Multi-Agent Poli-RRT*

Optimal constrained RRT-based Planning for Multiple Vehicles with Feedback Linearisable Dynamics

Matteo Ragaglia, Maria Prandini, and Luca Bascetta

Politecnico di Milano - Piazza Leonardo da Vinci, 32 - 20133, Milan, Italy
{matteo.ragaglia, maria.prandini, luca.bascetta}@polimi.it

Abstract. Planning a trajectory that is optimal according to some performance criterion, collision-free, and feasible with respect to dynamic and actuation constraints is a key functionality of an autonomous vehicle. Poli-RRT* is a sample-based planning algorithm that serves this purpose for a single vehicle with feedback linearisable dynamics. This paper extends Poli-RRT* to a multi-agent cooperative setting where multiple vehicles share the same environment and need to avoid each other besides some static obstacles.

Keywords: Planning, Optimal Control, Feasability, Safety, Autonomous vehicles, Multi-Agent Systems

1 Introduction

In the past few years the interest towards autonomous unmanned vehicles is considerably increased, mainly because 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.
Among the functionalities that make a vehicle autonomous, planning plays a crucial role. As a matter of fact, the planner not only has the responsibility to deliver a trajectory that takes the vehicle to the desired goal but it also needs to guarantee both "feasibility" (with respect to possibly nonlinear dynamics, kinodynamic and actuation constraints) and "safety" (in terms of avoiding obstacles and dangerous kinematic configurations, i.e., vehicle roll-over) of the computed trajectory. Moreover, in most practical applications, it is typically required to find trajectories that are optimal according to some cost metric.
Unfortunately, given a complex system characterized by nonlinear dynamics, moving from a standard planning problem to an optimal planning problem, or even to a constrained and optimal one, computational intractability comes at no surprise. As a consequence, sample-based planning algorithms have emerged as an appealing alternative to search-based planning techniques [1, 2] and model predictive control approaches [3].

1.1 State of the art

The success that sample-based planners like Rapidly-exploring Random Trees (RRT) [4] have achieved in the last fifteen years is due to a rather simple yet effective idea: a set of points is sampled from the free-space and connected in order

to build a tree (roadmap) of feasible trajectories, that are then used to determine the solution to the planning problem. Interestingly, probabilistic completeness has been shown for this approach in [5].
RRT-based planning algorithms [4] have been originally introduced to solve the trajectory planning problem for holonomic robots. Then, they have been extended to optimal and constrained trajectory planning. Several solutions have been proposed in the latest few years, including optimal/non-linear RRTs [6, 7], Rapidly-exploring Random Graphs (RRG) and RRT* [8–13]. A generalization of the RRT* planning approach to arbitrary kino-dynamic systems is presented in [14], where the shooting method [15] is used to connect pairs of nodes, thus obtaining feasible yet inherently suboptimal trajectories.
In [16] the authors present an algorithm, named “Kinodynamic RRT*”, that guarantees asymptotic optimality for systems characterised by linear differential constraints. The same approach can be applied to non-linear dynamics as well, by using their first-order Taylor approximations. Furthermore, in [17] the “LQR-RRT*” algorithm is proposed to solve planning problems with complex or underactuated dynamics, by locally linearising the system and applying linear quadratic regulation (LQR), while in [18] a new method for applying RRT* to kino-dynamic motion planning problems is introduced, using a finite-horizon quadratic criterion to assess the cost and extend the tree.
Finally, [19] introduces the “Poli-RRT*” algorithm, which is the first RRT-based planner that takes into account vehicle constraints without either representing the vehicle dynamics with an approximate linearised model or using the shooting method. In fact, the proposed methodology relies on an exact linearisation of the model, that allows to efficiently recast the optimal control problem used to extend the tree into a quadratic program, without any model simplification.
The main contribution of this work to the field of planning for autonomous vehicles consists in extending Poli-RRT* to a multi-agent framework.

1.2 Paper structure

The remainder of this paper is organized as follows. Section 2 briefly describes the original Poli-RRT* algorithm, while the extension to multi-agent systems is detailed in Section 3. Section 4 presents simulation results. Finally, some concluding remarks are drawn in Section 5.

2 Background on Poli-RRT* for a single agent

The Poli-RRT* algorithm computes a solution to the optimal constrained planning problem for a vehicle with feedback linearisable dynamics

$$\dot{x} = f(x, u)$$

subject to constraints on the actuation input $u \in \Omega$, while accounting also for collision avoidance in presence of static obstacles.
Given the initial state x_{start} and the set of goal states X_{goal} within the obstacle-free set X_{free}, Poli-RRT* builds a tree $T = (X_T, E_T)$, where $X_T \subset X_{free}$ are the nodes,

and E_T are the edges that correspond to collision-free trajectories connecting two nodes in X_T. Among all sequences of m nodes $x_0, x_1, x_2, \ldots, x_m$, satisfying

$$\begin{aligned}
&x_i \in X_T,\ i = 0, 1, \ldots, m,\\
&x_0 = x_{start}\\
&x_m \in X_{goal}\\
&e_i = (x_i, x_{i+1}) \in E_T,\ i = 0, 2, \ldots, m-1
\end{aligned}$$

Poli-RRT* chooses the one with minimal overall cost:

$$C(\rightarrow x_m) = \sum_{i=0}^{m-1} C(e_i) \tag{1}$$

with the cost per edge satisfying $C(e) \geq 0$ for any edge e.

2.1 Edge calculation procedure

Given two nodes x and x', the edge $e = (x, x')$ represents the optimal trajectory that connects x to x' while minimising the cost function $C(e)$ and satisfying actuation constraints. The edge calculation procedure relies on a two-step approach that combines optimal control and a receding horizon strategy.
By applying feedback linearisation we can express the original system together with the input constraints in the new state coordinates s and in the new input v as follows:

$$\begin{aligned}
&\dot{s} = As + Bv\\
&h\left(g\left(s\right), v\right) \in \Omega
\end{aligned}$$

where

$$\begin{aligned}
x &= g\left(s\right)\\
u &= h\left(x, v\right)
\end{aligned}$$

According to the approach proposed in [16], it is possible to compute a minimum-time optimal trajectory connecting two states s_i and s_{i+1} with respect to the cost metric

$$J_\tau(v) = \int_0^\tau \left(1 + v(t)^T R v(t)\right) dt$$

$R > 0$ being an input weight matrix, subject to

$$\begin{aligned}
s\left(0\right) &= s_i\\
s\left(\tau\right) &= s_{i+1}
\end{aligned}$$

More in details, for each given final time $\tau > 0$, the optimal control input v and the corresponding cost $J^\star_\tau$ is computed analytically and the minimum time $\tau^\star$ is determined as

$$\tau^\star = \underset{\tau > 0}{\text{argmin}}\ J^\star_\tau$$

Given $\tau^\star$, the minimum-time optimal input and state variables $v^\star(t)$ and $s^\star(t)$, $t \in [0, \tau^\star]$, are given by:

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

where n is the dimension of the state s, I_n is the $n \times n$ identity matrix, 0_n is the $n \times n$ zero matrix and we set

$$
d^\star = G(\tau^\star)^{-1}\left(s_{i+1} - e^{A\tau^\star} s_i\right)
$$

with $G(\tau)$ equal to the weighted controllability Gramian of the system.
Let

$$
\begin{aligned}
x^\star(t) &= g(s^\star(t)), \;\; t \in [0, \tau^\star] \\
u^\star(t) &= h(x^\star(t), v^\star(t)), \;\; t \in [0, \tau^\star]
\end{aligned}
$$

be the optimal trajectory in the original state and input variables. If the input constraints are satisfied by $u^\star(t)$, for all $t \in [0, \tau^\star]$, then, the edge e is set equal to

$$
e = (x^\star(0), x^\star(\tau^\star))
$$

with the understanding that the corresponding trajectory is given by $u^\star(t)$ and $x^\star(t)$, with an associated cost

$$
C(e_i) = J_{\tau^\star}(u^\star)
$$

If that is not the case, a receding horizon strategy is put in place so as to enforce the constraints, while keeping the resulting trajectory close to the optimal unconstrained one (see [19] for the details).

2.2 Poli-RRT* algorithm

The steps of the Poli-RRT* algorithm are given by:

1. **tree initialisation**: an empty tree is initialised, setting

$$
\begin{aligned}
X_T &= \{x_{start}\} \\
E_T &= \emptyset
\end{aligned}
$$

2. **random sampling**: a state configuration x_{rand} is randomly sampled within X_{free} according to a uniform distribution;
3. **neighbour radius computation**: set

$$
r = \underset{x \in X_{reach}}{\operatorname{argmax}} \left(\max\{C(e_1), C(e_2)\}\right)
$$

where $e_1 = (x_{rand}, x)$, $e_2 = (x, x_{rand})$ and

$$
X_{reach} = \{x \in X_T \mid ||x_{rand} - x||_2 \leq \gamma_{ball}\}
$$

γ_{ball} being computed according to [10];

4. **minimum-cost trajectory selection**: in order to connect x_{rand} to the tree, a minimum-cost trajectory is determined as $e_{min} = (x_{min}, x_{rand})$ where

$$x_{min} = \underset{x \in \{x \in X_T \; | C(e) \leq r \wedge CFree(e)\}}{\operatorname{argmin}} (C(\to x) + C(e))$$

where r is the neighbour radius, $CFree(e)$ is a function that returns true when e is a collision-free trajectory, false otherwise, and $C(\to x)$ represents the cost of the current-best trajectory going from x_{start} to x. Then

$$X_T = X_T \cup \{x_{rand}\}$$
$$E_T = E_T \cup \{e_{min}\}$$

5. **tree rewiring**: whenever a node x_{rand} is added to the tree, in order to ensure trajectory optimality, it is necessary to check the existence of minimum-cost trajectories starting from x_{start}, passing through x_{rand} and reaching any other node within the neighbour radius r of x_{rand}. In other words, for every node $x \in X_T$, if $e = (x_{rand}, x)$ satisfies

$$CFree(e) = true, \quad C(e) \leq r, \quad C(\to x_{rand}) + C(e) < C(\to x)$$

the tree is rewired by setting

$$E_T = \big\{E_T \setminus \{e_{prev}\}\big\} \cup \{e\}$$

where e_{prev} is the edge that was previously connecting x to the tree and that is replaced by e;
6. **termination**: the algorithm iterates steps 2), 3), 4) and 5) until $|X_T| = N$, where N is a given maximum cardinality for X_T;
7. **optimal trajectory**: if the goal area has been reached, the minimum cost-to-go node inside X_{goal} is selected and the trajectory connecting x_{start} with x_{goal} is returned along with the entire tree T:

$$X_{goal} \cap X_T \neq \emptyset \implies x_{goal} = \underset{x \in (X_{goal} \cap X_T)}{\operatorname{argmin}} C(\to x).$$

3 Extension to multi-agent systems

The main contribution of this work is the extension of the Poli-RRT* algorithm to a multi-agent setting, by adopting a priority-based approach.
All the K agents are ranked according to a priority criterion and the algorithm plans trajectories in sequence, starting from the highest-priority agent A_1 and moving to the lowest-priority one A_K, each time considering the trajectories that have already been designed as obstacles to avoid.
A pseudo-code version of the procedure is given in Algorithm 1, whose parameters are:

- N, the maximum tree cardinality;
- $AgentsSet$, the set of agents;
- $ObstaclesSet$, the set of obstacles (that initially contains only the static ones);
- $safeDist$, the minimum safe distance that the algorithm must always ensure between each couple of agents.

Clearly, once the i-th iteration of the algorithm is completed, the list of obstacles must be updated in order to keep track of the newly designed trajectory $Traj_i$ for agent i. At each iteration of the multi-agent algorithm, Poli-RRT* is run for a specific agent A_i using an updated list of obstacles that account for previously designed trajectories. Within Poli-RRT* every time a new edge is instantiated, its initial and final time instants t_0 and t_f are set, and the edge is checked against the already planned trajectories within the time window $[t_0, t_f]$. If the edge is able to ensure that the distance from the agents whose trajectories have been already set is greater than the minimum safe distance, the edge is collision-free and is added to the tree, otherwise it is discarded.

Algorithm 1 Multi-agent Poli-RRT*

```
 1: $ObstaclesSet \leftarrow \{O_1, O_2, \dots, O_M\}$
 2: $AgentsSet \leftarrow \{A_1, A_2, \dots, A_K\}$
 3: procedure MULTIAGENTPOLIRRT*($N, AgentsSet, ObstaclesSet, safeDist$)
 4:     $TrajList \leftarrow \emptyset$
 5:     for $i = 1$ to $K$ do
 6:         $Traj_i \leftarrow PoliRRT^*\,(A_i, ObstaclesSet, safeDist)$
 7:         $TrajList \leftarrow TrajList \cup Traj_i$
 8:         $ObstaclesSet \leftarrow ObstaclesSet \cup Traj_i$
 9:     end for
10: return $TrajList$
11: end procedure
```

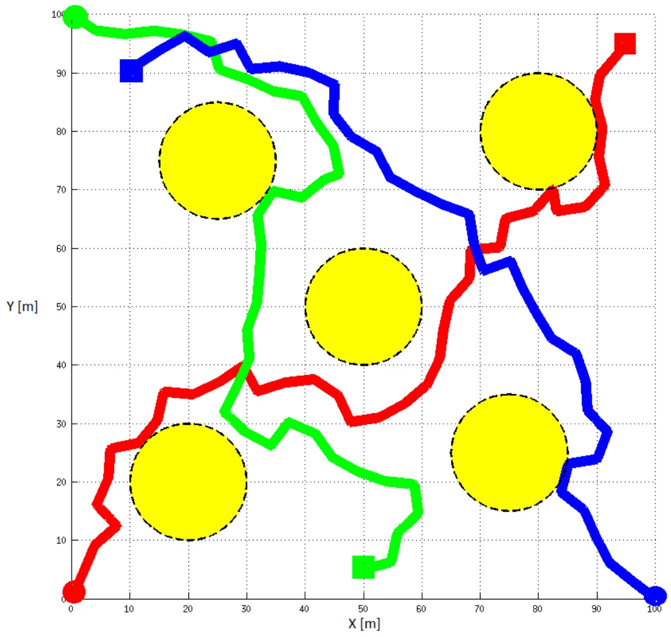
4 Simulation Results

In this section an example is shown where the multi-agent version of Poli-RRT* is applied to a three-agent system.
The three agents must reach their respective targets while moving in the same environment. The following unicycle model is adopted for each vehicle dynamics:

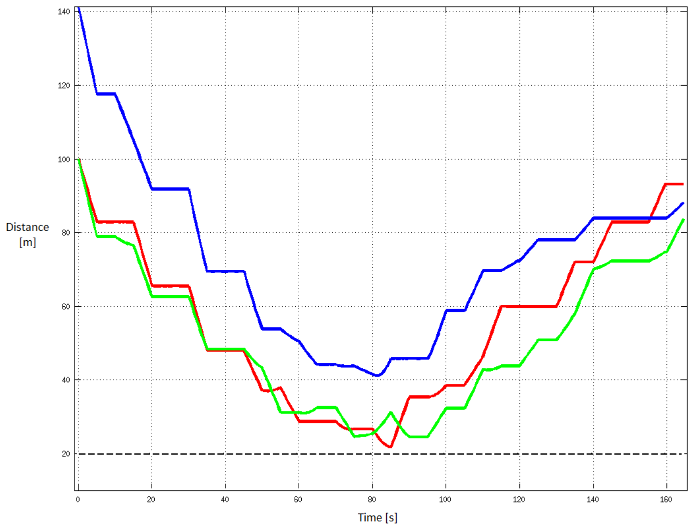
$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \\ \dot{v} &= a \end{aligned} \tag{2}$$

where (x, y) is the vehicle position, θ is the orientation and v the velocity. The control inputs are the angolar velocity ω and the linear acceleration a.
By applying to (2) the feedback linearisation strategy [20]

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



(a)



(b)

Fig. 1. A multi-agent Poli-RRT* run. 1(a) trajectories for agent 1 (blue), 2 (red), and 3 (green), circular and square markers represent starting and goal positions, respectively. 1(b): pairwise agent distance (1-2 blue, 2-3 red, 1-3 green) and minimum safety distance (black dashed).

we obtain the following double integrator linear model

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

where we set $v_x = v\cos(\theta)$ and $v_y = v\sin(\theta)$.
The start and goal configurations are listed in the following:

- Agent 1:

$$x^1_{start} = \{\, x = 0,\ y = 0,\ \theta = \pi/3,\ v = 0 \,\}$$
$$X^1_{goal} = \{\, x \in [92,97]\,,\ y \in [92,97]\,,\ \theta \in [2\pi/5, 3\pi/5]\,,\ v \in [0,0.1] \,\}$$

- Agent 2:

$$x^2_{start} = \{\, x = 0,\ y = 100,\ \theta = -\pi/9,\ v = 0 \,\}$$
$$X^2_{goal} = \{\, x \in [48,52]\,,\ y \in [3,8]\,,\ \theta \in [-3\pi/5, -2\pi/5\,]\,,\ v \in [0,0.1] \,\}$$

- Agent 3:

$$x^3_{start} = \{\, x = 100,\ y = 0,\ \theta = 5\pi/6,\ v = 0 \,\}$$
$$X^3_{goal} = \{\, x \in [8,13]\,,\ y \in [87,92]\,,\ \theta \in [4\pi/5, 6\pi/5]\,,\ v \in [0,0.1] \,\}$$

where linear positions are expressed in *m*, angles in *rad*, and linear velocities in m/s.
State variables are subject to the following bounds

$$x \in [0,100] \qquad y \in [0,100] \qquad \theta \in [-\pi, +\pi] \qquad v \in [0,1]$$

whereas actuation constraints on linear acceleration and angular velocity are given by

$$a \in A = [-0.50, +0.50] \qquad \omega \in \Omega = [-0.50, +0.50]$$

The control variables are weighted in the optimal control problem by matrix $R = 10I_2$.
Figure 1(a) shows the simulation results obtained when the maximum tree cardinality is set equal to 200. Note that the algorithm is able to plan the required trajectories. Moreover, the minimum safety distance between each couple of agents is guaranteed (Figure 1(b)).

5 Conclusions

This paper extends the Poli-RRT* algorithm to the case of multi-agent systems. Agents are sorted according to a given hierarchy and the Poli-RRT* algorithm is executed for each agent following the priority order. The algorithm is validated in a simulated environment. The resulting solution is sub-optimal from the perspective of the multi-agent system, as it depends on the agents ordering. In turn, single agent trajectory planning can be exploited to make the problem tractable.

References

1. Pivtoraiko, M., Knepper, R., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics **26** (2009) 308–333

2. Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. The International Journal of Robotic Research **28** (2009) 933–945
3. Tahirovic, A., Magnani, G.: General framework for mobile robot navigation using passivity-based MPC. IEEE Transactions on Automatic Control **56** (2011) 184–190
4. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research **20** (2001) 378–400
5. Barraquand, J., Kavraki, L., Latombe, J., Motwani, R., Li, T., Raghavan, P.: A random sampling scheme for path planning. The International Journal of Robotics Research **16** (1997) 759–774
6. Branicky, M., Curtiss, M., Levine, J., Morgan, S.: RRTs for nonlinear, discrete, and hybrid planning and control. In: IEEE Conference on Decision and Control (CDC). Volume 1. (2003) 657–663
7. Branicky, M., Curtiss, M., Levine, J., Morgan, S.: Sampling-based planning, control and verification of hybrid systems. IEEE Proceedings Control Theory and Applications **153** (2006) 575–590
8. Karaman, S., Frazzoli, E.: Optimal kinodynamic motion planning using incremental sampling-based methods. In: IEEE Conference on Decision and Control (CDC), Atlanta, GA (2010)
9. Karaman, S., Frazzoli, E.: Incremental sampling-based algorithms for optimal motion planning. In: Robotics: Science and Systems (RSS), Zaragoza, Spain (2010)
10. Karaman, S., Walter, M., Perez, A., Frazzoli, E., Teller, S.: Real-time motion planning using the RRT*. In: IEEE International Conference on Robotics and Automation (ICRA). (2011)
11. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research **30** (2011) 846–894
12. Karaman, S., Frazzoli, E.: Sampling-based optimal motion planning with deterministic μ-calculus specifications. In: American Control Conference (ACC). (2012)
13. Perez, A., Karaman, S., Walter, M., Shkolnik, A., Frazzoli, E., Teller, S.: Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2011)
14. hwan Jeon, J., Karaman, S., Frazzoli, E.: Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In: IEEE Conference on Decision and Control and European Control Conference (CDC - ECC). (2011) 3276–3282
15. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes: The Art of Scientific Computing. Cambridge University Press (2007)
16. Webb, D., van den Berg, J.: Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In: IEEE Intenrational Conference on Robotics and Automation (ICRA). (2013) 5054–5061
17. Perez, A., Platt, R., Konidaris, G., Kaelbling, L., Lozano-Perez, T.: LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In: IEEE International Conference on Robotics and Automation (ICRA). (2012) 2537–2542

18. Goretkin, G., Perez, A., Platt, R., Konidaris, G.: Optimal sampling-based planning for linear-quadratic kinodynamic systems. In: IEEE Intenrational Conference on Robotics and Automation (ICRA). (2013)
19. Ragaglia, M., Prandini, M., Bascetta, L.: Poli-RRT*: optimal RRT-based planning for constrained and feedback linearisable vehicle dynamics. In: European Control Conference (ECC). (2015)
20. Stipanovic, D., Inalhan, G., Teo, R., Tomlin, C.: Decentralized overlapping control of a formation of unmanned aerial vehicles. Automatica **40** (2004) 1285–1296