UNIVERSITÉ GRENOBLE ALPES

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : Automatique-Productique
Arrêté ministériel : 7 août 2006

Présentée par

Saed AL HOMSI

Thèse dirigée par **Bernard BROGLIATO**
et codirigée par **Pierre-Brice WIEBER**

préparée au sein de l'**INRIA**
dans **l'école doctorale EEATS**

Online Generation of Time-Optimal Trajectories for Industrial Robots in Dynamic Environments

Thèse soutenue publiquement le **17 Mars 2016**,
devant le jury composé de :

M. Nicolas Mansard
Chargé de Recherches au LAAS-CNRS, Rapporteur
M. Yannick Aoustin
Professeur à l'Université de Nantes, Rapporteur
M. Philippe Poignet
Professeur à l'Université de Montpellier 2, Examinateur
M. Nicolas Andreff
Professeur à l'Université de Franche-Comté, Examinateur
M. Bernard Brogliato
Directeur de Recherches à l'INRIA Grenoble Rhône-Alpes, Directeur de thèse
M. Pierre-Brice Wieber
Chargé de Recherches à l'INRIA Grenoble Rhône-Alpes, Co-directeur de thèse

Abstract

In the field of industrial robots, there is a growing need for having cooperative robots that interact with each other and share work spaces. Currently, industrial robotic systems still rely on hard coded motions with limited ability to react autonomously to dynamic changes in the environment. This thesis focuses on providing a novel framework to deal with real-time collision avoidance for robots performing tasks in a dynamic environment. We develop a reactive trajectory generation algorithm that reacts in real time, removes the fastidious optimization process which is traditionally executed by hand by handling it automatically, and provides a practical way of generating locally time optimal solutions.

The novelty in this thesis is in the way we integrate the proposed time optimality problem in a task priority framework to solve a nonlinear optimization problem efficiently in real time using an embedded system with limited resources. Our approach is applied in a Model Predictive Control (MPC) setting, which not only improves reactivity of the system but presents a possibility to obtain accurate local linear approximations of the collision avoidance constraint. The control strategies presented in this thesis have been validated through various simulations and real-world robot experiments. The results demonstrate the effectiveness of the new control structure and its reactivity and robustness when working in dynamic environments.

Résumé

Nous observons ces dernières années un besoin grandissant dans l'industrie pour des robots capables d'interagir et de coopérer dans des environnements confinés. Cependant, aujourd'hui encore, la définition de trajectoires sûres pour les robots industriels doit être faite manuellement par l'utilisateur et le logiciel ne dispose que de peu d'autonomie pour réagir aux modifications de l'environnement. Cette thèse vise à produire une structure logicielle innovante pour gérer l'évitement d'obstacles en temps réel pour des robots manipulateurs évoluant dans des environnements dynamiques. Nous avons développé pour cela un algorithme temps réel de génération de trajectoires qui supprime de façon automatique l'étape fastidieuse de définition d'une trajectoire sûre pour le robot.

La valeur ajoutée de cette thèse réside dans le fait que nous intégrons le problème de contrôle optimal dans le concept de hiérarchie de tâches pour résoudre un problème d'optimisation non-linéaire efficacement et en temps réel sur un système embarqué aux ressources limitées. Notre approche utilise une commande prédictive (MPC) qui non seulement améliore la réactivité de notre système mais présente aussi l'avantage de pouvoir produire une bonne approximation linéaire des contraintes d'évitement de collision. La stratégie de contrôle présentée dans cette thèse a été validée à l'aide de plusieurs expérimentations en simulations et sur systèmes réels. Les résultats démontrent l'efficacité, la réactivité et la robustesse de cette nouvelle structure de contrôle lorsqu'elle est utilisée dans des environnements dynamiques.

Acknowledgments

I would like to express my sincere gratitude to my supervisors for their encouragement, helpful guidance, and valuable advices. I thank Bernard Brogliato for showing interest in my work (also for signing a bunch of documents every now and then). I would like to deeply thank Pierre-Brice Wieber with whom I worked most of the time, thanks for his continuous support, patience, advice and availability.

I would like to thank the Embedded Software team in Omron Adept Technologies: I am thankful to Matthieu Gilbert, the one who stand behind the idea of this research. I consider myself lucky to have worked with him for almost two years he motivates me a lot to work hard and to touch the value of what I am doing. Thanks a lot to Sebastien Delmas, for his kindness, flexibility, and helpful feedback. Thanks to Ben Dart for his help with writing. Thanks for Ruben Zhao, for his great help on the implementation.

I would like to thank the members of jury: Yannick Aoustin, Nicolas Mansard, Philippe Poignet and Nicolas Andreff for accepting to be in my jury and for the great feedback I got from them. I would like to thank the members of BIPOP team at INRIA, especially Dimitar Dimitrov and Alexander sherikov who aided me to realize the ideas of my thesis. I am thankful to the employees of Omron Adept Technologies France for all their kindness and helpful: Bruno, Jean-luc, Evelyne, Sonia, Christelle, Pascal, and the others. I would like to thank my friends: Amr, Samer, Waleed, Ammar, Abdul-Rahman, Baker, Jamal, Feras and Tarek for their friendship and support.

Last but not the least; I would like to thank my family: My mother, for the everyday supporting calls, for her nice words and comments that helped me to pass all the hard moments and for all her prays for me. My father, without his support and help I can't reach such a moment. my brothers Mohammad and Yazan, and my dear sister Zeina for their support. My lovely wife Roua for being an endless source of energy, enthusiasm, and encouragement.

Thanks for you all

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Industrial Motivation

The first industrial robot was created by George Devol in the USA in 1954 and joined the assembly lines at General Motors in 1961 [40]. Since then robotic technology has started handling numerous applications in the industry: manufacturing of integrated circuits, cars, food, and other automated lines that assemble, handle, move and package products. Nevertheless, recent publications [14, 21] show that industrial robots are under-utilized in the Small and Medium Enterprises (SMEs) due to several factors. The major issues are the lack of flexibility of industrial robots and the difficulty to program a complete automatic application.

SMEs usually produce various products in small quantities. Figure 1.1 from [21] illustrates that in mass production, robotic systems are in use for long production phases, while in SMEs robotic systems are reconfigured and changed frequently. For this reason, production lines in SMEs need to be reconfigured in easy and inexpensive ways to fit different tasks, what is not possible with the current industrial robot systems. Installing and programming a robotic system is a complex task, it can require an expert robotic engineer some weeks to configure a given required task. This complexity rises the cost of production. The study in [34] shows that for some industrial applications the cost of programming

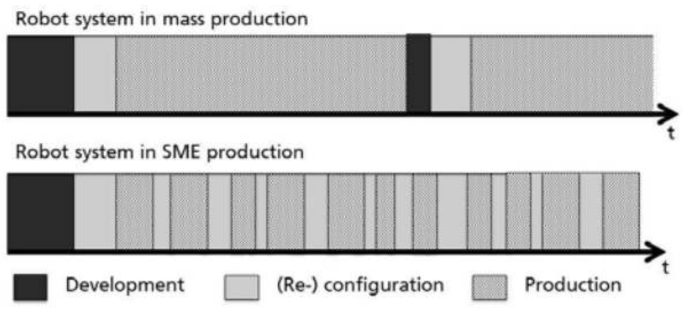

Robot system in mass production
t
Robot system in SME production
t
Development
(Re-) configuration
Production

Figure 1.1: Comparison of phases of robot use in mass production and small and medium enterprises (SMEs) production. Reprinted from [21].

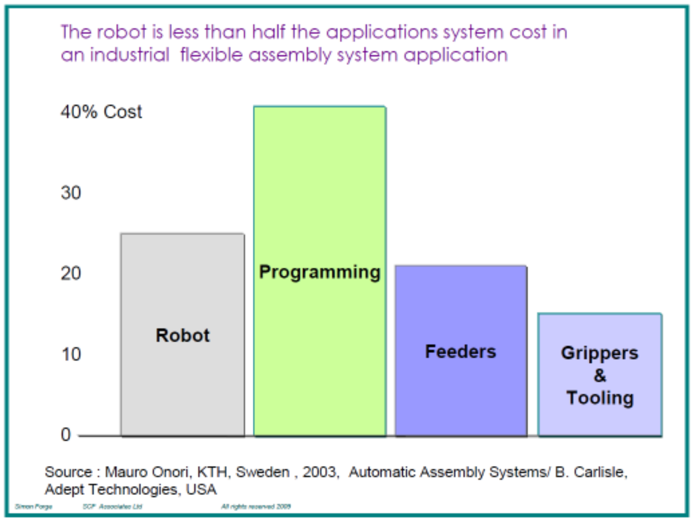


Figure 1.2: The industrial robot is only one cost item for the total production system. Reprinted from [34].

a robot can reach around 40% of the actual application cost (see figure 1.2). As a result, industrial robots cannot always be used cost effectively for production tasks in SMEs. There is a real need for a new generation of robot controllers that simplifies and facilitates the installation and programming of industrial robots.

Industrial robots will typically follow a designed path described by a series of trajectory segments. These segments can be straight lines or interpolations. This principle of programming a succession of segments has been mainly adopted because it offers some flexibility for the end user. This means that for a complex robotic application the programmer must teach a large amount of points: this can reach more than 1000 in certain applications. The programming methods used by industry can be categorized into two categories: online programming and offline programming [65]. In the former, the programmer teaches the robot in the field by moving manually the robot end effector to the desired points, the concept is simple but it is only suitable for programming simple applications and it can require a lot of experience to get a well formed program. On the other hand, offline programming is based on a 3D model of the complete work cell; it shifts most of the work from the field to the office. It is efficient for complex applications but again it needs a lot of experience and takes considerable time and effort. These complexities make robot integration a difficult task, and deep knowledge in robotic engineering is required, but usually not present in SMEs. Thus simplifying the programming methods of industrial robots will open the door to huge markets and expand the use of robots in SMEs and other areas. An enhancement to enable simple robot programing for SMEs is smart trajectory generation which can create a safe robot trajectory only knowing the physical constraints and work environment.

As outlined above, the programming of industrial robots is generally based on the application engineer experiences. In addition to planning a collision free path and teaching the robot the desired points, engineers should optimize robot trajectory so that the physical limitation of the actuators are respected and the robot works in an efficient way, satisfying the required criteria: go as fast as possible, consume minimum energy and so on. However, even experienced engineers can only reach a limited level of efficiency because robotic systems are complex. Hence, a better way is to shift this task to a smarter trajectory generator, using computer aided optimization methods. In this thesis, we will focus on generating trajectories with a minimum time objective (go as fast as possible) subject to actuator limitations.

Robot manipulators may share their workspace with other robots or humans. They may cooperate with each other to achieve a common goal or they may share the same production line to increase the throughput. This can save space which for many companies is of great benefit through reduction of real estates costs. Figure 1.3 shows an industrial application where two robots share the same workspace and each one has its own task. There is a new

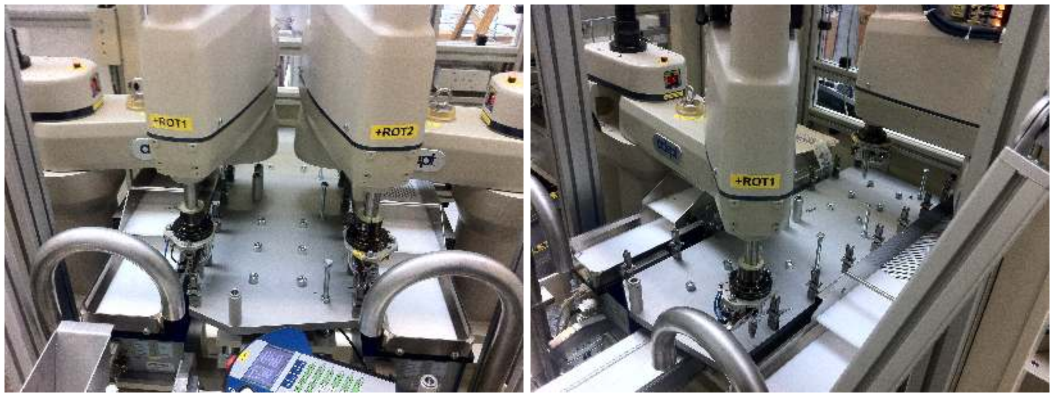


Figure 1.3: An industrial application: two robotic manipulators share the same work space.

emergent market for collaborative robots, either collaborating between robots or between humans and robots [14]. This is an exciting and challenging development for industrial robots, which could open the door to new industrial and even nonindustrial applications. This is still a futuristic subject because of the lack of safety standards. But one essential step towards having collaborative robots is to have a trajectory generator able to avoid dynamic obstacles and react in real time to sudden changes.

1.2 Scientific Motivation

Trajectory planning and optimization have been central problems in robotics for more than 30 years, and is still an active subject today. Unlike the early eighties, industrial robots now evolve in dynamic environments: conveyor tracking, working on moving parts, bin picking, robots sharing the same workspace and collaborative robots. They need to have the highest possible productivity. As a result, trajectory generators must work at a high rate, reaching 1 kHz, running on an embedded system with limited power (e.g. 400 MHz, 16kB L1 cache). The scientific literature shows some very interesting algorithms and model simplifications but they are still not relevant for industrial robots.

A robot is a nonlinear system, so the transcription of the trajectory planning to an optimization problem is clearly nonlinear with nonlinear equality and inequality constraints. Thus one optimization iteration is computationally expensive because of the non linearity, and the management of the non-linear inequality constraints which is a complex process. All this is to show how difficult it is to design a trajectory generator running in real time. Usually the literature proposes two solutions to solve trajectory optimization problems (optimal control problem):

- Indirect methods, based on solving the equations coming from Pontryaguin's principle, but finding the optimal initial conditions for the adjoint variable is complex and can not be done online. Moreover, the solution is sensitive to the accuracy of the initial conditions [88].

- Direct methods, which consist in discretizing the optimal control problem, what leads to a standard optimization problem of high dimension solved using a SQP (Sequential Quadratic Programming). This method is much more stable than the previous one but we can still see some problems in handling the inequality constraints efficiently [9].

It has been proposed in [8] and [30] to add continuation methods to the direct multiple shooting methods; this allows a highly efficient way of handling inequality constraints and online nonlinear optimization in the case of feedback controlled system. Such methods have been widely used in the chemistry industry and are more and more popular in the robotics area [23]. All these approaches consider that the problem of trajectory planning must be translated into a constrained non-linear optimization problem.

Additionally, we can consider that a robot must execute tasks with different priorities, for example one high priority task can be to avoid obstacles, then another task with lower priority could be to minimize cycle time, so at the end the robot will avoid obstacles even if it will make a longer motion. It is possible to imagine a robot completely programmed with tasks and priorities. This concept of tasks linked with optimization has been formalized in [61]. Some numerical and theoretical aspects for this concept have been presented in [22, 26, 25]. This latest approach seems promising and takes a new approach to how robots must be programmed.

To summarize, this thesis will have to focus on the following scientific questions:

- What are the best theoretical and numerical approaches to optimize in real time industrial robots trajectories while integrating their own physical limitations and constraints of their environment?

Figure 1.4: Cobra s600 SCARA Adept robot. Reprinted from [1]

- What is the impact of such approach in the way industrial robots are programmed?

1.3 Thesis Contribution

The main contribution of this dissertation lies in introducing a novel framework for industrial robot trajectory generator. It will propose a smart trajectory generation technique to replace the traditional one. The prominent features of this trajectory generator are:

1. Reactivity in real time: It can instantly perform collision avoidance in the presence of multiple static and dynamic obstacles and react on-the-fly to dynamic changes in the workspace using limited memory and computation resources (400 MHz CPU).
2. Automatic trajectory optimization: it removes the fastidious optimization process which is traditionally executed by hand by handling it automatically.
3. Change the way of programming industrial robot applications using the proposed generator, the robot can be completely programmed with tasks and priorities.

This is an appealing and challenging development for industrial robots. It will bring new utilities in the world of robotics. The novelties of the proposed framework revolve around the following:

First, it provides a novel approach to solve minimum time control problem for discrete time system subject to linear constraints in the sate and control variables. The proposed approach differs from existing ones in: (i) it does not rely on an ad-hoc selection of weighting factors (which is highly non-trivial), (ii) it does not lead to any approximation and results in time-optimal behavior for arbitrary linear constraints, (iii) and yet it is tractable in real-time.

Second, the novelty in this thesis is in the way we integrate the proposed time optimality problem in a task priority framework to solve a nonlinear optimization problem efficiently in real time using an embedded system with limited resources. Our formulation hinges on recent developments of efficient hierarchical solvers in the field of robotics [22, 26, 33] and integrates seamlessly in existing hierarchical control frameworks. Our approach is applied in a Model Predictive Control (MPC) setting, which not only improves reactivity of the system but presents a possibility to obtain accurate local linear approximations of the collision avoidance constraint.

All the techniques presented in this thesis are validated through various simulation and real-world robot experiments. The experiments conducted on a Cobra s600 SCARA Adept robot figure 1.4. The underlying optimization problems for robots were solved on a PowerPC CPU of 400 MHz.

1.4 Thesis Outline

Here is a brief overview of the following chapters:

Chapter two: Theoretical Background

Chapter 2 reviews several techniques in robotics that share some common ground with the research work considered in this thesis.

Chapter three: Obstacle Avoidance as an Optimization Problem

Chapter 3 reviews the state of the art on controlling robots in the presence of obstacles. Then it introduces a reactive trajectory generator based on a hierarchy between a strict obstacle avoidance behavior, and a control law which is time optimal in the absence of obstacles. The algorithm will be validated through various simulations and real world robot experiments.

Chapter four: An MPC Approach to Time Optimal Control

Chapter 4 presents two different numerical approaches to find the minimum time control for discrete time systems. These approaches are applied in a Model Predictive Control (MPC) setting. The first will be modeled as a Linear Problem (LP) and the second, as a hierarchical optimization Problem.

Chapter five: Implementation and Validation in the Presence of Obstacles

Chapter 5 validates the MPC approaches presented in Chapter 4 by applying them to online trajectory generation for industrial robots performing pick and place operations in the presence of dynamic obstacles. Then we evaluate the different approaches presented in this thesis in terms of time optimality, reliability and computation time.

Chapter six: Conclusion

Chapter 6 concludes this dissertation by summarizing its contributions, discussing its limitations and mentioning the possible future developments and research directions.

Chapter 2

Theoretical Background

The work developed in this thesis is based on a number of techniques taken from different research domains. In this chapter, we provide a brief overview of these techniques which is important for understanding the remainder of the thesis. This overview is not exhaustive and is not aimed to provide a complete account of what has been done within this domain. Instead, it is intended to provide the reader with enough information to situate this work among the relevant state of the art approaches. The structure of this chapter will be as follow: Section 2.1 presents an overview of different trajectory planning approaches. In Section 2.2, we start by providing an overview of standard constrained optimization techniques. Then a brief survey of numerical approach in the field of optimal control and nonlinear optimization will be given. Finally in Section 2.3, we review the techniques that can be used to compute the shortest distance between different objects.

2.1 Trajectory Planning

Trajectory planning is a major research area in robotics. Research in this area started in the 1970s. It deals with the problem of finding a collision free path from an initial state to a final state given a complete description of robot's physical limitations, geometry and environment. In this section, we start by showing the difference between path planning and trajectory planning. Then, a brief overview about local and global planning approaches will

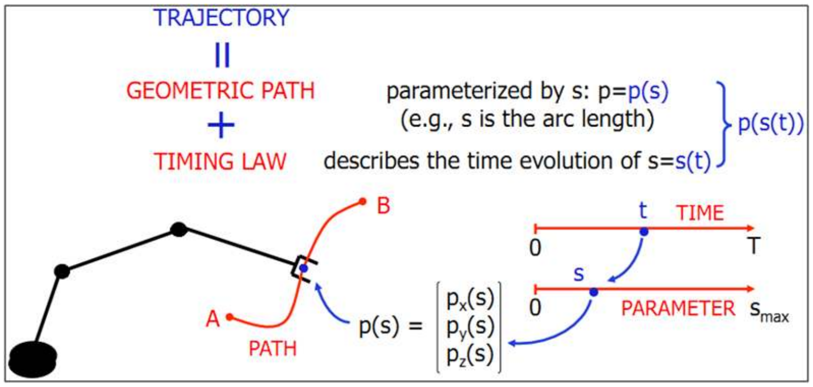

TRAJECTORY
=
GEOMETRIC PATH
+
TIMING LAW
parameterized by s: p=p(s)
(e.g., s is the arc length)
describes the time evolution of s=s(t)
p(s(t))
B
A
PATH
p(s) = [px(s), py(s), pz(s)]
t
TIME
0
T
s
0
PARAMETER
smax

Figure 2.1: The standard procedure for trajectory planning in industrial robots. Reprinted from [58].

be presented. Finally, we go discuss the positive and negative sides of defining trajectories in operational space or joint space.

2.1.1 Path Planning vs Trajectory Planning

Path planning is the geometric description of the robot motion, usually an interpolation of a set of points in space that the robot must follow. The velocity, acceleration and jerk profile for these points are called the Motion planning; it is also called the time law of the trajectory.

Path planning and trajectory generation are often used synonymously but path planning is actually a part of trajectory generation. Trajectory generation is formed by both path and motion planning. Standard algorithms in trajectory generation for industrial robots are illustrated in figure 2.1. The path is defined in a parametric form as $p = p(s)$. The parameter is defined as a function of time and represents the curvilinear abscissa of the path also called the time law. The curvilinear abscissa $s(t)$ is defined by specifying the path and the robot constraints such as velocity, acceleration and jerk limitations.

The algorithm proposed in this thesis generates trajectories directly without this distinction between the path and the time law. It determines the control signals that will cause a system to satisfy the physical limitations of the robot (e.g velocity and acceleration constraints) and its environment constraints (obstacles avoidance), and at the same time minimizes some performance criteria. The performance criterion chosen here is the time needed for the robot to reach its destination.

2.1.2 Global Planning vs Local Planning

Many path planning algorithms have been presented in the literature, they can be divided in two categories [16]: global and local algorithms.

Global path planning involves strategies to find the shortest collision free path for a complex work space respecting all the constraints of the robot. It will find the path based on a known environment map. A large number of approaches have been proposed, the main two categories are cell decomposition[57, 71] and road maps[45, 39]. Cell decomposition divides the free space into regions called cells. A path then consists of a sequence of cells and points at which the transition from on cell to another occurs. Road maps represent the free space in the form of nodes that form a graph. The edges that join the nodes represent the relationship between the nodes. A path is generated by connecting the initial and destination state of the robot to the road map and sequentially connecting the resulting states by applying a graph search method. Despite the possibility to parallelize the algorithms in order to reduce computation time [87], it is still too large to consider online. Moreover, in the case of unexpected obstacles, a replanning would be required. Thus, considering the limited resources (CPU, memory) in industrial robots and the requirements of planning in real time in a dynamic environment, global approaches are not an appropriate solution.

On the other hand, local or reactive trajectory planning needs less computation time, and there is no need for prior information on the environment. It will react to obstacles and generate collision free paths in real time. So it is suitable for online planning in dynamic environments. However, due to the lack of prior information on the environment, the motion can stop in a local minimum even if a collision free path does exist.

In this thesis, we propose a reactive trajectory planning algorithm that works in real time avoiding dynamic obstacles. Chapter 3 introduces techniques to reduce the probability of falling into local minima. In Chapter 5 a new trajectory generator will be designed based on a model predictive control approach (MPC). This will anticipate the future and help avoid falling into a local minima by reacting to it in advance.

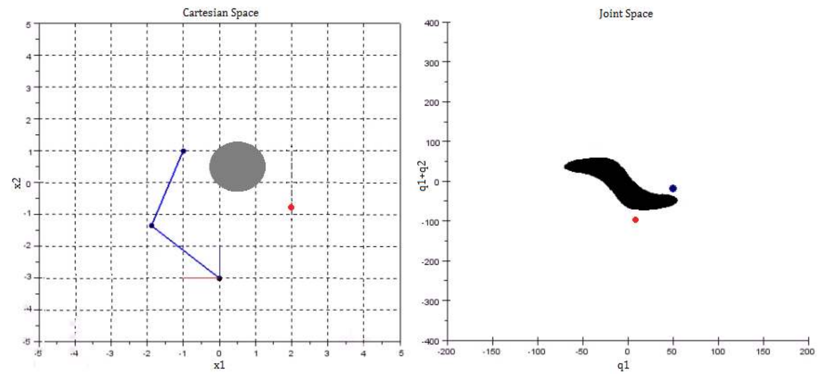


Figure 2.2: In the left, (Cartesian Space) the robot is blue, the obstacle is gray and the destination is the red point. In the right, (Joint Space) the end effector of the robots is the blue point, the destination is the red point and the obstacle is the black shape.

2.1.3 Joint Space vs Operational Space

Trajectories can be defined either in operational (Cartesian) space or joint space. For articulated manipulators, defining trajectories in joint space is computationally simpler, no need for online kinematic inversion. In addition, since the actual control is in joints, applying constraints in joint position, velocity, and acceleration will be straightforward. On the other hand, trajectories in Cartesian space are easy to describe where the motion of the manipulator is completely specified while trajectories in joints space need forward kinematic inversion to know the actual Cartesian position of the robot. However, since trajectories are generated in Cartesian space, care must be taken that the trajectories do not pass through, or come close to singularities.

In this thesis, we choose to define trajectories in joint space for two reasons. Firstly, this allows simple application of constraints on joint range, velocity and acceleration. Secondly, in articulated manipulators, the actuators are in the joints so to satisfy the time optimality the optimization should be applied to the joints directly, which is not the case if we choose to work in Cartesian space. However, obstacle avoidance constraints are expressed in Cartesian space, where the shortest distance to the obstacle is calculated and controlled. We choose to construct the obstacle avoidance constraints in Cartesian space, because finding the shortest distance from robot to obstacle is easy and time saving in Cartesian space whereas it is difficult and time consuming in joint space. To find the shortest distance from robot to obstacle in joint space, the obstacle shape should be transformed to joint space (see figure 2.2), this transformation is time consuming and not easy to do.

2.2 Optimization

Trajectory planning can be seen as an optimization problem where some performance criteria has to be optimized while respecting physical and environment constraints: finding the best solution among all feasible solution. The standard form is:

$$\begin{aligned} &\underset{x}{\text{minimize}}\ f(x) \\ &\text{subject to}\ g_i(x) \leq 0, \quad i = 1, ..., n, \\ &\qquad\qquad h_i(x) = 0, \quad i = 1, ..., m \end{aligned} \tag{2.1}$$

where $f(x)$ is the objective function to minimize over the variable x, the functions $g_i(x)$ are the inequality constraints and $h_i(x)$ are the equality constraints. The goal is to find an optimal value x^* such that $f(x^*)$ has the smallest value in the feasible set that satisfies all the constraints. Since the late 1940s, a large effort has gone into developing algorithms for solving various classes of optimization problems.

The optimization problem is called linear programming (LP) [17, 18], if the objective and the constraints functions are linear, satisfying

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \tag{2.2}$$

for all $x, y \in R^n$ and all $\alpha, \beta \in R$. A general linear program has the form

$$\begin{aligned} &\underset{x}{\text{minimize}}\ c^T x \\ &\text{subject to}\ a_i^T x \leq b_i, \quad i = 1, ..., n, \end{aligned} \tag{2.3}$$

where $a \in R^n$ and $b \in R$. If the objective function or one of the constraints are not linear, it is called a nonlinear problem.

Optimization problems can be split also into another two categories: convex optimization where the objective and constraint functions are convex, satisfying

$$f(\alpha x+\beta y) \leq \alpha f(x)+\beta f(y) \tag{2.4}$$

for all $x,u \in R^n$ and all $\alpha,\beta \in R$ with $\alpha+\beta = 1, \alpha \geq 0, \beta \geq 0$. And non-convex optimization where there is at least one non-convex function. We can see from (2.2) and (2.4) that linearity is a special case of convexity so any linear program is a convex optimization problem. Another well known convex optimization problem is a quadratic programming (QP) where the objective function is quadratic, and the constraint functions are linear. It can be expressed in this form

$$\begin{aligned} &\underset{x}{\text{minimize}}\ \frac{1}{2}x^T Qx+p^T x \\ &\text{subject to}\ Ax \leq b \\ &\qquad\qquad Cx = d, \end{aligned} \tag{2.5}$$

where Q is a $n \times n$ real positive definite symmetric matrix, A is a $m \times n$ real matrix , and $b \in R^m$.

Various approaches are proposed to solve convex optimization: Interior point [42], active set [64], augmented Lagrangian [90] , and simplex algorithm [60]. In convex optimization, if there is an optimal solution, it will be the global solution [10], while in non-convex optimization it could be a local solution, where several local minima may exist and make solving non-convex problems harder to solve globally than convex problems. A solution x is a local minimum if the objective function is smaller than feasible nearby x, but it is not smaller than all points in the feasible set.

Section 2.2.3 presents different approaches to solve nonlinear optimization, while Section 2.2.1 will introduce Optimal control and an example of an optimal control problem will be presented in Section 2.2.2.

2.2.1 Optimal Control

Optimization is usually performed in a finite dimensional Euclidean space: the optimal solution will be a vector $x^* \in R^n$. However, some problems require that the optimization be performed in a real function space, an infinite dimensional space. The interest is in extremal functions that make the functional attain a maximum or minimum value. The problem written:

$$\begin{aligned} &\underset{x(t)}{\text{minimize}} \int_{t_1}^{t_2} f(x(t))dt \\ &\text{subject to } g(x(t)) \leq 0 \end{aligned} \tag{2.6}$$

The calculus of variations refers to the latter class of problems. Optimal control is an extension of the calculus of variations. The aim is to determine the control signal that will cause the system to satisfy the physical constraint and at the same time, optimize some performance criterion. Variables are separated into state variables **x** and control variables **u**, which are related via a set of differential equations forming the dynamics of the system. Further, the constraint functions could involve in this case both state and control variables. The standard form goes as follows. Minimize a cost function

$$\underset{u(t)}{\text{minimize}} \int_{t_1}^{t_2} f(t, x(t), u(t))dt \tag{2.7}$$

subject to the dynamic system

$$\dot{x}(t) = h(x(t), u(t)) \tag{2.8}$$

the boundary condition

$$x(t_1) = x_1 \tag{2.9}$$

and equality/inequality constraints

$$g(x(t), u(t)) \leq 0 \tag{2.10}$$

the solution is an admissible control $u^* \in u[t_1, t_2]$ which satisfies the constraints (2.8), (2.9), and (2.10) in such a manner that the cost function (2.7) has a minimum value.

Optimal control has developed into a well established research area and finds its applications in many scientific fields, ranging from mathematics and engineering to biomedical and management sciences. The maximum principle, developed in the late 1950s by Pontryagin and his coworkers [66] can properly be regarded as the birth of the mathematical theory of optimal control.

For some simple optimal control problems, finding the analytical solution by solving the necessary condition "Pontryagin Maximum Principle (PMP)" will be applicable. One well known example is solving a linear time optimal problem, it will be presented in 2.2.2 and we will use the solution and build on it in the proposed approach presented in Chapter 3. More complex optimal control problems, with nonlinear constraints on state and control variables will be too difficult for an analytical solution. Computational algorithms are inevitable in solving such problems. A brief survey of numerical methods in the field of optimal control will be proposed in Section 2.2.3.

2.2.2 Linear Time Optimal Problem

Consider a linear time-invariant system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.11}$$

with bounded input

$$u \in [-U, +U]. \tag{2.12}$$

we consider the problem of finding a piecewise continuous control $u(t)$ that brings the system from an initial state x_0 to the origin, in minimum time:

$$\begin{aligned}
&\underset{u}{\text{minimize}}\; J(u) = \int_{t_0}^{t_f} dt = t_f - t_0 = T. \\
&\text{subject to } \dot{x}(t) = Ax(t) + Bu(t) \\
&\qquad\qquad x(t_0) = x_0 \\
&\qquad\qquad x(t_f) = 0 \\
&\qquad\qquad u(t) \in [-U, +U].
\end{aligned} \tag{2.13}$$

The solution of this time optimal control problem (2.13) is the well-known bang-bang control. The derivation of the optimal solution based on Pontryagin Minimum Principle (PMP) will be presented briefly in this section.

The Hamiltonian function can be defined for this problem as

$$H(x(t), u(t), p(t), t) = 1 + p(t)^T [Ax(t) + Bu(t)] \tag{2.14}$$

Then based on PMP [80], a necessary condition for u^* to be an optimal control is

First

$$u^* = arg \min_{u \in [-U, +U]} H(x^*(t), u(t), p(t), t) \tag{2.15}$$

Second

$$\dot{p}(t) = \frac{\partial H}{\partial x} \tag{2.16}$$

where $p(t)$ can be seen as Lagrange multipliers, but it is more commonly known as the costates of the system. From condition (2.15), we conclude that:

$$u^* = \begin{cases} +U, & p(t)^T B < 0, \\ -U, & p(t)^T B \geq 0. \end{cases} \tag{2.17}$$

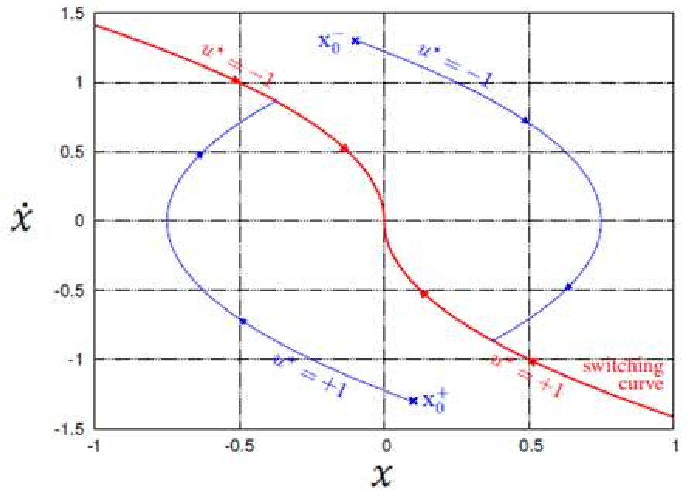


Figure 2.3: The state space of a Bang-bang controller.

for each $t_0 \leq t \leq t_f$, and from condition (2.16), satisfied so

$$\dot{p}_1(t) = -\frac{\partial H}{\partial x} = 0 \qquad \Rightarrow p_1(t) = C_1 \tag{2.18}$$

$$\dot{p}_2(t) = -\frac{\partial H}{\partial \dot{x}} = -p_1(t) \qquad \Rightarrow p_2(t) = -C_1 * t + C_2 \tag{2.19}$$

where C_1 and C_2 are constants. The switching function (2.19) is a linear function of time, this indicates that the time-optimal control is a bang-bang control, where the control variable switches between its maximum and minimum values.

For the time interval on which $u(t) = +U$

$$\ddot{x} = +U \tag{2.20}$$

$$\dot{x} = Ut + K_1 \tag{2.21}$$

$$x = \frac{U}{2}t^2 + K_1 t + K_2 \tag{2.22}$$

$$\Rightarrow x = \frac{\dot{x}^2}{2U} + K \tag{2.23}$$

Thus, the portion of the optimal response for which $u(t) = +U$ is an arc of a parabola, along which the phase points move upwards ($u > 0$). Analogously, the portion of the optimal response for which $u(t) = -U$ is an arc of a parabola, along which the phase points move downwards $(u < 0)$ (see figure 2.3). Consequently, the control signal $u(t)$ is chosen as follows:

$$u^* = -Usgn(\sigma), \tag{2.24}$$

$$\sigma = x + \frac{|\dot{x}|\,\dot{x}}{2U}, \tag{2.25}$$

where $sgn(.)$ is the sign function and $\sigma = 0$ is the switching curve.

2.2.3 Nonlinear Optimization

Many methods have been developed to solve trajectory optimization methods. They can be categorized in three basic families of approaches [20]: state-space, indirect, and direct.

The state-space approach starts from the principle of optimality of sub-arcs where each sub-trajectory of an optimal trajectory is an optimal trajectory as well. This is the basics of dynamic programming in discrete time. The equivalent continuous time case is called the Hamilton-Jacobi-Bellman (HJB) equation. The main idea is to discretize the work environment into a grid of cells and then attempt to solve the optimization problem by solving a large number of sub-problems. For example, in path planning for a robot, the problem can be solved by dividing the work space into cells and calculating the distance to the goal in each cell. Then the cell is searched to find the optimal path from start to goal. Several techniques are proposed in the literature to solve this problem: A* algorithm [77], Rapidly-exploring random trees (RRT) [55], probabilistic road maps (PRM) [45], Artificial Neural Network [94], and genetic algorithms [59]. In this kind of approaches the true global solution of the optimization problem is found even for non convex optimization problem. The worst-case complexity of global optimization methods grow exponentially with respect to the dimension of the state space, and limits the use of such approaches to applications where computation time is not critical. This drawback has been well known since the origins of optimal control theory, and it was what Richard Bellman referred to as the “curse of dimensionality” [4, 5].

Indirect approaches use the necessary conditions of the optimality problem (PMP) to transfer the optimal control problem into a boundary value problem (BVP) [89] and then solve the problem numerically. This approach is often called as "first optimize, then discretize". The main drawbacks of this method is that the underlying ordinary differential equations (ODE) are often difficult to solve due to non linearity and the control structure complexity.

Direct methods transform the optimal control problem into an equivalent nonlinear programming problem. Then the solution can be found using numerical optimization methods [64]. This approach is often called as "first discretize, then optimize". Compared to indirect methods, direct methods can more easily take into account inequality constraints. They have better convergence properties and are more stable even with poor initial guess. For solutions of constrained optimal control problems in real world applications, direct methods are by far the most widespread and successfully used techniques [20].

One of the most effective methods for nonlinearly constrained optimization, generates steps by solving quadratic subproblems. This sequential quadratic programming (SQP) approach is appropriate for small or large problems [64]. SQP methods show strength when solving problems with significant nonlinearities in the constraints. This is the case in this thesis, where collision avoidance constraints are nonlinear. A sequence of linearized sub-problems is solved using an SQP type of approach. Each sub-problem identifies a minimum-time trajectory from the current stat of the robot with respect to local linear approximations of the collision avoidance constraints.

Time optimal trajectories for linearized sub problems will be obtained by applying a hierarchical approach where a particular choice of lexicographic objective is introduced to guarantee a general solution to minimum time problem in discrete time system. Chapter 4 will explain in detail this approach. To solve efficiently the hierarchical problem, we use a recent solver [22, 26] based on a novel matrix factorization that can be cheaply updatable. In Chapter 5 a brief description about this solver will be given. Our approach is applied in a Model Predictive Control (MPC) setting, which not only improves reactivity of the system but presents a possibility to obtain accurate local linear approximations of the collision avoidance constraints. Next section will give a brief overview of MPC.

2.2.4 Model Predictive Control

Model predictive control (MPC) is an advanced method of control that has been used increasingly since the 1980s in several industrial sectors, especially in chemical plants and oil fields. It is a model-based control scheme that defines an open loop optimal control problem which is easier to solve than the original problem and then iteratively solves it online, using the current state of the system as initial condition, providing in the end a closed loop feedback controller.

MPC can be solved efficiently as nonlinear optimization problem using direct optimal control methods. MPC algorithms typically exploit the fact that consecutive optimal control problems are similar to each other. This allows to initialize the solver procedure efficiently by a suitably shifted guess from the previously computed optimal solution, saving considerable amounts of computation time.

2.3 Distance Computation

The closest points between robot and obstacles must be calculated and used to formulate collision avoidance constraints. Determining the minimum distance between objects is a common problem in robotics. Many approaches have been proposed to compute the shortest distance between objects, especially between two convex polyhedrals [7, 37, 56]. In our choice of algorithm, computation time will be a driving aspect. For this reason, we choose to simplify the shape of the robot and the obstacles to a composition of spheres and swept sphere lines as suggested in [81], and then use these simple models to calculate the closest points. For more complex shapes, we present in Section 2.3.2 a simple algorithm that formulates the shortest distance computation as an optimization problem. Section 2.3.1 will present some properties of the distance to obstacle function. Then Section 2.3.3 will end by showing a solution to discontinuity that may happen in distance function.

2.3.1 Distance Notation

Finding the shortest distance between two sets of points is equivalent to finding the infimum of the distances between these sets:

$$\inf_{y \in O} d_0(x, y) \tag{2.26}$$

where O is representing the obstacle set, x is the robot set(assuming it is a point), and $d_0(x, y)$ is the distance between two points. If the obstacle is a parametric function

$$\begin{aligned} &\underset{x,y,\tau}{\text{minimize}}\ d_0(x, y) \\ &\text{subject to } y = f(\tau) \\ &\qquad x = x_c, \end{aligned} \tag{2.27}$$

The Lagrangian equation and the necessary condition

$$L(x, y, \tau) = d_0(x, y) + \lambda_1^T (y - f(\tau)) + \lambda_2^T (x - x_c) \tag{2.28}$$

$$\frac{\partial L}{\partial x} = \frac{\partial d_0(x,y)}{\partial x} + \lambda_2 \qquad = 0 \tag{2.29}$$

$$\frac{\partial L}{\partial y} = \frac{\partial d_0(x,y)}{\partial y} + \lambda_1 \qquad = 0 \tag{2.30}$$

$$\frac{\partial L}{\partial \tau} = -\lambda_1^T \frac{\partial f(\tau)}{\partial \tau} \qquad = 0 \tag{2.31}$$

$$\frac{\partial L}{\partial \lambda_1} = y - f(\tau) \qquad = 0 \tag{2.32}$$

$$\frac{\partial L}{\partial \lambda_2} = x - x_c \qquad = 0 \tag{2.33}$$

from (2.30), (2.31) and the symmetric property of the distance function

$$\frac{\partial d_0(x,y)}{\partial y} = \frac{\partial d_0(x,y)}{\partial x}$$

$$\frac{\partial d_0(x,y)}{\partial x} \cdot \frac{\partial f(\tau)}{d\tau} = 0 \tag{2.34}$$

knowing that

$$d_0(x,y) = \|y-x\| = \sqrt{(y-x)^T(y-x)} \tag{2.35}$$

$$\frac{\partial d_0(x,y)}{\partial x} = \frac{-(y-x)}{\sqrt{(y-x)^T(y-x)}} \tag{2.36}$$

$$\Rightarrow \left\| \frac{\partial d_0(x,y)}{\partial x} \right\| = 1. \tag{2.37}$$

Then $\frac{\partial d_0(x,y)}{\partial x}$ is a unit vector perpendicular to $\frac{\partial f(\tau)}{d\tau}$ at the shortest distance. The variations of this distance will be

$$d = d_0(x,y) \tag{2.38}$$

$$\dot{d} = \frac{\partial d_0(x,y)}{\partial x}\dot{x} \tag{2.39}$$

$$\ddot{d} = \frac{\partial d_0(x,y)}{\partial x}\ddot{x} + \frac{d}{dt}\left[\frac{\partial d_0(x,y)}{\partial x}\right]\dot{x} \tag{2.40}$$

$$= \vec{a}\ddot{x} + \frac{d}{dt}[\vec{a}]\dot{x} \tag{2.41}$$

$$= \vec{a}\ddot{x} + \vec{b}\dot{x} \tag{2.42}$$

where $\vec{a}$ is a unit vector in the direction of the obstacle and the time derivative of any unit vector is a vector orthogonal to it $\vec{b}$. Thus if the distance function is convex, the projection of the velocity $\vec{b}\dot{x}$ takes the robot away from the obstacle so controlling $\vec{a}\ddot{x}$ as proposed in Chapter 3 is enough to avoid a convex obstacle.

2.3.2 Distance computation as an optimization problem

There are different ways to formulate the shortest distance computation as an optimization problem, here we give a brief overview of two methods. More details about Geometric optimization can be found in [73].

The first way is to define polytopes as intersections of half spaces. Consider there are two sets of half spaces $G = (g_1, ..., g_n)$ and $H = (h_1, ..., h_n)$ in dimension d, the goal is to find points $g \in \bigcap_{i=1}^{n} g_i$ and $h \in \bigcap_{j=1}^{m} h_j$ such that $\|g - h\|$ is minimized. Knowing that a point x is contained in the halfspace g_i if the function $g_i(x)$ defining the corresponding hyperplane is non-positive, the optimization problem will be

$$\begin{aligned} &\text{minimize } (x-y)^T(x-y) \\ &\text{subject to } g_i(x) \leq 0, \quad i = 1, ..., n \\ &\qquad\qquad h_i(y) \leq 0, \quad i = 1, ..., m, \end{aligned} \tag{2.43}$$

Another way is to consider polytopes as a set of points. $P = (p_1, ..., p_n)$ and $Q = (q_1, ..., q_m)$ are two point sets in d-dimensional space, the goal is to minimize $\|p - q\|$ such that p and q are points belong to the convex hull $p = \sum_{i=1}^{n} \lambda_i p_i$ and $q = \sum_{i=1}^{m} \mu_i q_i$ where λ_i and μ_i are in $[0, 1]$ and sum up to one. The optimization problem will be:

$$\begin{aligned} &\text{minimize } x^T C^T C x \\ &\text{subject to } \sum_{i=1}^{n} x_i = 1 \\ &\qquad\qquad \sum_{i=n+1}^{n+m} x_i = 1 \\ &\qquad\qquad x \geq 0, \end{aligned} \tag{2.44}$$

where $C = (p_1, ..., p_n, -q_1, ..., -q_m)$.

2.3.3 Strictly and Non Strictly Convex Obstacle

The closest points between two strictly convex objects move continuously. It is not the case for non-strictly convex objects. It is proposed in [43] tp keep track of several pairs of points that move continuously on the facets of the polyhedral composing the obstacles and the robot (see figure 2.4). In [27] a method to convert a polyhedral convex hull into a strictly convex hull is presented. The idea is to prevent jumps of the pair of closest points by curving them. This strictly convex hull/shape will guarantee a unique closest point and at least C_1 continuity of the distance function (see figure 2.5). Thus the gradient of the

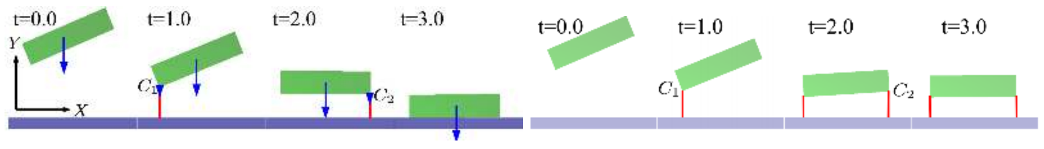


Figure 2.4: (a) Example of a discontinuous constraint. (b) Another pair of the closest points constrained so that the constraints move continuously. Reprinted from [43]

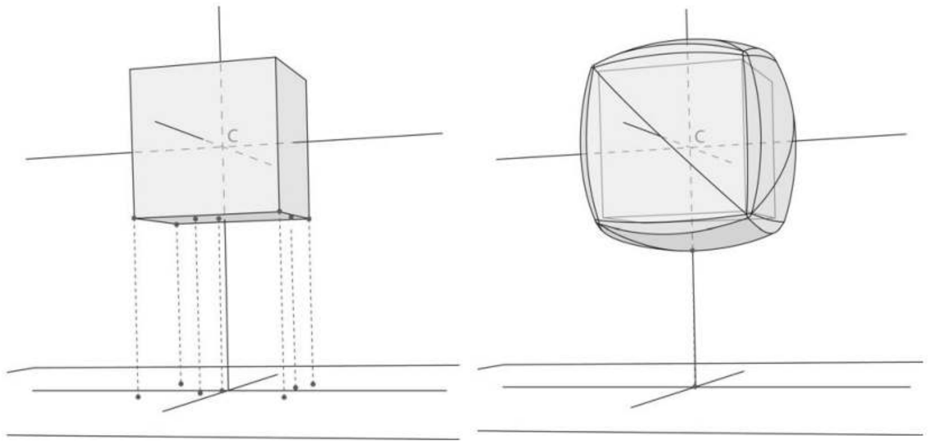


Figure 2.5: (a) Parallel faces with infinite number of closest points. (b) Removing the flat part by curving it then only one pair of closest points is considered. Reprinted from [27]

distance function is continuous, which is useful for integrating distance as a constraint in robotic motion planners.

2.4 Discussion and Conclusion

In this chapter, we have reviewed several techniques in robotics that share some common ground with the research work considered in this thesis. Here, we aim at presenting the current challenging requirements considered in this thesis, and pointing out the contribution of the presented work in this regard.

This thesis aims at providing an online trajectory generator for industrial manipulators performing tasks in a dynamic environment. Specifically, our goal is to design a reactive trajectory generator that achieves the following requirements: 1) Move as fast as possible: generates trajectories that are time optimal in absence of obstacles and as close as possible from optimality in presence of obstacles. 2) Generates collision free path: the robot must not collide with static and dynamic obstacles. 3) Works in real time: the algorithm must generate adequate control action online each 16 ms in the worst case, using limited computation resources (CPU of 400 MHz). 4) The robot must arrive precisely at the commanded destination without any vibration or overshoot. 5) Satisfy the physical limitations of the robot: velocity, acceleration, jerk and torque constraints.

We have presented an overview of trajectory planning approaches in Section 2.1 Global planning algorithms need high computation power and cannot react in real time to any change in the environment. This type of trajectory generation will break the goal number 3) and can't do 2) in case of moving obstacles. On the other side, local planing algorithms will provide such reactivity without the need of high computation power, but they introduce local minima and none of the algorithm in the literature tackle the question of time optimality. In this thesis, we develop a reactive trajectory algorithm that plan a bit in advance, with a certain horizon based on model predictive control, and provide a practical way of generating locally time optimal solutions.

Since the collision avoidance constraints are in general nonconvex, the trajectory optimization problem to be solve in this thesis will be a nonlinear optimization problem. In Section 2.2, we gave an overview of different approaches used in the literature to solve such problems. As outlined there, for solving such problems in real world applications, direct methods are by far the most widespread and successfully used techniques. However, solving nonlinear optimization problems online on embedded system with limited resources (CPU 400 MHz) is still a challenging problem. The novelty in this thesis is in the way we formulate the time optimality problem and integrate it in a task priority framework (see Chapter 4). Our formulation hinges on recent developments of efficient hierarchical solvers in the field of robotics [22, 26, 33] and integrates seamlessly in existing hierarchical control frameworks. Thus, solving nonlinear optimization problems can be achieved in few milliseconds using 400 MHz CPU.

Chapter 3

Obstacle Avoidance as an Optimization Problem

We have presented in the first chapter the main motivation behind having a trajectory generator able to avoid dynamic obstacles, and react in real time to sudden changes. In this chapter, we propose a reactive trajectory generator based on a hierarchy between a strict obstacle avoidance behavior, and a control law which is time optimal in the absence of obstacles. The algorithm has low computational requirements, and is therefore suitable for robots lacking very high computational capabilities. The undesired equilibrium points, as may be generated by the optimization problem, are handled by introducing circular fields around the obstacles, with strength being inversely proportional to the squared distance between the robot and the obstacle. Simulation results and real experiments show the effectiveness of the proposed solution.

3.1 State of the art

An early approach for a reactive control of robots in the presence of obstacles was to use potential fields [48], with repulsive forces pushing away from obstacles, added to an attractive force pulling towards the goal. But this can lead the robot to get stuck in local minima and not reach the goal. Local minima can be avoided by combining local and global informations, as in the elastic band framework [67, 11], but this requires offline planning, and even re-planning in case the path being executed becomes infeasible at some point. Another option, not requiring offline planning, is to combine circular and potential fields [38]. But the main problem with these approaches is that for robots moving at high-speed, obstacles will be avoided only if the repulsive forces are strong enough, what can in turn lead to unstable oscillations [28].

Harmonic potential functions, simulating the dynamics of fluids around obstacles, are a way to avoid these problems [52, 29, 92], but they require that the motion of both the robot and obstacles follow harmonic functions, what can be too restrictive in practice. An alternative, based on the modulation of a dynamical system embedding the trajectory tracking task, doesn't have this limitation, but reintroduces some difficulties with local minima [47, 72]. Anyway, none of these approaches tackles the question of time-optimality.

Interestingly, this modulation approach is conceptually similar to the velocity damper proposed earlier in [28], which selectively damps the motion of a speed-controlled robot in the direction of obstacles, in a way that strictly enforces obstacle avoidance. The use of damping instead of repulsive forces avoids introducing oscillations. Hopefully, this earlier approach can be easily extended to consider time-optimal motion of acceleration-controlled robots. This will be the basis of the work described in this chapter.

A standard approach for optimal control problems is Dynamic Programming [6, 12], but it usually requires very intensive computations. Branch and bound methods such as the A* algorithm [77], or probabilistic approaches [45, 55] can accelerate these computations substantially, but still not enough for high-frequency control of fast robots in dynamic environments. A usual option then is to resort to sub-optimal solutions. This will be the goal of this chapter. A very interesting and efficient approach focuses on the bang-bang structure of time-optimal control to simplify the on-line computation of near time-optimal obstacle avoidance [76]. But avoiding multiple obstacles is realized with a heuristics which can fail, what is not acceptable for fast and therefore dangerous robots.

We propose therefore to begin with an analytical solution to the problem of time-optimal control of velocity and acceleration constrained systems in the absence of obstacles, as presented in [95]. And we propose to selectively limit the acceleration of the robot in the direction of obstacles in a way that strictly enforces obstacle avoidance, following the idea presented in [28]. The result is a control law which is time-optimal in the absence of obstacles, and near time-optimal in the presence of obstacles.

One way then to describe the obstacle avoidance constraint is the so-called Velocity Obstacle [31], recently generalized to the Acceleration-Velocity Obstacle [83], but this formulation is based on specific velocity and acceleration profiles (a constant velocity in the former case, a continuous acceleration in the latter one), which do not correspond to the profiles generated by time-optimal control. As a result, we are going to propose in this chapter a specifically adjusted formulation of the obstacle avoidance constraint.

The structure of this chapter is as follows: our proposition is presented first on a one-dimensional example in Section 3.2, before being applied to single planar mobile robots in Section 3.3, and manipulator robots in Section 3.4.

3.2 A one-dimensional example

In order to simplify the problem and put aside the difficulties coming from the complexity of manipulators and mobile robots, the proposed solution will be presented first on a double integrator system.

3.2.1 Minimum time control

Let us consider a one-dimensional standard double integrator

$$\ddot{x} = u \tag{3.1}$$

with bounded input

$$|u| \leq U. \tag{3.2}$$

The classical minimum time control for this system to reach the origin $x = 0$, $\dot{x} = 0$ is the well-known bang-bang control which is presented in detail in Section 2.2.2.

$$u = -U\operatorname{sign}(\sigma) \tag{3.3}$$

with sign$(\cdot) = \pm 1$ the standard sign function and σ the switching curve

$$\sigma = x + \frac{\dot{x}|\dot{x}|}{2U}. \tag{3.4}$$

Continuous time system:

The control signal (3.3) supposes an ideal sliding on the switching curve, when the state $(x, \dot{x}$ reaches it, which is not the case in real life. The control signal $u(t)$ will oscillate at high frequency between the two values $\pm U$. Consequently, applying this controller in any real application will cause chattering in the input. The chattering can be reduced by introducing a boundary layer in the neighborhood of the switching curve $\sigma = 0$. This can be done by using in (3.3) a standard saturation function $sat(.)$ (3.6) instead of sign function $sgn(.)$:

$$u = -U sat(\lambda \sigma), \tag{3.5}$$

and

$$sat(x) = \begin{cases} -1, & x \leq -1 \\ +1, & x \geq +1. \\ x, & |x| \leq 1 \end{cases} \tag{3.6}$$

However, this solution does not guarantee the absence of overshoot. To cope with this, different techniques have been suggested in the literature e.g. proximate time-optimal servomechanisms (PTOS) [93, 19], nonlinear controller [63], variable structure sliding mode control [82]. In the controller (3.7), a boundary layer has been introduced around the switching curve $\sigma = 0$ by using the saturation function $sat(.)$. Moreover, the sliding mode variable σ defined in (2.25) has been properly modified in order to guarantee the absence of overshoot. (For more detail see [95]).

$$\begin{aligned} u &= -U sat(\lambda \sigma) \\ \sigma &= x + max(|\dot{x}|, k)\dot{x}/2U + \frac{3}{2\lambda} sat(\frac{\dot{x}}{k}). \end{aligned} \tag{3.7}$$

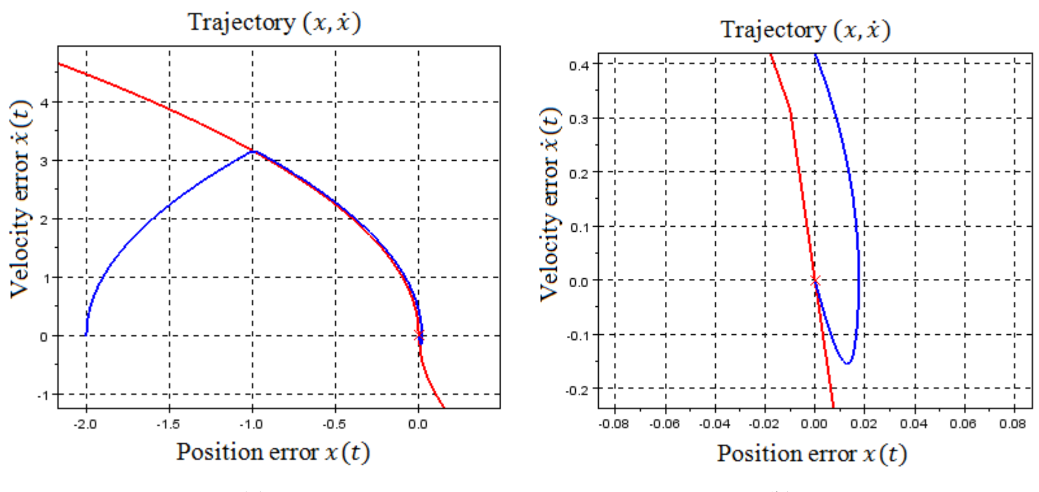


Figure 3.1: Phase space when a discretised version of controller in continuous time is used: (a) the trajectory of the tracking error; (b) a detail around the origin showing the overshoot.

Discrete time system:

Direct discretization of the previous controller (3.7) does not insure that the state x remains within the boundary layer during the switching phase, also overshoot may arise in the system (see figure 3.1). For a time sampled system like a computer controlled robot, such a minimum time control is slightly more involved, as shown in [35, 95]:

$$u_k = -U \operatorname{sat}(\sigma) \tag{3.8}$$

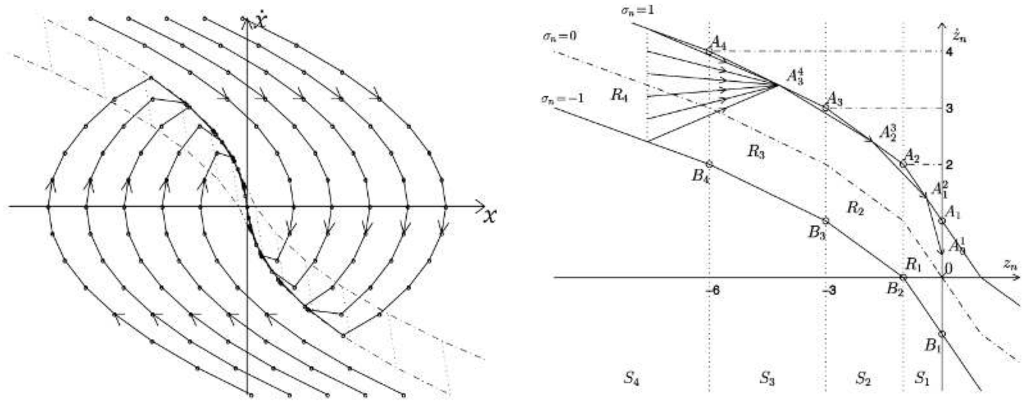
with $\operatorname{sat}(\cdot) \in [-1, 1]$ the standard saturation function and σ a modified, piece-wise linear switching curve

$$\sigma = \dot{z}_k + \frac{z_k}{m} + \frac{m-1}{2}\operatorname{sign}(z_k) \tag{3.9}$$

with

$$z_k = \frac{x_k}{\tau^2 U} + \frac{\dot{x}_k}{2\tau U}, \; \dot{z}_k = \frac{\dot{x}_k}{\tau U}, \tag{3.10}$$

$$m = \operatorname{Int}\left(\frac{1+\sqrt{1+8|z_k|}}{2}\right), \tag{3.11}$$



(a) (b)

Figure 3.2: (a) System trajectory when the discrete time controller designed in [95] is used. (b) Boundary layer around the switching curve, shows how the controller maps the system state to origin. Reprinted from [95].

where τ is the sampling period and $\text{Int}(\cdot)$ is the integer part (the truncation).

Equation (3.8) shows that the control signal u_k is saturated with value $\pm U$ all the time except when the system state $z = (z_k, \dot{z}_k)$ belongs to the boundary layer $|\sigma| < 1$ $(z \in R_m)$. The work of the controller proposed by [95] is to map any system state $z \in R_m$ into a segment A_{m-1}^{m} during the next sampling period τ by applying appropriate control signal u_k. Then the control law maps segment A_{m-1}^{m} into segment A_{m-2}^{m-1} and so on until the system state z reaches the origin (see figure 3.2). Finally, the system state z reaches the origin in a finite number of sampling periods τ without overshoot (For more detail see [95]).

In the following, we will refer to this minimum time controller as

$$u_k = f(x_k, \dot{x}_k, U) \tag{3.12}$$

and drop the sampling time index $(\cdot)_k$ when not necessary to simplify notations.

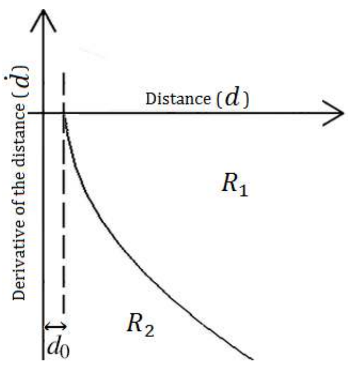


Figure 3.3: The distance state space and the switching curve generated by the constraint.

3.2.2 Obstacle Avoidance

Suppose that there is an obstacle at position x_O and the distance d to this obstacle,

$$d = a^T(x - x_O) \tag{3.13}$$

with

$$a = \frac{x - x_O}{|x - x_O|}, \tag{3.14}$$

must remain greater than a minimum safety distance d_0:

$$d \geq d_0. \tag{3.15}$$

We can observe that for the double integrator with bounded input (3.1)-(3.2) to always satisfy this minimum distance constraint, the variations of this distance (in this simple $1D$ case, $\vec{a}$ is supposed constant)

$$\dot{d} = a^T \dot{x},\ \ddot{d} = a^T u, \tag{3.16}$$

must satisfy

$$\ddot{d} \geq f(d - d_0, \dot{d}, U). \tag{3.17}$$

What we want therefore is to be as close as possible to the minimum time controller (3.12):

$$\underset{u}{\text{minimize}}\ |u - f(x, \dot{x}, U)|^2, \tag{3.18}$$

while always satisfying the obstacle avoidance constraint

$$a^T u \geq f(d - d_0, \dot{d}, U). \tag{3.19}$$

The proposed solution is an optimization (minimization) of a quadratic function subject to linear constraint on u. Without the constraint the result of the minimization is a bang-bang controller, with the constraint a switching curve appears in the distance state space $(d, \dot{d})$ as can be seen in figure 3.3.

When $(d, \dot{d})$ is in R_2, the robot will collide with the obstacle, there is not enough time and power to drive the robot away from the obstacle. On the other hand, when $(d, \dot{d})$ is in R_1, the constraint will be $a^T u \geq -U$. It doesn't affect the minimization because it is always satisfied ($a^T u \in [-U, +U]$ from (3.2) and (3.14)). However when $(d, \dot{d})$ is at the switching curve, the constraint will be $a^T u \geq +U$; knowing that $a^T u \in [-U, +U]$, the solution of this optimization problem should satisfy $a^T u = +U$ (decelerate in the direction of the obstacle) in order to make sure that the robot doesn't hit the obstacle.

3.2.3 Velocity limits

We propose to follow the approach proposed in [32] in order to deal with the velocity limit

$$\dot{x}_{min} \leq \dot{x}_{k+1} \leq \dot{x}_{max}. \tag{3.20}$$

With the time sampled dynamics

$$\dot{x}_{k+1} = \dot{x}_k + \tau u_k, \tag{3.21}$$

we easily obtain the corresponding bounds that apply on the control u_k:

$$\frac{1}{\tau}(\dot{x}_{min} - \dot{x}_k) \leq u_k \leq \frac{1}{\tau}(\dot{x}_{max} - \dot{x}_k), \tag{3.22}$$

which have to be added then to the optimization problem (3.18)-(3.19).

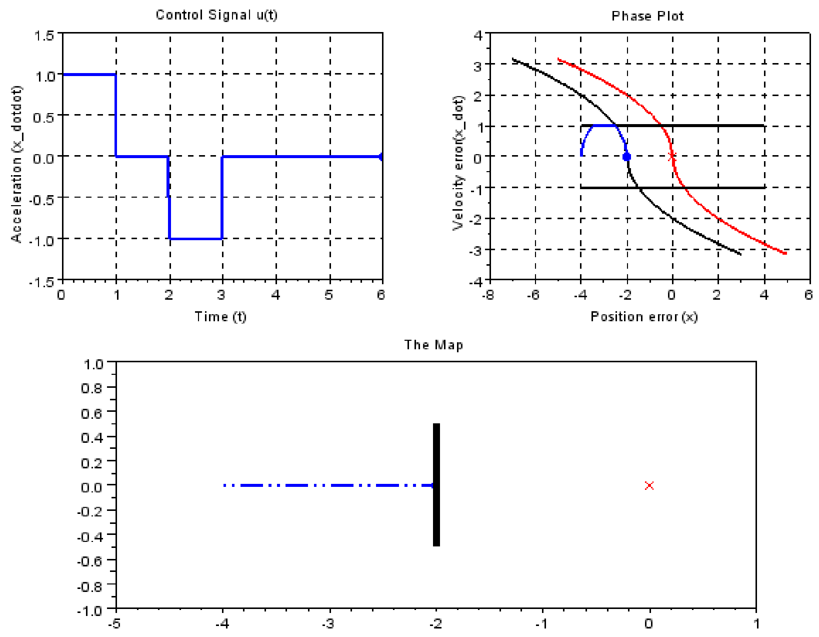


Figure 3.4: Control signal (upper left) and phase plot (upper right) where the switching curve for the minimum time controller appears in red, velocity constraints are the black horizontal lines, and the switching curve for the obstacle avoidance constraint is in black. We can see that the motion of the robot in blue always stays within the constraints as required. The lower figure shows the resulting trajectory, with the goal in red and the obstacle in black.

3.2.4 Simulation result

Consider a simple situation where the robot starts from $x = -4$ with zero velocity and tries to reach the origin $x = 0$, $\dot{x} = 0$ in minimum time, but there is an obstacle on the way at $x_O = -1$ (with security margin, $d_0 = 1$). The maximum acceleration and deceleration is $U = 1$ and the minimum and maximum velocities are -1 and $+1$. We can see the motion resulting from our controller in figure 3.4: the robot accelerates first until it reaches its maximum velocity, but the obstacle avoidance constraint imposes an early deceleration so that the robot stops before the obstacle at $x = -2$, unable to reach the target position $x = 0$.

3.3 Planar mobile robot

The approach proposed in the previous section is generalized in this section to planar mobile robots. The size of the problem will increase, the obstacle avoidance constraint will especially need to be modified, and the potential presence of undesired stable equilibrium points will be tackled. Furthermore, an application to two mobile robots sharing the same working space will be simulated using the proposed solution.

3.3.1 Minimum time control

Consider a two-dimensional double integrator

$$\ddot{x}_1 = u_1,\ \ddot{x}_2 = u_2, \tag{3.23}$$

with $x = (x_1, x_2)$, $u = (u_1, u_2)$ and bounded input

$$|u_1| \leq U,\ |u_2| \leq U. \tag{3.24}$$

In this case, the minimum time control may not be unique, and we will simply use two decoupled one-dimensional minimum time controllers

$$u_1 = f(x_1, \dot{x}_1, U),\ u_2 = f(x_2, \dot{x}_2, U) \tag{3.25}$$

what we will note

$$u = f'(x, \dot{x}, U). \tag{3.26}$$

3.3.2 Obstacle avoidance

Suppose that there is an obstacle at position x_O and the distance d to this obstacle,

$$d = a^T(x - x_O) \tag{3.27}$$

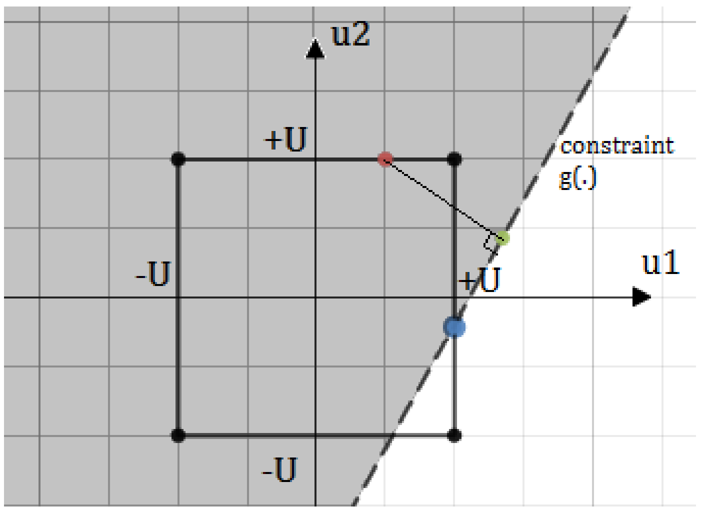


Figure 3.5: Constraints in the control space (u_1, u_2). Admissible controls can have a norm greater than U. If the unconstrained minimum of (3.38) is the red dot, we can see that only imposing the constraint (3.39) can lead to a solution, here the green dot, which does not satisfy the constraint (3.40). This constraint must therefore be enforced explicitly, giving as a solution the blue dot.

with

$$a = \frac{x - x_O}{|x - x_O|}, \tag{3.28}$$

must remain greater than a minimum safety distance d_0:

$$d \geq d_0. \tag{3.29}$$

The variations of this distance

$$d = \sqrt{x^T x} = \frac{x^T x}{d} = a^T x, \tag{3.30}$$

$$\dot{d} = \frac{2x^T \dot{x}}{2\sqrt{x^T x}} = \frac{x^T \dot{x}}{d} = a^T \dot{x}, \tag{3.31}$$

$$\ddot{d} = \frac{x^T \ddot{x}}{d} + \frac{1}{d}\left(\dot{x}^T \dot{x} - (a^T \dot{x})^2\right) \tag{3.32}$$

$$= a^T \ddot{x} + \frac{1}{d}(b^T \dot{x})^2, \tag{3.33}$$

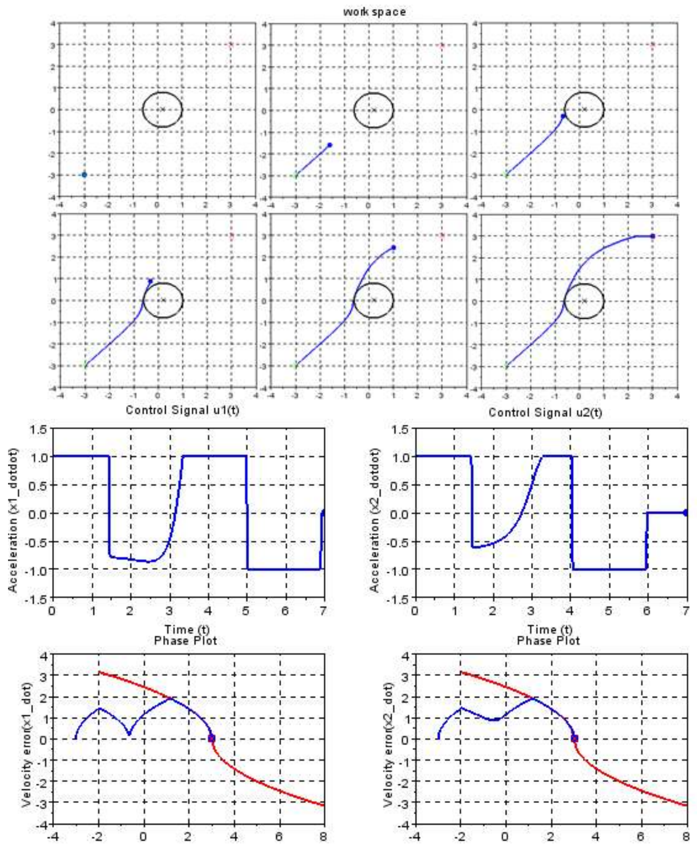


Figure 3.6: Trajectory of the robot, control signals and phase space plots in each dimension, with the minimum time controller switching curves in red.

with b a unit vector orthogonal to a , must satisfy a constraint related to the minimum time control $f(d-d_0,\dot{d},U)$. However, in the present case, some modifications are necessary.

As we can see in figure 3.5, the Euclidean norm of the input u can be greater than U, so the norm of the acceleration $\ddot{d}$ can also be greater than U. Hence, considering the same constraint as before,

$$\ddot{d} \geq f(d-d_0,\dot{d},U), \tag{3.34}$$

continuously imposing a lower limit $\ddot{d} \geq -U$, could slow down the robot unnecessarily. On the other hand, simply dropping this constraint when $f=-U$ quickly results in chattering. What we propose then is to base the obstacle avoidance constraint on a modified, unilateral saturation function:

$$\ddot{d} \geq g(d - d_0, \dot{d}, U) \tag{3.35}$$

with

$$g = -U \underline{\mathrm{sat}}(\sigma) \tag{3.36}$$

and

$$\underline{\mathrm{sat}}(\sigma) = \begin{cases} -1 & \text{if } \sigma \leq -1, \\ \sigma & \text{otherwise.} \end{cases} \tag{3.37}$$

Furthermore, we can also see in figure 3.5 that considering the optimization problem (3.18) with the constraint (3.35) can lead to control values u outside of the limits (3.24). These limits must therefore be enforced explicitly. In the end, the controller is based on the following optimization problem:

$$\underset{u}{\text{minimize}} \; \|u - f'(x, \dot{x}, U)\|^2 \tag{3.38}$$

s.t.

$$\ddot{d} \geq g(d - d_0, \dot{d}, U) \tag{3.39}$$

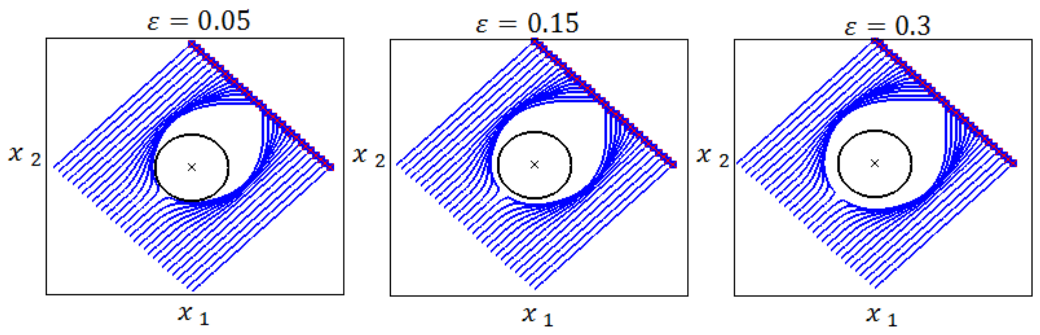
and

$$|u_1| \leq U, \; |u_2| \leq U. \tag{3.40}$$

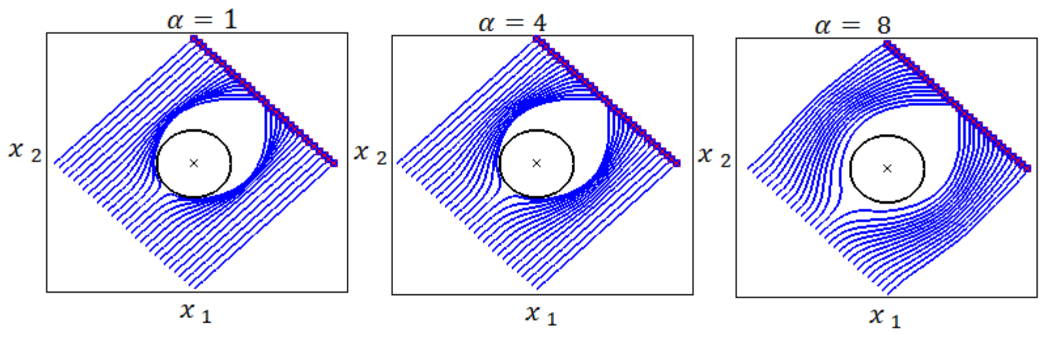
A simulation result can be seen in figure 3.6, where the robot is first heading directly at the obstacle, because its goal is right behind. The obstacle avoidance behavior is activated, and the resulting trajectory of the robot altered, only when there is no other choice to avoid collision. A truly time optimal behavior would have managed its way around the obstacle earlier to minimize in the end the time to reach the goal.

This obstacle avoidance behavior can be tuned however, to be activated earlier if desired. One option is to change the minimum safety distance, as shown in figure 3.7(a). Another option is to introduce a safety margin on the maximum deceleration with respect to obstacles, modifying the constraint (3.35) in the following way:

$$\ddot{d} \geq g(d - d_0, \dot{d}, U/\alpha) \tag{3.41}$$



(a)



(b)

Figure 3.7: (a) Varying obstacle avoidance behaviors for different minimum distance $d_0 = \varepsilon$. (b) Varying obstacle avoidance behaviors for different deceleration safety margins α.

with $\alpha \geq 1$, as shown in figure 3.7(b).

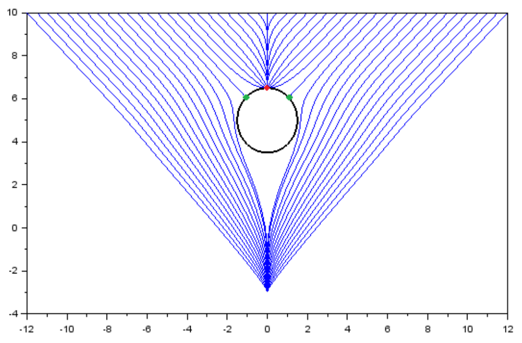
In the presence of multiple obstacles, the distances d_i to each obstacle i must be constrained independently:

$$\ddot{d}_i \geq g(d_i - d_0, \dot{d}_i, U), \tag{3.42}$$

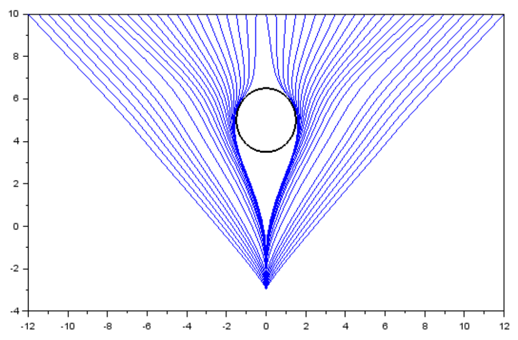
in order to ensure unconditional avoidance of all obstacles. Simulation results can be seen in figures. 3.9 and 3.10.

3.3.3 Undesired equilibrium points

In the absence of obstacles, the only equilibrium point for the control law (3.25) is the goal $(0,0)$. It is moreover globally stable, and always reached in finite time. But in the presence of obstacles, the constrained control law (3.38)-(3.40) potentially faces other stable and



(a)



(b)

Figure 3.8: (a) Undesirable stable equilibrium points may lie on the boundary of the obstacles when the obstacles and goals are aligned with the main coordinate axes and the angle of the boundary is more than 45 degrees. (b) The effect of applying a circular field around the obstacle.

unstable equilibrium points, on the boundary of the safety distance d_0 to obstacles. As a result, there is a risk that the robots get stuck there instead of reaching their goals.

With the control law (3.38)-(3.40), such equilibrium points arise when the obstacles are aligned with the goals along one of the main coordinate axes. In this case, equilibrium points will lie on the boundary of the obstacles. This can be seen for example in figure 3.8(a) for various initial positions.

One way to avoid these equilibrium points is to introduce circular fields around the surface of the obstacles, as in [38, 79]. This requires a small modification of the control objective (3.38), considering instead:

$$\underset{u}{\text{minimize}} \; \|u - f'(x, \dot{x}, U)\|^2 + w\|b^T u - U\|^2, \tag{3.43}$$

where b is the direction of the circular field generated around the obstacle, perpendicular to a, and $w = \frac{10}{d^2}$ is a weight which increases when approaching the obstacle. The effectiveness of this approach can be seen in simulation in figure 3.8(b).

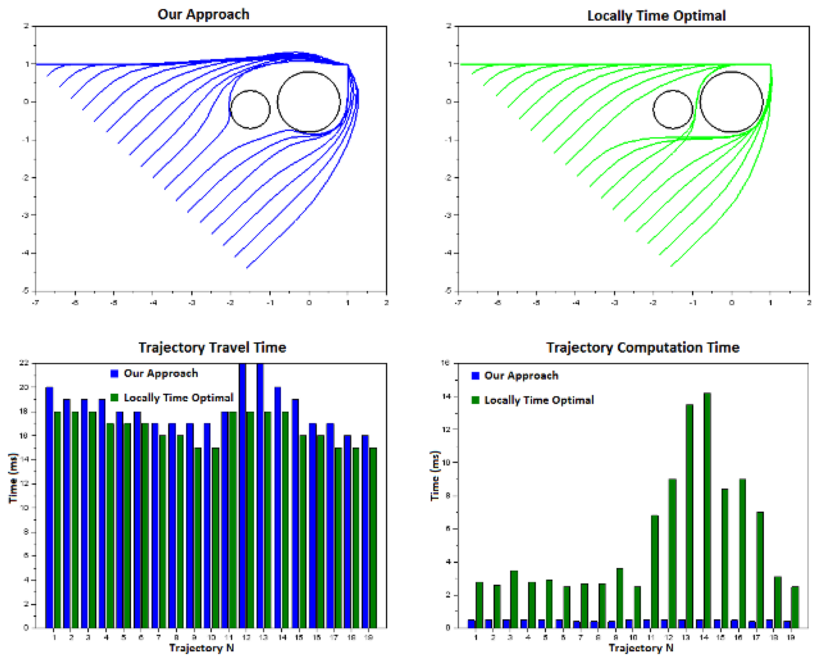


Figure 3.9: Avoiding two obstacles comparing Locally time Optimal approach Vs Our approach

3.3.4 Sub optimality

In the absence of obstacles, the control law (3.12) is time-optimal. In the presence of obstacles, we can assess the degree of sub-optimality of our controller by comparing the resulting behavior with time-optimal trajectories obtained with a standard local optimization method (Sequential Quadratic Programming). We can see in figure 3.9 that when there are few obstacles, the time required to reach the goal is always very close to optimal, with a computation time which is always kept very small. This is the standard situation for the industrial manipulator robots considered in the next section. However, when there are more obstacles, we can see in figure 3.10 that the sub-optimality degrades significantly. In such situations, more complex methods would be required to generate faster motions of the robot.

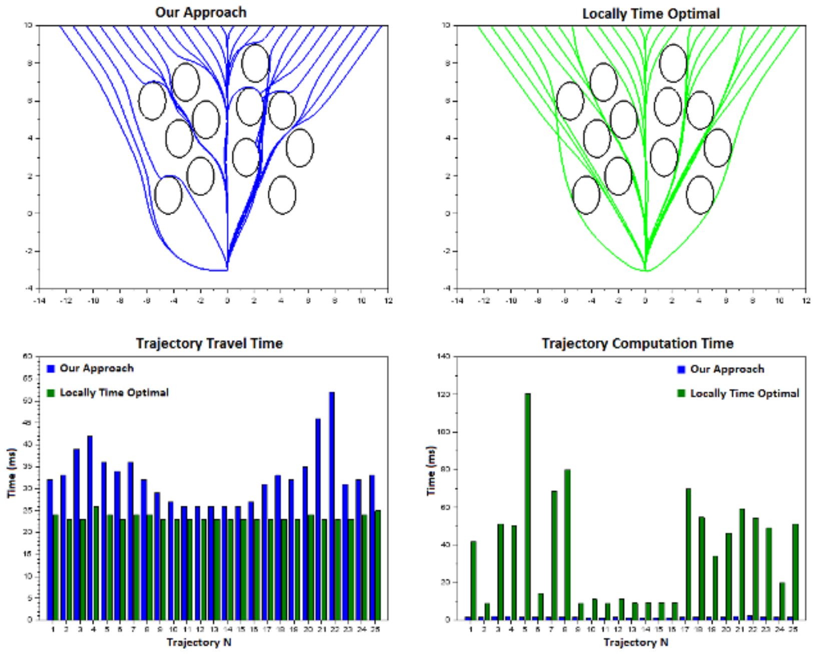


Figure 3.10: Avoiding 12 obstacles comparing Locally time Optimal approach Vs Our approach

3.3.5 Collision avoidance between cooperating robots

If two robots x and y have to share the same working environment, each one appears to be an obstacle for the other. In this case, a collaboration between the robots regarding collision avoidance can lead to improved performance. Such a collaboration can be approached in various ways, one way is to consider each robot as an independent system where the other robot will be seen as a dynamic obstacle. In this case, the solution proposed in the previous section can be applied directly. We propose here a joint controller for the two robots considered as a unique system.

Suppose the distance d between the two robots,

$$d = a^T(x-y) \tag{3.44}$$

with

$$a = \frac{x-y}{\|x-y\|}, \tag{3.45}$$

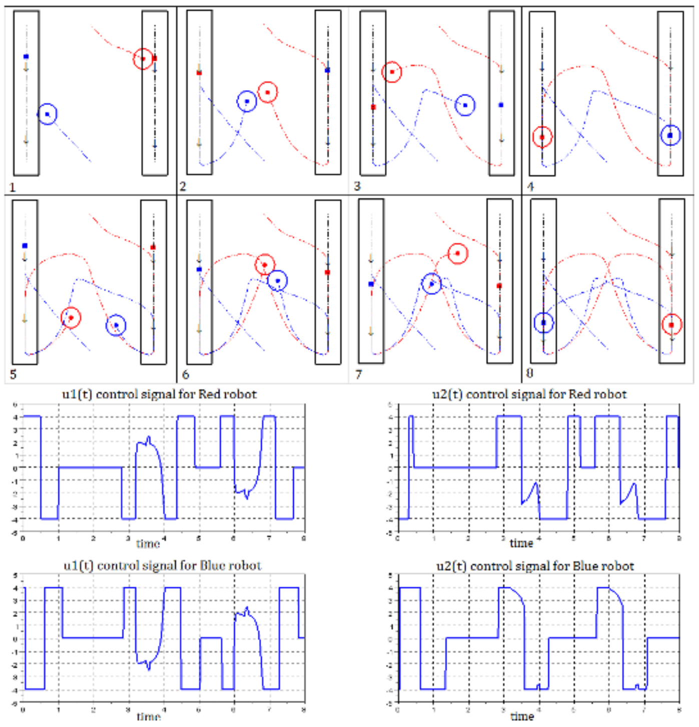


Figure 3.11: The red and blue robots should track respectively the small red and blue rectangles alternating between two treadmills. Control signals are given below.

must remain greater than a minimum safety distance d_0:

$$d \geq d_0. \tag{3.46}$$

Then, as before, the variations of this distance

$$\dot{d} = a^T(\dot{x} - \dot{y}),\ \ddot{d} = a^T(\ddot{x} - \ddot{y}) + \frac{1}{d}\left(b^T(\dot{x} - \dot{y})\right)^2 \tag{3.47}$$

must satisfy the collision avoidance constraint

$$\ddot{d} \geq g(d - d_0, \dot{d}, U). \tag{3.48}$$

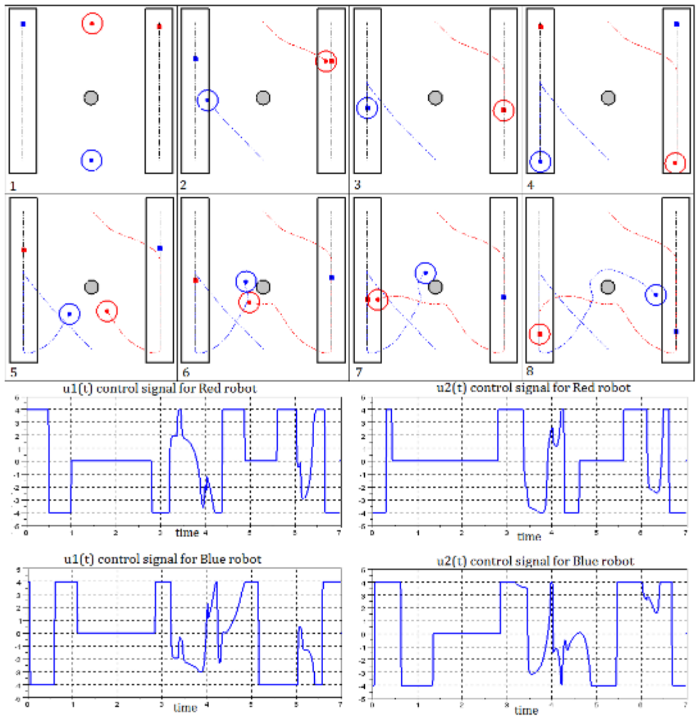


Figure 3.12: The red and blue robots should track respectively the small red and blue rectangles alternating between two treadmills. A grey static obstacle is added in the middle. Control signals are given below.

The joint controller takes then the simple form of a combined optimization problem

$$\underset{\ddot{x},\ddot{y}}{\text{minimize}} \; \|u - f'(x - x_d, \dot{x} - \dot{x}_d, U)\|^2 + \|v - f'(y_d, \dot{y} - \dot{y}_d, U)\|^2 \tag{3.49}$$

s.t.

$$\ddot{d} \geq g(d - d_0, \dot{d}, U) \tag{3.50}$$

and

$$|\ddot{x}_1| \leq U, \; |\ddot{x}_2| \leq U, \tag{3.51}$$

$$|\ddot{y}_1| \leq U, \; |\ddot{y}_2| \leq U. \tag{3.52}$$

Different weightings can be considered in (3.49) to give a higher priority to one robot or the other.

A simulation result where two mobile robots have their goals alternating between two treadmills is shown in figure 3.11 with $U = 4$. In figure 3.12, a static obstacle is added in the middle. In both cases, the two robots manage to reach their moving and alternating goals without collision, respecting their velocity and acceleration constraints through cooperation.

3.4 Manipulator Robots

The control scheme proposed in the previous sections can also be applied to industrial articulated manipulator robots. The motion of these robots is usually limited mostly by maximum joint range, speed and acceleration. The previous scheme can therefore be applied directly in the joint space:

$$\underset{\ddot{q}}{\text{minimize}}\ \|\ddot{q} - f'(q, \dot{q}, U)\|^2. \tag{3.53}$$

Obstacle avoidance is better approached, however, in Cartesian space.

3.4.1 Obstacle avoidance formulation

For collision avoidance, it is more appropriate to compute distances and closest points between robots and obstacles in Cartesian space, where it is difficult and time consuming in joint space as we saw in Section 2.1.3. Many methods for this have been proposed not only in robotics but also in computer graphics [24, 85].However the computation time will be a driving aspect in our choice. We will simplify the shape of the robot and the obstacles to a composition of spheres and swept sphere lines [81]. We will compute the closest point based on this model.

For simplicity, we consider a SCARA robot avoiding circular obstacle. Suppose that the obstacle is centered at position x_o with radius r. Let p_1 and p_2 be the points on the robot links that are closest to the obstacle, as illustrated in figure 3.13. Then the shortest distance

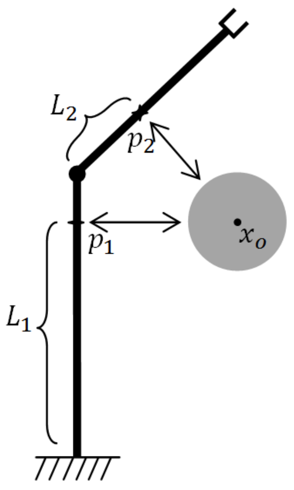


Figure 3.13: The closest points from each link of the robot to the obstacle.

d_1,d_2 between the robot links and obstacle,

$$\begin{aligned} d_1 &= a_1^T(x_o - p_1) - r \\ d_2 &= a_2^T(x_o - p_2) - r \end{aligned} \tag{3.54}$$

must remain greater than a minimum safety distance d_0. Then as before, the variation of distances d_1,d_2

$$\begin{aligned} \dot{d}_1 &= a_1^T(\dot{x}_o - \dot{p}_1),\ \ddot{d}_1 = a_1^T(\ddot{x}_o - \ddot{p}_1) + \frac{1}{d_1}\left(b_1^T(\dot{x}_o - \dot{p}_1)\right)^2 \\ \dot{d}_2 &= a_2^T(\dot{x}_o - \dot{p}_2),\ \ddot{d}_2 = a_2^T(\ddot{x}_o - \ddot{p}_2) + \frac{1}{d_2}\left(b_2^T(\dot{x}_o - \dot{p}_2)\right)^2 \end{aligned} \tag{3.55}$$

must satisfy the collision avoidance constraints

$$\begin{aligned} \ddot{d}_1 &\geq g(d_1 - d_0, \dot{d}_1, U) \\ \ddot{d}_2 &\geq g(d_2 - d_0, \dot{d}_2, U). \end{aligned} \tag{3.56}$$

However, p_1 and p_2 will change with time. Therefore the calculation of $\dot{p}_1$, $\dot{p}_2$, $\ddot{p}_1$, and $\ddot{p}_2$

	joint 1	joint 2
angle $[deg]$	[-105, 105]	[-150, 150]
velocity $[deg/s]$	[-322, 322]	[-600, 600]
acceleration $[deg/s^2]$	[-2000, 2000]	[-3000, 3000]

Table 3.1: Bounds on joint angles, velocities and accelerations.

is not straight forward because the rate of change in their position should be integrated.

$$p_1 = \begin{pmatrix} L_1\cos(q_1) \\ L_1\sin(q_1) \end{pmatrix} \tag{3.57}$$

$$\dot{p}_1 = \begin{pmatrix} -L_1\sin(q_1) & 0 \\ L_1\cos(q_1) & 0 \end{pmatrix}\dot{q} + \begin{pmatrix} \cos(q_1) \\ \sin(q_1) \end{pmatrix}\dot{L}_1 \tag{3.58}$$

$$= J_{p_1}\dot{q} + J^L_{p_1}\dot{L}_1 \tag{3.59}$$

$$\ddot{p}_1 = J_{p_1}\ddot{q} + J^L_{p_1}\ddot{L}_1 + \dot{J}_{p_1}\dot{q} + \dot{J}^L_{p_1}\dot{L}_1 \tag{3.60}$$

where L_1 illustrated in figure 3.13 and $\dot{L}_1$ is the rate of change of L_1 which can be approximated by collecting the previous values of L_1 ($\dot{L}_1 \approx \frac{L_1^{k-1}-L_1^k}{\Delta t}$). The same calculation can be done for $\dot{p}_2$, and $\ddot{p}_2$.

Finally, the optimization problem will be written in term of $\ddot{q}$

$$\begin{aligned} &\underset{\ddot{q}}{\text{minimize}}\ \|\ddot{q} - f'(q,\dot{q},U)\|^2 \\ &\text{subject to}\ a_1^T J_{p_1}\ddot{q} \geq g(d_1-d_0,U) - a_1^T(J^L_{p_1}\ddot{L} + \dot{J}_{p_1}\dot{q} + \dot{J}^L_{p_1}\dot{L}) \\ &\qquad a_2^T J_{p_2}\ddot{q} \geq g(d_2-d_0,U) - a_2^T(J^L_{p_2}\ddot{L} + \dot{J}_{p_2}\dot{q} + \dot{J}^L_{p_2}\dot{L}) \\ &\qquad |\ddot{q}_1| \leq U \\ &\qquad |\ddot{q}_2| \leq U, \end{aligned} \tag{3.61}$$

3.4.2 Experimental Results

In this section, we evaluate the performance of the proposed approach by testing it on real industrial applications. In the experiments presented here, in addition to bounding the joint accelerations, constraints on the joint angles and velocities were imposed (see Table 3.1).

The underlying optimization problems for *both* manipulators were solved on a PowerPC CPU of 400 MHz (16 ms control sampling time was used).

Avoiding obstacles: In this experiment the robot must avoid two obstacles, while performing a pick and place task. Figure 3.14 shows a snapshot from the experiment, while figure 3.15 depicts joint profiles. The end effector of the robotic arm reaches its goal while respecting joint velocity and acceleration constraints, and without hitting the obstacles.

Tracking a moving goal: In this experiment, the robot has to track a moving goal on a treadmill, while avoiding an obstacle. The strict priority given to obstacle avoidance can be seen here, where the tracking is automatically canceled in order to avoid the obstacle, and recovered as soon as possible. Figure 3.16 shows a snapshot from the experiment, while figure 3.16 depicts joint profiles.

Collaborative robots: In this experiment, two Adept Cobra s600 SCARA robots are performing pick and place operations while sharing the same working environment. Each manipulator has a dedicated controller and considers the other manipulator as a dynamic obstacle. Both robots share the same PowerPC CPU of 400 MHz. This poses a challenge to our proposed approach. Figure 3.18 shows how the robots are able to share the same workspace without collision, respecting their velocity and acceleration constraints through cooperation. In figure 3.19, a static obstacle is added in the middle.

In such industrial applications, the programmer should traditionally specify the intermediate trajectory for each robot by teaching a large amount of points, and must tune the maximum acceleration, deceleration and speed for each segment. This process needs a lot of experience and takes considerable time and effort which prevents many manufacturers from using multi-robot systems or considering applications with obstacles. By using our approach the user input for such application is simply the desired endpoints without the need to teach any intermediate points.

The experimental results demonstrate the effectiveness of the proposed approach. However, due to non linearity in manipulators, we found some real problems and limitations. Using the proposed trajectory generator could lead to invalid reaction to the obstacle. In figure 3.20, the robot reacts to the obstacle even though it is outside of the robot's workspace. Note how the control variable u_1 in figure 3.21(a) is reacting to the obstacle during the interval $[0.13, 0.35]$. Figure 3.21(b) depicts the status of the shortest distance between link

2 of the robot and the obstacle. As can be seen, the obstacle avoidance constraint (the red switching curve) leads to unnecessary reaction to the obstacle. In another example, in figure 3.22, the robot reacts lately to the obstacle, leading to high control action. Note how the values of the control variables u_1 and u_1 in figure 3.23 are twice bigger than their limitation during the interval $[0.1, 0.15]$. Some work around can be proposed to deal with such problems, but the motivation and need for a more comprehensive solution is clear.

3.5 Conclusion

In this chapter, we have introduced a real time reactive trajectory generator for mobile and manipulator robots. We proposed to begin with an analytical solution to the problem of time optimal control in the absence of obstacles, and to selectively limit the acceleration of the robot in the direction of obstacles, in a way that strictly enforces obstacle avoidance. The result is a control law which is time-optimal in the absence of obstacles, and sub-optimal in the presence of obstacles. The proposed method can handle multiple obstacles; however, they may lead to the appearance of undesired equilibrium points. These points are handled by introducing circular fields around the obstacles. The algorithm can tune the obstacle avoidance behavior to be activated earlier to copy with uncertainty in the obstacles state. Experimental results show that the desired behavior is indeed obtained: collisions with obstacles are strictly avoided, the algorithm runs in real time using limited computation power and the general motions of the robots still maintain some sub-optimality in the presence of obstacles. However, some problems and limitations remain due to nonlinearity in manipulators, and motivate us to start a new approach that will be presented in chapters 4 and 5.

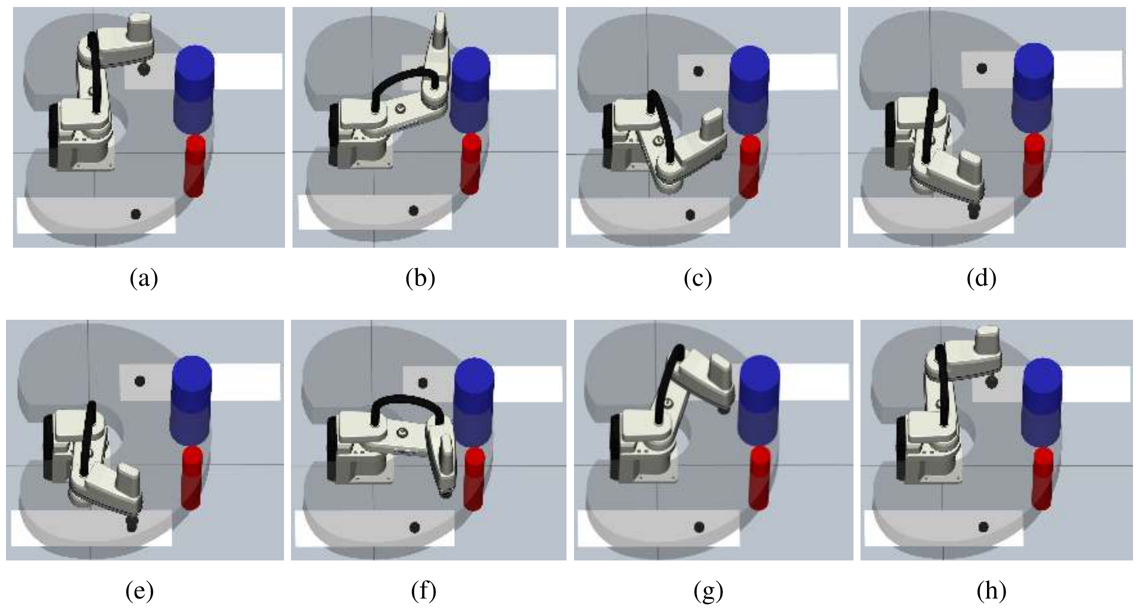


Figure 3.14: Snapshots from typical pick and place operations (a) $\rightarrow$ (b) $\rightarrow$ (c) $\rightarrow$ (d). Then another pick and place (e) $\rightarrow$ (f) $\rightarrow$ (g) $\rightarrow$ (h).

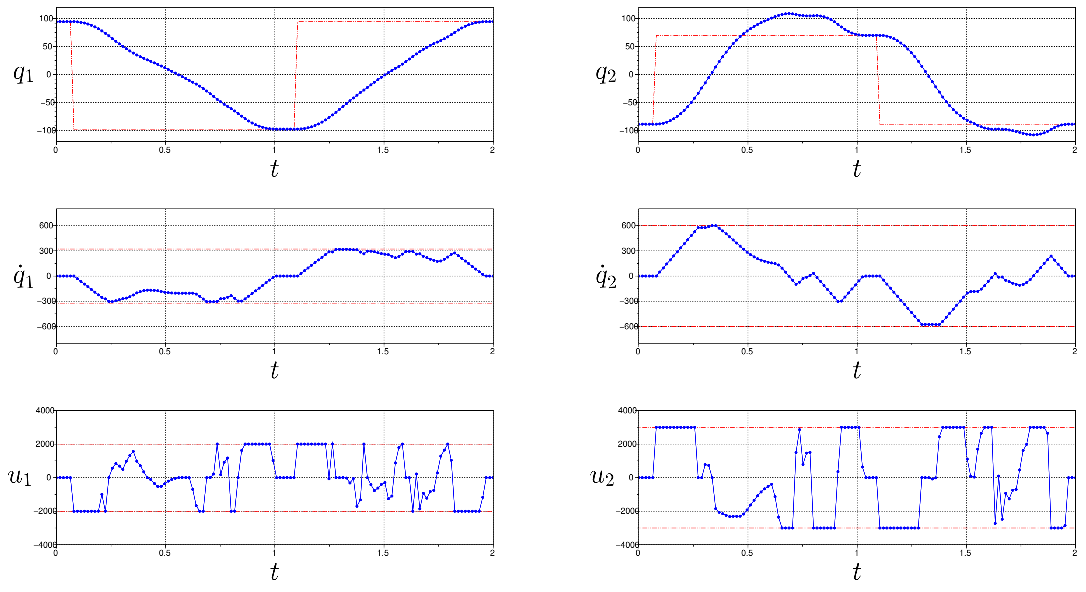
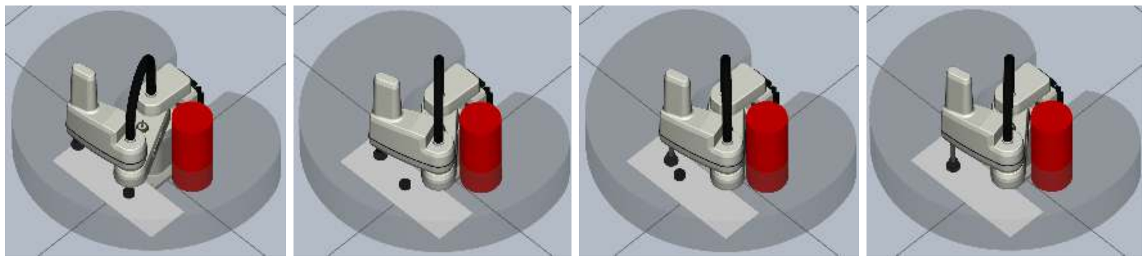


Figure 3.15: Joint profiles for the experiment in figure 3.14.



(a) (b) (c) (d)

Figure 3.16: Snapshots from the experiment, (a) $\rightarrow$ (b) $\rightarrow$ (c) $\rightarrow$ (d). Robot avoid hitting the obstacle and wait until it is able to track the moving goal.

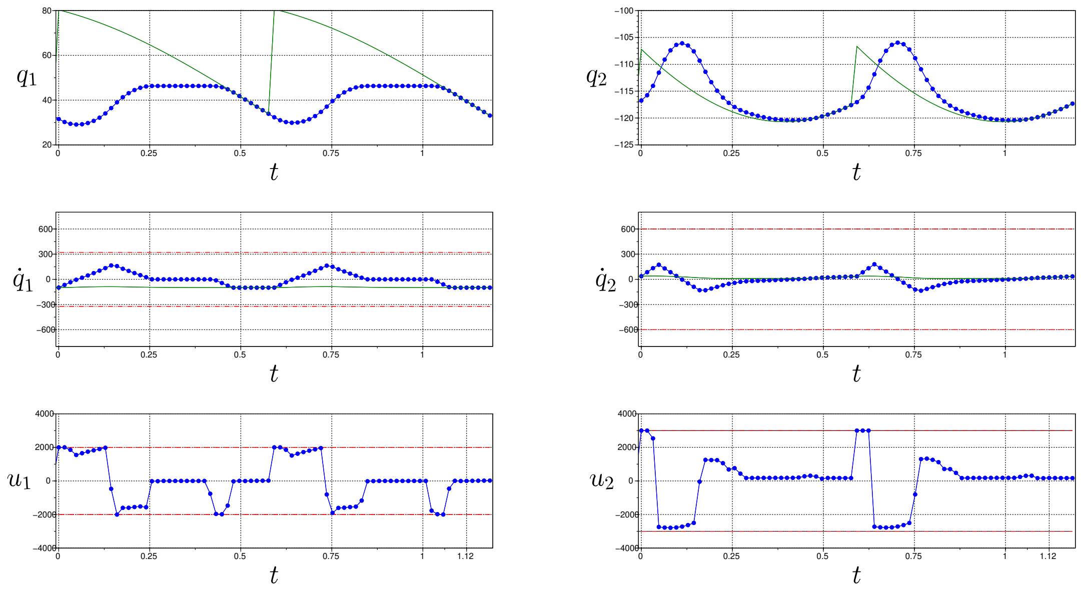


Figure 3.17: Joint profiles for the experiment in figure 3.16.

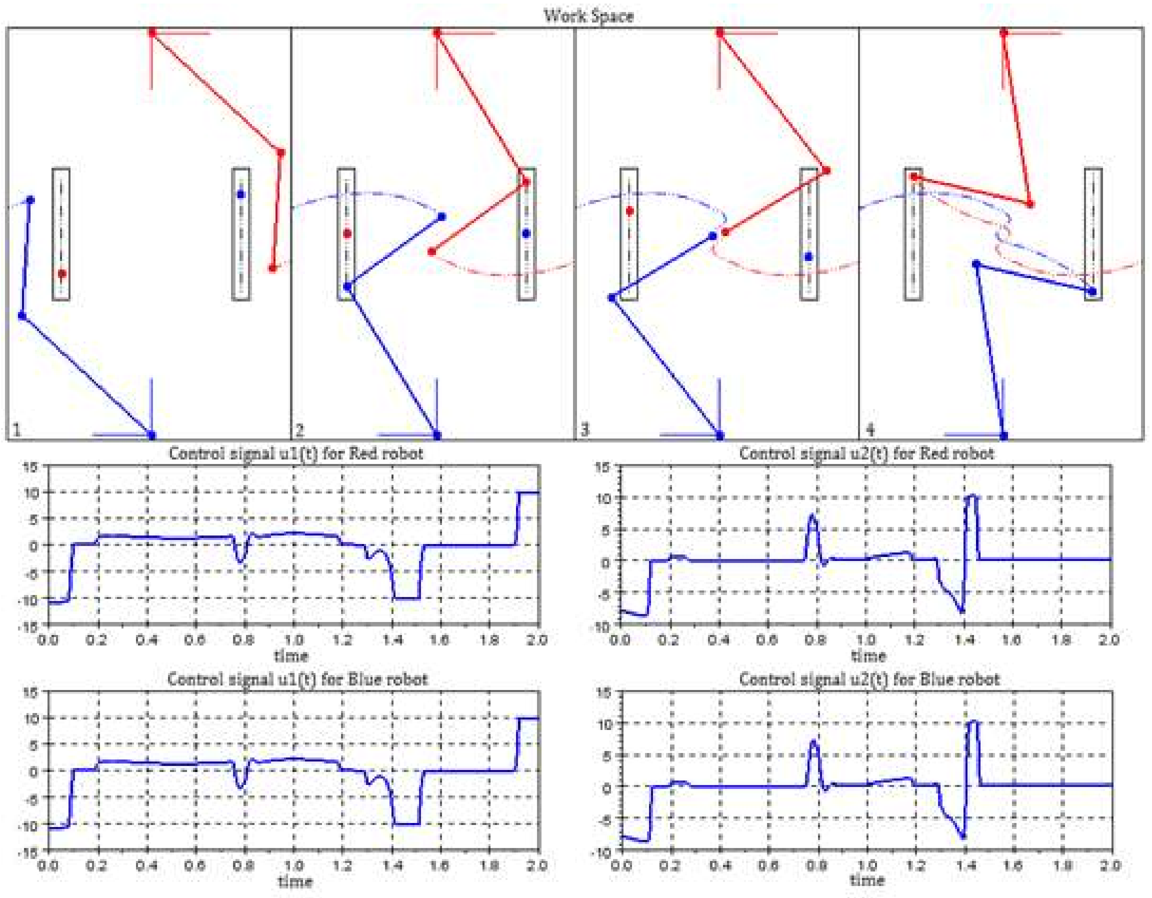


Figure 3.18: The red and blue robots should track respectively the small red and blue rectangles alternating between two treadmills. Control signals are given below.

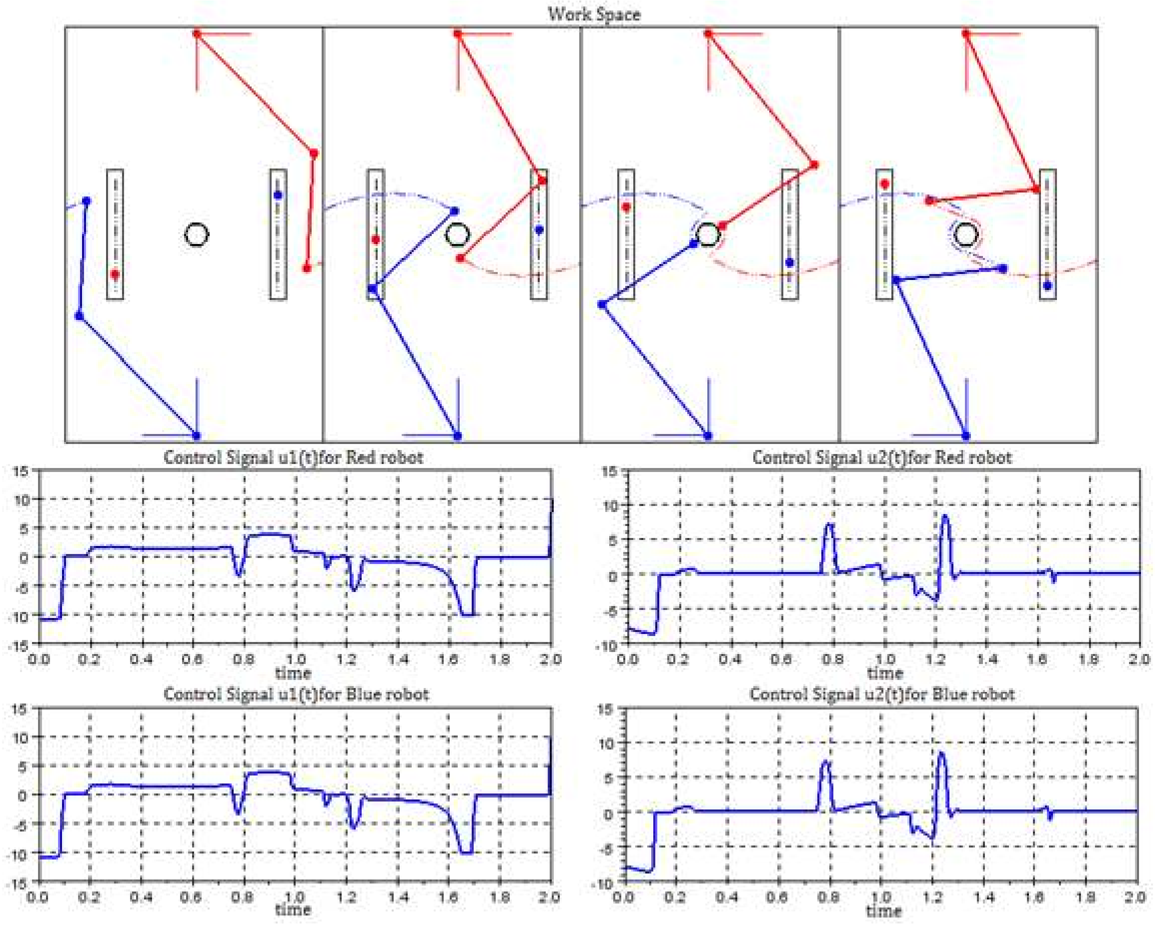
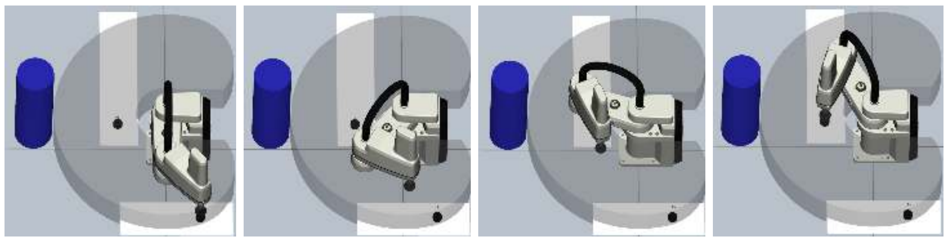
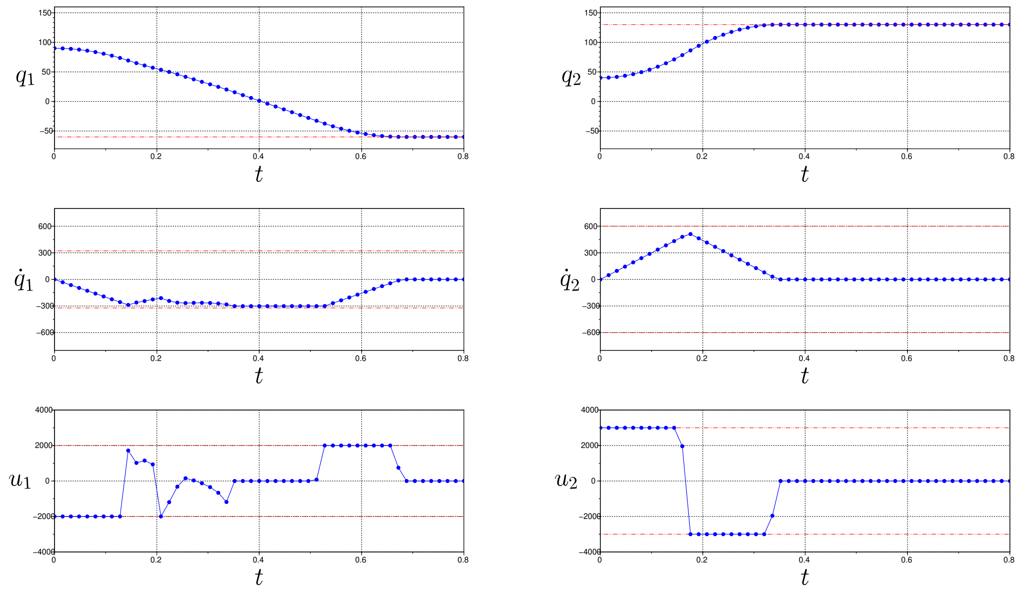


Figure 3.19: The red and blue robots should track respectively the small red and blue rectangles alternating between two treadmills. A grey static obstacle is added in the middle. Control signals are given below.

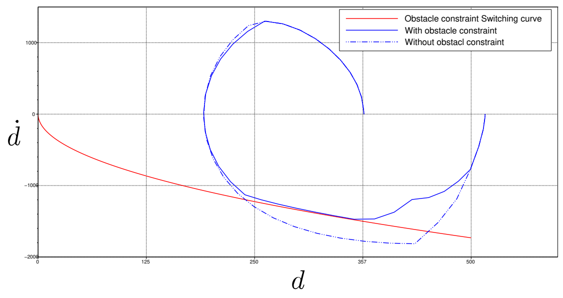


(a) (b) (c) (d)

Figure 3.20: Snapshots from typical pick and place operations (a) $\rightarrow$ (b) $\rightarrow$ (c) $\rightarrow$ (d).

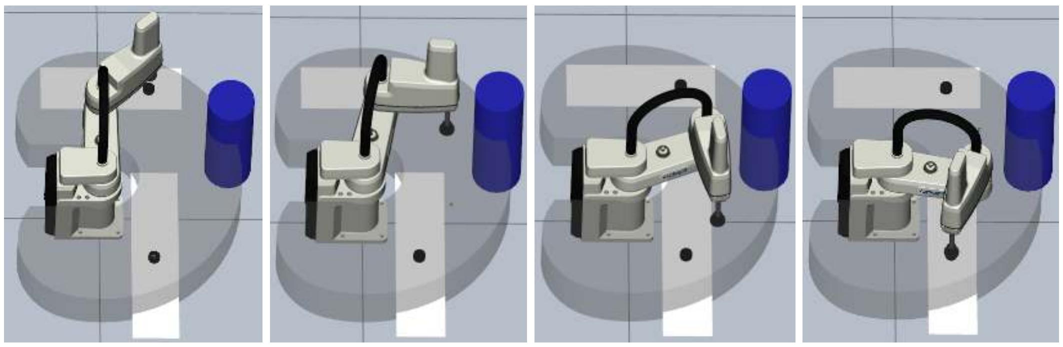


(a)



(b)

Figure 3.21: (a) Joint profiles and (b) the state space of the shortest distance between link 2 and the obstacle for the experiment in figure 3.20.



(a) (b) (c) (d)

Figure 3.22: Snapshots from typical pick and place operations (a) $\to$ (b) $\to$ (c) $\to$ (d).

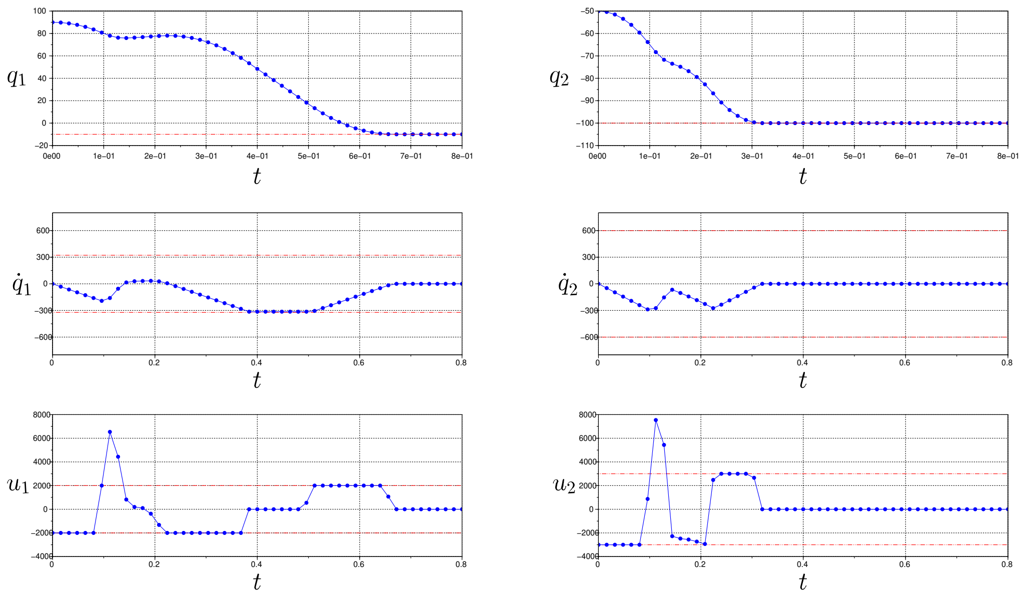


Figure 3.23: Joint profiles for the experiment in figure 3.22.

Chapter 4

An MPC Approach to Time Optimal Control

Finding the solution to the minimum time control problem for discrete time systems subject to linear constraints in the state and control variables is not a trivial problem. In special cases, an analytical solution can exist, but this is rarely the case. In this chapter we propose two different approaches that find numerically a general solution to this problem. The approach proposed in Section 4.3 will find the solution by numerically identifying the sparsest possible state vector sequence over a large enough control horizon by approximating the sparse optimization by a weighted ℓ_1-norm minimization. In Section 4.4 a novel approach to minimum-time control is presented based on a hierarchical optimization problem, which is standard in the field of robotics. This is advantageous as already existing tools can be used to approach its solution. Simulation results will demonstrate the effectiveness of the proposed approaches. Moreover, an online trajectory generator for real industrial manipulator will be designed in Chapter 5 base on the approaches proposed in this chapter.

4.1 Related work

Time optimal control refers to the problem of transferring the state of a dynamic system from a given initial state to a certain target state in minimum time. For continuous time linear dynamical systems the minimum-time problem has been demonstrated to be a bang-bang control policy (see Section 2.2.2), the implementation of such a switching control on microcomputers requires its discretization. This, however, may be challenging in practice because of high frequency chattering in the control signal [35]. As a result, it has been considered advantageous to develop minimum-time controllers directly for discrete-time plants [95].

However, analytical solutions are available only for special cases. In [35, 95] solution is derived for a standard double integrator. The work in [96, 36] designs a discrete minimum time control for a triple integrator system with bounded velocity, acceleration, and jerk. There is no simple and generic closed form characterization of the optimal control policy as in the continuous time case [15]. The motivation for more general solution to minimum time problem in discrete time system necessitates the use of numerical techniques.

We address the problem of driving the state of an arbitrary discrete-time linear dynamical system to the origin by numerically identifying the sparsest possible state vector sequence over a large enough control horizon. There are various ways to formulate a problem that identifies such a sequence. In [15], these sparse optimization problem is solved by minimizing a cost functional constructed as an appropriately weighted sum of ℓ_2-norms of the system states. In Section 4.3 we propose a similar approach however we approximate the sparse optimization with a weighted ℓ_1-norm minimization. We end solving a Linear program (LP) which is simpler and faster to solve compare to a weighted ℓ_2-norm minimization; however, in both cases we still rely on a selection of weighted factors.

In contrast to [15] and our proposition in Section 4.3 , we propose another approach in Section 4.4 that: (i) does not rely on the ad hoc selection of weighting factors (which is highly non-trivial), (ii) does not lead to any approximation and results in time-optimal behavior for arbitrary linear constraints (iii) and yet is tractable in real-time. Our formulation hinges on recent developments of efficient hierarchical solvers in the field of robotics [22, 26, 33] and can be integrated seamlessly in existing hierarchical control frameworks.

Apart form introducing our hierarchical formulation to minimum-time trajectory generation, we discuss practical issues related to its application. One such issue is the high frequency chattering in the control signal in the presence of noise when the setpoint has been reached [97]: a common drawback of aggressive control strategies. Following the ideas in [86] we formulate our controller in a way that leads to smooth behavior in the vicinity of the goal state.

This chapter is organized as follows. Section 4.2 reviews a classical approach to the minimum-time control problem. Section 4.3 introduces our ℓ_1-norm approach. Section 4.4 explains the hierarchical approach to the minimum time problem and discusses the effect of noise in the state estimates and how it can be improved. Section 4.4.2 includes a numerical comparison with an analytical solution in a simplified setting. Section 4.5 shows the theoretical and practical differences between the ℓ_1-norm approach and the hierarchical approach.

4.2 The Minimum-Time Problem

Let us consider a discrete time linear dynamic system

$$x_{k+1} = Ax_k + Bu_k, \tag{4.1}$$

where $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ are the state variables and control input, respectively. The system matrices $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$ could be arbitrary (however, we assume that the origin is reachable). In this chapter we consider a double integrator system as an example where

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \tag{4.2}$$

with sampling period Δt.

Transferring a given initial state $x^{(c)}$ (at discrete sampling time c) to the origin in minimal time can be achieved by solving

$$\begin{aligned} &\text{minimize } N \\ &\text{subject to } x_{k+1} = Ax_k + Bu_k \\ &\qquad x_0 = x^{(c)} \\ &\qquad x_N = 0 \\ &\qquad u_k \in \mathscr{U}_k \\ &\qquad g(x) \geq 0 \\ &\qquad N \in \{N^{\min}, \dots, N^{\max}\}, \end{aligned} \tag{4.3}$$

with $k = 0, \dots, N-1$ and $\mathscr{U}_k$ being a closed and bounded set containing zero in its interior (we assume it to be convex) [86]. The decision variables are $x_1, \dots, x_N$, $u_0, \dots, u_{N-1}$ and the number of discrete-sampling intervals is N. Note that, by design, we are not interested in reaching the origin faster than $N^{\min}$ sampling intervals in order to avoid aggressive behavior near the origin (as we will discuss in Section 4.4.4). $g(x) \geq 0$ includes collision avoidance constraints, which are in general nonconvex, as well as possibly other state related constraints , *e.g.,* position and velocity limits. We will use $N_c^\star$ to denote the value of N at the solution of (4.3) (the subscript emphasizes the dependence on $x^{(c)}$). Note that this is a mixed integer programming problem (due to the decision variable N). Solving such a problem can require high computation and memory resources, which make it not applicable to execute in real time on a limited resources system.

A computationally less demanding control strategy will be introduced in this chapter based on an equivalent reformulation of (4.3). Our proposed approaches are applied in a Model Predictive Control (MPC) setting (see Section 2.2.4) that defines an open loop optimal control problem and then iteratively solves it on-line, providing in the end a closed loop feedback controller with a finite prediction horizon N.

The prediction of the system behavior for N sampling times will be

$$x = Sx_0 + Tu$$

where the sequence of states $x = (x_1, \dots, x_N)$ and sequence of control inputs $u = (u_0, \dots, u_{N-1})$ are related through the matrices S and T, which can be obtained by iterating (4.1) for $k = 0, \dots, N-1$ [68].

$$S = \begin{bmatrix} A^1 \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \; T = \begin{bmatrix} A^0B & 0 & \dots & \dots & 0 \\ A^1B & A^0B & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \dots & \dots & A^0B \end{bmatrix}$$

where N is the length of the horizon.

4.3 An ℓ_1-norm Approach

In this section, we introduce a numerical approach to solve the minimum time control problem for discrete time linear systems by looking for the sparsest possible state vector sequence over a large enough control horizon. We approximate the sparse optimization by a weighted ℓ_1-norm minimization.

4.3.1 Formulation

We defined of time optimal control as transferring a given initial state $x_0 \in \mathbb{R}^2$ to the origin, in the smallest number of discrete time steps (N). An alternative point of view is to find a sequence of control inputs $u = (u_0, \dots, u_{N-1})$ such that the sequence of state $x = (x_1, \dots, x_N)$ has the most number of zeros. This can be achieved by solving in the following optimization problem

$$\begin{aligned} &\underset{s,\,u}{\text{minimize}} \;\; \textbf{card}\,(h(x)) \\ &\text{subject to } x = Sx_0 + Tu \\ &\qquad u \in \mathscr{U}, \end{aligned} \tag{4.4}$$

where $h(x) = (\|x_1\|_1, \ldots, \|x_N\|_1)$.[1]

Note that (4.4) is a very particular cardinality problem, in the sense that the zero components of h can appear only from the tail [15]. That is, if the origin can be reached in exactly $N^\star$ sampling times, we would have

$$h = (\|x_1\|_1, \ldots, \|x_{N^\star-1}\|_1, 0, \ldots, 0). \tag{4.5}$$

This structure suggests that the solution of a minimum-time control problem could be approached by solving a sequence of convex problems [86] (while using *e.g.*, bisection for identifying $N^\star$). However, we are interested in finding (4.5) with a single, properly chosen, convex problem (of course, we assume that $N^\star$ is not known).

A common alternative to solving the cardinality problem (4.4) is to use

$$\begin{aligned} \underset{x,\,u}{\text{minimize}}\ & \|Wx\|_1 \\ \text{subject to}\ & x = Sx_0 + Tu \\ & u \in \mathscr{U}, \end{aligned} \tag{4.6}$$

with a diagonal weighting positive-definite matrix W depending on the parameters [13]. Here, we choose to parametrize $W = \mathbf{diag}(w_1 + \beta w_2)$,

$$w_1 = (1, 0, 1, 0, \ldots, 0), \quad w_2 = (0, 1, 0, 1, \ldots, 1) \in \mathbb{R}^{2N},$$

with a positive scalar parameter β, which means that we are introducing a weight β to the velocity components of state $x = (x_1, \ldots, x_N)$.

For convenience, instead of (4.6), the ℓ_1-norm minimization problem can be cast as a

[1] We use the ℓ_1-norm here, but any norm could be used without impacting $\mathbf{card}(h)$, which gives the number of nonzero components of h.

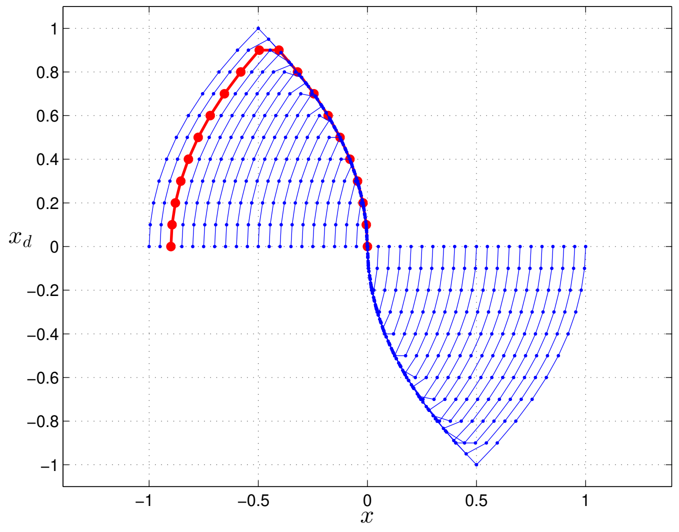


Figure 4.1: Optimal trajectories state space corresponding to the ℓ_1-norm optimization problem (4.7) where $\hat{N} = N = 20$ and $\beta = 0.4$, for a single double integrator system (4.2),(4.1) with sampling time $\Delta t = 0.1$ s

linear program (LP)

$$\begin{aligned} &\underset{u,\, r}{\text{minimize}}\ (w_1 + \beta w_2)^T r \\ &\qquad u \in \mathscr{U} \\ &\qquad -r \leq Sx_0 + Tu \leq r. \end{aligned} \tag{4.7}$$

where the vector r is part of the optimization variables [10].

We proved numerically that for a given Δt, $\hat{N}$ and $N \geq \hat{N}$, there are values of β, for which a solution of (4.7) solves problem (4.4) for any $x \in \mathscr{K}_{\hat{N}}$. The $\hat{N}$ step controllable set $\mathscr{K}_{\hat{N}}$ is the set of all states that can be driven to the origin in at most $\hat{N}$ sampling times (while satisfying the constraints).

4.3.2 Numerical results

In order to validate our approach, we test it on a double integrator system and compare the results with the analytical closed form solution proposed by [95].

Consider the system (4.1), (4.2) with sampling time $\Delta t = 0.1$ s, the goal is to determine

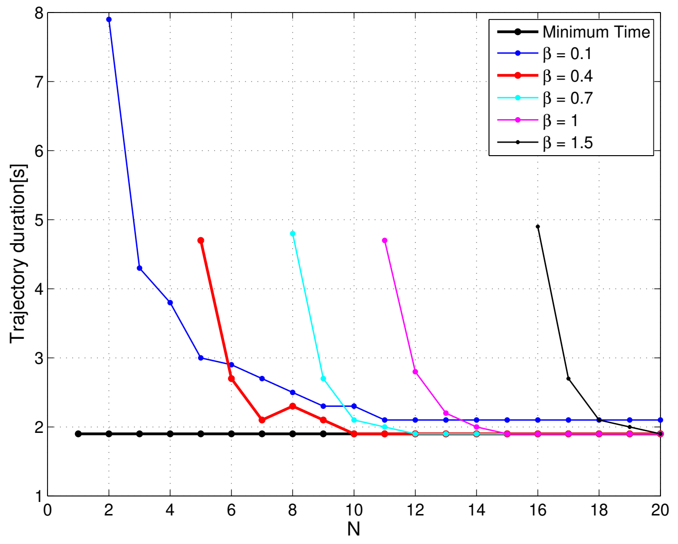


Figure 4.2: Duration of trajectory with red color in figure 4.1 for varying N with different values of β.

the control signal u that forces the system to zero in minimum time starting from different initial states and respecting the control constraint $|u| \leq 1$. For all trajectories, we have used $N = 20$ and $\beta = 0.4$ knowing that $\hat{N} = 20$. Figure 4.1 shows the state space plots for all trajectories. The minimum times calculated analytically by [95] are exactly the same we obtained by our approach. This confirms the capability of the proposed numerical solution to recover an exact time-optimal control sequence.

Figure 4.2 depicts the influence of N and β on the duration of the trajectory with red color in figure 4.1. As can be seen, time-optimality can not be achieve for some values of β e.g $\beta = 0.1$. For other values, a sufficient length of horizon $\widetilde{N}$ is required to achieve time-optimality. Due to the feedback effect introduced by applying the optimization problem (4.7) in an MPC setting, $\widetilde{N}$ can be smaller than $\hat{N}$ e.g $\beta = 0.4$, 0.7, or 1 [68]. However, choosing $\beta = 0.4$ is the best choice in this particular case where time optimality is achieved for value $\widetilde{N} = 10$, considerably smaller than $\hat{N}$.

In Section 4.5, another numerical test in a more complex scenario will be presented. Moreover, to further illustrate the applicability of this ℓ_1-norm approach, we will use it in Chapter 5 to design an online trajectory generation for an industrial manipulator performing pick and place operations in the presence of dynamic obstacles.

4.4 Hierarchy Approach

The approach introduced in this section is based on an equivalent reformulation of (4.3) as a hierarchical optimization problem: a standard multi-objective problem, where objectives can be assigned with different levels of priority. Hierarchical formulations are popular in robotics because they ensure that objectives with lower priority are optimized as far as they do not interfere with the optimization of objectives with higher priority [78, 49].

4.4.1 Formulation

Let us consider $N^{\max} \geq N_c^{\star}$, and define a sequence of states $x = (x_1, \dots, x_{N^{\max}})$ and control inputs $u = (u_0, \dots, u_{N^{\max}-1})$. We introduce the following hierarchical problem

$$\begin{aligned} \operatorname{lex}\underset{x,u}{\text{minimize}}\ & v = (\|x_{N^{\max}}\|^2, \dots, \|x_{N^{\min}}\|^2) \\ \text{subject to}\ & x_{k+1} = Ax_k + Bu_k \\ & x_0 = x^{(c)} \\ & u \in \mathscr{U} \\ & g(x) \geq 0, \end{aligned} \tag{4.8}$$

with $k = 0, \dots, N^{\max} - 1$ and $\mathscr{U}$ being a closed and bounded set containing zero in its interior. The "lex minimize" operator implies that the vector v is to be minimized according to a lexicographic order [41], that is, minimizing v_i (in a least-squares sense) is infinitely more important than minimizing v_j, for $i < j$. We will use $\mathbb{P}_c$ to refer to (4.8) when we want to emphasize the dependence on the initial state $x^{(c)}$.

The novelty of formulation (4.8) is in the particular choice of a lexicographic objective. It states that the most important thing, after satisfying the constraints, is to reach the origin in $N^{\max}$ number of sampling intervals. Then, if possible, try to reach the origin in $N^{\max} - 1$ sampling intervals, and so on until $N^{\min}$ intervals. This formulation ensures that each state $x_{N^{\min}}, \dots, x_{N^{\max}}$ would be as close as possible to the origin (in Euclidean norm), and once the origin has been reached, the states $x_{N^{\min}}, \dots, x_{N^{\max}}$ would remain there. Note that we have chosen the origin as the target state only for convenience. An arbitrary target state can

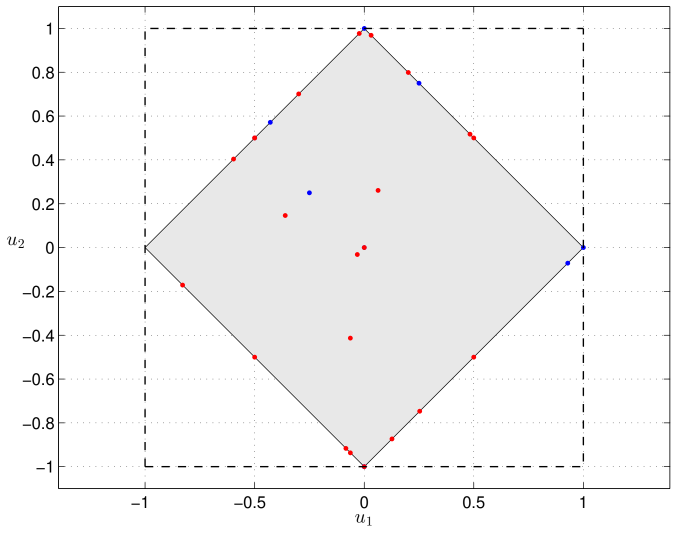


Figure 4.3: Coupling constraint (diamond) and box constraints with dashed line. Blue and red dots depict the control profiles from figure 4.5.

be used by a simple change of variable [46]. Furthermore, if necessary, target regions can be considered by using a similar formulation.

4.4.2 Comparison with the analytical solution for a double integrator

In special cases, the minimum-time problem for discrete-time linear dynamical systems subject to linear constraints has an analytical solution. One such case is when using a double integrator subject to simple bounds on the accelerations [35, 95]. Here, numerical results from our hierarchical formulation are compared to this analytical solution. The purpose of this comparison is not so much to demonstrate the equivalence (which should be apparent from the analysis in Section 4.4.1) but to emphasize the potential advantages of using numerical techniques for approaching the solution of the minimum-time problem.

Let us recall from Chapter 3 the analytical solution for the minimum-time problem for a two-dimensional double integrator (3.8) to (3.12), (3.26) and (3.25)

$$u_k = f'(x_k, \dot{x}_k, U).$$

It will ensure time optimal transition toward the goal for any given stat x_k, while taking into

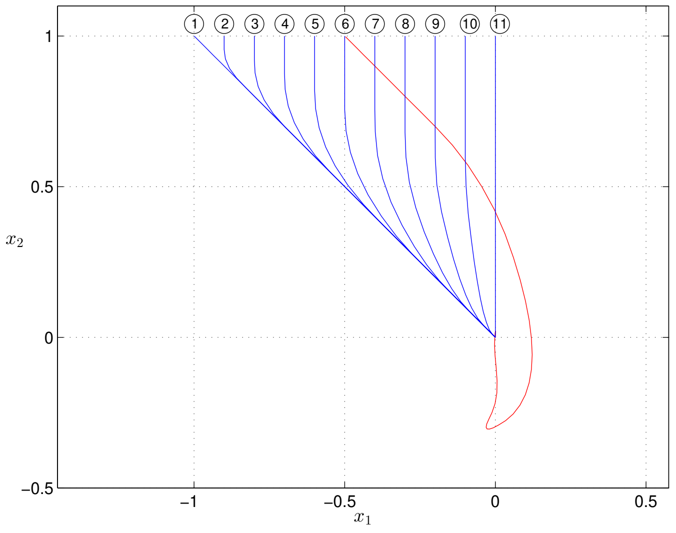


Figure 4.4: Eleven trajectories. Each trajectory starts with zero velocity and converge to the origin.

account the simple bounds on the controls $u \leq U$.

However, we expect to have additional linear constraints coupling the motion (*e.g.,* due to the linearization of the collision avoidance constraints we will see it in detail in Chapter 5). One possible option for still using the analytical solution would be to find $u_k^\star$ by solving

$$\begin{aligned} &\underset{u_k}{\text{minimize}} \ \left\| u_k - f'(x_k, \dot{x_k}, U) \right\|^2 \\ &\text{subject to } u_k \in \mathscr{U}_k. \end{aligned} \tag{4.9}$$

The motivation behind (4.9) is to stay as close as possible (in Euclidean norm) to $f'(x_k, \dot{x_k}, U)$ while respecting the additional constraints defined by $\mathscr{U}_k$. In order to evaluate the performance of (4.9) we compare it to (4.8) on a simple example, with a more restrictive constraint on u_k that couples the motion.

Figure 4.3 depicts this constraint as a gray diamond (contained in the box defined by the simple bounds). Figure 4.4 depicts in blue eleven minimum-time trajectories converging to the origin generated using (4.8). The effect of using heuristics (4.9) for the 6-th trajectory

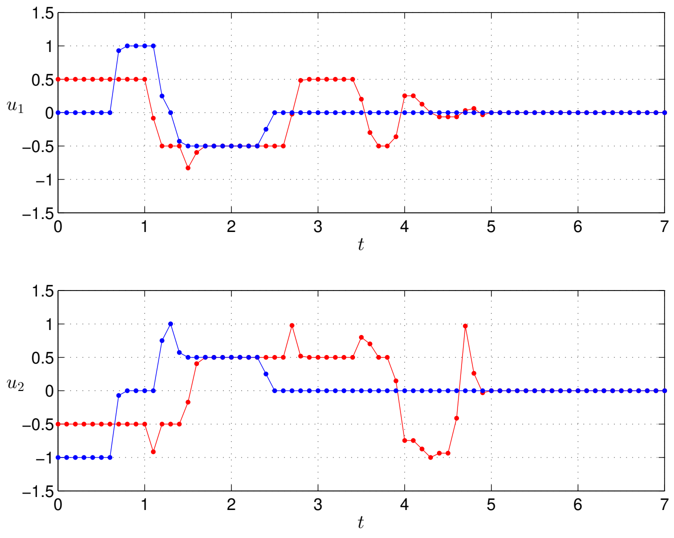


Figure 4.5: Evolution of the control inputs for trajectory number 6 in figure 4.4. Blue and red correspond to formulation (4.8) and heuristics (4.9), respectively.

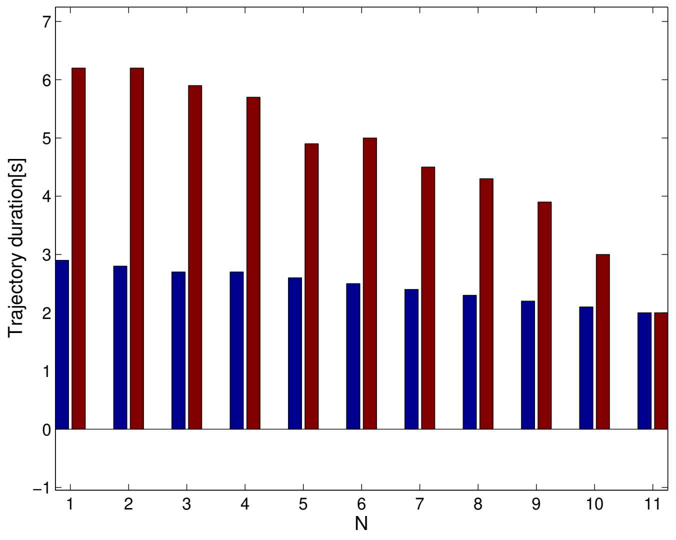


Figure 4.6: Duration of each of the eleven trajectories in figure 4.4. The minimal time as computed using (4.8) is depicted in blue, while the time required when using the heuristics (4.9) is depicted in red (a more than two times difference can be observed).

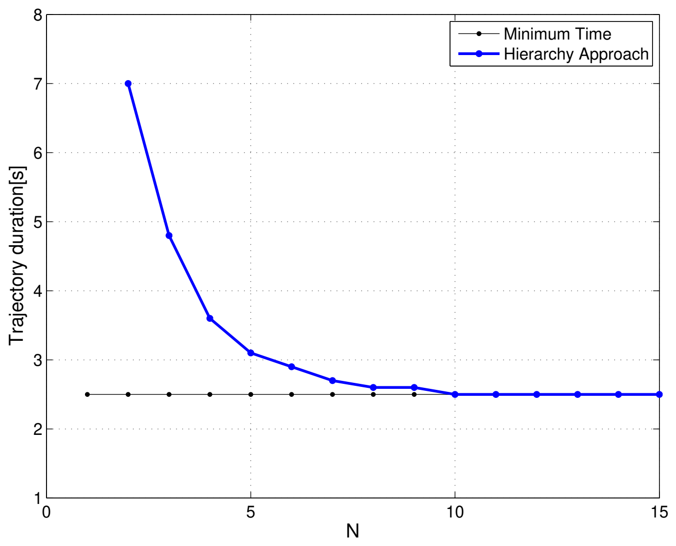


Figure 4.7: Duration of trajectory number 6 in figure 4.4 for varying $N^{\max}$. Similar pattern can be observed across all trajectories. Note that $N_6^\star = 25$, hence even though in theory time-optimality is guaranteed only for $N^{\max} \geq N_6^\star$, it appears that in practice satisfactory results can be obtained with much smaller $N^{\max}$.

can be seen in red (the corresponding control inputs are given in figure 4.5). For all trajectories we have used $N^{\min} = 1$ and $N^{\max} = 29 \geq N_i^\star$, $i = 1, \dots, 11$ (*e.g,* $N_6^\star = 25$), where i is the number of trajectory in figure 4.4, with a control sampling time $\Delta t = 0.1$ s. The duration of each trajectory is depicted in figure 4.6. As can be seen, using the heuristics (4.9) may result in more than twice slower transitions.

Based on these results we could conclude that even small modification of the constraints may render the analytical solution unsatisfactory. Since finding an analytical solution for arbitrary linear constraints is not straightforward it is beneficial to consider the numerical approach introduced here.

4.4.3 Choosing $N^{\max}$

Formulation (4.8) involves parameters $N^{\min}$ and $N^{\max}$ which should be specified by the user. The choice of $N^{\max}$ reflects the length of the preview horizon and thus can be used to influence the reactivity of the system. If it satisfies $N^{\max} \geq \hat{N} = \max(N_1^\star, N_2^\star, \dots)$, time-optimality would be guaranteed. Note, however, that $N^{\max}$ should not be chosen to be too large as it directly impacts the size of the problem to be solved.

Figure 4.7 depicts the influence of $N^{\max}$ on the duration of the trajectory number 6 from figure 4.4 for which $N_6^\star = 25$ with corresponding time of 2.5 s and a control sampling time $\Delta t = 0.1$ s. As can be seen, time-optimality is achieved even for values considerably smaller than $N_6^\star$, due to the feedback effect introduced by applying the optimization problem (4.8) in an MPC setting. Even $N^{\max} \in [7,8,9]$ appears to be acceptable, as the impact on the trajectory duration is rather small. We have observed that such behavior is very common even when additional state constraints are considered.

Lets call $\widetilde{N}$ the minimum value of N that satisfy time optimality for the system. A reasonable guess for $\widetilde{N}$ can be made based on the system setting (*e.g.,* by considering factors like sampling time, velocity and acceleration limits). If we take a double integrator system having acceleration limits $|u| \leq U_{max}$ and velocity limits $|v| \leq V_{max}$ as an example. When the state x is on the system limits, having maximum velocity, the system will need the maximum length of horizon t_{max} to be at rest at the desired set point.

$$t_{max} = \frac{V_{max}}{U_{max}} \tag{4.10}$$

$$\widetilde{N} > \left\lceil \frac{V_{max}}{\Delta t * U_{max}} \right\rceil \tag{4.11}$$

where $\lceil . \rceil$ is a ceiling function that maps a real number to the smallest following integer.

4.4.4 Choosing $N^{\min}$

The choice of $N^{\min}$ has an impact on the behavior of (4.8) in the vicinity of the setpoint when state measurement noise is present. On one hand, using $N^{\min} = 1$ results in a rather aggressive controller that always attempts at reaching the setpoint in one step. In the presence of noise this would result in high frequency chattering in the control signal. On the other hand, a too high value for $N^{\min}$ might have a significant impact on the time optimal behavior. Finding a proper trade-off has been considered as an important problem [97].

Note that when the setpoint can be reached in m sampling intervals, using $N^{\min} > m$ leads to redundancy (the solution of (4.8) is not unique) which can be exploited to optimize additional criteria (that can be used to formulate a desired trade-off). This can be achieved

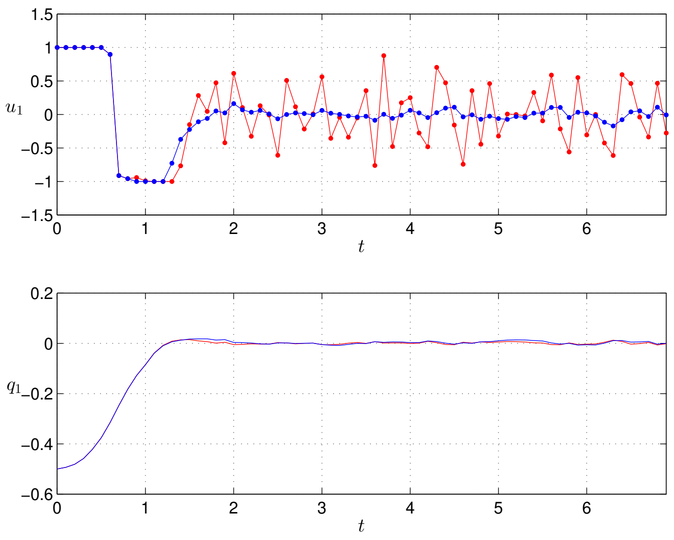


Figure 4.8: Test with trajectory 6 from figure 4.4 when the state measurement is corrupted by Gaussian noise with zero mean and standard deviation 0.005. The blue and red curves represent cases with $N^{\min} = 6$ and $N^{\min} = 1$, respectively ($N^{\max} = 29$).

by simply adding more hierarchical levels to (4.8).

Figure 4.8 depicts the influence of $N^{\min}$ on the trajectory number 6 from figure 4.4, when the state measurement is corrupted by Gaussian noise with zero mean and standard deviation 0.005. The objective of (4.8) is modified to

$$\operatorname{lex\,minimize}\ (\|x_{N^{\max}}\|^2, \ldots, \|x_{N^{\min}}\|^2, \|u\|^2),$$

i.e., an additional optimization criterion is introduced.

The blue and red curves represent cases with $N^{\min} = 6$ and $N^{\min} = 1$, respectively. Figure 4.8 illustrates the profile of the control input of joint 1. As can be seen, the minimization of $\|u\|^2$ has a filtering effect on the high frequency chattering (which is desirable in practice). The lower plot depicts the resultant profiles of the angle of joint 1: they are hardly distinguishable. This implies that a proper choice of $N^{\min}$ can have a smoothing effect on the control profiles without degrading the time-optimal behavior significantly.

In summary, the parameters $N^{\min}$ and $N^{\max}$ can be used to achieve a trade-off between time-optimality, problem size and smoothness of the solution (in the vicinity of the set-point).

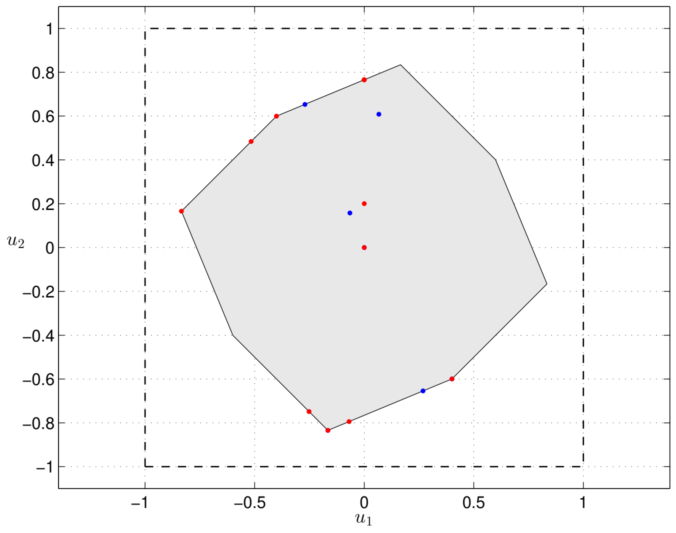


Figure 4.9: Coupling constraint (polygon) and box constraints with dashed line. Blue and red dots depict the control profiles from figure 4.12.

4.5 ℓ_1-norm vs Hierarchy Approach

In this section we present a comparison between the ℓ_1-norm and the hierarchical approach. Figure 4.9 shows the constraints applied on the control variable u as a gray polygon. As can be seen it is not a simple constraint so there is no straightforward analytical solution. The duration of each trajectory is illustrated in figure 4.11.

Figure 4.10 depicts in blue and red, the trajectories generated using (4.8) and using (4.7) respectively. Note that for the same initial and final states two totally different minimum time trajectories generated. The time optimal solution is not unique, there are alternative sequences of control inputs that lead to time optimal state transitions (see figure 4.12).

The reason behind the differences in the generated trajectories between the two approaches is that, the sequences of control generated by ℓ_1 norm will lead also to states that have the minimum ℓ_1 norm distances to the goal, while using hierarchical approach will lead to states that have the minimum ℓ_2 norm distances to the goal. Each approach has another criterion to satisfy in addition to time optimality. For example, if we want to generate a minimum time trajectories that have a minimum energy, we will get different trajectories. None of these approaches can be used in this case (see figure 4.10), we should reformulate

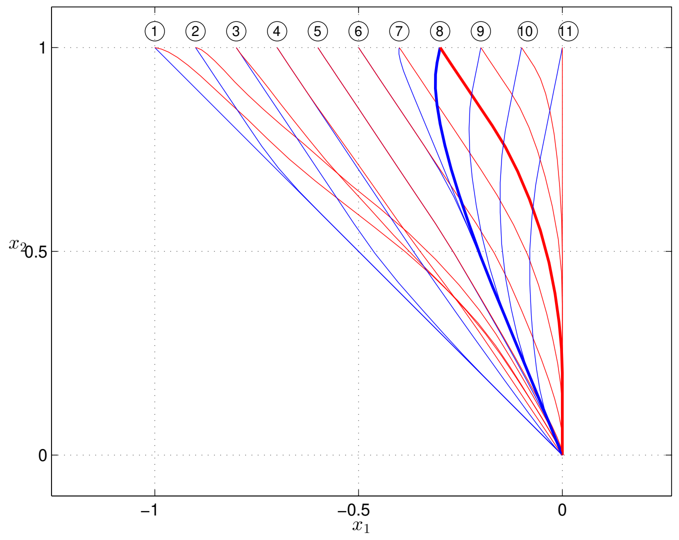


Figure 4.10: Eleven trajectories. Each trajectory starts with zero joint velocity and converge to the origin. Trajectories generated by hierarchical approach is depicted in blue, while the ones generated by ℓ_1-norm approach is depicted in red.

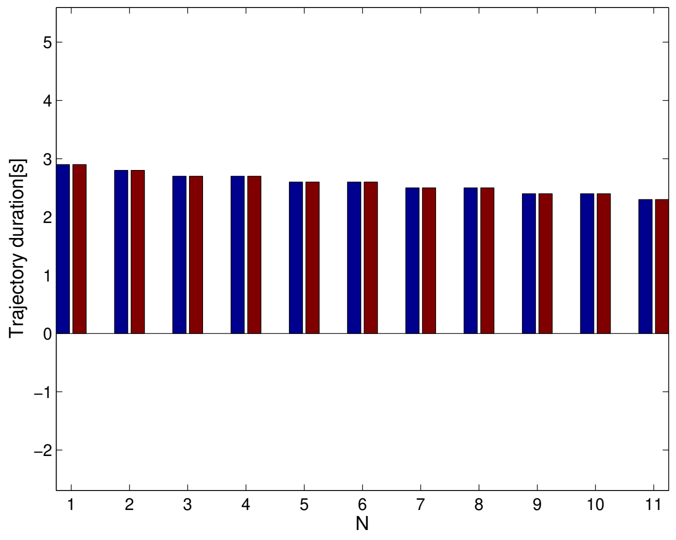


Figure 4.11: Duration of each of the eleven trajectories in figure 4.10. The minimal time as computed using hierarchical approach (4.8) is depicted in blue, while the time required when using ℓ_1-norm approach(4.7) is depicted in red.

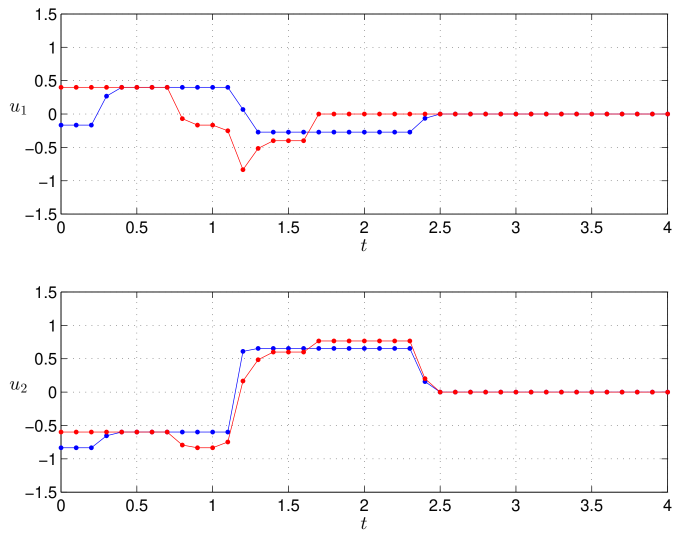


Figure 4.12: Evolution of the control inputs for trajectory number 8 in figure 4.10. Blue and red correspond to hierarchical formulation (4.8) and ℓ_1-norm formulation (4.7), respectively.

the problem such that we introduce a minimum energy criterion instead of minimum ℓ_1 or ℓ_2 norm criterion.

Figure 4.13 depicts the influence of the horizon length N on the duration of the trajectory number 8 in figure 4.10. Note that ℓ_1-norm approach required longer horizon ($N = 16$) than hierarchical approach ($N = 9$) in order to achieve time-optimality. Moreover, hierarchical approach does not rely on the ad-hoc selection of weighting factors (β parameter), while an appropriate value of β should be chosen to make sure that ℓ_1-norm optimization (4.7) generates time optimal trajectory.

Based on these results, we conclude that using hierarchical approach is preferable over ℓ_1-norm approach. To further illustrate and evaluate the performance of the presented methods, an online trajectory generation for an industrial manipulator performing pick and place operations in the presence of dynamic obstacles based on them will be presented in Chapter 5.

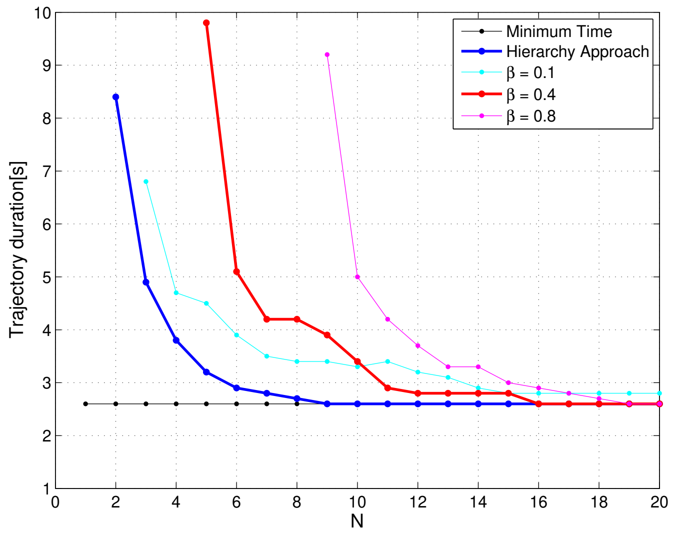


Figure 4.13: Duration of trajectory number 8 in figure 4.10 for varying N and different values of β.

4.6 Conclusion

In this chapter, we have introduced two different numerical approaches to find the minimum time control for a discrete time system. The first one formulates the problem as a weighted ℓ_1-norm minimization problem, as an approximation to a cardinality problem which is an alternative definition of time optimality. The second approach is based on hierarchical optimization problem which is standard in the field of robotics. A comparison with the analytical solution for a double integrator system shows the effectiveness of the proposed approaches. They give a time-optimal behavior for arbitrary linear constraints where finding analytical solution to such arbitrary linear constraints is not straightforward. Finally we did a comparison between the ℓ_1-norm and the hierarchical approaches. the hierarchical approach is preferable since it doesn't rely on an ad-hoc selection of weighting factors, and in addition, time-optimality is achieved with a shorter horizon length.

Chapter 5

Implementation and Validation in the Presence of Obstacles

In this chapter, we validate the numerical formulation introduced in Chapter 4 by applying them to online generation of trajectories for industrial robots performing pick and place operations in the presence of obstacles. Model predictive control is used in order to achieve a reactive behavior and to obtain accurate local approximations of the collision avoidance constraints (which are nonlinear and nonconvex). An experiment using two SCARA robots that share the same working environment is used to evaluate the proposed approach.

5.1 Introduction

This chapter addresses the problem of online trajectory generation for an industrial manipulator performing pick and place operations in the presence of dynamic obstacles. Since in many mechatronic applications the control input cost is less important than the task execution time [86], we focus on fast transitions by attempting to achieve time-optimality. The user input for the proposed scheme is simply the desired endpoints without the need to specify an intermediate trajectory. This can simplify greatly the deployment of industrial technology, leading to decreased cost and thus may have impact on various industrial applications [3]. Accounting for the full-body dynamics when generating this intermediate trajectory is usually not essential as most industrial robots are position controlled. That is why we model the evolution of the joint positions and velocities of the manipulator using a discrete-time linear dynamical system while accounting for input and state constraints.

Since the collision avoidance constraints are in general nonlinear (and nonconvex), we employ a Sequential Quadratic Programming (SQP) approach [64] where a sequence of linearized sub-problems is solved (one for each control sampling time). Each sub-problem identifies a minimum-time trajectory from the current state of the robot with respect to local linear approximations of the collision avoidance constraints. While such a sequence of problems is not guaranteed to converge to a time-optimal solution for the original nonconvex problem, it provides a practical way of generating locally optimal solutions, which is sufficient for most applications [50, 51]. Our approach is applied in a Model Predictive Control (MPC) setting, which not only improves reactivity of the system but presents a possibility to obtain accurate local linear approximations of the collision avoidance constraints.

Time-optimal trajectories for the above mentioned SQP sub-problems will be guaranteed by applying the hierarchical approach presented in Chapter 4. This formulation generates a time optimal behavior for arbitrary linear constraints and is tractable in real-time. To solve efficiently the hierarchical problem, we use a recent solver [22] base on a novel matrix factorization called "lexicographic QR" or $\ell - QR$ in short.

We present an experimental evaluation of the proposed approach using a typical industrial setup where two manipulators share the same working environment (see Fig. 5.1). Each manipulator has its own controller and considers the other manipulator as a potential obstacle. This is a problem of practical interest and presents a very good test bed for our approach due to the limited computational resources (the underlying optimization problems for both manipulators are solved on a *single* CPU at 400 MHz).

The rest of this chapter is structured as follows: Section 5.2 formalizes the nonlinear collision avoidance constraints and introduces the importance of continuous collision detection algorithms. Section 5.3 poses priorities between robot tasks. Section 5.4 includes the tools used to decrease the computation time needed to solve the proposed approach. Section 5.5 shows the experimental evaluation, and Section 5.6 presents a comparison with the previous approach presented in Chapter 3.

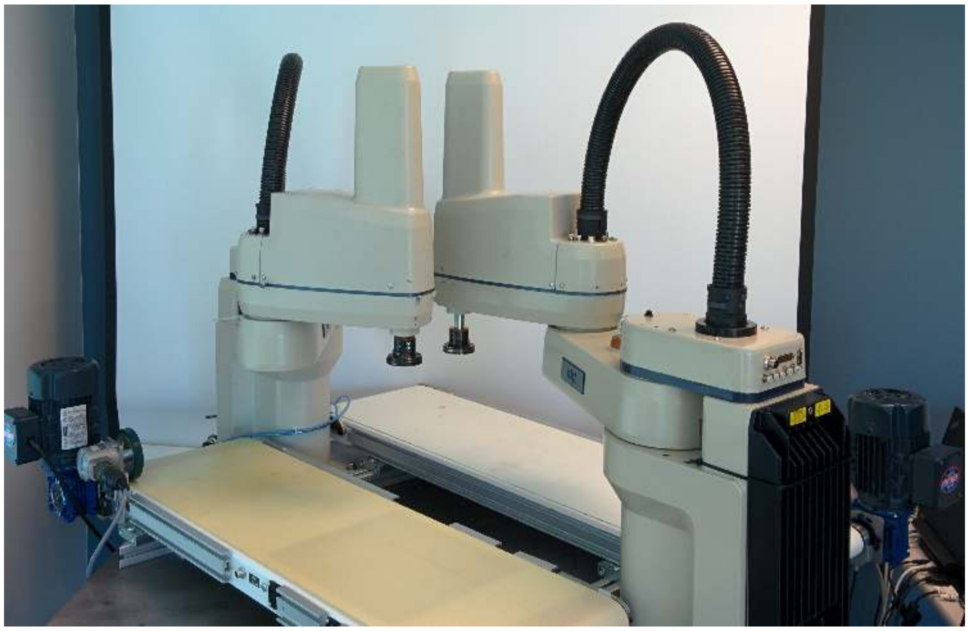


Figure 5.1: Experimental setup. Two Adept Cobra SCARA [1] robots sharing the same working environment.

5.2 Collision avoidance constraints

The work introduced in this chapter is based on the ℓ_1 norm optimization problem (4.7) and the hierarchical optimization problem (4.8) presented in Chapter 4; however, in this case collision avoidance constraints, which are in general nonlinear and nonconvex will be added to the problem. As a result, (4.8) and (4.7) will be nonlinear optimization problems. We approach the solution of this nonconvex optimization problem by adopting an SQP scheme in an MPC context. That is, problems (4.8) and (4.7) with linearized collision avoidance constraints are solved during each control sampling interval c and only $u^{(c)} = u_0^\star$ is applied to propagate the state from $x^{(c)}$ to $x^{(c+1)}$. More details on the linearization are provided in the following section. Each sub-problem can be integrated seamlessly in existing control frameworks in robotics. As we did in Chapter 4, we will use $\mathbb{P}_c$ to refer to the optimization problem ((4.8) or (4.7)) when we want to emphasize the dependence on the initial state $x^{(c)}$.

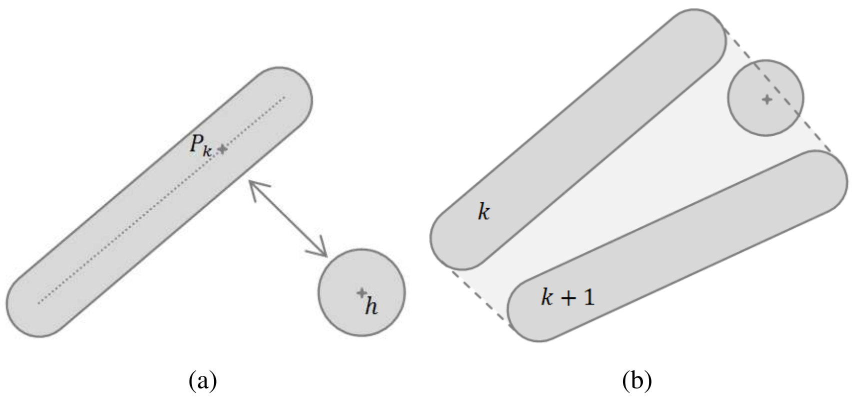


Figure 5.2: (a) The closest points between a given link of the manipulator and a static circular obstacle. (b) The convex hull of a given link at k-th sampling interval and at $k+1$-th sampling interval of the preview.

5.2.1 Formulation

Collision avoidance constraints $g(x) \geq 0$ can be defined in terms of various primitive shapes [81, 32]. We consider a standard model that approximates the shape of the robot and the obstacles using a composition of spheres and swept sphere lines [81]. Due to the nature of the envisioned application, the collision avoidance constraints are dynamically changing *i.e.,* not known in advance, and are moreover nonconvex. The MPC scheme that we have adopted here can be used to address both issues. Not only it increases the reactivity of the controller but also it can be used to develop accurate local linear approximations of $g(x) \geq 0$.

For clarity, first we consider collision avoidance constraints between a given link of the manipulator and a static circular obstacle (see figure 5.2(a)). Suppose that the obstacle is centered at position $h \in \mathbb{R}^3$. Let $p_k^{(c)}$ be the point on the link that is closest to the obstacle during the k-th sampling interval of the preview associated with $\mathbb{P}_c$. Then, in order to avoid collision, the Euclidean distance between $p_k^{(c)}$ and h:

$$d_k^{(c)} = \left(a_k^{(c)}\right)^T \left(p_k^{(c)} - h\right), \quad a_k^{(c)} = \frac{p_k^{(c)} - h}{\left\|p_k^{(c)} - h\right\|}, \tag{5.1}$$

must remain greater than a minimal safety distance d_s:

$$d_k^{(c)} \geq d_s. \tag{5.2}$$

This is a nonconvex constraint and accounting for it explicitly can be computationally costly. That is why, we approximate it by observing that $\mathbb{P}_c$ is closely related to $\mathbb{P}_{c-1}$. This fact is heavily used in the field of predictive control not only to formulate simple and expressive constraints but to warm-start each optimization process with an adequate initial guess [91]. Following the exposition in [75], we use an approximation:

$$a_k^{(c)} \approx \frac{p_{k-1}^{(c-1)} - h}{\left\| p_{k-1}^{(c-1)} - h \right\|}, \quad p_k^{(c)} \approx p_{k-1}^{(c-1)} + J_{k-1}^{(c-1)} \dot{q}_k^{(c)}, \tag{5.3}$$

where $J_{k-1}^{(c-1)}$ is the Jacobian matrix associated with $p_{k-1}^{(c-1)}$. This way, constraint (5.2) can be approximated using

$$\left(a_k^{(c)}\right)^T \left(J_{k-1}^{(c-1)} \dot{q}_k^{(c)} + p_{k-1}^{(c-1)}\right) \geq d_s, \tag{5.4}$$

which is linear in $\dot{q}_k^{(c)}$ (a part of the decision variables of $\mathbb{P}_c$). Alternatively one can use

$$\dot{q}_k^{(c)} = \frac{q_k^{(c)} - q_{k-1}^{(c-1)}}{\Delta t}, \tag{5.5}$$

with Δt being the sampling time, to reformulate (5.4) in terms of $q_k^{(c)}$. Approximating $g(x) \geq 0$ by using linear constraints like (5.4) for each link of the manipulator for $k = 1, \ldots, N^{\max}$, renders problem (4.8) with only linear constraints and a lexicographic least-squares objective, which is a class of problems commonly solved in robotics.

The only modification needed in case of a dynamic obstacle, assuming that its position over the preview horizon is known, is that one has to consider a time-varying h in the above derivations. Using other primitive shapes instead of a sphere to model obstacles is readily possible (this would only alter how the closest point is computed [24]).

5.2.2 Continuous collision avoidance

The previous section describes how to formulate a collision avoidance constraint that makes sure the robot will not collide with obstacles at each time step; however, collisions between these time steps may still happen. Figure 5.2(b) illustrates that.

To ensure collision avoidance in all cases, we should reformulate the obstacle avoidance constraint such that it makes sure that the robot will avoid hitting the obstacle continuously even between the specified time steps. In [69, 70, 98], given two discrete configurations of the links of an articulated model, an "arbitrary in-between motion" is used to interpolate its motion between two successive time steps and check the resulting trajectory for collisions. In [54, 74], multi-vehicle formation control with collision avoidance based on model predictive control (MPC) is proposed. Since MPC based methods can only consider collision avoidance at discrete time steps, a collision may occur in intervals between prediction time steps. As a result, they propose a method based on mixed-integer programming, where transition constraints are imposed so that they ensure collision avoidance between the prediction time steps. In order to apply such approaches, high computation power is needed which make them not applicable for an algorithm running in real time and using limited computation and memory resources. In [75], the swept volume for the links of the robot between successive time steps is calculated and then a collision avoidance constraint on the distance between the swept volume and the obstacle is applied. In this approach, a good approximation for calculating the swept volume is suggested, which fits our situation: The convex hull of the initial and final volumes [84] (see figure 5.2(b)). Then the same formulation as in the previous section can be applied.

This approach can double the computation time [75] in itself, but it allows using a smaller number of sampling times what reduces the total computation cost. However in our case the thickness of the manipulator and the small size of the time step will make sure that no collision will occur in the intervals between the prediction time steps. Therefore, there is no need to apply this in our problem at least for the near future.

5.3 Task prioritization

The robot has to reach a target in minimum time, respect robot limitations (*i.e.,* joint angles, velocities, and acceleration constraints.), and safely avoid obstacles. It have several goals to satisfy. However, there might not exist any control action u that satisfies all these tasks due to conflicts among them. Let us get inspiration from the famous three laws of robtics introduced by Isaac Asimov:

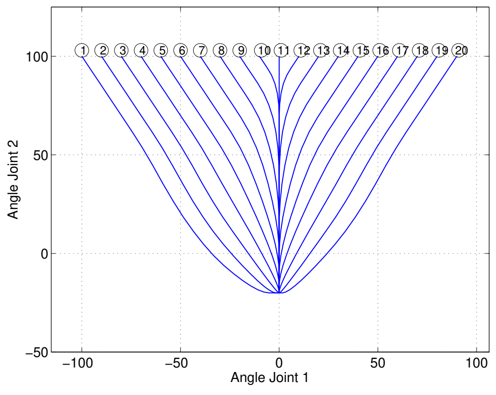
1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the 1^{st} law.

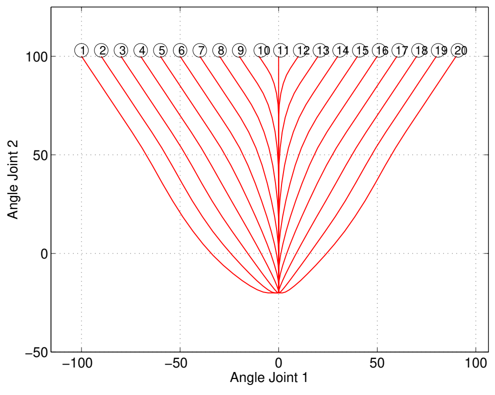
3. A robot must protect its own existence as long as such protection does not conflict with the 1^{st} or 2^{nd} law.

These laws are stated in a way that clearly imposes a strict hierarchy of importance between different goals. The safety of humans (collision avoidance constraints) is considered to be strictly more important than the safety of the robot (joint angles, velocities, and acceleration constraints), and to reach the target in minimum time may be attempted only after these safety concerns have been taken care of.

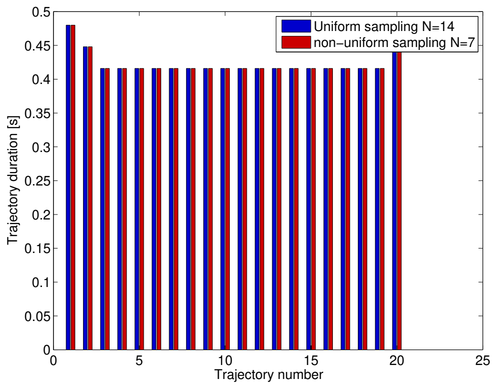
One way to give priority between tasks is to apply a weighted least square [53] and give different weights for tasks, to have a strict priority between them, appropriate weights should be chosen. However, It is not easy and not achievable in many problems [62]. In [44], a strict hierarchy between tasks is achieved by solving a sequence of constrained least squares, where lower priority objectives are optimized as far as they do not interfere with the higher priority objective. A more efficient approach is suggested by [22, 26], where the solver considers all the priority levels at the same time. It solves the complete hierarchical problem at once, avoiding the unnecessary iteration encountered in solving a sequence of QP. This latest approach will be used in the real time implementation for industrial robots performing pick and place operations in the presence of obstacles.



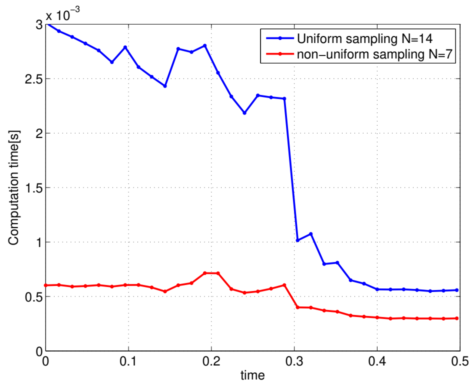
(a)



(b)



(c)



(d)

Figure 5.3: Twenty trajectories. (a) Uniform sampling of the predicted horizon with $N = 14$. (b) Non-uniform sampling of the predicted horizon $N = 7$. (c) Duration of each of the twenty trajectories. (d) Solver computation time at each iteration.

5.4 Real-Time Implementation

This section discusses various aspects used in the implementation to have an efficient algorithm in terms of computation time. The basic element of this implementation is the hierarchical solver. First, a brief description about the multi objective active set method [22, 26] used in the solver will be given. Second, we highlight the capacity to warm start the solver. Finally, we discuss the ability to change the way the predicted horizon in MPC is parametrized and how it will effect the computation time.

5.4.1 Multi objective active set method

The proposed method by [22, 26] provide a solver that considers all the priority levels at the same time. It is based on a novel matrix factorization called lexicographic QR decomposition (ℓ-QR) [22] that is cheaply updatable and faster to compute than the hierarchical complete orthogonal decomposition (HCOD) factorization proposed in [26] while . The proposed algorithm is derived from the classical primal active set method for QPs. It is composed of two parts: the first part incrementally builds an active set such that all the constraints in all levels are activated or satisfied. The second part searches for the corresponding optimal active set by deactivating unnecessary constraints.

Using this method, the hierarchical optimization problem appears as a single problem, where the active sets of all hierarchical levels are computed at the same time, avoiding the unnecessary iterations encountered in [44], where a sequence of QPs is proposed to solve such a problem. As a result, the computation time decreases and it is possible to use a warm start in robot control problems, so fewer iterations are needed to converge to the solution.

5.4.2 Warm-starting in optimization

In robot control, the solver is used at each sampling time. The problem definition doesn't vary too much from one control cycle to another. The optimum active set for the optimization problem at time t is nearly the same at time $t+\Delta t$. Thus, the optimum active set of time t can be used as an initial guess for the active set of time $t+\Delta t$. This is known as warm-starting in optimization. The warm-start improves the efficiency of the solver, the solution for the new cycle can be obtained very quickly, where only a few iterations of the solver are required to get the optimum active set.

The results show a significant reduction in the computation time using warm start. Figure 5.13 depicts the computation time with and without warm start. As can be seen, warm-starting in optimization helps in decreasing the computatoin time.

5.4.3 Control parametrization

The size of the optimization problem is related to the length of the predicted horizon N. For instance, in the current implementation, the size of the optimization problem is: $2*N$ decision variables, $4*N$ equality constraints and $12*N+k*N$ inequality constraints where k is the number of obstacles in the work space (see Table 5.1). Control parametrization is a technique that enables a drastic reduction of the size of the the optimization problem that underlines the MPC design without noticeable reduction in the control performance [2]. Instead of having a uniform sampling of the predicted horizon, we can use a non-uniform one, while preserving the same control performance.

Figure 5.3 depicts the different in computation time between using a uniform sampling where $N = 14$, and a non-uniform sampling where $N = 7$. As can be seen, both of them give the same control performance: trajectory travel time is the same. For lack of time, we weren't able to investigate the effects of non-uniform sampling of the preview window in the presence of obstacles, so we run the real time implementation using a uniform sampling.

5.5 Experimental verification

We consider the industrial setup in figure 5.1. Two Adept Cobra s600 SCARA robots are performing pick and place operations while sharing the same working environment. Each manipulator has a dedicated controller and considers the other manipulator as a dynamic obstacle. Snapshots from a typical operation are depicted in figure 5.4.

In such industrial applications, the typical approach is for a programmer to specify intermediate paths between a large number of endpoints (for each robot). On these paths, acceleration profiles must then be defined. This process requires a lot of experience and takes considerable time and effort which prevents many manufacturers from using multi-robot systems. In contrast, the approach proposed here requires simply the desired endpoints to be specified by the user, while the intermediate trajectory is generated online.

In the experiment presented here, in addition to bounding the joint accelerations, constraints on the joint angles and velocities were imposed (see Table 3.1). The underlying optimization problems for *both* manipulators were solved on a single PowerPC CPU at 400

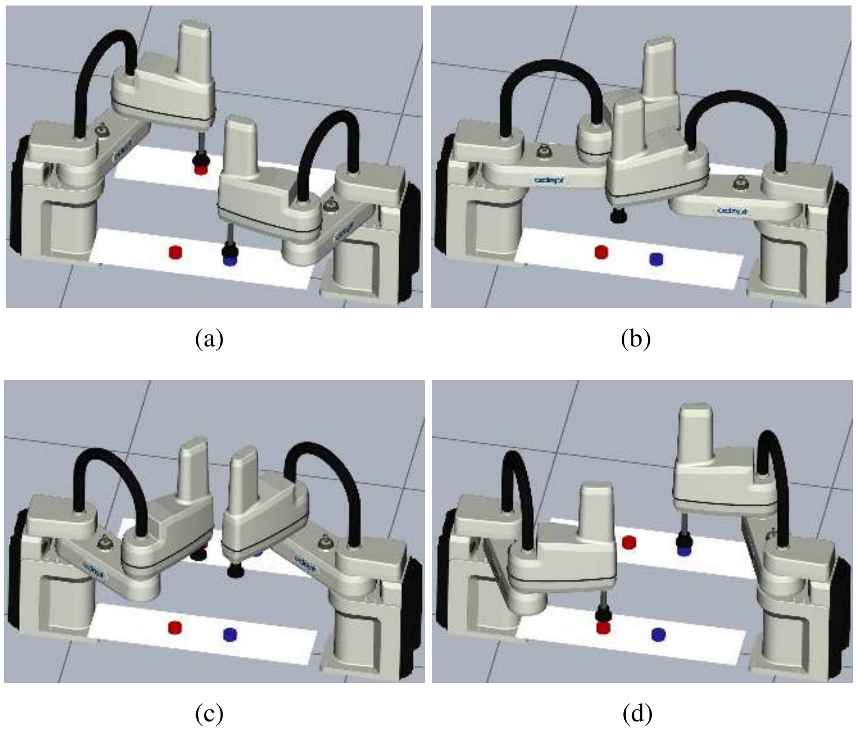


Figure 5.4: Snapshots from a typical pick and place operation (a) $\rightarrow$ (b) $\rightarrow$ (c) $\rightarrow$ (d).

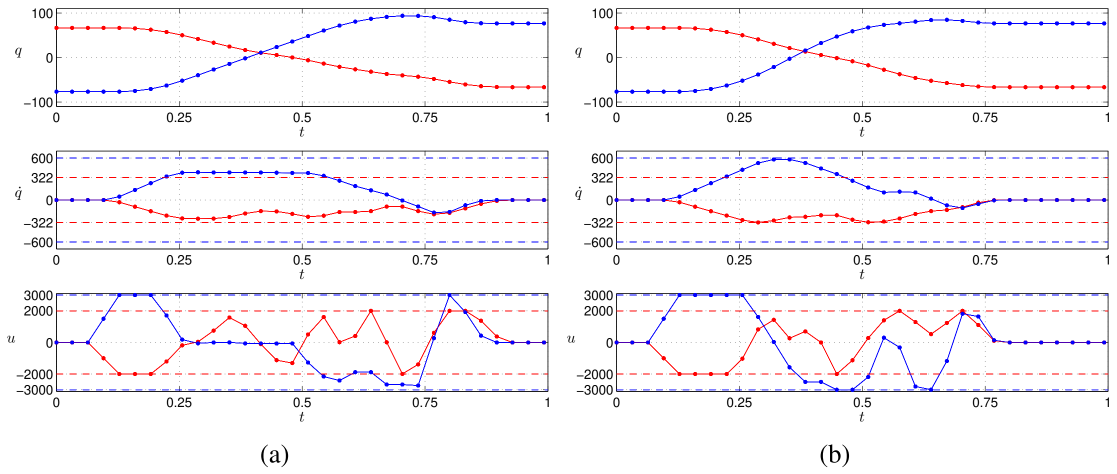


Figure 5.5: (a) Typical joint profiles of one pick and place cycle from the experiment in the video (for one of the robots) with $N^{max} = 5$. (b) A simulation result for the same end-points as in (a), however, with $N^{\max} = 7$ (the associated snapshots are depicted in Fig. 5.4). Profiles of joints 1 and 2 are depicted using red and blue, respectively.

MHz (32 ms control sampling time was used). This poses a challenge to our numerical approach (the hierarchical problems were solved using an implementation of the method in [22]).

Collision avoidance constraints were formed by considering each link of one manipulator (modeled using a swept sphere lines) as an obstacle for the other. We followed the linearization procedure described in Section 5.2.1. Since there was no notable state estimation noise, $N^{\min} = 1$ was used. Although we were aiming at having a preview length $N^{\max} = 7$ (which was verified to lead to very satisfactory results in a simulation study), due to the hardware limitations, $N^{\max} = 5$ was considered. The impact of this will be discussed later.

Figure 5.5 (a) depicts typical joint profiles of one pick and place cycle from the experiment (for one of the robots). The results demonstrate that online generation of fast manipulator motions with the proposed hierarchical approach is readily possible even with limited resources. Although our choice of $N^{\max}$ makes online computations feasible it, however, leads to an undesired "velocity saturation". Note how the velocity of joint 2 saturates at approximately 400 deg/s during the interval $[0.25, 0.5]$. This is a good indicator that by increasing $N^{\max}$ one can expect to achieve faster transitions. The results with $N^{\max} = 7$ (obtained in a simulation) confirm this. As can be seen on figure 5.5 (b), the velocities of both joints are very close to the actual limits and, in our experience, increasing further $N^{\max}$ leads to only a marginal gain. The resulting transition duration is 30% faster compared to the case with $N^{\min} = 5$. Our current efforts are in the direction of reducing this gap by means of improving our numerical tools so that a larger $N^{\max}$ can be used or by enhancing our formulation. For example, we are investigating the effects of non-uniform sampling of the preview window and alternative warm-starting techniques.

5.6 Evaluation

In Chapter 3, we proposed a reactive trajectory generator based on a hierarchy between a strict obstacle avoidance behavior and a control law which is time optimal in the absence of obstacles. The algorithm shows good results but there were some real problems and

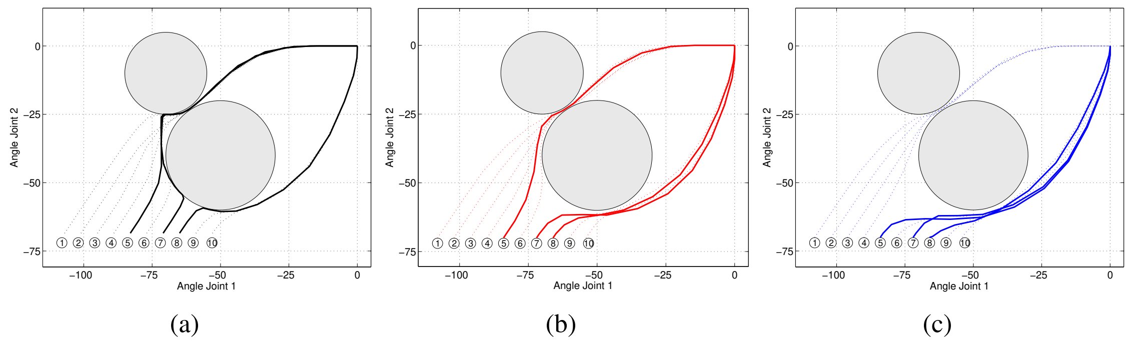


Figure 5.6: Ten trajectories. Each trajectory starts with zero velocity, converges to the origin and avoids two convex obstacle in joint space using (a) Bang-bang control approach (b) ℓ_1 norm approach ($\beta = 0.1$, $N = 8$) and (c) Hierarchical approach ($N = 8$). The effect of using MPC can be seen clearly in trajectories number 5, 7 and 8.

limitations due to nonlinearity in manipulators. In order to overcome those problem and limitations, two other approaches were introduced in Chapter 4 based on MPC. In this section, we aim at providing a more clear understanding of the advantages and limitation of all three proposed methods: bang-bang control approach (Chapter 3), ℓ_1 norm approach and hierarchical approach (Chapter 4). To reach this goal, we test them in different simulation examples and compare the performance of these approaches in terms of time optimality, reliability and computation time.

In the simulation tests presented here, we consider a SCARA robot with constraints in joints acceleration, velocity and angle (see Table 3.1). All the computation times in this section are recorded while running the simulation in OCTAVE on a PC with an Intel Core 2.70 GHz CPU.

5.6.1 Time optimality

In order to produce higher productivity and profit, industrial automation requires faster robots. A comparison in terms of trajectory duration will be presented in this section.

In the absence of obstacles, the trajectory generated by all approaches are time optimal. A SCARA robot will do any pick and place operation in the fastest way respecting acceleration, velocity, and angle joint constraints defined in Table 3.1. However, as we

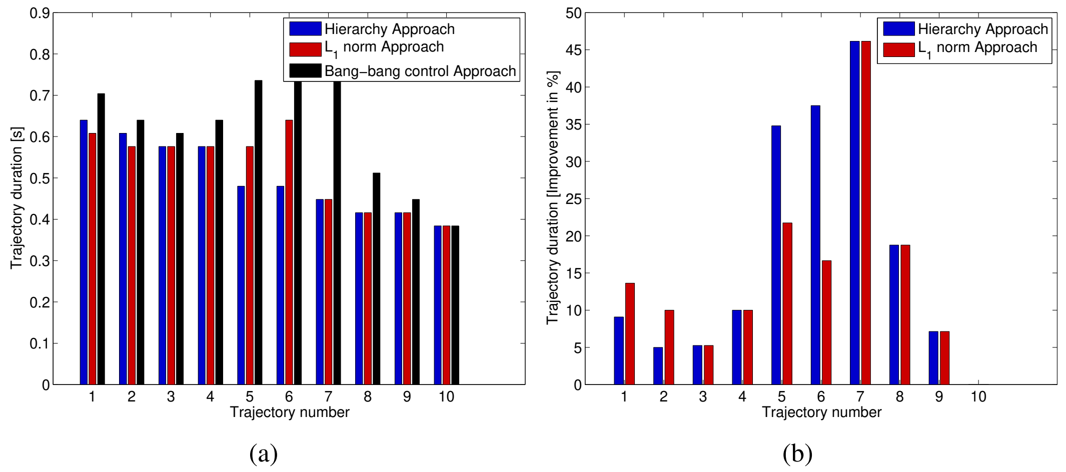


Figure 5.7: (a) Duration of each of the ten trajectories in Fig. 5.6 using the three different approaches. (b) The percentage of improvement in trajectories duration relative to the duration of the trajectories generated by bang-bang control approach.

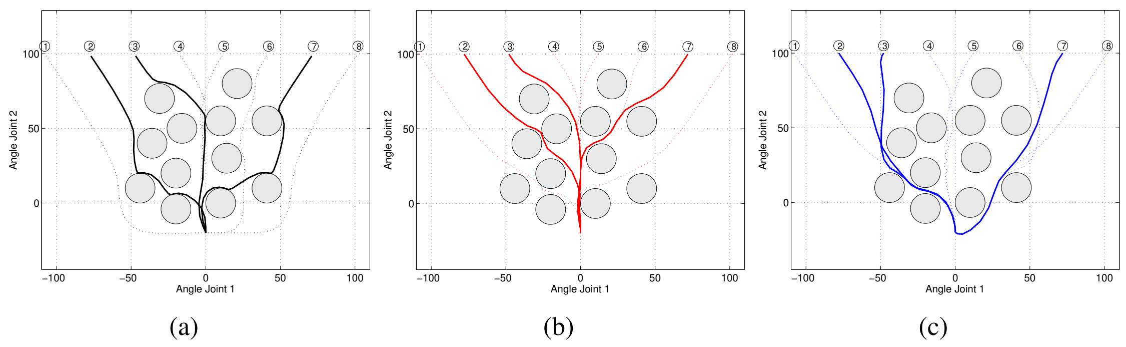
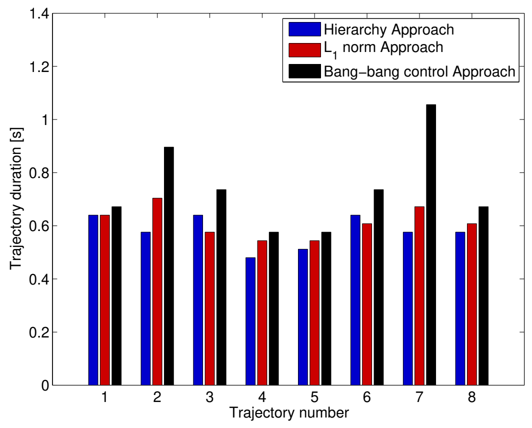
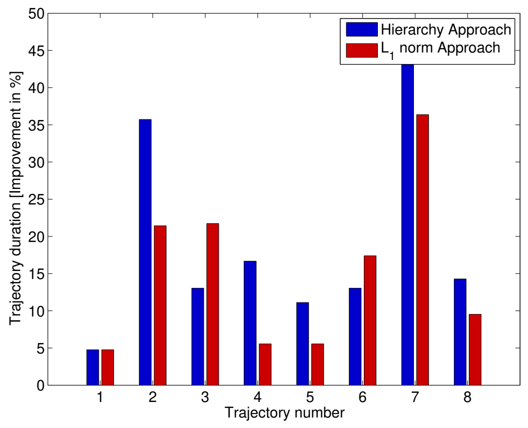


Figure 5.8: Ten trajectories. Each trajectory starts with zero velocity, converges to the origin and avoids two convex obstacle in joint space using (a) Bang-bang control approach (b) ℓ_1 norm approach ($\beta = 0.1$, $N = 8$) and (c) Hierarchical approach ($N = 8$). The effect of using MPC can be seen clearly in trajectories number 2, 3 and 7.

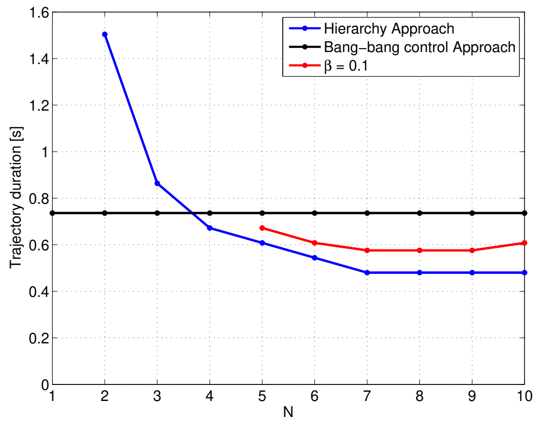


(a)



(b)

Figure 5.9: (a) Duration of each of the eight trajectories in Fig. 5.8 using the three different approaches. (b) The percentage of improvement in trajectories duration relative to the duration of the trajectories generated by bang-bang control approach.



(a)

Figure 5.10: Duration of trajectory number 5 in Fig. 5.6 for varying N.

saw in Section 4.4.2 a small modification of the constraints even any arbitrary linear constraints except a simple box constraints may affect the solution generated by the bang-bang control approach and make it far from time optimal. This is not the case with the other two approaches where a numerical solution is proposed to find the minimum time optimal solution in the presence of linear constraints.

In the presence of obstacles, ℓ_1 norm and hierarchical approach will anticipate future events and take control action accordingly, which is not the case of the bang-bang control approach where the control action is a function of the current state only (see Chapter 3). Thus, we are expecting to reach the destination faster and be closer to time optimal solution using ℓ_1 norm and hierarchical approach. Figures 5.6 and 5.8 show clearly this. The duration of each trajectory is depicted in figures 5.7 and 5.9 respectively. As can be seen, trajectories based on model predictive control (MPC) may result in more than 40% faster transition. Moreover, even with a small horizon length ($N = 4$), the hierarchical approach generates trajectories faster than the bang-bang control approach (see figure 5.10). To conclude, approaches based on MPC (ℓ_1 norm and hierarchical approach) overcome the bang-bang control approach in terms of time optimality.

Figure 5.10 depicts the influence of the horizon length N on the duration of the trajectory number 5 in figure 5.6. Note that the ℓ_1 norm approach always requires longer horizons than the hierarchical approach to achieve the same transition time. Furthermore, the hierarchical approach does not rely on the ad hoc selection of a weighting factor, while the other requires to choose an appropriate value of β. The comparison shows that trajectories generated by the hierarchical approach have mostly the fastest transition time (the minimum trajectory duration in figures 5.7 and 5.9).

5.6.2 Reliability

As we saw in Section 3.4.2, using the bang-bang control approach for articulated manipulators leads to some problems and limitations due to nonlinearity. Using MPC approaches allows a highly efficient way of handling online nonlinear optimization, where a sequence of linearized sub-problems is solved using an SQP type of approach. Each sub-problem identifies a minimum-time trajectory from the current state of the robot with respect to

The Algorithm	Type of problem	The solver	# of decision variable		# of inequality constraints			# of equality constraint
			Control variable	Slack variable	For slack variable	For acceleration, velocity and joint range	For each obstacle avoidance	
Bang-bang control approach	Quadratic Problem	QPsolve	2	1	0	12	1	0
ℓ_1 norm approach	Linear Problem	GLPK	2*N	2*N	8*N	12*N	N	0
Hierarchy approach	Sequential Quadratic Problem	LEXLSI	2*N	0	0	12*N	N	4*N

Table 5.1: The size of the optimization problem for the three proposed approach.

local linear approximations of the collision avoidance constraints.

In figure 5.11, MPC approaches will react earlier to the obstacle and respect the constraints, which is not the case if we are using the bang-bang control approach. Another example in figure 5.12 shows the advantage of using MPC against the bang-bang control approach. Due to nonlinearity in manipulators, the robot reacts to the obstacle knowing that the obstacle is outside robot's workspace. MPC approaches solve correctly this problem.

5.6.3 Computation time

The trajectory generator must work at high rates (100 Hz to 1 kHz) to react in real time to any change in the workspace on an embedded system with limited processing and memory resources. Generating safe, reliable and fast trajectories using a slow algorithm is not possible. Thus, comparing the proposed approaches in term of computation time is an essential criterion.

We have three different approaches, the size of the optimization problem for each approach is depicted in Table 5.1. The LEXLSI solver [22] is used to solve the hierarchical problem, while GLPK and QPsolve solvers are used to solve the LP and QP respectively. More powerful solvers can be used to solve the QP and LP however, we choose to work with open source ones. As we mentioned before, all the computation times are recorded while running the simulation in OCTAVE on a PC with an Intel Core 2.70 GHz CPU.

Figure 5.13(a) shows the computation time consumed by the solver at each sampling time to generate a trajectory for a SCARA robot performing pick and place operations in

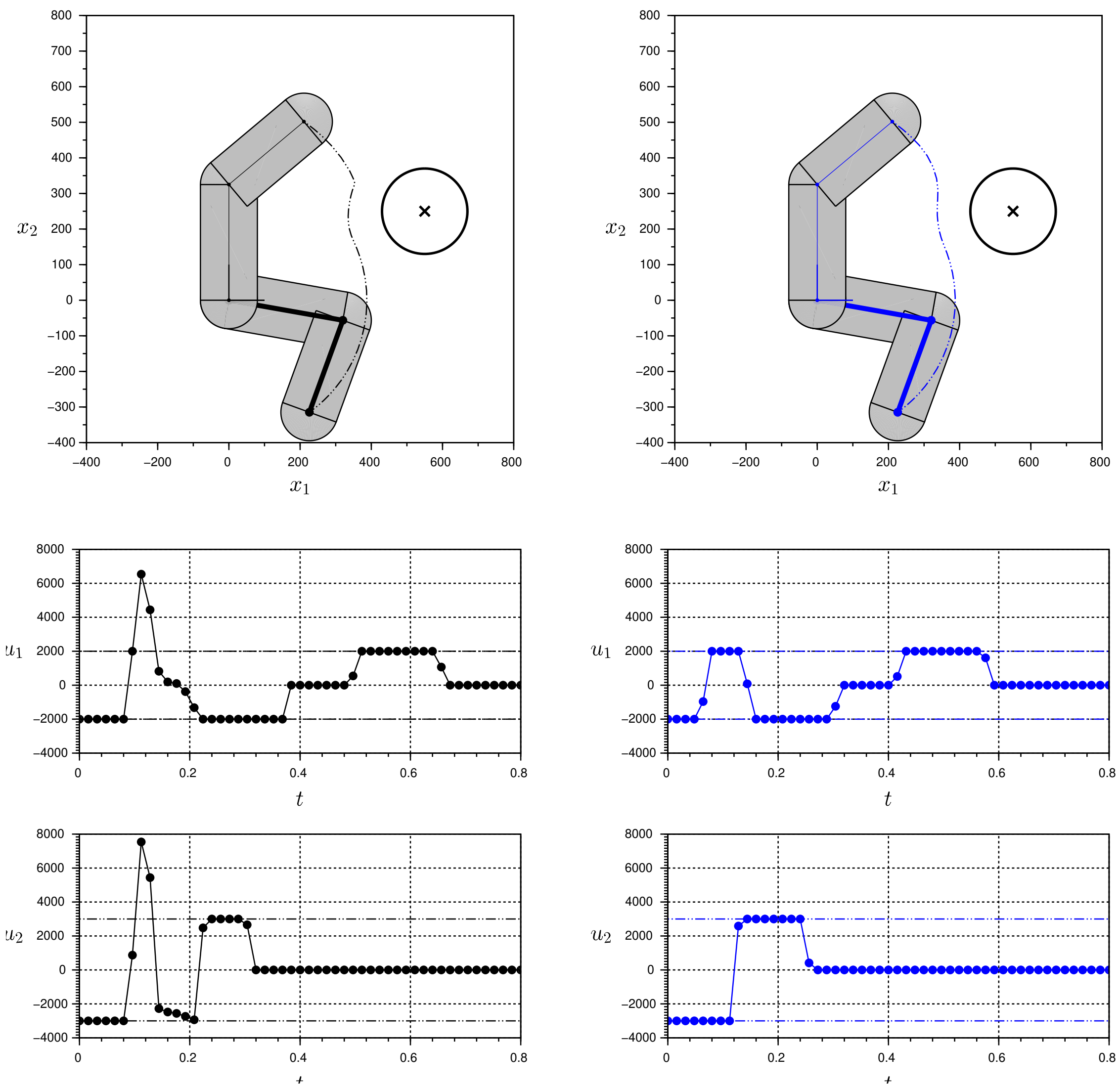


Figure 5.11: (a) shows the results of applying bang bang control approach, while (b) shows the results of applying MPC based approach.

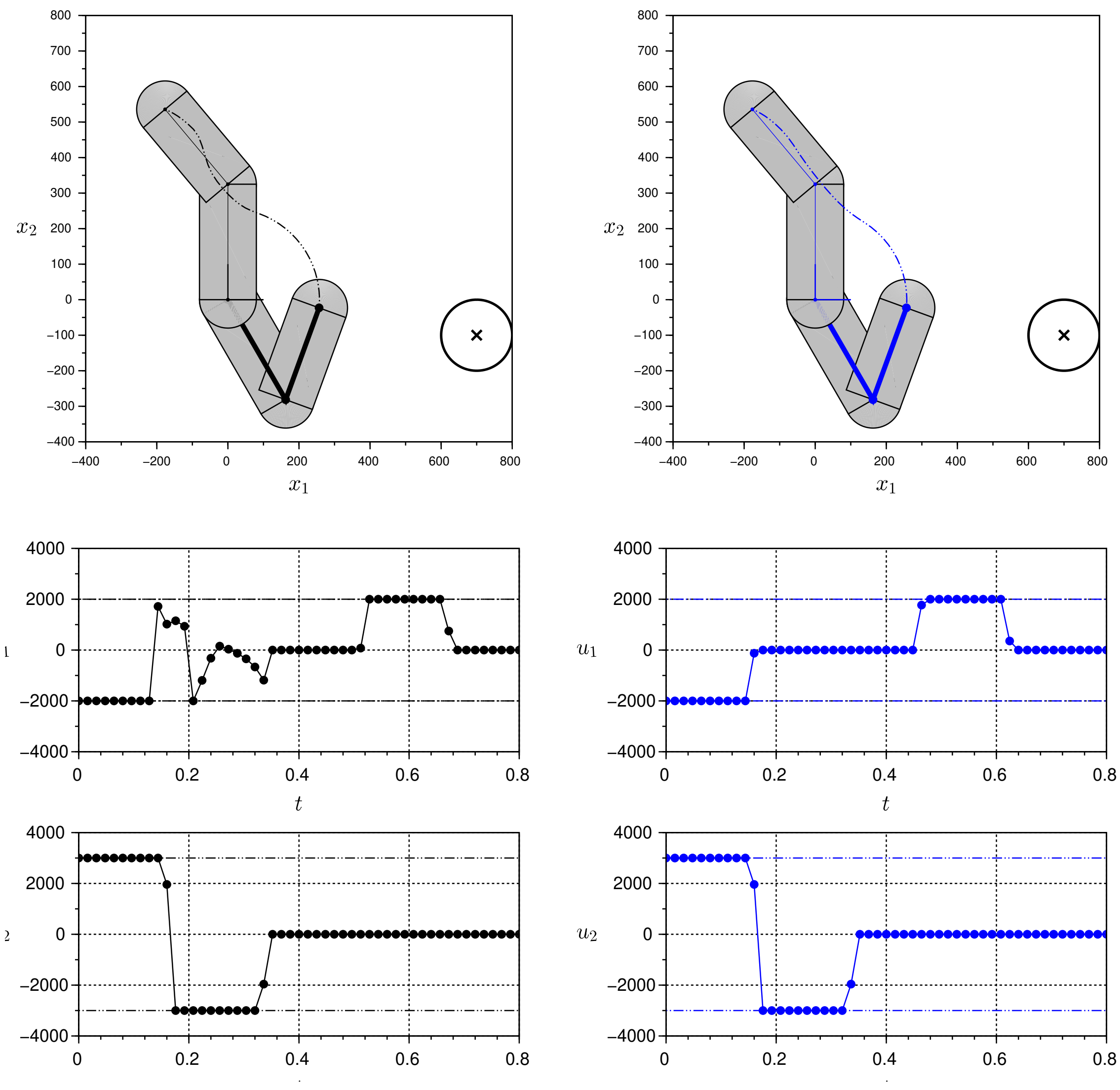


Figure 5.12: (a) shows the results of applying bang bang control approach, while (b) shows the results of applying MPC based approach.

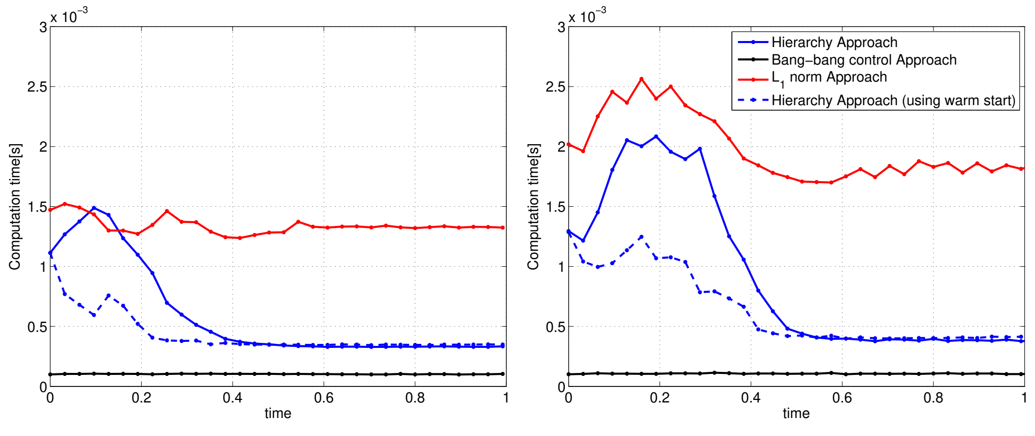


Figure 5.13: Solver computation time at each iteration using the three proposed approach (a) for the trajectory number 5 in Fig. 5.6 and (b) for the trajectory number 7 in Fig. 5.8.

the presence of two convex obstacles in joint space (see figure 5.6). The comparison shows that the bang-bang control approach is the fastest one in terms of computation time which is expected because this approach controls the current state without predicting any step in the future so this requires very little computation to determine the shortest distance to the obstacle and simple optimization problem, as Table 5.1 shows.

Figure 5.13(b) shows that increasing the number of obstacles in the workspace will increase the computation time in all approaches but with different ratios. In bang-bang control approach, each obstacle will add just one inequality constraint to the optimization problem. However, in MPC based approaches as each obstacle will add N inequality constraints, in addition to the time required to calculate the shortest distance for N predictions. The effect of introducing warm start to the LEXLSI solver helps in decreasing the computation time. However, it remains slower than the bang-bang control approach.

5.7 Conclusion

In this chapter, we validated the MPC approaches presented in Chapter 4 by applying them to online trajectory generation for industrial robots performing pick and place operations in the presence of dynamic obstacles. In particular, we presented an experimental evaluation using two SCARA robots that share the same working environment. The proposed formulation simplifies greatly the deployment of industrial technology, as it does not rely on the tedious and time consuming task of manually specifying paths between a large number of endpoints. We closed this chapter by evaluating the three different approaches proposed in this thesis. We tested them in different simulation examples and compared the performance of these approaches in terms of time optimality, reliability and computation time. We conclude that MPC approaches show the desired behavior; however they still require more work to reduce computation time.

Chapter 6

Conclusion

The work presented in this thesis opens new interesting doors in the fields of programming and controlling industrial robots. In this chapter, we provide a brief summary of the major ideas of this work, and bring to light its main limitations along with possible directions of improvement.

6.1 Summary

We have presented in this thesis a novel framework to deal with real-time collision avoidance for robots performing tasks in a dynamic environment. We developed a reactive trajectory generation algorithm that plans a bit in advance, following an MPC design, and provides a practical way of generating locally time optimal solutions.

We proposed first a reactive trajectory generator based on a hierarchy between a strict obstacle avoidance behavior, and a time optimal control law. This algorithm has a simple structure and low computational requirements, but it doesn't show the desired behavior when we apply it to a nonlinear system: some problems and limitations appear when we use it to generate reactive trajectories for manipulators.

In Chapter 4, we provided a novel approach to solve minimum time control problems for discrete time systems. Our approach is applied in an MPC setting, which not only improves reactivity of the system but presents a possibility to obtain accurate local linear approximations of the collision avoidance constraint. The proposed approach differs from existing ones in: (i) it does not rely on an ad-hoc selection of weighting factors (which is highly non-trivial), (ii) it does not lead to any approximation and results in time-optimal behavior for arbitrary linear constraints, (iii) and yet it is tractable in real-time.

In Chapter 5, we validated this approach by applying it to online trajectory generation for industrial robots performing pick and place operations in the presence of dynamic obstacles. It shows the desired behavior: 1) it reacts on-the-fly to dynamic changes in the workspace using limited embedded system, 2) it simplifies greatly the deployment of industrial technology, as it removes the fastidious optimization process which is traditionally executed by hand by handling it automatically, 3) it provide a practical way of generating locally time optimal solutions.

6.2 Future work

The concepts introduced in this thesis give rise to various questions for future work.

- Throughout this thesis, we have defined robot trajectories at the kinematic level, knowing that there is a low level tracking controller that converts kinematic variables into motor commands. The dynamics of the robot and its hardware limitations are not explicitly considered. They are implicitly taken into account through different constraints e.g. velocity, acceleration, and jerk constraints. An interesting extension to this work is to define robot trajectories at the dynamic level and apply the real physical constraints of the robot by considering explicitly the robot's hardware limitations e.g. maximum motor torques.

- In Chapter 5, we have validated our formulations by implementing them for online trajectory generation for industrial robots. We concluded that the MPC approaches show the desired behavior, but using a longer horizon N is required to generate faster trajectories and be closer to time optimal solution. Increasing further N will increase the size of the optimization problem so computation time will go up. According to the experimental results, solving the hierarchical optimization problem consumes around 70% of the computation time. An essential future work will be to improve our numerical tools so we can save some computation time and be able to use larger N and generate faster trajectories.

- In the framework presented in this thesis, we assumed that the information about the environment needed to avoid obstacles is already available. We are skipping the environment perception problem. An interesting long term future work would be to integrate sensors in the robot environment so we can estimate robot-obstacle distances based on data collected from sensors. This is an interesting development that will bring new utilities and open the door to human-robot cooperation.

- In this thesis, we developed a reactive trajectory controller that plans a bit in advance, following a standard MPC design. It provides a practical way of generating locally time optimal solutions. Another long term future work would be to integrate our current approach to a global planning algorithm so that we have at the end a reactive trajectory working at real time and generating a global time optimal solution.

Bibliography

[1] Inc Adept Technology. Cobra robots. http://www.adept.com.

[2] M. Alamir. *A Pragmatic Story of Model Predictive Control: Self Contained Algorithms and Cases Studies*. CreateSpace Independent Publishing Platform, 2013.

[3] Henrik Andreasson, Abdelbaki Bouguerra, Marcello Cirillo, Dimitar Dimitrov, Dimiter Driankov, Lars Karlsson, and et al. Autonomous transport vehicles: where we are and what is missing. *IEEE Robotics and Automation Magazine (IEEE-RAM)*, 22(1):64–75, 2015.

[4] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[5] R. E. Bellman. *Adaptive control processes: a guided tour*. Princeton University Press, 1961.

[6] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[7] J. E. Bobrow. A direct minimization approach for obtaining the distance between convex polyhedra. In *The International Journal of Robotics Research*, volume 8, pages 65–76, 1989.

[8] H.G. Bock, M. Diehl, P. Kühl, E. Kostina, J. P. Schlöder, and L. Wirsching. Numerical methods for efficient and fast nonlinear model predictive control. *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 162–179, 2007.

[9] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, Budapest, 1984.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University press, 2004.

[11] O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research,*, 21:1031—-1052, 2002.

[12] A.E. Bryson. *Applied Optimal Control: Optimization, estimation, and control*. Hemisphere Publishing Corporation, 1975.

[13] E Candes, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877 – 905, 2008.

[14] Centre Technique des industries mécaniques (CETIM). *Des robots pour les PME, Sécurité collaborative et intégration des robots en PME*, January 2010.

[15] D. Chen, L. Bako, and S. Lecoeuche. The minimum-time problem for discrete-time linear systems: a non-smooth optimization approach. In *IEEE International Conference on Control Applications (CCA)*, pages 196 – 201, 2004.

[16] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory,Algorithms, and Implementations*. MIT Perss, 2005.

[17] George B. Dantzig and Mukund N. Thapa. *Linear Programming 1 : Introduction*. Springer-Verlag, 1997.

[18] George B. Dantzig and Mukund N. Thapa. *Linear Programming 2 : Theory and Extensions*. Springer-Verlag, 2003.

[19] A. Dhanda and Gene F. Franklin. An improved 2-dof proximate time optimal servomechanism. In *IEEE TRANSACTIONS ON MAGNETICS*, volume 45, 2009.

[20] M. Diehl. Numerical optimal control. Course note from Optimization in Engineering Center (OPTEC) and ESAT-SCD, K.U. Leuven , Belgium, October 2011.

[21] T. Dietz, U. Schneider, M. Barho, S. Oberer-Treitz, M. Drust, R. Hollmann, and M. Hägele. Programming system for efficient use of robots for deburring in sme environments. In *7th German Conference on Robotics*, 2012.

[22] D. Dimitrov, A. Sherikov, and P.-B. Wieber. Efficient resolution of potentially conflicting linear constraints in robotics. *IEEE Transactions on Robotics (under review)*, 2015.

[23] D. Dimitrov, P.B. Wieber, O. Stasse, J. Ferreau, and H. Diedam. An optimized linear model predictive control solver for online walking motion generation. *IEEE International Conference on Robotics and Automation*, 2009.

[24] C. Ericson. *Real-time collision detection*. Elsevier, 2004.

[25] A. Escande, O. Kanoun, F. Lamiraux, N. Mansard, and P.B. Wieber. Control of redundant robots with equality and inequality tasks : theroritical and numerical aspects. *International Symposium on Robotics (ISR)*, 2011.

[26] A. Escande, N. Mansard, and P.-B. Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research (IJRR)*, 33(7):1006–1028, 2014.

[27] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar. Strictly convex hull for computing proximity distances with continuous gradients. In *IEEE TRANSACTIONS ON ROBOTICS*, 2013.

[28] B. Faverjon and P. Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. In *INRIA Domain de Voluceau*, pages 1152–1159, LE CHESNAY Cededx FRANCE, 1987.

[29] H. Feder and J. Slotine. Real-time path planning using harmonic potentials in dynamic environments. *IEEE Int.Conf. on Robotics and Automation,*, 1:874—-881, 1997.

[30] H.J. Ferreau, G. Lorini, and M. Diehl. Fast nonlinear model predictive control of gasoline engines. *IEEE International Conference on Control Applications*, October 2006.

[31] P. Fiorini and Z. Shiller. Time optimal trajectory planning in dynamic environments. In *IEEE International Conference on Robotics and Automation*, pages 1553–1558, Minneapolis, Minnesota, 1996.

[32] F. Flacco, A. De Luca, and O. Khatib. Motion control of redundnat robots under joint constraints saturation in the null space. In *IEEE International Conference on Robotics and Automation*, pages 285–292, 2012.

[33] F. Flacco, A. De Luca, and O. Khatib. Control of redundant robots under hard joint constraints: Saturation in the null space. *IEEE Transactions on Robotics*, 31(3):637–654, 2015.

[34] S. Forge and C. Blackman. The competitive outlook for the eu robotics industry. Technical report, Joint Research Center (JRC) Scientific and Technical Reports, 2010.

[35] Z. Gao. On discrete time optimal control: A closed-form solution. In *American Control Conference*, 2004.

[36] O. Gerelli and C. G. L. Bianco. A discrete-time filter for the on-line generation of trajectories with bounded velocity, acceleration, and jerk. In *IEEE International Conference on Robotics and Automation*, 2010.

[37] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. In *IEEE Journal of Robotics and Automation*, volume 4, 1988.

[38] S. Haddadin, R. Belder, and A. Albu-Schaeffer. Dynamic motion planning for robots in partially unknown environments. *The International Federation of Automatic Control*, 18:6842–6850, 2011.

[39] D. Hsu, J. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research*, 2006.

[40] The International Federation of Robotics – IFR. *History of Industrial robot*, 2012.

[41] H. Isermann. Linear lexicographic optimization. *Operations Research Spektrum*, 4(4):223–228, 1982.

[42] Stephen J.Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

[43] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond. A local collision avoidance method for non-strictly convex polyhedra. In *Conf. Robotics Science and Systems*, 2008.

[44] O. Kanoun, F. Lamiraux, and P.B. Wieber. Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks. *IEEE Transactions on Robotics*, pages 785–792, 2011.

[45] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Transactions on Robotics and AUtomation*, pages 566–580, 1996.

[46] H. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[47] S.M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots,*, 4:433–454, 2012.

[48] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(2):90–98, 1986.

[49] O. Khatib, L. Sentis, J. Park, and J. Warren. Whole-body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43, 2004.

[50] D. Kim and J. Turner. Near-minimum-time control of asymmetric rigid spacecraft using two controls. *Automatica*, 50(8):2084–2089, 2014.

[51] H. Kim, S. Lim, C Iuraşcu, F. Park, and Y Cho. A robust, discrete, near time-optimal controller for hard disk drives. *Precision Engineering*, 28(4):459 – 468, 2004.

[52] J-O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation,*, 8:338—-349, 1992.

[53] T. Kitahara and T. Takashi. Proximity of weighted and layered least squares solutions. *SIAM Journal on Matrix Analysis and Applications*, 31:1172–1186, 2009.

[54] K. Kon, S. Habasaki, H. Fukushima, and F. Matsuno. Model predictive based multi-vehicle formation control with collision avoidance and localization uncertainty. In *System Integration (SII), 2012 IEEE/SICE International Symposium on*, pages 212–217, Dec 2012.

[55] J.J. Kuffner and A.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995–1001, 2000.

[56] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation*, pages 1008–1014, 1991.

[57] F. Lingelbach. Path planning using probabilistic cell decomposition. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 467–472 Vol.1, April 2004.

[58] A. De Luca. Trajectory planning lecture at sapienza university. http://www.diag.uniroma1.it/ deluca.

[59] M. Melanie. *An Introduction to Genetic Algorithms. Cambridge*. Cambridge, MA: MIT Press, 1996.

[60] Katta G. Murty. *Linear complementarity, linear and nonlinear programming*. Sigma Series in Applied Mathematics 3. Berlin: Heldermann Verlag., 1988.

[61] Y. Nakamura. *Advanced robotics: redundancy and optimization*. Addison-Wesley Longman, 1990.

[62] H. Nakayama. Aspiration level approach to interactive multi-objective programming and its applications. In *Advances in Multicriteria Analysis (Kluwer Academic Publishers)*, pages 147–174, 1995.

[63] W. S. Newman. Robust, near time-optimal control. In *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, volume 35, 1990.

[64] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

[65] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish. Recent progress on programming methods for industrial robots. *Robotics and Computer-Integrated Manufacturing*, 28(2):87–94, April 2012.

[66] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Pergamon Press, 1964.

[67] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. *IEEE Int.Conf. on Robotics and Automation,*, 2:802—-807, 1993.

[68] J. Rawlings and D. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

[69] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Fast Continuous Collision Detection between Rigid Bodies. *Computer Graphics Forum*, 21(3):279–287, 2002. The definitive version is available at www.blackwell-synergy.com.

[70] Stephane Redon, Young J. Kim, Ming C. Lin, and Dinesh Manocha. Fast Continuous Collision Detection for Articulated Models. *Journal of Computing and Information Science in Engineering*, 5(2):126–137, 2005.

[71] J. Rosell and P. Iniguez. Path planning using harmonic functions and probabilistic cell decomposition. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1803–1808, April 2005.

[72] M. Saveriano and D. Lee. Point cloud based dynamical system modulation for reactive avoidance of convex and concave obstacles. *IEEE Int.Conf.Intelligent Robots and Systems (IROS),*, pages 5380–5387, 2013.

[73] Sven Schonherr. *Quadratic Programming in Geometric Optimization: Theory, Implementation, and Applications*. PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, 2002.

[74] Tom schouwenaars. *Safe Trajectory Planning of Autonomous Vehicles*. PhD thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, USA, 2006.

[75] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research (IJRR)*, 33(9):1251 – 1270, 2014.

[76] Z. Shiller and S. Sharma. Online obstacle avoidance at high speeds. *The International Journal of Robotics Research,*, pages 1030–1047, 2013.

[77] A. Shukla, R. Riwari, and R. Kala. Mobile robot navigation control in moving obstacle environment using a* algorithm. In *International Conference on Artificial Neural Networks in Engineerying*, volume 18, pages 113–120, 2008.

[78] B. Siciliano and J.-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Fifth International Conference on Advanced Robotics (ICAR)*, pages 1211–1216, 1991.

[79] L. Singh, H. Stephanou, and J. Wen. Real-time robot motion control with circulatory fields. *IEEE Int.Conf. on Robotics and Automation,*, pages 2737—-2742, 1996.

[80] R.F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1986.

[81] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *International Conference on Intelligent Robots and Systems*, pages 2053–2058, 2007.

[82] V.I. Utkin. Variable structure systems with sliding modes. In *IEEE Transactions on Automatic Control*, volume 22, pages 212–222, 1977.

[83] J. van den Berg, J. Snape, S.J. Guy, and D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. In *IEEE International Conference on Robotics and Automation*, pages 3475–3482, 2010.

[84] G. van den Bergen. Proximity queries and penetration depth computation on 3d game objects. *Game Developers Conference*, 2001.

[85] G. van den Bergen. *Collision Detection in Interactive 3D Environments.* The Morgan Kaufman Publishers, 2004.

[86] L. Van den Broeck, M. Diehl, and J. Swevers. A model predictive control approach for time optimal point-to-point motion control. *Mechatronics*, 21(7):1203 – 1212, 2011.

[87] A. Vazquez-Otero, J. Faigl, and A. P. Munuzuri. Path planning based on reaction-diffusion process. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 896–901, 2012.

[88] R. Vinter. *Optimal Control.* Modern Birkhaeuser Classics, Boston, 2000.

[89] O. von Stryk and R. Burlisch. Direct and indirect methods for trajectory optimization. In *Annals of Operations Research*, volume 37, page 357–373, 1992.

[90] O. von Stryk and R. Burlisch. Global linear convergence of an augmented lagrangian algorithm for solving convex quadratic optimization problems. In *Journal of Convex Analysis*, volume 12, pages 45–69, 2005.

[91] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267 – 278, 2010.

[92] S. Wardo and R. Murray. Vehicle motion planning using stream functions. *IEEE Int.Conf. on Robotics and Automation,*, 2:2484—-2491, 2003.

[93] M.L. Workman. *Adaptive Proximate Time-Optimal Servomechanisms*. PhD thesis, Stanford University, Stanford, CA, 1987.

[94] S. X. Yang and M. Meng. An efficient neural network approach to dynamic robot motion planning. In *Neural Networks*, volume 13, page 143–148, 2000.

[95] R. Zanasi, C. Guarino Lo Bianco, and A. Tonielli. Nonlinear filters for the generation of smooth trajectories. In *Automatica*, 2000.

[96] R. Zanasi and R. Morselli. Discrete minimum time tracking problem for a chain of three integrators with bounded input. In *Automatica*, 2003.

[97] V. Zanotto, A. Gasparetto, A. Lanzutti, P. Boscariol, and R. Vidoni. Experimental validation of minimum time-jerk algorithms for industrial robots. *Journal of Intelligent & Robotic Systems*, 64(2):197 – 219, 2011.

[98] Xinyu Zhang, Stephane Redon, Minkyoung Lee, and Young J. Kim. Continuous Collision Detection for Articulated Models using Taylor Models and Temporal Culling. *ACM Transactions on Graphics*, 26(3):Article 15, 2007.