# A Machine Learning Approach for Real-Time Reachability Analysis

Ross E. Allen,* Ashley A. Clark,* Joseph A. Starek,* and Marco Pavone

***Abstract*— Assessing reachability for a dynamical system, that is deciding whether a certain state is reachable from a given initial state within a given cost threshold, is a central concept in controls, robotics, and optimization. Direct approaches to assess reachability involve the solution to a two-point boundary value problem (2PBVP) between a pair of states. Alternative, indirect approaches involve the characterization of reachable sets as level sets of the value function of an appropriate optimal control problem. Both methods solve the problem accurately, but are computationally intensive and do no appear amenable to real-time implementation for all but the simplest cases. In this work, we leverage machine learning techniques to devise query-based algorithms for the approximate, yet real-time solution of the reachability problem. Specifically, we show that with a training set of pre-solved 2PBVP problems, one can accurately classify the cost-reachable sets of a differentially-constrained system using either (1) locally-weighted linear regression or (2) support vector machines. This novel, query-based approach is demonstrated on two systems: the Dubins car and a deep-space spacecraft. Classification errors on the order of 10% (and often significantly less) are achieved with average execution times on the order of milliseconds, representing 4 orders-of-magnitude improvement over exact methods. The proposed algorithms could find application in a variety of time-critical robotic applications, where the driving factor is computation time rather than optimality.**

## I. Introduction

Autonomous dynamical systems such as the recently-proposed Google Driverless Car or the SpaceX Dragon capsule present challenging control questions, including: How can a car choose a safe path through moving, unpredictable traffic while maintaining lawful behavior? Which spacecraft approach configurations will inevitably lead to collision with a target, regardless of future avoidance maneuver attempts? These types of questions appear regularly in the fields of optimal control, kinodynamic motion planning, and differential games, and all center on the concept of *reachability*. Conceptually-speaking, the *reachable set* from some initial state $\mathbf{x}_0$ is the set of states traversed by application of all possible admissable control sequences, $\mathbf{u}(t)$, from $\mathbf{x}_0$ [1]. Specifically, for dynamical systems, we emphasize the *cost-limited reachable set* – the set of states for which the optimal cost to reach these states, $J^*$, from $\mathbf{x}_0$ is less than or equal to a given cost threshold, $J_{th}$. The *reachability* problem is then to assess whether a certain state $\mathbf{x}$ belongs to the cost-limited reachable set, see Figure 1.

The authors are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305, {rallen10,aaclark,jstarek,pavone}@stanford.edu.

*These authors contributed equally to this work.

This work was supported in part by the National Aeronautics and Space Administration STRO-ECF Grant #NNX12AQ43G and the United Technology Research Council's Graduate Fellowship in Aerospace Systems.

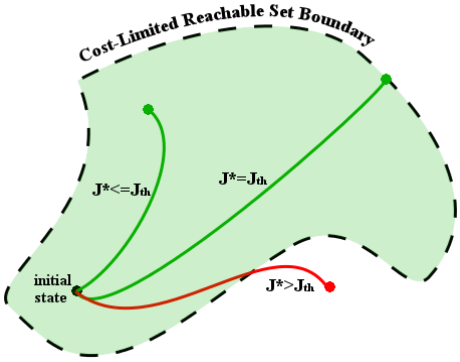


**Fig. 1:** Simplified, 2D illustration of a cost-limited reachable set for an initial state of a dynamical system. Green endpoints are reachable and red endpoints are non-reachable for the given threshold $J_{th}$.

In the geometric case (without dynamic constraints), the freedom of robots to maneuver on straight lines in any direction makes reachability easy to assess for shortest-path objective functions: cost-limited reachable sets form $n$-spheres about the current state $\mathbf{x} \in \mathbb{R}^n$. For dynamically-constrained vehicles, on the other hand, boundaries demarcating cost-limited reachable sets can be much more difficult to assess. Numerical approximations of the reachable set become necessary; however, the cost of computing convergent approximations has a complexity that is exponential in the dimension of the state space [2], making them prohibitively expensive for real time applications. This motivates the need of alternative methods for the rapid determination of cost-limited reachable set approximations.

*Statement of contributions*: In this paper, we show that supervised machine learning techniques [3] can be used to accurately predict cost-limited reachable sets of dynamical systems in real-time. To demonstrate, we use two different machine learning algorithms and compare the accuracy and efficiency of each. The algorithms employed are (1) a locally-weighted regression algorithm for predicting cost function values between query points $\mathbf{x}_a$ and $\mathbf{x}_b$, which determines if $\mathbf{x}_b$ is cost-limited reachable from $\mathbf{x}_a$, and (2) a support vector machine learning algorithm for nonlinear, binary classification of a state's reachable set. The approach generalizes to any type of system dynamics so long as cost training data can be generated. This flexibility suggests that the proposed strategy is applicable to a variety of fields including hybrid system reachability analysis, kinodynamic motion planning, or differential games. In this paper we restrict our attention to continuous, nonlinear, ordinary dynamical systems, and demonstrate our learning strategies on two classical systems of interest: the Dubins car model (see Fig. 2) and a deep-

space spacecraft model (see Fig. 6). These techniques can asses a classification query on the order of milliseconds; a improvement of 4 orders-of-magnitude over an exact classification.

*Organization*: The remainder of the paper is structured as follows: In Section II we review available results for the solution of the reachability problem. In Section III we formally state the problem we wish to solve, and in Section IV we present both the locally-weighted regression algorithm and the support vector machine learning algorithm for reachability analysis. In Section V we present the results from numerical experiments, and in Section VI we draw our conclusions.

## II. Related Work

Reachability analysis is a well-studied field, which has produced a number of strong theoretical results for certain types of systems. Arguably, the most thoroughly studied class is represented by discrete, linear, time-invariant (LTI) systems, for which it is well-known that an exact solution exists for the case of convex polyhedral control and state spaces [4]. For the controllable, continuous-time case without control constraints, it is well-known that the controllability Gramian can be used to formulate a control law connecting any two states in any specified period of time. However, once control constraints or a control-dependent cost are introduced, analytic solutions are generally no longer available and numerical approximations become necessary. A direct approach requires the numerical solution to a 2PBVP. An alternative, indirect method relies on the numerical computation of the zero-level set of the viscosity solution of the Hamilton-Jacobi-Bellman equations, though with exponential time complexity [2]. Modern research on reachability analysis has therefore focused on producing efficient over- or under-approximations of reachable sets and has progressed from the continuous-time linear case [5], to general nonlinear dynamics [6], [7], to nonlinear differential-algebraic dynamics [8] and hybrid systems [9]. These techniques have been tailored to specific applications such as spacecraft dynamics [10] and aircraft dynamics. All these methods solve the problem accurately, but are computationally intensive and do no appear amenable to real-time implementation for all but the simplest cases.

The *machine learning* strategy presented in this paper offers a radically different approach when compared to previous work on reachability analysis. To date, the most relevant application of machine learning to reachability analysis is in the related problem of classifying sets of $k$ nearest neighbors in Euclidean spaces for some positive integer $k$ [11], called the $k$-$NN$ or pattern recognition problem. Approximate methods for the $k$-$NN$ problem, analogous to our approach for continuous 2PBVP's, include [12] - which computes "fuzzy" nearest-neighbor sets, and [13] - which employs a genetic algorithm to discover nearest neighbors. These approaches, however, are generally limited to straight-line distance metrics. By using nonlinear, binary classification and locally-weighted linear regression, our approach eliminates these restrictions. It should be noted that the downside of applying heuristics like machine learning is that theoretical correctness guarantees can no longer be made – only empirical data can be given. For many applications, such as real-time sampling-based motion planning, this may be sufficient.

## III. Problem Statement

Consider a dynamical system $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t)$ evolving over a time horizon $[t_0, t_f]$ and a Bolza-form cost

$$J = K[\mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} L[\mathbf{x}(\tau), \mathbf{u}(\tau), \tau]\,\mathrm{d}\tau. \tag{1}$$

Define the *cost-limited reachable set*, for the arbitrary state $\mathbf{x}_a$, as [1]:

$$\begin{aligned} R(\mathbf{x}_a, \mathcal{U}, J_{th}) = \{&\mathbf{x}_b \in \mathcal{X} \mid \exists \mathbf{u} \in \mathcal{U} \text{ and} \\ &\exists t' \in [t_0, t_f] \text{ s.t. } \mathbf{x}(t') = \mathbf{x}_b \text{ and } J^* \leq J_{th}\}, \end{aligned} \tag{2}$$

where $\mathbf{x}(t)$ is the state vector as a function of time, $\mathbf{u}(t)$ is the control vector as a function of time, $\mathcal{X}$ is the state space, $\mathcal{U}$ is the set of admissible controls, $J^*$ is the optimal cost from $\mathbf{x}_a$ to $\mathbf{x}_b$, $K$ is the terminal cost function, $L$ is the incremental cost function, and $J_{th}$ is a given threshold. Specifically, the optimal cost $J^*$ is given by the solution to the optimal control problem:

$$\begin{aligned} J^* = \underset{\mathbf{u}}{\text{minimize}} \quad & J(\mathbf{x}, \mathbf{u}, t) \\ \text{subject to} \quad & \mathbf{x}(t) \in \mathcal{X} \quad \text{for all } t \in [t_0, t_f] \\ & \mathbf{u}(t) \in \mathcal{U} \quad \text{for all } t \in [t_0, t_f] \\ & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \\ & \mathbf{x}(t_0) = \mathbf{x}_a, \mathbf{x}(t_f) = \mathbf{x}_b. \end{aligned} \tag{3}$$

The problem in equation (3) is referred to as 2-point Boundary Value Problem (2PBVP) or *steering problem*.

The objective of this paper is to design fast, query-based algorithms to assess whether a certain state $\mathbf{x}$ belongs to the cost-limited reachability set $R(\mathbf{x}_a, \mathcal{U}, J_{th})$.

## IV. Query-Based Reachability Algorithms

To solve the reachability problem stated in Section III we consider two approaches leveraging two different techniques from machine learning. The first approach approximates the solution $J^*$ using a locally-weighted linear regression algorithm [3, Ch. 3]; see Section IV-A. The second uses a support vector machine (SVM) [3, Ch. 7], described in section IV-B, to approximate the nonlinear boundary between a state's reachable and non-reachable set. To generate training data for these learning algorithms we exactly[1] solve a "large" number of 2PBVPs given by equation (3) for various, randomly-generated query pairs $(\mathbf{x}_a, \mathbf{x}_b)$. The 2PBVP solver employs Chebyshev Pseudospectral Methods and sequential quadratic programming and is described in Section IV-C.

Note that since our supervised learning algorithms are not derived from an assumed form of system dynamics (e.g., linear, polynomial, etc.), we expect that they may be applied to arbitrarily complex systems – e.g. differential-algebraic systems, differential-inclusion systems, hybrid systems or

[1]The term 'exactly' is used to distinguish this solution from the approximate solutions provided by the supervised learning techniques. In practice, equation (3) is solved to a given tolerance.

even systems with black-box dynamics – without sacrificing on-line running time (perhaps requiring more offline training time, however).

### A. Linear Regression Approach

For a given system (i.e., fixed cost function, dynamics, control constraints and state constraints) each 2PBVP in equation (3) differs only in the initial and final conditions. As a result, the optimal cost, $J^*$, can be thought of as a function of the boundary states, $\mathbf{x}_a$ and $\mathbf{x}_b$ (also referred to as a query pair), parameterized by system dynamics and constraints. Accordingly we write $J^* = J^*(\mathbf{x}_a, \mathbf{x}_b; f, \mathcal{X}, \mathcal{U})$. The idea then is interpolate neighboring pre-computed queries using regression techniques. Due to its robustness and simplicity, we demonstrate the idea using locally-weighted linear regression, implemented as:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \sum_{i=1}^{m} w^{(i)} \left( J^{*(i)} - \boldsymbol{\theta}^{\mathrm{T}} \phi(\tilde{\mathbf{x}}^{(i)}) \right)^2$$

where $\tilde{\mathbf{x}}^{(i)} \in \mathbb{R}^d$ is the $i$th training example (query pair), $m$ is the number of training examples, $\boldsymbol{\theta} \in \mathbb{R}^n$ is the training parameter vector, $\phi : \mathbb{R}^d \to \mathbb{R}^n$ is the feature mapping, $J^{*(i)}$ is the optimal cost of the $i$th training query, and $w^{(i)}$ is the weight given to the $i$th residual. As it is well known, the solution is $\boldsymbol{\theta}^* = \left( \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{W} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{W} \mathbf{J}^*$, yielding $\hat{J} = \boldsymbol{\theta}^{*\mathrm{T}} \phi(\tilde{\mathbf{x}})$ as the estimated cost, where $\mathbf{W} \in \mathbb{R}^{k \times k}$ is a diagonal matrix of weights $w^{(i)}$, $\mathbf{J}^* \in \mathbb{R}^k$ is the vector of optimal costs, and $\boldsymbol{\Phi} \in \mathbb{R}^{k \times n}$ is the matrix of all $k$ feature vectors.

*Model Selection:* To improve interpolation performance, feature vector components are normalized to unit scaling by defining the weights $w^{(i)}$ as:

$$w^{(i)} = \exp\left( \frac{\left|\left| \mathbf{v}^{(i)} \right|\right|^2}{2\tau^2} \right), \text{ with } \mathbf{v}_k^{(i)} = \frac{\phi\left(\tilde{\mathbf{x}}^{(i)}\right)_k - \phi(\tilde{\mathbf{x}})_k}{\text{range}\left(\phi(\tilde{\mathbf{x}})_k\right)},$$

where $\tau > 0$ is the bandwidth parameter, and $\text{range}(\phi(\tilde{\mathbf{x}})_k)$ is the maximum extent of the $k$th feature ($k = 1, \ldots, n$). Due to the tradeoff between over- and under-fitting as $\tau$ is varied, we ran $k$-fold cross-validation [3, Ch. 1] to find the value of $\tau$ that yielded the lowest average percent error over all $k$ training queries.

*Feature Selection:* As previously mentioned, 2PBVP query pairs are mapped to feature vectors. In order to more effectively approximate the true cost function, one should carefully choose a mapping that produces linearly-independent features that are relevant to the given 2PBVP (e.g., endpoint norms, energy values, ratios of boundary states, etc). In an effort to identify features with the most significant impact on cost approximation accuracy, we ran a backward feature selection search as part of our cost-prediction algorithm training. This not only illustrates the trade-off in approximation error and feature vector size, but also provides feature vector design intuition for the particular application.

### B. SVM Approach

The idea behind this approach is to approximate the reachability boundary using a nonlinear classifier, separating training queries into reachable and non-reachable sets. The classifier can then be used to estimate whether new query point pairs $(\mathbf{x}_a, \mathbf{x}_b)$ 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 nonlinear kernel function was employed; a basic description of which follows. SVM seeks to design a function $h\left(\mathbf{w}^{\mathrm{T}} \tilde{\mathbf{x}} + b\right)$ affinely dependent on examples $\tilde{\mathbf{x}}$ with coefficients $\mathbf{w}$ and intercept or "bias" $b$ that correctly maps an attribute vector $\tilde{\mathbf{x}} \in \mathbb{R}^d$ to a label $y \in \{-1, +1\}$. For this application, a label of -1 correlates to an unreachable case, where +1 correlates to a reachable case. These parameters are designed to minimize the number of misclassifications over a set of training examples $\mathcal{S} = \left\{ \left( \tilde{\mathbf{x}}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{m}$, called the *training set*, without overfitting so as to generalize well to newly-encountered queries (i.e., test cases). SVM accomplishes this by solving the optimization problem:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle \tilde{\mathbf{x}}^{(i)}, \tilde{\mathbf{x}}^{(j)} \right\rangle \\ \text{subject to} \quad & \alpha_i \geq 0, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned} \tag{4}$$

where $\langle \cdot \rangle$ represents the inner product of two vectors. The points $\tilde{\mathbf{x}}^{(i)}$ with non-zero Lagrange multipliers $\alpha_i$ are called *support vectors*. In practice, $\ell^1$-regularization – which introduces slack variables to reform equation (4) – is performed to handle the case in which the decision boundary cannot perfectly separate the data [3]. Note that the optimization problem and predictor, given by

$$h\left(\mathbf{w}^{\mathrm{T}} \tilde{\mathbf{x}} + b\right) = h\left( \sum_{i=1}^{m} \alpha_i y^{(i)} \left\langle \tilde{\mathbf{x}}^{(i)}, \tilde{\mathbf{x}} \right\rangle + b \right),$$

depend only on the inner products of examples. To extend SVM to nonlinear boundaries, we replace these inner products with a function $K(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) = \phi(\tilde{\mathbf{x}})^{\mathrm{T}} \phi(\tilde{\mathbf{z}})$ called the *kernel function*, which allows the SVM to operate with a linear boundary in a feature space defined by feature vectors $\phi(\tilde{\mathbf{x}})$ that is nonlinear in terms of attributes $\tilde{\mathbf{x}}$. For the nonlinear SVM classifier in this paper, we use a $p$th order polynomial kernel (with constant $c \in \mathbb{R}$), that is:

$$K(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) = \left( \phi(\tilde{\mathbf{x}})^{\mathrm{T}} \phi(\tilde{\mathbf{z}}) + c \right)^p.$$

### C. Training Data Generation

In order to train the locally-weighted regression and SVM algorithms previously discussed, it is necessary to generate "true" examples of steering problems with known optimal cost. This was accomplished by solving – during an offline-phase – a large number of 2PBVP problems for our chosen

systems and recording the query pairs (i.e., initial and final states) as attributes and optimal costs as outputs. The solution to the optimal control problem stated in equation (3) is obtained in a two-step fashion. First the continuous-time problem is *time-discretized* and transformed into a nonlinear programming problem (NLP). Subsequently an NLP solution technique, such as sequential quadratic programming (SQP) [14, Ch. 18], is employed to solve the NLP and, therefore, approximate the solution to the original optimal control problem. The time-discretization method chosen for our work is the Chebyshev Pseudospectral Method [15] because of its accuracy and ability to extend to more general dynamic constraints (e.g., differential-algebraic equations or differential inclusions). This method works by approximating the state, $\mathbf{x}(t)$, and control, $\mathbf{u}(t)$, trajectories with $N$th degree Lagrange polynomials and then only enforcing the dynamic constraints, $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t)$, at the Chebyshev-Gauss-Lobatto (CGL) points. This creates a NLP problem where the solution vector contains the values of the state and control variables at these CGL points. As shown in [15], the NLP transformation of the problem in equation (3) is given as:

$$\begin{aligned} \underset{\boldsymbol{X},\boldsymbol{U}}{\text{minimize}} \quad & J^N[\boldsymbol{X}, \boldsymbol{U}, \tau_f] \\ \text{subject to} \quad & \boldsymbol{g}_l \leq \boldsymbol{g}[\boldsymbol{x}_k, \boldsymbol{u}_k, \tau_k] \leq \boldsymbol{g}_u \\ & \boldsymbol{f}\left[\frac{2}{\tau_b - \tau_a}\boldsymbol{d}_k, \boldsymbol{x}_k, \boldsymbol{u}_k, \tau_k\right] = 0 \\ & \mathbf{x}(\tau_0) = \mathbf{x}_a, \mathbf{x}(\tau_f) = \mathbf{x}_b \\ & \text{for } k = 0, ..., N, \end{aligned} \tag{5}$$

where $J^N$ is the $N$th order approximation of the cost function, $\boldsymbol{x}_k \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$ are the state and control values at the CGL points, $\tau$ is a transformed time variable, $\boldsymbol{g}$ is a condensed representation of the state and control constraints, and $\boldsymbol{d}_k$ is a product of a differentiation operation. For brevity we define $\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N]$ and $\boldsymbol{U} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_N]$. See [15] for more information. The solution to the NLP presented in equation (5) requires an initial guess. For the cases studied in this paper, a linear interpolation between boundary values proved sufficient for such a guess.

For the training of the machine learning algorithms, any 2PBVPs with infinite cost, i.e., those with no feasible solution to the NLP given in equation (5), were neglected from training. To ensure a well-distributed sampling of the training queries, training data was generated using the Halton sequence, assuming a hypercubical state space.

## V. Application and Results

To evaluate the effectiveness of the proposed algorithms we selected two example models. The first, a Dubins car, is a canonical model for which analytical results are available. The second, a spacecraft in deep-space, is more complex and more relevant to practical autonomous vehicle control. For each case, we train our algorithms on sets of 1000 pre-computed reachability queries and test the resulting parameters against a separate set of 100 new queries.

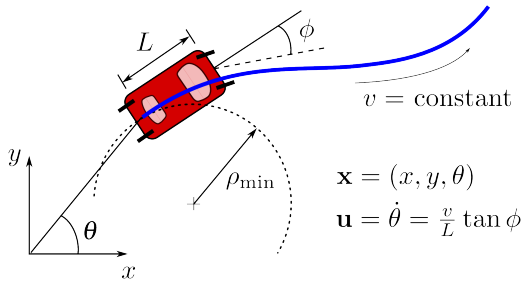


**Fig. 2:** Dynamics for the Dubins car model.

### A. Dubins Car

*Model:* The Dubins car models a simple non-holonomic car-like land vehicle that is constrained by a maximum turning angle $\phi_{\max}$, and has a fixed forward speed $v \in \mathbb{R}_{>0}$. This maximum steering angle imposes a minimum turning radius $\rho_{\min}$ on the vehicle. The dynamics of a Dubins car [1] are given as:

$$\dot{x} = v\cos\theta, \quad \dot{y} = v\sin\theta, \quad \dot{\theta} = u = \frac{v}{L}\tan\phi,$$

where the set of admissible controls is $\mathcal{U} = \left[-\frac{v}{L}\tan\phi_{\max}, \frac{v}{L}\tan\phi_{\max}\right]$. The state is given by $\mathbf{x} = [x, y, \theta]$. 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 speed is constant. The scenario is depicted visually in Figure 2.

The cost-limited reachable set for a given Dubins car is known analytically [16], the boundary of which can be described by the intersection of two congruent cardiods. The cardiods translate and rotate depending upon the time horizon under consideration. For time horizons and rotation angles that satisfy the inequality $0 \leq \theta \leq vt/\rho$, the boundary can be simplified and described by:

$$\begin{aligned} x(\theta) &= \rho\sin\theta + (vt - \rho\theta)\cos\theta, \\ y(\theta) &= -\rho(1 - \cos\theta) + (vt - \rho\theta)\sin\theta. \end{aligned} \tag{6}$$

The projection of this simplified set into the $x$-$y$ plane is depicted in Figure 3. While heading constraints are accounted for in the analytical solution, they are not displayed in the figure for the sake of simplicity. The dynamics of the Dubins car are invariant with respect to its initial state. Therefore, without loss of generality, we may assume the initial state $\mathbf{x}_a$ is always the origin, and set the feature vector to consist of combinations of the target state only.

*Results:* The results of our training and feature selection are listed in Table I with the features listed in the order of relevance, from most relevant feature to least relevant feature. The corresponding average cost estimation error for each group of $n$ most relevant features are shown in Figure 4. Results show that one can achieve 5.3% error[2] from the true cost with just twelve features. Note that, even though the learning algorithm does not have prior knowledge of the

[2]Here, error is defined as the difference between the true and predicted costs divided by the true cost averaged over all training examples.

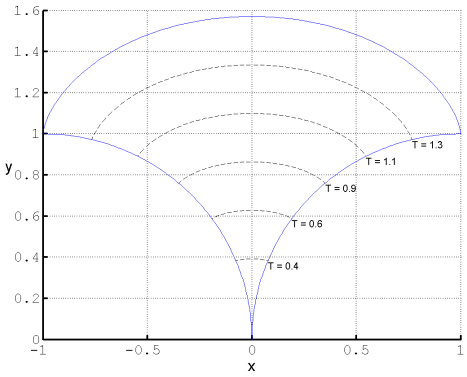


**Fig. 3:** Reachability sets for an instance of the Dubins car vehicle ($\rho_{\min} = 1, v = 1, L = 1$) for cost threshold horizon times $T \leq \frac{\pi}{2}$.

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

| 1-12 | 13-24 | 25-36 |
|---|---|---|
| $\cos\theta$ | $y\cos\theta$ | $1/\tan\theta$ |
| $\sin^2\theta$ | $y\theta$ | $x/\tan\theta$ |
| $\cos^2\theta$ | $\theta$ | $1/\cos\theta$ |
| $\lvert\theta\rvert$ | $\theta\sqrt{x^2+y^2}$ | $x/\cos\theta$ |
| $\theta^2$ | $y\sin\theta$ | $y/\tan\theta$ |
| $\sqrt{x^2+y^2+\theta^2}$ | $x\sin\theta$ | $1/\sin\theta$ |
| $\sqrt{x^2+y^2}$ | $\lvert x\rvert$ | $x/\sin\theta$ |
| $x$ | $xy$ | $y/\sin\theta$ |
| $x\cos\theta$ | $y^2$ | $y\tan\theta$ |
| $\sin\theta$ | $x^2$ | $y/\cos\theta$ |
| $x\theta$ | $\lvert y\rvert$ | $tan^2\theta$ |
| $y$ | $\tan\theta$ | $x\tan\theta$ |

analytical form of reachable set as given in equation (6), the feature selection process gives high importance to the terms that appear in the analytical form. This is a crucial result as it implies that the machine learning approach is robust enough to generate accurate fits for unknown, nonlinear dynamics and cost functions.

Given the same 2PBVP training examples as for cost prediction, reachability was assessed using a nonlinear SVM classifier with a 4th-order polynomial kernel. The classification results on a test set of 100 new query points are shown in Figure 5. Several kernel functions were attempted

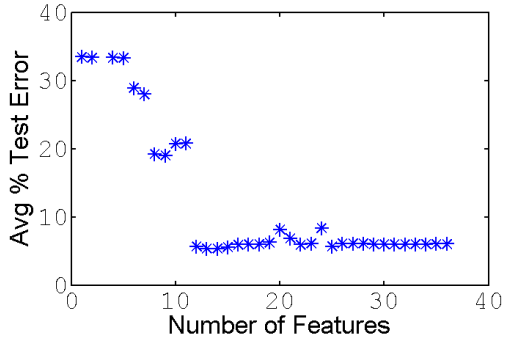


**Fig. 4:** Results of feature selection for the Dubins car showing the average test error percentage vs number of features used.

and the 4-th order kernel proved to be the most accurate for this system. Test errors[3] were 4.17%, 7.29%, and 3.13% respectively for each of three cost thresholds used to establish the cost-reachable boundary, namely, $J_1 = \mu - \sigma, J_2 = \mu$, and $J_3 = \mu + \sigma$, where $\mu$ was the training set mean cost and $\sigma$ the standard deviation.

Due to the complexity of displaying the final $\theta$-heading values in addition to the $x$-$y$ position values, these results are not shown directly superimposed on the reachability set in Figure 3. Instead, Figure 5 plots the test classifications generated for both the linear regression and SVM approach with the cost threshold, $J_{th}$, shown as a solid black line. It can be seen that most misclassifications occur in the direct cost-vicinity of the cost threshold. In simple terms this means that even when the algorithms misclassify points, they aren't too far from the actual cost.

One noticeable outlier occurs for the lowest threshold data where the SVM misclassifies a high-cost test case. This misclassification occurred because the final heading happens to point back toward the initial state for this testing example. Note that feature selection revealed a high importance of $|\theta|$ and $\theta^2$. Since $-\pi \leq \theta \leq \pi$, the anti-parallel alignment can spoof the SVM into thinking it's nearly a straight-line path to the end point when, in fact, it requires almost an entire loop to reach the final state. Altering the feature vector eliminates this outlier.

### *B. Deep-Space Spacecraft*

*Model:* Here we explore a more challenging dynamic system, a three degree-of-freedom deep-space spacecraft, for which reachable sets are not described analytically. "Deep-space" refers to the fact that the spacecraft operates in a 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 $m$ is considered constant. This allows system dynamics to be linear, with each coordinate direction acting as classical double integrator:

$$\dot{x} = v_x, \quad \dot{y} = v_y, \quad \dot{z} = v_z,$$
$$\dot{v}_x = T_x / m, \quad \dot{v}_y = T_y / m, \quad \dot{v}_z = T_z / m.$$

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 $\mathcal{U} = \mathcal{S}^2 \times [0, 1]$. While the dynamics are linear, the optimal control problem is nonlinear due to the fact that the control constraints are norm bounded and the final time is free. The control task is concerned with minimizing the transfer time, $J = t_f - t_0$. The scenario is illustrated in Figure 6.

*Results:* 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

[3]Here, test error is defined as the number of misclassifications divided by total number of test examples.

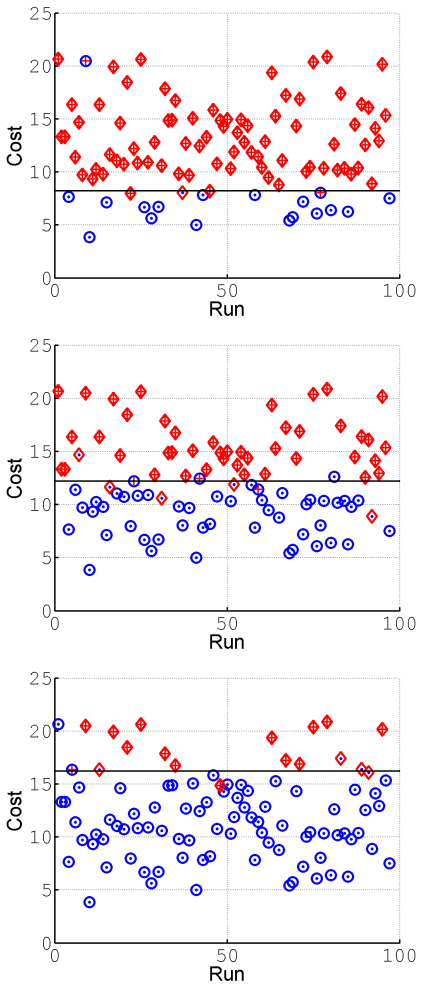


**Fig. 5:** Predicted reachability set plotted with true cost for the Dubins Car with 3 different cost thresholds (black lines). Blue circles = SVM predicted reachable, red diamonds = SVM predicted non-reachable, blue dot = linear regression predicted reachable, red cross = linear regression predicted non-reachable.

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.

For the full feature set, the SVM reachability analysis was again implemented with 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-point test set of new 2PBVP query points can be seen in Figure 8, yielding test errors 8.08%, 11.11%, and 7.07%, respectively, again for cost thresholds $J_1 = \mu - \sigma, J_2 = \mu$, and $J_3 = \mu + \sigma$. Figure 8 directly compares the classifications made be the linear regression and SVM algorithms. Again, most misclassifications occur near the cost threshold.

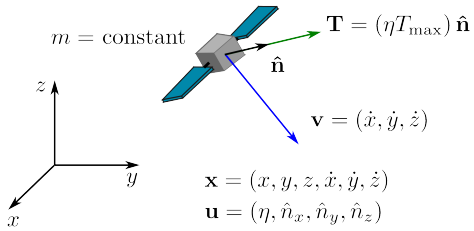


**Fig. 6:** Dynamics for the deep space spacecraft.

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

| 1-9 | 10-18 | 19-26 |
|---|---|---|
| $(x/\dot{x})^2$ | $y^2$ | $\dot{z}^2$ |
| $(y/\dot{y})^2$ | $\|\|x, y, z\|\|/\|\|\dot{x}, \dot{y}, \dot{z}\|\|$ | $\dot{y}^2$ |
| $(z/\dot{z})^2$ | $z^2$ | $\|\|\dot{x}, \dot{y}, \dot{z}\|\|$ |
| $\|\|x/\dot{x}, y/\dot{y}, z/\dot{z}\|\|$ | $y$ | $\dot{x}$ |
| $x/\dot{x}$ | $\|\dot{z}\|$ | $\dot{y}$ |
| $y/\dot{y}$ | $x$ | $\dot{z}$ |
| $z/\dot{z}$ | $\|\dot{x}\|$ | $\|\|x, y, z\|\|$ |
| $z$ | $\|\dot{y}\|$ | $\|\|x, y, z, \dot{x}, \dot{y}, \dot{z}\|\|$ |
| $x^2$ | $\dot{x}^2$ | |

### C. Execution Time & Accuracy

Table III compares the computation time and classification accuracy of each of the three approaches: 1) truth-value determined by a 2PBVP solver[4], 2) locally-weighted linear regression cost estimation, and 3) support vector machine classification. These results confirm that machine learning techniques are able to drastically reduce the computation time for reachability analysis by as much as four orders of magnitude – which comes at the cost of misclassifying some queries.

By definition, the truth-value calculation produces no misclassifications but take seconds or tens-of-seconds to perform each classification. Locally-weighted linear regression produces the most accurate classifications of the machine learning

[4]As previously mentioned, the techniques used to determine the true classifications are not guaranteed to be absolutely correct, but they do represent the state of the art in solving optimal 2PBVPs. See Section IV-C for more details.

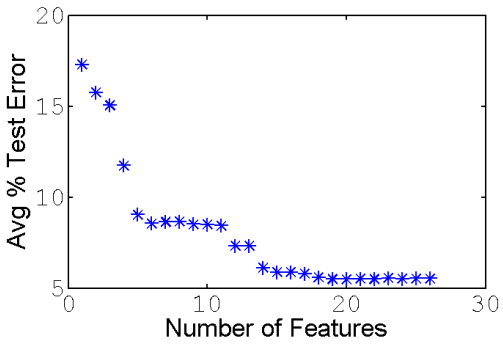


**Fig. 7:** Results of feature selection for the deep-space spacecraft showing the average test error percentage vs number of features used.

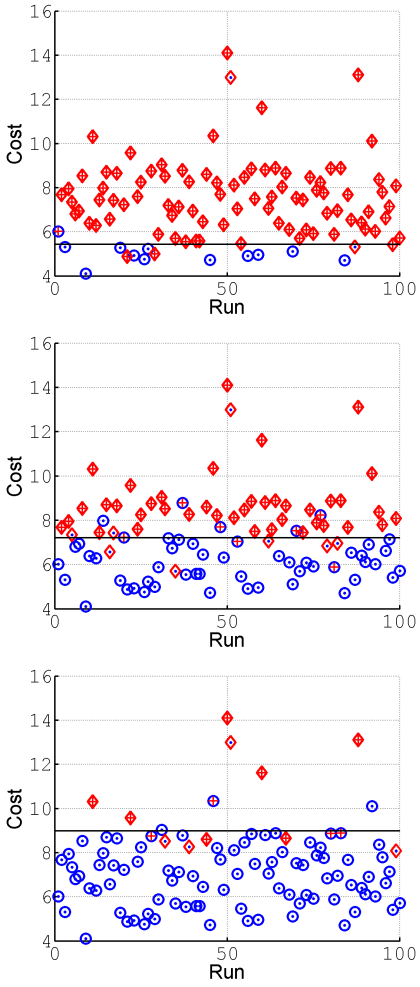


**Fig. 8:** Predicted reachability set plotted with true cost for the deep-space spacecraft with 3 different cost thresholds (black lines). Blue circles = SVM predicted reachable, red diamonds = SVM predicted non-reachable, blue dot = linear regression predicted reachable, red cross = linear regression predicted non-reachable

techniques but can take 20x longer per query to compute when compared to the SVM approach. The relatively large computation time for linear regression is due to the $n \times n$ matrix inversion required for each classification query. This matrix inversion can be avoided if a non-locally-weighted regression scheme is used, but accuracy of classification is severely impacted (results are omitted due to page limitations). Linear regression techniques have the benefit of generating more information by providing an estimate of the optimal cost. The SVM only returns a binary, true-false result. This cost estimation may be valuable beyond reachable set analysis, depending on application.

## VI. Conclusion

We have presented two novel, query based algorithms that leverage machine learning techniques to solve the reachability problem for dynamical systems. By training locally-weighted regression algorithms or nonlinear SVM classifiers with sufficiently-many offline numerical 2PBVP solutions, the online computation time required to assess new reachability queries can be cut dramatically – by up to 4 orders of magnitude in our numerical experiments, provided the user is willing to accept some misclassifications (for the two representative systems studied here, both approaches achieved less than 10% prediction error). Due to its success and its relatively unexplored use within the reachability analysis community, we are optimistic about the potential of machine learning for future reachability applications such as kinodynamic motion planning and differential games.

**TABLE III:** Average computation time and percent of misclassification for the two-point boundary value problem solver, the linear regression cost estimation (best fit model using all features), and SVM classification (average over all 3 cost thresholds).

| System | 2PBVP Solve | | Lin. Reg. | | SVM | |
|---|---|---|---|---|---|---|
| | Time | % Err | Time | % Err | Time | % Err |
| Dubins | 1.23 s | 0.0 | 9.4 ms | 3.8 | 0.44 ms | 4.9 |
| Spacecraft | 10.3 s | 0.0 | 9.0 ms | 5.8 | 0.40 ms | 8.8 |

## References


[1] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, July 2006.

[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 *Dyn. of Cont., Discrete and Impulsive Systems*, volume 11 of *A: Mathematical Analysis*, pages 397–411. Watam Press, 2004.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, volume 1 of *Info. Science and Stat.* Springer, New York, NY, 2006.

[4] Francesco Borrelli, A. Bemporad, and M. Morari. Predictive Control, 2014. In Preparation.

[5] Antoine Girard and Colas Le Guernic. Efficient Reachability Analysis for Linear Systems Using Support Functions. In Myung Jin Chung and Pradeep Misra, editors, *IFAC World Congress*, volume 17, pages 8966–8971, Gangnam-gu Seoul, South Korea, July 2008. Int. Fed. of Automatic Control, IFAC PapersOnLine.

[6] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability Analysis of Nonlinear Systems Using Conservative Approximation. In Oded Maler and Amir Pnueli, editors, *Hybrid Systems: Comp. and Control*, volume 2623 of *Lecture Notes in Comp. Science*, pages 20–35. Springer, Prague, Czech Republic, April 2003.

[7] Romain Testylier and Thao Dang. NLTOOLBOX: A Library for Reachability Computation of Nonlinear Dynamical Systems. In Dang Van Hung and Mizuhito Ogawa, editors, *Autom. Tech. for Verif. and Analysis*, volume 8172 of *Lecture Notes in Comp. Science*, pages 469–473. Springer, Hanoi, Vietnam, Oct 2013.

[8] Matthias Althoff and Bruce H. Krogh. Reachability Analysis of Nonlinear Differential-Algebraic Systems. *IEEE Trans. on Automatic Control*, 59(2):371–383, Feb 2014.

[9] Hervé Guéguen, Marie-Anne Lefebvre, Janan Zaytoon, and Othman Nasri. Safety Verification and Reachability Analysis for Hybrid Systems. *Annual Reviews in Control*, 33(1):25–36, April 2009.

[10] Erik Komendera, Elizabeth Bradley, and Daniel Scheeres. Efficiently Locating Impact and Escape Scenarios in Spacecraft Reachability Sets. In *AAS/AIAA Astrodynamics Specialist Conf.*, pages 1–15, Minneapolis, MN, USA, Aug 2012. AAS, AIAA.

[11] Chin-Liang Chang. Finding Prototypes For Nearest Neighbor Classifiers. *IEEE Trans. on Automatic Control*, C-23(11):1179–1184, Nov 1974.

[12] J. M. Keller, M. R. Gray, and J. A. Givens. A Fuzzy K-Nearest Neighbor Algorithm. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-15(4):580–585, July 1985.

[13] Shinn-Ying Ho, Chia-Cheng Liu, and Soundy Liu. Design of an Optimal Nearest Neighbor Classifier Using an Intelligent Genetic Algorithm. *Pattern Recognition Letters*, 23(13):1495–1503, Nov 2002.

[14] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, 2 edition, 2006.

[15] Fariba Fahroo and I. Michael Ross. Direct Trajectory Optimization by a Chebyshev Pseudospectral Method. *AIAA J. of Guidance, Control, and Dynamics*, 25(1):160–166, Jan-Feb 2002.

[16] E. J. Cockayne and G. W. C. Hall. Plane Motion of a Particle Subject to Curvature Constraints. *SIAM J. of Control*, 13(1):197–220, Jan 1975.