Projet de Fin d'Études (PFE)

Ingénierie système : robotique et systèmes embarqués

2014/2015

Réf : DIASI / 15-351

Decentralized Motion Planning for Multi-robot System in Human Environment

Unclassified Report

Can be made public on the internet

Author:

José Magno Mendes Filho — Promotion 2014

Advisor - ENSTA:
David Filliat

Advisor - CEA:
Éric Lucet

Internship from 05 Mars 2015 to 28 August 2015

CEA LIST Digiteo Moulon

Bât. 660 91191 GIF-SUR-YVETTE Cedex, France

Acknowledgements

Firstly, I would like to express my sincere gratitude to both of my advisors, to Prof. Dr. David Filliat who welcomed me in the U2IS - Unité Informatique et Ingénierie des Systèmes at ENSTA for the two first months of my internship as well as to Dr. Eric Lucet, my advisor at CEA, for the continuous support and incentive of my work during the whole internship.

My sincere thanks also goes to all my colleagues at both U2IS and CEA for all the coffee breaks and stimulating discussions (sometimes even related to our works). They helped to make these last six months a very pleasant time.

Last but not the least, I must not forget to thank my family who always supported my life and career choices no matter how strange they may have seemed to them.

Abstract

This work 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 method's parameters is studied with regard to critical performance criteria, being mainly computation time, obstacle avoidance and travel time.

Résumé

Ce travail propose la mise en oeuvre d'un algorithme "real-time" de planification de trajectoire avec évitement de collision basé sur le concept de fenêtre glissante. Il est destiné à l'évolution autonome d'une flottille des robots mobiles dans un contexte industriel en présence de différents types d'obstacles. La méthode est validée en simulation pour un système de trois robots. Finalement, l'impact de paramètres de la méthode sur des critères de performance critiques, notamment le temps de calcul, l'évitement d'obstacle et le temps total de déplacement.

Contents

Chapter 1

Introduction

This document describe the work developed during my final year internship at CEA as an engineering student from ENSTA ParisTech and UPMC - Paris VI. The internship subject was named "Replanification dynamique locale de trajectoire dun cariste autonome en milieu humain" and was proposed by the "Laboratoire de Robotique Interactive" at CEA.

Due to some delay imposed by administrative and security procedures at CEA the first two months of my work, from beginning of Mars until end of April, took place at the "Unité Informatique et In- génierie des Systèmes" at ENSTA under the supervision of Prof. Dr. David Filliat. Later on, from May until the end of August, I worked at CEA LIST Digiteo Moulon under the supervision of Dr. Eric Lucet.

1.1 Internship context

The work developed during this internship falls within the context of an applied research project on automation of a forklift truck fleet for the effective supply of assembly lines where human can be present.

Therefore, the autonomous forklift trucks have to be able to go from an initial to a goal configuration in an environment that is partially known (at a given moment) while efficiently avoiding collisions with obstacles (e.g., boxes, shelves and other robots) and above all else avoiding collisions with humans. Put in other words, the problem in hand is a collision-free motion planning problem for a cooperative multi-robot system.

1.2 Objectives

The main objective of this internship is to implement, test and evaluate a motion planner applicable to the scenario described before. In order to do so we based ourselves mainly in the work presented in [3].

As stated in [3] compared with other solutions this approach presents good advantages for multi-robots systems evolving in the presence static obstacles.

The two main challenges that may be confronted during this work are how to insure real-time performance for our specific application and how to generalize the algorithm in order to account for dynamic obstacles (such as humans).

1.3 Related work

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

Decentralized methods based in probabilistic [?] and artificial potential fields [?] approaches, for instance, are computationally fast. However, they are inapplicable to real-live scenarios. 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, for security sake, collision avoidance has to be considered as a problem's constraint.

Another group of decentralized algorithms are based on receding horizon approaches. In [?] a brief comparison of the main decentralized receding methods is made as well as the presentation of the base approach extended in our work. In this approach each robot optimizes only its own trajectory at each computation/update horizon. In order to avoid robot-to-robot collisions and lost of communication, neighbors 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. Those trajectories are computed by solving nonlinear optimization problems [1] using flatness property to reduce the size of the problem and B-splines for representing the flat output [8].

Identified drawbacks of this approach presented in [?] 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.

1.4 Report outline

The rest of this final internship report is structure as follows. In Chapter 2 we make some assumptions about our particular motion problem in order to be able to construct a set of constraints and cost function that will characterize our problem.

In Chapter 3 the our distributed local motion planning algorithm is developed, giving emphasis to where it differs from previous work and to important implementation techniques.

Chapter 4 presents the examples of solutions that can be generated by the developed motion planner, some results obtained after implementing the algorithm in a physics simulation environment and a performance analisys of the algorithm according to its parametrization.

The fifth and last chapter presents our conclusions and perspectives about this work.

Chapter 2

Problem Statement

Let us present some assumptions about the problem in hand. These assumptions and the notation established as we state them will help to model the motion planning problem as set of nonlinear optimization problems (NPLs) to be solved.

2.1 Assumptions

1. The travel time 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. Their kinematic model can be written in the form:

$$\dot{q}(t) = f(q(t), u(t))$$

   where $q \in \mathbb{R}^n$ is the robot configuration vector and $u \in \mathbb{R}^p$ is the input vector.
3. A robot (denoted $R_b,\ R_b \in \mathcal{R},\ b \in \{0, \ldots, B-1\}$) is geometrically represented (for planning purposes) as a 2D object, a circle of radius ρ_b centered at (x_b, y_b).
4. The travel time of a single robot in the team starts at the instant $t_{b,init}$ and goes until the instant $t_{b,final} \leq t_{final}$.
5. Initial and goal configuration (q_{init}, q_{goal}) as well as their derivatives $(\dot{q}_{init}, \dot{q}_{goal})$ are known.
6. All obstacles in the environment are considered static. They can be represented by a set $\mathcal{O}$ of M static obstacles.
7. An obstacle (denoted $O_m,\ O_m \in \mathcal{O},\ \ m \in \{0,\ \ldots, M-1\}$) is geometrically represented (for planning purposes) either as a circle or as a convex polygon. In the case of a circular obstacle its radius is denoted r_{O_m} centered at (x_{O_m}, y_{O_m}).

8. For a given instant $t \in [t_{init},\ t_{final}]$, 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 the detected obstacles of R_b.

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

10. A given robot in the team can access any information known by another robot in the same team by using a wireless communication link.

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

12. Dynamics is neglected.

13. The input for each mobile robot R_b is limited, thus $|u_{b,i}(t)| \leq u_{b,i,max}, \quad \forall i \in [1, p], \forall t \in [t_{init},\ t_{final}]$.

14. The configuration and input trajectories that are solution for the motion planning problem are denoted $(q^*(t), u^*(t))$. Analogously, the solution trajectories for a particular robot in the team are denoted $(q_b^*(t), u_b^*(t))$.

2.2 Constraints and cost functions

Similarly to what is done in reference [?] a list of constraints that must be satisfied by the solution $(q^*(t), u^*(t))$ can be defined. Also, a cost for the multi-robot system navigation (function of the solution $(q^*(t), u^*(t))$) is stated.

1. The solution of the motion planning problem for each robot R_b $(q_b^*(t), u_b^*(t))$ must satisfy the robots' kinematic model equation:

$$\dot{q}_b^*(t) = f(q_b^*(t), u_b^*(t)), \quad \forall t \in [t_{init}, t_{final}],\ \forall b \in \{0, \ldots, B-1\}. \tag{2.2.1}$$

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

$$q_b^*(t_{init}) = q_{b,init}, \tag{2.2.2}$$

$$u_b^*(t_{init}) = u_{b,init},\ \forall b \in \{0, \ldots, B-1\}. \tag{2.2.3}$$

3. The planned final configuration and final input for each robot R_b must be equal to

the goal configuration and goal input for R_b:

$$q_b^*(t_{final}) = q_{b,goal}, \tag{2.2.4}$$

$$u_b^*(t_{final}) = u_{b,goal}, \ \forall b \in \{0, \ldots, B-1\}. \tag{2.2.5}$$

4. Practical limitations of the input impose the following constraint: $\forall t \in [t_{init}, t_{final}]$, $\forall i \in [1, 2, \cdots, p]$, $\forall b \in \{0, \ldots, B-1\}$,

$$|u_{b,i}^*(t)| \leq u_{b,i,max}. \tag{2.2.6}$$

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

$$L(q(t), u(t)) = \sum_{b=0}^{B-1} L_b(q_b(t), u_b(t), q_{b,goal}, u_{b,goal}) \tag{2.2.7}$$

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

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:

$$\mathrm{d}(R_b, O_m) \geq 0. \tag{2.2.8}$$

For the circular 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 [4] where R_b is located. Figure 2.1 shows an quadrilateral $ABCD$ representation of an obstacle where three of the nine regions were distinguished.

For these three regions, the distance robot-to-quadrilateral is computed as follows:

(a) If robot in region 1 ("in front of" the vertex A):

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

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

(b) If robot in region 2 ("in front of" the side DA):

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

where

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

The distance $d(s_{DA}, (x_b, y_b))$ represents the distance from the robot to the side DA and the constants $a_{s_{DA}}$, $b_{s_{DA}}$ and $c_{s_{DA}}$ are the line equation coefficients of the line s_{DA}.

(c) If robot in region 3 ("inside" the quadrilateral $ABCD$):

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

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

The distance computation for other regions are analogous to the equations in items 6a and 6b.

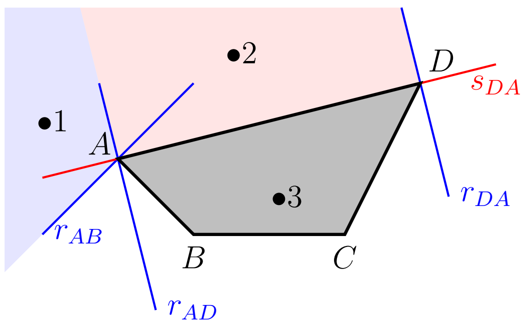


Figure 2.1 – Voronoi regions used for case differentiation.

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

$$d(R_b, R_c) - \rho_b - \rho_c \geq 0 \tag{2.2.9}$$

where $d(R_b, R_c) = \sqrt{(x_b - x_c)^2 + (y_b - y_c)^2}$ and $\mathcal{C}_{b,coll}$ 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:

$$d(R_b, R_c) - \min(d_{b,com}, d_{c,com}) \leq 0 \tag{2.2.10}$$

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

Chapter 3

Distributed local motion planning

As said before, each robot in the team has a detection radius ($d_{b,sen}$) for perceiving the obstacles in the environment. Consequently, as the robots move in their environment, they progressively perceive new obstacles. This poses a limitation about how to approach the problem of finding the solution $(q^*(t), u^*(t))$.

Finding a complete solution ($\forall t \in [t_init, t_final]$) based on the information acquired by the robots at the initial moment t_{init} may turn out to be unsatisfactory. Important information about the obstacles position may have not yet be acquired and will be neglected by this global approach.

Aiming to produce a local solution, valid for the motion performed in the near future and replanning afterwards is more suitable for taking new information, as they come, into account.

Besides, the computation cost of finding a motion plan using the a global planning approach may be prohibited high if the planning complexity depends on the distance between initial and goal configurations.

Another important notion about how to approach the problem is whether there is a way of dividing the work of finding the solution not only in time as discussed before (local vs. global), but also dividing it in space, among the planning agents composing the multi-robot system. In other words, whether is possible to have a decentralized (distributed) approach where each robot tries to find a part of the solution as opposed to a centralized one, where one central planning agent provided with the information acquired by all robots finds the solution.

In reference [3] all these possibilities are studied in details. For our specific context a distributed local approach was necessary We addressed locality by using a receding horizon [?] approach while decentralization is done by postponing the evaluation of coupling constraints, i.e., by performing a first step where each robot plans its own motion ignoring other robots in the team followed by a second step where they individually optimize their solution trajectory found in the first step by take those coupling constraints into account.

3.1 2-step Receding Horizon approach

Two fundamental concepts of the receding horizon approach are the planning horizon T_p and update/computation horizon T_c. T_p is the timespan for which a solution (plan) 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. Figure 3.2 helps to illustrate these two concepts.

For each receding horizon planning problem, the following steps are performed in order to make decentralization possible:

Step 1. Each robot in the team compute a intended solution trajectory (denoted $(\hat{q}_b(t), \hat{u}_b(t))$) by numerically solving a constrained optimization problem $(NLP_{b,1})$ that take into account all constraints listed in section TODO except the coupling constraints 2.2.9 and 2.2.10, and the goal state constraints **??** and **??**. Ignoring the coupling constraints is what enables the robot to compute its intended trajectory without taking into account the other robots in the team. Goal state constraints are ignored because the produced solution is local, valid for only the planning horizon T_p ahead in time.

Step 2. Robots involved in a potential conflict (that is, risk of collision or lost of communication) update their trajectories computed during 3.1 by solving another constrained optimization problem $(NLP_{b,2})$ that additionally takes into account coupling constraints (2.2.9 and 2.2.10). This is done by using the other robots' intended trajectories computed in the previous step as an estimate of those robots' final trajectories for that planning horizon T_p. If a robot is not involved in any conflict, 3.1 is not executed and its final solution trajectory is identical to the one estimated in 3.1. For the same reason as before, goal state constraints are still left out of the NPLs.

All robots in the team use the same T_c for assuring synchronization when exchanging information about their positions and intended trajectories. Although different T_p could be used for different robots, only tests with the same T_p were done.

For each of those two steps and for each robot in the team, one constrained optimization problem is then numerically resolved. The cost function to be minimized in those optimization problems is the 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 details in [?, 3] where the constrained optimization problems associated to the receding horizon planning problem are formulated. They are presented in this document, with some modifications, in the Appendix ??.

The resolution of this receding horizons problems is done interactively through time until the robots get close to their goal configurations.

The problem of reaching precisely the goal state is not addressed by this approach. Constraints ?? and ?? are left out of the planning process. For taking them into account, we proposed a termination procedure in the following that enables the robots to reach their goal state.

3.2 Motion planning termination

After stopping the 2-step receding horizon planning algorithm, we propose a termination planning that considers those constraints related to the goal state.

The criterion used to finish the receding horizon planning and start the termination planning is based on the distance between goal and current position of the robots. It is defined by the inequation 3.2.1:

$$d_{rem} \geq d_{min} + T_c \cdot v_{max} \tag{3.2.1}$$

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 the solution representation and computation are calculated by taking into account the estimate 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.

The following flowchart aims to

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

$$\min_{\hat{q}_b(t),\hat{u}_b(t),T_f} L_{b,f}(\hat{q}_b(t), \hat{u}_b(t), q_{b,goal}, u_{b,goal}) \tag{3.2.2}$$

under the following constraints for $\tau_k = kT_c$ with k the number of receding horizon problems

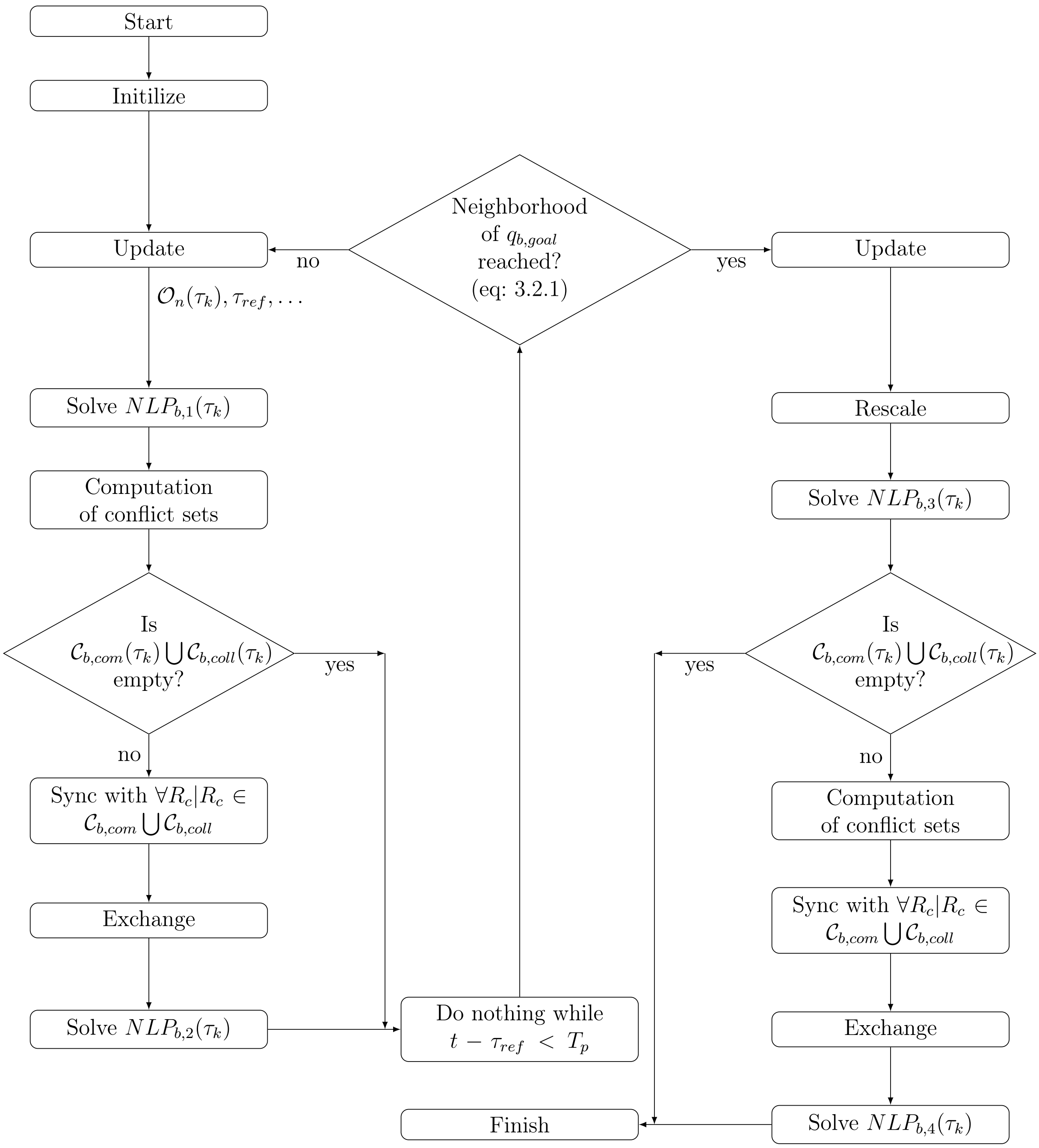


Figure 3.1 – Flowchart illustrating the distributed local motion planning

solved before the termination problem:

$$\begin{cases} \dot{\hat{q}}_b(t) = f(\hat{q}_b(t), \hat{u}_b(t)), \quad \forall t \in [\tau_k, \tau_k + T_f] \\ \hat{q}_b(\tau_k) = q_b^*(\tau_{k-1} + T_c) \\ \hat{u}_b(\tau_k) = u_b^*(\tau_{k-1} + T_c) \\ \hat{q}_b(\tau_k + T_f) = q_{b,goal} \\ \hat{u}_b(\tau_k + T_f) = u_{b,goal} \\ |\hat{u}_{b,i}(t)| \leq u_{b,i,max}, \quad \forall i \in [1,p], \forall t \in (\tau_k, \tau_k + T_f) \\ d(R_b, O_m) \geq 0, \quad \forall O_m \in \mathcal{O}_b, t \in (\tau_k, \tau_k + T_f) \end{cases} \tag{3.2.3}$$

And for generating the final solution:

$$\min_{q_b^*(t), u_b^*(t), T_f} L_{b,f}(q_b^*(t), u_b^*(t), q_{b,goal}, u_{b,goal}) \tag{3.2.4}$$

under the following constraints:

$$\begin{cases} \dot{q^*}_b(t) = f(q_b^*(t), u_b^*(t)), \quad \forall t \in [\tau_k, \tau_k + T_f] \\ q_b^*(\tau_k) = q_b^*(\tau_{k-1} + T_c) \\ u_b^*(\tau_k) = u_b^*(\tau_{k-1} + T_c) \\ q_b^*(\tau_k + T_f) = q_{b,goal} \\ u_b^*(\tau_k + T_f) = u_{b,goal} \\ |u_{b,i}^*(t)| \leq u_{b,i,max}, \quad \forall i \in [1, p], \forall t \in (\tau_k, \tau_k + T_f) \\ d(R_b, O_m) \geq 0, \quad \forall O_m \in \mathcal{O}_b, \forall t \in (\tau_k, \tau_k + T_f) \\ \mathrm{d}(R_b, R_c) - \rho_b - \rho_c \geq 0, \quad \forall R_c \in \mathcal{C}_b, \forall t \in (\tau_k, \tau_k + T_f) \\ \mathrm{d}(R_b, R_d) - \min(d_{b,com}, d_{d,com}) \geq 0, \quad \forall R_d \in \mathcal{D}_b, \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall t \in (\tau_k, \tau_k + T_f) \\ \mathrm{d}(q_b^*(t), \hat{q}_b(t)) \leq \xi, \quad \forall t \in (\tau_k, \tau_k + T_f) \end{cases} \tag{3.2.5}$$

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

procedure PLAN
- $q_{latest} \leftarrow q_{initial}$
- $d_{rem} \leftarrow |\text{POS}(q_{final}) - \text{POS}(q_{latest})|$
- **while** $d_{rem} \geq d_{min} + T_c \cdot v_{max}$ **do**
  - INITSOLREPRESENTATION$(\cdots)$
  - $q_{latest} \leftarrow$ PLANSEC$(\cdots)$
  - $d_{rem} \leftarrow |\text{POS}(q_{final}) - \text{POS}(q_{latest})|$
- **end while**
- RESCALEREPRESENTATION$(\cdots)$
- $T_f \leftarrow$ PLANLASTSEC$(\cdots)$

end procedure

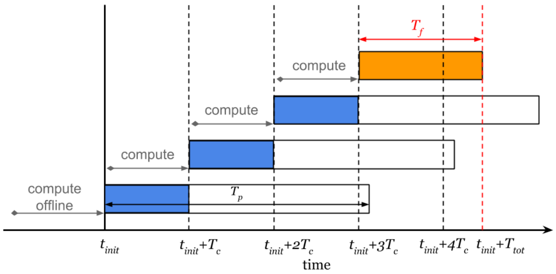


Figure 3.2 – Receding horizon scheme with termination plan. The timespan T_f represents the duration of the plan for reaching the goal configuration.

3.3 Strategies for solving the constrained optimization problems

3.3.1 Improving robot-to-polygon distance computation

Of course that the performance of this approach is "number of edges"-dependent and present fast results only or polygons with few edges (less than 10).

10 was arbitrary, improve this finding a meaningful value or delete it

An important remark though is that for a given planning horizon N_s point-to-obstacle distances have to be calculated. Intuitively we can say that there is a high probability that most of the N_s points are inside the same region defined by theirs relative positions to the obstacle. Besides, the probability of finding points inside regions that are "far" from the already occupied zones is smaller. This heuristic can be used to speed up the planning process by having a smarter initialization of point-to-obstacle distance computation when using a convex polygon representation.

Finally, when dealing with more complex obstacles representations and/or with a more complex representation of the mobile robot geometry the Enhanced Gilbert-Johnson-Keerthi distance algorithm [4] is a more suitable and efficient approach.

some code is available on the internet, Google code written in D language and/or the other one on stackoverflow, see bookmarks

3.3.2 Flatness property

As explained in [3], 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 states and inputs of the kinematic model of the mobile robot can be written in terms of a new variable, called flat output (z), and its lth first derivatives. The value of $l \mid l \leq n$ depends on the kinematic model of the mobile robot. Therefore, the flat output can completely determine 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 system (constraint 2.2.1) 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 back the original configuration and input trajectories.

3.3.3 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 [?], 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 – 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 [?].

This choice for parameterizing the flat output introduces a new parameter to be set in the motion planning algorithm which is 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.

3.3.4 Optimization solver

There is a variety of numerical optimization packages implemented in many different programming languages available for solving optimization problems [10]. Each of them may have their own way of defining the optimization problem and may or may not support specific kinds of constraints (equations, inequations or boundaries).

For the initial implementation written in python two packages stood out as good, easy-to-use options for solving the constrained optimization problem that models the planning motion task.

Scipy is a vast open-source scientific package based on python that happens to have a minimization module. Within this module many minimization methods can be found. For this specific optimization problem, only the method SLSPQ was appropriate. It was the only one to handle constrained minimization where the constraints could be equations as well as inequations.

pyOpt is a much smaller ecosystem than Scipy that is specialized in optimization. It gathers many different numerical optimization algorithms some of them free and some licensed. Again, among all of them there were only a few suitable for this problem which were also free: SLSQP (same as the one implemented within Sicpy), PSQP and ALGENCAN.

SLSQP and PSQP are both SQP (for sequential quadratic programming) methods. A SQP method attempts to solve a nonlinearly constrained optimization problem where the object function and the constraints are twice continuously differentiable. It does so by modeling the object function ($\min f(x)$) at the current iterate x_k by a quadratic programming subproblem and using the minimizer of this subproblem to define a new iterate x_{k+1} [9].

The ALGENCAN method

describe algecan

Among the optimization solvers with C++ interface the following were the ones that had at least one method suitable for ...

— OPT++ is another library that uses whether OptNIPS, a free nonlinear interior-point algorithm or NPSOL, a licensed sequential quadratic programming algorithm. Both require Hessians implementation.

— IPOPT (Interior Point OPTimizer) is a software package for large-scale nonlinear optimization. IPOPT implements an interior-point algorithm for continuous, nonlinear, nonconvex, constrained optimization problems. It is meant to be a general purpose nonlinear programming (NLP) solver. However, it is mainly written for large-scale problems with up to million of variables and constraints. IPOPT presents a reasonably easy to use C++ interface but, like the previous library, it requires the implementation of gradients, Jacobians and Hessians for the objective function and constraints. However, an excellent example code is available on their website that shows how to use the ADOL-C (Automatic Differentiation by OverLoading in C++) package in order to facilitate the evaluation of those first and higher derivatives.

— NLOPT is a free/open-source library for nonlinear optimization, providing a common interface for a number of different free optimization routines available online as well as original implementations of various other algorithms. Within the NLOPT library three methods were applicable to our NLP. ISRES (a global optimizer) that combined with the augmented Lagrangian method could handle nonlinear constraints. COBYLA is a local, derivative-free optimizer and as such does not need computation of gradients, Jacobians nor Hessians. At last the SLSQP method is also available, but no computation of the needed derivatives is done.

— RobOptim: a C++ Library for Numerical Optimization applied to Robotics. Although this library looks very

Not all implementations of numerical optimizers are suitable for solving the particular kind of optimization problems presented before. The need of a solver that supports nonlinear equality and inequality constrains restricts the number of possible choices.

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 constrains, its availability in the minimization module of the open-source scientific package Scipy [?] helps to facilitate the motion planner implementation.

However, an error was experienced using this optimizer which uses the SLSQP Optimization subroutine originally implemented by Dieter Kraft [?]. 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 experienced by other users as discussed in [?].

For working 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 current position).

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

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

Where $p_{b,goal}$ and $p_b(\tau_{s-1} + T_c)$ are the positions associated with configurations $q_{b,goal}$ 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 $T_p v_{b,max}$ the maximum possible distance covered by R_b during a planning horizon and $s \mid s \in [0, k), s \in \mathbb{N}$ the current receding horizon problem index.

Numerically solving the constrained optimization problems presented before introduces a new parameter in the algorithm: the time sampling for optimization N_s.

Chapter 4

Results

4.1 Generated solution examples

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

The trajectory and velocities shown in Figures 4.1 and 4.2 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.

It is possible to see on those figures how the planner generates configuration and input trajectories satisfying the constraints associated with the goal states.

In particular, in Figure 4.1, the resulting plan is computed ignoring coupling constraints (3.1 is never performed) and consequently two points of collision occur. A collision-free solution is presented in Figure 4.2. Specially near the regions were collisions occurred a change in the trajectory is present from Figure 4.1 to Figure 4.2 to avoid collision. Complementary, changes in the robots (linear) velocities across charts in both figures can be noticed. Finally, the bottom charts show that the collisions were indeed avoided: inter-robot distances in Figure 4.2 are greater than 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. **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 detection radius of the robot (d_{sen}).

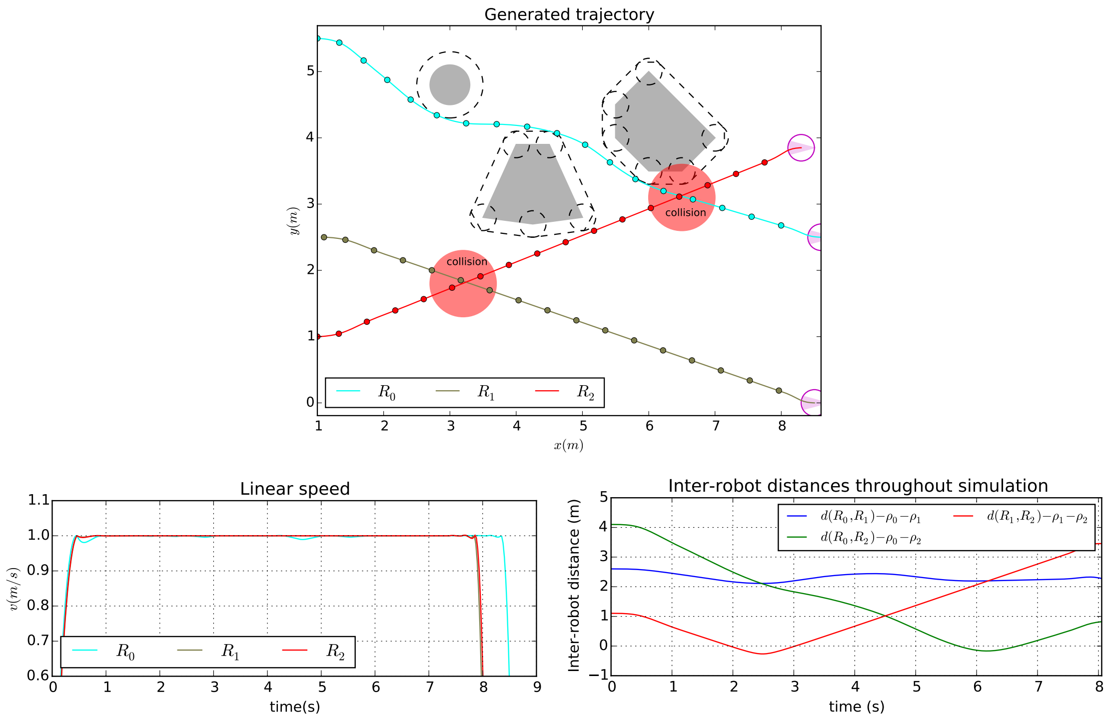


Figure 4.1 – Motion planning solution without collision handling

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 a maximum number of iterations and a stop condition parameters.

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

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

4.2 Analysis of parameters' impact on performance

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

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.

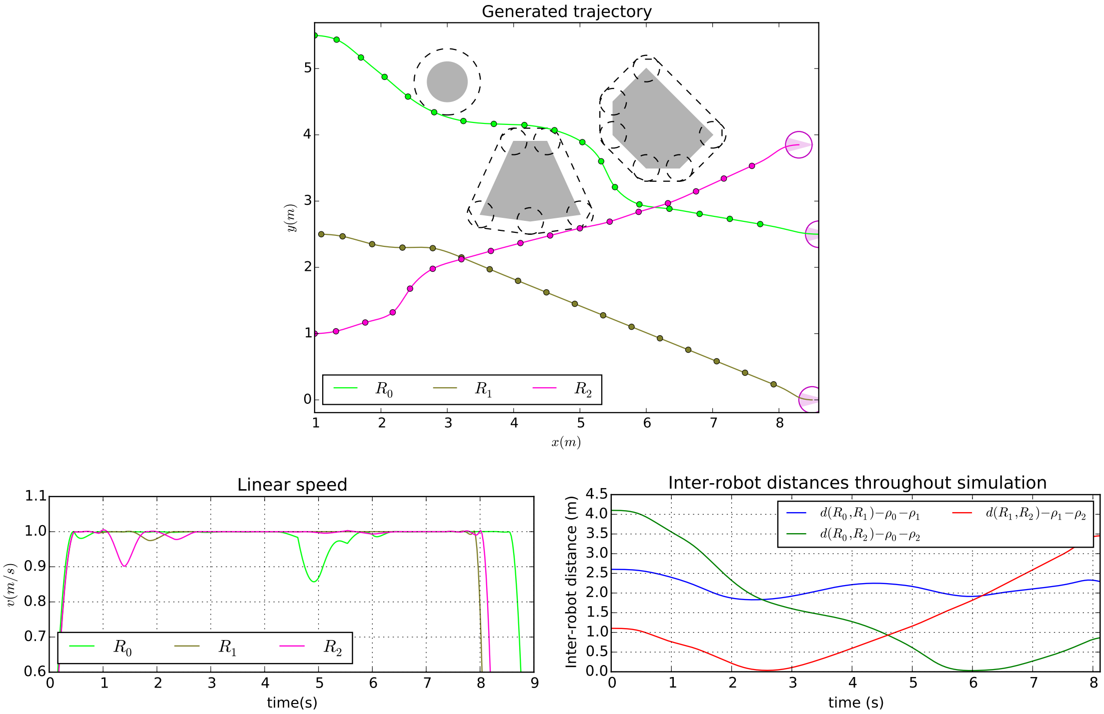


Figure 4.2 – Motion planning solution with collision handling

Table 4.1 summarizes one of the scenarios studied for a single robot. Results obtained from simulations in that scenario are presented in Figure 4.3, for different parameters set.

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 where the algorithm is run. Those simulations were run in an Intel Xeon CPU 2.53GHz processor.

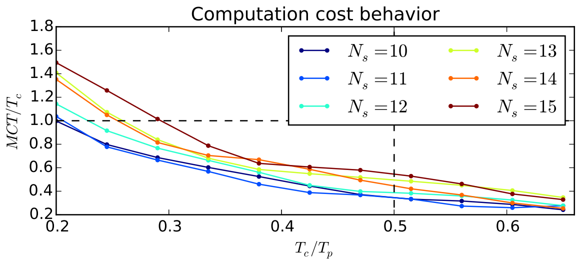
Rather than observing the absolute values, it is interesting to analyze the impact of changes in the parameters values. In particular, an increasing number of N_s increases MCT/T_c for a given T_c/T_p. Similarly, an increasing of MCT/T_c as the number of internal knots N_{knots} increases from charts 4.3a to 4.3c is noticed.

Further analyses of those data show that finding the solution using the SLSPQ method requires $O(N_{knots}^3)$ and $O(N_s)$ time. Although augmenting N_{knots} can yield to an impractical computation time, typical N_{knots} values did not need to exceed 10 in our simulations, which is a sufficiently small value.

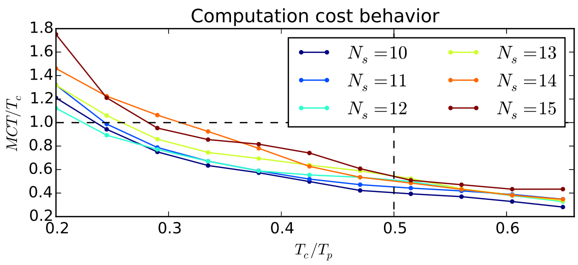
Another parameter having direct impact on the MCT/T_c ratio is the detection radius of the robot's sensors. As the detection radius of the robot increases, more obstacles are seen at once which, in turn, increases the number of constraints in the optimization problems. The impact of increasing the detection radius d_{sen} in the MCT/T_c ratio can be seen in the Figure 4.4 for a scenario with seven obstacles. The computation time stops increasing as

Table 4.1 – Values for scenario definition

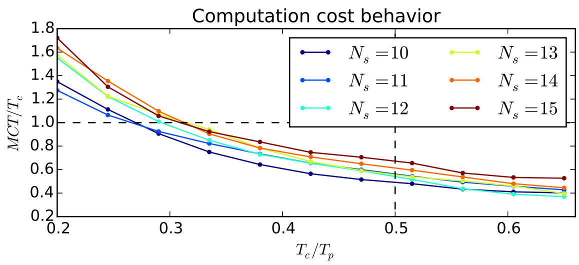
v_{max}	1.00 m/s
ω_{max}	5.00 rad/s
q_{inital}	$[-0.05\ 0.00\ \pi/2]^T$
q_{final}	$[0.10\ 7.00\ \pi/2]^T$
$u_{initial}$	$[0.00\ 0.00]^T$
u_{goal}	$[0.00\ 0.00]^T$
O_0	$[0.55\ 1.91\ 0.31]$
O_1	$[-0.08\ 3.65\ 0.32]$
O_2	$[0.38\ 4.65\ 0.16]$



(a) Four internal knots



(b) Five internal knots



(c) Six internal knots

Figure 4.3 – Three obstacles scenario simulations

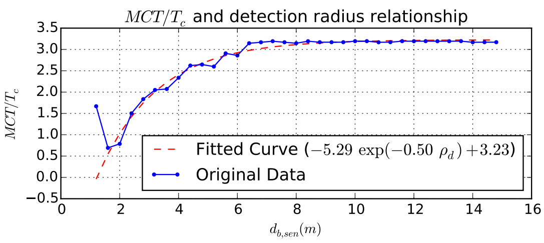


Figure 4.4 – Increasing of detection radius and impact on a MTC/T_c ratio

soon as the robot sees all obstacles present in the environment.

Obstacle penetration P

Obstacle penetration area P gives a metric for obstacle avoidance and consequently for solution quality. A solution where the planned trajectory does not pass through an object at any instant of time gives $P = 0$. The greater the P the worse is the solution. 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 obstacles radius computed by the robot's perception system by the maximum distance that the robot can run within the time spam 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 is 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 4.5 shows the penetration area decreasing as the number of samples increases.

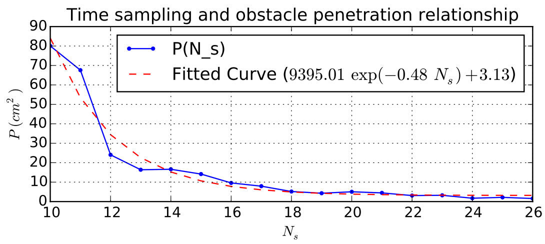


Figure 4.5 – Obstacle penetration decreasing as sampling increases

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 T_c, a more optimal 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 take into account the fact that the fewer the samples the greater will be the obstacle penetration area as shown previously in Figure 4.5.

Furthermore, the Figure 4.6 shows travel time invariance for changes in the detection radius far from degraded values that are too small. This points out that a local knowledge of the environment provides enough information for finding good solutions.

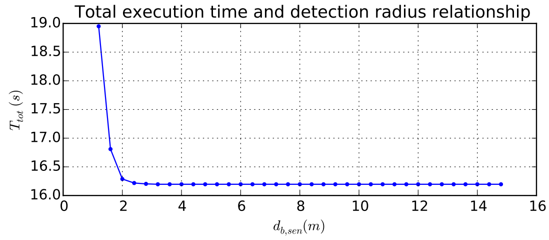


Figure 4.6 – Increasing of detection radius and impact on T_{tot}

4.3 Tests in a physics simulation environment

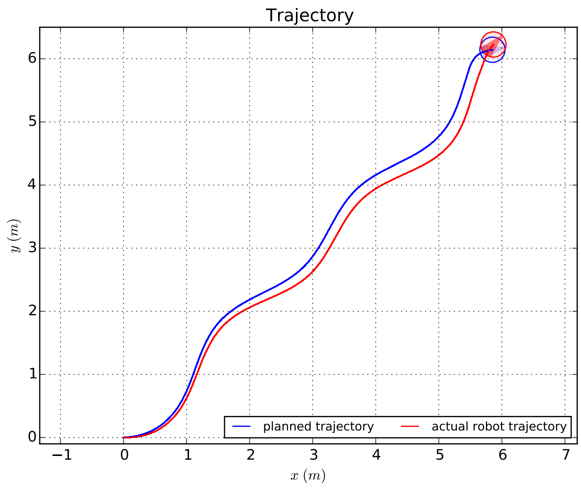


Figure 4.7 – Increasing of detection radius and impact on T_{tot}

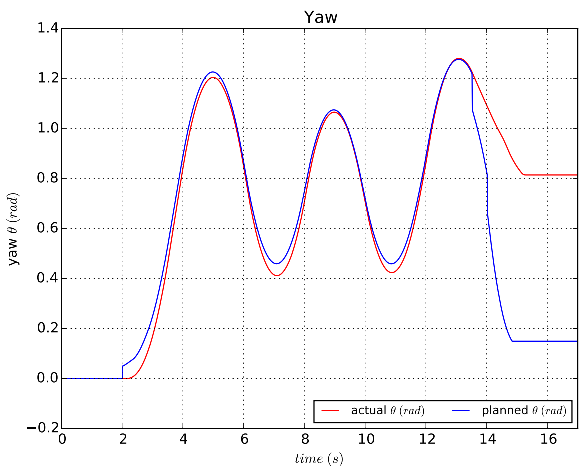


Figure 4.8 – Increasing of detection radius and impact on T_{tot}

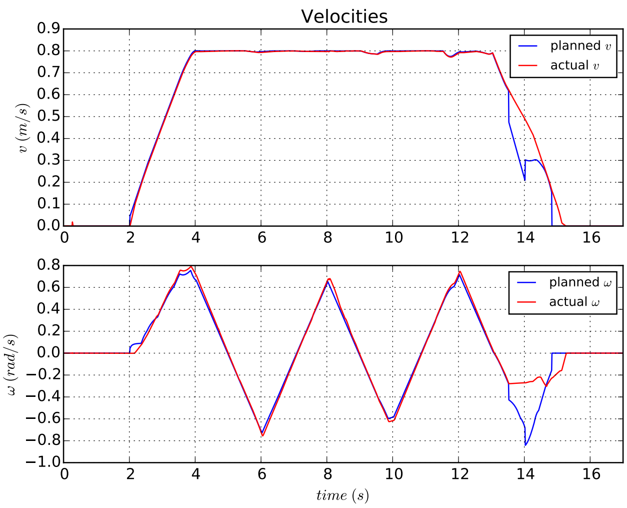


Figure 4.9 – Increasing of detection radius and impact on T_{tot}

Chapter 5

Conclusion

A motion planner for cooperative multi-robot systems has been developed. A base algorithm presented in [?] was implemented, extended and analyzed aiming for the development of an optimal, distributed, collision-free and local motion planner for multi-robot systems composed by nonhonolomic mobile robots in the presence of obstacles.

There is several points of this method that can be improved.

Although the use of the free optimization solver SLSQP provided in the SciPy library was enough to produce good results, it will be interesting to test licensed solvers as well. In particular, the solver CFSQP, which is also a sequential quadratic programming solver, should be tested. This solver produces feasible solutions at each iteration while searching for the locally optimal solution of the NLP. This aspect would be convenient for when the T_c interval expires before an optimal solution is found as the intermediary solution would still be useful for the robot's motion.

Better initialization of the "first guesses" that must be provided to the NLP solvers could be done by performing a fast non-optimal collision-free path planning and feeding its solution to the solver.

Bibliography

[1] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[2] M. Defoort, J. Palos, a. Kokosy, T. Floquet, W. Perruquetti, and D. Boulinguez. Experimental Motion Planning and Control for an Autonomous Nonholonomic Mobile Robot. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, (April):10–14, 2007.

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

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

[5] Dieter Fox, Wolfram Burgard, Sebastian Thrun, et al. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[6] a. Kelly and B. Nagy. Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[7] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[8] Mark B Milam. *Real-time optimal trajectory generation for constrained dynamical systems*. PhD thesis, California Institute of Technology, 2003.

[9] Jorge Nocedal and Steve J. Wright. *Numerical optimization : with 85 illustrations*. Springer series in operations research. Springer, New York, Berlin, Heidelberg, 1999. Tirage corrigé: 2000.

[10] Ruben E. Perez, Peter W. Jansen, and Joaquim R. R. A. Martins. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Structures and Multidisciplinary Optimization*, 45(1):101–118, 2012.

[11] Jacob T. Schwartz and Micha Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37(1):157–169, 1988.

[12] Workshop. Workshop on on-line decision-making in multi-robot coordination. http://robotics.fel.cvut.cz/demur15/, 2015. Accessed: 2015-06-22.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.