NONHOLONOMIC MOTION PLANNING IN PARTIALLY UNKNOWN ENVIRONMENTS USING VECTOR FIELDS AND OPTIMAL PLANNERS

Guilherme A. S. Pereira*, Sanjiban Choudhury†, Sebastian Scherer†

*School of Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil

†Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.

Emails: gpereira@ufmg.br, sanjibac@andrew.cmu.edu, basti@andrew.cmu.edu

Abstract— This paper presents a methodology to integrate vector field-based robot motion planning techniques with optimal trajectory planners. The main motivation for this integration is the solution of planning problems that are intuitively solved using vector fields, but are very difficult to be even posed as an optimal motion planning problem, mainly due to the lack of a clear cost function. Among such problems are the ones where a goal configuration is not defined, such as circulation of curves and road following. While several vector field based methodologies were proposed to solve these tasks, they do not explicitly consider the robot's differential constraints and are susceptible to failures in the presence of previously unmodeled obstacles. Our methodology uses a vector field as a high level specification of a task and an optimal motion planner (in our case RRT*) as a local planner that generates trajectories that follow the vector field and also consider the kinematic and dynamic constraints of the robot, as well as the new obstacles encountered in the workspace. To illustrate the approach, we show simulations with a Dubins like vehicle moving in partially unknown planar environments.

Keywords— Robotics, motion planning, vector fields, optimal planning, RRT*.

Resumo— Este artigo apresenta uma metodologia para integração de técnicas de planejamento de movimento para robôs baseadas em campos vetorias e planejadores ótimos. A principal motivação para essa integração é a solução de problemas de planejamento que são intuitivamente solucionados usando campos vetoriais, mas são muito difícies de serem modelados como um problema de planejamento ótimo, principalmente devido à falta de uma função de custo. Entre esses problemas estão aqueles em que um alvo não é definido, como circulação de curvas e seguimento de rodovias. Enquanto várias metodologias baseadas em campos vetoriais foram propostas para solucionar essas tarefas, elas não consideram explicitamente as restrições diferenciais dos robôs e estão sujeitas à falhas na presença de obstáculos que não foram previamente modelados. A metodologia proposta nesse artigo usa um campo vetorial como uma especificação de alto nível para uma tarefa e um planejador ótimo (nesse caso o RRT*) como um planejador local que gera trajetórias que seguem o campo vetorial e também consideram as restrições cinemáticas e dinâmicas do robô, bem como os novos obstáculos encontrados em seu espaço de trabalho. Para ilustrar a metodologia, o artigo apresenta simulações utilizando um robô com cinemática de Dubins se locovendo em um ambiente planar parcialmente conhecido.

Palavras-chave— Robótica, planejamento de movimento, campos vetorias, planejamento ótimo, RRT*.

1 Introduction

In recent years, optimal motion planning techniques have been proposed to solve problems encountered by several types of robots, ranging from humanoids with several degrees of freedom (Sohn and Oh, 2015) to high speed nonholonomic vehicles (Jeon et al., 2013). Besides finding optimal trajectories, these planners may consider the robot's differential constraints to guarantee that the resultant trajectories can be followed by the robot. In this category, an important global motion planning technique is the asymptotically optimal version of The Rapidly-Exploring Random Tree (RRT) (LaValle and Kuffner, 2001), called RRT* (Karaman and Frazzoli, 2011). RRT* is used to search for optimal trajectories from a given initial configuration to a specific goal region. In fact, RRT* is a very powerful tool that can be interpreted as "an anytime computation framework for optimization problems with complex differential and geometric constraints" (Karaman et al., 2011). In this context, "anytime" means that the method quickly finds a feasible, but not necessarily optimal solution for the problem, then incrementally improves it towards optimality. Although RRT* and other optimal planners are used in several robotic problems, they are limited to bounded workspaces with a well defined goal region. It is then difficult to use this tool in some tasks that include persistent monitoring of ground areas, tracking of moving targets, and navigation in urban environments, where a goal configuration is not specified. Moreover, for this kind of task, where the workspace can be very large, the computational time of RRT* can prevent its use for real time operations. Solutions for these problems have been proposed by some authors, generally combining RRT* with other strategy. One example is (Kuwata et al., 2009), which uses a higher level planner to define a sequence of waypoints and use RRT* to reach each waypoint in minimum time avoiding obstacles and respecting the vehicle constraints.

In this paper we combine RRT* with a vector field methodology. We do not assume any specific vector field, which may be computed as the gradient of a potential or navigation function (Khatib, 1986; Rimon and Koditschek, 1992) but, more important, can be the ones that aim,

for example, circulation of curves for perimeter surveillance (Gonçalves et al., 2010) or corridor following (Pereira et al., 2009). Although vector fields have been used alone to control several kinds of real world robots, ranging from manipulators (Khatib, 1986) to fixed wings unnamed aerial vehicles (UAV's) (Frew et al., 2008), most of the methodologies cannot be easily extended to multidimensional spaces and complex environments. Also they do not explicitly consider the robot's differential constraints, what requires the use on nonlinear controllers to track the field (what in some cases cannot be achieved) or the combination of the field with other techniques (Owen et al., 2014). Moreover, vector fields are better applied for static and previously known workspaces. Although the technique was originally created to deal with dynamic environments, global convergence properties are generally lost in such situations. A few works have locally modified the vector field in real time to avoid dynamical obstacles and still maintained convergence properties (Esposito and Kumar, 2002; Loizou et al., 2003), but this modification generally does not consider the quality of the final trajectory.

To allow motion planning for nonholonomic robots moving in partially unknown and dynamic workspaces, this paper builds upon the path planning framework proposed by the authors in (Pereira et al., 2016). The main contribution of this paper is the inclusion of differential constraints on the optimization problem proposed by Pereira et al. (2016) to allow the computation of trajectories for nonholonomic robots. With this modification, the framework can now: i) extend the use of RRT* and other optimal planners to the several tasks that can be easily solved using vector fields but that could not be directly solved with optimal planners, and ii) add some of the important properties of the optimal planners, such as the consideration of the vehicle differential constraints and optimality, to well known vector field methodologies.

By the author's knowledge, besides (Pereira et al., 2016), RRT* and vector fields were only used together in (Qureshi et al., 2013), where the authors propose a new sampling strategy for RRT* that, using a vector field, guides the search tree towards the goal, thus speeding up the method. Differently from (Qureshi et al., 2013), in (Pereira et al., 2016), and consequently in the present paper, RRT* is tightly coupled with the vector field in the sense that the field is used to define the cost function to be minimized by RRT*. This is only possible because, as it is usual in the RRT* literature (Karaman et al., 2011), although constraints in the robot's state space are considered, the planing is actually done in the lower dimensional task space or even in the configuration space, where the vector field is generated. Next section will formally define our problem

2 Problem Definition

Let $\mathcal{Q} \subset \mathbb{R}^n$ be the configuration space of a robot and $\mathcal{Q}_{\text{obs}} \subset \mathcal{Q}$ be an invalid set of configurations that result in collision. We assume that $\mathcal{Q}_{\text{obs}} = \mathcal{Q}^k_{\text{obs}} \cup \mathcal{Q}^m_{\text{obs}}$, where $\mathcal{Q}^k_{\text{obs}}$ is the set of previously known obstacles, and $\mathcal{Q}^m_{\text{obs}}$ is the set of movable and previously unknown obstacles. The free configuration space is defined as $\mathcal{Q}_{\text{free}} = \mathcal{Q}/\mathcal{Q}_{\text{obs}}$. Also, let $\mathbf{u} : \mathcal{Q} \backslash \mathcal{Q}^k_{\text{obs}} \to \mathbb{R}^n$ be a continuous vector field that assigns a vector $\mathbf{u}(\mathbf{q})$ to each configuration $\mathbf{q} \in \mathcal{Q}_{\text{free}} \cup \mathcal{Q}^m_{\text{obs}}$. This vector field is responsible for the specification of the robot task and is computed by a global planner that has no knowledge about $\mathcal{Q}^m_{\text{obs}}$. The robot dynamics is specified by constraints of the form $g\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \dots\right) \leq 0$, $\mathbf{q} \in \mathcal{Q}$. Finally, let the trajectory $\xi : [0,1] \to \mathcal{Q}$ be a smoothing mapping from time to configuration. Our local motion planning problem can then be posed as:

Problem 1 Find, inside the ball $\mathcal{B}_r \subset \mathcal{Q}$ of radius r centered at the initial configuration $\mathbf{q}_0 \in \mathcal{Q}_{\text{free}}$, the smallest collision free and dynamically feasible trajectory that starts at $\mathbf{q}_0$ and follows the vector field as close as possible. This problem can be written as:

$$\underset{\xi}{\text{minimize}} \quad F[\xi, \mathbf{u}] = \int_0^1 f\left(\xi(\tau), \mathbf{u}\left(\xi(\tau)\right)\right) d\tau$$

subject to:

$$\begin{aligned} &\xi(0) = \mathbf{q}_0\,, \\ &\|\xi(1) - \xi(0)\| = r\,, \\ &g\left(\xi, \dot{\xi}, \ddot{\xi}, \dots\right) \leq 0\,, \\ &\xi(t) \in \mathcal{Q}_{\text{free}}, \forall t \in [0,1]\,. \end{aligned}$$

It is important to notice that Problem 1 presents two major differences in relation to the standard motion planning problem. First, there is no definition of a goal configuration. This is replaced by a constraint that enforces the distance between the initial configuration and the end of the trajectory. Second, $F[\xi, \mathbf{u}]$, which represents the cost functional for the optimization problem, substituted the traditional euclidean distance function. Since no goal configuration is defined, it is the role of this functional to dictate the direction of the robot movement. To consider both the vector field and the length of the trajectory, we propose in (Pereira et al., 2016) a cost functional of the form:

$$F[\xi, \mathbf{u}] = \int_0^1 \left(a - b\,\frac{\dot{\xi}(\tau)}{\|\dot{\xi}(\tau)\|} \cdot \frac{\mathbf{u}(\xi(\tau))}{\|\mathbf{u}(\xi(\tau))\|}\right) \|\dot{\xi}(\tau)\| d\tau\,, \tag{1}$$

where $a, b \in \mathbb{R}_+$ and $a > b$. The values of a and b are chosen so that the cost is small when the trajectory is parallel to the vector field (the inner

product between the normalized field vector and the unit vector tangent to the trajectory is one), and increases when the trajectory is not parallel to the field (inner product is smaller than one). If $a = 2$ and $b = 1$, for example, the cost for the case in which the trajectory is anti-parallel to the field (inner product is -1) will be three times higher than the one in which it is parallel to field, considering the same length of the tangent vector. As shown by Pereira et al. (2016), cost functional (1) is non-symmetric, strictly positive, bounded, additive and, as long as $a - b > 0$, it is also monotonic.

3 Methodology

Following (Pereira et al., 2016), in this paper we solve Problem 1 using RRT*. Although another optimization based motion planner could be used, sampling based motion planners such as RRT* have two important characteristics that are suitable for our problem, which requires real time planning in partially unknown workspaces: 1) they do not require an explicit geometric representation of the environment, important when new obstacles are detected, and 2) because they are anytime planners, a solution, albeit not necessarily optimal, can be found very fast and improved if more time is available. We rapidly explain RRT* in the next subsection.

*3.1 RRT**

RRT* basically works in two phases. In the first phase, a tree (graph without loops) with root at the origin is computed. The nodes of the tree are free random samples of the robot's configuration space and the edges represent the existence of a trajectory between two nodes. The edges are chosen so that paths (sequence of nodes) that start at the root represent the best possible trajectories from the root to any node of the tree. In the second phase of the method, called query phase, the trajectory to be followed by the robot is then extracted from the tree.

The algorithm for computing an optimal tree using RRT* is shown in Algorithm 1. The basic functions called by this algorithm are:

- SampleFree: Generates a random configuration in $\mathcal{Q}_{\text{free}}$.
- Nearest: Finds the node of graph G that is the closest to $\mathbf{q}_{\text{new}}$ in terms of a given distance function.
- Steer($\mathbf{q}_{\text{i}}$, $\mathbf{q}_{\text{j}}$): Computes a trajectory ξ that starts in $\mathbf{q}_{\text{i}}$ and finishes in $\mathbf{q}_{\text{j}}$. This trajectory must respect differential constrains of the form $g\left(\xi, \dot{\xi}, \ddot{\xi}, \dots\right) \leq 0$.
- CollisionFree(ξ): Returns True if the trajectory ξ lies in $\mathcal{Q}_{\text{free}}$ and False otherwise.

Algorithm 1 RRT*

```
1: V ← {q_0}; E ← ∅; t ← 0;
2: while t < t_max do
3:     q_new ← SampleFree;
4:     q_nearest ← Nearest(G = (V, E), q_new);
5:     ξ_new ← Steer(q_nearest, q_new);
6:     if CollisionFree(ξ_new) then
7:         Q_near ← Near(G = (V, E), q_new, η);
8:         V ← V ∪ {q_new};
9:         q_min ← q_nearest;
10:        c_min ← Cost(q_nearest) + TCost(ξ_new);
11:        for all q_near ∈ Q_near do
12:            ξ' ← Steer(q_near, q_new);
13:            if CollisionFree(ξ') ∧
14:             Cost(q_near) + TCost(ξ') < c_min then
15:                q_min ← q_near;
16:                c_min ← Cost(q_near) + TCost(ξ');
17:            end if
18:        end for
19:        E ← E ∪ {(q_min, q_new)};
20:        for all q_near ∈ Q_near do
21:            ξ' ← Steer(q_new, q_near);
22:            if CollisionFree(ξ') ∧
23:           Cost(q_new)+TCost(ξ') < Cost(q_near) then
24:               q_parent ← Parent(q_near);
25:               E ← E\{(q_parent, q_near)}∪{(q_new, q_near)}
26:            end if
27:        end for
28:    end if
29: end while
30: return G = (V, E)
```

- Near($G = (V, E), \mathbf{q}_{\text{i}}, \eta$): Computes the set of nodes of graph G that are inside the ball centered in $\mathbf{q}_{\text{i}}$ and radius determined by η (Karaman and Frazzoli, 2011).
- Cost($\mathbf{q}_{\text{i}}$): Returns the cost of the trajectory that starts in $\mathbf{q}_0$ and finishes in $\mathbf{q}_{\text{i}}$. This information is generally stored in the tree.
- TCost(ξ): Computes the cost of trajectory ξ using the specified cost function.
- Parent($\mathbf{q}_{\text{i}}$): Returns the parent node of $\mathbf{q}_{\text{i}}$ in tree G. By convention, if $\mathbf{q}_{\text{i}}$ is the root of G, the function will return $\mathbf{q}_{\text{i}}$.

The RRT* algorithm starts with a empty graph $G = (V, E)$, where V is the node set and E is the edge set, and returns a tree (graph without loops) with root node at the initial configuration (line 1 of Algorithm 1). For a given constant time t_{max}, the algorithm generates random samples of the free configuration space and try to compute trajectories that connects these samples to a node of the tree. For each random sample, if a free trajectory from the sample's closest node exist, the sample is included as a node of the tree (lines 4–8). After that, the algorithm looks for even best trajectories from other nodes of the tree to the new node (lines 9–18). The best trajectory found will define a new edge on the tree (line 19). In a second optimization step, the algorithm tries to "rewire" the tree by checking if the inclusion of the new node can reduce the cost of reaching other nodes of the tree (lines 20–27).

Once the tree is computed, either one of their nodes is chosen to be the end of the trajectory

(what will be done in this work), or the goal position is connected to tree using the same procedure used to connect $\mathbf{q}_{\text{new}}$. Since each node has a reference to its parent, a trajectory is then computed by simply following these references from the goal to the start position. To solve Problem 1 using RRT*, the correct definition of functions SAMPLEFREE, STEER and TCOST is necessary. These functions are defined next.

3.2 Sampling function

Since Problem 1 specifies that a trajectory must start at $\mathbf{q}_0$ and finish at the surface of a ball of radius r centered at $\mathbf{q}_0$, it makes sense to uniformly generate samples only inside this ball. A way to do this is to represent the ball in spherical coordinates and sample for each coordinate. In 2D, the spheric coordinates are the radius, d, and azimuthal angle, ϕ. Since the probability that a given sample is inside the ball of radius r is proportional to r^2, we can generate samples for the radius as $d = r\sqrt{v_1}$, where v_1 is a real random number sampled with uniform distribution over the interval $[0, 1]$. The other coordinate, ϕ, can be generated as $\phi = 2\pi v_2$, where v_2 is a random variable similar to v_1, but independent of v_1. For planar nonholonomic mobile robots, besides position, the robot's orientation θ, must also be sampled. Following the way ϕ is sampled, we can generate the orientation as $\theta = 2\pi v_3$, where v_3 is a random variable independent of v_1 and v_2. Therefore each random configuration for a planar mobile robot is given by $\mathbf{q} = [d\cos(\phi), d\sin(\phi), \theta]^T$. This procedure can be extended for robots with more degrees of freedom.

Since function SAMPLEFREE assumes free configurations, after generated, the configuration must be checked for collisions. This can be done using a global map, if it is available, or a local grid computed on-line with the robot's sensors (Pereira et al., 2016).

3.3 Steering function

A correct and efficient implementation of the steering function is essential to allow the use of RRT* in real time, once function steer is called $O(|V|)$ in each iteration of Algorithm 1. In the version of RRT* used in this paper, its is assumed that the steering function STEER($\mathbf{q}_\text{i}$, $\mathbf{q}_\text{j}$) finds a differential constrained trajectory that starts in $\mathbf{q}_\text{i}$ and finishes in $\mathbf{q}_\text{j}$. More generic versions of the algorithm allows the trajectory to finish in a region around the final configuration (Karaman et al., 2011). This is indeed a way to reduce the complexity of the steering function, which includes the determination of an optimal arrival time at the final configuration. A solution to this problem assuming linear system dynamics is presented in (Webb and van den Berg, 2013).

For planar nonholonomic robots, an efficient solution to the steering problem is to assume that the robot can be modeled using the Dubins' model with constant linear velocity and fixed turning radius. This model is given by:

$$\begin{aligned} \dot{x} &= v\cos\theta\,, \\ \dot{y} &= v\sin\theta\,, \\ \dot{\theta} &= \omega\,, \qquad \omega = v/\rho\,, \end{aligned} \tag{2}$$

where x, y and θ form the robot's configuration $\mathbf{q}$, v and ω are the robot's input velocities and ρ is the robot's turning radius. In this situation, it is always possible to steer the robot between two configurations using a sequence of simple motions that either make the robot to move straight or to turn with its fixed turning radius. This idea was implemented in this paper using the same strategy proposed by Karaman et al. (2011) where four of the six possible sequences of motions are used to steer the robot.

3.4 Trajectory cost function

In the methodology proposed in this paper, the cost of a trajectory given by function TCOST(ξ) in Algorithm 1 is obtained by computing the functional in Equation (1). Since this functional depends on the vector field and on the derivative of the trajectory, both dependent on the robot's configuration, the computation of the exact value of the functional in continuous time would be very time consuming. A practical solution is then to discretize the trajectory into small intervals and assume that the both the derivative of the trajectory and the vector field are constant during each interval. The integral in Equation (1) is then replaced by a sum, which can computed in time $O(k)$, where k is the number of intervals.

3.5 Other considerations

In this paper, it is assumed that the vector field is computed over a reduced version of the robot's configuration space composed by position only. Therefore, for a planar robot with configuration given by $\mathbf{q} = [x, y, \theta]^T$, we assume that, given a position (x, y), the vector field is constant for all values of orientation θ. Orientation is also ignored by functions NEAR($\cdot$) and NEAREST($\cdot$), which we assume to apply an Euclidean distance function over the linear components of the configuration.

However, it is important to observe that the complete configuration (position and orientation) is used in all other steps of the method since $\mathbf{q}_{\text{new}}$ is sampled over a region of the complete configuration space, as explained in Subsection 3.2.

Another important consideration is related to the query phase of RRT*, when the final trajectory will be determined. Since no goal configuration is specified by the problem, we grow a

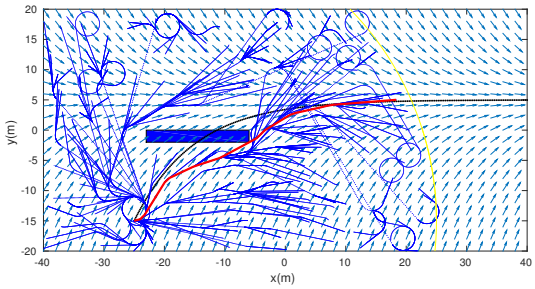

Figure 1: Illustrative example. A vector field (arrows) was computed without the knowledge of the obstacle (blue box). Thus, the integral of the vector field (black) would lead the robot to a collision. The proposed methodology uses an optimal search tree (blue lines) to find a dynamically feasible, and asymptotically optimal trajectory (red) that avoids the obstacle and follows the vector field.

tree that may slightly exceeds the limits of the planning region given by radius r and, during the query phase, choose, among the configurations within distances $(r-\delta, r+\delta)$, where δ is a small constant, the one with the smallest cost to represent the final configuration.

Next section presents some simulations that illustrate the proposed approach.

4 Simulations

We have implemented the proposed methodology in C++ using OMPL (Şucan et al., 2012) and ROS (Quigley et al., 2009) for $\mathcal{Q} \subset SE(2)$. Therefore, $\mathbf{q} = [x, y, \theta]^T$. We assumed that the robot can be modeled as Dubins' vehicle with turning radius $\rho = 2\,\text{m}$. All simulations run in an Intel Core i7-3.00GHz Linux computer.

In our first simulation, shown in Figure 1, we have a robot in a large environments with a single unmodeled obstacle (blue box). The robot starts at configuration $\mathbf{q}_0 = [-25, 15, 0]^T$. A vector field was constructed over the environment to make the robot to move on the $y = 5$ line. Since the obstacle was not known during the field computation, the integral of the vector field (black trajectory) would lead the robot to a collision. Using the proposed methodology, the robot was able to construct an optimal tree (shown by the blue lines) inside its sensor radius, $r = 50\,\text{m}$ (shown in yellow), and find a feasible trajectory (shown in red) that both avoids the obstacle and follow the vector field as close as possible. This trajectory was computed with $t_{\max} = 2\,\text{s}$.

In our second set of simulations, we used the vector field proposed by Gonçalves et al. (2010) for curve circulation. Curve circulation is among the tasks that cannot be solved by RRT* alone due to the lack of a goal position. We set the radius of the planning region to be $r = 10\,\text{m}$. In

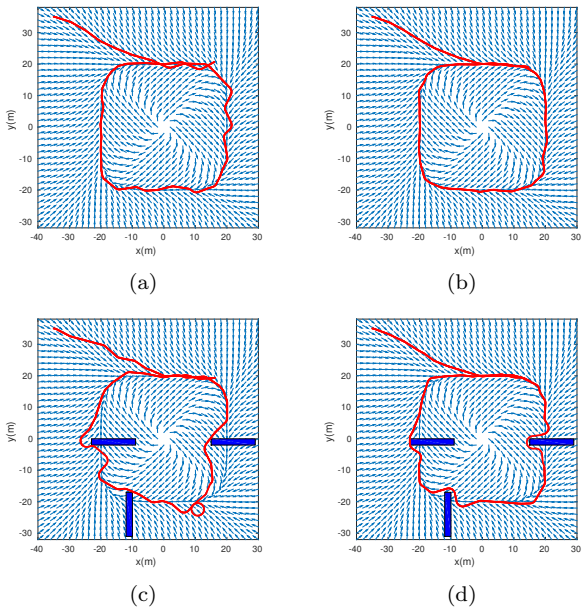

Figure 2: Circulation of curves. Effect of the execution time $t_{\max}$. On the left $t_{\max} = 1\,\text{s}$ and on the right $t_{\max} = 5\,\text{s}$. (a)-(b) free workspace; (c)-(d) presence of unmodelled obstacles.

this simulation, the robot computes, for a fixed amount of time $t_{\max}$, a trajectory that starts at $\mathbf{q}_0$ and finishes at the borders of the sampling region. Then, it follows 50% of the path and computes, for the same fixed amount of time $t_{\max}$, a new path that starts at its current configuration and finishes at the borders of the new sampling region centered at the new start configuration. This is repeated until the end of the simulation. Figure 2 shows the results of the method for $t_{\max} = 1\,\text{s}$ (figures (a) and (c)) and $t_{\max} = 5\,\text{s}$ (figures (b) and (d)) with and without obstacles (blue rectangles). Notice that, although all trajectories computed are dynamically feasible, the quality of the trajectory increases if the method has more time to execute. The trajectory in Figure 2(c), for example, has a loop at the bottom right of the figure, which is removed in the result in Figure 2(d), where more optimization time was spent. A video of this simulation, which includes the search trees for each step, can be found in `www.cpdee.ufmg.br/~coro/movies/cba2016/`.

5 Conclusions

This paper presented a methodology to integrate vector field methodologies with optimal motion planners to allow the safe navigation of nonholonomic robots in partially unknown workspaces. Simulated results shown the potentiality of the method to control actual robots in tasks that cannot be solved by optimal motion planners alone.

Next steps of this research include the use of the method to control ground and aerial mobile

robots working in urban environments in the presence of obstacles and people. To do so, the current implementation needs to be speeded up to allow better trajectories in less time. This can be done by the proposition of new sampling approaches such as the ones suggested in (Pereira et al., 2016) and (Qureshi et al., 2013).

Acknowledgments

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq/Brazil), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG/Brazil) and National Science Foundation (NSF/USA).

References

Esposito, J. M. and Kumar, V. (2002). A method for modifying closed-loop motion plans to satisfy unpredictable dynamic constraints at runtime, *IEEE Intl. Conf. Robotics and Automation*, pp. 1691–1696.

Frew, E. W., Lawrence, D. A. and Morris, S. (2008). Coordinated standoff tracking of moving targets using Lyapunov guidance vector fields, *J. Guidance, Control, and Dynamics* **31**(2): 290–306.

Gonçalves, V. M., Pimenta, L. C. A., Maia, C. A., Dutra, B. C. O. and Pereira, G. A. S. (2010). Vector fields for robot navigation along time-varying curves in n-dimensions, *IEEE Trans. Robotics* **26**(4): 647–659.

Jeon, J., Cowlagi, R. V., Peters, S. C., Karaman, S., Frazzoli, E., Tsiotras, P. and Iagnemma, K. (2013). Optimal motion planning with the half-car dynamical model for autonomous high-speed driving, *American Control Conf.*, pp. 188–193.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning, *The Intl. J. of Robotics Research* **30**(7): 846–894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E. and Teller, S. (2011). Anytime motion planning using the RRT*, *IEEE Intl. Conf. Robotics and Automation*, pp. 1478–1483.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *The Intl. J. Robotics Research* **5**(1): 90–98.

Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J. P. and Fiore, G. (2009). Real-time motion planning with applications to autonomous urban driving, *IEEE Trans. Control Systems Technology* **17**(5): 1105–1118.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning, *The Intl. J. Robotics Research* **20**(5): 378–400.

Loizou, S. G., Tanner, H. G., Kumar, V. and Kyriakopoulos, K. J. (2003). Closed loop motion planning and control for mobile robots in uncertain environments, *IEEE Conf. Decision and Control*, Vol. 3, pp. 2926–2931.

Owen, M., Beard, R. W. and McLain, T. W. (2014). Implementing Dubins Airplane Paths on Fixed-Wing UAVs, *Handbook of Unmanned Aerial Vehicles*, pp. 1677–1701.

Pereira, G. A. S., Choudhury, S. and Scherer, S. (2016). A framework for optimal repairing of vector field-based motion plans, *Intl. Conf. Unmanned Aircraft Systems*, pp. 261–266.

Pereira, G. A. S., Pimenta, L. C. A., Chaimowicz, L., Fonseca, A. R., Almeida, D. S. C., Corrêa, L. Q., Mesquita, R. C. and Campos, M. F. M. (2009). Robot navigation in multi-terrain outdoor environments, *The Intl. J. Robotics Research* **28**(6): 685–700.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y. (2009). ROS: an open-source robot operating system, *ICRA Workshop on Open Source Software*.

Qureshi, A. H., Iqbal, K. F., Qamar, S. M., Islam, F., Ayaz, Y. and Muhammad, N. (2013). Potential guided directional-RRT* for accelerated motion planning in cluttered environments, *IEEE Intl. Conf. Mechatronics and Automation*, pp. 519–524.

Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions, *IEEE Trans. Robotics and Automation,* **8**(5): 501–518.

Sohn, K. and Oh, P. (2015). Optimization of humanoid's motions under multiple constraints in vehicle ingress task, *Intelligent Service Robotics* **9**(1): 31–48.

Şucan, I. A., Moll, M. and Kavraki, L. E. (2012). The Open Motion Planning Library, *IEEE Robotics & Automation Magazine* **19**(4): 72–82. `http://ompl.kavrakilab.org`.

Webb, D. J. and van den Berg, J. (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics, *IEEE Intl. Conf. Robotics and Automation*, pp. 5054–5061.