Acta Polytechnica **00**(0):1–4, 0000

PREPRINT 2015-07-08

MULTI ROBOT OPTIMAL TRAJECTORY GENERATION

José Magno Mendes Filho[a, b], Eric Lucet[a, *]

[a] *CEA, LIST, Interactive Robotics Laboratory, Gif-sur-Yvette, F-91191, France*

[b] *ENSTA Paristech, Unité informatique et Ingénierie des Systéme, 828 boulevard des Marechaux, 91762, France*

[*] corresponding author: `eric.lucet@cea.fr`

Abstract. Abstract.

Keywords: multi robot path planning, mobile robot.

1. Introduction

The command and control of mobile robots is a long-standing subject of research in the iterative robotics domain. As many modern companies started to adopt mobile robot systems such as fleets of autonomous forklift trucks [1] to improve their commercial performance, this subject becomes increasingly important and complex. Especially when those robots are executing theirs tasks in human-occupied environment.

One basic requirement for such robotic systems is the capacity of generating a trajectory that connects two arbitrary configurations. For that, different constraints must be taken into account to ideally find an appropriated trajectory, in particular:

- kinematic and dynamic constraints;
- geometric constraints;
- constraints associated with uncertainties about the current state of world and the outcome of actions.

The first constraints derive directly from the mobile robot architecture implying in nonholonomic constraints for many types of mobile robots. Geometric constraints result from the impossibility of the robots to assume some specific configurations due to the presence of obstacles or environment bounds. In turn, uncertainty constraints come from the impossibility of the robots to have an absolute and complete perception of the world (including theirs own states) as well as the outcome of non stochastic events.

We are particularly interest in solving the problem of dynamic planning a trajectory for a fleet of nonholonomic mobile robots in an partially known environment occupied by static obstacles being near optimal with respect to the execution time (time spend from going from initial to final configurations).

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

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

Cell decomposition methods as presented in [**?**] have the downside of requiring a structured configuration space and an *a priori* model of its connectivity. Besides, the cell decomposition constrains the space of admissible solutions. TODO verify/understand.

Initially proposed in [**?**] the vector field obstacle avoidance method was improve 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 extend to mobile manipulators in [**?** **?**] uses approximations of the trajectory shape (combinations of arcs and straight lines) limiting the space of solutions and make this method inappropriate for really constrained environments.

The dynamic window approach [**?**] can handle trajectory planning for robots at elevated speeds and in presence of obstacles but is not flexible enough to be extended to a multi robot system.

In this paper, we focus on the development of a trajectory planning algorithm. This algorithm finds collision-free trajectories for a multi robot system in presence of static obstacles which are perceived by the robots as they evolve in the environment. The dynamic trajectories found are near optimal with respect to the total time spend 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 not being able to handle dynamic obstacles as it is.

This algorithm is based mainly on the work done in [2] but we made changes in particular to respect a precise final configuration for the multi robot system and about how to set parameters for solving the nonlinear programming problem (NLP).

This paper is structured as follows: The second section presents a trajectory planning algorithm that solves the problem of one robot going from an initial configuration to a final one in the presence of static obstacles. The third section extends the method presented in the second section so a collision-free trajectory that maintains the communication link between

the robots in the fleet can be computed. The forth section is dedicated to the results found by this method and the analysis of the computation time and solution quality and how they are impacted by the algorithm parameters. The fifth section presents the comparison of our approach to another one presented in []. Finally, in section six we present our conclusions and perspectives.

2. Sliding window approach

Let us consider the problem of planning a trajectory for a unique mobile robot to go from an initial configuration $q_{initial}$ to a final one q_{final} while respecting kinematic, dynamic and geometric constraints.

2.1. Flatness property

As explained in [2] 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 kinetic 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 behavior of the system can be completely determined by the flat output.

TODO put image of flat bijective application.

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 nonlinear problem. After finding (optimal) trajectories in the flat space it is possible to retrieve back the original state and input trajectories as shown in Figure **??**

2.2. Parametrization of the flat output

Another important aspect of our approach is the parametrization of the flat output trajectory. As done 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, i.e., changes in parameters values have a local impact on the resulting curve.

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

2.3. Planning for sliding time horizon by solving NLPs

At this point a nonlinear programming problem could be written as in the equations **??** to **??** for finding the B-spline parameters that can minimize the mission time respecting some constraints.

The equations **??** and **??** express the we see a first group of constraints that are expressed as equations a NLP

However, two aspects of the implementation of this algorithm are neglected in [2]: the initial values for the control points and the procedure for reaching a desired final state.

2.4. NLP initialization and objective function

The initialization of the solution paths used for each NLP solving is important for two reasons:

A good initialization allows the optimization solver to find a better solution for a given timespan.

When using a local optimization method the initialization can drag the final solution to one or other local minima.

The simplest of initializations was performed in ours studies. Linear spacing from current flat output value to the estimate final flat output. The estimate final output is simply the flat output computed from the estimate final states and inputs. The estimate final states and inputs are computed assuming a displacement from the current position of the maximum linear speed of the robot times the planning horizon, and assuming that the direction of the movement is equal to (final position - current pos) vector.

TODO talk about the too close obstacles problem and the dumb solution.

2.5. Stop condition and last NLP

As the robot evolves its state approximates to the final state. be minimized. At some point the constraints associated to the final state shall be integrated into the NLP and the timespan for performing this last step shall not be fixed and must be one of the values calculated. final state.

The criterion used to pass from the NLP used during for the initial and intermediates steps to the last step NLP is define below in the equation **??**:

$$d_{rem} \geq d_{min} + T_c \cdot v_{max} \tag{1}$$

This way we insure that the last planning section will be done for at least a d_{min} distance from the robot's final position. This minimal distance is assumed to be sufficient for the robot to reach the final state.

After stopping the sliding window algorithm we calculate new parameters for the solution representation and computation taking into account the estimate remaining distance.

The following pseudo code summarizes the algorithm:

Algorithm 1 Sliding window planning algorithm

procedure PLAN
  $knots \leftarrow$GENKNOTS$(t_p, d_{spl}, n_{knots})$
  $time \leftarrow$LINESPACING$(0, t_p, n_s)$
  $q_{latest} \leftarrow q_{initial}$
  $d_{rem} \leftarrow |$POS$(q_{final}) -$POS$(q_{latest})|$
  while $d_{rem} \geq d_{min} + T_c \cdot v_{max}$ **do**
    $q_{latest} \leftarrow$PLANSEC
    $d_{rem} \leftarrow |$POS$(q_{final}) -$POS$(q_{latest})|$
  end while
  $s \leftarrow$MIN$(\frac{d_{rem}}{v_{max} \cdot t_p}, 1.0)$
  $n_{knots} \leftarrow$MAX(ROUND$(s \cdot n_{knots}), d_{spl})$
  $n_s \leftarrow$MAX(ROUND$(s \cdot n_s), n_{knots} + d_{spl})$
  $\Delta t \leftarrow$PLANLASTSEC
end procedure

3. DECENTRALIZED MULTI ROBOT SLIDING WINDOW PLANNING ALGORITHM

A straight forward extension of the previous algorithm can be done in order to support a multi robot system. The sliding window algorithm presented before remains virtually the same. The changes are done within the PLANSEC PLANLASTSEC routine.

After solving the NLP stated before each robot will have generated an intended trajectory that would be valid if we were dealing with a mono robot system. For the multi robot system some exchange of information among the robots and possibly some replanning has to be done.

Right after solving the standalone NLP a given robot represented by the index i computes a conflict list that is based on all robots' positions as of when they started planning their intended trajectories (solving the latest standalone NLP). This conflict list contains the indexes of the robots in the fleet that can possibly cause some conflict. The word conflict here is understood as a collision or a loss of communication between robots in the fleet.

Notice that the i robot can compute its conflict list as soon as it finishes its planning even though other robots may still be doing so.

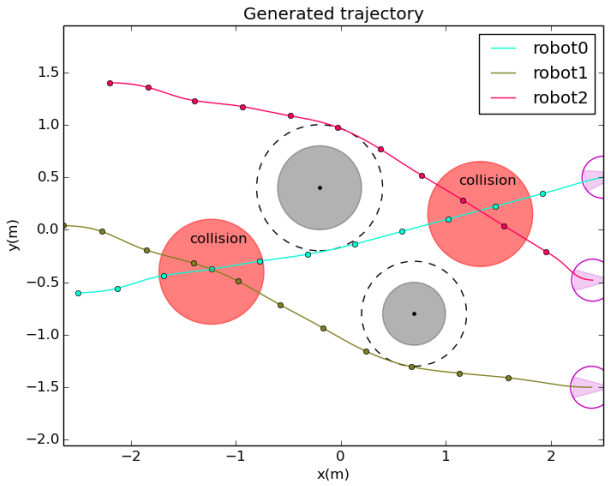
For the next step of replanning all robots involved in a conflict have to be done computing the first standalone planning. This is needed simply because all intended trajectories will be taken into account on the replanning part.

Using the intended trajectory as the initialization of the optimization parameters a new NLP is solved where collision avoidance between robots and keeping communication are translated into constraints.

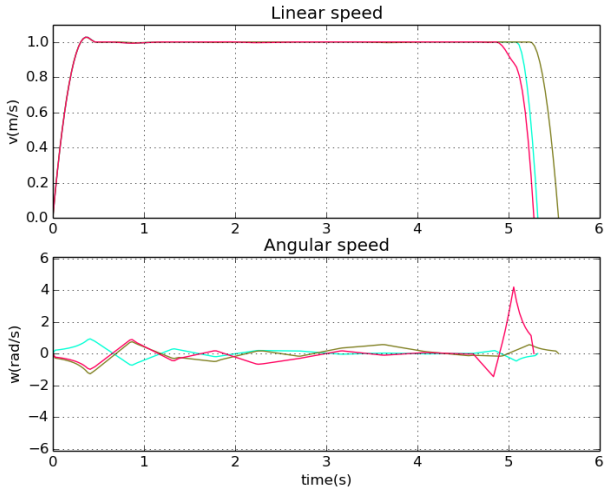
After solving this second NLP, the trajectories are updated and the planning goes on to the next section.

In Figures 1 and 2 the results of the decentralized multi robot algorithm can be seen. In Figure 1 no conflict handling is done and two collisions zones can be identified. For trajectory showed in the Figure 2 the robots optimize their trajectories using the multi robot adaptation of the algorithm. No conflict occurs and we can observe a change in the robots velocities and total execution time.

the multi robot system TODO



<– use this for your graphics



<– use this for your graphics

FIGURE 1. Our results: black box (top) and black box (bottom).

3.1. CONFLICT DETECTION

Conflict detection is computed TODO

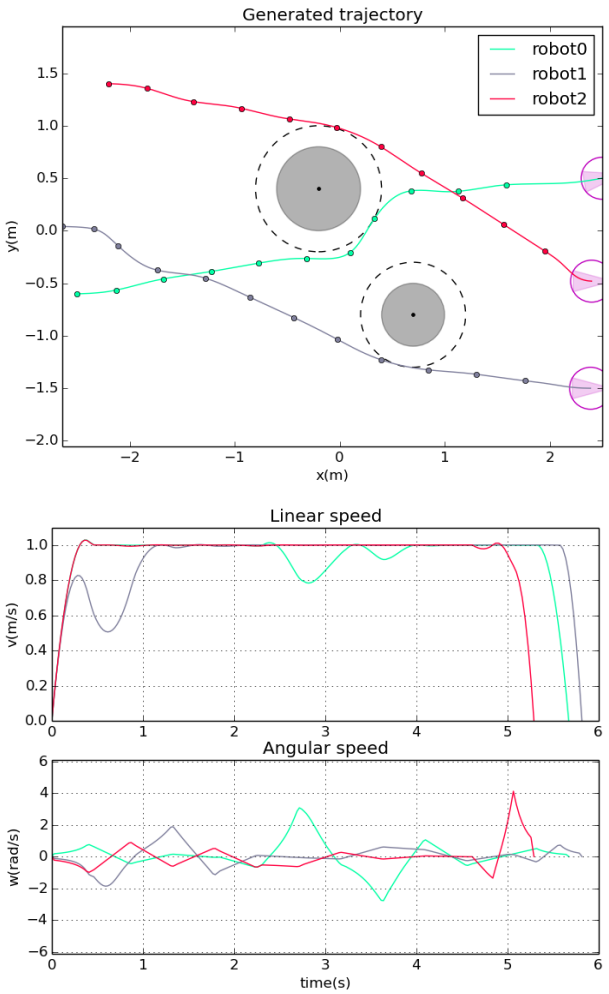
3.2. ADITIONAL CONSTRAINTS

The additional constraints associated to the multi robot system TODO

4. PARAMETERS' IMPACT ANALYSES

Four 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 four criteria were:

- *Maximum computation time* over the computation horizon (MCT/T_c ratio);



<– use this for your graphics

Figure 2. Our results: black box (top) and black box (bottom).

- Obstacle penetration area (P).
- The total execution time (T_{tot});
- Additional time for conflict handling???.

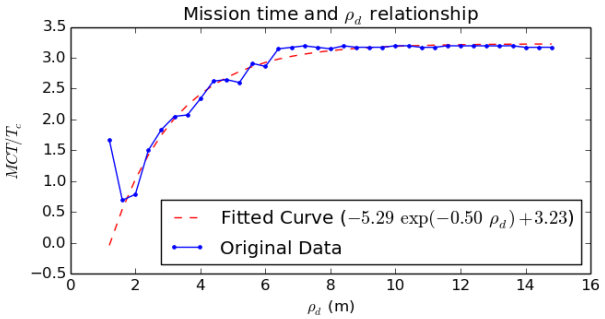
4.1. Maximum computation time over computation horizon MCT/T_c

The significance of this criterion lays in the need of quarantining 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 planning section (robot-to-robot conflict taken into account).
Based on several simulations with different scenarios we were able to TODO

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

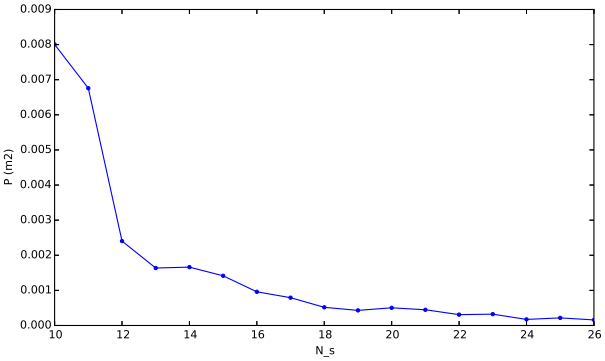
4.2. Obstacle penetration P

4 TODO rescale images



for your graphics

Figure 3. Increasing of detection radius and impact on a MTC/T_c ratio



use this for your graphics

Figure 4. Obstacle penetration decreasing as sampling increases

4.3. Total execution time T_{tot}

4.4. Additional time for conflic handling$P$

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

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: 2015-07-07.

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