2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Distance Metric Learning for RRT-Based Motion Planning with Constant-Time Inference

Luigi Palmieri Kai O. Arras

***Abstract*— The distance metric is a key component in RRT-based motion planning that deeply affects coverage of the state space, path quality and planning time. With the goal to speed up planning time, we introduce a learning approach to approximate the distance metric for RRT-based planners. By exploiting a novel steer function which solves the two-point boundary value problem for wheeled mobile robots, we train a simple nonlinear parametric model with constant-time inference that is shown to predict distances accurately in terms of regression and ranking performance. In an extensive analysis we compare our approach to an Euclidean distance baseline, consider four alternative regression models and study the impact of domain-specific feature expansion. The learning approach is shown to be faster in planning time by several factors at negligible loss of path quality.**

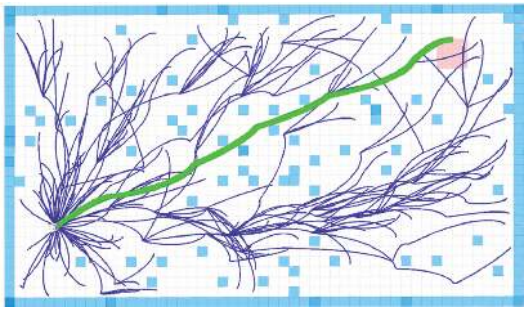

Fig. 1. An example tree and path generated with our learned distance metric. The robot starts at the bottom-left and plans a path to the goal region marked in red. The first-solution path is shown in green.

I. Introduction

Sampling-based methods have become a popular approach to motion planning particularly in high dimensions or under complex constraints. Rapidly exploring Random Trees (RRT) solve a single planning query by growing and expanding a tree in the configuration space towards newly sampled configurations. An optimal RRT variant, named RRT* by Karaman and Frazzoli [1], rewires the tree based on the notion of cost: under the assumptions given in [2] for holonomic systems and in [3] for nonholonomic systems the solution converges to the optimum as the number of samples approaches infinity.

A key component in the extension of the tree in RRT is the distance pseudo-metric, or cost-to-go pseudo-metric, used to select the nearest vertex from where to grow the tree. In RRT* this function has an even more important role as it guides the rewiring procedure. To do so, the pseudo-metric has to be computed as many times as there are vertices in the near-neighbour ball [2] or near-neighbour box [3].

For general kinodynamic systems, it is hard to determine the true cost-to-go function. It implies solving both a two-point boundary value problem (2P-BVP), which may be as expensive as solving a motion planning query on its own, and an optimal control problem. This is why, in the original RRT paper, LaValle and Kuffner [4] suggested to use approximations of the optimal cost-to-go as functions of path length, difference between initial and final orientation, and translational and rotational velocities. Such a near-optimal distance metric was shown in [5] to enable a motion planner to entirely cover the configuration space and to solve hard planning problems.

A number of different pseudo-metrics have been used in previous work [6, 7, 8, 9, 10, 11]. Variants of the Euclidean distance have been studied in [6, 7]. Amato *et al.* [6] compare several distance metrics defined in the configuration space and show that for nonholonomic systems, the weighted Euclidean distance – commonly used for holonomic systems – is unable to correctly cover the space. They give recommendations on how to select a metric based on efficiency and effectiveness. Kuffner [7] defines a proper distance metric for the configuration space of a 3D rigid body, the Lie group $\mathbb{SE}(3)$. The author proposes a weighted metric with a translation component that uses a standard Euclidean norm and a scalar function that returns an approximate measure of the distance between the initial and final orientation. Distance metrics have been derived by linearizing the system dynamics in [8, 9, 10]. Glassman and Tedrake [8] describe how the Voronoi bias of RRT only applies when a proper metric is defined for the case of an extend function that forward simulates a dynamical system. They use an affine quadratic regulator design and show that it can be used to approximate the exact minimum-time distance pseudo-metric at a reasonable computational cost. Perez *et al.* [9] use an optimal infinite-horizon LQR controller to connect pairs of states. The method linearizes the domain dynamics locally. In this case the cost-to-go pseudo-metric is defined as the solution of the Riccati's equation used in the LQR extender. Webb and van den Berg [10] use a finite-horizon optimal controller as local planner. They can optimize a certain class of cost functions that trades off time and control effort.

L. Palmieri, K.O. Arras are with the Social Robotics Lab, Dept. of Computer Science, University of Freiburg, Germany. {palmieri,arras}@cs.uni-freiburg.de.

A drawback of these linearization methods is that the approximations are valid only as long as the linearization is valid. When non-linearities increase, the accuracy of the metric degrades. They may also suffer from high computational costs and numerical issues.

Li and Bekris [11] approximate the optimal cost-to-go pseudo-metric by an offline learning method: the distance between two states is approximated by the cost of an A* path between their closest sampled states on a learned graph. The graph is generated off-line by using forward propagation of the system dynamics. For speeding up the method, they map the offline samples into a higher-dimensional Euclidean space. The method makes approximations on two levels: the graph is built using a discretized set of controls, not solving the 2P-BVP, and the mapping of the samples which compromises the method's ability to cover the state space.

A recent idea, which has been developed independently by three research groups at the same time [12, 13, 14], is to approximate the distance pseudo-metric by supervised learning of a nonlinear regression model. Bharatheesha *et al.* [13] approximate the optimal cost-to-go using locally weighted projection regression (LWPR). The optimal cost-to-go is obtained by iteratively solving a linear quadratic regulator problem where nonlinear dynamics are still linearized around a nominal trajectory instead of a point. The learning approach is incremental which makes that cost prediction scales with the increasing number of nodes in the RRT tree. Allen *et al.* [14] use locally weighted linear regression (LWR) to predict optimal cost-limited reachable sets of dynamical systems in real-time and a binary SVM classifier to learn the nonlinear boundary between a state's reachable and non-reachable set. While achieving good regression accuracy, inference times scale quadratically with the number of LWR training samples and linearly with the number of SVM support vectors, respectively.

In this paper, we extend our approach presented in [12]. We approximate the cost-to-go metric by a simple, offline-learned regression model with constant-time inference. We consider differential drive robots in the configuration space $\mathbb{R}^2 \times \mathbb{S}^1$, although the same approach can be extended to systems with higher dimensions. Our distance metric estimates the cost of local paths from a novel extender called POSQ which solves the 2P-BVP and produces smooth cusp-free trajectories [15]. POSQ is able to connect any pair of 2D poses and produces RRT trees that cover the entire state space. The latter is not true for forward propagation approaches of discretized controls (motion primitives) as discussed in [8]. Furthermore, the POSQ extender makes no linearization or approximation, is efficient to compute and was shown to produce smoother paths in shorter time with smaller trees than motion primitives and a spline-based extender approach [15]. In this paper, with the goal of making the planner even more efficient, we make the following contributions:

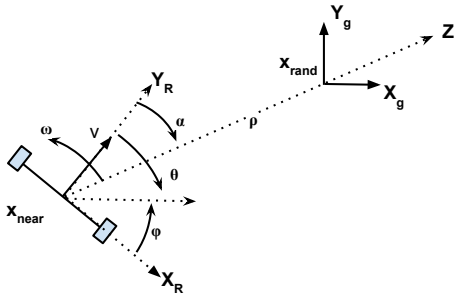

Fig. 2. Robot and goal pose relations and notation

- We show how the distance pseudo-metric for the case of the POSQ extender can be learned offline using a set of domain-specific features and a simple basis function model with constant-time inference. In addition of being very fast to compute, the learned model is accurate in terms of regression and ranking performance at negligible loss of path quality.
- In addition to [12], we present a comprehensive comparison to an Euclidean distance baseline and four alternative regression models namely neural network regression, LWPR, SVM regression, and random forest regression and analyze the impact of domain-specific feature expansion.
- The comparison demonstrates that the Euclidean distance – although fast to compute – is highly inaccurate in terms of ranking and regression and leads to paths of poor quality and length. The experiments also show that our learning approach is able to cover the state space in the same way than the ground truth function.

The paper is structured as follows: in Section II we briefly present the POSQ extender and in Section III we describe how the distance metric is learned. Experiments and their results are described in Section IV. Section V concludes the paper.

II. The POSQ Extender

We briefly summarize the POSQ extend function as introduced in [15]. The function solves the two-point boundary value problem (2P-BVP) for differential drive robots and generates smooth trajectories between any pair of 2D poses. Within RRT, it connects randomly sampled poses, $\mathbf{x}_{rand}$, with their nearest pose in the tree, $\mathbf{x}_{near}$. The method is an extension of the approach by Astolfi [16] and allows to describe – thanks to a coordinate transform from Cartesian to polar – the kinematic model of a wheeled mobile robot by the open loop model

$$\begin{aligned} \dot{\rho} &= -\cos\alpha \; v \\ \dot{\alpha} &= \frac{\sin\alpha}{\rho} v - \omega \\ \dot{\phi} &= -\omega. \end{aligned} \tag{1}$$

ρ is the Euclidean distance between $\mathbf{x}_{near}$ and the goal pose $\mathbf{x}_{rand}$, ϕ denotes the angle between the x-axis of the

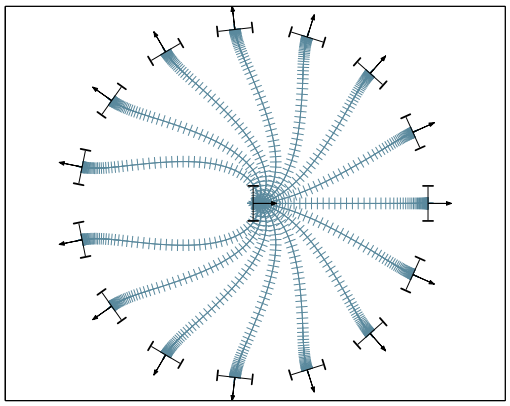

Fig. 3. Trajectories of the POSQ controller when steering the robot from the center to the poses on the circle

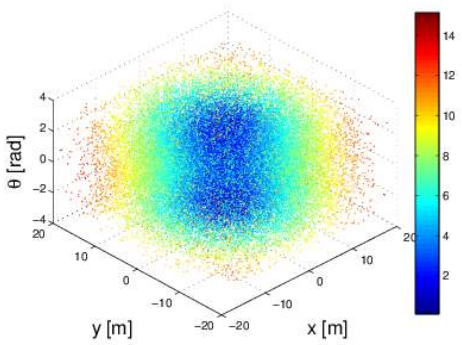

Fig. 4. The cost-to-go function $C(\mathbf{0}, \mathbf{x})$ for paths generated by the POSQ controller.

robot reference frame (X_R) and the one associated to the desired position (X_g), and α is the angle between the y-axis of the robot reference frame (Y_R) and the vector connecting the robot with the desired position (Z), see Fig. 2.

Considering the open loop model obtained by the transform in Eq. (1), in [15], we propose the non-linear feedback law

$$\begin{aligned} v &= K_\rho \tanh(K_v \rho) \\ \omega &= K_\alpha \alpha + K_\phi \phi \,. \end{aligned} \tag{2}$$

which produces paths of quasi-constant forward velocity as opposed to the original law that causes the velocity to strongly drop towards the goal. We could prove that the new law assures asymptotically heading convergence and system's local stability. Eq. (2) generates smooth trajectories $\mathbf{x}(t)$ and controls $\mathbf{u}(t)$, $t \in [0, T]$, $T > 0$, that connect any given pair of 2D poses by computing closed-loop forward simulations based on the kinematic model of a non-holonomic wheeled mobile robot (see Fig. 3). Thanks to its ability to solve the 2P-BVP, trajectories generated by POSQ can be readily characterized by a cost which we seek to learn hereafter.

III. Our Approach

In RRT-based planning, a tree is grown by connecting randomly sampled configurations $\mathbf{x}_{rand}$ to their nearest vertex $\mathbf{x}_{near}$ in the tree. For the selection of $\mathbf{x}_{near}$, the algorithm evaluates the distances from all or a subset of tree vertices to $\mathbf{x}_{rand}$. The evaluation of this distance metric is a frequent operation deep within every RRT algorithm and a speed up at this point would have a strong impact onto planning times.

The idea is, instead of computing an extension path and then evaluating its cost, to learn a parametric regression model that directly predicts the cost. This is fast to compute and we can expect a speed up even though the forward simulation by POSQ is already efficient to implement. Formally, we have a regression model

$$y \approx g(\mathcal{X}, \beta) \tag{3}$$

with $\mathcal{X}$ being the set of independent variables (features or attributes) and β the parameters of g. In this section, we first define the class of distance metrics considered here, design a set of features, choose a regression model and a learning algorithm to fit its parameters.

A. The distance metric

Following [4], we consider a class of distance metrics $C(\mathbf{x}_1, \mathbf{x}_2)$ defined as a linear combination of path length and sum of heading changes between states $\mathbf{x}_1$ and $\mathbf{x}_2$,

$$C(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=0}^{N_e - 1} w_d ||\mathbf{P}_{i+1} - \mathbf{P}_i|| + w_q \left(1 - |\mathbf{q}_{i+1} \cdot \mathbf{q}_i|\right)^2 . \tag{4}$$

$\mathbf{P}_i$ are the $N_e + 1$ intermediate points of the path and $\mathbf{q}_i$ the associated quaternions. The time needed to compute this cost expression depends on the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ and the integration time step Δt (leading to more or fewer intermediate points $\mathbf{P}_i$). Fig. 4 shows the cost distribution for paths generated by the POSQ controller.

B. Features

Let the set of independent variables $\mathcal{X}$ be the vector of features $\mathbf{f}$ that we define in this section. Naively, we could directly use the inputs of the extend function, the two poses $\mathbf{x}_1$ and $\mathbf{x}_2$, as features $\mathbf{f} = (x_1, x_2, y_1, y_2, \theta_1, \theta_2)$ since they fully define the problem.

However, this choice encodes the relevant information only very implicitly. We expect interesting interactions between those features which we seek to make explicit by performing feature expansion to obtain more meaningful inputs. But instead of an uninformed, generic method such as quadratic expansion or kernel methods, we can take advantage of our domain knowledge to capture those interactions. For example, it is obvious that the Euclidean distance $\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$ will be a dominant feature for predicting the cost of paths that connect $\mathbf{x}_1$ and $\mathbf{x}_2$. Finally, in multiple validation runs, we have found a set of fourteen features to characterize the cost-to-go function, see Table I. The features make the geometry of POSQ paths under the cost model Eq. 4

Description	Expression
Displacement in x	$\Delta x = x_2 - x_1$
Displacement in y	$\Delta y = y_2 - y_1$
Displacement in θ	$\Delta\theta = \theta_2 - \theta_1$
Euclidean distance between poses	$d = \|\mathbf{x}_2 - \mathbf{x}_1\|$
x-projection of the orientation change	$\cos\Delta\theta$
y-projection of the orientation change	$\sin\Delta\theta$
Orientation change multiplied by Euclidean distance	$d\,\Delta\theta$
x-projection of the orientation change multiplied by Euclidean distance	$d\cos\Delta\theta$
y-projection of the orientation change multiplied by Euclidean distance	$d\sin\Delta\theta$
Angular difference between $\mathbf{x}_1$ and connecting line of the two poses	$\operatorname{atan}\frac{\Delta y}{\Delta x} - \theta_1$
Angular difference between $\mathbf{x}_2$ and connecting line of the two poses	$\operatorname{atan}\frac{\Delta y}{\Delta x} - \theta_2$
Ratio between the previous two features	$\frac{\operatorname{atan}(\Delta y/\Delta x)-\theta_1}{\operatorname{atan}(\Delta y/\Delta x)-\theta_2}$
Angular difference between $\mathbf{x}_1$ and connecting line multiplied by Euclidean dist.	$d\,(\operatorname{atan}\frac{\Delta y}{\Delta x} - \theta_1)$
Angular difference between $\mathbf{x}_2$ and connecting line multiplied by Euclidean dist.	$d\,(\operatorname{atan}\frac{\Delta y}{\Delta x} - \theta_2)$

TABLE I
INPUT FEATURES

more explicit and, as will be shown in the experiments, facilitate the learning process.

C. Learning

We choose a basis function model (BFM) for learning to predict path costs, fitted to the training set $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^N$ using Levenberg-Marquardt. The model is defined as

$$y = \sum_{m=1}^{M} \Phi_m(\mathbf{f}, \beta) \tag{5}$$

where M is the number of basis functions Φ. Given our initial goal of speeding up planning time, this choice appears promising for its simplicity and constant-time inference, independent on the number of training samples. Concretely, we choose quadratic basis functions given by

$$y = \sum_{m=1}^{M_f} \beta_{m_1}(f_m - \beta_{m_2})^2 \tag{6}$$

where M_f is the number of features.

Training samples $\mathbf{s}_i = [\mathbf{f}_i,\ c_i]$ are pairs of feature vectors and ground truth costs. Here, we randomly generate pose pairs $(\mathbf{x}_1, \mathbf{x}_2)$ in the configuration space, compute their 14-dimensional feature vector $\mathbf{f}_i$ and determine the corresponding ground truth cost from the POSQ extend function, $c_i = C(\mathbf{x}_1, \mathbf{x}_2)$.

IV. Experiments and Results

In the experiments we compare our approach to an Euclidean distance baseline, consider four alternative regression models and study the impact of the domain-specific feature expansion described above. We perform two sets of experiments, first we evaluate the prediction accuracy of the different methods in terms of regression and ranking metrics, and second, we analyze how the learned distance metrics impacts planning time, path quality, and state space coverage. Concretely, we consider

- The proposed basis function model with the fourteen domain-specific features (BFM) in Table I.
- The basis function model with naive features $\mathbf{f} = (x_1, x_2, y_1, y_2, \theta_1, \theta_2)$ (BFM naive).
- A neural network model (NN) with two hidden layers, 30 neurons in the first and 20 in the second one. The architecture has been found through 5-fold cross validation
- A random forest regression model (Rnd Forest) [17] with 100 trees each with 5 terminal leaves. Both hyperparameters have been found through cross validation and provide a good trade off between prediction time and accuracy.
- A ν-SVR model [18] with a RBF kernel $Q_{ij} = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$ with parameters $C = 1e^{-8}$ and $\gamma = 1e^{-3}$ which have been found via cross validation.
- A locally weighted projection regression (LWPR) model with Gaussian kernels and eight receptive fields. Initial parameters and hyperparameter have again been found through cross validation.
- We also consider the Euclidean distance (Eucl. Dist.), the most commonly used distance metric, as an approximation of the true cost.

A. Regression and Ranking Performance

To evaluate the prediction accuracy of the learned regression models, we use the following metrics: median of the residuals, mean squared error normalized by the residuals' variance (NMSE), and the coefficient of determination. We also determine the average runtime t_{pred} of a single prediction.

Note that although we framed our task as a regression problem it is actually a *learning-to-rank problem*. When searching the tree for the nearest state $\mathbf{x}_{near}$ given $\mathbf{x}_{rand}$, we are actually interested in the correct ranking of the tree vertices under the cost model rather than the predicted costs as such. The typical strategy is then to choose the best ranked (lowest cost) vertex as $\mathbf{x}_{near}$. Therefore, we also evaluate the model with respect to its ability to correctly rank a set of states and use the following ranking metrics: Kendall τ coefficient, Kendall τ_d distance and Spearman ρ coefficient [19]. Kendall τ and Spearman ρ coefficients are both correlation measures between two-ordinal level variables equal to 1 if the two rankings agree perfectly and equal to -1 if they disagree perfectly. Kendall τ_d distance measures the number of disagreements between two ordered lists equal to 0 if the two ranks are equal. Here, we consider the ranking of the five best vertices.

We learn all models with the same 50,000 training samples and validate with 10,000 samples in terms of regression performance. For ranking, we predict the cost and evaluate the metrics for a grid of poses over the entire

Regression Performance							
Metric	**BFM**	**BFM naive**	**NN**	**Rnd Forest**	**ν-SVR**	**LWPR**	**Eucl. Dist.**
Median residuals	0.030	7.8376	**1.607 e^{-6}**	0.0054987	0.000261	0.6055	0.167
NMSE	0.005	0.8843	**7.729 e^{-7}**	0.0011037	0.028991	0.0806	0.0118
Determination	0.999	0.8040	**1**	0.99976	0.99349	0.9821	0.9895
Runtime Performance							
Metric	**BFM**	**BFM naive**	**NN**	**Rnd Forest**	**ν-SVR**	**LWPR**	**Eucl. Dist.**
t_{pred} [sec]	6.22 e^{-07}	4.09 e^{-07}	1.24 e^{-05}	6.81 e^{-05}	0.0012	1.87 e^{-05}	**3.853 e^{-07}**
Ranking Performance							
Metric	**BFM**	**BFM naive**	**NN**	**Rnd Forest**	**ν-SVR**	**LWPR**	**Eucl. Dist.**
τ	**1**	-0.2	**1**	**1**	**1**	0.40	-0.40
τ_d	**0**	0.6	**0**	**0**	**0**	0.30	0.70
ρ	**1**	-0.3	**1**	**1**	**1**	0.40	-0.50

TABLE II

Regression and ranking performance

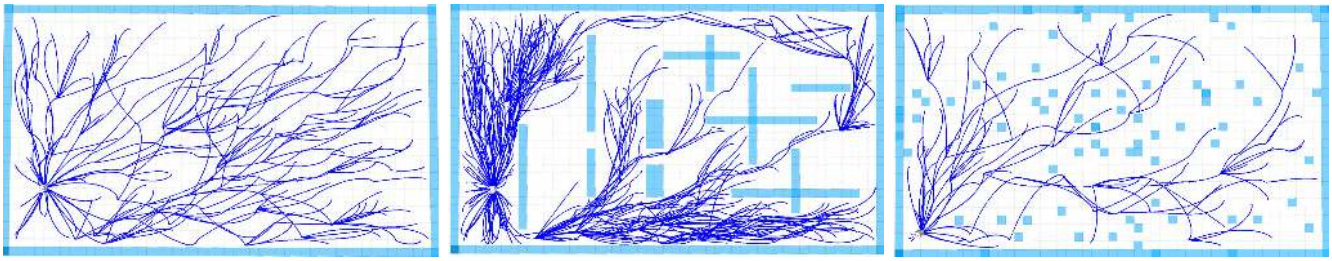

Fig. 5. The three environments and example trees obtained with our approach. **Left:** *open space* scenario. **Center:** *hallway* scenario. **Right:** *random map* scenario

configuration space without obstacles with a resolution of $0.1m$ in x, y and $\pi/4$ rad in θ.

B. Regression and Ranking Results

The results for the regression and ranking metrics are reported in Table II. The poor performance of the naive feature approach clearly demonstrates the necessity to design informative features for this learning task. Also the Euclidean distance fails to approximate the true distance metric accurately in terms of both regression and ranking error. Furthermore, the results show the relation between a model's ranking and regression performance: less accurate regression does not prevent perfect ranking (of the best five states). Among the four methods with perfect ranking results, the BFM approach is a clear winner with two orders of magnitude better runtime performance. The worst model in this sense is ν-SVR where inference time scales with the number of support vectors which in our experiments exceeds 16,000.

C. Planning Performance

We now investigate how the learned regression model impacts planning time as well as path quality and – further below – coverage ability. To this end, we compare the learned distance metric with two ground truth baselines, the second best model in the previous experiment (NN), and, due to its common usage, the Euclidean distance. The baselines are the ground truth cost of the POSQ extender with the regular high-resolution integration time of $\Delta t = 0.1$ *sec* (POSQ 0.1) and a lower integration time of $\Delta t = 0.5$ *sec* (POSQ 0.5). The latter is to analyze the performance of a faster but "rougher" version of POSQ with fewer path points. For the sake of a fair comparison the Euclidean distance uses a *k-d* tree data structure to speed up nearest neighbor search.

We consider three simulated test environments (Fig. 5) that stress different properties of a planner. The *open space* scenario has no obstacles, it serves to study the planner's behaviour when the tree can grow freely. The *hallway* scenario contains many areas of open space, alternative paths to the goal and local minima. The *random map* scenario contains 100 randomly placed square obstacles. There are many homotopy classes, some require more or less maneuvers along paths than others. The map size in all scenarios is $50m \times 30m$.

To quantify planning performance we compute the averages of the following metrics: time for a single extension ($\mathbf{t}_{ext}$), time to find a solution ($\mathbf{T}_{path}$), and path length in meters ($\mathbf{l}_{path}$). Smoothness, although being an intuitive concept, is less straightforward to assess. We compute three measures relevant in our context: let v_{max} be the maximum magnitude of the robot velocity vector $\mathbf{v}$, $\tilde{\mathbf{v}} = \frac{\mathbf{v}(t)}{v_{max}}$ the normalized velocity, and $[t_1, t_2]$ the time interval over which the movement is performed.

1) η_{nmaJ}, the average of the mean absolute jerk normalized by v_{max}, for which the best value is zero,
$\eta_{nmaJ} = -\frac{1}{v_{max}(t_2-t_1)} \int_{t_1}^{t_2} \left| \frac{d^2\mathbf{v}}{dt^2} \right| dt,$

2) η_{spal}, average of the speed arc length, for which the best value is zero,

Open space scenario						
Method	$\mathbf{t}_{ext}$ $[s]$	$\mathbf{T}_{path}$ $[s]$	$\mathbf{l}_{path}$ $[m]$	$\eta_{\mathbf{nmaJ}}$	$\eta_{\mathbf{spal}}$	$\eta_{\mathbf{pm}}$
POSQ 0.1	0.01019	14.5215	47.6487	$-7.32075\ e^{-05}$	-0.802622	0.62
POSQ 0.5	0.006093	5.3704	50.4071	$-6.77003\ e^{-05}$	-0.73929	0.38
BFM	0.001325	1.0902	48.608	$-6.44915\ e^{-05}$	-0.759564	0.31
NN	0.003556	2.6722	48.2047	$-6.71913\ e^{-05}$	-0.777919	0.55
Eucl. Dist.	0.001353	0.1417	56.0124	-0.0034127	-2.28714	0.36
Hallway scenario						
Method	$\mathbf{t}_{ext}$ $[s]$	$\mathbf{T}_{path}$ $[s]$	$\mathbf{l}_{path}$ $[m]$	$\eta_{\mathbf{nmaJ}}$	$\eta_{\mathbf{spal}}$	$\eta_{\mathbf{pm}}$
POSQ 0.1	0.01082	92.9419	54.6069	$-7.44230\ e^{-05}$	-0.844397	0.90
POSQ 0.5	0.007975	70.6947	56.0821	$-8.01868\ e^{-05}$	-0.8833	3.23
BFM	0.00179	26.2349	61.5938	$-4.28115\ e^{-05}$	-0.9543	0.42
NN	0.005318	65.2463	63.3869	$-7.22698\ e^{-05}$	-1.01336	0.39
Eucl. Dist.	0.0007944	1.4498	83.0162	-0.00705978	-3.36954	0.52
Random map scenario						
Method	$\mathbf{t}_{ext}$ $[s]$	$\mathbf{T}_{path}$ $[s]$	$\mathbf{l}_{path}$ $[m]$	$\eta_{\mathbf{nmaJ}}$	$\eta_{\mathbf{spal}}$	$\eta_{\mathbf{pm}}$
POSQ 0.1	0.01180	37.7666	49.8373	$-6.04633\ e^{-05}$	-0.794841	0.45
POSQ 0.5	0.01089	20.88	50.129	$-6.12429\ e^{-05}$	-0.795457	0.68
BFM	0.003976	23.3264	53.2168	$-6.85253\ e^{-05}$	-0.844749	3.89
NN	0.003783	19.452	53.5202	$-6.98598\ e^{-05}$	-0.865452	0.33
Eucl. Dist.	0.001503	0.6197	61.7909	-0.00487451	-2.76645	2.63

TABLE III

SMOOTHNESS AND EFFICIENCY RESULTS

$$\eta_{spal} = -ln\left(\int_{t_1}^{t_2} \sqrt{\left(\frac{1}{t_2 - t_1}\right)^2 + \left(\frac{d\tilde{\mathbf{v}}}{dt}\right)^2} dt\right),$$

3) $\eta_{pm} = |\mathcal{V}_{peaks}|$, average number of peaks with $\mathcal{V}_{peaks} = \{\mathbf{v}(t) : \frac{d\mathbf{v}}{dt} = 0, \frac{d^2\mathbf{v}}{dt^2} < 0\}$ being the set of local velocity maxima.

For each environment and method we perform 100 runs and compute the average of all metrics. We use uniform sampling in the entire state space. All experiments were carried out in a C++ implementation on a single core of an regular laptop with 2.70 GHz Intel i5 and 12 GB RAM.

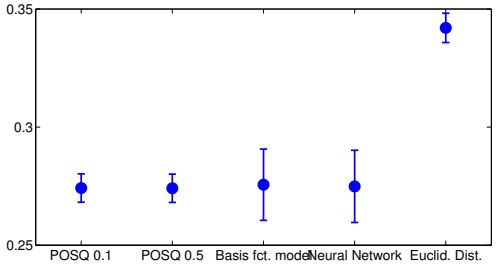

Fig. 7. State space coverage means and standard deviations after 5,000 iterations.

D. Planning Performance Results

The results, given in Table III, show the expected speed advantage of the Euclidean distance over all other methods but also that this cost metric produces by far the longest and least smooth paths. Among the learned distance metrics, the BFM model generally improves both runtime metrics by several factors at a negligible loss of path smoothness. This remains true even for the "rough" version of the POSQ extender, POSQ 0.5.

The speed up in planning time of the learned distance metric is most dramatic in the *open space* scenario. This is because without obstacles, the other RRT heuristics that influence tree growth (extend function, collision checking and random state generation) have no effect and the improvement is fully visible.

In the *hallway* scenario, the BFM approach is still able to find a solution more than twice as fast while in the cluttered *random map* environment, the learning method is on par with the POSQ extender. The reason is that in cluttered environments a lot of time is spent for collisions checking and short extensions for which the acceleration by the learning approach is less visible. Path smoothness remains largely unaffected, the values are all within the same order of magnitude than the original approach.

E. State Space Coverage

Different distance metrics may lead to different state space coverage behaviors [8]. We thus compare the ability of our learning approach to cover the state space with the previous four methods in the *random map* scenario. To this end, we divide the entire state space into a grid of 3D cells and determine state space coverage as the ratio of grid cells covered by the tree. We perform 100 runs of 5,000 iterations for each method.

The results are shown in the Fig. 7. The Euclidean distance is best at covering the state space but it does so because it fails to approximate the true cost and picks incorrect (quasi random) nearest vertices. The resulting trees lead to poor solutions in terms of path length and quality (see Fig. 6 for example trees). Thanks to its good approximation abilities, the learned distance metric does not degrade in terms of state space coverage. On average, its trees are able to cover the same amount of grid cells than the baseline method.

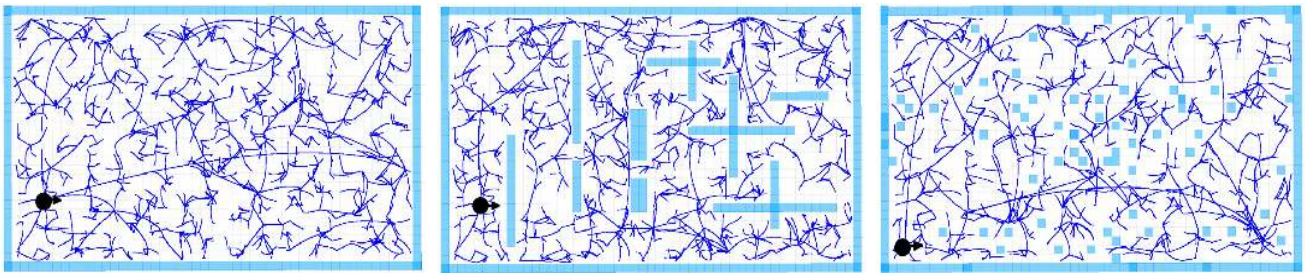

Fig. 6. Example trees obtained using the Euclidean distance in the three scenarios. The initial pose of the robot is shown by a black dot. Although state space coverage is good, the planner generates trees that lead to poor solutions in terms of path length and quality.

V. Conclusions

In this paper, we have presented a new learning approach to approximate the distance pseudo-metric (or cost-to-go metric) for RRT-based motion planning for wheeled mobile robots. Instead of computing local extension paths and then evaluating their cost when growing the tree, we learn a parametric regression model that directly predicts the cost. In a comparison with four alternative regression models we could show that a simple basis function model with constant-time inference is the best choice in terms of regression and ranking accuracy as well as planning time and path quality. The resulting speed up in planning time is significant particularly in less cluttered environments. Using a set of domain-specific features, we have also demonstrated the need to design informative features for this learning task.

Despite good results for planning time and state space coverage, our experiments have shown the Euclidean distance to be a poor choice for the approximation of the true cost with respect to regression and ranking accuracy, and consequently, path quality and length.

In future work, we will aim for higher dimensional configuration spaces. We believe that for complex systems such as humanoids and mobile manipulators our approach will lead to an even more dramatic improvement in planning time. Clearly, new features and a 2P-BVP solver will be required. We also plan to incorporate the learned model into RRT variants like RRT* or T-RRT.

Acknowledgement

The authors thank Frank Hutter for valuable discussions. This work has been supported by the EC under contract number FP7-ICT-600877 (SPENCER)

References

[1] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. of Robotics: Science and Systems (RSS)*, 2010.

[2] ——, "Sampling-based algorithms for optimal motion planning," in *Int. Journal of Robotics Research (IJRR)*, vol. 30, no. 7, 2011, pp. 846–894.

[3] ——, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[4] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," in *Int. Conf. on Robotics and Automation (ICRA)*, Detroit, USA, 1999.

[5] P. Cheng and S. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2001.

[6] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," *IEEE Trans. on Robotics and Automation (TRO)*, vol. 16, no. 4, pp. 442–447, Aug 2000.

[7] J. J. Kuffner, "Effective sampling and distance metrics for 3d rigid body path planning," in *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, USA, 2004.

[8] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, USA, 2010.

[9] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, USA, 2012.

[10] D. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[11] Y. Li and K. Bekris, "Learning approximate cost-to-go metrics to improve sampling-based motion planning," in *Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[12] L. Palmieri and K. O. Arras, "Distance metric learning for RRT-based motion planning for wheeled mobile robots," in *IROS 2014 Workshop on Machine Learning in Planning and Control of Robot Motion*, Chicago, USA, 2014.

[13] M. Bharatheesha, W. Caarls, W. Wolfslag, and M. Wisse, "Distance metric approximation for state-space RRTs using supervised learning," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[14] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[15] L. Palmieri and K. O. Arras, "A novel RRT extend function for efficient and smooth mobile robot motion planning," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[16] A. Astolfi, "Exponential stabilization of a wheeled mobile robot via discontinuous control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 121, no. 1, 1999.

[17] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[19] C. Spearman, "The proof and measurement of association between two things," *Int. Journal of Epidemiology*, vol. 39, no. 5, pp. 1137–1150, 2010.