Acta Polytechnica **00**(0):1–8, 0000

PREPRINT 2015-07-23

MULTI ROBOT OPTIMAL TRAJECTORY GENERATION

José M. Mendes Filho[a,b], Eric Lucet[a,*]

[a] *CEA, LIST, Interactive Robotics Laboratory, Gif-sur-Yvette, F-91191, France*
[b] *ENSTA Paristech, Unité d'Informatique et d'Ingénierie des Systèmes, 828 bd des Marechaux, 91762, France*
[*] corresponding author: eric.lucet@cea.fr

Abstract.
This paper proposes the real-time implementation of a multi-robot optimal collision-free motion planner algorithm based on a receding horizon approach, for the navigation of a team of mobile robots evolving in an industrial context in presence of different structures of obstacles. The method is validated in simulation environment for a team of three robots. Then, impact of the algorithm parameters setting is studied with regard to critical performance criteria, being mainly real-time implementation, obstacle avoidance and time to complete the task.

Keywords: multi-robot motion planning, nonholonomic mobile robot, decentralized planning, receding horizon.

1. Introduction

The control of mobile robots is a long-standing subject of research in the iterative robotics domain. A trending application of mobile robots systems is its use in industrial supply-chains for processing orders and optimizing products storage and distribution. Companies such as Amazon and the logistic provider IDEA Groupe employ moble multi-robot systems (Kiva systems, Scallog) for autonomously processing clients orders [1, 2]. Such logistics tasks became increasingly complex as uncertainties sources, such as human presence, are admitted in the work environment.

One basic requirement for such mobile robot systems is the capacity of motion planning, i.e., generating admissible configuration and input trajectories that connects two arbitrary states. For solving the motion planning problem, different constraints must be taken into account, in particular:

- robot's kinematic and dynamic constraints;
- geometric constraints;

The first constraints derive directly from the mobile robot architecture implying, for example, in nonholonomic constraints and sliding phenomena. Geometric constraints result from need of preventing the robot to assume specific configurations in order to avoid collisions, communication lost, etc.

We are particularly interested in solving the problem of planning a trajectory for a team of nonholonomic mobile robots in a partially known environment occupied by static obstacles being near optimal with respect to the execution time (time spent from going from initial to goal configurations).

In recent years, a great amount of work towards collision-free trajectory planning has been proposed.

Some work has been done towards analytic methods for solving the problem for certain classes of systems ([]) TODO cite . However [3] shows that analytic methods are inapplicable for nonholonomic systems in presence of obstacles.

Cell decomposition methods, as presented in [4], have the downside of requiring a structured configuration space and an *a priori* model of its connectivity. Besides, the cell decomposition reduces the space of admissible solutions.

Initially proposed in [**?**], the vector field obstacle avoidance method was improved along time for treating problems such as oscillation of the solution for narrow passages. This method does not present a near optimal generated trajectory.

Elastic band approach initially proposed by [**?**] and extended to mobile manipulators in [**? ?**] uses approximations of the trajectory shape (combinations of arcs and straight lines) limiting the space of solutions and thus making this method inappropriate for really constrained environments.

The dynamic window approach [5] can handle trajectory planning for robots at high speeds and in presence of obstacles. But it is not flexible enough to be extended to a multi-robot system.

In this paper, we focus on the development of a motion planning algorithm. This algorithm finds collision-free trajectories and computes the corresponding angular and longitudinal velocities for a multi-robot system in presence of static obstacles perceived by the robots as they evolve in their environment. The dynamic trajectories computed are near optimal with respect to the total time spent going from the initial configuration to the final one. Besides, this algorithm uses a decentralized approach making the system more robust to communication outages and individual robot failure than compared to a centralized approach. Identified drawbacks are the dependence on several parameters for achieving real-time performance and good solution optimality, and the difficulty of handling dynamic obstacles as it is.

This algorithm is an extension of the work done in [6]. In particular, improvements are proposed for the respect of a precise final configuration of the multi-robot system. Also, investigation about how to set parameters for achieving satisfying solutions was performed.

This paper is structured as follows: The second section states the problem to be resolved pointing out the cost function for motion planning and all constraints that need to be respected by the computed solution. The third explains the method to resolve the motion planning problem and gives some remarks on how to resolve the constrained optimization problems associated with the method. The forth section is dedicated to the results found using this method and the analysis of the certain performance criteria and how they are impacted by the algorithm parameters. In the fifth section this approach is compared to another one presented in [7]. Finally, in last section we present our conclusions and perspectives.

2. Problem Statement

2.1. Assumptions

In the development of this approach, the following assumptions are made:

(1.) The motion of the multi-robot system begins at the instant t_{init} and goes until the instant t_{final}.

(2.) The team of robots consists of a set $\mathcal{R}$ of B nonholonomic mobile robots.

(3.) A robot (denoted $R_b,\ R_b \in \mathcal{R},\ b \in \{0, \ldots, B-1\}$) is geometrically represented by a circle of radius ρ_b centered at (x_b, y_b).

(4.) All obstacles in the environment are considered static. They can be represented by a set $\mathcal{O}$ of M static obstacles.

(5.) An obstacle (denoted $O_m,\ O_m \in \mathcal{O},\ m \in \{0, \ldots, M-1\}$) is geometrically represented either as a circle or as a convex polygon. In the case of a circle its radius is denoted r_{O_m} centered at (x_{O_m}, y_{O_m}).

(6.) For a given instant $t_k \in [t_{init},\ t_{final}]$, any obstacle O_m having its geometric center apart from the geometric center of the robot R_b of a distance inferior than the detection radius $d_{b,sen}$ of the robot R_b is considered detected by this robot. Therefore, this obstacle is part of the set $\mathcal{O}_b$ ($\mathcal{O}_b \subset \mathcal{O}$) of detected obstacles.

(7.) A robot has precise knowledge of the position and geometric representation of a detected obstacle, i.e., obstacles perception issues are neglected.

(8.) A robot can access information about any robot in the team by using a wireless communication link.

(9.) Latency, communication outages and other problems associated to the communication between robots in the team are neglected.

(10.) Dynamics was neglected.

(11.) The input of a mobile robot R_b is limited.

2.2. Constraints and cost functions

After introducing the motion planning problem in Section 1 and giving the assumptions in the previous Subsection, we can define the constraints and the cost function for the multi-robot navigation.

(1.) The solution of the motion planning problem for the robot R_b represented by the pair $(q_b^*(t), u_b^*(t))$ – $q_b^*(t) \in \mathbb{R}^n$ being the solution trajectory for the robot's configuration and $u_b^*(t) \in \mathbb{R}^p$ the solution trajectory for the robot's input – must satisfy the robots kinematic model equation:

$$\dot{q}_b^*(t) = f(q_b^*(t), u_b^*(t)), \quad \forall t \in [t_{init}, t_{final}]. \tag{1}$$

(2.) The planned initial configuration and initial input for the robot R_b must be equal to the initial configuration and initial input of R_b:

$$q_b^*(t_{init}) = q_{b,init}, \tag{2}$$

$$u_b^*(t_{init}) = u_{b,init}. \tag{3}$$

(3.) The planned final configuration and final input for the robot R_b must be equal to the goal configuration and goal input for R_b:

$$q_b^*(t_{final}) = q_{b,goal}, \tag{4}$$

$$u_b^*(t_{final}) = u_{b,goal}. \tag{5}$$

(4.) Practical limitations of the input impose the following constraint: $\forall t \in [t_{init}, t_{final}]$, $\forall i \in [1, 2, \cdots, p]$,

$$|u_{b,i}^*(t)| \leq u_{b,i,max}. \tag{6}$$

(5.) The cost for the multi-robot system navigation is defined as:

$$L(q(t), u(t)) = \sum_{b=0}^{B-1} L_b(q_b(t), u_b(t), q_{b,goal}, u_{b,goal}) \tag{7}$$

where $L_b(q_b(t), u_b(t), q_{b,goal}, u_{b,goal})$ is the integrated cost for one robot motion planning (see [8]).

(6.) To ensure collision avoidance with obstacles, the euclidean distance between a robot and an obstacle (denoted $\mathrm{d}(R_b, O_m) \mid O_m \in \mathcal{O}_b, R_b \in \mathcal{B}$) has to satisfy:

$$\mathrm{d}(R_b, O_m) \geq 0. \tag{8}$$

For the circle representation of an obstacle the distance $\mathrm{d}(R_b, O_m)$ is defined as:

$$\sqrt{(x_b - x_{O_m})^2 + (y_b - y_{O_m})^2} - \rho_b - r_{O_m}.$$

For the convex polygon representation, the distance was calculated using three different definitions, according to the Voronoi region [9] R_b is located. Voronoi regions are defined by the lines containing the sides (s lines) and by the lines passing through the vertexes that are orthogonal to the sides (r lines) of the convex polygon. Figure 1 shows an quadrilateral $ABCD$ representation of an obstacle where three of the nine regions were distinguished.

In this example, the region in which the robot R_b is located at an instant t_k can be computed by evaluating the line equations s_{AB}, s_{BC}, s_{CD}, s_{DA}, r_{AB}, r_{AD}, r_{BA}, r_{BA}, r_{BC}, r_{BA}, r_{BC}, r_{CB}, r_{CD}, r_{DC} and r_{DA} for the position associated with the configuration $q_b^*(t_k)$.

In addition, the distance robot-to-quadrilateral is computed as follows:

(a) If robot in region 1:

$$\sqrt{(x_b - x_A)^2 + (y_b - y_A)^2} - \rho_b$$

which is simply the distance of the robot to the vertex A.

(b) If robot in region 2:

$$\mathrm{d}(s_{DA}, (x_b, y_b)) - \rho_b$$

where

$$\mathrm{d}(s_{DA}, (x_b, y_b)) = \frac{|a_{s_{DA}} x_b + b_{s_{DA}} y_b + c_{s_{DA}}|}{\sqrt{a_{s_{DA}}^2 + b_{s_{DA}}^2}}.$$

The distance $\mathrm{d}(s_{DA}, (x_b, y_b))$ represents the distance from the robot to the side DA.

(c) If robot in region 3:

$$-\min\left(\mathrm{d}(s_{AB}, (x_b, y_b)), \cdots, \mathrm{d}(s_{DA}, (x_b, y_b))\right) - \rho_b$$

which represents the amount of penetration of the robot in the obstacle.

The distance computation for other regions can be easily inferred from the above equations.

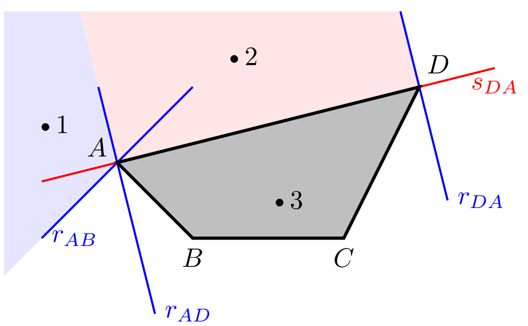


FIGURE 1. Voronoi regions used for case differentiation.

(7.) In order to prevent inter-robot collisions, the following constraint must be respected: $\forall\ (R_b, R_c) \in \mathcal{R} \times \mathcal{R}, b \neq c, c \in \mathcal{C}_b$,

$$\mathrm{d}(R_b, R_c) - \rho_b - \rho_c \geq 0 \tag{9}$$

where $\mathrm{d}(R_b, R_c) = \sqrt{(x_b - x_c)^2 + (y_b - y_c)^2}$ and $\mathcal{C}_b$ is the set of robots that present a collision risk with R_b.

(8.) Finally, the need of a communication link between two robots (R_b, R_c) yields to the following constraint:

$$\mathrm{d}(R_b, R_c) - \min(d_{b,com}, d_{c,com}) \leq 0 \tag{10}$$

with $d_{b,com}, d_{c,com}$ the communication link reach of each robot and $\mathcal{D}_b$ is the set of robots that present a communication lost risk with R_b.

3. Distributed motion planning

3.1. Receding horizon approach

Since the environment is progressively perceived by the robots and new obstacles may appear as time passes, planning the whole motion from initial to goal configurations is not a satisfying approach. Planning locally and replanning is a more suitable approach for taking new information, as they come, into account. Besides, the computation cost of finding a motion plan from start to goal may be prohibited high as planning complexity is most certainly dependent of the start and goal distance. Therefore, a on-line planner that computes the trajectories as the multi-robot system evolves in the environment is proposed. One possible way to do so is to use a receding horizon control approach [10].

This approach is based in two concepts: planning horizon T_p and update/computation horizon T_c. All robots in the team use the same T_p and T_c. T_p is the timespan for which a solution will be computed and T_c is the time horizon during which a plan is executed while the next plan, for the next timespan T_p, is being computed.

For each receding horizon planning problem the following is done:

Step 1. Compute an intended solution trajectory (denoted $(\hat{q}_b(t), \hat{u}_b(t))$) by ignoring coupling constraints, i. e., constraints 9 and 10 that involve other robots in the team.

Step 2. Robots involved in a conflict (collision or lost of communication) update their trajectories by solving another constrained optimization problem that take into account coupling constraints (9 and 10). This is done by using the other robots' intended trajectories computed in the previous step as an estimative of the robots final trajectories. If a robot is not involved in any conflict its final solution trajectory is identical to the one estimated in the Step 1.

This scheme is explained in details in [6] where the receding horizon optimization problems are formulated based on the constraints and cost function defined in Section 2.

However, constraints related to the goal configuration and input of the motion planning problem are neglected in the receding horizon scheme presented in [6]. The constraints 4 and 5 are not considered in the receding horizon optimization problems. For that, a termination procedure is proposed in the following that enables the robots to reach their goal state.

3.2. Motion planning termination

As the robots evolve theirs states approximate to the goal states. But simply stopping the motion planner as the robots are in the neighbourhood of their final configuration is not a satisfying approach (it leaves constraints 4 and 5 unsatisfied).

By considering those constraints in the termination optimization problem and by planning for an undetermined planning horizon the goal configuration can be reached.

The criterion used to pass from the receding horizon optimization problem to the termination optimization problem is define below in the equation 11:

$$d_{rem} \geq d_{min} + T_c \cdot v_{max} \tag{11}$$

This equation insures that the termination plan will be planned for at least a d_{min} distance from the robot's goal position. This minimal distance is assumed to be sufficient for the robot to reach the goal configuration.

After stopping the receding horizon planning we calculate new parameters for the solution representation and computation taking into account the estimate remaining distance and the typical distance travelled for a T_d planning horizon (for instance, the discretization of the time).

The following pseudo code 1 summarizes the planning algorithm and the Figure 2 illustrates its results.

In the pseudo code we see the call of a PlanSec procedure. It represents the resolution of the of the receding horizon planning problem as defined in subsection 3.1.

PlanLastSec is the procedure that solves the termination planning problem. This problem is similar to the receding horizon planning problems. It also has the two steps presented before for computing an intended plan and for updating it, if need be, so conflicts (collision and communication lost) are avoided. The difference consists in how the optimization problems associated are defined. The optimization problem defined in equations 12 and 13 is the problem solved at the first step which generates an intended plan (denoted ($\hat{q}_b(t)$, $\hat{u}_b(t)$)). The optimization problem associated with the update step is defined in equations 14 and 15 and produces the final solution ($q_b^*(t)$, $u_b^*(t)$).

In these new optimal problems the planning horizon is not a constant as before, instead it is a part of the solution to be found.

Algorithm 1 Motion planning algorithm

```
procedure Plan
    q_latest ← q_initial
    d_rem ← |Pos(q_final)−Pos(q_latest)|
    while d_rem ≥ d_min + T_c · v_max do
        InitSolRepresentation(···)
        q_latest ←PlanSec(···)
        d_rem ← |Pos(q_final)−Pos(q_latest)|
    end while
    RescaleRepresentation(···)
    T_f ←PlanLastSec(···)
end procedure
```

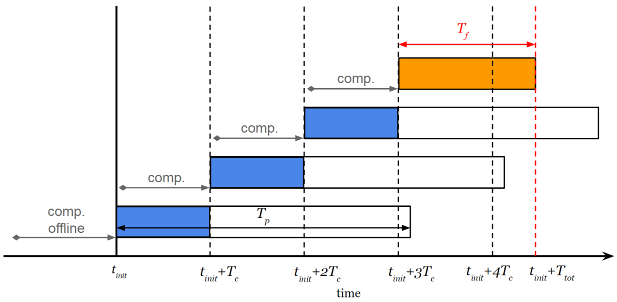


Figure 2. Receding horizon scheme with termination plan. The timespan T_f represents the duration of the plan for reaching the goal configuration.

$$\min_{\hat{q}_b(t),\hat{u}_b(t),T_f} L_{b,f}(\hat{q}_b(t),\hat{u}_b(t),q_{b,goal},u_{b,goal}) \tag{12}$$

under the following constraints for $\tau_k = kT_c$ with k the number of receding horizon problems solved before the termination problem:

$$\begin{cases} \dot{\hat{q}}_b(t) = f(\hat{q}_b(t),\hat{u}_b(t)), \quad \forall t \in [\tau_k, \tau_k + T_f] \\ \hat{q}_b(\tau_k) = q_b^*(\tau_{k-1} + T_c) \\ \hat{u}_b(\tau_k) = u_b^*(\tau_{k-1} + T_c) \\ \hat{q}_b(\tau_k + T_f) = q_{b,goal} \\ \hat{u}_b(\tau_k + T_f) = u_{b,goal} \\ |\hat{u}_{b,i}(t)| \leq u_{b,i,max}, \quad \forall i \in [1,p], \forall t \in (\tau_k, \tau_k + T_f) \\ d(R_b, O_m) \geq 0, \quad \forall O_m \in \mathcal{O}_b, t \in (\tau_k, \tau_k + T_f) \end{cases} \tag{13}$$

$$\min_{q_b^*(t),u_b^*(t),T_f} L_{b,f}(q_b^*(t),u_b^*(t),q_{b,goal},u_{b,goal}) \tag{14}$$

under the following constraints:

$$\begin{cases} \dot{q^*}_b(t) = f(q_b^*(t),u_b^*(t)), \quad \forall t \in [\tau_k, \tau_k + T_f] \\ q_b^*(\tau_k) = q_b^*(\tau_{k-1} + T_c) \\ u_b^*(\tau_k) = u_b^*(\tau_{k-1} + T_c) \\ q_b^*(\tau_k + T_f) = q_{b,goal} \\ u_b^*(\tau_k + T_f) = u_{b,goal} \\ |u_{b,i}^*(t)| \leq u_{b,i,max}, \quad \forall i \in [1,p], \forall t \in (\tau_k, \tau_k + T_f) \\ d(R_b, O_m) \geq 0, \quad \forall O_m \in \mathcal{O}_b, \forall t \in (\tau_k, \tau_k + T_f) \\ \mathrm{d}(R_b, R_c) - \rho_b - \rho_c \geq 0, \quad \forall R_c \in \mathcal{C}_b, \forall t \in (\tau_k, \tau_k + T_f) \\ \mathrm{d}(R_b, R_d) - \min(d_{b,com}, d_{d,com}) \geq 0, \quad \forall R_d \in \mathcal{D}_b, \\ \qquad \forall t \in (\tau_k, \tau_k + T_f) \\ \mathrm{d}(q_b^*(t), \hat{q}_b(t)) \leq \xi, \quad \forall t \in (\tau_k, \tau_k + T_f) \end{cases} \tag{15}$$

A possible definition for the $L_{b,f}$ cost function can be simply T_f. The sets $\mathcal{O}_b$, $\mathcal{C}_b$ and $\mathcal{D}_b$ are functions of τ_k.

3.3. Strategies for solving the constrained optimization problems

3.3.1. Flatness property

As explained in [6] all mobile robots consisting of a solid block in motion can be modelled as a flat system. This means that a change of variables is possible in a way that states and inputs of the kinematic model of the mobile robot can be written in terms of the new variable, called flat output (z), and its lth first derivatives. Thus, the behaviour of the system can be completely determined by the flat output.

Searching for a solution to our problem in the flat space rather than in the actual configuration space of the system present advantages. It prevents the need for integrating the differential equations of system and reduces the dimension of the problem of finding an optimal admissible trajectory. After finding (optimal) trajectories in the flat space it is possible to retrieve back the original configuration and input trajectories.

3.3.2. Parametrization of the flat output by B-splines

Another important aspect of this approach is the parametrization of the flat output trajectory. As done done in [11] the use of B-spline functions present interesting properties:

- It is possible to specify a level of continuity C^k when using B-splines without additional constraints.
- B-spline presents a local support, i.e., changes in parameters values have a local impact on the resulting curve.

The first property is very well suited for parametrizing the flat output since its lth first derivatives will be needed when computing the system actual state and input trajectories. The second property is important when searching for an admissible solution in the flat space; such parametrization is more efficient and well-conditioned than, for instance, a polynomial parametrization.

TODO cite

3.3.3. Optimization solver

There is a variety of numerical optimization packages implemented in many different programming languages available for solving optimization problems [12].

Obviously not all of those implementations are suited for solving the particular kind of optimization problems presented before.

As explained in **??** a Feasible Sequential Quadratic Programming is best suited for solving the motion planning problem as stated in thins paper. This class of solvers compute at every iteration solution that respects the problem constraints.

TODO keep explaining, present SLSQL and ALGENCAN, present reasons why we used SLSQP (i.e. availability in python, free license, and faster then ALGENCAN)

SLSQP numerical stability

4. Simulation results

Here we show the results and analyses founded for the motion planner presented in the previous sections.

The trajectory and velocities shown in the Figures 3 and 4 illustrate a motion planning solution found for a team of three robots. The robots move in an environment where three static obstacles are present. Each point along the trajectory line of a robot represents the beginning of a Tc computation horizon.

In Figure 3 we show the resulting plan when coupling constraints are ignored (Step 2 is never performed). In Figure 4 we have a collision-free solution. The blue zones in Figure 4 are in the same position as the red ones in Figure 3. Specially near these regions a change in the trajectory is present. Complementary, changes in the robots velocities across charts in both figures can be notice. Finally, the bottom charts show that the collisions were indeed avoid: inter-robot distances in Figure 4 are greater then or equal to zero all along the simulation.

For performing these two previous simulations a reasonable number of parameters have to be set. These parameters can be categorized into two groups. The **algorithm related** parameters and the **optimization solver related** ones. Among the former group, the most important ones are:

- The number of sample for time discretization (N_s);
- The number of internal knots for the B-splines curves (n_{knots});
- The planning horizon for the sliding window (T_p);
- The computation horizon (T_c).

The latter kind depends on the numeric optimization solver adopted. However, since most of them are iterative methods, it is common to have at least the two following parameters:

- Maximum number of iterations;
- Stop condition.

This considerable number of parameters having influence on the solution and/or on the time for finding a solution makes the search for a satisfactory set of parameters' values a laborious task.

Thus, it is important to have a better understanding of how some performance criteria are impacted by the changes in algorithm parameters.

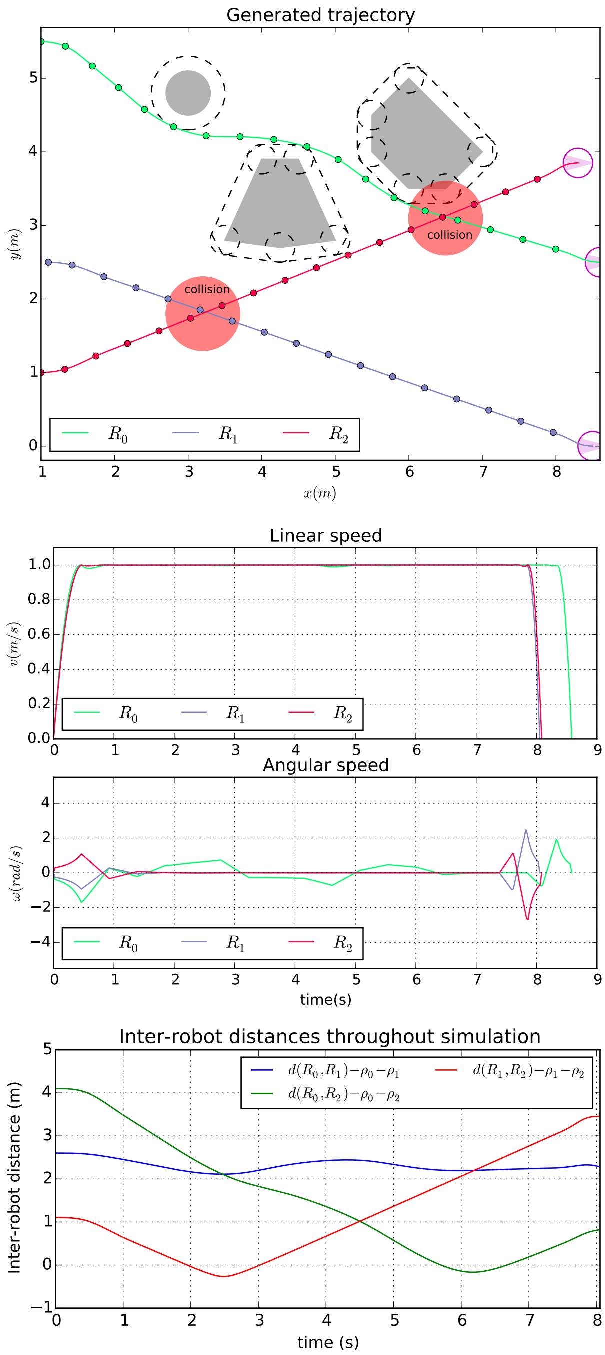


Figure 3. Motion planning solution without collision handling

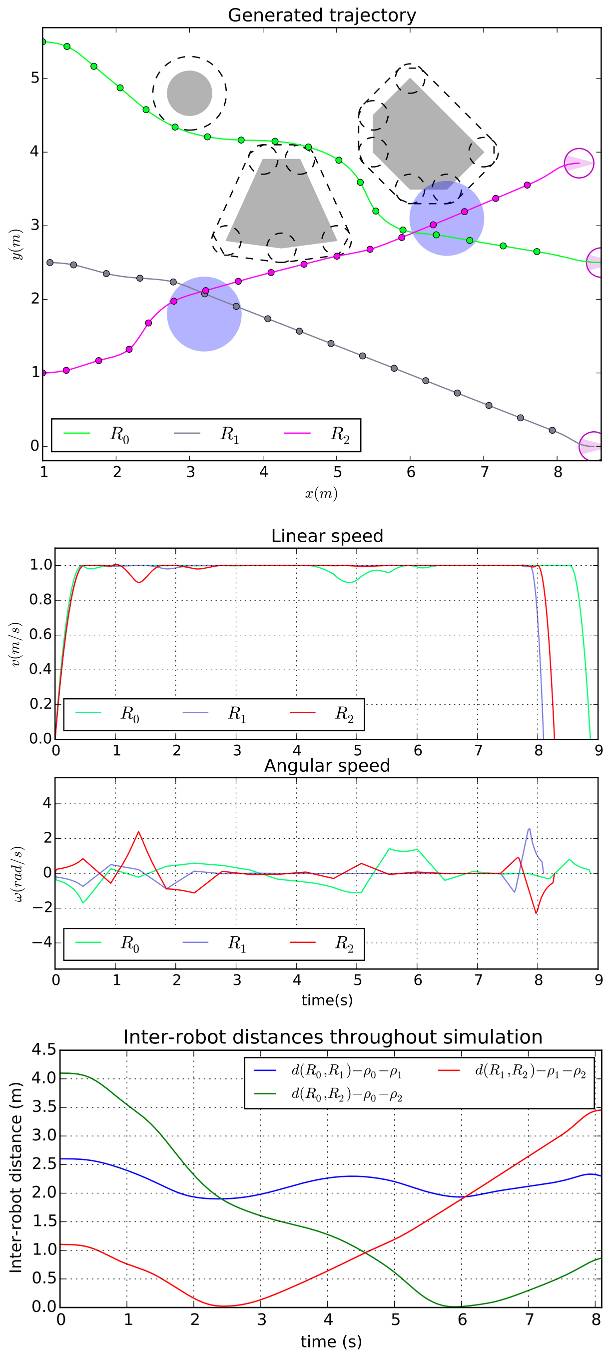


Figure 4. Motion planning solution with collision handling

4.1. Parameters' impact analyses

Three criteria considered important for the validation of this method were studied. We tested different parameters configuration and scenario in order to understand how they influence those criteria. The three criteria defined for a given robot R_b are:

- *Maximum computation time* over the computation horizon (MCT/T_c ratio).
- Obstacle penetration area (P).
- Total execution time (T_{tot}).

4.1.1. Detection radius impact

As the detection radius of the robot increases more obstacles are seen at once which, in turn, increases the number of constraints in the optimization problems. The impact of increasing the detection radius $d_{b,sen}$ in the computation performance (*Maximum computation time* over computation horizon MCT/T_c) can be seen in the Figure 5 for a scenario where seven obstacles where present. Naturally, the computation time stops increasing as the robot sees all obstacles present in the environment.

Furthermore, the Figure 6 shows how the total time

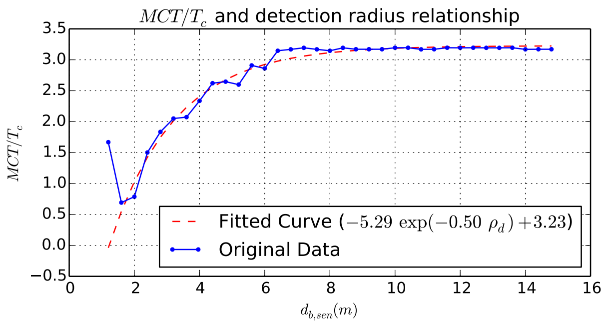


FIGURE 5. Increasing of detection radius and impact on a MTC/T_c ratio

of execution decreases as the radius increases pointing out how a better knowledge of the environment produces a more optimal solution.

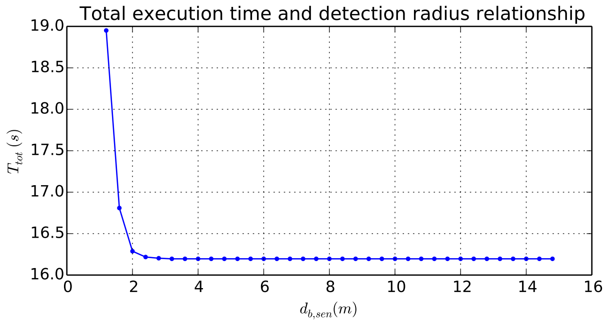


FIGURE 6. Increasing of detection radius and impact on a T_{tot} ratio

These two behaviors indicate how a compromise between computation time and optimality must be found.

4.2. Maximum computation time over computation horizon MCT/T_c

The significance of this criterion lays in the need of assuring the real-time property of this algorithm. In a real implementation of this approach the computation horizon would have always to be superior than the maximum time took for computing a plan (coupling constraints taken into account).

Based on several simulations with different scenarios we were able to produce the charts shown in the Figure 7

- SLSPQ method request $O(n^3)$ time, n being the number of knots;

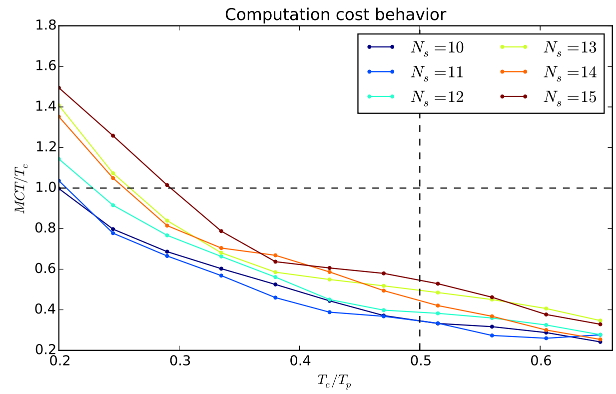
4.3. Obstacle penetration P

?? TODO rescale images

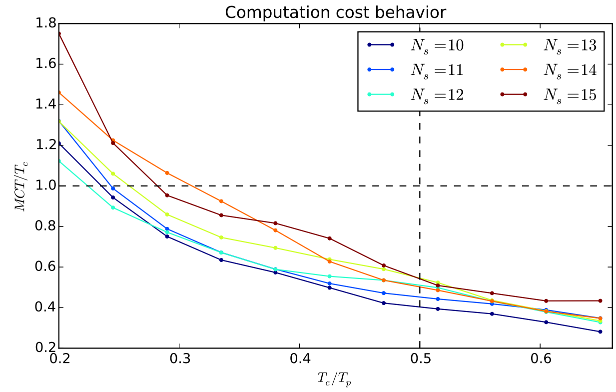
4.4. Total execution time T_{tot}

TODO Comparison with the other method;

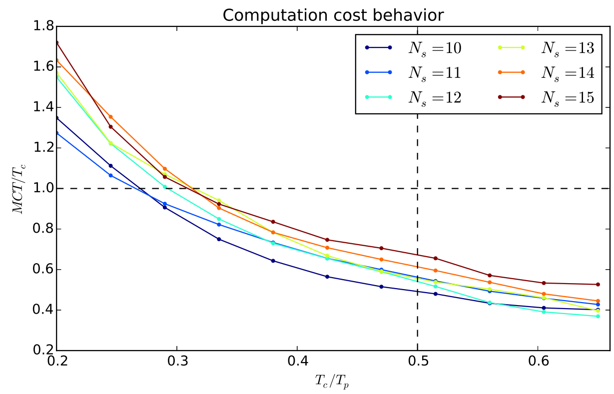
TODO Before concluding do comparison with other approach and make sure to have multi-robot stuff



(A) . Four internal knots. Average variance between lines is 1.047×10^{-2}



(B) . Five internal knots. Average variance between lines is 0.972×10^{-2}



(C) . Six internal knots. Average variance between lines is 0.587×10^{-2}

FIGURE 7. Three obstacles scenario

5. Conclusions

TODO perspectives

Analise influence of dynamics of system, sensors, communication latency;

References

[1] Autonomous robots are helping to pack your Amazon orders. `http://www.gizmag.com/amazon-kiva-fulfillment-system/34999/`. Accessed: 2015-07-22.

[2] Idea Groupe met en place Scallog pour sa

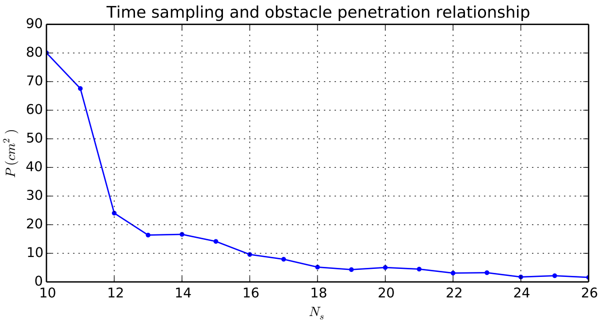


FIGURE 8. Obstacle penetration decreasing as sampling increasesfig:pen

préparation de commandes. `http://supplychainmagazine.fr/NL/2015/2085/`. Accessed: 2015-07-22.

[3] J. Schwartz, M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence* **37**(1-3):157–169, 1988. DOI:10.1016/0004-3702(88)90053-7.

[4] J.-C. Latombe. *Robot Motion Planning*, vol. 124. Springer US, Boston, MA, 1991. DOI:10.1007/978-1-4615-4022-9.

[5] D. Fox, W. Burgard, S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* **4**(1):23–33, 1997. DOI:10.1109/100.580977.

[6] M. Defoort. Contributions à la planification et à la commande pour les robots mobiles coopératifs. *Ecole Centrale de Lille* 2007.

[7] A. Kelly, B. Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *int j robot res* **22**(7):583–601, 2003. DOI:10.1177/027836403128965277.

[8] M. Defoort, A. Kokosy, T. Floquet, et al. Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A distributed receding horizon approach. *Robotics and Autonomous Systems* **57**(11):1094–1106, 2009. DOI:10.1016/j.robot.2009.07.004.

[9] C. Ericson. *Real-Time Collision Detection*. M038/the Morgan Kaufmann Ser. in Interactive 3D Technology Series. Taylor & Francis, 2004.

[10] T. Keviczky, F. Borrelli, G. J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica* **42**(12):2105–2115, 2006. DOI:10.1016/j.automatica.2006.07.008.

[11] M. B. Milam. *Real-time optimal trajectory generation for constrained dynamical systems*. Ph.D. thesis, California Institute of Technology, 2003.

[12] R. E. Perez, P. W. Jansen, J. R. R. A. Martins. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Structures and Multidisciplinary Optimization* **45**(1):101–118, 2012. DOI:10.1007/s00158-011-0666-3.