Poli-RRT*: optimal RRT-based planning for constrained and feedback linearisable vehicle dynamics

Matteo Ragaglia, Maria Prandini and Luca Bascetta

Abstract— **This paper proposes a Rapidly exploring Random Trees (RRT)-based planning strategy (Poli-RRT*) that computes optimal trajectories in presence of vehicle constraints (like, for instance, differential and actuation constraints) without approximating the system dynamics through linearisation, but relying on exact linearisation. In this way, the optimal control problem that is introduced to determine the trajectories for extending the tree can be expressed as a quadratic program and efficiently solved. Poli-RRT* is formulated and tested via simulation on a unicycle-like model of a vehicle subject to actuation constraints. Notably, the approach can be applied to any other feedback linearisable vehicle model, subject to different types of constraints.**

I. Introduction

In the past few years autonomous aerial, underwater, and ground vehicles have gained considerable popularity in the robotics field, as they allow to perform critical tasks without endangering the life of human pilots/drivers. Their applications range from scientific exploration to provision of commercial services, from search and rescue to military operations like for instance reconnaissance or intelligence gathering.

Nowadays most of such unmanned vehicles are teleoperated by humans through dedicated control stations. However, as application scenarios become more and more complex, the possibility to let human operators focus on high-level tasks rather than on control of the vehicle becomes more and more interesting. As a consequence, there is a strong perceived need for autonomy, in order to improve the whole system's efficiency, reliability, and safety.

Among the huge number of functionalities that are required to develop an autonomous vehicle, three are particularly important, i.e. localisation, planning, including obstacle avoidance, and trajectory tracking. Though they are all crucial to let the vehicle complete a task, it must be noticed that the planner has the responsibility to compute a path/trajectory[1] that should not only take the vehicle to the desired location, but that must be also "feasible", i.e., compatible with the vehicle kino-dynamic constraints, and "safe". For example, in natural outdoor environments, characterised by poorly traversable terrains, slopes and a variety of obstacles, a safe path is characterised by the easiest traversable terrains, and have to minimise the vehicle roll-over risk due to the presence of slope.

The authors are with Politecnico di Milano, Dipartimento di Elettronica Informazione e Bioingegneria, Piazza L. da Vinci, 32 - 20133 Milano, Italy {matteo.ragaglia, maria.prandini, luca.bascetta}@polimi.it

[1]In this paper we use the term "planning" and "path planning", with a slight abuse of notation, to refer to either path and trajectory planning problems.

Considering the state-of-the-art in planning techniques (see Section II for further details), sampling-based algorithms represent nowadays the most widespread "practical approach" to the planning problem. The outstanding success they achieved in the last decade is due to a rather simple yet effective idea: using a collision-checking module to provide information about feasibility of candidate trajectories. In this way, a set of points sampled from the free-space is connected in order to build a graph (roadmap) of feasible trajectories that are used to construct the solution to the original planning problem.

In most practical applications, however, one may be interested in finding paths of minimum cost, and in accounting for vehicle constraints like, for example, actuator limitations. Unfortunately, moving from a standard planning problem, i.e., finding a path, to an optimal planning problem, i.e., finding a path that optimises a desired cost function, or even to an optimal planning problem with constraints, computational intractability comes at no surprise. As a matter of fact, there have been several attempts [1]–[5] to propose an algorithm to solve an optimal planning problem with vehicle constraints, and all these approaches were based either on the shooting method [6] or on linearisation of the vehicle model.

The aim of this work is to propose Poli-RRT*, which is the first RRT-based planner that takes into account vehicle constraints but it does not need to represent the vehicle dynamics with an approximate linearised model. In fact, the proposed methodology relies on an exact linearisation of the model that allows to efficiently recast the optimal control problem used to calculate the trajectories for extending the tree as a quadratic program, without any model simplification.

Poli-RRT* is formulated considering a unicycle-like vehicle and actuation constraints, but can be easily applied to any other feedback linearisable vehicle model with constraints of different type.

II. Related work

Two main approaches to address the planning problem have been developed in the literature: search-based and sampling-based techniques. A third methodology, based on the MPC technique [7], exists as well, though it is less popular.

Search-based algorithms [8], [9] look for the sequence of actions that allow the vehicle to go from a start to a goal state in an optimal way, constructing a graph of states and searching a solution into the graph.

Sampling-based algorithms exploits the idea of sampling a continuous state space, instead of considering a pre-

defined finite set of configurations, as it often occurs with search-based methods. They proved to be very effective for planning in high-dimensional spaces since, even though they are not complete, they provide probabilistic completeness: the probability that the planner returns a solution, if one exists, goes exponentially to one as the number of samples approaches infinity [10].
The most important sampling-based planners to date are Probabilistic RoadMaps (PRM) [11], [12] and Rapidly-exploring Random Trees (RRT) [13]. A brief description of the PRM approach and an overview of RRT-based algorithms is presented in the following.

PRM algorithm is a multiple-query method that first constructs a graph (the roadmap) representing a set of collision-free trajectories, and then answers queries by computing the shortest path connecting the initial state with the final state through the roadmap. PRM algorithms are probabilistically complete as the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap [12].
Multiple-query methods, however, are valuable in a static, a-priori known, highly structured environment, while in most on-line planning problems the environment is unstructured, dynamic, not known a priori or it is computationally challenging (or even infeasible) to compute a roadmap a priori. This is the main reason behind the success of single-query counterpart to PRM: RRT.

First introduced in [13], RRT is an incremental sampling-based planning algorithm developed for searching high-dimensional continuous state spaces. It represent the single-query counterpart to PRM, since the incremental nature of RRT does not require to set the number of samples a priori, and returns a solution as soon as a path from the start to the goal state is found (or alternatively as soon as the set of trajectories built by the algorithm is rich enough), enabling fast on-line implementations.
Rapidly-exploring Random Graphs (RRG), RRT* and Optimal RRTs have been introduced in [1], [14]–[18] to address the optimal path planning problem, as well as path planning problems with complex task specifications. RRG algorithm builds incrementally a connected roadmap, randomly sampling the state space. Starting from RRG, RRT* can be obtained by simply removing cycles from graph, ensuring that vertices are reached through minimal-cost paths.
Not only RRTs represent an effective solution for the planning problem in case of linear robot dynamics, but they have been shown to work effectively for systems with differential constraints and non-linear dynamics [19]–[21]. More in depth, in [1] the authors derive sufficient conditions to ensure asymptotic optimality of the RRT* algorithm for systems with differential constraints and show how to apply RRT* to a unicycle-based vehicle and to an holonomic double integrator robot. The same approach is generalised to arbitrary kino-dynamic systems in [2] by using the shooting method to connect pairs of states, thus obtaining feasible yet inherently suboptimal trajectories.
Other attempts to obtain a more sophisticated merge between RRT-based planning and optimality concerns, in the case of non-linear dynamics, have also been recently proposed. At first [22] considers the problem of planning in continuous state and action spaces with non-linear deterministic dynamics. The proposed solution combines heuristics, to bias the search, with repeated runs of the RRT algorithm (with tighter constraints on each iteration) to improve the solution quality.
In [3] the authors present an algorithm, named "Kinodynamic RRT*", that guarantees asymptotic optimality for any system characterised by linear differential constraints, in state spaces of any dimension. The same approach can be applied to non-linear dynamics as well, by using their first-order Taylor approximations.
Furthermore, in [4] the "LQR-RRT*" algorithm is proposed to solve planning problems with complicated or under-actuated dynamics, by locally linearising the system and applying linear quadratic regulation (LQR).
Finally, in [5] a new method for applying RRT* to kino-dynamic motion planning problems is introduced, using finite-horizon linear quadratic regulation (LQR) to measure cost and to extend the tree.

III. Proposed approach

In this section we describe the proposed RRT-based planning algorithm, namely Poli-RRT*, which computes a solution to the optimal constrained planning problem, for a unicycle-like vehicle moving in a 2-dimensional Euclidean space, without linearising/approximating the nonlinear system dynamics, while obeying to actuation constraints and avoiding collisions with a-priori known static obstacles.

The following model is adopted for describing the vehicle dynamics

$$\begin{cases} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \\ \dot{v} &= a \end{cases} \tag{1}$$

The state q of the system includes the vehicle pose (x,y,θ) and linear velocity v, i.e., $q=[x,y,\theta,v]^T$, and is confined to some bounded set $Q=[x_{min},\,x_{max}]\times[y_{min},\,y_{max}]\times[\theta_{min},\,\theta_{max}]\times[v_{min},\,v_{max}]$. A value $q\in Q$ is called *node* according to the RRT terminology.
The control input is given by the linear acceleration a and the angular velocity ω, which are subject to the following constraints

$$a\in A=[a_{min},a_{max}]\,,\ \ \omega\in\Omega=[\omega_{min},\omega_{max}]\,. \tag{2}$$

Given two nodes q and q', we let $e=(q,q')$ identify the optimal trajectory that connects q to q' while minimising some cost function and satisfying the actuation constraints (2). The cost associated with e is denoted by $C(e)$. According to the RRT terminology e is called an *edge*.

System (1) is initialised at a certain node $q_{start}=[x_{start},y_{start},\theta_{start},v_{start}]^T\in Q$ and has to be steered to some goal set $Q_{goal}\subset Q$, while avoiding obstacles and following

an optimal trajectory. The subset of the state space that is free of obstacles is denoted by Q_{free}.

Poli-RRT* first builds a tree $T = (Q_T, E_T)$, where $Q_T \subset Q_{free}$ is a set of nodes with cardinality N (defined by the user), and E_T is a set of collision-free edges connecting nodes in Q_T. Among all sequences of nodes $q_0, q_1, q_2, \ldots, q_m$ with length $2 \leq m \leq N$, satisfying $q_i \in Q_T$, $i = 0, 1, \ldots, m$, $q_0 = q_{start}$, $q_m \in Q_{goal}$, and $e_i = (q_i, q_{i+1}) \in E_T$, $i = 0, 2, \ldots, m-1$, Poli-RRT* then chooses the one with minimal overall cost $C(\rightarrow q_m) = \sum_{i=0}^{m-1} C(e_i)$, which is a non-decreasing cost over the entire node sequence.

A key ingredient of Poli-RRT* algorithm is represented by the procedure adopted to build the edge $e_i = (q_i, q_{i+1})$, i.e., an optimal trajectory that connects node q_i to node q_{i+1}, while satisfying the actuation constraints (2), and its cost $C(e_i)$. This procedure rests on a two-step approach combining optimal control with a receding horizon strategy and is explained in Section III-A. The description of Poli-RRT* algorithm is postponed to Section III-B.

A. Two-step approach to edge design

By applying to (1) the feedback linearisation strategy [23]

$$\begin{bmatrix} a \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\dfrac{\sin(\theta)}{v} & \dfrac{\cos(\theta)}{v} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{3}$$

we obtain the following double integrator linear model

$$\begin{cases} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v}_x &= u_1 \\ \dot{v}_y &= u_2 \end{cases} \tag{4}$$

where we set $v_x = v\cos(\theta)$ and $v_y = v\sin(\theta)$. In turn, constraints (2) can be rewritten as

$$\begin{aligned} a_{min} &\leq \cos(\theta)u_1 + \sin(\theta)u_2 \leq a_{max} \\ \omega_{min} &\leq -\frac{\sin(\theta)}{v}\, u_1 + \frac{\cos(\theta)}{v}\, u_2 \leq \omega_{max} \end{aligned} \tag{5}$$

1) Optimal control without accounting for actuation constraints: Given that the feedback linearised system (4) is a double integrator with state $s = [x, y, v\cos(\theta), v\sin(\theta)]^T$, we can apply the approach in [3] to obtain the minimum time trajectory connecting two states $q_i = [x_i, y_i, \theta_i, v_i]^T$ and $q_{i+1} = [x_{i+1}, y_{i+1}, \theta_{i+1}, v_{i+1}]^T$.

The idea is to first fix the final time $\tau > 0$ and determine the input $u = [u_1, u_2]^T : [0, \tau] \rightarrow \Re^2$ for (4) that optimises the finite horizon cost

$$J_\tau(u) = \int_0^\tau \left(1 + u(t)^T R u(t)\right) dt \tag{6}$$

with $R = R^T \in \Re^{2\times 2}$ positive definite, subject to the constraints $s(0) = s_i = [x_i, y_i, v_i\cos(\theta_i), v_i\sin(\theta_i)]^T$ on the initial state and $s(\tau) = s_{i+1} = [x_{i+1}, y_{i+1}, v_{i+1}\cos(\theta_{i+1}), v_{i+1}\sin(\theta_{i+1})]^T$ on the final state. Let $J_\tau^\star$ be the resulting optimal cost, i.e.,

$$J_\tau^\star = \min_{u:[0,\tau]\rightarrow\Re^2} \int_0^\tau \left(1 + u(t)^T R u(t)\right) dt.$$

Then, the optimal value $\tau^\star$ for τ can be obtained by minimising $J_\tau^\star$. Correspondingly, one can obtain the optimal input $u^\star$ and state $s^\star = [x^\star, y^\star, v_x^\star, v_y^\star]^T$, which, in turn, maps back to the optimal $q^\star = [x^\star, y^\star, \theta^\star, v^\star]^T$. In particular, from [3], we have the following analytic expression for $J_\tau^\star$:

$$J_\tau^\star = \tau + \left(s_{i+1} - \exp(At)s_i\right)^T G(\tau) \left(s_{i+1} - \exp(At)\, s_i\right), \tag{7}$$

where $G(\tau)$ is the weighted controllability Gramian

$$G(\tau) = \int_0^\tau \exp\left(A(\tau - t)\right) B R^{-1} B^T \exp\left(A^T(\tau - t)\right) dt.$$

Once $\tau^\star = \text{argmin}_{\tau>0} J_\tau^\star$ is determined, then, $u^\star(t)$ and $s^\star(t)$, $t \in [0, \tau^\star]$, are given by:

$$\begin{aligned} u^\star(t) &= R^{-1} B^T \exp\left(A^T(\tau^\star - t)\right) d^\star \\ s^\star(t) &= [I_4\ 0_4] \exp\left(\begin{bmatrix} A & BR^{-1}B \\ 0_4 & -A^T \end{bmatrix} (t - \tau^\star) \right) \begin{bmatrix} s_{i+1} \\ d^\star \end{bmatrix} \end{aligned}$$

where I_4 is the 4×4 identity matrix and 0_4 is the 4×4 zero matrix and we set $d^\star = G(\tau^\star)^{-1}\left(s_{i+1} - \exp(A\tau^\star)\, s_i\right)$.

Let $q^\star(t)$, $t \in [0, \tau^\star]$, be the optimal trajectory $s^\star(t)$, $t \in [0, \tau^\star]$, mapped back to the original state coordinates of the unicycle model (1). If the actuation constraints (5) are satisfied by $u^\star(t)$ and $q^\star(t)$, for all $t \in [0, \tau^\star]$, then, the edge $e_i = (q_i, q_{i+1})$ is given by the optimal trajectory $q^\star$ that takes the system from node q_i to node q_{i+1} within a time interval of length $\tau^\star$ with a cost $C(e_i) = J_{\tau^\star}^\star = J_{\tau^\star}(u^\star)$. If that is not the case, a receding horizon strategy is put in place so as to force the constraints (5) to hold, while keeping the resulting trajectory close to the optimal one along a time horizon of length $\tau^\star$. This is explained in the following.

2) Receding horizon with actuation constraints: To simplify computation, we introduce a discrete time version of the feedback linearised system (4):

$$\begin{cases} x(k+1) = x(k) + v_x(k)\Delta \\ y(k+1) = y(k) + v_y(k)\Delta \\ v_x(k+1) = v_x(k) + u_1(k)\Delta \\ v_y(k+1) = v_y(k) + u_2(k)\Delta \end{cases} \tag{8}$$

where $\Delta > 0$ is the sample time interval. By setting $s(k) = [x(k), y(k), v_x(k), v_y(k)]^T$ and $u(k) = [u_1(k), u_2(k)]^T$, system (8) can be rewritten in the compact form

$$s(k+1) = Fs(k) + Hu(k)$$

where F and H are appropriately defined matrices.

Let $k_f := \lfloor \frac{\tau^\star}{\Delta} \rfloor$ be the discrete time version of the final time $\tau^\star$ as computed in Section III-A.1.
Our goal is to choose $u(0), u(1), \ldots, u(k_f - 1)$ so as to keep $s(k)$ close to the optimal constrained-free trajectory $s^\star(k\Delta)$ computed in Section III-A.1, while forcing the constraints (5) to hold at each discrete time instant k along the discrete time horizon $[0, k_f]$.

To this purpose, we introduce the following constrained optimisation problem $C-OPT(k)$:

$$\min_{u(k),u(k+1),\ldots,u(k_f-1)} \sum_{h=k+1}^{k_f} (s(h)-\bar{s}_h)^T W_h (s(h)-\bar{s}_h) \quad (9)$$

subject to:

$$s(h+1) = Fs(h)+Hu(h)$$
$$s(k) = \bar{s}_k$$
$$a_{min} \leq [\cos(\bar{\theta}_h)\ \ \sin(\bar{\theta}_h)]u(h) \leq a_{max}$$
$$\omega_{min} \leq [-\frac{\sin(\bar{\theta}_h)}{\bar{v}_h}\ \ \frac{\cos(\bar{\theta}_h)}{\bar{v}_h}]u(h) \leq \omega_{max}$$
$$\forall h = k,\ldots,k_f-1,$$

with W_h, $h = k+1,k+2,\ldots,k_f$, positive definite matrices of dimension 4. Once $\bar{s}_h$, $h = k,k+1,\ldots k_f$, and $\bar{\theta}_h$ and $\bar{v}_h$, $h = k,k+1,\ldots k_f-1$, are given, $C-OPT(k)$ is an easy to solve convex optimisation problem with quadratic cost ($s(h)$ is a linear function of the optimisation variables) and linear constraints.

Let $u^{\star,k}(h)$, $h = k,\ldots k_f-1$, be the solution $C-OPT(k)$, and $s^{\star,k}(h)$, $h = k,\ldots k_f$, the trajectory obtained by applying $u^{\star,k}$ to (8), starting from $\bar{s}_k$ at time k.

The receding horizon strategy then unfolds as follows. We initially solve the $C-OPT(k)$ for $k = 0$ by setting

$$\bar{s}_h = s^\star(h\Delta), h = 0,\ldots,k_f$$
$$\bar{\theta}_h = \theta^\star(h\Delta), h = 0,\ldots,k_f-1$$
$$\bar{v}_h = v^\star(h\Delta), h = 0,\ldots,k_f-1$$

where $s^\star$ (and, correspondingly, $\theta^\star$ and $v^\star$) is the optimal constrained-free trajectory computed in Section III-A.1. The obtained solution $u^{\star,k}$, $k = 0$, will necessarily satisfy the constraints (5) at time k since the trajectory $s^{\star,k}$ satisfies $s^{\star,k}(k) = \bar{s}_k$ and constraints (5) are imposed in the $C-OPT(k)$ by using $\bar{\theta}_h$ and $\bar{v}_h$, $h = k,k+1,\ldots k_f-1$, derived from $\bar{s}_h$, $h = k,k+1,\ldots k_f$. However, the solution $u^{\star,k}$ will not necessarily satisfy the constraints at time $k+1,k+2,\ldots,k_f-1$. This is because, in general $s^{\star,k}(h)$ will be different from $\bar{s}_h$ for $h = k+1,k+2,\ldots,k_f-1$.

If $u^{\star,k}$, $k = 0$, does satisfy the constraints at time $k+1,k+2,\ldots,k_f-1$, then, we have a feasible solution and we can define the edge e_i as given by the optimal constrained trajectory $q^{\star,k}$ that takes the system from node q_i to a node close to q_{i+1} (the larger is the weighting matrix W_{k_f} the closes the system will get to q_{i+1}) within a time interval of length $\tau^\star$ with a cost $C(e_i) = J_{k_f\Delta}(u^{\star,k})$ obtained by plugging into (6) a piecewise constant version of $u^{\star,k}$.

If, instead, $u^{\star,k}$, $k = 0$, does satisfy the constraints at some time $h \in [k+1,k+2,\ldots,k_f-1]$, then, we do not have a feasible solution yet. We hence apply only the first input sample $u^{\star,k}(k)$, and determine the further input values by solving the $C-OPT(k)$ for $k = 1$ with

$$\bar{s}_h = s^{\star,k-1}(h\Delta), h = k,\ldots,k_f$$
$$\bar{\theta}_h = \theta^{\star,k-1}(h\Delta), h = k,\ldots,k_f-1 \quad (10)$$
$$\bar{v}_h = v^{\star,k-1}(h\Delta), h = k,\ldots,k_f-1.$$

Again if $u^{\star,k}$, $k = 1$, does satisfy the constraints at all times $k+1,k+2,\ldots,k_f-1$, then, we have a feasible solution and we can define the edge e_i as the optimal constrained trajectory $q^{\star,k}$ that takes the system from node q_i to a node close to q_{i+1} through $u(0) = u^{\star,0}(0)$, $u(h) = u^{\star,1}(h)$, $h = 1,2,\ldots,k_f-1$. The cost associated to e_i will be given by $C(e_i) = J_{k_f\Delta}\left([u^{\star,0}(0),u^{\star,1}(1),\ldots,u^{\star,1}(k_f-1)]^T\right)$.

If, instead, $u^{\star,k}$, $k = 1$, does satisfy the constraints at some time $h \in [k+1,k+2,\ldots,k_f-1]$, then, we apply only the input sample $u^{\star,k}(k)$, increment k to $k = 2$, and determine the further input values by solving the $C-OPT(k)$ with $\bar{s}_h$, $h = k,k+1,\ldots k_f$, and $\bar{\theta}_h$ and $\bar{v}_h$, $h = k,k+1,\ldots k_f-1$, defined in (10).

This kind of reasoning is repeated until a feasible solution is found at some iteration $k < k_f-1$ or $k = k_f-1$ is reached.

B. Poli-RRT* Algorithm

The algorithm Poli-RRT* works as follows:

1) **tree initialisation**: an empty tree $T = (Q_T,E_T)$ is instantiated, where Q_T is the set of nodes and E_T is the set of edges/trajectories connecting the nodes. Then, node $q_0 = q_{start}$ is added to T: $Q_T = \{q_{start}\}$;
2) **random sampling**: a state configuration q_{rand} is randomly sampled within the free portion of the state space Q_{free} according to an uniform distribution;
3) **neighbour radius computation**: the neighbour radius is computed as the value r such that the set

$$\mathscr{R}(q_{rand},r) = \{\, q \in Q_T \mid e_1 = (q,q_{rand}) \implies C(e_1) < r \vee e_2 = (q_{rand},q) \implies C(e_2) < r\}$$

contains a d-dimensional ball of radius $\gamma_{ball} = \gamma_{RRT*}(\log|Q_T|\ /|Q_T|)^{1/d}$ centred in q_{rand}, where $d = 4$ is the state space dimension and $\gamma_{RRT*} > 2(1+1/d)^{1/d}(\mu(Q_{free})/\eta_d)^{1/d}$, with $\mu(Q_{free})$ and η_d the volume of Q_{free} and of the unit ball in the d-dimensional Euclidean space, respectively (see [1], [14] for further details). By defining Q_{reach} as the subset of Q_T containing all the nodes that are inside a 4-dimensional ball of radius γ_{ball}, centered in q_{rand}:

$$Q_{reach} = \{q \in Q_T \mid ||q_{rand}-q||_2 \leq \gamma_{ball}\},$$

it is possible to determine r as

$$r = \operatorname*{argmax}_{q \in Q_{reach}} \left(\max\left\{C(e_{q,1}),C(e_{q,2})\right\}\right),$$

where we set $e_{q,1} = (q,q_{rand})$ and $e_{q,2} = (q_{rand},q)$.
4) **minimum-cost path selection**: in order to connect q_{rand} to the tree along a minimum-cost path the following steps are taken:
   - the minimum cost collision-free edge $e_{min} = (q_{min},q_{rand})$ among the edges $e = (q,q_{rand})$, $q \in Q_T$, is determined by computing

$$q_{min} = \operatorname*{argmin}_{q \in \{q \in Q_T\ |C(e) \leq r\ \wedge CollisionFree(e)\}} (C(\to q)+C(e)),$$

where r is the neighbour radius, $CollisionFree(e)$ is a function that returns true when e is a collision-free trajectory, false otherwise, and $C(\rightarrow q)$ represents the total cost of the current-best path going from q_{start} to q;

- q_{rand} and e_{min} are added to T:

$$Q_T = Q_T \cup \{q_{rand}\}, \quad E_T = E_T \cup \{e_{min}\}$$

5) **tree rewiring**: every time a new node q_{rand} is added to the tree it is necessary to check if there exist a minimum-cost path reaching any other node inside the tree and passing through q_{rand}. More in detail, for every node $q \in Q_T$, if $e = (q_{rand}, q)$ satisfies $CollisionFree(e) = true$, $C(e) \leq r$, and $C(\rightarrow q_{rand}) + C(e) < C(\rightarrow q)$, then, the tree is rewired:

$$E_T = \left\{E_T \setminus \{e_{prev}\}\right\} \cup \{e\},$$

where r is the neighbour radius, $e_{prev} = (Parent(q), q)$ is the previous edge connecting q to the tree and $Parent(q)$ is a function returning the node from which edge e_{prev} starts.

6) **termination**: the algorithm keeps iterating steps 2), 3), 4) and 5) until $|Q_T| = N$ and then it stops;

7) **optimal trajectory**: if the goal area has been reached, the minimum cost-to-go node inside Q_{goal} is selected:

$$Q_{goal} \cap Q_T \neq \emptyset \implies q_{goal} = \underset{q \in (Q_{goal} \cap Q_T)}{\operatorname{argmin}} \; C(\rightarrow q)$$

and the trajectory $e_{goal} = (q_{start}, q_{goal})$ connecting q_{start} with q_{goal} is returned along with the entire tree T.

Remark Note that at steps 4 and 5 in Poli-RRT* algorithm, condition $C(e) < r$ needs to be verified on an edge $e = (q, q')$ involving some node that is candidate for further processing. Violation of such a condition is easy to evaluate since it holds that $C(e) \geq J^{\star}_{\tau^{\star}} = \min_{\tau \geq 0} J^{\star}_{\tau}$, where $J^{\star}_{\tau}$ is the cost of an unconstrained optimal trajectory driving system (1) from q to q' in an interval of length τ and is given in analytic form in (7). Thus, the adoption of the neighbour radius r introduced at step 3 is effective in reducing the number of nodes to be processed at each iteration of the algorithm. Furthermore, it does not modify the outcome of the algorithm in terms of the computed optimal trajectory, [1], [14].

IV. Simulation Results

A MATLAB© implementation of the proposed algorithm was developed and tested, considering the dynamic model (1), with the state bounded as follows: $x \in [0, 100]$, $y \in [0, 100]$, $\theta \in [-\pi, +\pi]$, and $v \in [0, 1]$. In order to solve the constraint-free optimal control problem (see Section III-A.1), a matrix $R = 10 I_2$ equally penalising linear acceleration a and angular velocity ω was chosen. As for the receding horizon strategy in Section III-A.2, eventually refining the solution to the constraint-free optimal control problem so as to impose the saturation constraints, a sample time interval $\Delta = 0.1s$ was selected. Furthermore, matrices W_h, $h = 1, \ldots, k_f$, in the $C - OPT(k)$ problem (9) were set to

$$W_h = 10 I_4, \; h = 1, \ldots, k_f - 1, \; W_{k_f} = 100 I_4,$$

so as to assign a stronger weight to the tracking error at the last time instant k_f with respect to that at the previous time instants $h = 1, \ldots, k_f - 1$.

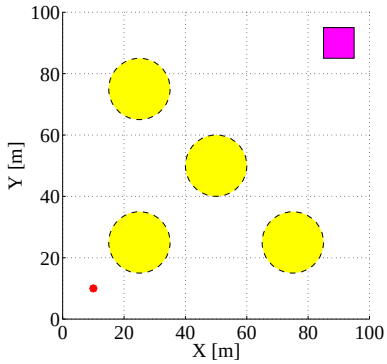

Fig. 1. Virtual environment: start pose (red dot), goal area (magenta square) and obstacles (yellow circles).

The virtual environment used as a test-bed is depicted in Figure 1, and it is composed of a start configuration, a goal area, and four non-overlapping circular obstacles.

The algorithm has been tested considering different values of the maximum tree cardinality N, i.e., 200, 250, 300, 400, 500, 600, 750, 1000, 1500, 2000, 2500, 3000 and 5000 nodes. For each of these values, several runs were realised imposing two different actuation constraints (see equation (2)): the former characterised by larger bounds

$$a \in A_1 = [-0.50, +0.50], \quad \omega \in \Omega_1 = [-0.50, +0.50]$$

the latter by more strict limits

$$a \in A_2 = [-0.20, +0.20], \quad \omega \in \Omega_2 = [-0.20, +0.20]$$

Figure 2 shows the trees obtained considering different maximum cardinality values and imposing actuation constraints A_1 and Ω_1.

Figures 3(a) and 3(b) show the actuation profiles (linear acceleration a and angular velocity ω) corresponding to the optimal trajectory found with maximum tree cardinality N set to 1000 nodes and imposing the actuation limits A_1 and Ω_1. The pictures clearly show that the values of the control variables never exceed the upper and lower bounds. On the other hand, Figures 3(c) and 3(d) show the actuation profiles corresponding to a different optimal trajectory obtained considering the same maximum tree cardinality, but A_2 and Ω_2 as actuation limits. Note that in this case, the actuation profiles partially saturate the upper/lower bounds, but they still never exceed them.

Finally, Figure 4 refers to the saturation limits A_1 and Ω_1, and demonstrates that, when more nodes are added to the tree up to the maximum tree cardinality of 3000 nodes, the cost of the optimal solution decreases and asymptotically

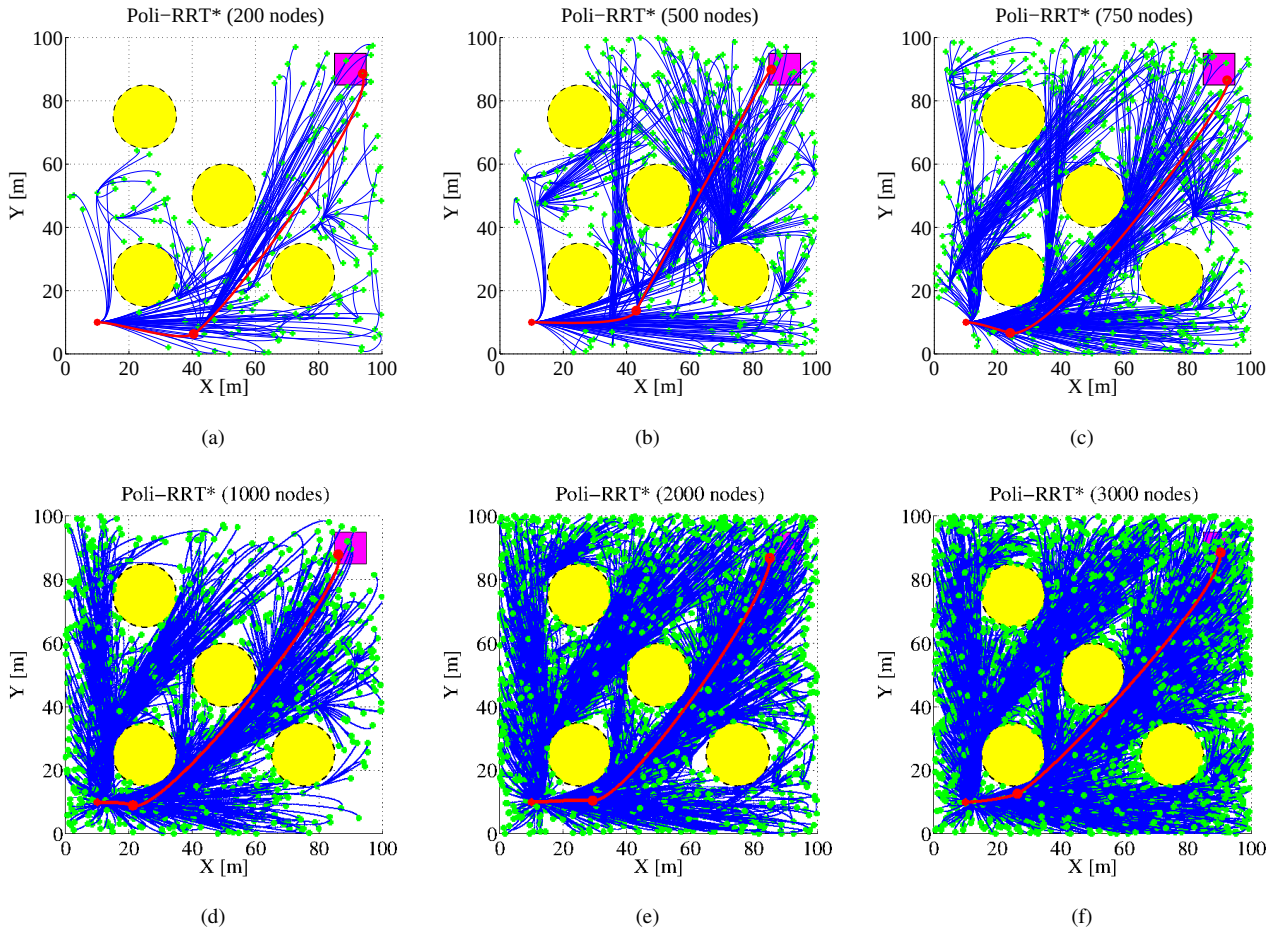

Fig. 2. Planning experiments: tree nodes (green crosses), tree edges (solid blue), optimal solution (solid red) and optimal trajectory nodes (red circles). Maximum tree cardinality set to 200 nodes 2(a), 500 nodes 2(b), 750 nodes 2(c), 1000 nodes 2(d), 2000 nodes 2(e) and 3000 nodes 2(f).

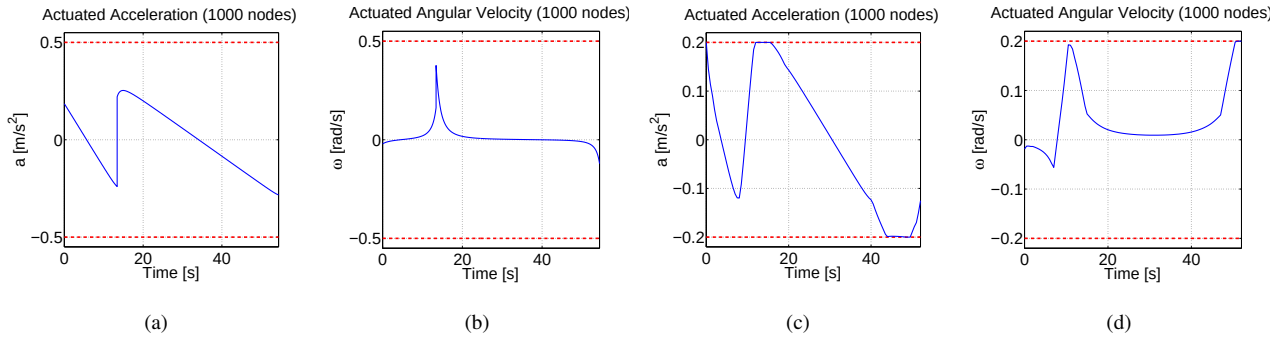

Fig. 3. Actuation profiles of the optimal solution with $N = 1000$ (solid blue) and actuation limits (dashed red) posed by A_1 and Ω_1 (plots (a) and (b)), and A_2 and Ω_2 (plots (c) and (d)).

approaches the optimum value. On the other hand, Figure 5 shows the average cost of the optimal solution over multiple runs performed with the same maximum cardinality. As expected, not only the average cost decreases with respect to the maximum tree cardinality, but a better average cost is achieved while considering as actuation limits A_1 and Ω_1 (blue line) rather than the more restrictive A_2 and Ω_2 sets (red line).

V. Conclusions

This paper proposes the first RRT-based planner able to compute an optimal and dynamically feasible trajectory (taking into account both differential and actuation constraints), without representing the system dynamics with an approximate linearised model, but relying on an exact linearisation. This approach was used to generate feasible and optimal trajectories for a unicycle-like vehicle moving in a virtual environment populated by obstacles.

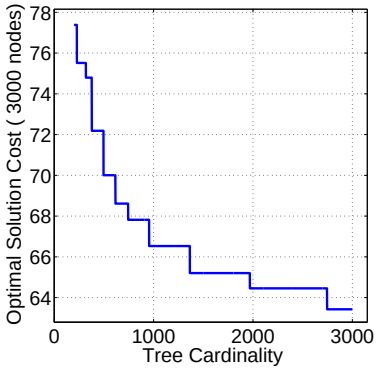

Fig. 4. Cost of the optimal solution as a function of the tree cardinality up to a maximum value $N = 3000$ (actuation limits A_1 and Ω_1).

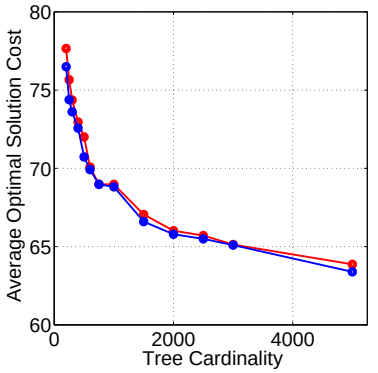

Fig. 5. Average cost of the optimal solution as a function of the tree cardinality considering as actuation limits A_1 and Ω_1 (blue line), and A_2 and Ω_2 (red line).

Regarding future developments, an extension to hybrid dynamic models will be considered. Furthermore, in order to enable real-time computation, an efficient implementation of the proposed planning strategy will be realised, possibly adopting an heuristic that enables a branch-and-bound approach so as to avoid attempts to connect nodes that cannot contribute to an optimal solution.

REFERENCES

[1] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010.

[2] J. hwan Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 3276–3282.

[3] D. J. Webb and J. V. D. Berg, "Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints," *CoRR*, vol. abs/1205.5088, 2012.

[4] 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 *IEEE International Conference on Robotics and Automation*, May 2012, pp. 2537–2542.

[5] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *IEEE Conference on Robotics and Automation*, 2013.

[6] R. Burden and D. Faires, *Numerical Analysis*. Brooks/Cole, 2001.

[7] A. Tahirovic and G. Magnani, "General framework for mobile robot navigation using passivity-based MPC," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 184–190, 2011.

[8] M. Pivtoraiko, R. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, pp. 308–333, 2009.

[9] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotic Research*, vol. 28, pp. 933–945, 2009.

[10] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 759–774, 1997.

[11] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] E. Kavraki, M. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

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

[15] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Real-time motion planning using the RRT*," in *IEEE Conference on Robotics and Automation*, April 2011.

[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[17] A. Perez, S. Karaman, M. Walter, A. Shkolnik, E. Frazzoli, and S. Teller, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[18] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning with deterministic μ-calculus specifications," in *American Control Conference*, 2012.

[19] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," in *American Control Conference*, vol. 1, 2001, pp. 43–49.

[20] M. Branicky, M. Curtiss, J. Levine, and S. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in *IEEE Conference on Decision and Control*, vol. 1, 2003, pp. 657–663.

[21] ——, "Sampling-based planning, control and verification of hybrid systems," *IEE Proceedings Control Theory and Applications*, vol. 153, no. 5, pp. 575–590, 2006.

[22] Y. Abbasi-Yadkori, J. Modayil, and C. Szepesvari, "Extending rapidly-exploring random trees for asymptotically optimal anytime motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 127–132.

[23] D. Stipanovic, G. Inalhan, R. Teo, and C. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," *Automatica*, vol. 40, no. 8, pp. 1285–1296, 2004.