Distance Metric Learning for RRT-Based Motion Planning for Wheeled Mobile Robots

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 compute the distance metric for RRT-based planners. By exploiting a novel extend function which solves the two-point boundary value problem for wheeled mobile robots, we train a nonlinear parametric model that achieves good performances in regression and ranking tests. We compare our approach to three baselines and show that the new approach is faster by several factors with negligible loss of path quality. We also show that the approach covers the state space in the same way than the true distance metric.**

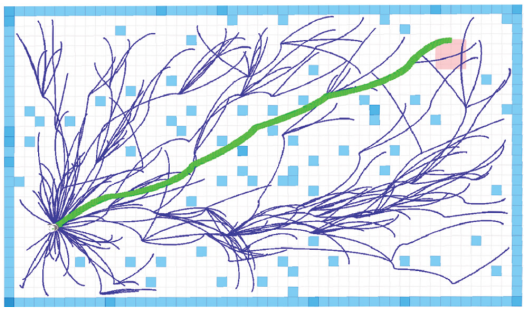

Fig. 1. An example tree and path generated with our learned distance metric. The robot starts at the bottom-left and plans a path to the goal region marked in red. The first-solution path is shown in green.

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 of the tree. To do so, the pseudo-metric has to be computed many times, as many as there are vertices in the near-neighbour ball [2] or near-neighbour box [3].

For kinodynamic systems having a distance metric close to the optimal path cost makes the motion planner be able to cover entirely the configuration space so allowing to solve hard problems in an easy way [4]. It is well known that to compute the optimal cost-to-go function one needs to solve a two-points boundary problem, which is as expensive as to solve a motion planning query.

Previous definitions of the pseudo-metrics in related work include variants of the Euclidean distance [5], a cost derived from the definition of a finite-horizon affine quadratic regulator [6], pseudo-metrics which penalize both the duration of the trajectory and the expended control effort [7], pseudo-metric based on locally linearizing the domain dynamics and applying linear quadratic regulation [8], and an approximated pseudo-metric based on an offline learned graph built by using forward simulation of the kinodynamic system [9].

In the original paper of RRT, LaValle and Kuffner [10] suggest to use an approximation of the optimal cost-to-go for a generic kinodynamic system that takes into account the path length, the difference between initial and final orientation, and the translational and rotational velocities of the considered system.

For holonomic systems a commonly used metric is the weighted Euclidean distance. Amato *et al.* [5] present a comparative evaluation of distance metrics defined in the configuration space. They show that for nonholonomic systems, weighted Euclidean distance metrics are unable to correctly cover the configuration space. Given a local planner they include recommendations on how to select the metrics based on efficiency and effectiveness.

Glassman and Tedrake [6] describe how the Voronoi bias only applies when a proper metric is defined for the case of an extend function that performs a forward simulation of the dynamic 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. One drawback of this method is linearization. The approximated pseudo-metric is valid only in a range where the linearization is valid. With increasing non-linearity, the accuracy of the pseudo-metric will further degrade.

Recently, Perez *et al.* [8] use an optimal infinite-horizon LQR controller to connect pairs of states. The method linearizes the domain dynamics locally which is interesting from an efficiency point of view. In this case the cost-to-go pseudo-metric is defined as the solution of

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.

the Riccati equation used in the LQR extender.

Webb and van den Berg [7] 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. Both methods are based on linearizing the dynamics of the system, which causes the accuracy of the cost-to-go metric to degrade as non-linearities increase. Further, these metrics may suffer from high computational costs and numerical issues that can make them unsuitable for motion planning in real-time.

Li and Bekris [9] approximate the optimal cost-to-go pseudo-metric by an offline learning method: the distance between two states is approximated by the A* path cost between their closest sampled states on a learned graph. The graph is generated off-line by using forward propagation of the system dynamics. To improve the poor computational performances, 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 two-point boundary value problem – and in the additional mapping, which cause loses in coverage of the state space.

Here, we learn the cost-to-go pseudo-metric in an offline fashion and approximate it by a parametric non-linear model. We consider the case of 2D motion planning for differential drive robots although the same approach can be easily extended to different systems with higher dimensions. Our cost-to-go function uses local paths from a novel extender called POSQ [11] which solves the two-point boundary value problem for wheeled mobile robots and optimizes the criteria path length and heading changes. The extend function is able to connect any pair of poses which allows the tree to cover the entire state space over time. This property is not true for forward propagation approaches of discretized sets of controls (a.k.a. motion primitives), as discussed by Glassman and Tedrake [6]. Furthermore, the POSQ extender makes no linearization or approximation, is very efficient to implement and was shown to produce smoother paths than motion primitives and a spline-based extender approach. With the goal of making the planner even more efficient, we make the following contributions:

- 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 nonlinear parametric model, achieving good performances in terms of regression and ranking errors.
- The learned model is very fast to compute with negligible loss of path quality. It is faster in extending the tree and finding a path also compared to alternative baseline methods.
- Finally, we demonstrate 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 summarize the POSQ extender and in Section III we describe how the distance metric is learned. In Section IV we give details on the experiments and their results to evaluate the approach. Section V concludes the paper.

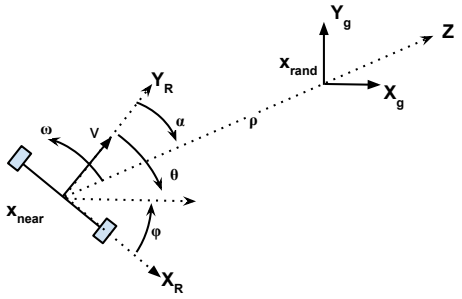

Fig. 2. Robot and goal pose relations and notation

II. The POSQ Extender

We briefly summarize the POSQ extend function as introduced in [11]. The function solves the two-point boundary value problem for differential drive robots and generates smooth trajectories between any two poses. Adopting typical RRT notation, it connects, for example, a randomly sampled pose, $\mathbf{x}_{rand}$, and its nearest pose in the tree, $\mathbf{x}_{near}$.

The method, which is an extension of the approach by Astolfi [12], exploits a coordinate transformation from Cartesian to polar which allows to describe the kinematic model of a wheeled mobile robot by the open loop model

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

where ρ is the Euclidean distance between the $\mathbf{x}_{near}$ and the $\mathbf{x}_{rand}$ goal pose, ϕ denotes the angle between the x-axis of the 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 coordinate transform in Eq. (1), we define the nonlinear feedback law

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

The law in Eq. (2) produces paths of quasi-constant forward velocity as opposed to the original law that causes the velocity to strongly drop towards the goal. This 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). The tree is grown in the configuration space $\mathbb{R}^2 \times \mathbb{S}^1$ where each configuration $\mathbf{x} = (x, y, \theta)$ consists of the (x, y)-position of the robot and its orientation θ. Thanks to its ability to solve the two-point boundary value problem in this case, the function can be readily used as the distance metric which we seek to learn hereafter.

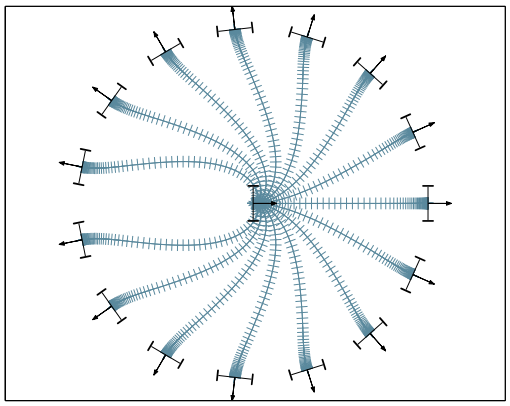

Fig. 3. Trajectories of the POSQ controller when steering the robot from the center to the poses on the circle

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 (or cost-to-go function) 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 use Levenberg-Marquardt to fit its parameters.

A. The distance metric to be learned

Following [10], we consider a class of distance metrics $C(\mathbf{x}_1, \mathbf{x}_2)$ defined to be a linear combination of path length and sum of heading changes between two states $\mathbf{x}_1$ and $\mathbf{x}_2$,

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

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

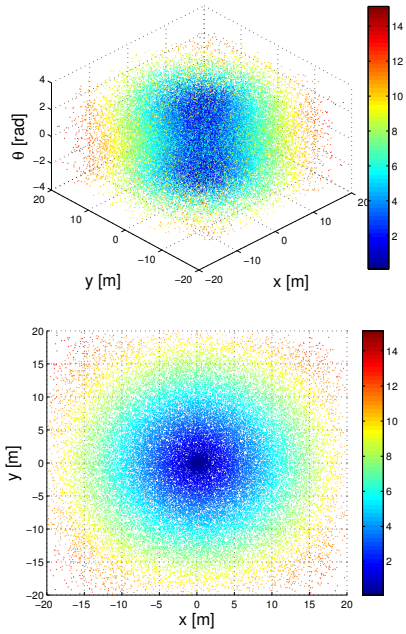

Fig. 4. The cost-to-go function $C(\mathbf{0}, \mathbf{x})$ for paths generated by the POSQ controller. **Top:** all three dimensions are showed. **Bottom:** the projection onto the (x, y)-Cartesian plane.

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 and we expect that there are interesting interactions between those features.

Thus, we perform feature expansion to find 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, via multiple validation iterations, we have found a set of fourteen features to characterize the cost-to-go function. The features make the geometric properties of connecting paths more explicit and facilitate the learning process (see Table I).

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, $c_i = C(\mathbf{x}_1, \mathbf{x}_2)$.

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 [13]. The model is

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

defined as

$$y = \sum_{m=1}^{M} \Phi_m(\mathbf{f}, \beta) \tag{4}$$

where M is the number of basis functions Φ. This choice is motivated by the simplicity of basis models and their fast inference. Concretely, we choose quadratic basis functions given by

$$y = \sum_{m=1}^{M_f} \beta_{m_1}(f_m - \beta_{m_2})^2 \tag{5}$$

where M_f is the number of features.

IV. Experiments

In the experiments we analyze both, the prediction accuracy of the model in terms of regression and ranking metrics, and how the learned distance metrics impacts planning time, path quality, and state space coverage.

A. Regression and ranking performance

To evaluate the prediction accuracy of the learned model, we use the following regression error metrics: the median of the residuals, the mean squared error normalized by the residual's variance (NMSE), and the coefficient of determination. Note, however, 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 a $\mathbf{x}_{rand}$, we are actually interested in the correct ranking of the tree vertices under the cost model rather than the predicted costs as such[1]. The typical strategy is then to choose the best ranked (lowest cost) vertex as $\mathbf{x}_{near}$. Thus, we also evaluate the model with respect to its ability to correctly predict the ranking of a set of states. To this end, we use the following ranking metrics: Kendall τ coefficient, Kendall τ_d distance and Spearman ρ coefficient [14]. Kendall τ and Spearman ρ coefficients are both a measure of correlation between two-ordinal level variables. τ and ρ are equal to 1 if the two rankings agree perfectly, if they disagree perfectly τ and ρ are equal to -1. Kendall τ_d distance measures the number of disagreement between two ordered lists: two ranks are equals if the distance is 0. We consider here the ranking of the 5 best vertices.

For training the distance metric, we used 50,000 samples and validated the results with 10,000 samples in terms of regression performance. For the ranking performance, we predict the cost and evaluate the metrics for a grid of poses over the entire configuration space without obstacles with a resolution of $0.1m$ in x, y and $\pi/4$ rad in θ.

For both regression and ranking performances we compare the basis function model to a neural network regression model with two hidden layers, 30 neurons in the first and 20 the second one. The architecture of the neural network was found through 5-fold cross validation. We train the neural network with the same data used to train the basis function model. We also give the regression and ranking results for the basis function model using the naive feature selection in which we only take the two poses $\mathbf{x}_1$ and $\mathbf{x}_2$ as input.

B. Planning performance

As our initial motivation was to speed up planning, we investigate how the learned model impacts planning time as well as path quality and coverage ability. To this end, we compare the learned distance metric with three baselines in three different environments. The baseline are the ground truth function (the POSQ extender) with the regular high-resolution integration time of 0.1 *sec*, the ground truth function with a low integration time of 0.5 *sec*, and the neural network regression model before described. 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 ordinary laptop with 2.70 GHz Intel i5 and 12 GB RAM.

To quantify planning performance we compute the averages of the following metrics: time for a single extension (T_e), time to find a solution (T_s), and path length in meters (l_p). Smoothness, although being an intuitive concept, is less straightforward to assess. We compute three measures that are 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:

$$\eta_{nmaJ} = -\frac{1}{v_{max}(t_2 - t_1)} \int_{t_1}^{t_2} \left| \frac{d^2\mathbf{v}}{dt^2} \right| dt,$$

[1]It could be argued that approximating a learning-to-rank problem by a classification problem is easier than by a regression problem but this discussion is left to future work.

Regression Performances			
Metric	**Basis fct. model**	**NN model**	**BFM naive features**
Median residuals	0.030	$1.607\ 10^{-6}$	7.8376
NMSE	0.005	$7.729\ 10^{-7}$	0.8843
Determination	0.999	1	0.8040
Ranking Performances			
Metric	**Basis fct. model**	**NN model**	**BFM naive features**
τ	1	1	-0.2
τ_d	0	0	0.6
ρ	1	1	-0.3

TABLE II

Regression and ranking performances

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

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

3) average number of peaks η_{pm}

$$\eta_{pm} = -|\mathcal{V}_{peaks}|.$$

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.

Different distance metrics may lead to different coverage of the state space [6]: we compare the ability of our approach to cover the state space with the three baselines in one environment. The entire state space has been divided in a set of 3D grid cells. The state space coverage is defined as the ratio of grid cells covered by the tree. In this experiment we perform 100 runs of 5,000 iterations per each metric.

The three simulated test environments, shown in Fig. 5, have been designed to 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 test the state space coverage, we considered the *random map* scenario.

C. Results

The results for the regression and ranking metrics are reported in Table II. They demonstrate the necessity to design informative features for this task given how clearly the naive feature selection fails to properly learn the true distance metric. The basis function model was chosen despite lower regression accuracy because it is able to achieve the same (perfect) results for the 5 best ranks than the neural network approach while being faster to compute.

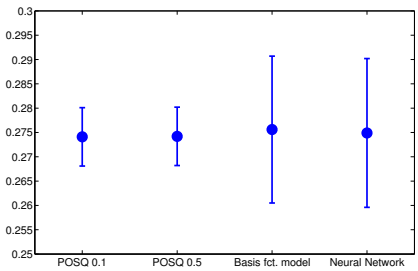

Fig. 6. Means and standard deviations of the state space coverage after 5000 iterations for all the distance metrics evaluated. To test the state space coverage, we considered the *random map* scenario.

The path quality, efficiency and smoothness results are given in Table III. They show that the learned distance metric is able to improve the time for the single extension by several factors and the time to find the first solution while having a negligible effect on the smoothness metrics. An important result is the one reported in Table III for the *open space* scenario where the impact of the learned distance metric is fully visible because, without obstacles, the other RRT heuristics that influence tree growth (extend function, collision checking and random state generation) have no effect. Therefore, the speed up in planning time is most dramatic in this environment. For the *hallway* scenario, the basis function model is still able to find a solution more than two times as fast while in the cluttered *random map* environment, the learning method is on par with the regular approach. This insight 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 smaller. As we can see the path quality remains largely unaffected, the values are all within the same order of magnitude than the original approach.

The coverage results are given in the Fig. 6. The learned distance metric thanks to its good approximation abilities does not degrade in terms of state space coverage: in average, its trees are able to cover the same amount of grid cells than the baseline methods.

V. Conclusions

In this paper, we have presented a new approach to learn the distance metric for RRT motion planning. Instead of computing local extension paths and evaluating their cost when growing the tree, we learn a parametric regression model that directly predicts the cost. With the proper choice of informative features and a simple regression model, we showed that this technique is very fast and leads to a significant speed up in planning time. The idea builds upon a novel RRT extend function for wheeled mobile robots which solves the two-point boundary value problem.

We compared our approach to three baseline methods including a neural networks regression model that was found through cross-validation. The learned model is faster by several factors with negligible loss of path quality and with the same state space coverage ability

Open space scenario						
Method	$\mathbf{T_e}$ $[s]$	$\mathbf{T_s}$ $[s]$	$\mathbf{l_p}$ $[m]$	$\eta_{\mathbf{nmaJ}}$	$\eta_{\mathbf{spal}}$	$\eta_{\mathbf{pm}}$
POSQ 0.1	0.0101886	14.5215	47.6487	$-7.32075\ 10^{-05}$	-0.802622	0.62
POSQ 0.5	0.00609328	5.3704	50.4071	$-6.77003\ 10^{-05}$	-0.73929	0.38
Basis fct. model	**0.00132488**	**1.0902**	48.608	$-6.44915\ 10^{-05}$	-0.759564	0.31
Neural Network	0.00355623	2.6722	48.2047	$-6.71913\ 10^{-05}$	-0.777919	0.55
Hallway scenario						
Method	$\mathbf{T_e}$ $[s]$	$\mathbf{T_s}$ $[s]$	$\mathbf{l_p}$ $[m]$	$\eta_{\mathbf{nmaJ}}$	$\eta_{\mathbf{spal}}$	$\eta_{\mathbf{pm}}$
POSQ 0.1	0.0108161	92.9419	54.6069	$-7.44230\ 10^{-05}$	-0.844397	0.90
POSQ 0.5	0.0079751	70.6947	56.0821	$-8.01868\ 10^{-05}$	-0.8833	3.23
Basis fct. model	**0.00178928**	**29.0099**	62.8529	$-6.70743\ 10^{-05}$	-0.961323	8.2
Neural Network	0.00531763	65.2463	63.3869	$-7.22698\ 10^{-05}$	-1.01336	0.39
Random map scenario						
Method	$\mathbf{T_e}$ $[s]$	$\mathbf{T_s}$ $[s]$	$\mathbf{l_p}$ $[m]$	$\eta_{\mathbf{nmaJ}}$	$\eta_{\mathbf{spal}}$	$\eta_{\mathbf{pm}}$
POSQ 0.1	0.0117979	37.7666	49.8373	$-6.04633\ 10^{-05}$	-0.794841	0.45
POSQ 0.5	0.0108875	20.88	50.129	$-6.12429\ 10^{-05}$	-0.795457	0.68
Basis fct. model	**0.0039755**	23.3264	53.2168	$-6.85253\ 10^{-05}$	-0.844749	3.89
Neural Network	**0.00378317**	**19.452**	53.5202	$-6.98598\ 10^{-05}$	-0.865452	0.33

TABLE III

SMOOTHNESS AND EFFICIENCY RESULTS

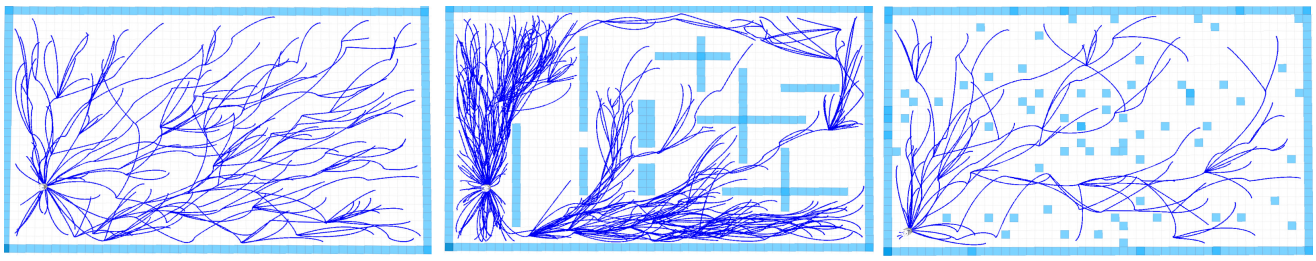

Fig. 5. The three environments and example trees obtained using our approach. **Left:** the *open space* scenario. **Middle:** *hallway* scenario. **Right:** *random map* scenario

than the true distance metric.

In future work, we aim for high dimensional spaces: we believe that for complex systems such as humanoids and mobile manipulators, our approach will lead to an even more dramatic speed up of planning time. Clearly, new features will be required and a two-point boundary value problem solver is needed. We are also interested to incorporate the learned model into RRT variants like RRT* or T-RRT.

ACKNOWLEDGEMENT

The authors thank Frank Hutter and Rudolph Triebel 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. x, no. x, 2011.

[3] ——, "Sampling-based optimal motion planning for nonholonomic dynamical systems," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5041–5047.

[4] P. Cheng and S. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, 2001, pp. 43–48 vol.1.

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

[6] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5021–5028.

[7] D. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013.

[8] 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 *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012.

[9] Y. Li and K. Bekris, "Learning approximate cost-to-go metrics to improve sampling-based motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 4196–4201.

[10] S. LaValle and J. Kuffner, J.J., "Randomized kinodynamic planning," in *Int. Conf. on Robotics and Automation (ICRA)*, vol. 1, 1999.

[11] 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)*, 2014.

[12] A. Astolfi, "Exponential stabilization of a wheeled mobile robot via discontinuous control," *Journal of dynamic systems, measurement, and control*, vol. 121, no. 1, 1999.

[13] J. J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[14] C. Spearman, "The proof and measurement of association between two things," *International journal of epidemiology*, vol. 39, no. 5, pp. 1137–1150, 2010.