability classification (average over all 3 cost thresholds).

Application	2PBVP Solver	ML Cost	ML Reachability
Dubins Car	1.23 s	7.5 μs	1.61 ms
Spacecraft	10.3 s	7.6 μs	1.62 ms

References

[1] Brendan Burns and Oliver Brock. Sampling-Based Motion Planning Using Predictive Models. In *Proc. IEEE Conf. on Robotics and Automation*, pages 3120–3125, Barcelona, Spain, April 2005. IEEE, IEEE.

[2] Dušan M. Stipanović, Inseok Hwang, and Claire J. Tomlin. Computation of an Over-Approximation of the Backward Reachable Set using Subsystem Level Set Functions. In *Dynamics of Continuous, Discrete and Impulsive Systems*, volume 11 of *A: Mathematical Analysis*, pages 397–411. Watam Press, 2004.

[3] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 40 West 20th Street, New York, NY 10011-4211, USA, July 2006.