Distance Metric Approximation for State-Space RRTs using Supervised Learning

Mukunda Bharatheesha[1], Wouter Caarls[1], Wouter Jan Wolfslag[1] and Martijn Wisse[1]

***Abstract*— The dynamic feasibility of solutions to motion planning problems using Rapidly Exploring Random Trees depends strongly on the choice of the distance metric used while planning. The ideal distance metric is the optimal cost of traversal between two states in the state space. However, it is computationally intensive to find the optimal cost while planning. We propose a novel approach to overcome this barrier by using a supervised learning algorithm that learns a nonlinear function which is an estimate of the optimal cost, via offline training. We use the Iterative Linear Quadratic Regulator approach for estimating an approximation to the optimal cost and learn this cost using Locally Weighted Projection Regression. We show that the learnt function approximates the original cost with a reasonable tolerance and more importantly, gives a tremendous speed up of a factor of 1000 over the actual computation time. We also use the learnt metric for solving the pendulum swing up planning problem and show that our metric performs better than the popularly used Linear Quadratic Regulator based metric.**

I. INTRODUCTION

Kinodynamic motion planning has been a subject of extensive research in the past two decades and addresses the problem of finding a motion plan for a robotic system that adheres to the kinematic and dynamic constraints. Often, the state space of the system is the chosen candidate for kinodynamic planning as it naturally allows for accounting for the dynamical constraints of the system. However, a well known difficulty is that this problem is PSPACE-hard [1]. Hence, several practical solutions rely on the idea of sampling-based planning. Sampling-based approaches rely on constructing a tree graph structure with the states of the system as the tree nodes and the trajectories of the system between two states as the tree edges. Two approaches that have been extensively researched in this area are the Rapidly Exploring Random Trees (RRT) [2] and Probabilistic Road Maps (PRM) [3]. One of the main difference between the RRT and PRM approaches is the way in which an existing node in the tree is chosen for further expansion. In an RRT, the node to expand from is chosen based on the notion of a *distance* to a randomly sampled node in the state space. On the contrary, PRM-based approaches typically choose a node to expand based on a probabilistic weighting of the density of nodes in different regions of the state space. In our work, we focus on the RRT-based approaches. The RRT approach is briefly outlined in Algorithm 1.

[1]All authors are with the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands {M.Bharatheesha, W.Caarls, W.J. Wolfslag, M.Wisse}@tudelft.nl

Algorithm 1 RRT ((V, E), N)

for $j = 1,\dots N$ **do**
  $\mathbf{x}_{\text{rand}} \leftarrow$ `Sample`
  $\mathbf{x}_{\text{nearest}} \leftarrow$ `Nearest` $(V, \mathbf{x}_{\text{rand}})$
  $(cost_{\text{xnew}}, \mathbf{x}_{\text{new}}) \leftarrow$ `Steer` $(\mathbf{x}_{\text{rand}}, \mathbf{x}_{\text{new}})$
  $X \leftarrow X \cup \{x_{\text{new}}\}$
  $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}}, cost_{\text{xnew}})\}$
end for
return $G = (V, E)$

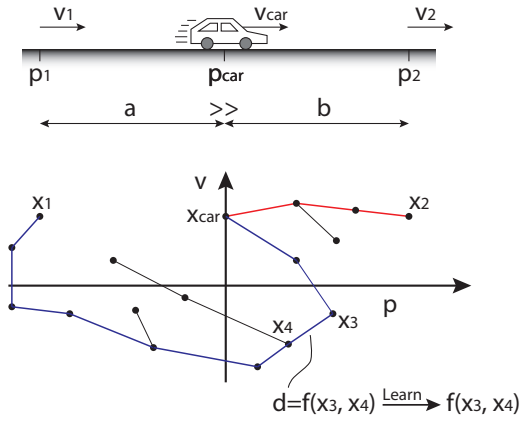

Fig. 1. Simple depiction of the problem of distance metric in state space. While driving a car forward, it is generally easier to reach a point ahead of the car rather than reaching a point behind the car. The tree structures indicate a candidate solution that could be found by RRT.

In the RRT algorithm mentioned above, N indicates the number of tree nodes to be generated, V is the vertex set and E is the edge set of the tree graph. The `Sample` procedure samples a random node from the state space. The `Nearest` procedure determines the nearest neighbour in the tree to the randomly sampled node based on an appropriate distance metric. The `Steer` procedure attempts to connect the nearest neighbour to the randomly sampled node and also gives the corresponding cost to connect the two nodes. Typically, this is accomplished by forward simulation of system dynamics for a specified duration. If this attempt is successful, the randomly sampled node is added as the new node to the set of tree nodes. Otherwise, the state that is eventually reached by the `Steer` procedure is added as the new node to the set of tree nodes.

We explain the importance of distance metric in the state space with a simple example depicted in Fig. 1. The position and velocity of the car are its states. Let us consider the

state at time instant t_0 as $(p_{car},\, v_{car})$. Let t_1 denote the time at which the state is $(p_1,\, v_1)$, which is $10\,m$ ahead of the car. For simplicity, we assume, $v_1 = v_{car}$. Similarly, let $(p_2,\, v_2)$ indicate the state of the car at t_2, which is $10\,m$ behind the car. In the context of state space, we consider $(p_1,\, v_1)$ is *nearer* to $(p_{car},\, v_{car})$ as the car is already moving forward and hence less work is needed. On the other hand, to reach $(p_2,\, v_2)$ which is $10\,m$ behind the car, with $v_2 = v_{car}$, we first reduce the car speed during which we move forward, come to a halt, drive backwards beyond p_2, again halt and then arrive at p_2 with velocity v_2. Hence, $(p_2,\, v_2)$ is considered to be *farther* from $(p_{car},\, v_{car})$ when compared to $(p_1,\, v_1)$. In this sense, the *distance* in state space has the notion of a *cost-to-go* between two states. Furthermore, the optimal cost-to-go between two states is considered as the ideal distance metric in state space [2]. In fact, solving this problem exactly would eventually solve the overall motion planning problem [2].

In the simple car example described earlier, we can exactly solve for the optimal cost-to-go between states (denoted by a and b in Fig. 1) in very short time and thus have a perfect distance metric. However, in systems with pronounced (and possibly non-linear) dynamical effects, computing the optimal cost takes a very large amount of time. This factor limits the possibility of using RRTs for online planning problems. It is also argued in [2] that approximations of these costs can significantly improve the feasibility of plans generated by RRT-based approaches. It is also well documented that the feasibility of the solution generated by state space RRTs is highly sensitive to the distance metric used [4], [5].

A. Relevant Background

In literature, the choice of the distance metric has evolved significantly ever since the basic RRT implementation in [2] used the Euclidean distance. In fact, Euclidean distance is used in an indirect sense in [6] to assess the proximity of the randomly sampled node to the reachable set of a node in the tree and subsequently make the appropriate choice. In a subsequent work, Glassman and Tedrake [5] propose a metric based on the theory of Linear Quadratic Regulators (LQR) and affine dynamics of a non-linear system around a linearization point. They refer to their approach as the Affine Quadratic Regulator (AQR) design. While the work on AQR primarily focuses on the distance metric alone, the authors of [7] use the LQR theory not only to estimate the metric but also to use the resulting optimal policy for propagating the tree further. They also use the asymptotically optimal version of the RRT algorithm, namely RRT*, and hence establish the asymptotic optimality of the resulting plans from their approach.

A primary reason for the choice of LQR-based approaches for approximating the cost-to-go metric is the ease and speed with which they can be numerically computed. These approaches have shown to work with reasonable efficiency in [5], [7], [8]. The strategy of linearizing around sampled random points with either zero inputs or zero velocities or both is however, not a good approximation of the cost-to-go. In principle, such an approach would completely diminish the possibility of utilizing natural system dynamics. This problem is discussed in greater detail in Section II-A. The authors in [5] discuss the situation of linearizing about points with non-zero inputs and velocities too, but mention that the performance of their approach drops off as nonlinear dynamics become prominent.

A solution to address this problem has been proposed in Li and Todorov [9], where nonlinear dynamics are linearized around a nominal trajectory instead of a point and an optimal cost-to-go is obtained iteratively by solving a modified LQR problem (iLQR). One of the primary reasons that this approach has not been popular in the sampling-based planning domain is the time needed to converge to the optimal cost. We defer further details on this approach to Section II-A.

We recall from the car example in Section I, that the nature of the distance metric in state space is nonlinear. In a mathematical sense, the distance metric is a nonlinear mapping $\rho\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$. The iLQR approach discussed earlier can also be considered one such mapping. Existing literature in the area of supervised learning indicates the possibility of approximating nonlinear functions. Our main contribution is showing that the cost calculated by iLQR can be accurately and efficiently approximated using supervised learning techniques. We use Locally Weighted Projection Regression (LWPR) [10], which allows for approximating nonlinear functions in high-dimensional spaces by using locally linear models. After off-line training on a dataset generated with iLQR, the on-line planning stage only requires a single evaluation of this model to approximate the distance between two points. In this way, we enable the RRT-based planning to avail the benefits of the iLQR approach without the increase in computation time. We substantiate the choice of LWPR learning over other supervised learning methods such as neural networks or SVM regression in Section II-B.

The rest of the paper is structured as follows. In Section II, we formally present our problem along with a brief description of iLQR and LWPR algorithms. We follow it up with our proposed solution to combine the two approaches in Section III. In Section IV, we present the experimental results and highlight the resulting benefits. We briefly discuss our results in Section V and finally conclude in Section VI.

II. Problem Description

Let us consider a robotic system with the following nonlinear dynamics:

$$\dot{\mathbf{x}} = f\left(\mathbf{x}, \mathbf{u}\right) \tag{1}$$

where $\mathbf{x}$ is the state vector of the system and $\mathbf{u}$ is the input vector. Given the dynamics described by (1), the Rapidly Exploring Random Tree (RRT) algorithm allows for finding a motion plan to steer the system from an initial state $\mathbf{x}_i$ to a final state $\mathbf{x}_f$. A main ingredient for generating a feasible plan via the RRT approach is the distance metric $\rho\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$. The distance metric is called at every iteration of an RRT. As described earlier, computing the metric in state space requires a significant amount of CPU time.

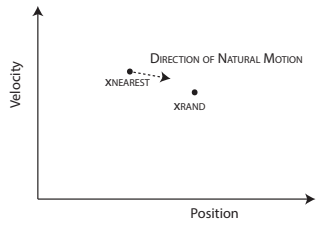

Fig. 2. Influence of Linearization.

Our work proposes a solution that significantly reduces the distance metric computation time. We use a combination of the optimal control approach Iterative Linear Quadratic Regulator (iLQR) and a supervised learning approach called Locally Weighted Projection Regression (LWPR) to achieve the reduction in computation time. We briefly describe the two approaches in the following.

A. Iterative Linear Quadratic Regulator (iLQR)

Typically, approaches in literature that aim to obtain an estimate of the distance metric in state space, use the concept of linearization of a nonlinear system around a point in the state space with zero velocity and/or zero inputs. This has been shown to work reasonably well in [5], [7]. However, these methods are only a good approximation of the nonlinear dynamics provided the point chosen for linearization is an equilibrium point. If this is not the case, the approximation is inaccurate. This problem is illustrated in Fig. 2, where, a typical scenario that arises during every iteration of an RRT is shown Fig. 2. The direction of evolution of the natural system dynamics is also shown. With a linearization approach such as in [7], the linearized model would be deprived of the velocity information at x_{rand} and hence, the cost-to-go to x_{rand} from x_{nearest} would yield a certain input value which eventually influences the cost. However, in reality, this input may not be needed or a very small input might be needed to steer the system to x_{rand}. Thus, approaches that use linearization around an equilibrium point could yield an inaccurate estimate of the actual cost-to-go.

As a consequence, approaches such as the LQR-RRT [7] would need a sufficiently large number of nodes before which a motion plan to the goal can be found. In case of simple nonlinear systems such as a pendulum, the effect might not be very evident. With more complex nonlinear dynamics such as in [9], the resulting increase in the number of nodes due to approximations inaccuracies leads to a significantly large amount of time for the LQR-RRT approach to converge to a solution.

In our work, we use the iLQR approach proposed in [9] for the approximation of the nonlinear dynamics. In this approach, linearization is carried out along a nominal trajectory instead of just a point in the state space. In doing so, a better approximation of the nonlinear dynamics can be achieved. The iLQR approach is presented here as a simple procedure in Algorithm 2. We limit the complete mathematical details due to space restrictions and those that are provided here are directly taken from [9].

Let us consider the discretized version of the system dynamics of Eq. (1):

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k\right) \tag{2}$$

with, the cost function is defined as,

$$J_0 = \frac{1}{2}\left(\mathbf{x}_N - \mathbf{x}^*\right)^T Q_f (\mathbf{x}_N - \mathbf{x}^*) + \frac{1}{2}\sum_{k=0}^{N-1}\left(\mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k\right) \tag{3}$$

where,

- $\mathbf{x}_N$ describes the final state after each execution of the input $\mathbf{u}_k$.
- $\mathbf{x}^*$ is the given target state.
- Q and Q_f are the state cost weighting matrices.
- R is the control-cost weighting matrix.

Using the the equations (2) and (3), the iLQR approach proceeds iteratively by obtaining a nominal open loop trajectory $\mathbf{x}_k$ by applying an input $\mathbf{u}_k$. With an initial input sequence $\mathbf{u}_k = 0$, each iteration produces an improved $\mathbf{u}_k$ by linearizing the system dynamics around the sequence $(\mathbf{x}_k,\, \mathbf{u}_k)$ and solving a modified LQR problem. The iterations continue until a cost convergence is achieved.

Algorithm 2 ILQR (uk, xi, xf, Qf, Q, R, dt, nIter)

$cost_{curr} \leftarrow$ `ILQRCost` $(\mathbf{u}_k,\, \mathbf{x}_i,\, \mathbf{x}_f,\, Q_f,\, Q,\, R,\, dt)$
for $j = 1,\ldots n_{\text{Iter}}$ **do**
  $\delta\mathbf{u} \leftarrow$ `ILQRIterate` $(\mathbf{u}_k,\, \mathbf{x}_i,\, \mathbf{x}_f,\, Q_f,\, Q,\, R,\, dt)$
  $\mathbf{u}_k^{'} \leftarrow \mathbf{u}_k + \alpha\, \delta\mathbf{u}_k$
  $cost_{new} \leftarrow$ `ILQRCost` $\left(\mathbf{u}_k^{'},\, \mathbf{x}_i,\, \mathbf{x}_f,\, Q_f,\, Q,\, R,\, dt\right)$
  if $cost_{new} - cost_{curr} < cost_{threshold}$ **then**
    `TerminateIteration`
  end if
  $cost_{curr} \leftarrow cost_{new}$
  $\mathbf{u}_k \leftarrow \mathbf{u}_k^{'}$
end for
return $\left(\mathbf{u}_k^{opt},\, cost^{opt}\right)$

In Algorithm 2, the `ILQRCost` procedure, computes the cost of the nominal trajectory using (3). The `ILQRIterate` implements the linearization procedure mentioned earlier. Eventually, once the cost converges, the `ILQR` procedure returns the optimal cost-to-go between the states $\mathbf{x}_i$ and $\mathbf{x}_f$.

Although iLQR is a reasonably good method to find the cost-to-go, it is not fast enough for use in planning. Therefore we will use Locally Weighted Projection Regression (LWPR) as a supervised learning technique to decrease the computation time of the cost-to-go, by initially incorporating a learning phase.

B. Locally Weighted Projection Regression (LWPR)

Locally Weighted Projection Regression (LWPR) is a supervised learning approach that has the potential to approximate nonlinear functions in high dimensional spaces [10]. It is a non-parametric and a fast learning approach.

The nonlinear function is learnt as a set of locally linear regression models along particular input dimensions. The linear models are eventually blended using a weighting function based on the area of validity of the local models to obtain the approximation of the nonlinear function.

Formally, given a data set in the form of input-output pairs $(\mathbf{x}, \mathbf{y})$, the LWPR method involves iterating the following function:

$$f(\mathbf{x}) = \frac{1}{W(\mathbf{x})} \sum_{k=1}^{K} \mathbf{w}_k(\mathbf{x}) \psi_k(\mathbf{x}) \tag{4}$$

where,

- $$W(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{w}_k(\mathbf{x}) \tag{5}$$
- $\psi_k(\mathbf{x})$ are the local linear models.
- $\mathbf{w}_k(\mathbf{x})$ is a weighting factor depending on the area of validity of the local linear models.

It is particularly highlighted in [11] that the strength of LWPR learning lies in its ability to incrementally learn from training samples rather than learning from a collected set of samples. We however use the latter approach in this paper. We undertake this approach only for simplicity and the incremental approach remains the main motivating factor for our choice of LWPR learning over other supervised learning approaches such as Artificial Neural Networks. In fact, our eventual goal is to be able to learn the distance metric while constructing the RRT and this work is a first step in that direction.

In the following section, we combine the iLQR and the LWPR approaches that results in a novel method to approximate the distance metric function in the state space.

III. Method

In this section, we describe our novel approach in two parts. In the first part, we explain the creation of the training samples for the LWPR algorithm and the consequent learning of the distance metric function. In the second part, we explain the use of the learnt distance metric function as a part of the RRT building process.

A. Learning the optimal cost function

Like for all supervised learning algorithms, providing adequate amount of training samples is integral to LWPR learning to make a good approximation of the underlying function. A training sample consists of arguments to the function to be learnt and the corresponding function value. In the context of our work, the function to be learnt is the optimal cost-to-go between two states of a nonlinear dynamical system, given by the iLQR algorithm. The arguments to this function are two states in the state space. The procedure we use to learn the optimal cost function is shown in Algorithm 3.

There are two main aspects of the `LearnMetric` procedure. The first is in the lines where the initial state x_i and desired final state x_f are sampled. In order to ensure an unbiased learning, these states are sampled randomly. Furthermore, the random sampling procedure also maintains the essence of the sampling used in RRT-based approaches where each incremental step involves a random sampling of a state from the state space.

Algorithm 3 LearnMetric (nSamples, (dt, nIter))

for $j = 1,\ldots n_{\text{Samples}}$ **do**
  $x_i \leftarrow$ `Sample`
  $x_f \leftarrow$ `Sample`
  $cost_{optimal} \leftarrow$ `ILQR` $(x_i,\, x_f,\, nIter,\, dt)$
  $X_{tr}(j) \leftarrow [x_i;\, x_f]$
  $Y_{tr}(j) \leftarrow cost_{optimal}$
end for
return $\rho_{\text{learn}} =$ `LWPRlearnmetric` $(X_{tr},\, Y_{tr})$

The second aspect is in the last line of the procedure where the distance metric function ρ_{learn} is learnt using the `LWPRlearnmetric` procedure. This is done by initially dividing the obtained samples into a training set and a test set. LWPR uses these two sets for evaluating the quality of the learnt function.

B. Using the learnt distance metric for RRT

Here, we describe the following step after learning the distance metric function. That is, using the learnt distance metric ρ_{learn} while constructing the RRT. The procedure for RRT construction with the learnt metric, `RRTLearn` is illustrated in Algorithm 4.

Algorithm 4 RRTLearn ((V, E), N)

for $j = 1,\ldots N$ **do**
  $x_{\text{rand}} \leftarrow$ `Sample`
  $x_{\text{nearest}} \leftarrow \rho_{\text{learn}}\ (V,\, x_{\text{rand}})$
  $[cost_{xnew}, x_{\text{new}}] \leftarrow$ `iLQR` $(x_{\text{nearest}},\, x_{\text{rand}})$
  $X \leftarrow X \cup \{x_{\text{new}}\}$
  $E \leftarrow E \cup \{(x_{\text{nearest}},\, x_{\text{new}})\}$
end for
return $G = (V,\, E)$

We verify the correctness of the approximated function in two steps. First, the `iLQR` procedure attempts to connect the random state with the nearest state x_{nearest}. The resulting new state x_{new} and the corresponding optimal cost, $cost_{xnew}$ are compared to the predicted cost between x_{nearest} and x_{new} using ρ_{learn}. The cost prediction error, $cost_{error}$ is determined as follows:

$$cost_{error} = \left(\rho_{learn}(x_{\text{nearest}},\, x_{\text{new}}) - cost_{xnew}\right)^2 \tag{6}$$

The performance of our algorithm is compared against the LQR-RRT approach in [7] in terms of the number of nodes required to find a solution. We now proceed by presenting the experimental results from our work.

IV. Experimental Results

We consider the simple pendulum swing up problem to evaluate the RRTLearn algorithm. The physical parameters

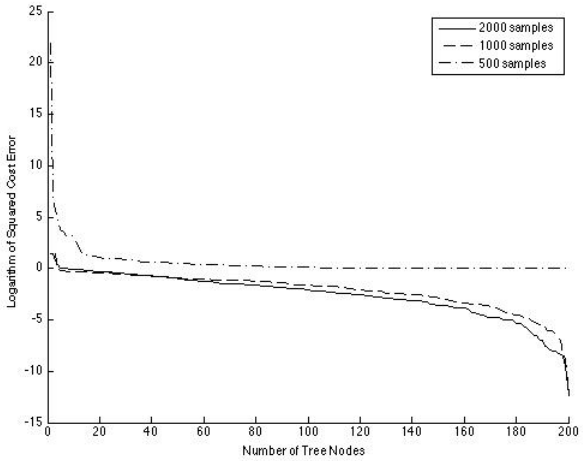

Fig. 3. Logarithm of the squared cost error over all tree nodes.

TABLE I

TIME FOR COMPUTING STATE SPACE DISTANCE (AVERAGED OVER 20 RUNS).

	iLQR Metric	Learnt iLQR Metric	LQR Metric
1 Node	$0.4127\,s$	$0.00016\,s$	$0.0045\,s$
100 Nodes	$41.18\,s$	$0.0075\,s$	$0.0057\,s$
500 Nodes	$205.2\,s$	$0.037\,s$	$0.0081\,s$

and the linearization approach are considered in accordance with [9]. We consider a pendulum with bob mass $m = 1\,kg$, length $l = 1\,m$ and a damping of $b_d = 0.01\,kg/s$ at the mounting joint. We assess the performance based on three factors; correctness of the learnt metric, state space coverage and using the metric to perform planning. Our work has been implemented in $MATLAB$ using the available resources for LWPR in [11] and for iLQR in [12]. For the RRT implementations, the Robotic Toolbox [13] has been used and adapted to our requirements.

We begin with the verification of the learnt metric with the actual iLQR cost based on different number of training samples used for LWPR learning. We use (6) for this purpose. It is clear from Fig. 3 that the squared error is small for a majority of the nodes and those with large values are the inevitable outliers which are bound to be present when learning approaches are used. It is pertinent to note that the accuracy of the prediction drops considerably if the number of training samples are reduced.

The main advantage of the RRTLearn approach is the speed up achieved in computing the optimal cost-to-go between two states in the state space due to the use of learning. This is presented in Table I for different number of nodes considered for selecting the node to expand from. The resulting speedup from learning is evident from Table I. It is important to note that as the number of nodes increase, the distance computation time using the learnt approximation is higher in comparison to the LQR approach. This aspect is discussed in further detail in Section V.

We complement our earlier initial claim related to better

TABLE II

COMPARISON OF RRTLEARN AND LQR-RRT (AVG. NODES).

	iLQR Steering	LQR Steering
Using RRTLearn	44.15 ± 9.52	167.05 ± 53.1
Using LQR-RRT	68.54 ± 16.05	208.7 ± 33.98

TABLE III

COMPARISON OF RRTLEARN AND LQR-RRT (AVG. TIME).

	iLQR Steering	LQR Steering
Using RRTLearn	$21.46\,s$	$12.5\,s$
Using LQR-RRT	$33.64\,s$	$9\,s$

state space coverage with our metric in comparison to existing approaches based on the number of tree nodes to reach a goal state. We believe that this is a reasonably good measure of state space coverage as it implies the existence of good node connectivity in the regions which represent the system dynamics in state space. This is different from the approach used in [5], where a discretization of the state space is used and number of nodes per discretized region is evaluated. We also compare our approach with the LQR-RRT approach. In all cases, we assume a goal region tolerance of $[0.1\,rad, 0.1\,rad/s]'$. This is a region around the goal state, which when reached, indicates a solution has been found. In practice, the actual goal state would be reached by using a linear feedback controller once the goal region is reached.

For evaluating the application of our metric to planning, we study four different cases based on the choice of the distance metric and the steering function. The results are presented in Table II. The first column of Table II indicates the distance metric used, and the first row indicates the choice of steering function. Each numerical entry in the table corresponds to a combination of the respective entry in the first column and first row and its associated 95% confidence interval. The average number of nodes to find an initial solution are computed over 20 runs of each choice. We note that the average number of nodes needed to find a plan to a goal position by RRTLearn using both iLQR and LQR steering is significantly less than the average number of nodes needed by the LQR metric ($p < 0.01$).

The LQR-RRT algorithm needs a higher average number of node evaluations to reach the goal region and this is a direct consequence of the linearization problem. However, it scores much better on the execution time (shown in Table III) in comparison to RRTLearn for two reasons. In the simple case of pendulum, the linearization works well for most situations and hence the LQR metric is more or less reliable consequently allowing for reaching the goal region. Furthermore, the computation time of the LQR cost does not scale up with the number of tree nodes, which is the case with ρ_{learn} computation. However, it remains to be studied as to how this effect varies for systems with more pronounced nonlinear dynamical effects. Before we conclude, we present a brief discussion of our results and

propose possible directions for future work.

V. Discussion and Future Work

From a kinodynamic planning perspective, we believe, RRTLearn with iLQR steering potentially yields better feasible plans compared to the LQR-RRT approach by virtue of the more appropriate linearization technique used. As a consequence, we expect RRTLearn to provide better solutions over the LQR-based approaches for systems with stronger nonlinear dynamics. With our current results, we are limited to this benefit alone and further work is necessary to ascertain the possible benefits for online kinodynamic planning.

An inherent limitation with learning approaches such as the LWPR is the read out time increase with the number of input arguments. In the context of our paper, as the number of nodes in the tree increases, the distance read out time also increases and as a consequence could nullify the speed up achieved. However, by restricting the number of nodes to be evaluated, for example, by using efficient nearest neighbour search [14], this problem can be avoided. Another way to overcome this could be to use supervised learning techniques that are more efficient to read out, such as artificial neural networks. However, we plan to learn the distance metric while building the tree, focusing the learning in the most relevant states, and this requires an incremental method. Furthermore, in an online planning situation such as obstacle avoidance, a potential online kinodynamic solution could be to construct a new tree around the obstacle. In such a scenario, the number of nodes would certainly be a small number as the new tree would be rooted at the current state of the system. From an offline planning perspective though, our approach could potentially yield better plans (in terms of feasibility) because the nonlinear dynamics are better approximated by the iLQR approach in comparison to the standard LQR approach. Another benefit with the learning approach is that, the dynamical information is embedded in the learnt function once during training and the same metric can be used multiple times, for instance, during re-planning.

As per our knowledge, our approach is one of the first in the literature that uses learning to approximate the optimal cost-to-go between states for RRT-based planning. In the future, we propose to evaluate the benefits of our approach on larger state-spaces by studying the feasibility of the generated plans and the amount of post processing needed to execute those plans on real robots. Additionally, constraints such as limits on the control inputs can be accounted for, while learning, by imposing cost penalties for constraint violations. An important aspect in our approach is the fact that the learnt approximations are strongly dependent on the environment used for learning. However, the influence of discontinuities on the learning created by situations such as state limits or new obstacles remains to be studied in further detail.

VI. Conclusions

Appropriate choice of distance metric for RRT-based approaches holds the key to successful planning in state space. Typically, this metric is the optimal cost-to-go between two states in the state space. However, obtaining this cost is computationally intensive for systems with nonlinear dynamics. In our work, we address this issue by decomposing the problem in two levels. We initially estimate the cost-to-go based on the Iterative Linear Quadratic Regulator (iLQR) principle proposed in [9] over a random state space distribution. This is followed by learning a nonlinear function that approximates this cost using a supervised learning technique called Locally Weighted Projection Regression (LWPR) [10]. We experimentally verify the correctness of the learnt function and subsequently use this function to compute distances in state space while planning with RRT. Thus our approach provides the benefit of using a closer approximation of the optimal cost at high speeds. Our experiments have shown a speed up in the distance metric computation of up to a factor of 1000.

ACKNOWLEDGMENT

The authors would like to extend their thanks to Michiel Plooij for his inputs and discussions related to this work. This work is part of the research program STW, which is (partly) funded by the Netherlands Organization for Scientific Research (NWO). The work leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 609206.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] S. M. LaValle and J. J. Kuffner-Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, pp. 378–400, 2001.

[3] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The In*, vol. 21, pp. 233–255, 2002.

[4] S. M. LaValle, *From dynamic programming to RRTs: Algorithmic design of feasible trajectories*. Springer Tracts in Advanced Robotics, 2003.

[5] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *IEEE International Conference on Robotics and Automation*, 2010.

[6] A. Shkolink, M. Walter, and R. Tedrake, "Reachability guided sampling for planning under differential constraints," in *Proceedings of IEEE Conference on Robotics and Automation*, 2009.

[7] A. Perez, R. Platt-Jr, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *IEEE International Conference on Robotics and Automation*, 2012.

[8] N. Rikovitch and I. Sharf, "Kinodynamic motion planning for uavs: A minimum energy approach," *AIAA Guidance, Navigation and Control (GNC) Conference*, 2013.

[9] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *International Conference on Informatics in Control, Automation and Robotics*, 2004.

[10] S. Vijayakumar and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, pp. 2602–2634, 2005.

[11] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *Journal of Machine Learning Research*, vol. 9, pp. 623–626, 2008.

[12] [Online]. Available: http://limb.biomed.queensu.ca/lab_members/timothylillicrap/projects.php

[13] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[14] A. Yershova and S. M. LaValle, "Improving motion planning algorithms by efficient nearest-neighbor searching," *IEEE Transactions on Robotics*, vol. 23, pp. 151–157, 2007.