Acta Polytechnica **56**(1):10–17, 2016
doi:10.14311/APP.2016.56.0010

available online at http://ojs.cvut.cz/ojs/index.php/ap

MULTI-ROBOT MOTION PLANNING: A MODIFIED RECEDING HORIZON APPROACH FOR REACHING GOAL STATES

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 an algorithm for collision-free motion planning based on a receding horizon approach, for the navigation of a team of mobile robots in the presence of obstacles of different shapes. The method is simulated with three robots. The impact of the parameters is studied with regard to computation time, obstacle avoidance and travel time.

Keywords: multi-robot motion planning; nonholonomic mobile robot; distributed planning; receding horizon.

1. Introduction

The control of mobile robots is a long-standing subject of research in the robotics domain. A trending application of mobile robotic systems is their use in industrial supply chains for processing orders and optimizing the storage and distribution of products. For example, Amazon employs the Kiva mobile multi-robot system, and the logistic provider IDEA Group employs the Scallog system for autonomously processing client orders [1, 2]. Such logistics tasks have become increasingly complex as sources of uncertainty, such as human presence, are admitted in the work environment.

One basic requirement for such mobile multi-robot systems is the capacity of motion planning, that is, generating an admissible configuration and input trajectories connecting two arbitrary states. To solve the motion planning problem, various constraints must be taken into account, in particular, the robot's kinematic and geometric constraints.

The first constraints derive directly from the mobile robot architecture implying, in particular, nonholonomic constraints. Geometric constraints result from the need to prevent the robot assuming specific configurations in order to avoid collisions, communication loss, 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 efficient with respect to the travel time (the amount of time to go from the initial configuration to the goal).

A great amount of work towards collision-free motion planning for cooperative multi-robot systems has been proposed. This work can be split into centralized and distributed approaches. Centralized approaches are usually formulated as an optimal control problem that takes all robots in the team into account at once. This produces solutions closer to the optimal one than distributed approaches. However, the computation time, security vulnerability and communication requirements can make it impracticable, specially for a great number of robots [3].

Distributed methods based in probabilistic [4] and artificial potential fields [5] approaches, for instance, are computationally fast. However, they deal with collision avoidance as a cost function to be minimized. But rather than having a cost that increases as paths leading to collision are considered, collision avoidance should to be considered as hard constraints of the problem.

Other distributed algorithms are based on receding horizon approaches. In [6], a brief comparison of the main distributed receding methods is made, and the base approach extended in our work is presented. In this approach each robot optimizes only its own trajectory at each computation/update horizon. In order to avoid robot-to-robot collisions and loss of communication, neighbor robots exchange information about their intended trajectories before performing the update. Intended trajectories are computed by each robot ignoring constraints that take the other robots into account.

Identified drawbacks of this approach are the dependence on several parameters for achieving real-time performance and good solution optimality, the difficulty to adapt it for handling dynamic obstacles, the impossibility of bringing the robots to a precise goal state and the limited geometric representation of obstacles.

Therefore, in this paper, we propose a motion planning algorithm that extends the approach presented in [6]. In this modified algorithm, goal states can be precisely reached and more complex forms of obstacles can be handled. Furthermore, we investigate how the method's parameters impact a set of performance criteria. Thus, this distributed algorithm is able to find collision-free trajectories and

computes the corresponding angular and longitudinal velocities for a multi-robot system in the presence of static obstacles perceived by the robots as they evolve in their environment.

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 section explains the algorithm for resolving the motion planning problem and makes some remarks on how to resolve the constrained optimization problems associated with the method. The fourth section is dedicated to the results found using this method and analyses the specific performance criteria and how they are impacted by the algorithm parameters. Finally, in last section we present our conclusions and perspectives.

2. Problem Statement

2.1. Assumptions

In the development of the approach presented in this paper, 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 $t_k \in [t_{\text{init}},\ t_{\text{final}}]$, any obstacle O_m is considered detected by the robot R_b whenever the distance between their geometric centers is less than or equal to the detection radius $d_{b,\text{sen}}$ of the robot R_b. Therefore, this obstacle O_m 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 the geometric representation of a detected obstacle, i.e., obstacle perception issues are neglected.

(8.) A robot can access information about any robot in the team by using a wireless communication link. Latency, communication outages and other problems associated to this communication link are neglected.

(9.) The dynamic model of the multi-robot systems is neglected.

(10.) The input (or control) vector of a mobile robot R_b is bounded.

2.2. Constraints and cost functions

After giving the assumptions in the previous subsection, we can define the constraints and the cost function for multi-robot navigation.

(1.) The solution of the motion planning problem for 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 $u_b^*(t) \in \mathbb{R}^p$ being the solution trajectory for the robot's input – must satisfy the robot kinematic model equation:

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

where $f\colon \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is the vector-valued function modeling the robot kinematics.

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

$$q_b^*(t_{\text{init}}) = q_{b,\text{init}}, \tag{2}$$
$$u_b^*(t_{\text{init}}) = u_{b,\text{init}}. \tag{3}$$

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

$$q_b^*(t_{\text{final}}) = q_{b,\text{goal}}, \tag{4}$$
$$u_b^*(t_{\text{final}}) = u_{b,\text{goal}}. \tag{5}$$

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

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

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

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

where $L_b(q_b(t), u_b(t), q_{b,\text{goal}}, u_{b,\text{goal}})$ is the integrated cost for one robot motion planning [6].

(6.) To ensure avoidance of collisions 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 [7] R_b is located.

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

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

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 for a communication link between two robots (R_b, R_c) yields the following constraint:

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

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

3. Distributed motion planning

3.1. Receding horizon approach

Since the environment is perceived progressively by the robots, and new obstacles may appear as time passes, planning the whole motion from initial to goal configurations before the beginning of the motion is not a satisfying approach. Planning locally and replanning is more suitable for taking new information into account as it comes. Besides, the computation cost of finding a motion plan using the first approach may be prohibited high if the planning complexity depends on the distance between the start and goal configurations.

Therefore, an on-line motion planner is proposed. In order to implement it, a receding horizon control approach [8] is used.

Two fundamental concepts of this approach are the planning horizon T_p and the update/computation horizon 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. The problem of producing a motion plan during a T_c time interval is called here a receding horizon planning problem.

For each receding horizon planning problem, the following steps are performed:

Step 1. All robots in the team compute an intended solution trajectory (denoted $(\hat{q}_b(t), \hat{u}_b(t))$) by solving a constrained optimization problem. Coupling constraints (9) and (10), which involve other robots in the team, are ignored.

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

All robots in the team use the same T_p and T_c for assuring synchronization when exchanging information about their positions and abouth their intended trajectories.

For each of these steps and for each robot in the team, one constrained optimization problem is resolved. The cost function to be minimized in these optimization problems is the geodesic distance of a robot's current configuration to its goal configuration. This assures that the robots are driven towards their goal.

This two step scheme is explained in detail in [6, 9], where constrained optimization problems associated to the receding horizon optimization problem are formulated.

However, constraints related to the goal configuration and the goal input of the motion planning problem are neglected in their method. Constraints (4) and (5) are left out of the planning. To take them into account, a termination procedure is proposed in the following that enables the robots to reach their goal state.

3.2. Motion planning termination

After stopping the receding horizon planning algorithm, we propose a termination planning that considers those constraints related to the goal state. This enables the robots to reach their goal states.

The criterion used to pass from the receding horizon planning to the termination planning is based on the distance between the goal and the current position of the robots. It is defined by the equation 11:

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

This condition ensures 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.

Before solving the termination planning problem new parameters for representing and computing the solution are calculated by taking into account the estimated remaining distance and the typical distance traveled for a T_p planning horizon. This is done in order to rescale the configuration intended for a previous planning horizon not necessarily equal to the new one. Potentially, this rescaling will decrease the computation time for the termination planning.

The following pseudo code 1 summarizes the planning algorithm, and Figure 1 illustrates how plans would be generated through time by the algorithm.

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

PLANLASTSEC is the procedure for solving the termination planning problem. This problem is similar to the receding horizon planning problems.

It also has the two steps already presented for computing an intended plan and for updating it, if need be, so that conflicts are avoided.

The difference consists in how the optimization problems associated to it are defined. The optimization problem defined in (12) and (13) is the problem solved at the first step. The optimization problem associated with the second step is defined (14) and (15). Besides, in both new constrained optimal problems, the planning horizon is not a fixed constant as before, instead it is a part of the solution to be found.

Then, for generating the intended plan the following is resolved:

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

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

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

And for generating the final solution:

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

under the following constraints:

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

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

Algorithm 1 Motion planning algorithm

```
1: procedure PLAN
2:     q_latest ← q_initial
3:     d_rem ← |POS(q_final) − POS(q_latest)|
4:     while d_rem ≥ d_min + T_c · v_max do
5:         INITSOLREPRESENTATION(···)
6:         q_latest ← PLANSEC(···)
7:         d_rem ← |POS(q_final) − POS(q_latest)|
8:     end while
9:     RESCALEREPRESENTATION(···)
10:    T_f ← PLANLASTSEC(···)
11: end procedure
```

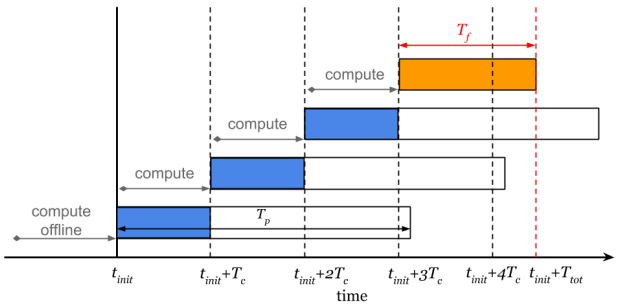


FIGURE 1. Receding horizon scheme with termination plan. The timespan T_f represents the duration of the plan for reaching the goal configuration.

3.3. STRATEGIES FOR SOLVING THE CONSTRAINED OPTIMIZATION PROBLEMS

3.3.1. FLATNESS PROPERTY

As explained in [9], all mobile robots consisting of a solid block in motion can be modeled as a flat system. This means that a change of variables is possible in a way that enables states and inputs of the kinematic model of the mobile robot to be written in terms of a new variable, called the flat output (z), and its lth first derivatives. The value of $l \mid l \le n$ depends on the kinematic model of the mobile robot. Therefore, the flat output can completely determine the behavior of the system.

Searching for a solution to our problem in the flat space rather than in the actual configuration space of the system presents advantages. It prevents the need for integrating the differential equations of the system (constraint 1) 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 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 in [10], the use of B-spline functions presents interesting properties.

- It is possible to specify a level of continuity C^k when using B-splines without additional constraints.

- A B-spline presents a local support – changes in the values of the parameters 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 actual state of the system and the input trajectories. The second property is important when searching for an admissible solution in the flat space; such parametrization is more efficient and better-conditioned than, for instance, a polynomial parametrization [10].

This choice for parameterizing the flat output introduces a new parameter to be set in the motion planning algorithm, i.e. the number of non-null knots intervals (denoted simply N_{knots}). This parameter plus the l value determines how many control points will be used for generating the B-splines.

3.3.3. Optimization solver

The optimization problems associated with finding the solution $q^*(t), u^*(t)$ are solved using a numerical optimization solver. For all time dependent constraints, time sampling is used. This introduces a new parameter into the algorithm: time sampling for optimization N_s. Each constraint that must be satisfied $\forall t \in (\tau_k, \tau_k + T_f)$ implies in N_s equations.

The need for a solver that supports nonlinear equality and inequality constraint restricts the number of numerical optimization solvers to be considered.

For our initial implementation of the motion planning algorithm, the SLSQP optimizer stood out as a good option. Besides being able to handle nonlinear equality and inequality constraint, its availability in the minimization module of the open-source scientific package Scipy [11] facilitates the motion planner implementation.

However, an error was experienced using this optimizer, which uses the SLSQP Optimization subroutine originally implemented by Dieter Kraft [12]. As the cost function value becomes too high (typically for values greater than 10^3), the optimization algorithm finishes with the "Positive directional derivative for linesearch" error message. This appears to be a numerical stability problem also experienced by other users as discussed in [13].

To work around this problem, we proposed a change in the objective functions of the receding horizon optimization problems. This change aims to keep the evaluated cost of the objective function around a known value when close to the optimal solution, instead of having a cost depending on the goal configuration (which can be arbitrarily distant from the current position).

We simply exchanged the goal position point in the cost function by a new point computed as follows:

$$p_{b,\text{new}} = \frac{p_{b,\text{goal}} - p_b(\tau_{s-1} + T_c)}{\text{norm}(p_{b,\text{goal}} - p_b(\tau_{s-1} + T_c))} \alpha T_p v_{b,\max},$$

where $p_{b,\text{goal}}$ and $p_b(\tau_{s-1} + T_c)$ are the positions associated with configurations $q_{b,\text{goal}}$ and $q_b(\tau_{s-1}+T_c)$ respectively, $\alpha \mid \alpha \geq 1, \alpha \in \mathbb{R}$ is a constant for controlling how far from the current position the new point is placed, the product $T_p v_{b,\max}$ is the maximum possible distance covered by R_b during a planning horizon, and $s \mid s \in [0, k), s \in \mathbb{N}$ is the current receding horizon problem index.

4. Simulation results

The results and their analysis for the motion planner presented in the previous sections are presented here.

The trajectory and the velocities shown in Figures 2 and 3 illustrate a motion planning solution found for a team of three robots. They plan their motion in an environment where three static obstacles are present. Each point along the trajectory line of a robot represents the beginning of a Tc update/computation horizon.

These figures show the planner generates the configuration and the input trajectories satisfying the constraints associated with the goal states.

In particular, in Figure 2, the resulting plan is computed ignoring coupling constraints (Step 2 is never performed) and consequently two points of collision occur. A collision-free solution is presented in Figure 3. Specially near the regions where collisions occurred, a change in the trajectory is present from Figure 2 to Figure 3 to avoid a collision. At the same time, changes in the (linear) velocities of the robots across the charts in both figures can be observed. Finally, the charts at the bottom show that the collisions were indeed avoided: the inter-robot distances in Figure 3 are greater than or equal to zero all along the simulation.

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

- the number of samples for time discretization (N_s);
- the number of internal knots for the B-spline curves (N_{knots});
- the planning horizon for the sliding window (T_p);
- the computation horizon (T_c).
- the detection radius of the robot (d_{sen}).

The optimization related parameters depend on the numeric optimization solver adopted. However, since most of them are iterative methods, it is common to have at least a maximum number of iterations parameter and a stop condition parameter.

This considerable number of parameters makes the search for a satisfactory set of parameter values a laborious task.

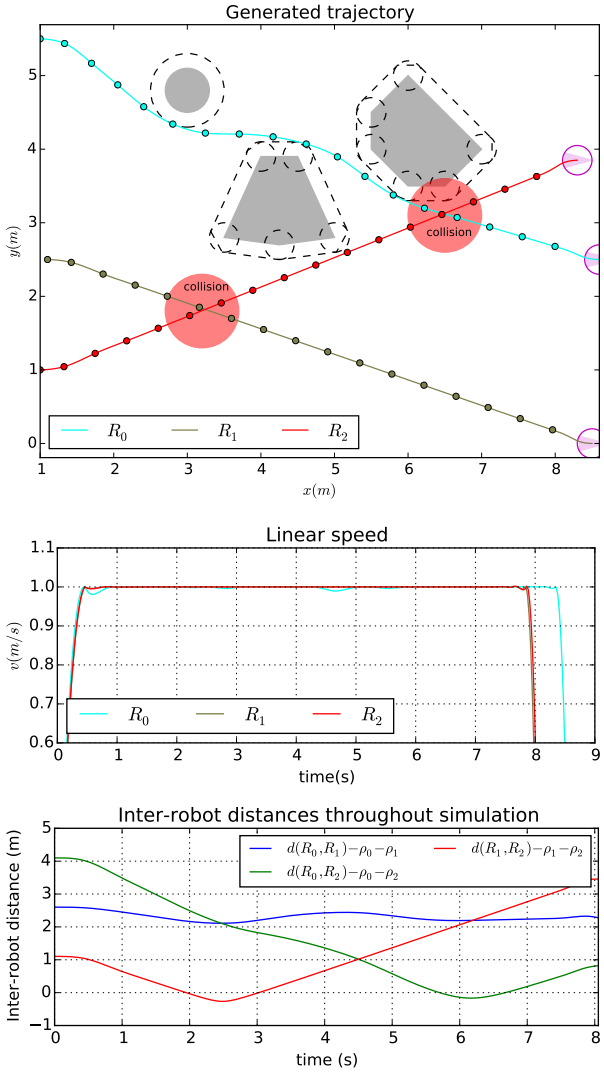


FIGURE 2. Motion planning solution without collision handling.

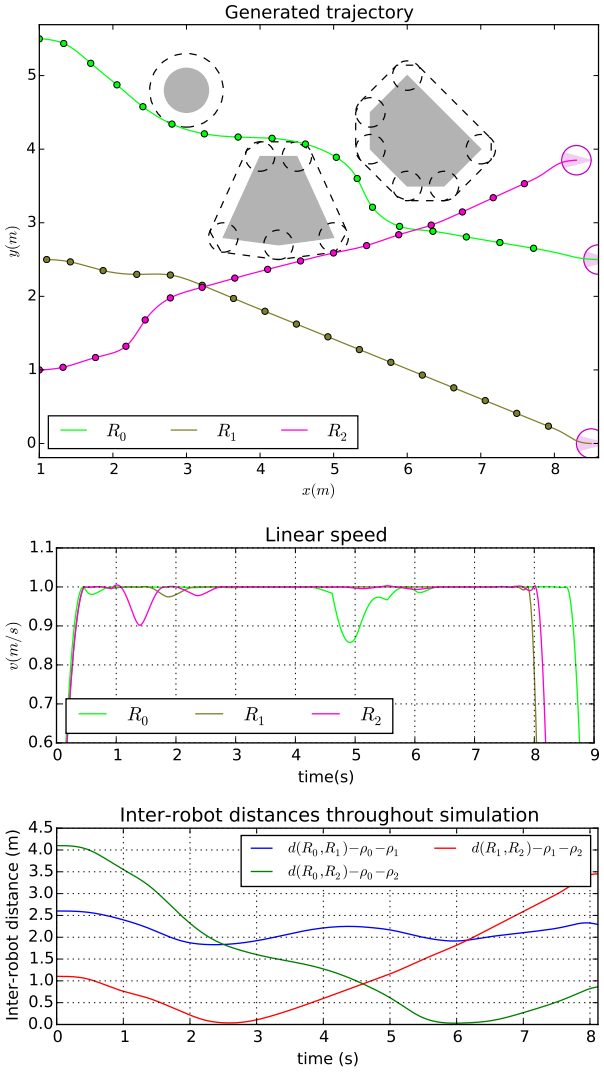


FIGURE 3. Motion planning solution with collision handling.

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

4.1. IMPACT OF PARAMETERS

Three criteria considered important for validating this method were studied: the maximum computation time during the planning over the computation horizon (MCT/T_c ratio); the obstacle penetration area (P); the travel time (T_{tot}). Different parameters configuration and scenarios where tested in order to highlight how they influence those criteria.

4.1.1. MAXIMUM COMPUTATION TIME OVER COMPUTATION HORIZON MCT/T_c

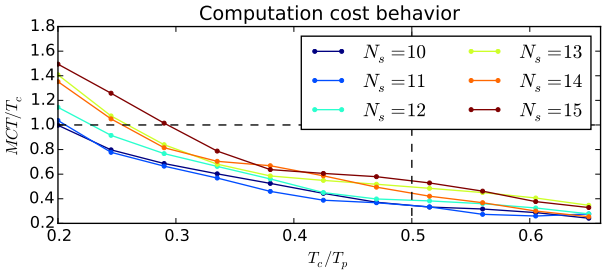
The significance of this criterion lies in the need to assure the real-time property of this algorithm. In a real implementation of this approach the computation horizon would always have to be superior to the maximum time taken to compute a plan.

Table 1 summarizes one of the scenarios studied for a single robot. The results obtained from simulations in this scenario are presented in Figure 4, for various parameters.

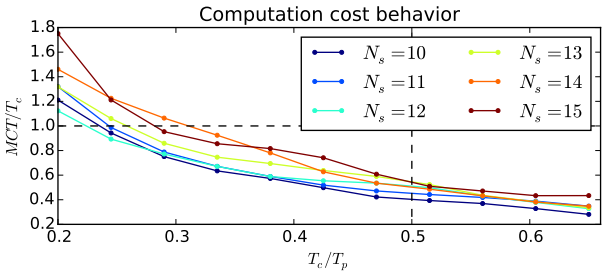
Each dot along the curves corresponds to the average of MCT/T_c along different T_p's for a given value of (T_c/T_p, N_s). The absolute values observed in the charts depend on the processing speed of the machine in which the algorithm is run. These simulations were run on an Intel Xeon CPU 2.53GHz processor.

Rather than observing the absolute values, it is interesting to analyze the impact of changes in the values of the parameters. In particular, an increasing number of N_s increases MCT/T_c for a given T_c/T_p. Similarly, an increase in MCT/T_c as the number of internal knots N_{knots} increases from charts 4a to 4c is observed.

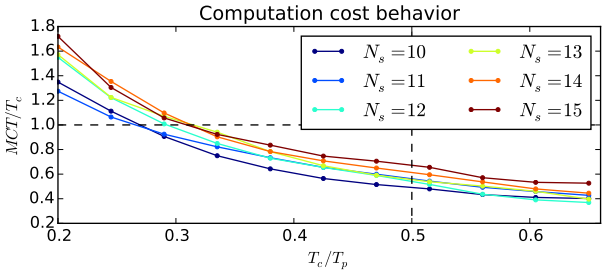
Further analyses of the data show that finding the solution using the SLSPQ method requires $O(N_{\text{knots}}^3)$ and $O(N_s)$ time. Although augmenting N_{knots} can



(A) . Four internal knots



(B) . Five internal knots



(C) . Six internal knots

FIGURE 4. Three obstacles scenario simulations

TABLE 1. Values for scenario definition

$v_{\max}$	1.00 m/s
$\omega_{\max}$	5.00 rad/s
q_{initial}	$[-0.05\ 0.00\ \pi/2]^T$
q_{final}	$[0.10\ 7.00\ \pi/2]^T$
u_{initial}	$[0.00\ 0.00]^T$
u_{goal}	$[0.00\ 0.00]^T$
O_0	[0.55 1.91 0.31]
O_1	[−0.08 3.65 0.32]
O_2	[0.38 4.65 0.16]

lead to an impractical computation time, typical N_{knots} values did not need to exceed 10 in our simulations, which is a sufficiently small value.

Another parameter having a direct impact on the MCT/T_c ratio is the detection radius of the robot's sensors. 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

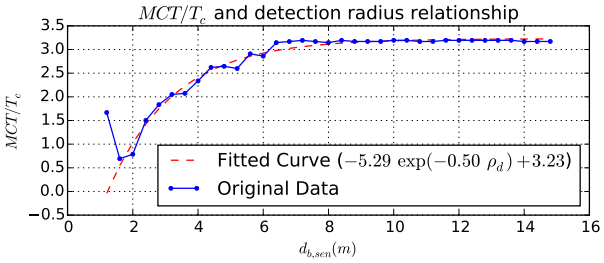


FIGURE 5. Increasing the detection radius, and impact on a MTC/T_c ratio

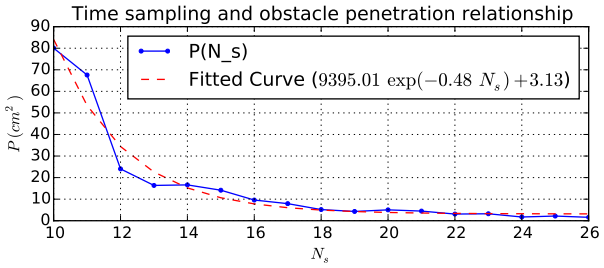


FIGURE 6. Obstacle penetration decreasing as sampling increases

radius d_{sen} in the MCT/T_c ratio can be seen in Figure 5 for a scenario with seven obstacles. The computation time stops increasing as soon as the robot sees all obstacles present in the environment.

4.1.2. Obstacle penetration P

Obstacle penetration area P gives a metric for obstacle avoidance and consequently for the solution quality. A solution where the planned trajectory does not pass through an object at any instant of time gives $P = 0$. The solution quality decreases with increasing P. However, since time sampling is performed during the optimization, P is usually greater than zero. A way of assuring $P = 0$ would be to increase the radius of the obstacles computed by the robot's perception system by the maximum distance that the robot can run within the time span T_p/N_s. However simple, this approach represents a loss of optimality and is not considered in this work.

It is relevant then to observe the impact of the algorithm parameters in the obstacle penetration area. T_c/T_p ratio, N_{knots} and d_{sen} impact on this criteria are only significant for degraded cases, meaning that around typical values those parameters do not change P significantly. However, time sampling N_s is a relevant parameter. Figure 6 shows the penetration area decreasing as the number of samples increases.

4.1.3. Travel time T_{tot}

Another complementary metric for characterizing solution quality is the travel time T_{tot}. Analyses of data from several simulations show a tendency that for a given value of N_{knots}, N_s and T_c the travel time decreases as the planning horizon T_p decreases. This can be explained by the simple fact that for a given

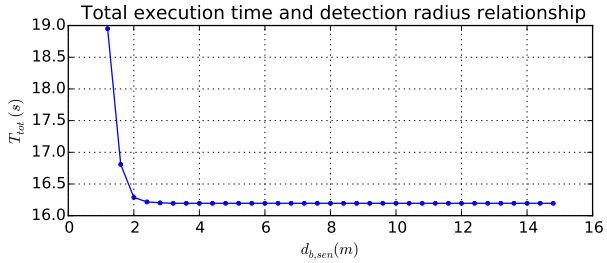


FIGURE 7. Increasing the detection radius, and its impact on T_{tot}

T_c, a better solution (in terms of travel time) can be found if the planning horizon T_p is smaller. Another relevant observation is that the overall travel time is shorter for smaller N_s's. This misleading improvement does not take into account the fact that the fewer the samples the greater will be the obstacle penetration area, as shown previously in Figure 6.

Furthermore, Figure 7 shows travel time invariance for changes in the detection radius far from degraded values that are too small. This indicates that local knowledge of the environment provides enough information for finding good solutions.

5. CONCLUSIONS

We have proposed a distributed motion planner based on a receding horizon approach, modified to take into account termination constraints. Near the goal configuration neighborhood, the receding horizon approach is finished and a termination planning problem is solved for bringing the robots to their precise final state. The problem is stated as a constrained optimization problem. It minimizes the time for reaching a goal configuration through a collision-free trajectory securing communication between robots. Circle and convex polygon representation of obstacles is supported. Key techniques for implementing the motion planner are: system flatness property, B-spline parametrization of the flat output and the SLSQP optimizer. Finally, solutions using this planner for different scenarios were generated in order to validate the method. The impact of different parameters on computation time and on the quality of the solution was analyzed. Future work will be performed in a physical simulation environment, where the dynamics is taken into account as well as models of the sensors and communication latency.

REFERENCES

[1] S. Robarts. 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 préparation de commandes. `http://supplychainmagazine.fr/NL/2015/2085/`. Accessed: 2015-07-22.

[3] F. Borrelli, D. Subramanian, a.U. Raghunathan, L. Biegler. MILP and NLP Techniques for centralized trajectory planning of multiple unmanned air vehicles. *2006 American Control Conference* pp. 5763–5768, 2006. DOI:10.1109/ACC.2006.1657644.

[4] G. Sanchez, J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *The International Journal of Robotics Research* **21**(1):5–26, 2002. DOI:10.1177/027836402320556458.

[5] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*, pp. 396–404. Springer Science and Business Media, 1986. DOI:10.1007/978-1-4613-8997-2_29.

[6] 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.

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

[8] 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.

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

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

[11] SciPy - Scientific Computing Tools for Python. `http://www.scipy.org/`. Accessed: 2015-07-31.

[12] D. Kraft. *A software package for sequential quadratic programming*. DLR German Aerospace Center – Institute for Flight Mechanics, Koln, Germany, 1988.

[13] Runtime errors for large gradients. `http://comments.gmane.org/gmane.science.analysis.nlopt.general/191`. Accessed: 2015-07-27.