A Novel RRT Extend Function for Efficient and Smooth Mobile Robot Motion Planning

Luigi Palmieri Kai O. Arras

***Abstract*— In this paper we introduce a novel RRT extend function for wheeled mobile robots. The approach computes closed-loop forward simulations based on the kinematic model of the robot and enables the planner to efficiently generate smooth and feasible paths that connect any pairs of states. We extend the control law of an existing discontinuous state feedback controller to make it usable as an RRT extend function and prove that all relevant stability properties are retained. We study the properties of the new approach as extender for RRT and RRT* and compare it systematically to a spline-based approach and a large and small set of motion primitives. The results show that our approach generally produces smoother paths to the goal in less time with smaller trees. For RRT*, the approach produces also the shortest paths and achieves the lowest cost solutions when given more planning time.**

I. Introduction

Planning with rapidly-exploring random trees (RRT) [1] has become a popular approach to robot motion planning. RRT planners are single-query sampling-based planners that grow a tree of configurations to eventually cover the entire state space. A probabilistically optimal RRT variant named RRT* has been introduced by Karaman and Frazzoli [2]. RRT* trees grow based on the notion of a cost: under the assumptions given in [3] the solution converges to the optimum as the number of samples approaches infinity.

For robots with kinematic or kinodynamic constraints, the *extend function*, the function that grows the tree by finding collision-free trajectories to new sampled configurations, becomes a key component. Its task is to connect any pair of states under differential constraints which represents by itself a local planning problem also referred to as the *two-point boundary value problem*.

Here, we consider the motion planning problem for non-holonomic wheeled robots in 2D with the goal of particularly smooth and natural real-time motion generation for robots in human environments. To this end, we propose a new extend functions for RRT and RRT* which shall enable the planner to efficiently generate smooth paths. Previously used extend functions include motion primitives [1, 4, 5, 6], optimal controllers [7, 8], shooting methods [9], splines [10], and closed-loop controllers [11].

Motion primitives have originally been proposed for RRT-based planning under differential constraints. LaValle and Kuffner [1] implement the extend function as a forward simulation of a set of predefined discretized controls, so called motion primitives. The approach satisfies the constraints, is efficient to compute and easy to implement: the tree is extended with the primitive that is found to come closest to the new sampled configuration $\mathbf{x}_{new}$. However, the method has several shortcomings: it does not fully solve the two-point boundary value problem as the orientation of $\mathbf{x}_{new}$ is ignored, the extension of the tree even by the closest motion primitive may still be far-off from $\mathbf{x}_{new}$, and the concatenation of primitives may lead to sequences of discontinuous inputs and non-smooth trajectories. The last point was addressed by Frazzoli *et al.* [4] who propose a finite-state machine called a Maneuver Automaton to allow correct (and thereby smooth) concatenation of motion primitives to complex motion trajectories. However, its use in RRT-based planning has not been studied.

Recently, Perez *et al.* [7] 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, but will in general not reach the target state exactly. Webb and van den Berg [8] use a finite-horizon optimal controller as local planner. They can optimize a certain class of cost functions to trade off between time and control effort. Although optimal control techniques may produce high-quality solutions to the two-point boundary value problem, they typically suffer from high computational costs and numerical issues that can make them unsuitable for motion planning in real-time.

Hwan Jeon *et al.* [9] use the shooting method to numerically solve the two-point boundary value problem to obtain an extend function for RRT*. The method allows for time-optimal maneuvers of a high-speed off-road vehicle. As with optimal control techniques, shooting methods may have issues with numerical stability and computational costs for our application.

In a recent work, Yang *et al.* [10] use splines as RRT extend function. The authors take cubic Bézier splines that guarantee curvature continuity of paths and are able to satisfy upper-bounded curvature constraints. With our goal of smooth and natural motion generation, we consider splines to be a potentially interesting approach and include a spline-based extend function into our experimental comparison. However, instead of cubic Bézier splines which are limited to curves with continuous curvature, we will use η^3 splines, introduced by Piazzi *et al.*

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.

[12] that produce curves with a continuous derivative of the curvature, therefore generating even smoother paths than cubic Bézier splines.

Kuwata *et al.* [11] introduce closed-loop RRT (CL-RRT), a modified RRT for real-time local lane following with a car using an extend function based on a closed-loop model. Given a sampled control input, the method runs a forward simulation using the vehicle and controller models to predict and then evaluate extend trajectories.

The contribution of our paper is as follows:

- We propose an extender based on closed-loop predictions for a non-holonomic wheeled mobile robot. It efficiently solves the two-point boundary value problem by exponentially converging to the goal state from any start state. We extend the control law of the original approach by Astolfi [13] with a term that leads to quasi-constant path velocities along local path concatenations – a key ability for RRT extend functions. We also prove the relevant stability properties under our modification.
- We systematically compare our approach to two alternative extenders, namely motion primitives (two sets of different size) and splines. The experiments demonstrate that our approach outperforms both methods in many relevant metrics: smoother paths and shorter planning time (with RRT), shorter paths (with RRT*), and significantly smaller trees (both). We also find that our method can benefit most from the incremental path improvement ability of RRT* resulting in the lowest cost solutions when given more planning time.
- To the best of our knowledge, this paper presents the first systematic study of the impact of different extend functions on RRT and RRT* performance and path quality. Its necessity is corroborated by the significant variations of key metrics only caused by the use of different extend functions.

The paper is structured as follows: the next Section reviews the RRT algorithm and typical extend functions. In Section III we describe our approach which is then experimentally evaluated in Section IV. We discuss the results in Section V, and Section VI concludes the paper.

II. Rapidly Exploring Random Trees

We briefly review the RRT algorithm for planning under differential constraints. Let $\mathcal{X} \subset \mathbb{R}^d$ be the configuration space and $\mathcal{U} \subset \mathbb{R}^m$ the control space. A non-holonomic wheeled mobile robot can be described by a differential equation as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{1}$$

where $\mathbf{x}(t) \in \mathcal{X}$, $\mathbf{u}(t) \in \mathcal{U}$, for all t, $\mathbf{x}_0 \in \mathcal{X}$ and f is a function describing the kinematics of the system. The RRT algorithm solves a feasible kinematic motion planning problem p: given an obstacle space $\mathcal{X}_{obs} \subset \mathcal{X}$, a free space $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$, an initial state $\mathbf{x}_{init} \in \mathcal{X}_{free}$ and a goal region $\mathcal{X}_{goal} \subset \mathcal{X}_{free}$, find a control $\mathbf{u}(t) \in \mathcal{U}$ with domain $[0, T]$, $T > 0$, such that the unique trajectory $\mathbf{x}(t)$ satisfies equation (1), is in the free space $\mathcal{X}_{free} \subseteq \mathcal{X}$ and goes from $\mathbf{x}_{init}$ to a goal $\mathbf{x}_{goal} \in \mathcal{X}_{goal}$. The RRT procedure is outlined in Algorithm 1.

Algorithm 1 Rapidly-exploring Random Tree

```
function RRT(x_init , x_goal)
τ.add_vertex(x_init)
while k ≤ K do
  x_rand ⇐ random_state(X)
  x_near ⇐ nearest_neighbor(τ, x_rand)
  x_new, u_best ⇐ extend(x_near, x_rand)
  τ.add_vertex(x_new)
  τ.add_edge(x_near, x_new, u_best)
  if x_new ∈ X_goal then
    return extract_traj(x_new)
  end if
end while
return failure
```

A. Extend Function

The purpose of the extend function is to connect new states to the tree: it grows a branch from $\mathbf{x}_{near}$ toward $\mathbf{x}_{rand}$. The terminal state of the new branch, $\mathbf{x}_{new}$, may differ (largely) from $\mathbf{x}_{rand}$ depending on the extend function used. $\mathbf{x}_{new}$ is then added to the tree τ together with the intermediate points of the new local path and the selected $\mathbf{u}$. The expansion fails if a collision along the path occurs.

We briefly review motion primitives, the originally proposed extenders as part of RRT. The approach is based on the forward propagation of a control input into a system simulator. Given an initial state $\mathbf{x}_0$, an integration time Δt, an integration time step t_s and a input $\mathbf{u}_i$ from a discrete set of controls $\mathcal{U} = \{\mathbf{u}_1, .., \mathbf{u}_m\}$, a trajectory $\mathbf{x}_i(t)$ is generated by integrating Eq. (1)

$$\mathbf{x}_i(t) = \int_0^{\Delta t} f(\mathbf{x}_i(t), \mathbf{u}_i(t))\ dt + \mathbf{x}_0, \quad i = 1, ..., m. \tag{2}$$

All the controls in $\mathcal{U}$ are checked, and the one that brings the expansion closest to $\mathbf{x}_{rand}$ (according to some distance metric) is stored together with the associated local trajectory that will be added to the tree τ. To minimize the time needed to extend the tree, the motion primitives can be precomputed off-line.

An alternative approach to extending the tree is to employ a full-fetched local planner that generates trajectories $\mathbf{x}(t) \in \mathcal{X}_{free}$ and the corresponding continuous controls $\mathbf{u}(t)$.

B. RRT* Extend Function

The extension procedure in RRT* is more complex. It is based on the concept of *near neighbors*, the neighbors within a specified radius of a node. The first step of the extension is to connect a newly added vertex to its neighbor vertex with minimal cost. The next step is to

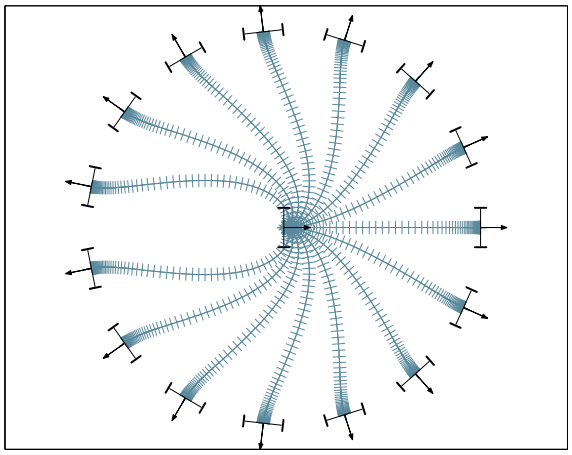

Fig. 1. Trajectories of the controller when steering the robot from the center to the poses on the circle

rewire the tree: if the path from the newly created vertex to a near neighbor node has a lower cost than the near neighbor, then the parent of the near vertex is changed to the new vertex. Each time the algorithm attempts to connect two vertices a *steer function* is called for which RRT extend functions can be used.

III. The Approach: POSQ

The proposed extend function computes closed-loop forward simulations based on the kinematic model of a non-holonomic wheeled mobile robot. It generates the trajectory $\mathbf{x}(t)$ and controls $\mathbf{u}(t)$, $t \in [0, T]$, $T > 0$, that connect any given pair of 2D poses. Thus, it solves the two-point boundary value problem for such kinematic systems, see Fig. 1. The tree is grown in the configuration space $\mathbb{R}^2 \times \mathbb{S}^1$ where each configuration $\mathbf{x}$ consists of the Cartesian position of the wheeled mobile robot and its orientation, i.e. (x, y, θ).

The approach, originally proposed by Astolfi [13], solves the problem of exponential stabilization of the kinematic and dynamic model of the wheeled mobile robot. It is not an optimal controller but has provable local and global stability, a light-weight implementation, and generates smooth trajectories. For extend functions, optimality may be less relevant than efficiency, smoothness and the ability to fully solve the two-point boundary value problem. This is particularly true for RRT* for non-holonomic dynamical systems [14].

We briefly summarize the original approach [13] and describe our extension in the next subsection.

Let ρ be the Euclidean distance between the initial pose and the goal pose ($\mathbf{x}_{near}$ and $\mathbf{x}_{rand}$ in an RRT notation), ϕ the angle between the x-axis of the robot reference frame $\{X_R\}$ and the x-axis of the goal pose frame $\{X_G\}$, α the angle between the y-axis of the robot reference frame and the vector Z connecting the robot with the goal position, v the translational and ω the angular robot velocity (Fig. 2). Then, the method makes a Cartesian-to-polar coordinate transform to describe the

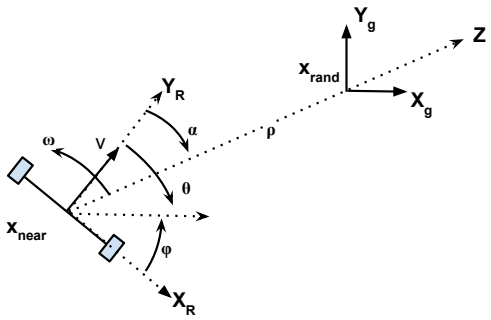

Fig. 2. Notation and robot to goal relations

kinematics using the open loop model

$$\begin{aligned} \dot{\rho} &= -\cos\alpha \; v, \\ \dot{\alpha} &= \frac{\sin\alpha}{\rho} v - \omega, \\ \dot{\phi} &= -\omega. \end{aligned} \tag{3}$$

and the feedback law

$$\begin{aligned} v &= K_\rho \rho, \\ \omega &= K_\alpha \alpha + K_\phi \phi. \end{aligned} \tag{4}$$

As shown in [13], this feedback law guarantees smooth trajectories without cusps.

A. Our control law

The original approach, however, generates trajectories of decaying forward velocity bringing the robot to a stop at each goal. The concatenation of such local paths would result in final paths of unnatural and slow movements.

Thus, we modify the feedback law so as to have quasi constant forward velocity at a desired maximum value across multiple expansions. We will prove that this modification retains local stability and that the robot's heading converges asymptotically to the desired equilibrium point.

Considering the open loop model in Eq. (3) obtained by the polar coordinate transform, we define the nonlinear feedback law

$$\begin{aligned} v &= K_\rho \tanh(K_v \rho), \\ \omega &= K_\alpha \alpha + K_\phi \phi \,. \end{aligned} \tag{5}$$

Substituting the control law (5) into the open loop model we obtain the following closed loop model

$$\begin{aligned} \dot{\rho} &= -K_\rho \cos\alpha \tanh(K_v \rho), \\ \dot{\alpha} &= K_\rho \frac{\sin\alpha}{\rho} \tanh(K_v \rho) - K_\alpha \alpha - K_\phi \phi, \\ \dot{\phi} &= -K_\alpha \alpha - K_\phi \phi \,. \end{aligned} \tag{6}$$

We now describe the conditions for which local stability holds and prove heading convergence.

1) Local Stability: We can locally approximate the closed loop model (6) by

$$\begin{aligned}\dot{\rho} &= -K_\rho K_v \rho,\\ \dot{\alpha} &= -(K_\alpha - K_\rho K_v)\alpha - K_\phi \phi,\\ \dot{\phi} &= -K_\alpha \alpha - K_\phi \phi.\end{aligned}$$

which is locally exponentially stable if and only if the eigenvalues of the matrix describing the linear approximation of the model have all negative real parts. For that, we need to have

$$\begin{aligned} K_v &> 0,\\ K_\rho &> 0,\\ K_\phi &< 0,\\ K_\alpha + K_\phi - K_\rho K_v &> 0\,. \end{aligned} \tag{7}$$

Considering the closed loop model (6), assume $\alpha(0) \in]-\frac{\pi}{2}, \frac{\pi}{2}]$, and $\phi(t) \in]n\pi, n\pi]$ for all t. Then, if

$$K_\alpha + 2nK_\phi - \frac{2}{\pi} K_\rho K_v > 0$$

holds one has $\alpha(t) \in]-\frac{\pi}{2}, \frac{\pi}{2}]$ for all $t > 0$ which means that the robot trajectory will always stay in this region: we have defined a *trapping region*. Thus, together with the condition $K_\rho K_v > 0$, the robot will move monotonically towards the origin.

2) Heading Convergence: We want the robot to move towards to goal $\mathbf{x}_{rand}$ but notice in (6) that the goal position (the origin) can not be reached because ρ is a singularity point. Thus, we define an arbitrarily small number to which ρ converges, $\rho \to \epsilon$, with $\epsilon > 0$, $\rho > \epsilon$. Let us focus on the following reduced subsystem which describes how the orientation evolves

$$\begin{aligned}\dot{\alpha} &= -K_\alpha \alpha - K_\phi \phi + K_\rho \sin\alpha \frac{\tanh(K_v \rho)}{\rho},\\ \dot{\phi} &= K_\alpha \alpha - K_\phi \phi\,.\end{aligned}$$

Given that $\dot{\rho}$ is strictly negative, we want to find the conditions for which the above vector field has a unique equilibrium point ($\alpha = 0$, $\phi = 0$) to which all trajectories converge asymptotically for all $\rho > \epsilon$. This is equivalent to minimizing the orientation error as well as stopping the robot at $\mathbf{x}_{new}$, ϵ meters away from $\mathbf{x}_{rand}$.

If we consider the candidate Lyapunov function

$$V(\alpha,\phi) = (-K_\alpha \alpha + K_\phi \phi)^2 + 2\ K_\phi K_\rho\ (\cos\alpha - 1)\ \tanh(K_v \rho), \tag{8}$$

we can show that

$$\dot{V}(\alpha,\phi) = 2\ K_\phi K_\rho \alpha\ \sin\alpha\ \tanh(K_v\rho)\ \left[K_\phi + K_\alpha - K_\rho \frac{\sin\alpha}{\alpha}\ \tanh(K_v\rho)\right], \tag{9}$$

Given that condition (7) holds, V is positive and $\dot{V}$ is non-positive in all $S_2 = \{(\alpha,\phi) \in R^2 \mid \alpha \in]-\frac{\pi}{2}, \frac{\pi}{2}],\ \phi \in (-2\pi, 2\pi]\}$, so the state of the system converges asymptotically to the origin according to the LaSalle invariance principle.

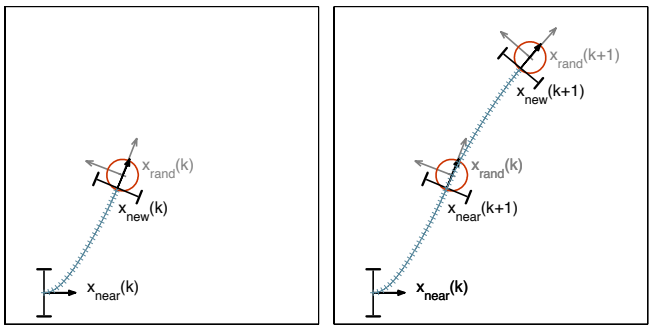

Fig. 3. A simple two-step expansion example. With k being the time index of successive extensions, the proposed controller extends the tree from $\mathbf{x}_{near}(k)$ to $\mathbf{x}_{rand}(k)$ until the local trajectory enters the disk of radius γ at $\mathbf{x}_{new}(k)$. The procedure is repeated for $k+1$.

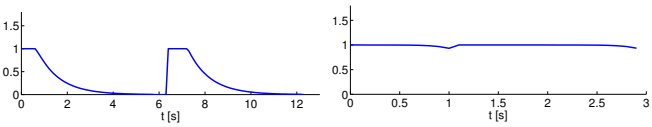

Fig. 4. Translational robot velocity in [m/s] with the original control law from [13] (left) and our control law (right) across the concatenation of two extensions. For the same two goal poses the new law allows for much faster movements.

It exists a positively invariant set

$$M = \left\{(\alpha,\phi) \in S_2 |\ \ V \leq \frac{9}{4} k_\phi^2 \pi^2 - 2K_\phi K_\rho \right\},$$

which is contained in S_2, and it contains $S_1 = \{(\alpha,\phi) \in R^2 \mid \alpha \in]-\frac{\pi}{2}, \frac{\pi}{2}],\ \phi \in (-\pi, \pi]\}$. M contains only the equilibrium point ($\alpha = 0$, $\phi = 0$). Thus, all trajectories starting in S_1 and contained in S_2 converge asymptotically to the origin according to the Poincare-Bendixson Theorem.

Notice that we are not solving a stabilization problem like in the original approach [13]. The new control law allows us, during expansion of the tree from $\mathbf{x}_{near}$ towards $\mathbf{x}_{rand}$, to minimize the error in orientation and stop when the local trajectory is close enough to the goal $\mathbf{x}_{rand}$. A $\gamma > 0$ threshold can be defined as minimum Euclidean distance that stops the expansion towards $\mathbf{x}_{rand}$. It is guaranteed that the terminal state $\mathbf{x}_{new}$ is not further away from $\mathbf{x}_{rand}$ than γ, which thus becomes a tunable error bound. Threshold γ can be seen as the radius of a circle centered at $\mathbf{x}_{rand}$, see Fig. 3. In practice, γ is chosen to be a few centimetres.

The new control law does not remove the velocity decay toward the goal but makes it significantly sharper. So sharp, that even small values for γ cause the decay to disappear and allow for quasi-constant forward velocities along the previously explained extension procedure. See Fig. 4 for a comparison. The method is named *POSQ* as it acts like a pose controller.

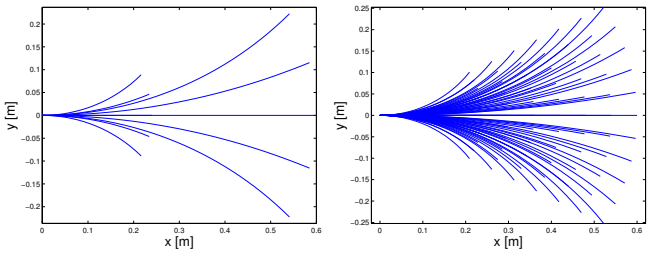

Fig. 5. The motion primitive sets, $\mathcal{U}_{small}$ (left) with 10 motion primitives and $\mathcal{U}_{large}$ (right) with 77 motion primitives.

IV. Experiments

In the experiments, we evaluate the new extend function and compare it to two alternative methods, namely motion primitives (two sets of different size) and splines. We quantify their impact on planning performance in terms of time, tree size and path quality in three different simulated environments. We use both RRT and RRT* as planning algorithms.

The POSQ parameters are $K_\rho = 1,\ K_\phi = -1,\ K_\alpha = 6,\ K_v = 3.8,\ \gamma = 0.15$. The two sets of motion primitives $\mathcal{U}_{small}$ with 10 controls and $\mathcal{U}_{large}$ with 77 controls are shown in Fig. 5. For the spline-based extend function we use η^3 splines [12], seventh order polynomial spline whose paths have continuous tangent vectors, curvature and curvature derivatives along the arc length.

All extenders share the same integration time step t_s and velocity limits. For each combination of extender/map/planning algorithm we perform 100 runs and compute the average and standard deviation of all metrics. We use uniform sampling and the Euclidean distance as distance metric.

The RRT* cost function, derived from [1], has two terms, one for the approximated path length and one that measures heading changes along the path, both with equal weights ($w_d = w_q$)

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

$N_e + 1$ are the intermediate points $\mathbf{P}_i$ of the local path and $\mathbf{q}_i$ the associated quaternions. The RRT* neighbor radius is constant at a high value with respect to the map size, we use a linear neighbor search.

Our implementation is based on the C++ SMP template library [15]. All experiments were running on an ordinary PC with 2.67 GHz Intel Core i7 and 10 GB of RAM.

A. Metrics

To quantify planning performance we compute the averages and standard deviations of the following metrics: tree size as the number of vertices (N_v), time to find a solution (RRT) or a first solution (RRT*) (T_s), and path length in meters (l_p).

Smoothness, although being an intuitive concept, is less straightforward to measure precisely. In [16], Balasubramanian *et al.* survey a number of metrics to quantify movement smoothness. We adopt the following 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:

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

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.

B. Test Environments

Planning is carried out in three simulated environments (Fig. 6). In the *office environment*, there are few alternative ways to the goal. It has several local minima, the goal lies behind a U-shaped obstacle, and an asymmetry makes that the shortest path go through a narrow passage. The *hallway scenario* contains more open spaces and alternative paths to the goal. The *random map environment* contains 100 randomly placed square obstacles. There are many homotopy classes, some require more or less maneuvers than others. The map size in all scenarios is $50m \times 30m$.

V. Results and Discussion

The RRT results are given in Table I, the RRT* results in Table II. The best values in each metric are highlighted in bold.

With RRT as the planning algorithm, the proposed extend function *POSQ* outperforms motion primitives and splines in all metrics except path length. It produces smoother paths and finds the goal in less time with significantly smaller trees. The low number of tree vertices and the smaller planning times are mainly due to the ability of our approach to better follow the Voronoi bias and deeply enter unexplored regions of the configuration space. This is unlike, for example, motion primitives that require the concatenation of many small local expansions for the same exploration effort. In fact, all continuous extend functions that fully solve the two-point boundary value problem possess this property as also confirmed by the similar trends in the results of the spline-based

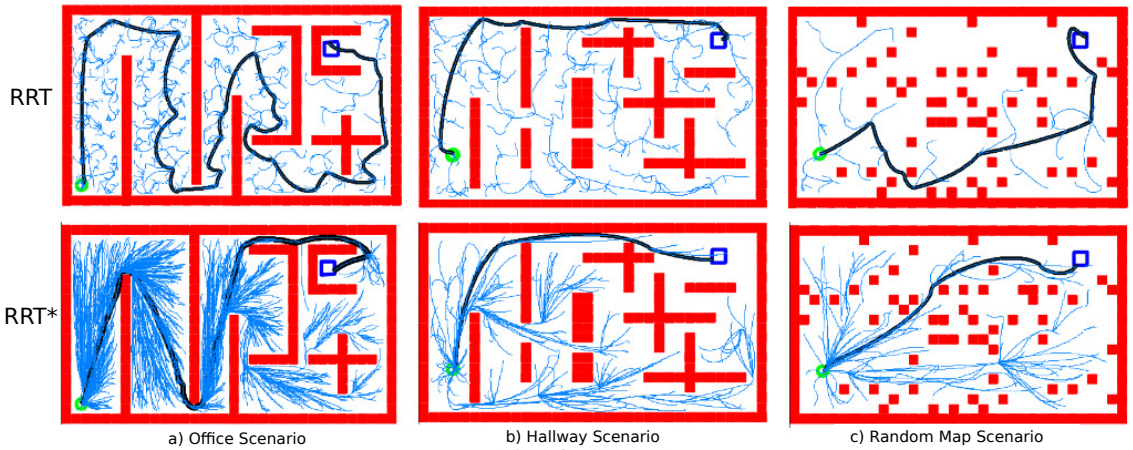

Fig. 6. The three environments and example solutions found with the proposed extend function for RRT (top row) and RRT* (bottom row, showing the first solution). The start state (green circle) is always in the bottom left, the goal region (blue square) in the top right.

Office scenario												
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$		$\eta_{\mathbf{nmaJ}}$		$\eta_{\mathbf{spal}}$		$\eta_{\mathbf{pm}}$	
POSQ	**1667**	± 713	**0.197**	± 0.09	150.739	± 18.446	**−0.051**	± 0.006	**−4.464**	± 0.166	**0**	± 0
MP $\mathcal{U}_{small}$	13335	± 3283.4	2.235	± 0.597	134.108	± 8.687	−0.062	± 0.0006	−5.8648	± 0.064	37.8	± 6.7
MP $\mathcal{U}_{large}$	14090	± 3523.1	2.583	± 0.720	**133.504**	± 8.85	−0.062	± 0.0007	−5.8901	± 0.066	21.09	± 6.1
η^3 splines	2369	± 939.9	0.274	± 0.108	159.78	± 14.88	−0.55	± 0.06	−6.88	± 0.16	**0**	± 0
Hallway scenario												
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$		$\eta_{\mathbf{nmaJ}}$		$\eta_{\mathbf{spal}}$		$\eta_{\mathbf{pm}}$	
POSQ	**520.4**	± 379.2	**0.050**	± 0.021	85.857	± 16.740	**−0.039**	± 0.007	**−3.602**	± 0.282	**0**	± 0
MP $\mathcal{U}_{small}$	2458.3	± 868.09	0.388	± 0.124	72.918	± 10.072	−0.0631	± 0.001	−5.237	± 0.138	22.32	± 5.7
MP $\mathcal{U}_{large}$	2358.6	± 922.2	0.367	± 0.127	**71.734**	± 9.254	−0.0632	± 0.001	−5.2528	± 0.13	11.12	± 4.4
η^3 splines	548.3	± 514.5	0.0526	± 0.026	86.659	± 18.49	−0.382	± 0.089	−5.93	± 0.359	**0**	± 0
Random map scenario												
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$		$\eta_{\mathbf{nmaJ}}$		$\eta_{\mathbf{spal}}$		$\eta_{\mathbf{pm}}$	
POSQ	**277.2**	± 351.5	**0.031**	± 0.022	62.465	± 9.003	**−0.027**	± 0.007	**−2.881**	± 0.345	**0**	± 0
MP $\mathcal{U}_{small}$	1095.1	± 664.2	0.176	± 0.104	**56.448**	± 5.242	−0.0638	± 0.001	−4.977	± 0.099	18.51	± 4.8
MP $\mathcal{U}_{large}$	1124.6	± 646.4	0.168	± 0.09	57.27	± 5.3	−0.0637	± 0.001	−5.029	± 0.098	9.48	± 4.10
η^3 splines	519.6	± 718.6	0.044	± 0.035	66.686	± 9.514	−0.3013	± 0.082	−5.4851	± 0.337	**0**	± 0

TABLE I
RRT Results

Office scenario												
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$		$\eta_{\mathbf{nmaJ}}$		$\eta_{\mathbf{spal}}$		$\eta_{\mathbf{pm}}$	
POSQ	**1825**	± 785.7	**315.1**	± 187.3	**105.33**	± 4.96	−0.3261	± 0.135	**−5.128**	± 0.29	**23.1**	± 11.1
MP $\mathcal{U}_{small}$	13571	± 3601.8	731.3	± 301.93	131.88	± 8.35	**−0.0622**	± 0.001	−5.865	± 0.06	38.59	± 8.1
MP $\mathcal{U}_{large}$	14146	± 2562.3	933.5	± 258.50	134.257	± 8.849	−0.0623	± 0.001	−5.897	± 0.07	30.65	± 5.6
η^3 splines	3438	± 4254.9	646.5	± 483.62	116.4909	± 6.337	−22.3	± 6.81	−8.49	± 0.09	47.30	± 27
Hallway scenario												
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$		$\eta_{\mathbf{nmaJ}}$		$\eta_{\mathbf{spal}}$		$\eta_{\mathbf{pm}}$	
POSQ	**697.9**	± 704	66.9	± 109.8	**54.16**	± 3.26	−0.1430	± 0.126	**−3.6479**	± 0.56	**3.1**	± 4.26
MP $\mathcal{U}_{small}$	2385.3	± 987	**51.3**	± 31.8	71.0350	± 9.5819	−0.0631	± 0.0007	−5.212	± 0.129	16.2	± 5.2
MP $\mathcal{U}_{large}$	2529.7	± 1020.4	68.4711	± 4.12	71.3390	± 8.4327	**−0.0630**	± 0.0002	−5.2485	± 0.12	11.3	± 3.99
η^3 splines	3787.2	± 14583	637	± 2921	57.302	± 4.2401	−9.365	± 11.91	−7.08	± 0.59	12.9	± 13.9
Random map scenario												
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$		$\eta_{\mathbf{nmaJ}}$		$\eta_{\mathbf{spal}}$		$\eta_{\mathbf{pm}}$	
POSQ	**400.8**	± 551.2	43.16	± 90.54	**46.11**	± 2.4	−0.0896	± 0.103	**−2.95**	± 0.697	**1.3**	± 2.71
MP $\mathcal{U}_{small}$	1028.1	± 597	**13.11**	± 12.48	56.66	± 5.08	**−0.0636**	± 0.0007	−4.98	± 0.10	18.7	± 4.95
MP $\mathcal{U}_{large}$	998.7	± 535.8	15.4411	± 14	56.9105	± 4.7867	−0.0637	± 0.0005	−5.02	± 0.08	10.09	± 4.2
η^3 splines	815.7	± 2421.1	157.8	± 715.8	48.5212	± 2.7370	−7.67	± 11.87	−6.46	± 0.97	5.0	± 7.54

TABLE II
RRT* Results

extender. The motion primitive extenders find shorter paths which is not surprising given the much denser trees from the multitude of small-sized extensions. With RRT* as planner, our extender outperforms the other methods

in tree size, path length and two of three smoothness measures. The fact that our method finds the shortest paths, and so the lowest cost in all the cases, suggests that it is particularly easy to rewire in the sense of the cost function, quite in contrast to motion primitives. This is also pointed out by Webb and van den Berg [8] who state that the RRT* rewiring procedure is well suited for continuous extension approaches where reachability of a state is not compromised. Figure 7 is another indication in this direction. It shows an example cost trend when given more planning time. The POSQ extender can benefit most from the incremental path improvement of RRT*. While the proposed extender is the smoothest

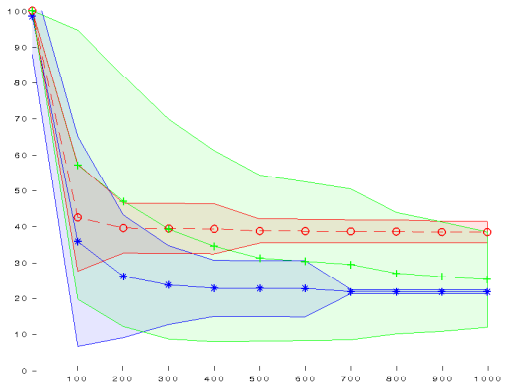

Fig. 7. The RRT* cost C computed over 1000 seconds for the Random Map Scenario. The trends are displayed (mean and standard deviation): in blue the POSQ results, in red the Motion Primitive ones and in green the Splines.

in terms of the η_{spal} and η_{pm} measures, it falls behind the motion primitive approach in the jerk-related metric η_{nmaJ}. This may be explained by the much denser trees with several factors more vertices that allow solutions with fewer maneuvers. Regarding the time to find the first solution, T_s, the results are inconclusive. The high variance is mainly due to the large number of homotopy classes, particularly in the random map and hallway environments. POSQ and the spline-based extender (the two continuous approaches) grow the tree deeply into unexplored regions and discover many different ways to the goal, also inefficient ones at times. However, as discussed before, such first solutions can be improved when given more planning time, particularly well by the POSQ extender.

VI. Conclusions

In this paper we have presented a novel RRT extend function for nonholonomic wheeled mobile robots. We have evaluated its impact on planning performance and path quality and found that it outperforms motion primitives and a spline-based approach in many relevant metrics. It enables RRT to find smoother paths in less time with smaller trees, and it enables RRT* to find shorter paths with smaller trees while being on par in planning time and smoothness. We also found that our method can benefit most from the cost-guided rewiring procedure of RRT* resulting in the lowest cost solutions when given more planning time. In future work we consider the extension of our method to kinodynamic models and the incorporation of recent RRT improvements such as regression avoidance and viability filtering. We also plan to develop specific heuristics for nonholonomic systems that help to select states in the tree for further expansion.

Acknowledgments

The authors thank Christoph Sprunk for valuable discussions and feedback. This work has partly been supported by the European Commission under contract number FP7-ICT-600877 (SPENCER)

References

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

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

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

[4] E. Frazzoli, M. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, 2005.

[5] M. Kalisiak and M. van de Panne, "RRT-blossom: RRT with a local flood-fill behavior," in *Int. Conf. on Robotics and Automation (ICRA)*, Orlando, USA, 2006.

[6] ——, "Faster motion planning using learned local viability models," in *Int. Conf. on Robotics and Automation (ICRA)*, Rome, Italy, 2007.

[7] 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)*, Minneapolis, USA, 2012.

[8] 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.

[9] J. Hwan Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Decision and Control and European Control Conference (CDC-ECC)*, Rome, Italy, 2011.

[10] K. Yang, S. Moon, S. Yoo, J. Kang, N. Doh, H. Kim, and S. Joo, "Spline-based rrt path planner for non-holonomic robots," *Journal of Intelligent and Robotic Systems*, vol. 73, no. 1-4, 2014.

[11] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, 2009.

[12] A. Piazzi, C. Bianco, and M. Romano, "η^3-splines for the smooth path generation of wheeled mobile robots," *IEEE Transactions on Robotics*, vol. 23, no. 5, 2007.

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

[14] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5041–5047.

[15] S. Karaman, "Sampling-based Motion Planning (SMP) Template Library," http://www.mit.edu/~sertac/smp_doc.

[16] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet, "A robust and sensitive metric for quantifying movement smoothness," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, 2012.