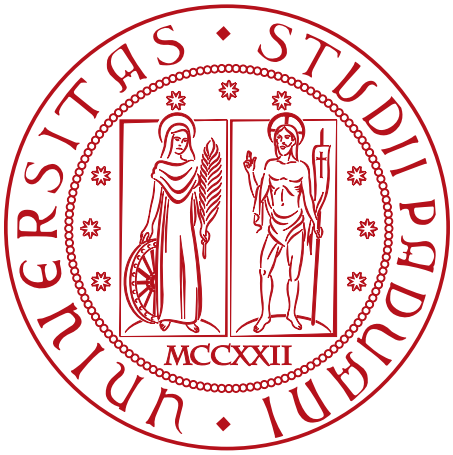


UNIVERSITY OF PADOVA
SCHOOL OF ENGINEERING
Department of Management and Engineering
Doctoral School in Mechatronics and Product Innovation Engineering
Cycle XXXII

OPTIMIZATION OF COMPLEX ROBOTIC TASKS FOR SMART MANUFACTURING APPLICATIONS

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mechatronics and Product Innovation Engineering

Prof. Giovanni Antonio Longo, Chair of the School
Prof. Daria Battini, Course Coordinator
Prof. Giulio Rosati, Candidate's Supervisor

Ph.D. Candidate: MATTEO BOTTIN

Padova
November 24, 2019

La famiglia è lo specchio in cui Dio si guarda e vede i due miracoli più belli che ha fatto: donare la vita e donare l'amore.

— San Giovanni Paolo II

ABSTRACT

Traditionally, the robots used in the industry field are made up of 6 links, so they can provide 6 degrees of freedom (DOF) in space. However, some particular applications or particular robot structures can provide additional degrees of freedom, thus creating a so-called "redundant task" or "redundant robot". Due to the structure of the robot, having 7 (or more) degrees of freedom results in a non-unique definition of the configuration of the robot: by choosing the values of the redundant axis angles, the robot is always able, within the working space, to satisfy the requested final position.

This feature can greatly increase the flexibility of the robot because can be used to avoid obstacles that could not be avoided with a traditional 6-axis serial robot. Moreover, taking advantage of the redundancy it is possible to avoid the singularities of the structure, thus improving motor consumption, movement times and transmitted forces. The reduced forces introduce a novel topic in robotics that has become more and more important in the last years: Human-Robot Collaboration.

Due to safety reasons, the speed and the contact forces that the robot can apply have to be limited. Moreover, the introduction of the operator within the workspace leads to an uncertainty of the obstacles placed in the workspace that the robot has to avoid. All these aspects can be managed with a redundant robot, and that's why most of the new collaborative robot arms are made by 7 or more joints.

The redundancy can be obtained also with passive tools: in specific tasks, such as welding, grinding and spraying, the same result can be obtained by rotating the robot structure around a specific axis, usually normal to the workpiece surface. Speaking about grinding, the redundant axis is coincident with the spindle axis: the grinder is circular, so every contact position around the grinder wheel circumference is suitable for the grinding process.

This work aims at providing a set of tools that can plan trajectories, avoid obstacles, schedule tasks and improve grinding finishing. The inspiration of this work relies on a real-world application: the robotic grinding. Even if most of the industry uses dedicated grinding machines, the flexibility of the robots makes them perfect to increase the flexibility of the machining process. However, the stiffness of the robot's structure is usually lower than the one of a dedicated machine, thus providing a worse finishing with the same cycle time.

In the first part of the thesis, a set of optimization tools are designed to find the optimal path that can move a redundant robot from one position to another without colliding the environment. This optimiza-

tion takes into account the redundancy of the structure. Moreover, an optimal task allocation algorithm is presented. To complete the dissertation, in the Appendix a novel collision detection algorithm is explained.

The second part focuses on a dynamic analysis of a robot: firstly, a modal study on a real six-axis serial robot has been performed; secondly, a comparison between a redundant under-actuated robot and a dynamically equivalent fully-actuated robot is illustrated.

Keywords: Redundant operation, serial robot, industrial robot, performance optimization, trajectory planning, Traveling Salesman Problem, robotic deburring

SOMMARIO

I robot utilizzati nell'ambito industriale sono prevalentemente composti da 6 elementi mobili, chiamati *links*, e dunque possono fornire 6 gradi di libertá (GDL, DOF in inglese) nello spazio. Esistono peró applicazioni e robot particolari che possono fornire gradi di libertá aggiuntivi, creando un cosiddetto "task ridondante" o "robot ridondante". A causa della struttura del robot, la presenza di 7 (o piú) gradi di libertá porta ad avere infinite configurazioni robotiche in grado di soddisfare la posizione finale: per farlo, é necessario solamente scegliere il valore delle rotazioni attorno agli assi ridondanti.

Grazie a questa funzionalitá é possibile aumentare la flessibilitá dei robot, in quanto cambiando la configurazione il robot potrebbe essere in grado di evitare gli ostacoli. Inoltre, si puó usare la ridondanza per evitare configurazioni singolari, migliorando quindi il consumo energetico dei motori, i tempi di movimento e le forze trasmesse dal robot. La riduzione delle forze permette di introdurre un argomento che sta diventando sempre piú popolare negli ultimi anni: la collaborazione uomo-robot.

Per ragioni di sicurezza, la velocitá e le forze di contatto che il robot puó imprimere devono essere limitate. Per di piú, l'introduzione del fattore umano all'interno della cella di lavoro porta ad un posizionamento incerto degli ostacoli nello spazio che il robot dev'essere in grado di evitare. Tutti questi aspetti possono quindi essere risolti utilizzando un robot ridondante, ed é per questo che la maggior parte dei robot collaborativi in commercio é dotato di 7 (o piú) giunti.

Anche un robot a 6 gradi di libertá puó essere ridondante: in particolari applicazioni, come la saldatura e la sbavatura, la stessa operazione puó essere eseguita ruotando attorno uno specifico asse, solitamente normale alla superficie del pezzo. La base della ridondanza é data dunque dall'end effector passivo. Parlando nello specifico della sbavatura robotizzata, l'asse ridondante é coincidente con l'asse del mandrino: la mola utilizzata nella lavorazione é circolare, quindi ogni punto attorno alla circonferenza puó essere utilizzato come punto di contatto.

Questa tesi vuole fornire degli strumenti in grado di pianificare traiettorie, evitare ostacoli, definire una sequenza di operazioni e migliorare la finitura della sbavatura robotizzata. L'ispirazione deriva proprio dalla sbavatura robotizzata, per la quale la maggior parte delle applicazioni industriali si basa su macchine di sbavatura dedicate. Il robot, in un contesto come questo, sarebbe perfetto per migliorare la flessibilitá del processo. Purtroppo, la rigidezza della struttura del

robot é decisamente inferiore a quella di una macchina dedicata, risultando in una finitura superficiale peggiore con lo stesso tempo ciclo.

Nella prima parte della tesi vengono presentati alcuni strumenti di ottimizzazione in grado di trovare il percorso ottimale che nuove un robot ridondante tra due posizioni senza collidere con l'ambiente. La ridondanza della struttura viene presa in considerazione per la definizione del movimento. Per di piú, viene presentato anche un algoritmo di allocazione del task basato sul famoso Traveling Salesman Problem. Per completare il progetto, nell'Appendice un nuovo algoritmo di collision detection é stato spiegato.

Nella seconda parte della tesi viene analizzata la risposta dinamica del robot: inizialmente é stato condotto uno studio modale su un robot a 6 assi presente nel Laboratorio di Robotica dell'Universitá di Padova; successivamente, é stata condotta una comparazione dinamica tra un robot sotto-attuato e un manipolatore completamente attuato dinamicamente equivalente.

CONTENTS

1 INTRODUCTION

1.1 ROBOT OFF-LINE PROGRAMMING

During the last years, the global installations of industrial robots have increased [1]. While their main segments of application are the automotive and electronic ones, during the last years there has been an increase of applications in which the robots are used for more complex tasks, such as the machinery.

The main difference between the first segments and the machinery one lies on the type of movement that the robot has to perform: while in the automotive and electronic segments most of the tasks are performed by simple point-to-point movements, in machinery most of the tasks require continuous movements between the working positions.

This leads to higher complexity in the definition of the tasks, whose via points are usually saved in a real workspace (on-line). This is a time-consuming process that can take days or weeks of work (so-called *setup time* or *reconfiguration time*), thus increasing the costs of the automation [2]. Since automation is usually used as a way of lowering the cost of labor in Western countries [3], the long reconfiguration times required to create new tasks do not suit well the unpredictable conditions that the industrial process has to face, such as different products with small (or varying) volumes [4].

It is possible to avoid this time consumption by simulating the task away from the work cell (off-line) and, then, upload it to the robot controller. Since the real environment is slightly different from the simulation, there can be a few position adjustments to be performed. This process is way faster than the previous one, while more reliable and flexible [5].

Off-line programming can be performed in parallel with product design [6] and improve work cell design. This can improve robot performances, since the shape of the product parts and the positions of the obstacles in the workspace are optimized to improve the final result (that is, usually, the cycle time).

1.2 ROBOT REDUNDANCY

Whilst many types of redundancy exists (such as software and sensing redundancy), one of the most common types is the kinematic redundancy. This type is not usually considered in many industrial

applications even if the performances of the work cell could be greatly increased taking advantage of this feature.

If a manipulator can provide more degrees of freedom than the minimum number required to execute a task, the kinematic redundancy occurs. For a complete definition of a position in space six degrees of freedom are required, so the most intuitive example of a redundant robot is provided by a 7 degrees of freedom manipulator that works in a 3D space. However, an even simpler structure could be considered: a SCARA robot built with more than the three usual parallel vertical joints is redundant because to define a position and an orientation on a plane only three degrees of freedom are required.

In this work, the term "redundancy" will be used as a synonym of "kinematic redundancy" unless otherwise specified.

When a manipulator is redundant, the inverse kinematic problem admits infinite solutions [7]. This aspect is crucial because implies that if the robot has to reach a certain position, it could perform it with different configurations. Moreover, it is possible to change one joint position and adjust all the other joints accordingly to maintain the end effector in the same position.

As stated by Yoshikawa [8], if a certain end effector speed is required, the configuration of the redundant robot can influence the speed that the joints have to reach. These joint speeds are directly related to the flexibility (or mobility) of the robot [9]. In fact, to obtain the best reliability in the workspace, an industrial manipulator should be made by at least 7 joints [10].

This feature can be analyzed from a different point of view: different joint speeds directly transform into different cartesian speeds. So, different configurations of the redundant robot can produce different cartesian speeds from the same end-effector position. Distinct speeds can be directly related to different movement times, an important aspect that can be related to the performances of the work cell.

Trajectory planning is very important in reducing cycle time. Many considerations can be made to reduce the overall cycle time [11]. Minimum kinematic parameters can be obtained by focusing on kinematics, dynamics or minimum kinetic energy factors. However, this optimization is already implemented into the controllers of common industrial robots. One of the parameters to be optimized is the path between the tasks, avoiding the objects placed in the environment.

Other types of redundancies could be called "discrete", in which the robot configurations allowed to complete a task are finite and depend on the external equipment. An example could be a pick-and-place task of a cube to be performed with an eccentric gripper: every configuration of the robot that places the gripper at 0°, 90°, 180°, 270° around the cube's base normal is acceptable.

In this dissertation, the redundant parameter will be called using "θ_{n+1}" or "q_{n+1}" depending on the application, where n is the num-

ber of degrees of freedom needed to completely identify a robot configuration in space.

1.3 STATE OF THE ART

A significant amount of research has been carried out in the area of redundant manipulators. Most of the proposed solutions use the Jacobian matrix to study a local solution [7], but there are quite a few other approaches.

Siciliano [7] has reviewed the main approaches proposed until 1989, and most of them require the Jacobian matrix. This means that in a real-time environment the algorithm has to calculate the matrix and its pseudoinverse to provide a solution to the problem.

Chirikjian and Burdick [12] introduced a class, named *hyper-redundant manipulators*, which include all those manipulators with very large, or infinite, number of degrees of freedom. This type of robots, with a unique continuous kinematics, is used to avoid obstacles. However, the design of this manipulator and the corresponding kinematics are complex, so they have mainly remained a laboratory curiosity [13].

Maciejewski and Klein [14] has used the Jacobian matrix of a manipulator to implement a dynamic collision avoidance system. Multiple sensors have to be used in the application, but the manipulator is able to avoid moving obstacles or to adapt the task to a moving object. The same approach can be used to be implemented in a neural network algorithm [15].

Several researchers [16, 17, 18, 19, 20] have proposed to use the Moore-Penrose generalized inverse of the Jacobian matrix to calculate the speeds of the joints required to obtain certain cartesian speeds [21]. From this approach, several applications have been developed.

Wang and Artemiadis [22] developed a closed-form inverse kinematics of a redundant robot with a spherical wrist. Dubey et al. [23] studied these manipulator movements using the pseudoinverse approach. This kind of manipulators is pretty easy to control since only the value of one joint angle is required.

Urrea and Kern [24], starting from the dynamic equations, have been able to control a 5 degree of freedom SCARA robot with different controllers.

In the industrial work cells, the robotic workspace is cluttered with objects. To avoid collisions, the volume can be simulated with off-line methods. Some of them are shown in [25]. In this case, some collision avoidance algorithms can be used to move away the robot from collision points and create a safe path [26].

One of the most efficient ways is to include the objects inside simpler safety volumes, such as the Swept Sphere Volume (SSV) [27], the rectangular Swept Sphere Volume [28] or the Oriented Bounding Box (OBB) [29]. The same approach can be applied to robot links [30].

Khatib O. [31] proposed a collision avoidance system based on potential fields: robots get repelled by the space zones in which the obstacles are placed, acting like repulsive forces. However, sometimes the robot can get stuck by particular obstacle shapes [32].

Redon et al. [33] described a continuous Collision Detection method for virtual objects that can be represented by 3-dimensional patches.

Rodriguez-Garavito et al. [34] used bounding boxes to encapsulate the robot and the objects in the environment. Then, the collision detection algorithm has to evaluate the intersection of the bounding rectangles to find out if a collision has occurred.

Multiple path planning algorithms have been proposed during the years [35]. Most of them, however, focus on the path planning of mobile robots in unknown environments [36, 37] or focus only on nonredundant robots [38].

Lozano-Pèrez and Wesley [39] proposed an interesting offline path planning algorithm based on the growing of obstacles and the shrinking of the moving object to the dimension of a point. Unfortunately, it cannot be applied to robots since they are made by multiple links that affect each other.

A common way to build a feasible path between two locations is using Sampling-ased methods [40], such as Rapidly-exploring Random Tree (RRT) algorithms [41, 42, 43] or Probabilistic Roadmap Methods [44], or computing robot's Jacobian matrix [45, 46].

These algorithms have been used both with kinematic and kinodynamic approaches [47] and can be used with robots with a high number of joints [48]. Usually, the Dijkstra algorithm [49] is used to find the optimal path.

Sciavicco and Siciliano [50] studied the redundancy of a manipulator, considering collision avoidance and limited joint range. In particular, the inverse kinematic problem for constrained manipulators is investigated, using optimization techniques that consider all the aspects listed before.

Tian and Collins [51] used a genetic algorithm to find the optimal trajectory of a generic redundant manipulator.

Doan and Lin [52] were able to optimize the task of a redundant robot while finding the best position of the robot base in relation to the task itself.

Panames-Garcia et al. [53] proposed a general formulation for the optimization of path placement of redundant manipulators considering single or multiple objective optimizations.

One of the algorithms that can be used to find the optimal task sequence is the traveling salesman problem (TSP) [54]. Several implementation techniques have been proposed to solve this problem [55, 56, 25, 57, 58] but for a large number of "cities" (the points that have to pass through) the complexity increases and the computational times can become unacceptable. However, with the increase in com-

putational capabilities, a near-optimal solution can be obtained in reasonable times, making it easily applicable [59, 60].

Edan et al. [61] used the TSP to find the optimal sequence of fruit picking. In this case, a set of sub-volumes has been considered to reduce computational time.

Kumar and Luo [62] proposed to solve a dynamic TSP by switching to a time-related resolution, by finding the optimal task sequence based on the position of a drum machine used in the assembly task (which require a fixed time to rotate). The same similar application had been solved using TSP a few years before by Chan and Mercier [63]. Balakrishnan and Jog [64] used a modified version of the TSP to group similar parts into families to be worked together in different work cells.

Most of the research efforts in the under-actuated robots rely on the definition of a good control system [65, 66]. Vibrations are crucial aspects [67] in many fields of applications since most of the time a high precision is required [68].

In the same way of the trajectory control, in mitigating the vibration effects of the under-actuated manipulators the research has focused on the definition of a robust control system [69, 70, 71, 72], but in most of the industrial applications the robot to be used is a pre-built industrial robot, on the end of which is attached a passive tool. This tool can be fixed or mobile, thus providing under-actuated joints.

It is clear how many studies have been performed on the field, but none of them have considered all the aspects of the optimization of the task at once. In this dissertation, an entire optimization procedure and analysis will be carried out to fill this gap.

1.4 REDUNDANT ROBOTS IN INDUSTRIAL APPLICATIONS

According to the International Federation of Robotics (IFR) [73], since 2010 the demand for industrial robots has accelerated considerably due to the ongoing trend toward automation and the continued innovative technical improvements of industrial robots. Between 2012 and 2017 the average robot sales increase was at 19% per year. Moreover, in 2017 robot sales increased by 30%, and the main drivers of this growth were the metal industry (+55%) and the electrical/electronic industry (+33%). It is expected to increase robot sales by 14% per year at least until 2021.

Most of the industrial robots are built with 6 joints because is cheaper for the manufacturers and is easier to be controlled and installed. This is perfectly fine in most of the applications: in 3D space, 6 degrees of freedom have to be provided to uniquely define a position.

However, in many applications, the task that the robot has to perform is redundant itself or has to be performed via a redundant tool. Let's think about one of the most common applications [74]: assembly.

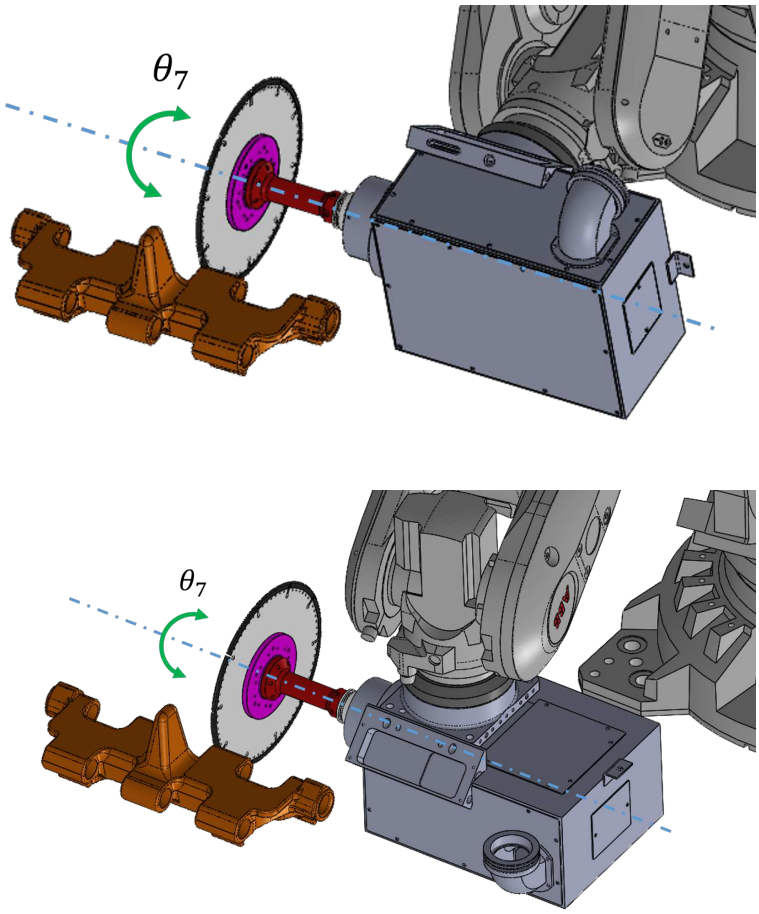


Figure 1: Deburring's redundancy is achieved by rotating the robot around the spindle's axis.

It is possible to pick up a workpiece in many positions: a cylindrical part can be grabbed by a gripper in any position around its axis of symmetry, a box can be grabbed by any position along its side and with any couple of opposite faces.

Speaking about the main drivers of 2017 growth, redundancy can be achieved in many ways: in the electronic field the chips can be grabbed with many robot configurations and can be placed accordingly (with discrete positions), and many applications in the metal industry rely on a redundant tool to complete the tasks.

Deburring is one of the redundant applications of metal industry: the grinding wheel is axis-symmetric, thus, within the robot joint's limits, it is possible to change the configuration of the robot by rotating around the wheel's axis (Figure 1).

A new category of industrial robots is called *cobots* [75], robots whose purpose is to physically interact with humans in a shared workspace. While their tasks can be very similar to traditional industrial robots, safety requirements (Section 1.5) usually demand the robot to be redundant.

Collaborative robot's sales are expected to grow by 57.5% CAGR until 2025 reaching a market share of 24.7% (while in 2018 the market share is 5.5%) [76]. Improving the path planning of collaborative redundant robots is a crucial point in the growth of this segment.

1.5 SAFETY REQUIREMENTS

Many of the collaborative robots available on the market are made by 7 joints, thus can provide 7 degrees of freedom, resulting in a redundant kinematic chain. The UNI EN ISO 10218 [77, 78] specifies requirements and guidelines for integrated safe design, protective measures, and information for the use of industrial robots. Within all the regulations, a few specifications about collaborative human-robot systems are provided.

Given the variable nature of the hazards of the different uses of industrial robots, the first part of ISO 10218 [77] specifies requirements and guidelines for the inherent safe design, protective measures, and information for use of industrial robots. It describes basic hazards associated with robots and provides requirements to eliminate, or adequately reduce, the risks associated with these hazards.

The second part of ISO 10218 [78] has been created in recognition of the particular hazards that industrial robot systems present when integrated and installed in industrial robot cells and lines. Hazards are frequently unique to a particular robot system. The number and types of hazards are directly related to the nature of the automation process and the complexity of the installation. The risks associated with these hazards vary with the type of robot used, its purpose and how it is installed, programmed, operated and maintained.

The most important requirements described within ISO 10218 that can affect the solutions proposed in this dissertation are correlated to the design and maximum speed of the collaborative robot. The design of the robot (including the end effector) has to be compliant to the risk assessments described in the second part of ISO 10218 [78] and previous regulations.

The maximum speed of the tool center point, according to the first part of ISO 10218 [77], must not exceed $250 mm/s$ (Section 5.6.2). Another important aspect to be considered is the singularity protection (ISO Section 5.11): cartesian space movements that pass near singularities can produce high axis speeds. These high speeds can be unexpected to an operator. The redundancy of the robot can be used to avoid singularities.

Right now many robot manufacturers are trying to push robot limits by adding safety features. Kuka [79] sells a 7-axis robot (Figure 2) with integrated joint torque sensors. The low weight of the robot coupled with the torque sensors allows the collaboration with a human operator. The same principle is applied in the design of other collab-

Figure 2: Collaborative robots: Kuka LBR iiwa (to the left) and Fanuc CR-35iA (to the right)

orative robots, such as ABB YuMi [80], Rethink Robotics Sawyer [81] and Universal Robots UR series [82].

Fanuc [83] and Comau [84], however, provide the collaborative robots with the highest payload: CR-35iA (Figure 2), with a maximum payload of 35kg, and AURA, with a maximum payload of 170kg. To achieve this result, both the manufacturers have equipped regular industrial robots with a rubberized soft skin. Within this skin there are FT sensors, so the robots are able to sense any impacts and react accordingly. Moreover, the rubberized skin removes all the sharp edges and provide cushioning.

1.6 AIM OF THE WORK

The use of a redundant robot is directly related to the human operator that has to define the robot recipe. The aim of this research is to provide a set of algorithms that can free the design of a robotic work cell from human factors, with a particular focus on robotic deburring.

The main research questions of this study can be summarized as follows:

1. Which is the best way to avoid obstacles in a still environment using the redundancy?
2. Is it possible to reduce the cycle time by changing the task order? Is it possible to adapt this order to the task by clustering the subtasks?
3. How should a compliant system behave during the deburring to provide a good finishing?

While the first two questions are directly related to the kinematics of the system, the last one includes dynamic aspects of the problem.

Consequently, the algorithms that will be proposed as answers to the first two questions are more related to off-line programming, trying to reduce the setup time of the work cells, the creation of new recipes and, thus, the operative costs. However, the same algorithms could be used in real-time with a static environment but a flexible task, such as assembly tasks with flexible assembly systems [85, 86, 87].

On the other hand, the dynamic analysis of a robotic arm provides the basic concepts for the future design of compliant systems as robot end effectors.

1.7 OVERVIEW OF THE DISSERTATION

Aiming to address the scientific and technical questions raised in Section 1.6, this dissertation presents software tools to be used in movement and task optimization (Figure 3). Some of the algorithms proposed are a first novelty for this work, while other improvements make more flexible something that is well known in the literature. The dynamic comparison between underactuated and fully actuated equivalent systems is a first novelty since most of the research has focused on the controllability of the underactuated systems.
The subsequent chapters of this dissertation are organized as follows.

Chapter 2 introduces the problem of robot deburring, the tools used in the machining process and how the forces are related to the amount of material that has to be removed.

Chapter 3 presents two novel algorithms to be used to define safe movement between working positions. While the first one uses a net of safe via points to draw the movement, the second one relies on an iterative process with continuous collision detection that results in a movement described by several safe via points close to the obstacles. The first approach is compared to a similar method, the PRM, to point up the improvements carried by this method. The collision detection method, developed for this work, is described in Appendix A.1.

Chapter 4 presents an extension of the well known Traveling Salesman Problem (TSP). This extension aims at increasing the capabilities of the TSP by including the possibility of dividing the working locations into clusters and the possibility of constraining a connection to be performed or, on the contrary, to not be performed. In this way, the TSP becomes more suitable for many other fields of application, such as logistics.

Chapter 5 presents, firstly, a modal analysis of an industrial six degree of freedom (DOF) serial robot with the aim at knowing which joints are the most responsible of the compliance on the end effector, and if the six DOF system can be simplified by removing the joints that are not so relevant in a vibration point of view. An implementation of the Mozzi axis approach has been included to highlight the

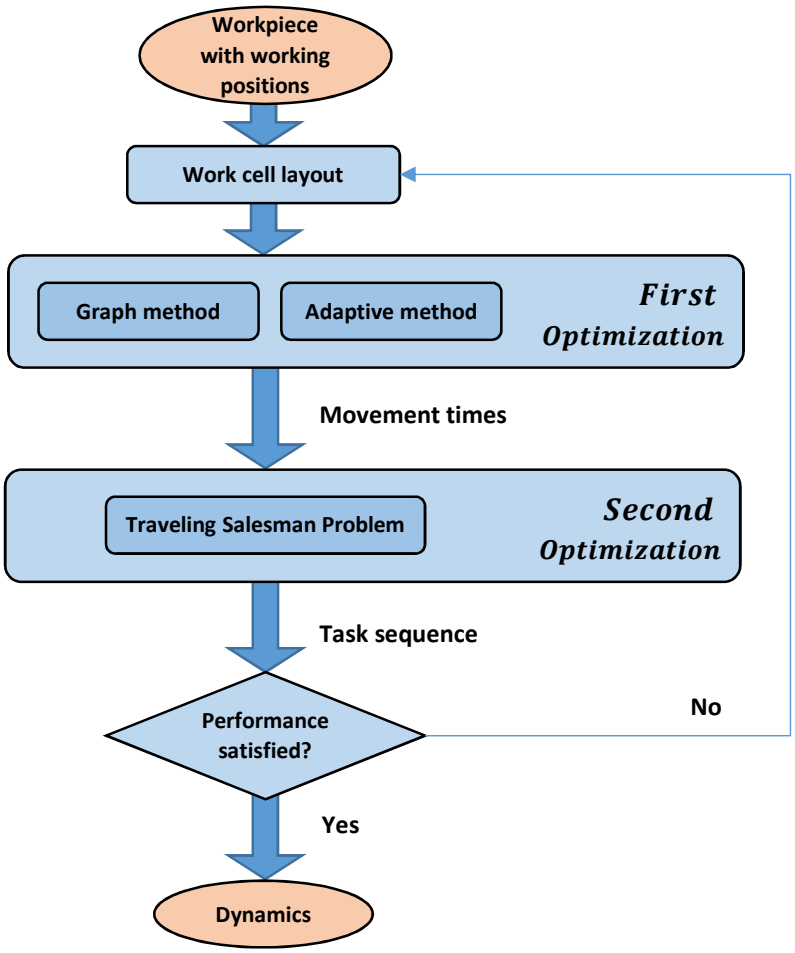


Figure 3: Overview of the optimization procedure that will be illustrated in the dissertation.

compliance of the robot with respect to the force applied to the end effector. Finally, a dynamic comparison between an underactuated 3 DOF robot arm and a fully actuated 2 DOF robot arm is illustrated.

Finally, in Chapter 6 conclusions and future work are presented.

Published work

Parts of this thesis have been published or will be published in the future. During the drafting of this thesis, the published works can be found in [26, 88, 89, 90].

2 ROBOTIC DEBURRING

This chapter presents the robotic deburring and the forces that are exerted by the material to be removed.

2.1 THE DEBURRING PROCESS

In the manufacturing process of a foundry product, after casting there are some part of the material that have to be removed: the molding results in small parts, the burrs, as extrusions of the exceeding material used in the process; the sand casting results in risers used to prevent cavities due to shrinkage.

This exceeding material must be removed before the final machining. Depending on the actual composition of the material, on the quantity used in the process and on other ambient factors, a high amount of material may be removed. Deburring is an intermediate machining process that anticipates milling or turning in order to remove most of the material before the finishing task. In other words, deburring is used to maintain the final geometry within certain limits.

The higher the amount of material to be removed, the higher the forces transmitted from the machining process to the robot's structure or, in another way, the time required to remove the burr - that can be non-linear [91].

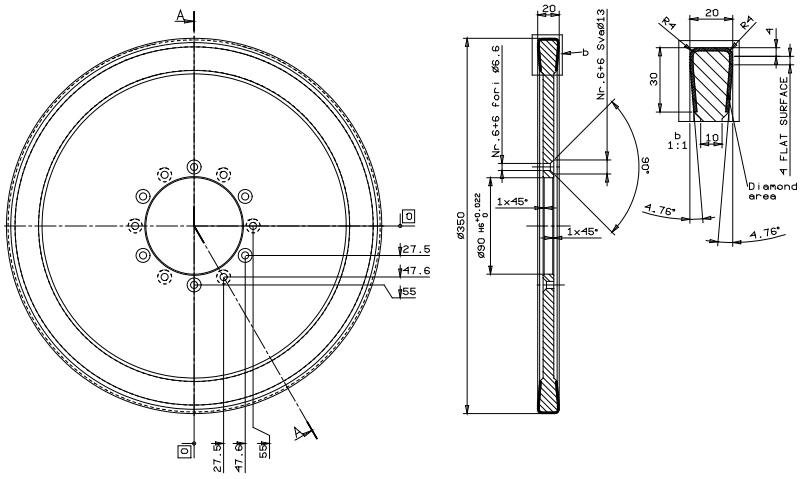
The main forces that arise during the deburring are tangent and normal to the burr [92], but their entity depends on the burr itself. As reported by Gillespie [93], two factors related to workpiece material are directly related to burr size: the ductility and the strain-hardening exponent of the material.

2.2 DEBURRING TOOLS

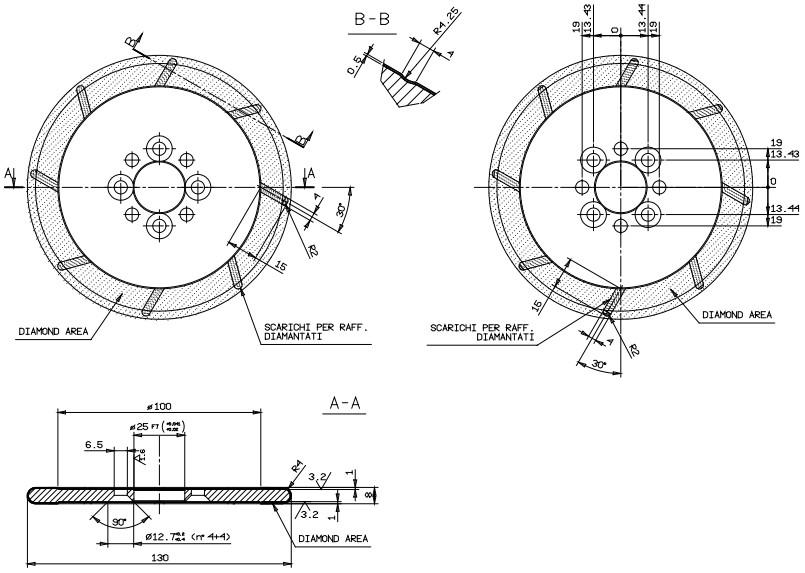
To remove the exceeding material, different tools can be used, depending on the material properties, the shape of the burr and its position, and the accessibility of the area. Other factors, such as economics, can be introduced in the selection of the right tool, but will not be included in the dissertation.

During the first part of my Ph.D., I have analysed and categorised the most common deburring tools to obtain a wide overview of the possible differences within the same machining process.

The most common deburring tool is the grinding wheel (Figure 4), a wide disk with a small height coated with phenolic resins or diamond (more common). Resin-bonded grinding wheels are less used



(a) Grinding wheel with a small diamond coating on the side

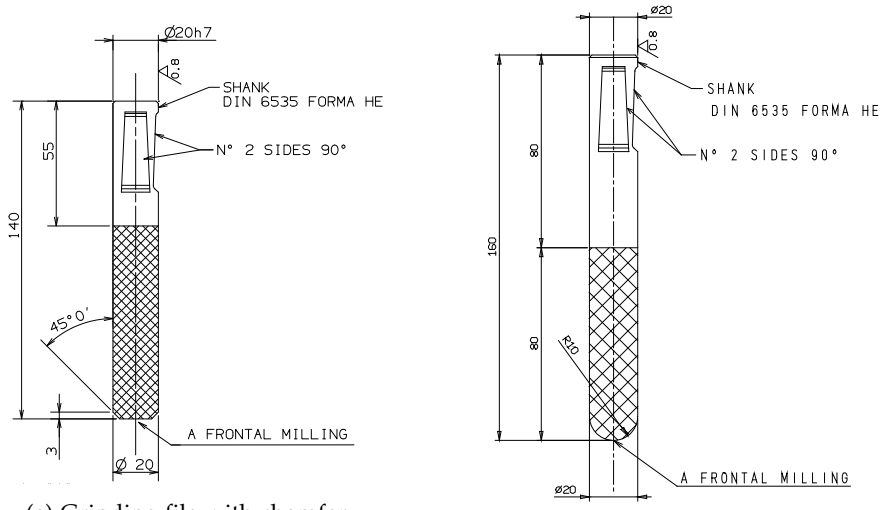


(b) Grinding wheel with a large diamond coating on the side

Figure 4: Two examples of grinding wheels used in the general grinding.

than the diamond wheels due to safety and application reasons: if overheated, these wheels are likely to shatter, sometimes with small explosions. Moreover, the continuous usage tends to remove the coating material, thus modifying the shape of the wheel: periodic shape checks have to be performed within the work cell. Diamond grinding wheels are more expensive, and the height of the coating (starting from the outer radius) may vary on the application: riser removing can be completed with a small diamond coating (Figure 4a), since most of the removal is performed with the cylindrical face of the wheel, while burr removing can be performed with the lateral surfaces of the wheel, thus requiring a higher coating (Figure 4b).

Another tool that is commonly used in the deburring process is the file (Figure 5). This (typically) cylindrical tool is used to remove



(a) Grinding file with chamfer.

(b) Grinding file with rounded tip.

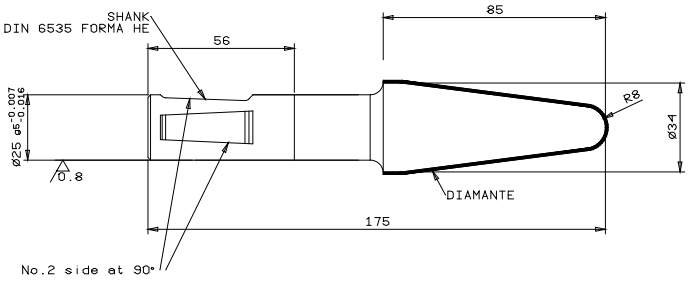
Figure 5: Two examples of files used in the grinding of small burrs around the holes.

the burr placed within the holes of the workpiece and in all those positions where is difficult to reach the burr with the big grinding wheel.

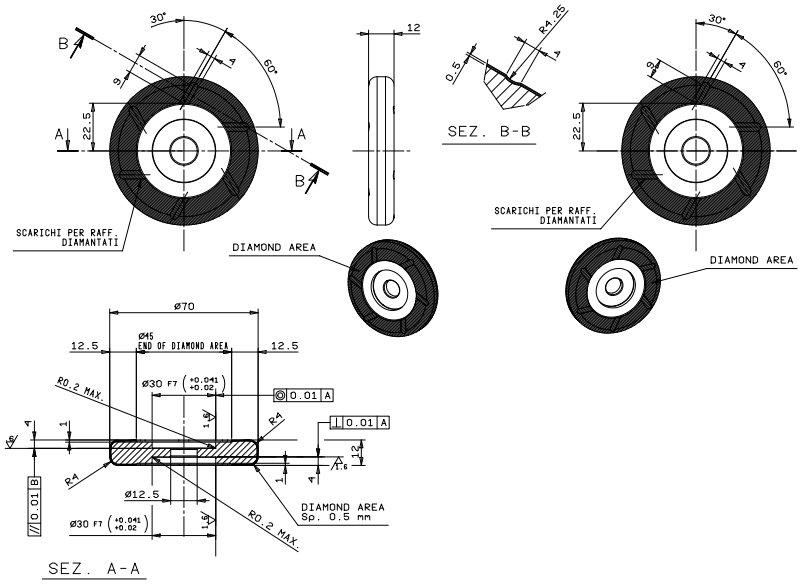
Particular tools can be manufactured depending on the applications: conically shaped tools (Figure 6a), small grinding wheels (Figure 6b) and composite tools can be used.

Opposite to the milling process, the grinding uses tools with a coarse surface, while the milling commonly uses tools with inserts and other pre-manufactured surfaces. All the deburring tools are coated with grains of different dimensions. Even if the deburring surface is irregular, it is possible to simplify a grinding tool with simple shapes, such as cylinders.

All these tools are used both in the traditional deburring process and in the robotic deburring. The main difference between the traditional and the robotic deburring lies in the work cell layout: while in the traditional deburring the workpiece is usually still and the machine moves around to remove all the burrs, the robotic deburring can be designed such as the workpiece is still and the end effector of the robot is equipped with the spindle and quick coupling tools (Figure 7), or the spindle is fixed and the workpiece is held by the robot with a gripper.
This particular aspect can provide high flexibility in the design of the system.



(a) Grinding file with a conic shape



(b) Small grinding wheel used in volumes with accessibility problems.

Figure 6: Two examples of special tools used in particular applications.

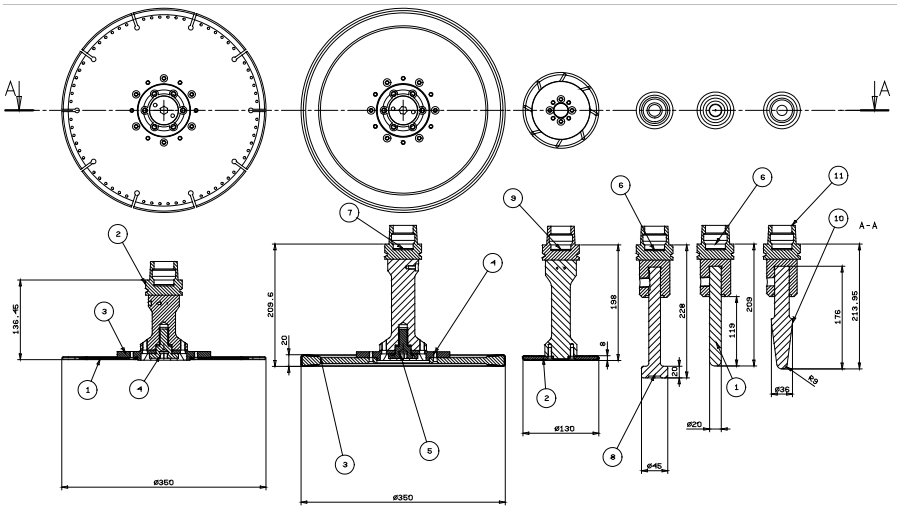


Figure 7: Assembly of quick coupling tools. Multiple tools can be installed within the same work cell to provide high flexibility.

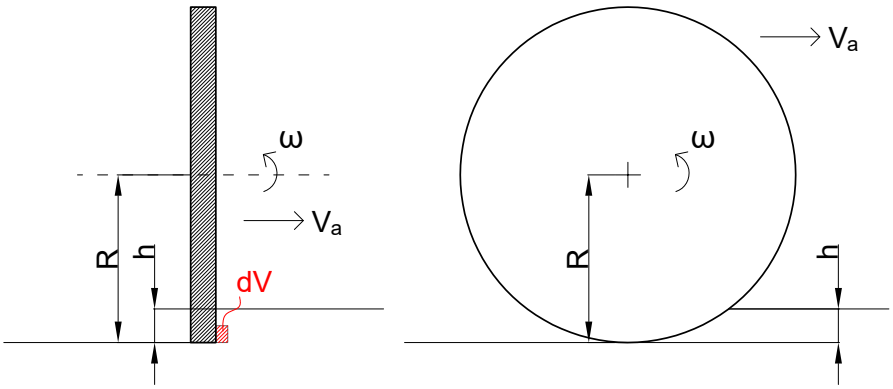


Figure 8: Two main approaches to the burr: deburring using the flat grinding wheel's face (to the left) and deburring using the cylindrical face of the wheel (to the right)

2.3 SIMPLIFIED MATHEMATICAL MODEL

Very little literature on the deburring is provided [94, 95], and usually, the working parameters are derived by experience. To obtain a result, since no actual robotic deburring equipment has been used to gather real data, a simplified mathematical model of the deburring has been developed.

To remove the material a grinding wheel (or a spindle) is used. There are two main different approaches to the burr (Figure 8): using the flat surface or using the cylindrical surface of the grinding wheel. The closer to the center of the wheel, the slowest the cutting speed of the abrasive grain: the cylindrical deburring (Figure 8 to the right) is removing the material at the same cutting speed: all the grains used in the process are placed on the outer side of the wheel, so their speed is calculated as the product of the rotating speed and the radius of the wheel, independently from the burr's height. The flat surface deburring (Figure 8 to the left), however, removes the material at different cutting speed, and this aspect is very important if it is also considered that the burr varies a lot in height during the task.

Even if the cylindrical deburring is preferable, it is usually impossible to use this process due to the workpiece configuration.

Since the flat deburring is more complex, the mathematical model is based on this variant. The fundamentals of this model are provided by the grinding process, while a similar approach has been developed in [94].

The infinitesimal volume dV of Figure 8 removed in the unit of time (also called *material removing rate*, *MRR*) can be easily calculated as:

$$dV = b \cdot dr \cdot v_a \qquad \left[\frac{mm^3}{s}\right] \tag{1}$$

where b is the width of the burr, dr is the infinitesimal height of the burr and v_a is the relative speed between the center of the wheel and the workpiece. If MRR is multiplied by the specific removal energy u [Ws/mm^3], the result is the power required by the motor to remove dV in a single unit of time:

$$dP = b \cdot dr \cdot v_a \cdot u \quad [W] \tag{2}$$

In the same way, however, the power provided by the tool attached to the grinding wheel can be calculated as the torque dC to be applied by the motor and the rotational speed ω. This speed is to be considered constant since the maximum power provided by the tool has to be adequate to the task:

$$dC = \frac{dP}{\omega} \quad \rightarrow \quad dC = \frac{b v_a u}{\omega} dr \quad [Nm] \tag{3}$$

From there it is possible to calculate the deburring force:

$$dF = \frac{dC}{r} \quad \rightarrow \quad dF = \frac{b v_a u}{\omega r} dr \quad [N] \tag{4}$$

In the unit of time, the wheel will remove lots of infinitesimal volumes starting from the wheel radius R to the burr height $R - h$. The total torque and the total forces required to remove all this material are obtained as a simple integration of the previous equations 3 and 4:

$$C = \int_{R-h}^{R} \frac{b v_a u}{\omega} dr \quad \rightarrow \quad C = \frac{b v_a u h}{\omega} \quad [Nm] \tag{5}$$

$$F = \int_{R-h}^{R} \frac{b v_a u}{\omega r} dr \quad \rightarrow \quad F = \frac{b v_a u}{\omega} \log\left(\frac{1}{1 - \frac{h}{R}}\right) \quad [N] \tag{6}$$

In Figure 9 the trend of Figure F while changing the burr height is shown. As the burr becomes higher, the force rapidly increases, thus transmitting high stress to the jointed structures.

Finally, it is interesting to identify the "equivalent radius" R_{eq} as the radius at which the total force F should be applied to generate the total torque C. The calculation is straightforward:

$$R_{eq} = \frac{C}{F} = \frac{h}{\log\left(\frac{1}{1 - \frac{h}{R}}\right)} \quad [m] \tag{7}$$

The result is shown in Figure 10: the effect of the distributed forces of deburring is equivalent to a single force of intensity F always

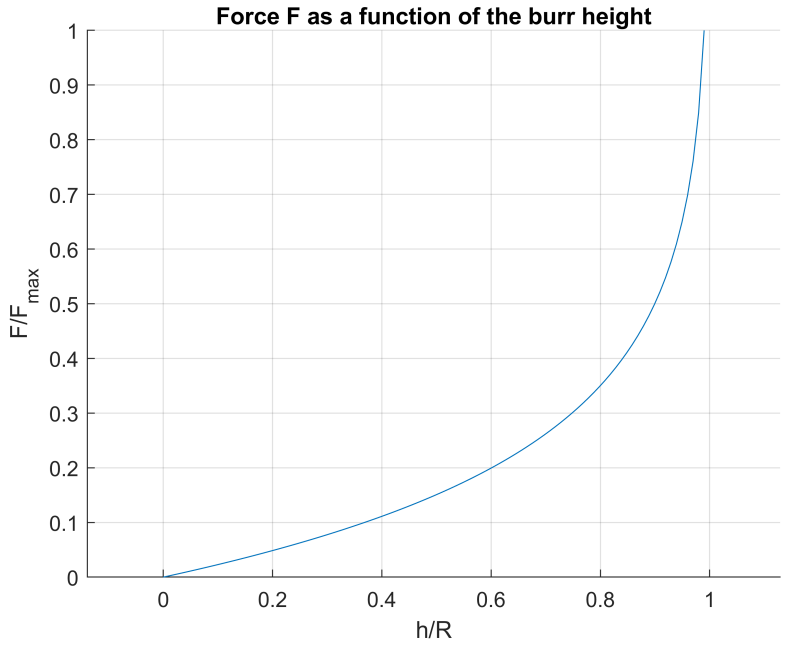


Figure 9: The deburring force F varies non-linearly as the burr height increases.

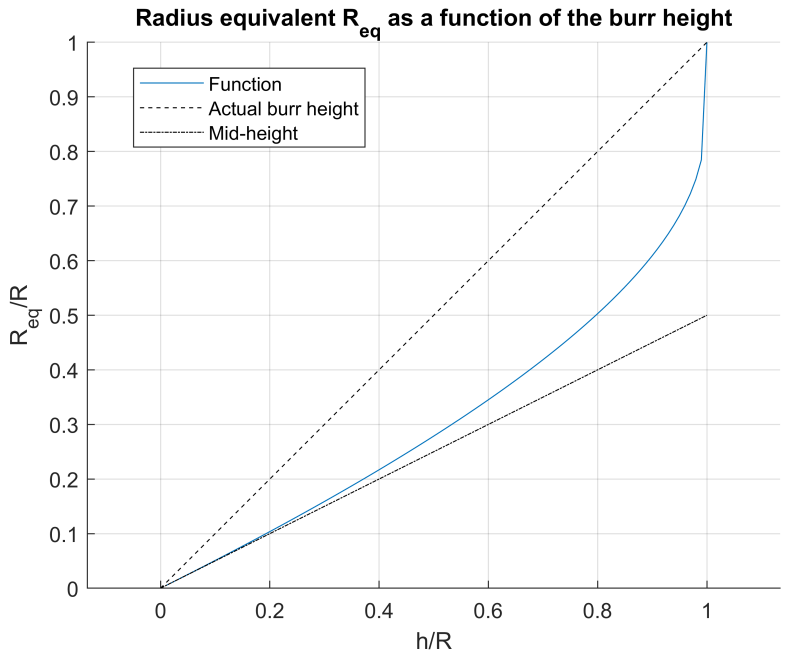


Figure 10: The equivalent radius R_{eq} is always placed within the upper part of the burr.

placed in the upper half of the burr. The closer the burr height to the center of the wheel, the higher the force.

In the design of the task, this variation of the force should be considered. There are three possible solutions to this problem:

1. It is possible to reduce movement speed v_a, thus increasing cycle time.
2. It is possible to increase the amount of material to leave after each single material wiping, thus increasing the number of wipes and the cycle time.
3. Design a compliant system that is able to adapt to rapid changes in the burr height.

This last option seems to be the most suitable, and could be performed by underactuated systems. A analysis of this option is described in Chapter 5 as the design of an under-actuated system.

3

TRAJECTORY OPTIMIZATION

This chapter presents the novelty studies on new collision avoidance systems that can be used to move a redundant robot between positions through a safe path that ensures no collisions between the robot and the environment. Two different methods are presented. To further improve the work, a comparison between the method is presented.

3.1 HOW CAN CYCLE TIME BE IMPROVED?

In order to increase the productivity of an industrial robot work cell, the cycle time has to be reduced. Sometimes the time needed to perform the tasks is fixed, so it is necessary to reduce the time to move the robot between the working positions. This can be achieved either by defining the best equation of motion that the robot's motors have to perform, and, if possible, by finding the optimal path that reduces movement time. However, the optimization of the motor equations of motion is already implemented into the controllers of common industrial robots. The only available option is to define an optimal path that the robot has to follow.

As stated before, in some applications, such as deburring, the task time is fixed by the process, so it is not possible to reduce it. However, the symmetry of the task makes it possible to achieve the same result with different robot configurations, so moving from a grinding spot to another is influenced by the starting and ending joint angles. This can lead to different cycle times that can be very far from the optimal one if defined by an inexperienced human operator.

Moreover, the movement between two working positions has to be completed without colliding with the workcell environment. The workpiece is not the only obstacle inside the workspace since there can be structures, conveyors, and barriers that do not have to be touched by the robot.

Another source of wasted time is the task planning: the working positions can be reached in different sequences, and these sequences can affect cycle time, due to different movement times and other possible sources of delays (such as tool change). This aspect will be considered in Chapter 4.

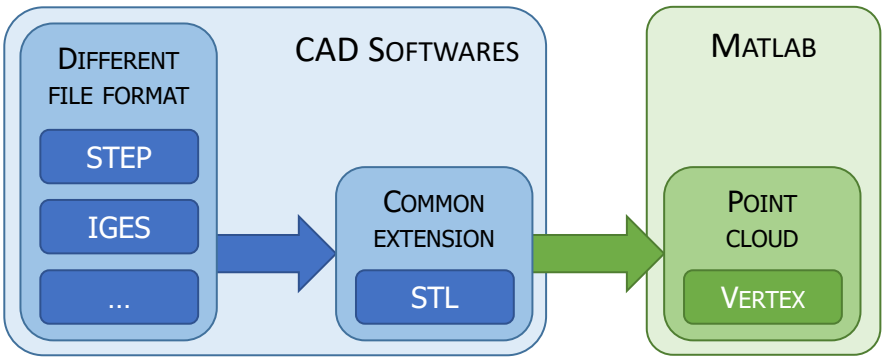


Figure 11: The source 3D CAD file is transformed to a STL file which will be imported in Matlab for the simulation.

3.2 THE SIMULATION ENVIRONMENT

Different robot models are made of links of different shapes and dimensions. It is impossible to simplify every serial robot with a generic model to obtain a safe path in a cluttered environment.

As a result, it is necessary to know the exact dimensions of a robot to provide a good result. This objective can be achieved by using 3D CAD models of the robots.

Different neutral file format are provided by the most common robot manufacturers, e.g. STEP [96] and IGES [97], but the simplest file format that can be used is the STL [98].

This format is particularly efficient since it is able to discretize a complex surface, such as a link, into small planar triangles defined by their vertexes (the so-called *patch*). Each vertex is identified by its Cartesian coordinates (x, y and z) with respect to the file reference frame. All the vertexes shared between multiple triangles are considered as a single one, thus reducing the computational effort of future calculations. Moreover, each triangle is created such as the unit vector normal to the plane points towards the space external to the link.

Finally, it is possible to move easily between STEP and IGES formats and STL, and STL can be easily read by any programming language (Figure 11).

The point cloud obtained from the STL file can be moved within the 3D space simply by multiplying any transformation matrix $\mathbf{T}$ (4×4) to the matrix containing all the vector data $\mathbf{P}$ ($N \times 4$, with N the number of vectors of the point cloud). This is very important since the point cloud position can be defined in every time instant t_i:

$$\mathbf{P}(t_i) = \mathbf{T}(t_i) \cdot \mathbf{P} \tag{8}$$

Every 3D CAD object included in this dissertation will be imported from an STL file in the simulation environment. All the robot link STL files are extracted from the 3D CAD file provided by robot manufac-

turers and adapted such as their reference frame is compliant to the Denavit-Hartenberg convention [99].

The collision detection method to be used in the dissertation has been developed during this Ph.D. program and is described in Appendix A.1. The choice of moving this part of the work to the appendix is made in order to reduce possible confusion with the following methods.

3.3 COLLISION AVOIDANCE - GRAPH METHOD

In a generic movement, the robot moves through several via points from one position to another. This is a trivial process, and usually do not involve any optimizations: a safe point in space is stored within the robot controller and is used as a pass-through to reach the following position.

This concept can be extended to create a grid of safe via points around the workpiece that can be used in any circumstance to move between the working positions.

Let A and B be two given positions on a workpiece. Such positions are given in the Cartesian space and define the position and orientation of the end effector at the working point. To avoid collisions between the robot and the workpiece or environment, moving from A to B usually requires to pass through several via points. My idea is to create a grid of safe via points in the Cartesian space, that the robot can use to move between the given positions, defining a suboptimal safe path between A and B. The final objective is to minimize the movement time between A and B. In this method, the via points describe the position of the wrist center, therefore their location influence the first three joints of the robot only. In this way, the path planning problem is decoupled from a kinematic point of view, in the sense that the motion of the first three joints is calculated and optimized regardless wrist initial and final rotations. As a result:

- the definition of the via points around the workpiece is simplified: in fact, they are 3D points (3 variables) instead of robot configurations (N variables);
- the rotations of joints 4, 5 and 6 are minimized, since such joints are simply moved from the initial to the final values; in fact, they are not involved in the definition of the safe path surrounding the workpiece, so their rotations at each via point can be chosen in the most convenient way;
- the rotations of joints 1, 2 and 3 may be not minimized, since the via points are at a non-negligible distance from the workpiece.

The main steps of the method (Figure 12), explained for a 6 DOF manipulator performing a redundant task around the tool axis (with redundant joint angle θ_7), are:

1. place the devices in the work cell (the robot, the workpiece, and auxiliary systems) in the desired positions (or by finding the optimal positions [52]);
2. define a safe volume around the workpiece (the SSV shown in green in Figure 13);
3. create a cloud of via points for the wrist center around the safe volume (Section 3.3.1), whose distance from the safe volume is equal to the distance between tool tip and wrist center; such points are safe in the sense that, if the wrist center lies in one of the points, the tool cannot collide with the workpiece for any joint 4, 5 and 6 values;
4. check if the via points are reachable by the robot from a kinematic point of view: if a collision occurs between the robot and the environment at some via points, they are removed from the cloud;
5. connect the via points to the 8 closest ones (in the Cartesian space) to form the branches that will be used in the search of the minimum time path;
6. check the connections between the via points: if a collision occurs along the point to point motion between two connected via points, the connection is removed;
7. choose the initial θ_7 values for starting and ending positions A and B ($\theta_{7,A}$ and $\theta_{7,B}$);
8. find the suboptimal path that connects A and B without collision using the Dijkstra algorithm to solve the graph whose nodes are the connected via points;
9. change $\theta_{7,A}$ and $\theta_{7,B}$ within a fixed grid around the initial values and loop from point 8 until all possible combinations are evaluated;
10. find the combination $\theta_{7,A,opt}$ and $\theta_{7,B,opt}$ which yields the minimum time motion between A and B on the chosen grid.

The points from 8 to 10 can be iterated by refining the grid around the final point, until a certain condition is met (e.g., a maximum number of iterations). At the end of iterations, the final values of redundant joint angles ($\theta_{7,A,opt}$ and $\theta_{7,B,opt}$) are stored, and the corresponding graph is considered as the optimal path that connects A and B.

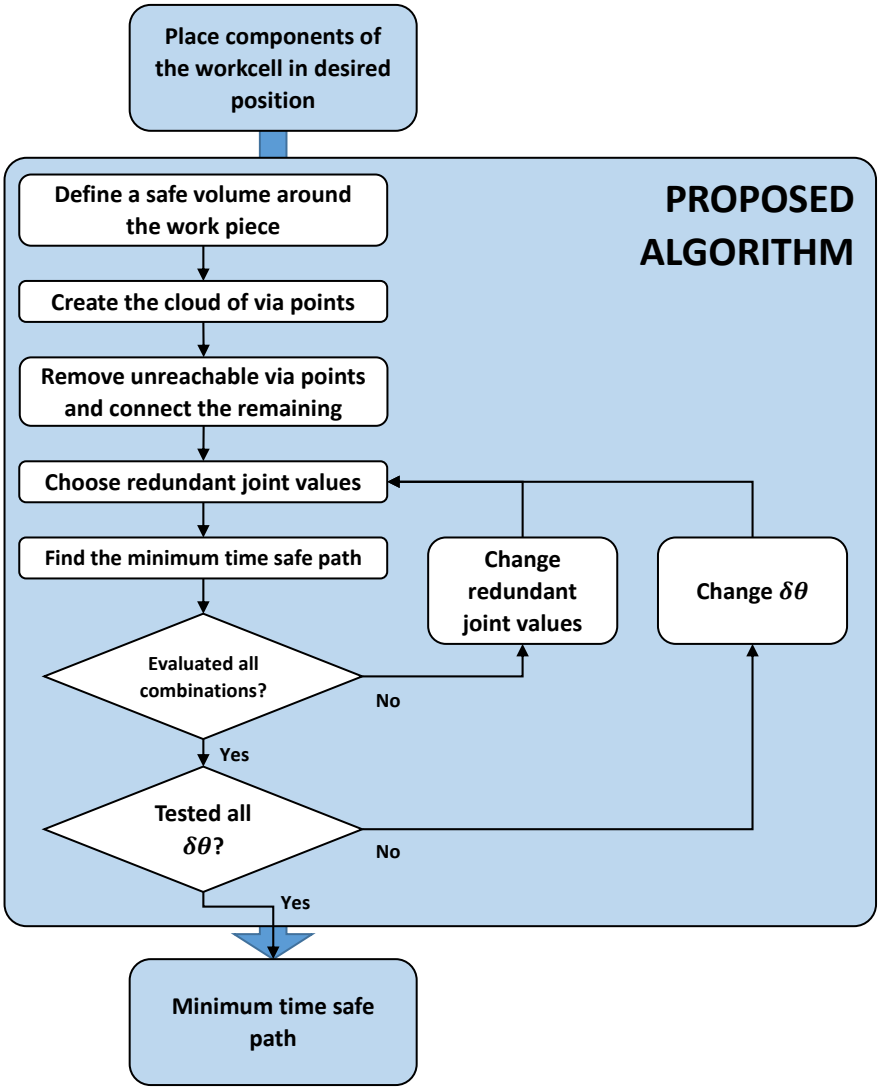


Figure 12: An overview of the proposed algorithm. At first, the operator has to place all the objects in the workspace. Then, the algorithm provides automatically the path between the working positions.

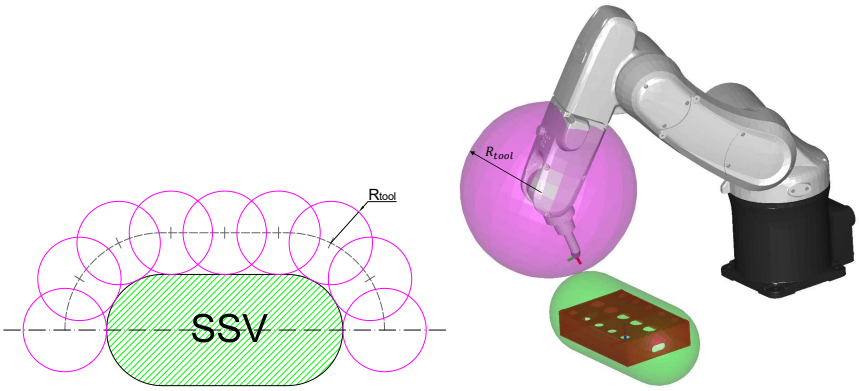


Figure 13: A simple scheme that shows the placement of the via points on a section of the SSV (to the left). Each section is then repeated many times around the symmetry axis of the SSV. All the via points are placed at a distance R_{tool} from the surface of the SSV so, regardless the orientation of the wrist, the end effector will not collide with the work piece (to the right).

3.3.1 *Via Points Generation and Selection*

A line Swept Sphere Volume (SSV) [27] is used to encapsulate the workpiece. The shape and orientation of the SSV are chosen in a way that minimizes its volume.

Joints 1, 2 and 3 describe the final position of the wrist center, while the other joints define the orientation of the robot positioning. To keep the safe via points independent from θ_7, the via points describe the wrist center position during the movement.

Since the configuration of the tool is unknown at prior, to be sure that the tool does not collide with the workpiece, we choose to place the wrist of the robot at a distance greater than R_{tool} from the SSV, where R_{tool} is the minimum radius of a sphere centered in wrist center and containing the tool. By using this criterion, a net of wrist center points equally distributed is placed on a surface at a distance of R_{tool} from the SSV (Figure 13).

To create the point could, a section of the SSV, containing the SSV symmetry axis, is considered (Figure 13 to the left). On this plane, m_1 points are evenly placed at a distance R_{tool} from the border of the SSV. Then, the section is cloned m_2 times around the SSV symmetry axis. The first and last points on each section that lie on the symmetry axis are considered once, so the total number of points is:

$$N_p = m_2(m_1 - 2) + 2 \tag{9}$$

The density of the net is a design choice (m_1 and m_2) and has a direct effect on the performance of the Dijkstra algorithm [100]. Each point of the net is connected to its 8 neighbours (Figure 14), similarly to the Uniform Space Sampling method described in [101].

The proposed definition of via points offers two benefits:

1. when the wrist center is in such points, the robot is in a safe position regardless the orientation of the tool. In this way, if we include one of the via points in a motion between the starting and the ending points, such via point will be safe for any configuration of the redundant axis in the given points. Thus, by creating a path which includes the given points and a subset of the calculated via points, such path will be safe in all intermediate points for any choice of the initial and final configurations;

2. since the via points are wrist center points, the inverse kinematic problem can be solved for the first three joints only (which is faster than solving the full problem), and an estimation of motion time between connected points can be done by considering only such joints (Section 3.3.2, Equation 10), thus drastically reducing computational time[1]. In fact, the estimated motion time on a given path depends only on the points chosen, regardless the initial, pass-through and final orientations of the tool. In this way, the time needed to move between different initial and final configurations of the robot can be evaluated without the need for re-calculating the inverse kinematics in the via points every time.

Whilst joints 4, 5 and 6 will never collide with the workpiece when the wrist center is placed in one of the via points, joints 1, 2 and 3 may collide with the workpiece and the environment. As a result, via points are checked for collision between the robot and the workpiece and the environment. If a collision is detected, the corresponding via point is deleted from the list. The collision test is performed as described in Appendix A.1: all the links are encapsulated inside SSVs, whilst all other objects are encapsulated inside Oriented Bounding Boxes (OBB [29]). In this way, it is possible to compute the interaction between the robot and the environment faster than with the usage of SSVs only [102].

3.3.2 *Graph*

The suboptimal path between two positions A and B is created using the Dijkstra algorithm [49]. To solve the algorithm, the branches and the nodes must be defined.

1 To test the reduction of the computational time due to the decoupling of the inverse kinematic problem, a simple test has been performed: the same position is used to calculate the entire configuration of the robot and only the first three joint values 100.000 times. The overall computational time for the whole inverse kinematic problem is 1.66s, whilst the computational time required for the calculation of the first three joints is 1.06s. As a result, the decoupling leads to a reduction of the computational time of the inverse kinematic problem of around 36%.

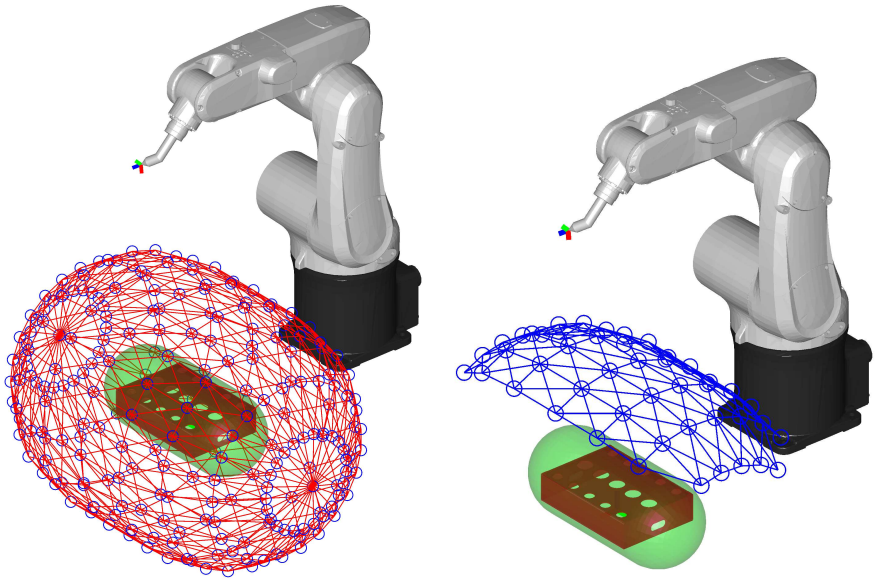


Figure 14: All the adjacent via points are connected to form the branches (to the left) and, then, the unreachable via points and the ones that provide collision are removed (to the right).

While the via points represent the nodes, the branches are represented by the connections between the via points. Considering the net of reachable via points, only adjacent via points (in Cartesian space) are connected to form branches (Figure 14 to the left), so that the total number of branches is reduced. To avoid unnecessary calculations, collision tests along the motions through all the branches are computed. If a collision is detected, the corresponding branch is removed (Figure 14 to the right).

Branches weight equals robot movement time between the nodes. Movement time between nodes h and k is estimated considering only the first three joints:

$$T_{hk} = \max_{j=1,2,3} \left\{ \frac{|\Delta q_{j,hk}|}{\dot{q}_{max,j}} c_v \right\} \tag{10}$$

where $\Delta q_{j,hk}$ is total rotation of joint j between the nodes h and k, $\dot{q}_{max,j}$ is the maximum speed of joint j and c_v is the velocity coefficient of the motion law [103].

Starting and ending nodes are defined by points A and B; A and B are connected to all the via points to form additional branches. In this way, it is possible to enter (from A) and exit (towards B) the cloud of via points from any one of the via points. The weights of the additional branches are calculated as the traveling times (as of Equation 10), and the connections are checked to find collisions: the branches that provide collision are removed. The Dijkstra algorithm calculates the graph using A as the first node and B ad the last one.

The first $\theta_{7,A}$ and $\theta_{7,B}$ values are null to start the optimization algorithm around the position provided by the user. To find the optimal solution, the redundant joint angles θ_7 at A and B are changed by fixed steps ($\delta\theta$), from minimum values ($\theta_{7,i,min}$) to the maximum ones ($\theta_{7,i,max}$). In this way, a fixed grid of possible combinations is created; the limits of the grid can be different for A and B (see Table 2 for an example).

To ensure that the calculated paths are feasible, a collision test along each suboptimal path is computed. To do so, the planning of robot motion is performed along the whole path, including wrist rotations. The angles of the first three joints are interpolated from A values to B values considering also the via points included in the path. The angles of the last three joints, on the other hand, are interpolated from A values to B values only, since their values are not assigned at the via points. If a collision is found, the graph is recalculated. This may occur due to a possible colliding configuration of the robot provided by the values of joints 4, 5 and 6 along one of the connections; in fact, wrist rotations have not been considered in the previous collision tests, and whilst the via points are safe regardless wrist orientation, connecting paths (especially from A to cloud and from cloud to B) may not.

The final movement time of the path is given by the following expression:

$$T_{final}(\theta_{7,A},\theta_{7,B}) = \max\left\{T_{Dijkstra}(\theta_{7,A},\theta_{7,B}), \max_{j=4,5,6}\left\{\frac{|\Delta q_{j,AB}(\theta_{7,A},\theta_{7,B})|}{\dot{q}_{max,j}}c_v\right\}\right\} \tag{11}$$

where $T_{Dijkstra}(\theta_{7,A},\theta_{7,B})$ is the minimum time yielded by the Dijkstra algorithm and $\Delta q_{j,AB}(\theta_{7,A},\theta_{7,B})$ is the total rotation of joint j between points A and B.

At the end of an iteration step, all the possible combinations of $\theta_{7,A}$ and $\theta_{7,B}$ are compared. The best combination of those two values, i.e., the one that minimizes T_{final}, provides $\theta_{7,A,opt}$ and $\theta_{7,B,opt}$ that are to be used in the next iteration step. To start the next iteration step, the value of $\delta\theta$ is reduced and the next boundaries of $\theta_{7,A}$ and $\theta_{7,B}$ are changed. The lower $\delta\theta$, the better the precision of the optimal position. In the first iteration $\theta_{7,A,min} = \theta_{7,B,min} = -180^\circ$ and $\theta_{7,A,max} = \theta_{7,B,max} = 180^\circ$ to evaluate the entire workspace (Table 1).

At the end of the procedure, the optimal couple of redundant joint angles $\theta_{7,A,opt}$ and $\theta_{7,B,opt}$ is provided.

The optimal solutions are not to be intended as the globally optimal solutions to the problem. Since this is a non linear stochastic problem that has to face lots of constraints (collision avoidance, movement between the via points, discretization of the values of the redundant joint angle) the purpose of the algorithm is to find a local minima

$\delta\theta$ [°]	60
$\theta_{7,A}$ values [°]	[-180,-120,-60,0,60,120,180]
$\theta_{7,B}$ values [°]	[-180,-120,-60,0,60,120,180]
Number of combinations	49

Table 1: Values of $\theta_{7,A}$ and $\theta_{7,B}$ when the first iteration step has $\delta\theta = 60°$.

Step	$\delta\theta$ [°]	A bounds [°] $[\theta_{7,A,min}, \theta_{7,A,max}]$	B bounds [°] $[\theta_{7,B,min}, \theta_{7,A,max}]$	$\theta_{7,A,opt}$ [°]	$\theta_{7,B,opt}$ [°]
1	60	[-180, 180]	[-180, 180]	0	60
2	45	[-135, 135]	[-75, 195]	0	60
3	30	[-120, 120]	[-60, 180]	0	30
4	20	[-80, 80]	[-50, 110]	0	10
5	10	[-40, 40]	[-30, 50]	-10	20
6	5	[-30, 10]	[0, 40]	-15	20
7	2	[-23, -7]	[12, 28]	-15	22
8	1	[-19, -11]	[18, 26]	-14	23

Table 2: An example of iterations changing $\delta\theta$ from 60° to 1°. Such values were used to run the algorithm in Test 1.

of the problem, which, however, can be good enough to satisfy the majority of the industrial scenarios.

3.4 VALIDATION

A Matlab® script has been created through which an ADEPT VIPER s650 6-axis robot is simulated (Figure 13) moving from one working position to another. The PC used for the simulation is the same as of Section A.1.

The robot is moved from one side of the workpiece to another so that the direct movement is impossible (the robot would collide with the workpiece).

The starting position is P_A (490,167.5,18.4) with the orientation defined by Cardan angles $\{x, y, z\} = \{0, 90, -90\}$. The ending position is P_B (410,-67.5,48.4) with the orientation defined by Cardan angles $\{x, y, z\} = \{0, 90, 90\}$. Robot base frame is coincident with the global reference frame. The parameters of the planes defining the workpiece's OBB (Equation 63), used by the collision detection algorithm, are shown in Table 3. Moreover, another plane has been added to simulate the table where the workpiece is placed. The parameters describing this plane are in the seventh column of Table 3.

Plane n°	1	2	3	4	5	6	7
a	1	-1	0	0	0	0	0
b	0	0	1	-1	0	0	0
c	0	0	0	0	1	-1	-1
λ [mm]	-368	532	67	167	1.6	55.4	1.6

Table 3: Parameters describing the 6 planes that define workpiece OBB (columns 1-6) and the extra plane describing the table where the workpiece is placed (column 7).

Step	$\delta\theta$	Lowest movement time [s]	Comparison to Step 1 [%]	Total computational time [s]
1	60	0.6035	100	1.77
2	45	0.6035	100	3.47
3	30	0.6016	99.7	6.54
4	20	0.5672	94.0	10.42
5	10	0.4893	81.1	14.35
6	5	0.4698	77.8	17.89
7	2	0.4689	77.7	21.37
8	1	0.4655	77.1	24.88

Table 4: Results of the test changing $\delta\theta$ from 60° to 1°.

The total net of via points is made of 227 points ($m_1 = 11$, $m_2 = 25$), while the number of feasible viapoints is 56 and the number of via points connected to the graph is 46 (Figure 14 to the right).

In this test I set the range limits of θ_7 angles to ± 3 times $\delta\theta$ for the first two steps and ± 4 times $\delta\theta$ for the other steps, see Table 2. This ensures a good trade-off between algorithm performance and computational time, since for each iteration it limits the number of computations of the graph (Figure 16).

At first, the lowest movement time at each iteration step is evaluated. In Figure 15 are shown the results: the lowest movement time decreases from the highest $\delta\theta$ (60°) to the lowest one (1°). In this case the reduction of the movement time from the higher value of $\delta\theta$ to the lowest one is of nearly 23% (Table 4). The computer took 24.88s to find out $\theta_{7,A,opt}$ and $\theta_{7,B,opt}$ with 8 steps. Best combinations at each step can be seen in Table 2.

Figure 16 shows the calculation times and the number of possible combinations of each $\delta\theta$ step. The calculation times nearly follow the trend of the number of combinations, is capped to the maximum of 81 starting from the third step. Then, at small $\delta\theta$ values the calculation times slightly drop due to a smaller amount of colliding paths: small

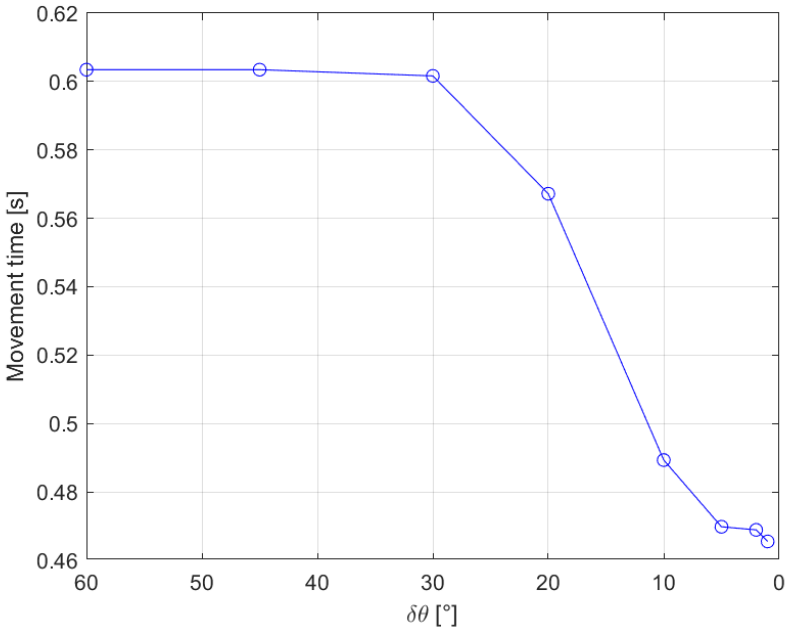


Figure 15: Minimum of moving total time for each step value. x-axis direction has been reversed to show the iteration process from left to right.

Step	δθ	Lowest movement time [s]	Comparison to Step 1 [%]	Total computational time [s]
1	30	0.6016	100	5.05
2	5	0.4698	78.1	11.41

Table 5: Results of the test changing δθ from 30° to 5°.

variations of $\theta_{7,A}$ and $\theta_{7,B}$ around the optimal solution usually do not lead to lots of path collisions.

Even if a single calculation of 24.88s reduces the movement time of nearly 23%, a similar result can be obtained using a reduced number of steps. I performed another test with the same starting and ending positions, but with only two iteration steps: $\delta\theta_1 = 30^\circ$ and $\delta\theta_2 = 5^\circ$. The results (Table 5) show that with reduced computational times (only 11.41s) the lowest movement time is nearly the same as in the previous test.

In Figure 17 the final movement provided by the algorithm with the parameters of Table 5 is shown. The shape of the path depends on the density of the via points defined in the workspace.

3.4.1 *Comparison with PRM*

A comparison of the proposed method with the Probabilistic Roadmap Method (PRM) has been performed, to compare the solutions ob-

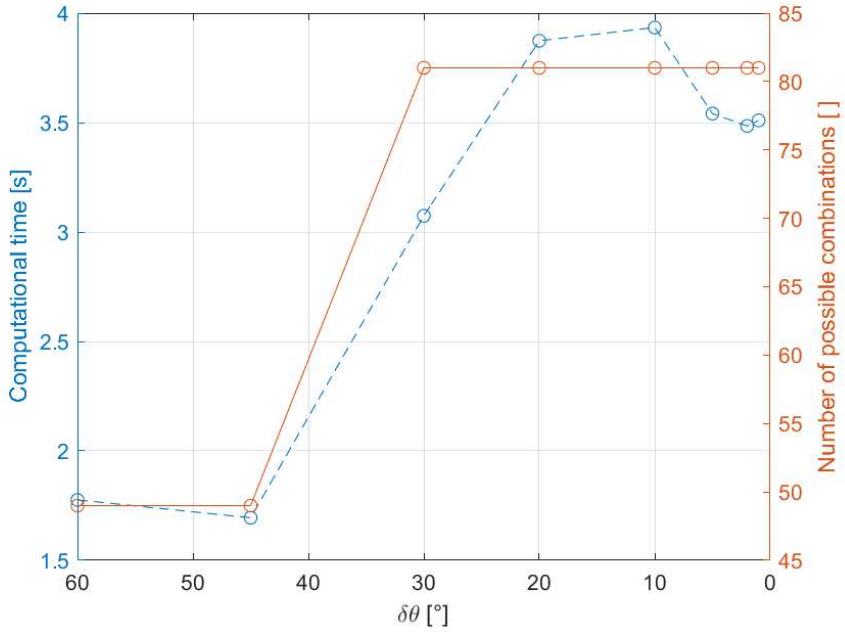


Figure 16: Computational time of the definition of the graph for each step value. The computational time nearly follows the number of possible combinations. x-axis direction has been reversed to show the iteration process from left to right.

tained and the computational times. The PRM is a sampling-based method that uses the Dijkstra algorithm to connect two points defined in Configuration space by using randomly generated via points defined in Configuration space. The main differences with the proposed method are that:

- the via points are defined randomly (i.e., without considering workpiece actual encumbrance);
- wrist rotations are assigned at the via points, and this will force the wrist joints to make wider rotations along the path with respect to the proposed method.

In the comparison, the PRM has been set as follows:

- the sampling of the Configuration space of the robot is performed by randomly choosing 227 configurations (equal to the number of via points of my point cloud);
- the Configuration space is searched within a limited range of joints 1, 2 and 3 rotations to reduce the dispersion of the randomly generated points (which would result in worse performance of the PRM algorithm); joints 4, 5 and 6 ranges were not limited, to avoid loosing dexterity;
- each configuration is connected to the nearest 8 configurations (in Configuration space).

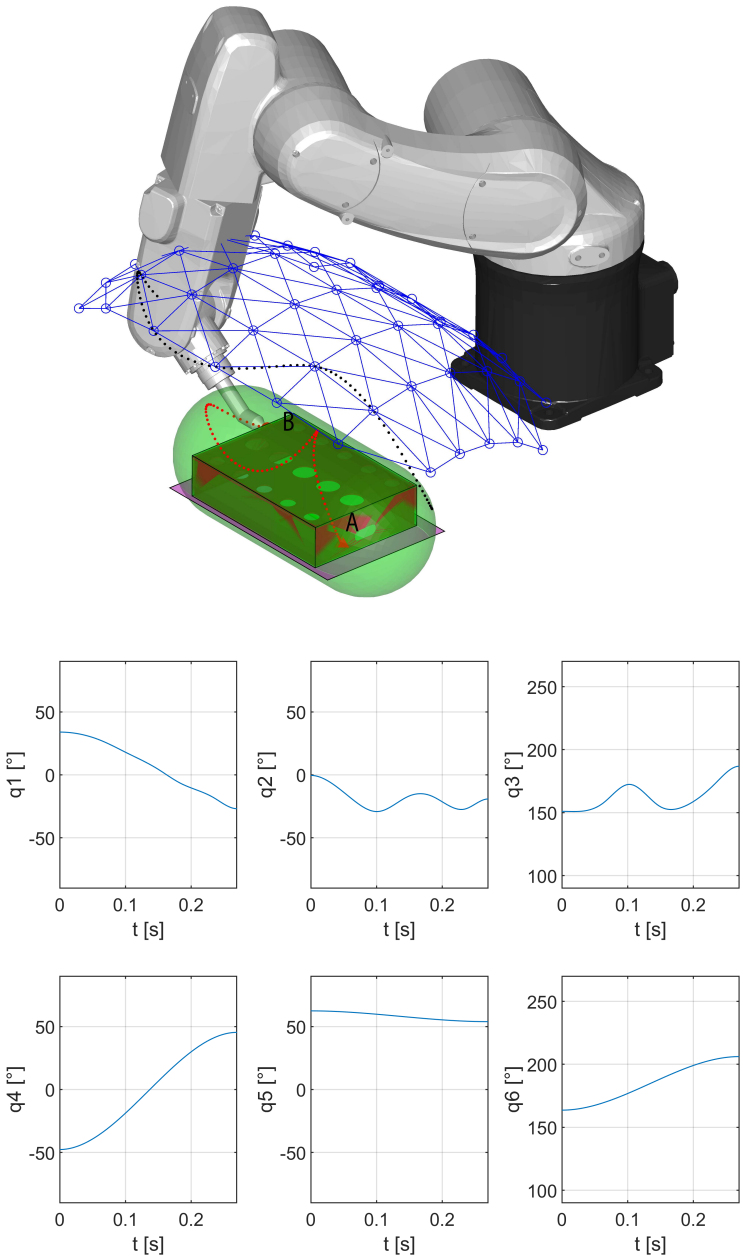


Figure 17: To the top: the final movement provided by the algorithm for Test 1, with the red dots that describe the end effector trajectory and the black dots that describe the wrist center trajectory. To the bottom: corresponding joint angles versus time.

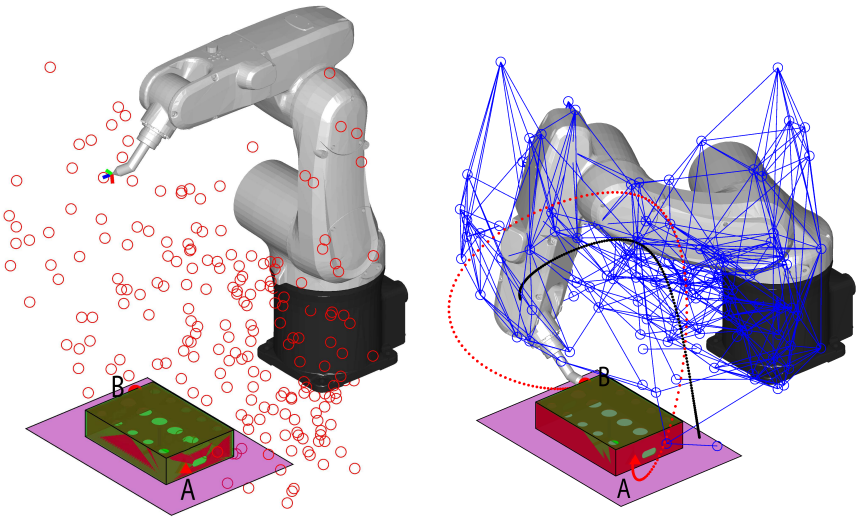


Figure 18: The wrist center points provided by PRM (to the left) and the corresponding final movement provided for Test 1, with the red dots that describe the end effector trajectory and the black dots that describe the wrist center trajectory (to the right). Blue circles are feasible and connected wrist center points.

With the aim of fitting the PRM to the method described in Section 3.3, the latter has been modified as follows:

- point 2 has been eliminated;
- point 3 has been modified since the via points are provided by randomly generated samples in the 6-dimensions Configuration space;
- point 5 has been modified since the via points are connected to form branches by finding the 8 closest points in Configuration space.

All the other points of the method are kept unchanged.

Since in the PRM the via points are randomly generated and may yield variable results, 400 instances of the PRM have been performed (with two $\delta\theta$ steps as of Table 5). The simulations show a mean computational time of 21.2s (standard deviation of 3.38s) and a mean robot movement time of 0.6915s (standard deviation of 0.2177s). An example of PRM solution is shown in Figure 18, where among 227 random samples only 133 via points are reachable and connected to the graph. In this particular case, both computational time and mean movement times are greater than with my method.

This is due to the sparse location of the via points in the PRM, most of which are not useful in the definition of a path with a low movement time. Moreover, the search for the suboptimal path is performed within a 6-dimensional space in the PRM (the Configuration

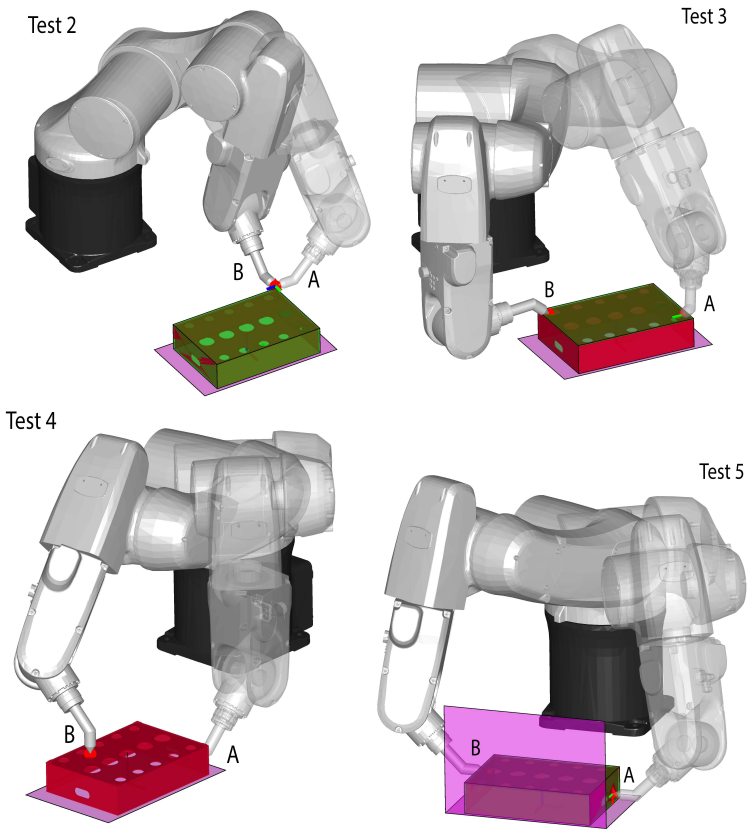


Figure 19: Starting (A) and ending (B) robot configurations of Test 2 to 5; redundant angles are null in all the positions depicted ($\theta_{7,A} = 0$ and $\theta_{7,B} = 0$).

space), while the decoupling of the inverse kinematic problem in the proposed approach allows to search a 3-dimensional space and then simply interpolate the start and end values of the last three joints, thus reducing the computational effort [104].

To better understand the difference between the proposed method and the PRM, five different cases have been considered, in which the starting/ending positions and/or the workcell layout have been changed: Test 1 is the same of Figure 17; Test 2 and Test 3 are very simple movements (Test 2 is a simple rotation around end effector position, Test 3 is a tool repositioning on the same surface on top of the work piece); Test 4 includes a tool repositioning from one side to the top of the workpiece; Test 5 includes the same movement of Test 1 with an additional obstacle to constrain the movement.

Figure 20 shows optimal movement times provided by the proposed method and by the PRM in the five tests (for the PRM, which was repeated 10 times for each test, minimum/maximum and mean values are depicted). Except from Test 2 (simple tool repositioning), where results are comparable, the proposed method outperforms the PRM in terms of minimization of robot movement time. It may be

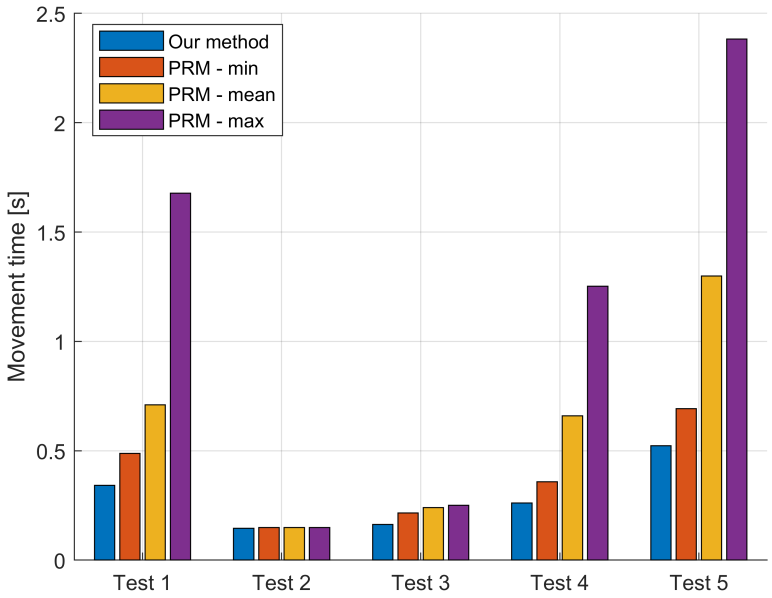


Figure 20: Optimized movement times retrieved from the proposed method and from PRM. Since PRM planning has been repeated 10 times for each test, the chart shows minimum, mean and maximum optimal times from PRM for each test.

that the PRM would perform better if the number of the generated via points are increased, however this would dramatically increase computational time.

Figure 21 shows the total absolute joint displacements along the optimal path provided by the proposed method and by the PRM in the five tests (for the PRM, which was repeated 10 times for each test, minimum/maximum and mean values are depicted). Cumulative displacement of the first three joints, which account for wrist center motion, and of the last three joints, which account for wrist rotation, are shown as well. This figure provides a possible explanation of the results shown in figure 9: the PRM tends to produce wider displacements of the last three joints (i.e., wider rotations of the wrist), since it assigns their values in the via points; as a result, the motion time increases with respect to the proposed method, where the displacement of the wrist joints is minimized (it equals the difference between initial and final rotations in A and B). Even though the proposed method may sometimes provide wider rotations of the first three joints (i.e., larger motion of the wrist center), this seems not to be the bottleneck in the tests performed.

Another interesting point is that, since the PRM is a stochastic method (in the sense that it produces a random set of via points), a single execution of it may yield very bad results in terms of robot

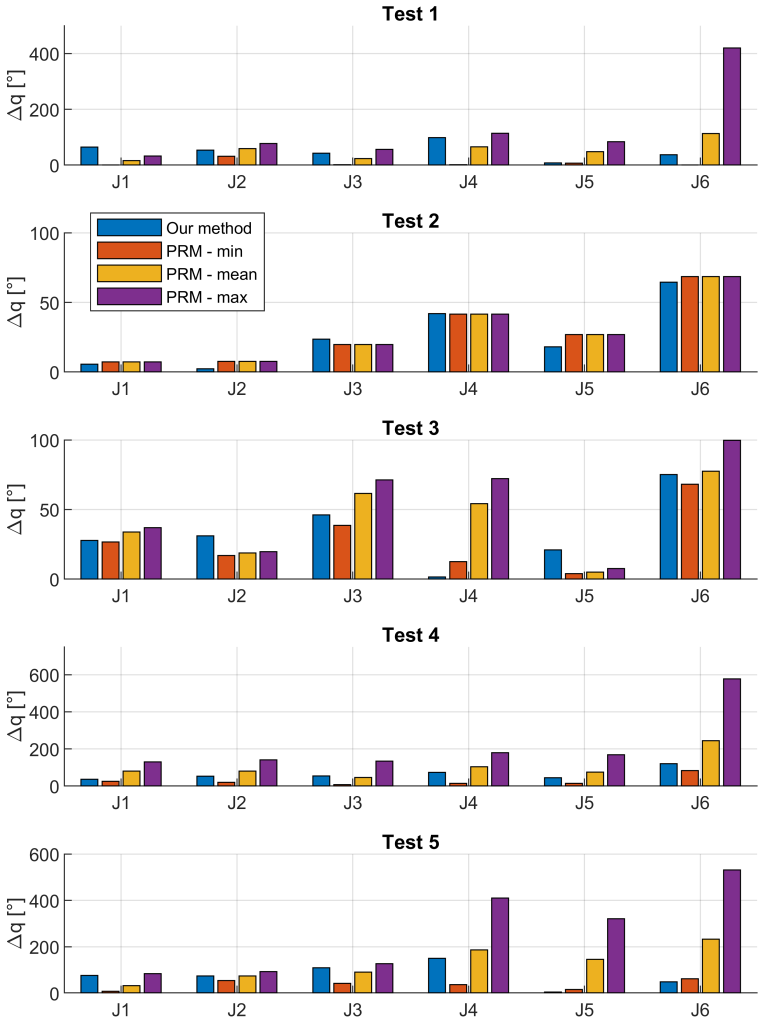


Figure 21: Joint displacement for each joint for each Test. Since PRM planning has been repeated 10 times for each test, the chart shows minimum, mean and maximum displacements from PRM for each test.

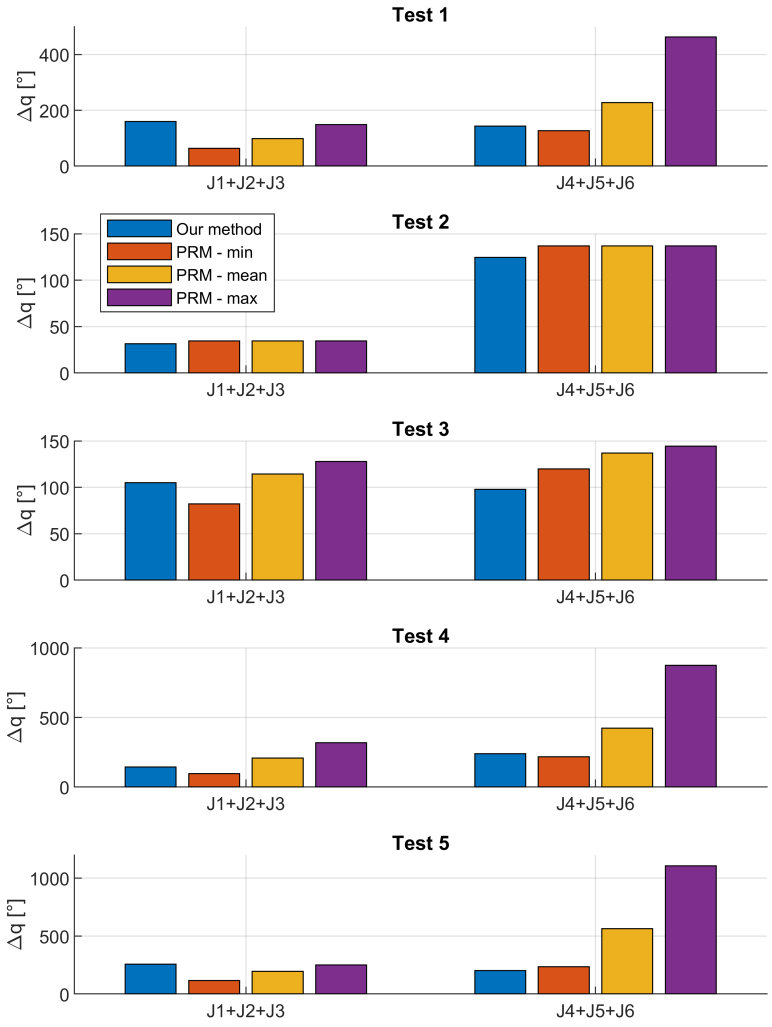


Figure 22: Sum of joint triplets for each Test. Since PRM planning has been repeated 10 times for each test, the chart shows minimum, mean and maximum displacements from PRM for each test.

movement time. On the other hand, the proposed method provides a very good solution in one single shot.

3.5 COLLISION AVOIDANCE ALGORITHM

The graph method is very suitable in a static environment, where most of the equipment is still and the net of via points can be used in any path planning. However, in many flexible applications, such as flexible feeding systems [86, 68, 105], the starting position of the movement is unknown at prior. This can bring unpredictable robot configurations during the work cycle, which in turn can lead to collisions with the surrounding obstacles in the approach or depart movements.

The only way to avoid such events is to adapt trajectory planning to the situation at hand, by using proper collision avoidance algorithms [106].

A robot position can be defined either in the joint space or in the operative space. To calculate a safe position that avoids collisions, I decided to work in the joint space, since this allows to take into account the actual working range of each joint. This is a fundamental aspect of robot positioning. In fact, a joint that is close to its limits should not (and sometimes could not) be rotated further; instead, all other joints should be used to move away from the obstacle. In this way, the positions will always be within the *reachable space*, i.e., all the movements are admissible by the robot.

Let $\mathbf{q_o}$ be the robot joint position at which the collision has been detected. The safe position can be calculated as:

$$\mathbf{q_s} = \mathbf{q_o} + \Delta\mathbf{q} \tag{12}$$

where the incremental rotation dq_i applied to the i-th joint is defined by:

$$dq_i = s_i \cdot \delta\theta_i \cdot g_i \cdot f_s \tag{13}$$

where s_i and $\delta\theta_i$ are the sign and the amplitude of the rotation; g_i is a scaling factor that takes into account the proximity of the $i-th$ joint to its limits; f_s is a global compensation factor, which is applied to all joints to balance the effects of the scaling factors on the resulting motion.

Amplitude and sign of rotation

Since the robot has to move away from the colliding object, $\delta\theta_i$ is calculated in such a way that it increases the distance of the i-th link to the nearest collision point O_b of a quantity d_i. The value of d_i

depends on the size of the link and on the safety distance to be provided. For example, a robot that could not move the first joint due to a cluttered environment could have $d_1 = 0$.

Figure 23 shows the scheme used to estimate the rotation $\delta\theta_i$. Let $O_i x_i y_i z_i$ and $O_i x'_i y'_i z'_i$ be the i-th link reference frames before and after applying the rotation, respectively (Figure 23, to the right). Let's assume, for example, that the link mainly spreads over the $x-z$ plane (Figure 23, to the left). In this case, the collision point moves away from the link when the module of the y coordinate increases[2], so it can be set:

$$|y'_{i,Ob}| - |y_{i,Ob}| \geqslant d_i \tag{14}$$

Let A be the projection of O_b on the xy plane (Figure 23, to the right); B and D be the projections of A on y_i and y'_i respectively; C be the intersection of the direction of AD with y_i. To satisfy equation 14, a good approximation of $\delta\theta_i$ and s_i is:

$$\delta\theta_i = \arctan\left(\frac{d_i}{|x_{i,Ob}|}\right) \quad \text{and} \quad s_i = -\operatorname{sgn}(x_{i,Ob} \cdot y_{i,Ob}) \tag{15}$$

In this way, BC is close to the target distance d_i. The sign of the rotation s_i depends on the quarter of the $x_i - y_i$ plane in which A falls (Figure 23, to the left, shows an example).

Some particular cases may occur: if the obstacle is placed to the side of the link ($x_{i,Ob} \to 0$), the z rotation has poor influence on the y coordinate, which in turn may yield very high values of $\delta\theta_i$ if equation 15 is used. To avoid such big unexpected rotations, a limit value

2 Clearly, a different formulation must be used when the structure of the link is different.

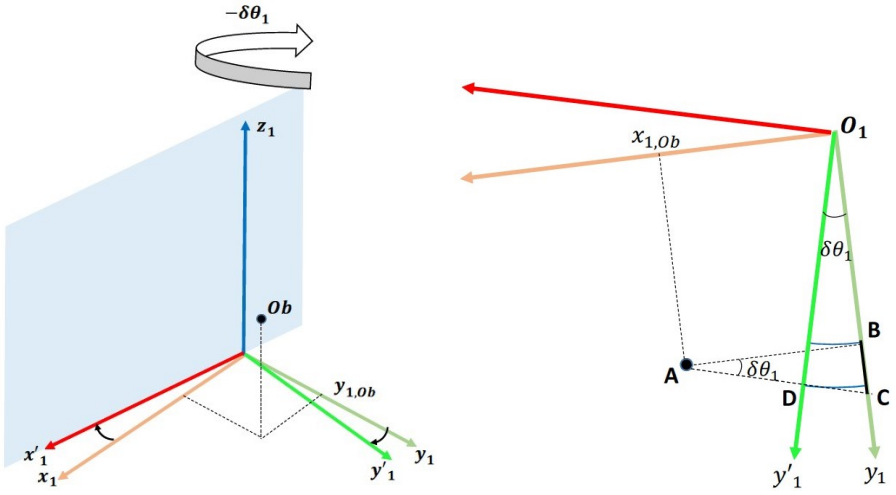


Figure 23: Schemes of the joint-1 rotation: on the left an example of sign of rotation, on the right the scheme used for the estimation of $\delta\theta_i$

$\delta\theta_{i,max}$ is chosen, and a corrective function to reduce the rotation when it overcomes the limit is applied (Figure 24, to the left):

$$\delta\theta_i = \begin{cases} \delta\theta_i & \delta\theta_i \leqslant \delta\theta_{i,max} \\ \delta\theta_{i,max} \cdot \left(\frac{90-\delta\theta_i}{90-\delta\theta_{i,max}}\right) & \delta\theta_i > \delta\theta_{i,max} \end{cases} \tag{16}$$

Figure 24 to the left shows the results of this function.

Scaling factor g_i

Each robot joint can move from a lower limit $q_{i,min}$ to an upper limit $q_{i,max}$, due to mechanical stops. With the aim of applying larger rotations to the joints that are at a greater distance from their limits, while reducing those applied to the joins that are close to the limits at the same time, a scaling factor g_i to each rotation $\delta\theta_i$ is applied. This is done by multiplying $\delta\theta_i$ by a factor g_i which depends on initial joint position $q_{i,0}$:

$$g_i = \begin{cases} 1 - k_i^8 \cdot \text{sgn}(q_{i,0} - q_{i,m}) & s_i \geqslant 0 \\ 1 - k_i^8 \cdot \text{sgn}(q_{i,m} - q_{i,0}) & s_i < 0 \end{cases} \tag{17}$$

where:

$$k_i = \frac{q_{i,0} - q_{i,m}}{q_{i,max} - q_{i,m}} \quad , \quad q_{i,m} = \frac{q_{i,min} + q_{i,max}}{2} \tag{18}$$

In this way: $k_i \to 1$ when $q_{i,0} \to q_{i,max}$; $k_i \to -1$ when $q_{i,0} \to q_{i,min}$; $g_i \in [0, 2]$, so it can amplify or reduce the final rotation based on joint proximity to the limits (see Figure 24, to the right). Not only each joint is prevented from approaching mechanical stops, but also rotations are amplified, if their direction allows to move the joint away from the stops.

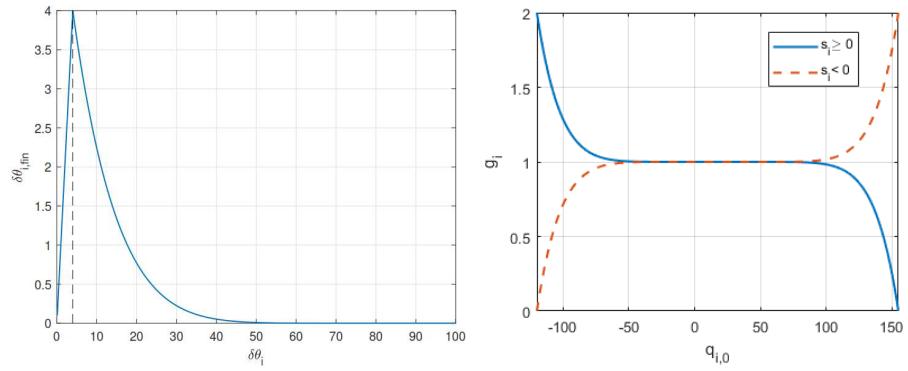


Figure 24: To the left, plot of the corrected $\delta\theta_i$, with $\delta\theta_{i,max} = 4°$. To the right, plot of scaling factor g_i, with $q_{i,min} = -125°$ and $q_{i,max} = 150°$.

Compensation factor f_s

Since by applying the scaling factors g_i the amplitude of the resulting movement could be reduced unduly, a compensation factor f_s is introduced. This parameter is always greater or equal to 1 and takes into consideration the sum of all the g_i values:

$$f_s = \begin{cases} 1 & \sum_{i \in I} g_i \geqslant n \\ \\ \frac{n}{\sum_{i \in I} g_i} & \text{otherwise} \end{cases} \tag{19}$$

where I is the set of joint considered in the algorithm, n is their number.

3.5.1 *System validation*

To validate the method, a Matlab® script has been created where a KUKA KR180 R2500 6-axis robot is simulated (Figure 25). In this case, the following considerations have been made to adapt the model: in an anthropomorphic robot, joints 2 and 3 work on parallel consecutive axis, so their behaviour can be handled together; joints 4 and 5 axes are concurrent in the wrist center, so they can be handled together.

Regarding joints 2 and 3, since they are applied on the same plane, some unnecessary movements can be avoided. In particular, if the object is placed behind the robot ($y_{2,Ob} < 0$ if $q_3 \geqslant 0$ or $y_{2,Ob} > 0$ if $q_3 \leqslant 0$) there is no need for the joint 3 to move, so $\delta\theta_3$ is set to 0. Referring to joints 4 and 5, s_4 depends on the position of the fifth link: e.g., if $y_{4,Ob} > 0$ (the point is on the upper left quadrant of x_4y_4 plane), then if $q_5 < 0$ then the robot has to rotate joint 4 clockwise ($s_4 < 0$), otherwise the rotation has to be anticlockwise ($s_4 > 0$). Moreover, if $y_{5,Ob} < 0$ (the object is behind the wrist centre) then the rotations of those joints are worthless, so $\delta\theta_4$ and $\delta\theta_5$ are set to 0. Those are particular cases that can optionally be avoided in the model estimation but makes the algorithm more precise.

The joint-6 position is not considered in the simulation since its behaviour depends on the application: a joint-6 axial symmetric end effector will not be affected by the movement of the last robot joint, while an articulated end effector could avoid the collision with proper joint positioning. This aspect can be further analyzed in specific applications. The model's parameters used for the 6-axis robot are described in Table 6. Due to Denavit-Hartenberg convention [107], some joints spread over $y-z$ planes, so x is the coordinate that defines their $\delta\theta_i$.

Table 6: Collision avoidance formulas and parameters for the KUKA KR180 R2500 robot

Joint	s_i	$\delta\theta_i$	$\delta\theta_{i,max}$	$q_{i,max}$	$q_{i,min}$
1	$-\text{sgn}(x_{1,Ob} \cdot y_{1,Ob})$	$\arctan(d_i/\lvert x_{1,Ob}\rvert)$	4°	185°	-185°
2	$-\text{sgn}(x_{2,Ob} \cdot y_{2,Ob})$	$\arctan(d_i/\lvert x_{2,Ob}\rvert)$	4°	-5°	-140°
3	$-\text{sgn}(x_{3,Ob} \cdot y_{3,Ob})$	$\arctan(d_i/\lvert y_{3,Ob}\rvert)$	4°	155°	-120°
4	$-\text{sgn}(x_{4,Ob} \cdot y_{4,Ob} \cdot q_5)$	$\arctan(d_i/\lvert x_{4,Ob}\rvert)$	4°	350°	-350°
5	$-\text{sgn}(x_{5,Ob} \cdot y_{5,Ob})$	$\arctan(d_i/\lvert y_{5,Ob}\rvert)$	4°	125°	-125°

3.5.2 *Results*

To validate my work, I move a point in space (simulating the object) and analyze the behaviour of the robot. Two tests have been executed: in the first test, the point is moved along a simple path, with $\mathbf{d} = [5\ 5\ 5\ 5\ 5\ 0]$. The path is defined by two segments connecting, in order, points $A(2200; 900; 1600)$, $B(1100; 550; 1900)$ and $C(500; 400; 2300)$. Figure 25 (on top: to the left and to the center) shows the results: the robot never gets in contact with the object (the minimum distance is always greater than 0), while the rotations applied are not high.

The second test moves the object along three different paths and saves the minimum distance of the robot to the object, changing the d_i value (equal for all joints) from 1mm to 10mm for each path (30 simulations overall). The three paths are defined by two segments connecting three points A, B and C: for the first path $A(2400; -100; 2000)$, $B(1100; 550; 1780)$ and $C(400; 900; 1900)$; for the second path $A(2100; 900; 2400)$, $B(1300; 550; 1950)$ and $C(500; 250; 1600)$; for the third path $A(2000; 100; 1300)$, $B(1000; 600; 1800)$ and $C(300; 700; 1200)$. Higher values of d_i allow to keep the object at a greater distance from the robot (Figure 25 on top, to the right), although even with low d_i values the robot is positioned at a safe distance (greater than 10cm). All the paths have been chosen such that they intersect the robot in its initial configuration $j_{1,5} = [0, -90, 90, 0, 0]$.

Computational time has been calculated as well: to compute the second test, with a total of 3900 points, the same PC used in Section A.1 took 1.915s.

This shows that the algorithm is capable of always moving away the robot from the collision point. In the next Section, an entire movement between two positions is defined using this algorithm.

3.6 COLLISION AVOIDANCE - ADAPTIVE METHOD

Let's consider a robot that has to move from a position A to a position B (see Figure 27), where A and B are defined in operational space.

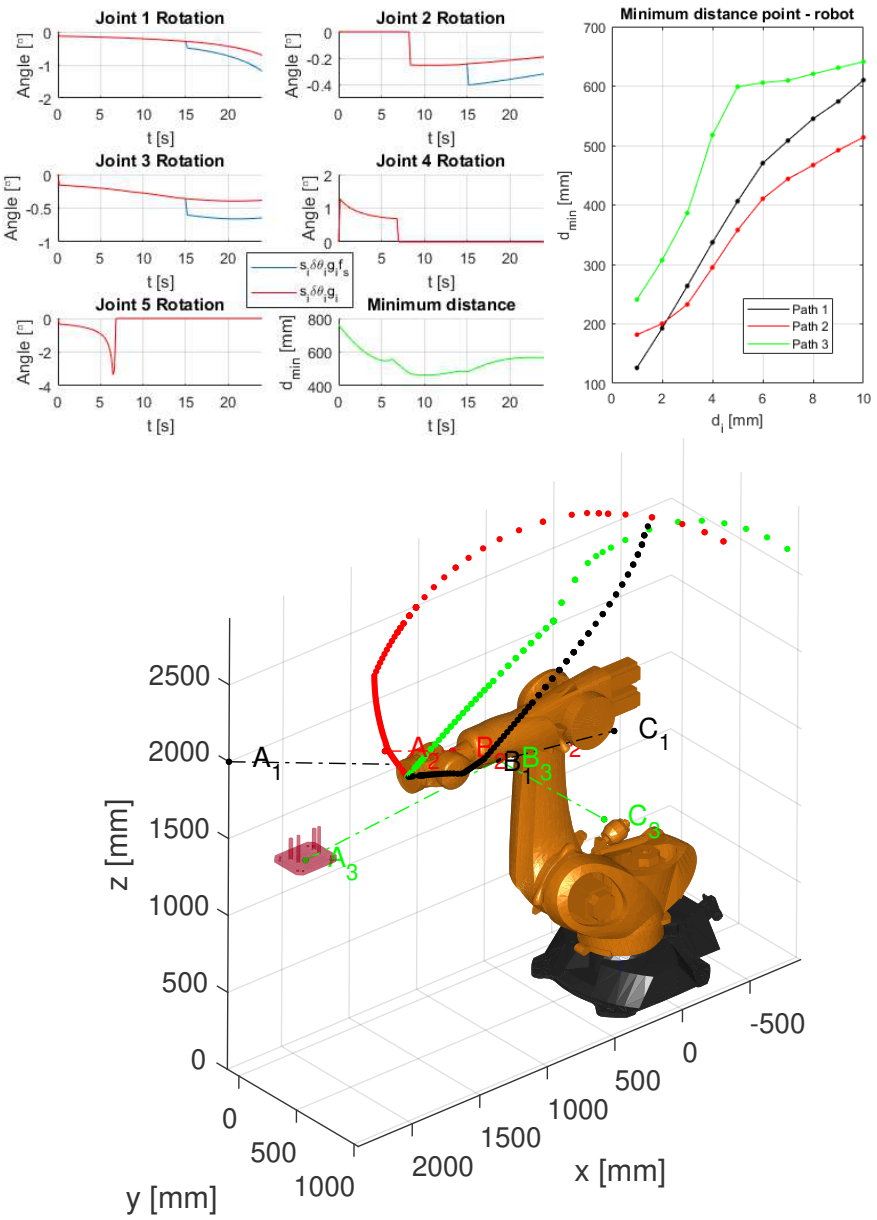


Figure 25: On top: to the left and to the center, rotations applied to the robot during obstacle movement. The red lines describe joint displacements without the compensation factor f_s, while the blue curve shows the final dq_i angle. To the right, minimum distance along three different paths for $d_i \in [1, 10]$. On bottom, the three object paths (dash-dotted lines) and the three corresponding paths described by the robot's flange (dots).

Starting from the collision avoidance algorithm proposed in the previous Section, it is to find an optimization procedure that finds the fastest path between those positions without colliding with the environment.

In the evaluation of the algorithm parameters, the computational time is considered, thus trying to find an acceptable trade-off between final robot performances and computational time.

To create the collision-free path between two positions, the proposed method follows these steps (Figure 26):

1. Place components in the work cell, including the robot and the workpiece.
2. Perform a simple optimization (Section 3.6.1): without considering possible collisions, redundant angle values at start and end positions that minimize movement times with simple joint interpolation are found. These values will be used in the following optimization step. The movement is described in joint space.
3. Optimize the previous movement computing collision detections and create safe via points.
4. Remove unnecessary via points that can complicate the path and make the movement slower (optional)

3.6.1 *First path*

The simplest path that connects these two positions is the joint interpolated movement. Many functions can interpolate joint values, and among them, the third-degree polynomial function is chosen, so that the i-th joint values can be calculated as:

$$q_i(t) = k_{1,i}t^3 + k_{2,i}t^2 + k_{3,i}t + k_{4,i} \tag{20}$$

where $i = \{1, \cdots, n\}$, with n the number of actuated joints, and $k_{1,i}$, $k_{2,i}$, $k_{3,i}$, $k_{4,i}$ are function parameters that depend on boundary conditions. The objective is to find the fastest path that connects the two positions ensuring no collisions.

Let's call t_f as the movement time length; the boundary conditions are:

$$\begin{aligned} q_i(0) = q_{i,A} \quad , \quad q_i(t_f) = q_{i,B} \\ \dot{q}_i(0) = \dot{q}_i(t_f) = 0 \quad , \quad |\dot{q}_i(t)| = \dot{q}_{i,max} \end{aligned} \tag{21}$$

with $\dot{q}_{i,max}$ as the maximum speed of i-th link.

Due to robot redundancy, there is not a unique configuration that can satisfy robot positioning in A and B.

Without any loss of generalization, let's consider the robot shown in Figure 27. By rotating around the redundant axis r by an angle $\theta_{7,B}$, robot's configuration changes but position B is satisfied. This means that $q_{i,B}$ can change, so the different boundary conditions (Equation 21) provide different parameters in Equation 20. This leads to different movement times t_f.

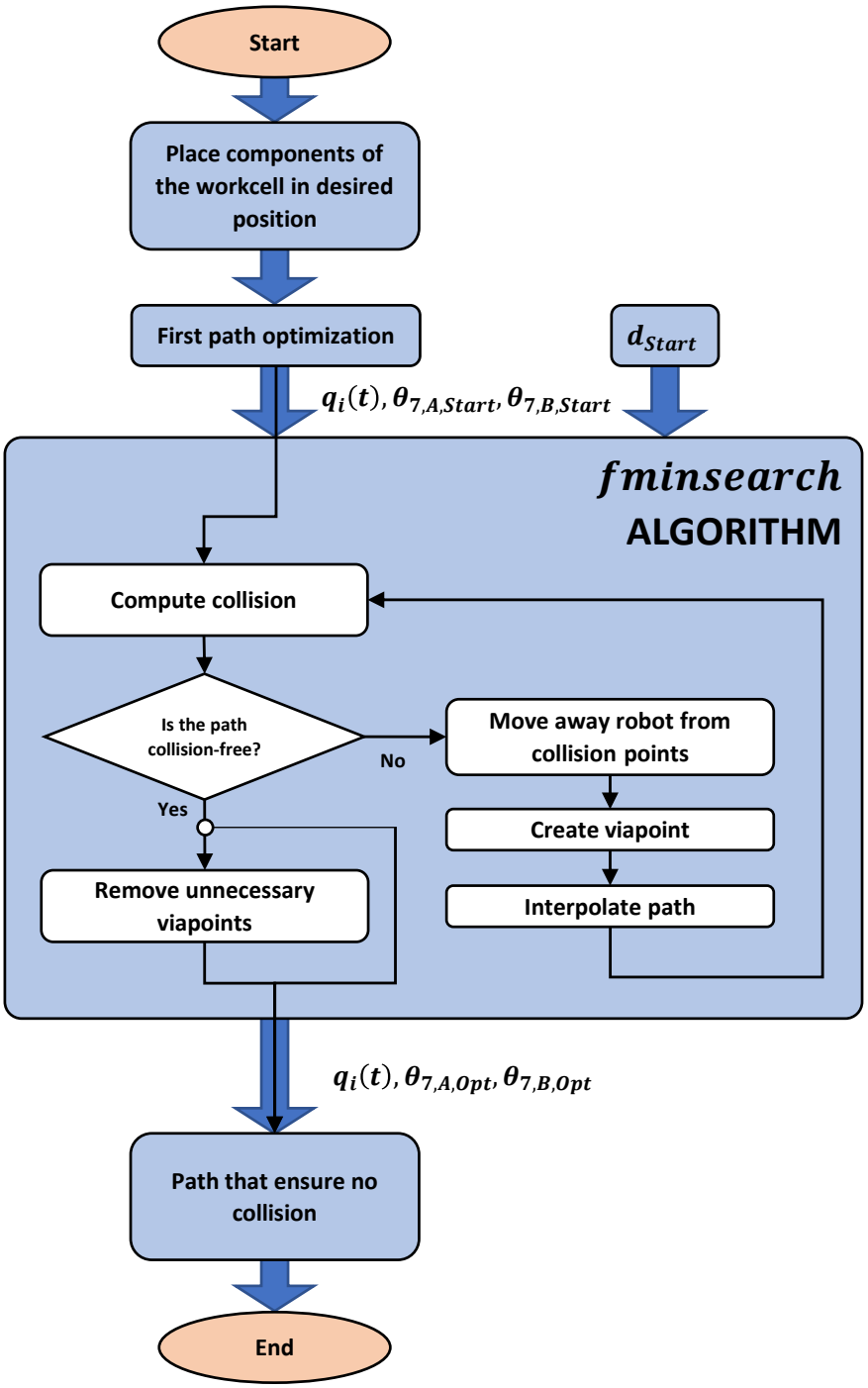


Figure 26: Flow chart that describes the optimization process

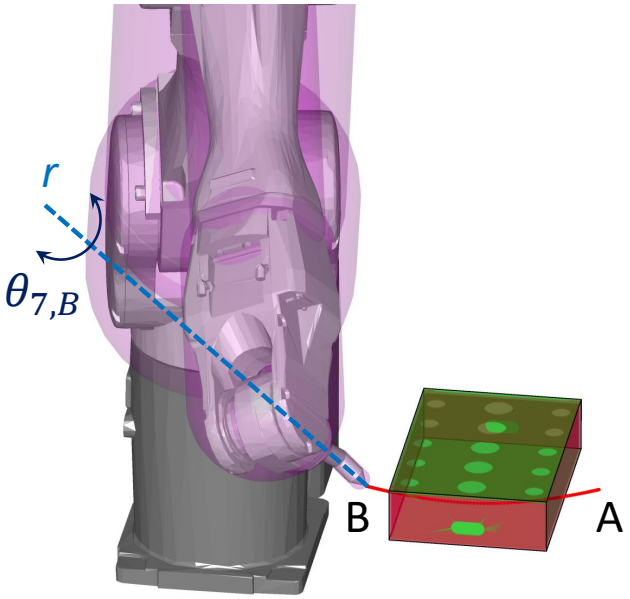


Figure 27: Example of a simple joint interpolated movement (red) from a position A to a position B.

The optimization of θ_7 values at positions A and B ($\theta_{7,A}$ and $\theta_{7,B}$) provides the best functions $q_i(t)$ that connects the two positions, so the lowest t_f value is found. The two values of θ_7 can be called $\theta_{7,A,Start}$ and $\theta_{7,B,Start}$ since they will be used at the beginning of the following step.

3.6.2 *Modified path*

Unfortunately, the functions $q_i(t)$ can make the robot collide with the environment. As can be seen in Figure 27, the red path crosses the green rectangular object. To avoid collisions, the path can be improved by adding safe via points.

To create the via points, collision tests are performed as described in Appendix A.1 along the path found in the previous Section 3.6.1. As soon as a collision is detected, the robot is moved away from the collision point using the method proposed in Section 3.5, and the resulting position is stored as a via point.

The three points (starting, ending and via point) are interpolated by cubic splines (to be consistent with the first path) and new functions $q_i(t)$ are created (Equation 20). Then, the process is iterated until a collision-free path is obtained.

This process can create multiple unnecessary via points. When the collision-free path is obtained, all the possible combinations of viewpoints are interpolated to find the fastest path that ensures no collisions. That's said, the unnecessary via points are removed. This step is optional because it can greatly increase computational times.

3.6.3 *Optimization process*

To find the optimal path the Matlab embedded function *fminsearch* has been used. This function is based on Lagarias et al. [108] method.

The problem variables are three: the distance d to be used in the collision avoidance algorithm and the start and end redundant angles $\theta_{7,A}$ and $\theta_{7,B}$.

fminsearch needs initial values. Instead of starting from generic values, $\theta_{7,A} = \theta_{7,A,Start}$ and $\theta_{7,B} = \theta_{7,B,Start}$ are chosen.

Unfortunately, initial d value influence the final result of the optimization process. Figure 28 shows how the minimum movement time varies by increasing d. This discontinuous behavior is due to the iterative process: there is not a fixed number of via points that are required to complete the movement, so different d values may provide a different number of via points, and this aspect can vary movement times.

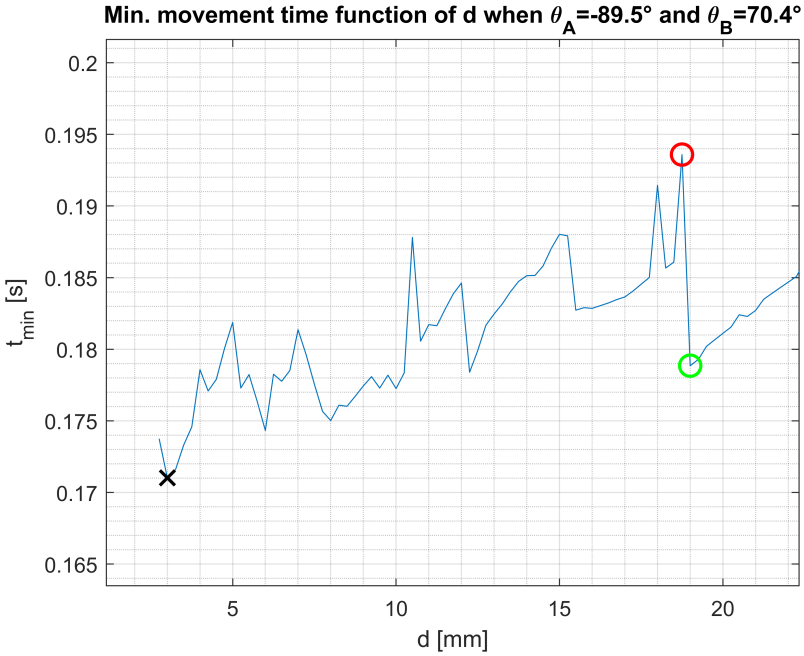


Figure 28: Lowest collision-free movement time depends on d. Black cross and red and green circles describe the solutions with $d = 3mm$, $d = 18.75mm$ and $d = 19mm$ respectively, and the corresponding collision-free paths are shown in Figure 29.

In Figure 29 three different collision-free path are shown. All of them show the optimal path that is found with different d value but the same $\theta_{7,A,Start}$ and $\theta_{7,B,Start}$. The black one is the one with the lowest movement time (equals to the solution described by the black cross in Fig. 28), while the red and green one describes two solutions with very similar d values ($d = 18.75mm$ for the red path and $d = 19mm$ for the green one).

Why the red and the green paths are so different in shape? The reason is simple: if d is smaller, accordingly to the collision avoidance algorithm the robot moves away from the collision point by a smaller distance. This results in via points that are closer to the obstacles, so the interpolation of these via points may result in additional collisions that could have been avoided with higher d values.

This is clear as we see Figure 29: the green path, between the last via point and position B, moves very close to the obstacle. If the via points were slightly closer to the obstacle, the robot would have collided.

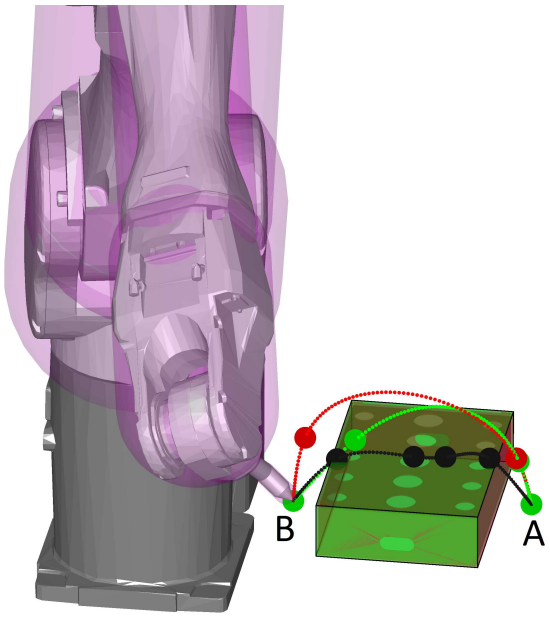


Figure 29: Different d values provide different collision-free paths. The black path describes the solution with $d = 3mm$, red path the solution with $d = 18.75mm$ and green path the solution with $d = 19mm$. The big circles identify the viapoints for each path.

So, due to the irregularities of t_{min} with respect of d and the behaviour of *fminsearch* algorithm, different d_{start} values provide different t_{min} results. In Figure 30 different t_{min} obtained from different d_{start} are shown.

Since it is an iterative process, computational time may vary a lot with d: the higher the number of collisions detected, the higher the number of via points to be created and the number of excessive via points to be removed.

So, it is also important to find a good trade-off between computational times and robot final performances. This is reasonable since this algorithm could be applied in real-time applications that require high flexibility, such as Fully Flexible Assembly Systems [68, 109] or Collaborative Assembly Systems [89].

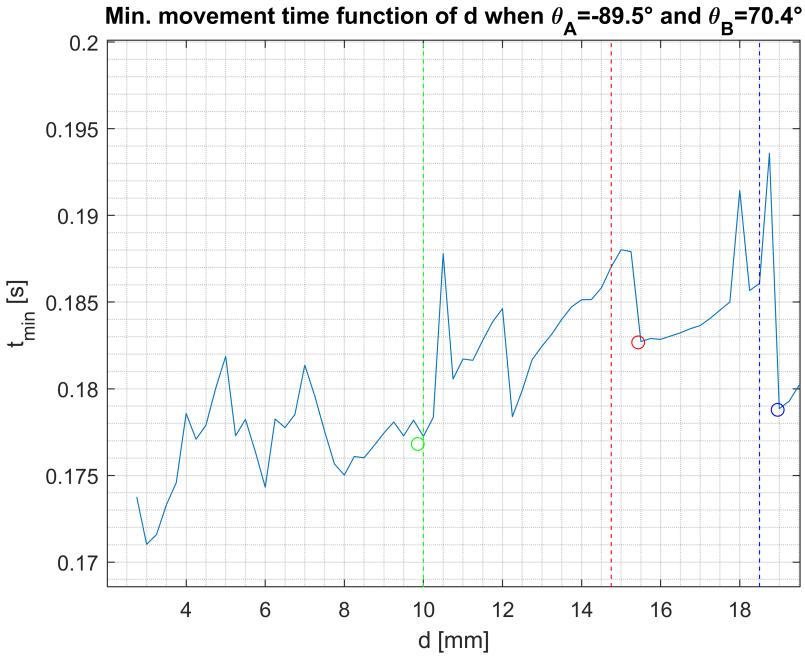


Figure 30: Different d_{start} values (dashed lines) provide different t_{min} results (same color circles).

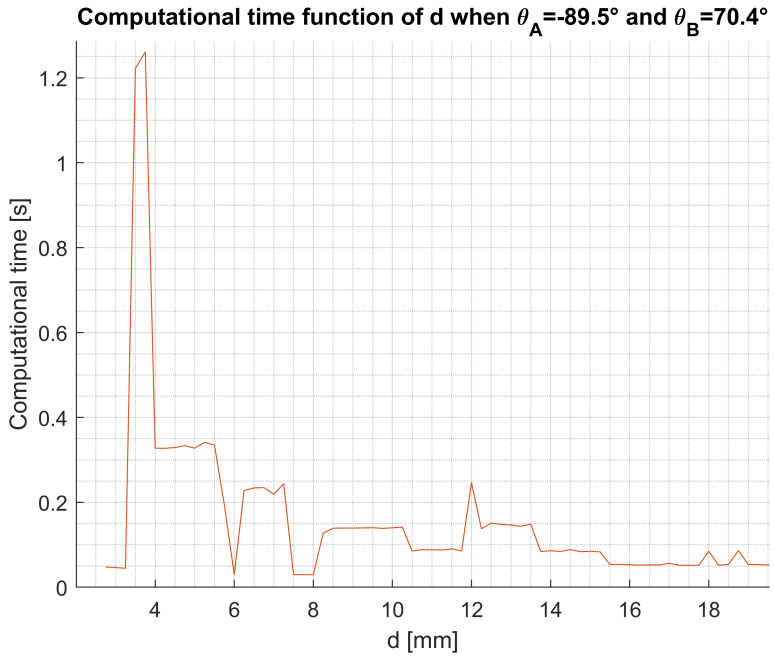


Figure 31: Computational time depends on d.

Figure 31 shows the computational time required to find a solution when $\theta_{7,A}$, $\theta_{7,B}$ are fixed and d varies. Since *fminsearch* evaluates the procedure near the starting values many times, the higher the computational time for a single evaluation, the higher the computational time to find the optimal solution.

The removal of the unnecessary via points is optional since it can greatly affect the computational time: at the end of the construction of the collision-free path, every combination of via points is evaluated to find out if any of them could be removed and reduce the overall movement time. A high number of via points provides a high number of combinations, thus increasing evaluation times. This aspect can be included in the evaluation of the optimization starting point to find out the best trade-off between movement and computational times. An example of the difference between the two solutions will be provided in Section 3.6.4.

As of my tests, a good trade-off between performances and computational time can be obtained with

$$d_{start} = \min(obst_{dist}) \tag{22}$$

where $obst_{dist}$ is the vector containing distances of the starting point A to the planes that define the obstacles.

3.6.4 *Illustrative example*

To prove the capabilities of the algorithm, a simple example is provided. Let A and B be identified in the operational space as the transformations:

$$T_A = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & -520 \\ 0 & -1 & 0 & 810 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 520 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_B = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & -236 \\ 0 & -1 & 0 & 820 \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 460 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

where the translational parts of the matrices are defined in mm. For simplicity, robot base has its reference frame coincident with operational space origin.

A single object (describing a generic workpiece) is placed in the environment ad is defined by 6 planes, whose parameters are shown in the first six rows of table 7. Also, another obstacle is added to the environment, below the workpiece. This obstacle could describe the table on which the workpiece is placed. In this case, only two planes are used, whose parameters are shown in the last two rows of table 7. In Figure 32, the workpiece planes are green, while the obstacle planes are yellow.

The collision detection method threats the workpiece and the obstacle as two different entities: the robot collides with the workpiece if any of its points fall within the volume restricted by the six planes, and collides with the obstacle if any point falls within the huge volume restricted by the two planes.

Object	i-th plane	a_i	b_i	c_i	λ_i
Workpiece	1	1	0	0	482
	2	-1	0	0	-318
	3	0	1	0	-683
	4	0	-1	0	917
	5	0	0	1	-500
	6	0	0	-1	557
Obstacle	1	0	0	-1	500
	2	1	0	0	-250

Table 7: List of plane parameters defining the object (first six planes) and the additional obstacle (as of equation 1 in [26]).

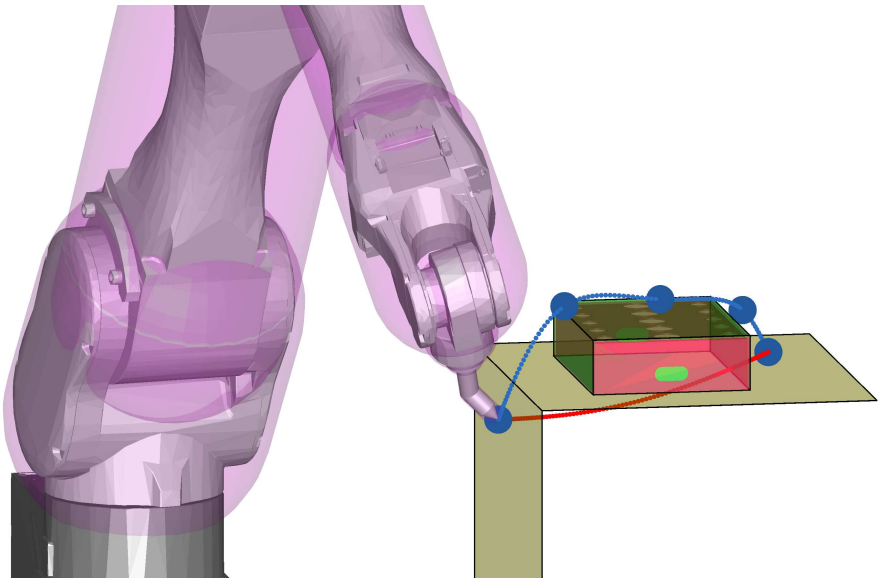


Figure 32: Final optimized movement for the collision-free algorithm of the example (blue) compared to the joint-interpolated movement (red).

By applying the method, the final path can be seen in Figure 32. While the direct movement (red) takes 0.1825s, the optimal movement (blue) requires 0.2109s with around 18.5s of computational time. The single joint movements and joint speeds are shown in Figure 33.

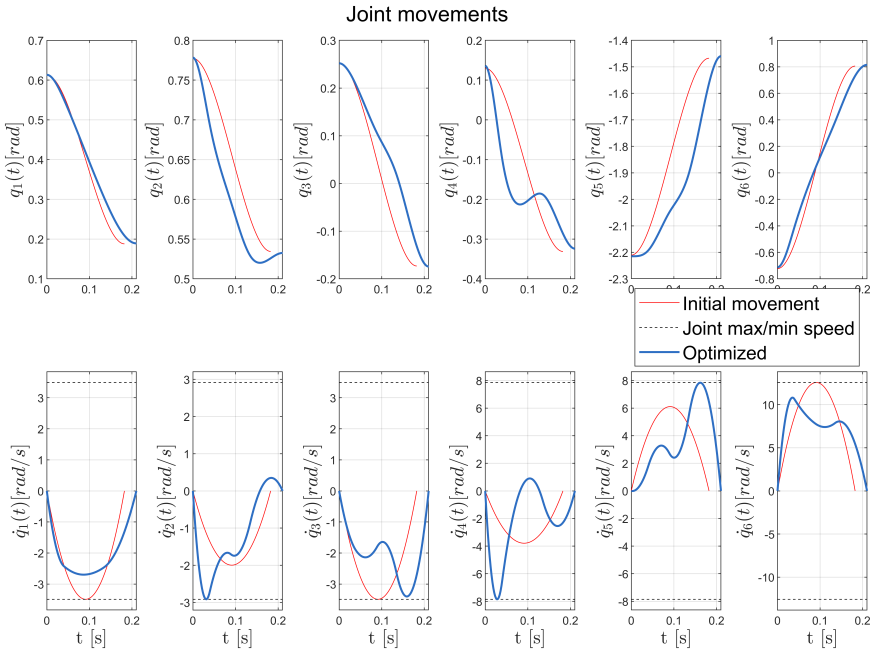


Figure 33: Joint values and joint speeds during the direct movement (red) and the collision-free movement (blue).

3.6.5 *The effect of the removal of unnecessary via points*

To explain the effect of the last, optional, step of the proposed method, let's change the final B position to:

$$T_B = \begin{bmatrix} -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & -280 \\ 0 & 1 & 0 & 820 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 510 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

The two possible solutions (with the removal of unnecessary via points and the one that skips this step) are shown in Figure 34.

It is clear that the removal of the via points provides a smoother path, without rapid direction changes. This aspect is enhanced if the single joint movements are analysed (Figure 35).

Since the robot has to pass through more via points, its movement is generally slower: as of this second example, the path with all the via points needs 0.1918s to be completed, while the one without the unnecessary via points needs only 0.1694s, so the former is 13% slower than the latter one.

Unfortunately, removing the via points requires a higher computational time: the fastest path needs 14.3s to be computed, while the other one requires only 1.58s. There is a huge difference between the two solutions.

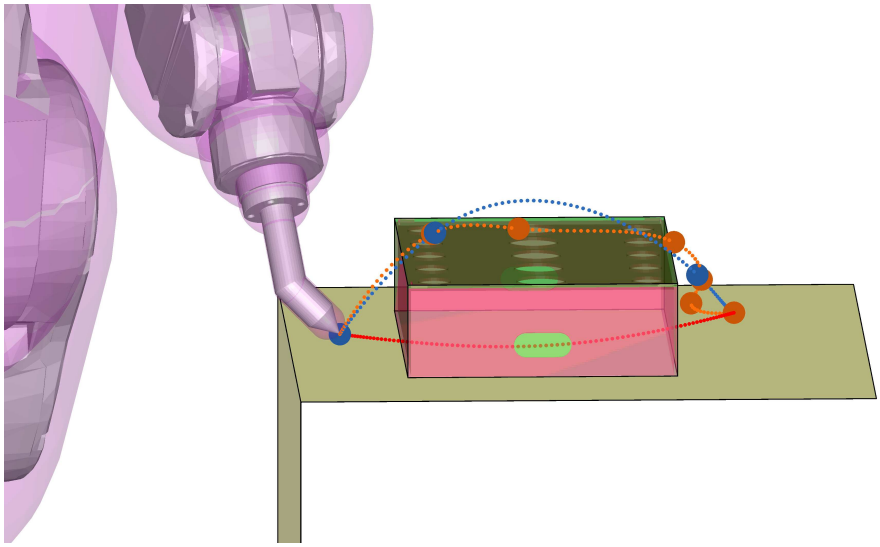


Figure 34: Final optimized movement for the collision-free algorithm of the example (blue) compared to the joint-interpolated movement (red) and the collision-free path with all the viapoints (orange).

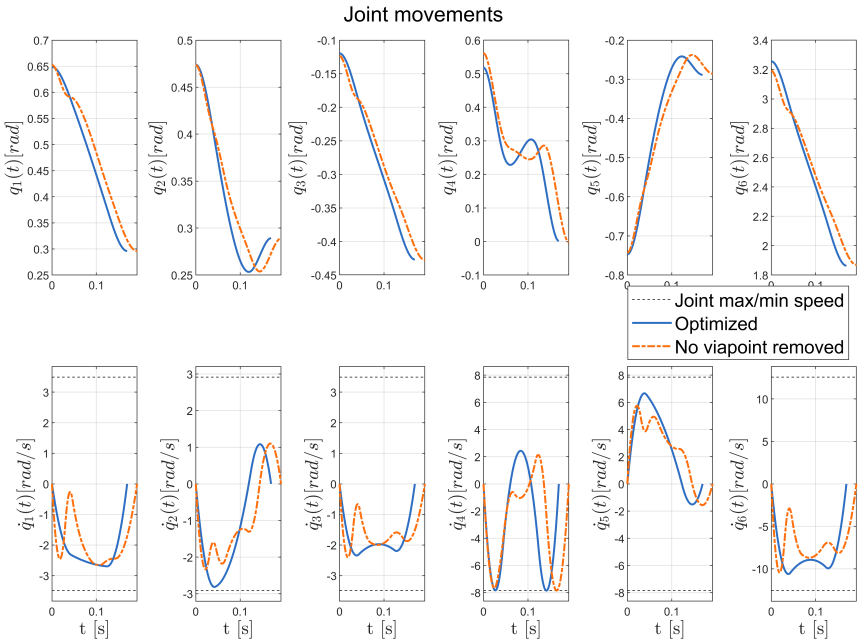


Figure 35: Joint values and joint speeds of the collision-free movement with the viapoints removed (blue) and the one with all the viapoints (orange).

So when it's better to choose one path or another? During the tests, it has been found out that when A and B are close it's better to skip this step since the two possible solutions are very similar. In the same way, in real-time applications, the removal of via points is infeasible due to the low cycle times required by most of the applications.

On the other hand, during the design of a static work cell with static working positions, it's better to remove the unnecessary via points: the total cycle time of the work cell will be lower, increasing productivity, while the computational time is not so relevant.

3.7 COMPARISON BETWEEN THE TWO METHODS

While the Graph and the Adaptive methods lead to the same results, their fields of application can be very different. Let's consider the application of Section 3.3 (with slightly different start and end locations). The objective is the same: move the robot from the start position to the end position optimizing the movement time while handling redundancy.

In Figure 36 the final paths with the two methods are shown. While their movements are very different, the movement times are comparable (Table 8).

Accordingly as stated in Section 3.6.5 the adaptive method has very fast computational times without removing the via points, but the final path is slower to complete. The computational times of the adaptive method with via point removal and one of the graph method are very similar, thus showing the efficiency of the solutions.

However, both the methods are more suitable in specific applications: while the adaptive method is more likely to be used in real-time applications where the flexibility is the most important aspect of the work cell, the graph method is to be used with lots of working positions and a very cluttered environment. In fact, the computation of the net of via points can be used for every combination of working positions, thus reducing the computational times.

	Computational time	Movement time
Graph method (two steps)	4.90s	0.361s
Adaptive method (without removing via points)	1.17s	0.409s
Adaptive method (removing via points)	4.11s	0.368s

Table 8: Comparison between the methods.

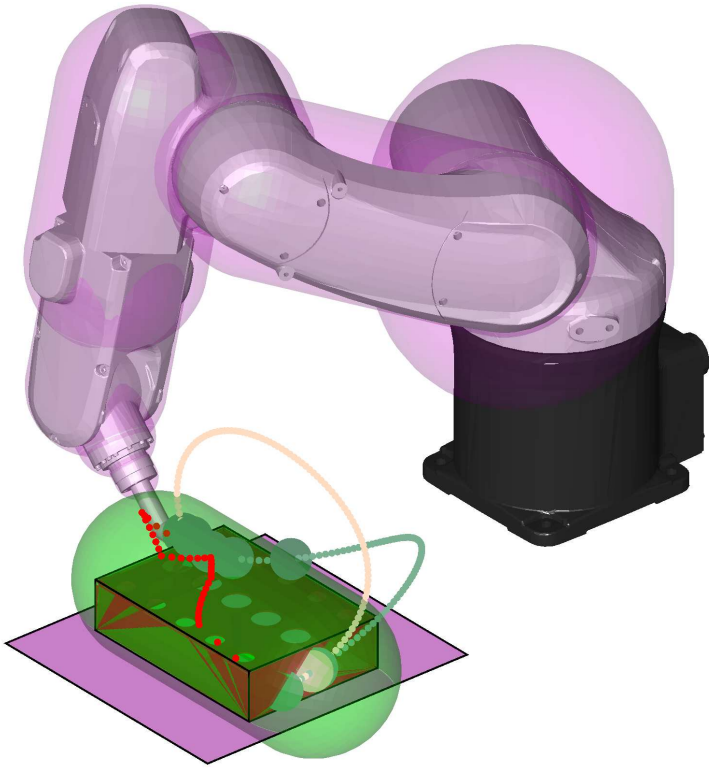


Figure 36: Comparison between the graph method (red) and the adaptive method with via point removal (yellow) and without removal (green).

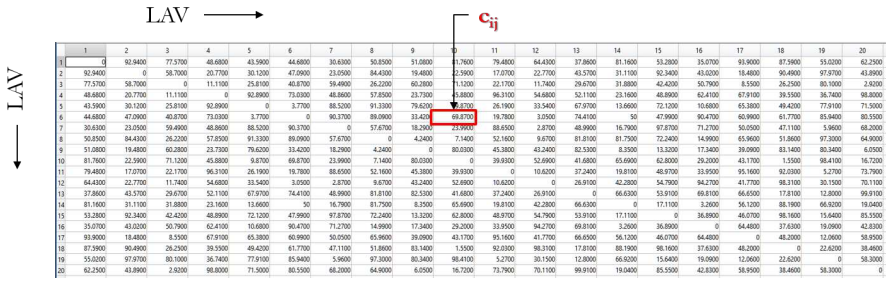


Figure 37: Example of matrix containing all the movement times between all the locations.

With a set of positions **P**, all the possible movements between the locations are optimized. Each movement is defined by its movement time.

All these times are stored within the same matrix C, where each element of the matrix c_{ij} is related to the movement between positions i and j. This matrix will be used in the second optimization described in Chapter 4.

Since the robot joints are moved by electric motors, $c_{ij} = c_{ji}$, and the problem is called "symmetric". Sometimes the movement between two positions could be constrained by a task process, such as a grinding along a path. In this case, the corresponding movement time will be substituted inside matrix C with the time needed to complete the task.

4

SEQUENCE OPTIMIZATION

This chapter presents an improvement on the famous Travelling Salesman Problem (TSP). Those improvements can be used to adapt this optimization problem to particular applications, in which clustering of working position or other constrictions have to be considered. By using the movements created in the previous chapter, the Traveling Salesman Problem is able to provide the best task order that reduces an objective index (e.g. total cycle time or energy consumption).

4.1 THE TRAVELLING SALESMAN PROBLEM

Let's call **P** the set containing all the positions that the robot has to pass through. The Travelling Salesman Problem aims to detect the fastest path through every position $loc \in \mathbf{P}$ just once. The path has to be closed, meaning that the starting position is the only one that has to be visited two times. Actually, there are no constraints on the starting position, so the closed loop can be covered by starting from any of them.

If X is a $N \times N$ matrix, where N is the number of positions, $x_{ij} \in X$ is the arch that moves from loc_i to loc_j. Every x_{ij} has to be an integer, and its value is defined by the status of the path:

$$x_{ij} = \begin{cases} 1 & \text{if } loc_i \text{ and } loc_j \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

The TSP finds X that solves the function f:

$$f = \min \left(\sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} \right) \tag{26}$$

where c_{ij} is the weight relative to the arch x_{ij} stored in the matrix C provided by the first optimization (Chapter 3).

Each arch describes a movement from a position to the next. As described before, this movement x_{ij} can derive from the first optimization, or is a well defined task, such as a grinding path around a work piece, where i and j are the starting and ending point of the path. It is obvious that x_{ji} describes the same path executed in the opposite direction.

To obtain a solution of the problem, the path has to be closed (to obtain a so called "tour"). So, the number of connections between the

positions has to be equal to the number of positions. This constrains can be expressed mathematically as:

$$\sum_{i=1}^{n} \sum_{j\neq i, j=1}^{n} x_{ij} = n \tag{27}$$

Also, to describe the fact that each position has to be visited once, other conditions have to be stated:

$$\sum_{j\neq i, j=1}^{n} x_{ji} = 1 \quad , \quad \sum_{j\neq i, j=1}^{n} x_{ij} = 1 \tag{28}$$

Equation 28 indicates that, for each location loc_i, the number of the entering connections must be equal to 1 (left part of the equation) and, in the same way, the number of exiting connections must be equal to 1 (right part of the equation).

At first, these constrains can lead to multiple closed paths ("subtours"). In this case, additional constrains to remove those subtours have to be implemented, depending on the chosen solving algorithm. This aspect is well known in literature, since is part of the problem (see, for example, [110] and [111]).

4.1.1 *Path constrains*

The original Travelling Salesman Problem, however, provides the optimal path that connects all positions without any constrains. Fortunately, some constrains can be imposed on the problem by applying some boundaries to X.

In some cases, the single task develops along path that connect two (or more) positions. In this case all the positions connected by a single path are constrained to be connected. For a single couple of positions loc_i and loc_j, this constrain can be described as:

$$x_{ij} + x_{ji} = 1 \tag{29}$$

since it is unknown the travelling direction a priori.

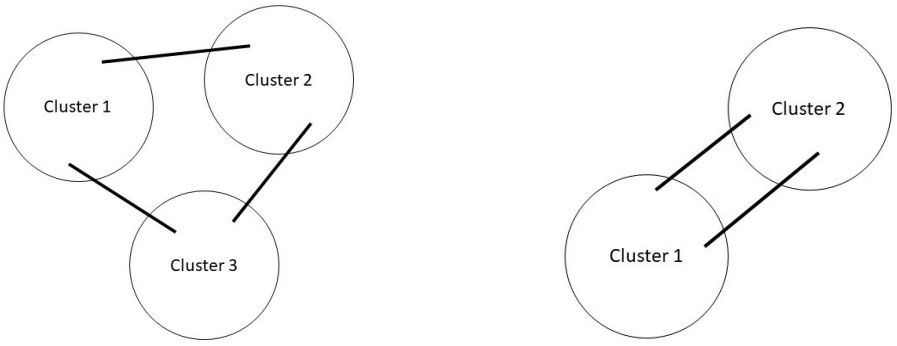
In the same way, it is possible to constrain a connection not to be performed. In this case, Equation 29 can be rearranged to fit the need in a very simple way:

$$x_{ij} + x_{ji} = 0 \tag{30}$$

since, again, it is unknown the travelling direction a priori.

4.1.2 *Cluster constrains*

In some cases the positions can be divided into clusters. This means that for each cluster $S \subset \mathbf{P}$, containing n_s positions, the TSP will have



(a) $n_c > 2$

(b) $n_C = 2$

Figure 38: Examples of clustering connections: if $n_c > 1$, the number of connections between the clusters is always equal to n_c.

to define a non-closed sub-tour containing only the positions of the cluster. This constrain can be defined as:

$$\sum_{i \neq j, i \in S, j \in S} x_{ij} = n_s - 1 \tag{31}$$

since the last connection that would close the tour has to be removed in order to connect the cluster with the other positions.

Let's introduce set $R \subset X$ as a set that includes all the connections between the positions of different clusters and n_c as the number of clusters.

If $n_c > 1$, in the same way as the single positions (Equations 27 and 28), all the clusters have to be connected by a number of arches equal to the number of clusters (Figure 38, Equation 32). Also, each cluster has to connect at most once with any other cluster, only if $n_c > 2$ (Equation 33, where S_1 and S_2 are two generic clusters). This last constrain does not have to be applied with $n_c = 2$ (Figure 38b) because in this case the two clusters must be connected by two arches in order to obtain a closed path.

$$\sum_{i \neq j, i \in R, j \in R} x_{ij} = n_c \tag{32}$$

$$\sum_{i \in S_1, j \in S_2} x_{ij} = 1 \qquad \text{if } n_c > 2 \tag{33}$$

4.2 SIMULATION MODEL AND RESULTS

To validate the method and to find out the capabilities of the algorithm, a generic 3D CAD file has been used, on which 22 points have

been distributed. In the following Figures (39, 40, 41, 42, 43, 44) the numbers of the points do not describe their position in the TSP solution, but are only identification numbers. Some tests have been performed and the computational times have been recorded. The tests are the following:

Test 1 At first all the points are connected by a simple TSP evaluation. No particular constrains are applied. This is a very simple application of the standard algorithm that can be found in previous works.

Test 2 The points are divided in clusters and the TSP is applied with this set of constrains (see Section 4.1.2).

Test 3 The points are divided in clusters and some positions are forced to be connected (see Section 4.1.1).

Test 4 The points are divided in clusters and subclusters (clusters of clusters) and some connections are fixed.

Test 5 The points are not divided into clusters but two additional points are added (simulating safe robot positions).

Test 5 is carried on with a particular basic idea: while working on a product, the robot usually starts from a safe position, moves to a working position, perform all the tasks and then moves again to a safe position. So, there is no need for the TSP to provide a closed path around working positions: the algorithm has to find the best path that moves from a point and end to another. These two positions has to connect to the safe "home" position.

The application of the TSP algorithm has been performed in a Matlab® environment powered by a Intel Core i7-8750H, 8 GB of RAM and 64 bit Windows 10 Home 1809 version. To compare all the solutions, a "total cost" for the whole path is provided (in generic units u). This total cost is to be interpreted as the total travelling time spent in moving through the path. Also, the computational time of each test is provided, where each test has been performed 1000 times for statistical relevance.

4.2.1 *Test 1 - Simple TSP*

The simple TSP algorithm proceeds all the 22 positions and finds the fastest path that connects all of them. In Figure 39 this path is shown. The total cost in this case is of 3425u, and the computational time is of 57 ± 3.6 ms. This is the standard behaviour of the TSP algorithm as can be found in many applications.

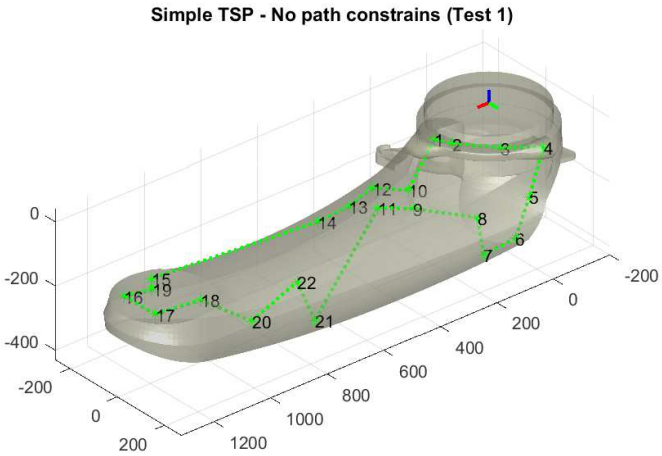


Figure 39: Fastest path of Test 1.

4.2.2 *Test 2 - Clustering*

Clustering the positions constrain the path to enter inside a cluster, complete it and continue to another one. In Figure 40 the path with this solution is shown, and Table 9 shows in which clusters the positions have been placed. To better understand the cluster division, in Figure 41 the single clusters are shown.

In this case the solution is actually slower than Test 1: the total cost lifts up to 3501u. However, computational time reduces to 11 ± 1.1 ms due to the higher number of constrains: the algorithm is able to obtain a single closed loop solution using a lower amount of iterations, thus reducing total computational time. So, if a lot of simulations have to be performed in order to evaluate lots of workpieces, it would be better divide the positions into clusters to speed up the computation.

4.2.3 *Test 3 - Clustering and connection constrains*

Using the same clusters as Test 2, some connections have been constrained. In this case the archs that connect, respectively, points 8 and 9, 20 and 22, 7 and 22 are obligated to be part of the tour (Equation 29). In the same way, using Equation 30, those paths could be obligated not to be performed.

Figure 42 shows the final path that connects all the positions with the described constrains. The obligated connections are described by thick red lines. As of Test 2, the solution of Test 3 is way slower than Test 1: the total cost is of 4652u, but the computational time lowers to 17 ± 1.0 ms. This low computational time is due to the reduction of possible paths that connects all the points: 3 of the 21 connections are already set up and fixed at the beginning of the optimization algorithm.

Cluster	Point numbers
1	1,2,3,4
2	5,6,7,8
3	9,10,11,12,13,14
4	15,16,17,18,19
5	20,21,22

Table 9: Clustering division. For each cluster, the list of positions (defined by its identification number of Figure 40) is provided.

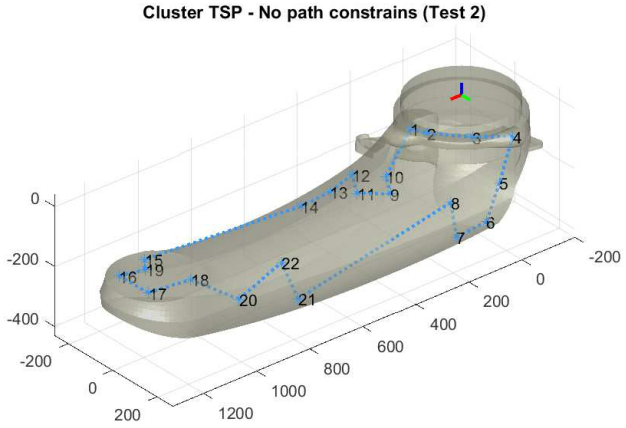


Figure 40: Fastest path of Test 2.

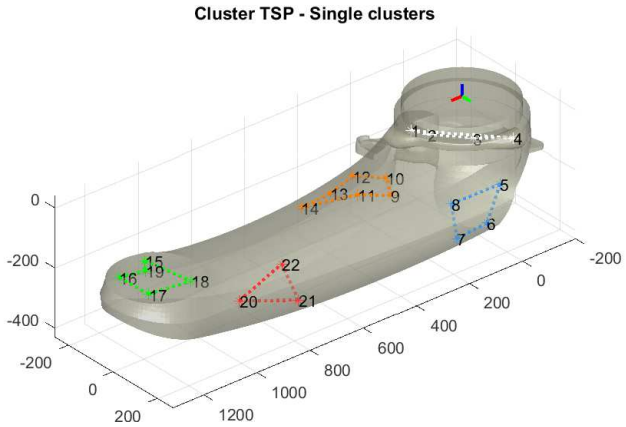


Figure 41: All the points divided into clusters. The single clusters have been connected to form a closed path to better identify the single groups.

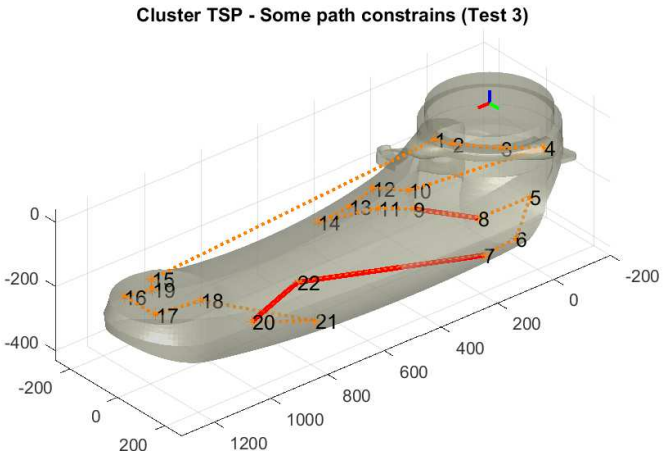


Figure 42: Fastest path of Test 3. The red thick lines identify the obligated connections.

So, the obligation of a list of connections can greatly reduce the computational time. Choosing a few connections and fixing them is a great way of computing lots of possible combinations in a very little time (the higher the number of positions, and so the number of possible paths, the higher the computational time [55]).

4.2.4 *Test 4 - Clustering, subclustering and connection constrains*

Grouping clusters into bigger clusters can be a good choice in reducing cycle times. In this test the connections between positions 20 and 22, 18 and 22, 8 and 9 are constrained. Among the clusters of Test 2, clusters 3 and 5 are merged together to form a bigger subcluster. Equation 31 is still valid, even if the subcluster contains multiple clusters.

This test shows that, even if the overall cost can increase (the solution costs 3872u), the computational time remains nearly constant, taking 17 ± 0.8 ms.

4.2.5 *Test 5 - Constrains on the slowest connection*

So far, the improvements have focused on obtaining the best solution that can connect all the positions to form a closed loop with minimum cost. However, in a real application scenario the operator (manual or robotic) starts to work at one position and ends in another one, without closing the loop. This means that, anyway, one of the connections chosen by the TSP is not going to be used.

The robot starts from one safe position, passes through all the working positions and exits to a safe position (that can be different from the starting one). This addition can be described by adding one (or

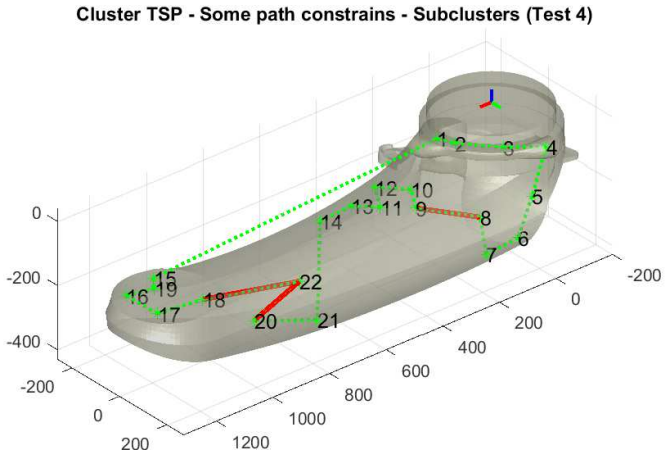


Figure 43: Fastest path of Test 4.

Test	Overall cost [u]	Cost without slowest connection [u]	Computational time [ms]	Standard deviation [ms]
1	3425	2875	57	3.6
2	3501	2948	11	1.1
3	4652	3730	17	1.0
4	3872	2950	17	0.8
5	3761	2762	24	1.6

Table 10: Costs and computational times of the 5 tests performed. The same data can be found in Figure 45.

two) safe position to the group of working positions and running the TSP with the new group.

To provide a good solution, the two safe positions have to be included into a cluster, while the entire set **P** has to be included inside a different cluster. In this way the TSP has to evaluate a higher number of positions (theoretically increasing computational time [55]) but the introduction of the clusters partially mitigate this factor.

As can be seen in Table 10, this solutions is not the optimal one if the overall cost is considered. However, this is not a valuable result since the safe positions are unknown at priori, so this is not the correct way of comparing this solution to the previous. Test 5 provide the fastest path if the slowest connection is removed from all the results of the tests and all the connections to the safe positions in Test 5 are not considered (third column of Table 10 and Figure 45). Computational times are also reduced, becoming one of the best trade-off between computational effort and overall solution.

In conclusion, by applying this constrains it is possible to use the TSP in particular applications, such as deburring, in which some tasks

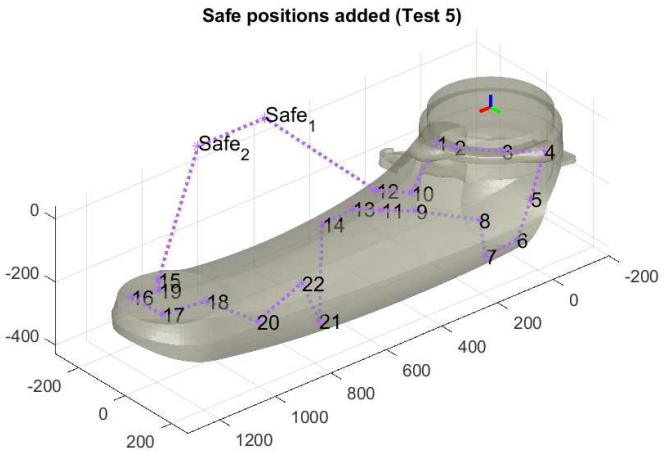


Figure 44: Fastest path of Test 5.

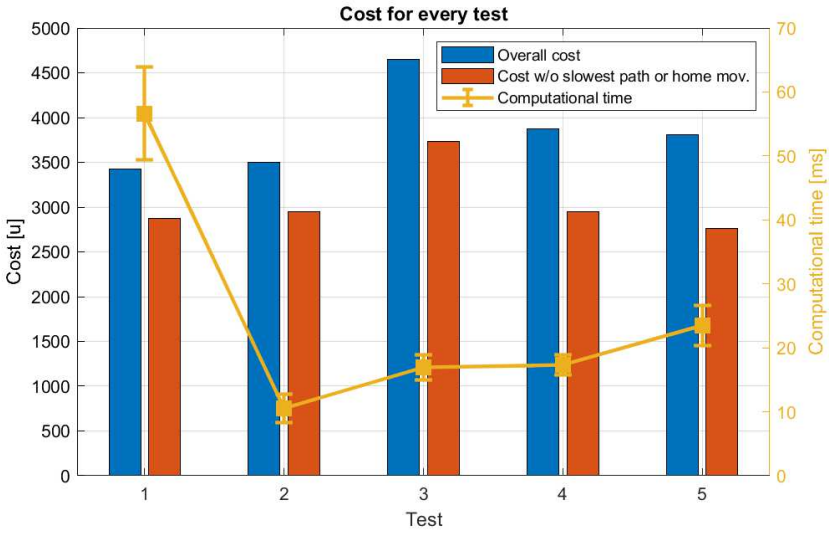


Figure 45: Costs and computational times of the 5 tests performed. The error bars shows 2 times the standard deviation (Table 10).

have to be divided into clusters due to some limitations. The implementation of these constrains is not complex: the only constrains lies in the fact that the optimization algorithm that has to solve the problem has to handle external constrains.

In the MATLAB environment, it is possible to achieve this objective by using the *intlinprog* built-in function [112].

4.3 REAL CASE APPLICATION

The optimization described in Chapters 3 and 4 can be used together to fully automate the optimization of the movement between the tasks. In this Section, a real case scenario is shown.

The workpiece, which is shown in Figure 46, is made of 19 grinding positions, 7 on the side and 12 on the top. Those have been divided into 5 clusters (Table 11): the 7 locations on the side, the 5 working positions to the top left, the 5 working positions to the top right, the single spot in the central part and the last location on the top of the central pyramid. Only two connections have been constrained: the one between the locations 13 and 14, and the one between locations 16 and 17.

Due to the particular shape of the workpiece, three different OBBs have been used for the encapsulation (Figure47). While the first one encapsulates the top pyramid, the other two are used for the medium and the bottom part of the workpiece.

To describe the movement between the locations the Adaptive method has been used. In this particular application, even the sixth joint adapts its value due to the shape of the end effector. Since no real end effectors have been provided for this test, a generic redundant end effector has been used.

The final result is shown in Figure 48. The overall movement requires 2.34s, which is not to be intended as the work cell cycle time, because the grinding tasks are not considered.

However, it is interesting to notice how the algorithm can improve the final performance of the workcell: even if the solution of Figure 49 looks like more intuitive, the overall movement requires 3.22s, so around 37% higher than the one provided via the algorithm. In this case, this is due to better exploitation of the redundancy of the task.

The calculation of the entire process, due to the high complexity of the system and the high number of combinations between the locations, took around 317s for the paths and only 0.12s for the TSP. This is not a very high computational time and can be further improved by using optimized code.

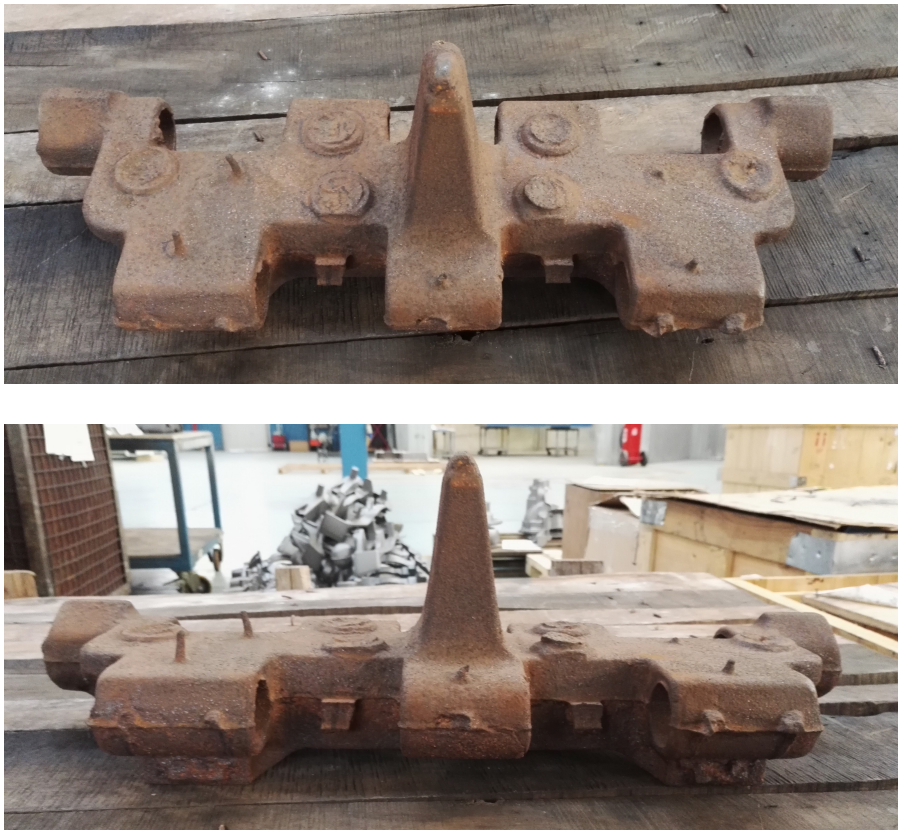


Figure 46: Real work piece on which the optimization procedure has been applied.

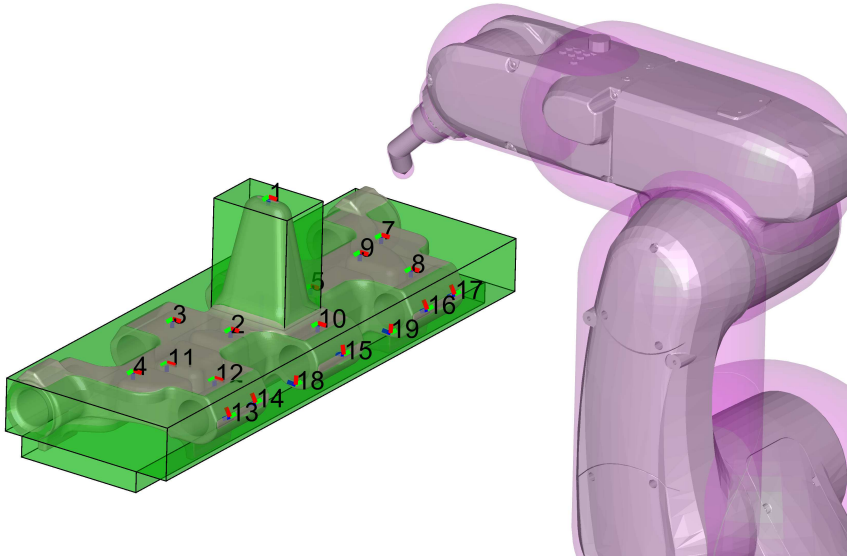


Figure 47: OBBs and working positions used in the simulation of the real case study.

Cluster	Point numbers
1	1
2	10
3	2,3,4,11,12
4	5,6,7,8,9
5	13,14,15,16,17,18,19

Table 11: Clustering division. For each cluster, the list of positions (defined by its identification number of Figure 47) is provided.

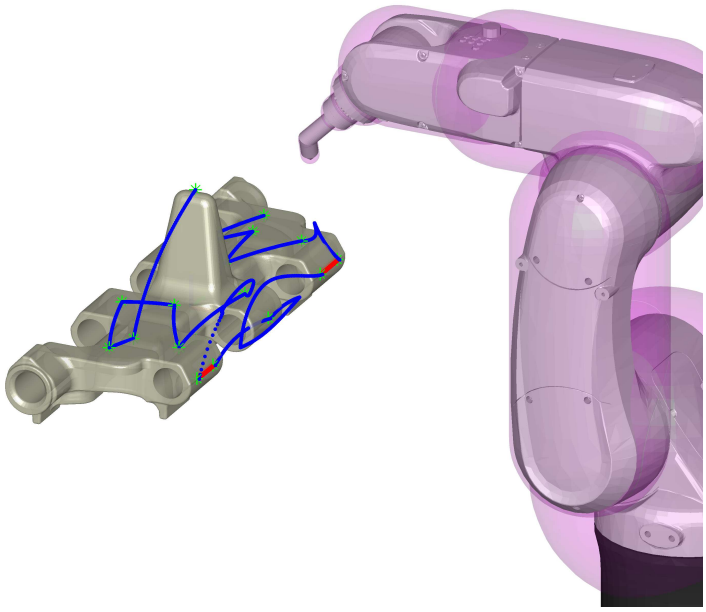


Figure 48: Real case application. The red paths show the constrained connections, while the blue paths describe the end effector trajectory.

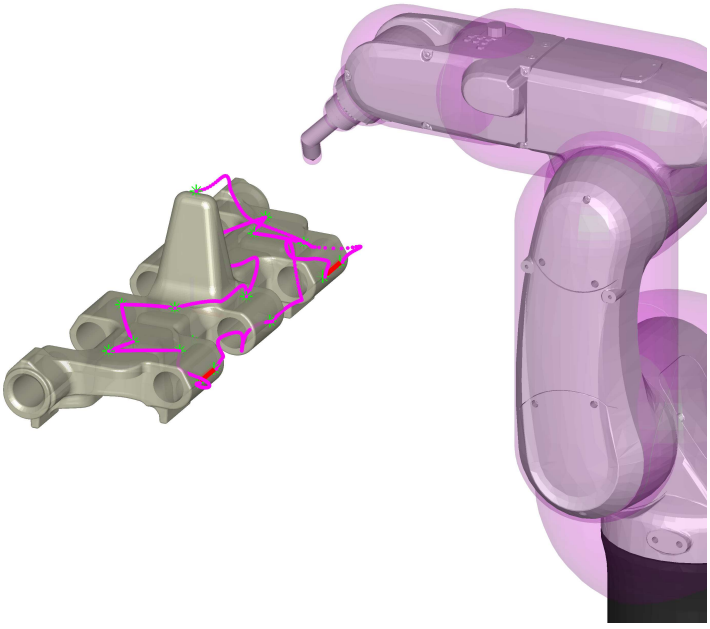


Figure 49: Real case application as the ideally more intuitive solution that a human operator could define.

5 ROBOT DYNAMICS

This chapter is divided into two parts. Firstly, it presents the vibratory analysis of a six degree of freedom (DOF) robotic arm. The objective is to identify the joints that are more responsible for robot compliance under high forces and the ones that are more important from a vibratory point of view. Secondly, a comparison between a 3 DOF under-actuated system and a 2 DOF fully-actuated system is presented, to see if a possible compliant tool would be a good solution to improve the capabilities of the robotic deburring.

5.1 THE IMPORTANCE OF DYNAMICS IN ROBOTIC DEBURRING AND COLLABORATIVE ROBOTICS

The compliance properties of industrial robots are very important for many industrial applications, such as automatic robotic assembly and material removal processes (e.g., machining and deburring). On the one hand, in robotic assembly, joint compliance can be useful for compensating dimensional errors in the parts to be assembled; on the other hand, in material removal processes, a low Cartesian compliance (high stiffness) of the end-effector is required. Indeed, still very few robots have been applied in this economically important application area [113] mainly due to their low stiffness. Moreover, the compliance properties of robots appear very important in emerging fields such as flexible assembly systems [68, 109] and collaborative robotics [89].

From a static point of view, low stiffness causes imprecise products, due to the robot deflections during the robotic task. From a dynamic point of view, low frequency chatter vibrations [114, 115, 116] can be induced when low-stiffness robots are used, with an impairment in the quality of the machined surface. Moreover, induced vibrations cause a reduction of tool life and can damage robot joint transmission.

In robotic deburring, the forces transmitted from the machining process to the robot structure mainly depends on the rate of material removal (Section 2.3), and usually the material is not uniformly distributed. The reasons behind this behavior rely on the manufacturing process, in which high forces, stress and wear tend to slightly modify the shape of the mold that provide the first shape of the final product. This results in burrs of different height and width along the path that the robot has to follow.

Rapid changes in the forces magnitude and direction can produce vibrations at the end effector, modifying the height of material removed and, thus, the force applied to the robot (Equation 6).

If the dynamic behavior of a robot is known, it is possible to adapt the configuration of the robotic arm to avoid undesired fluctuations.

On the other hand, the forces transmitted by the robot are a safety problem in the collaborative robotics field. The so-called *cobots* [75] allow taking advantage of the flexibility of the human operator while using the precision and the accuracy of the robot. However, safety reasons restrict the forces that the robot may transmit to the operator [77], thus reducing maximum speed [78].

Developing novel strategies that could reduce the forces transmitted in the interaction between human and robot could improve the speed limits of the robot thus increasing its productivity.

5.2 VIBRATIONS OF A SIX DEGREE OF FREEDOM ROBOT ARM

Low joint stiffness of the robot is one of the main issues in using robotic machining instead of CNC machining. Indeed, it is well established in the literature [117, 118, 119, 120, 121] that the dominant contribution factor for a large displacement of the robot end-effector is joint compliance (mainly due to gear transmission elasticity), while link flexibility can be neglected. Moreover, the stiffness of an industrial robot is usually on the level of 10^6 N/m (with a base natural frequency of robot around 10 Hz), while a standard CNC machine has stiffness on the level of 10^8 N/m (with a base natural frequency of several hundred Hz or even more than 1000 Hz) [114].

Sometimes, combinations of serial and parallel kinematic chains have been proposed to increase robot stiffness [122], but it is a solution that in many cases is impassable, since it would increase robot obstruction, reduce robot work space and increase the costs.

The most important properties regarding the dynamic behavior of the robot during the deburring process are joint stiffness and joint natural frequency. The former is important when neatly static forces are applied, while the latter control the frequency range in which the robot can operate.

5.2.1 *Testing equipment and method*

Stiffness (or compliance) properties of robot joints are usually measured by means of static tests [123, 121] carrying out measurements in the Cartesian space. Then, a mathematical model is used to correlate force vector $\{F\}$ and deformation vector $\{\Delta x\}$ in Cartesian space with the stiffness matrix $[K_q]$ in the joint space [117].

During my thesis, a different approach has been adopted. This is based on the modal testing, in which the structure the robot is exited in different configurations, in such a way that, for each configuration, a single joint dominates the compliance at the end effector.

From the modal analysis the modal frequencies are obtained. A linear vibrating system with N-DOF is described by a system of coupled second-order differential equations in space coordinates. The modal approach transforms the N-DOF vibrating system into N independent 1-DOF vibrating systems [124, 125] governed by these equations:

$$m_i\ddot{\eta}_i + c_i\dot{\eta}_i + k_i\eta_i = F_i(t) \qquad i = 1, \cdots, N \tag{34}$$

where m_i, c_i and k_i are the modal mass, damping, and stiffness, respectively. η_i is the i-th modal coordinate and $F_i(t)$ is the modal force. The natural frequency is:

$$\omega_i = \sqrt{\frac{k_i}{m_i}} \tag{35}$$

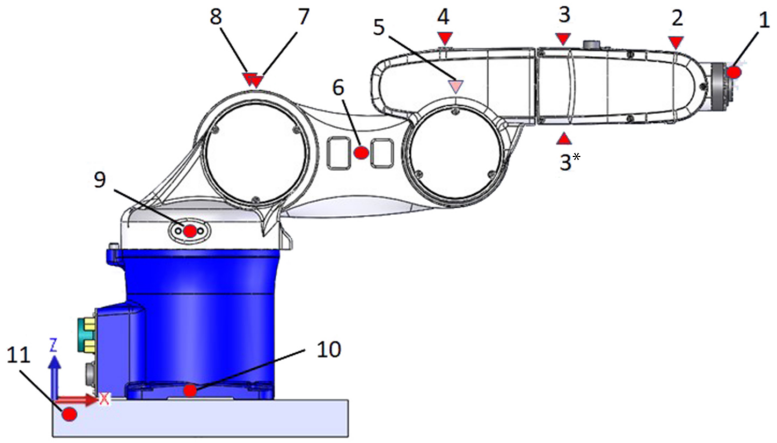
If robot vibrations are considered, in general, modal stiffness does not coincide with joint stiffness, because the modes of vibration involve rotation about several joints. However, if a mode j exists that is dominated by rotation about joint j, the modal stiffness k_j is a good approximation of joint stiffness k_{qj}. Therefore, joint stiffness can be calculated from Equation 35 with $i = j$.

In this case, modal mass m_j coincides with the moment of inertia of the robot about joint j. These moments of inertia were calculated using the CAD model of the robot. In particular, the moment of inertia about joint j includes all the links that are interested by the motion of j-th joint, so all the joints i with $i = j, \cdots, N$.

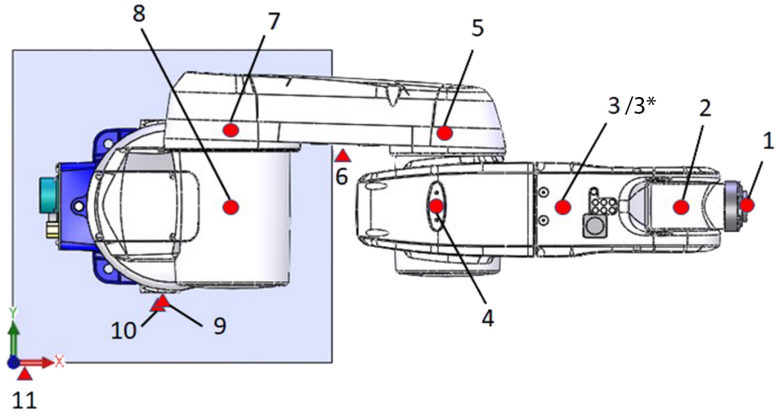
Since the manufacturer gives the total mass of the robot (28 kg) but does not give information about mass distribution, the total volume of the robot and of the various links were carefully calculated from the CAD model of the robot and the total mass was distributed to the links proportionally to their volumes. Due to this aspect, joint 6 compliance has not been studied, since no relevant information is provided for the last link: its moment of inertia cannot be established with enough precision to obtain a reliable value of last joint stiffness.

The equipment needed to carry out this kind of test is rather simple and cheap; it includes a hammer for modal testing, an accelerometer, and a data acquisition board. In the framework of this research, a PCB 086C01 hammer (with load cell sensitivity 0.2549 mV/N), a PCB 356A17 triaxial accelerometer (sensitivity 50.5 mV/(m/s2))[1], and a NI9234 data acquisition board[2] were used. After some preliminary tests, a sampling frequency of 1024 Hz and 2048 samples were selected. Measured signals were analyzed by means of ModalVIEW R2, a specific software for modal analysis. The tested robot is a six axis

1 Built by PCB Piezotronics, Inc., Depew, NY, USA
2 Built by National Instruments, Austin, TX, USA



(a) Lateral view



(b) Upper view

Figure 50: Accelerometer locations for the modal study.

serial robot Adept Viper s650, with a maximum payload of 5 kg and a maximum reach of 653 mm.

To perform the test, for each configuration the hammer will excite the same location with the same force F_h, while the accelerometer is moved in different spots of the robot arm. The accelerometer locations are shown in Figure 50, and are consistent for all the configurations that will be considered. To improve the accuracy of the evaluation of the fourth joint stiffness, an additional accelerometer, called 3^*, is placed underneath accelerometer 3, on the opposite side of link 4.

For each configuration, 33 frequency response functions (FRFs) were measured between the 3 acceleration components of the 11 grid points and the hammer impact force. In order to improve the repeatability and quality of measurements, each FRF was calculated averaging the results obtained with three hammer blows. Measured data were then processed with ModalVIEW in order to identify natural frequencies, modal dampings, and modal shapes.

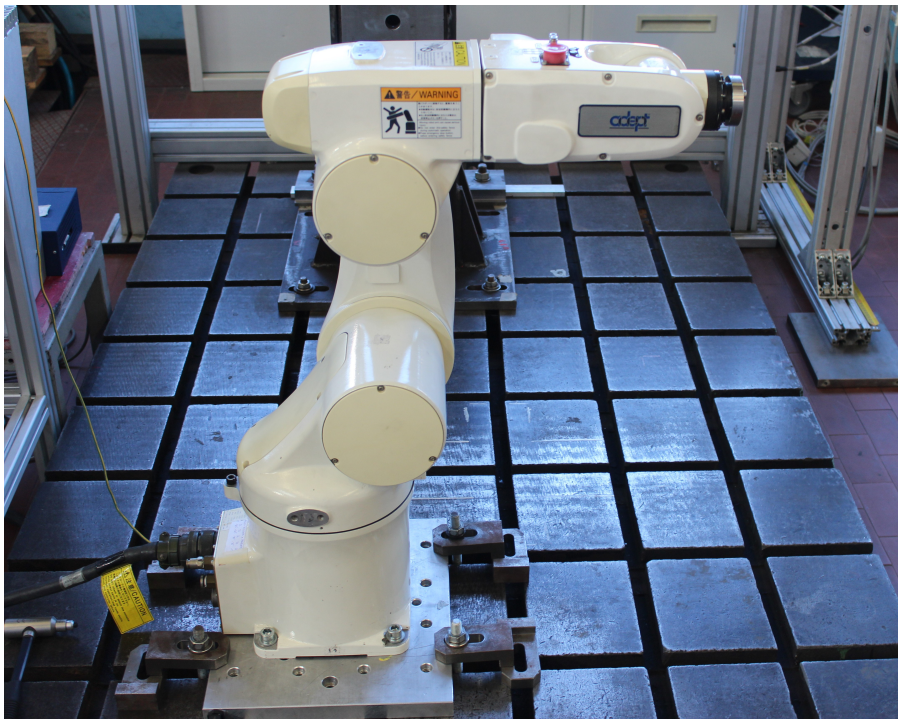


Figure 51: The robot in the testing rig.

Since motions of the base of the robot may have a negative effect on the quality of measurements, the robot was rigidly fastened to a large steel base (see Figure 51) and an accelerometer was used to monitor residual base motions (point 11 in Figure 50).

To find every joint stiffness, multiple configurations has to be tested. For each joint, a corresponding configuration has been chosen, in which the force applied by the hammer mainly excite a single joint. To be more clear, since the robot is made of rotational joints, and the compliance of these joints is due to the torques applied to them, each configuration is chosen in such a way that the force generate a relevant torque only about the joint to be studied.

The configurations are shown in Figure 52. To improve clarity, robot angles for each configuration are shown in Table 12, and have been chosen as follows:

Test 1 To find first joint stiffness, links 2 and 3 are placed parallel to the ground. Force F_h direction is parallel to joint axis 2, 3 and 5 and intersects joint axis 4 and 6. As a result, the only joint with a reasonable lever arm is joint 1 (Figure 52a). In this configuration, force F_h achieve maximum lever arm with respect to joint 1.

Test 2 To find second joint stiffness, link 2 is parallel to the ground and link 3 is placed at 90° with respect to the second link. Force F_h direction is parallel to joint axis 1, 4 and 6 and intersects joint axis 3 and 5. As a result, the only joint with a reasonable lever arm is joint 2 (Figure 52b).

Configuration	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Test 1	0	0	90	0	0	0
Test 2	0	0	16.96	0	-16.96	0
Test 3	0	-37.54	176.73	0	49.20	0
Test 4	0	-185.42	228.41	0	90	0
Test 5	0	-185.42	229.09	-90	0	0
Validation	0	30	30	0	0	0

Table 12: Robot configurations. All the angle values are provided by robot controller.

Test 3 To find third joint stiffness, force F_h direction intersects joint axis 1, 2, 4 and 5. As a result, the only joint with a reasonable lever arm is joint 3 (Figure 52c).

Test 4 To find forth joint stiffness, force F_h direction is parallel to joint axis 2, 3 and 5 and intersects joint axis 1. As a result, the only joint with a reasonable lever arm is joint 4 (Figure 52d).

Test 5 To find fifth joint stiffness, force F_h direction is parallel to joint axis 2 and 3 and intersects joint axis 1 and 4. As a result, the only joint with a reasonable lever arm is joint 5 (Figure 52e).

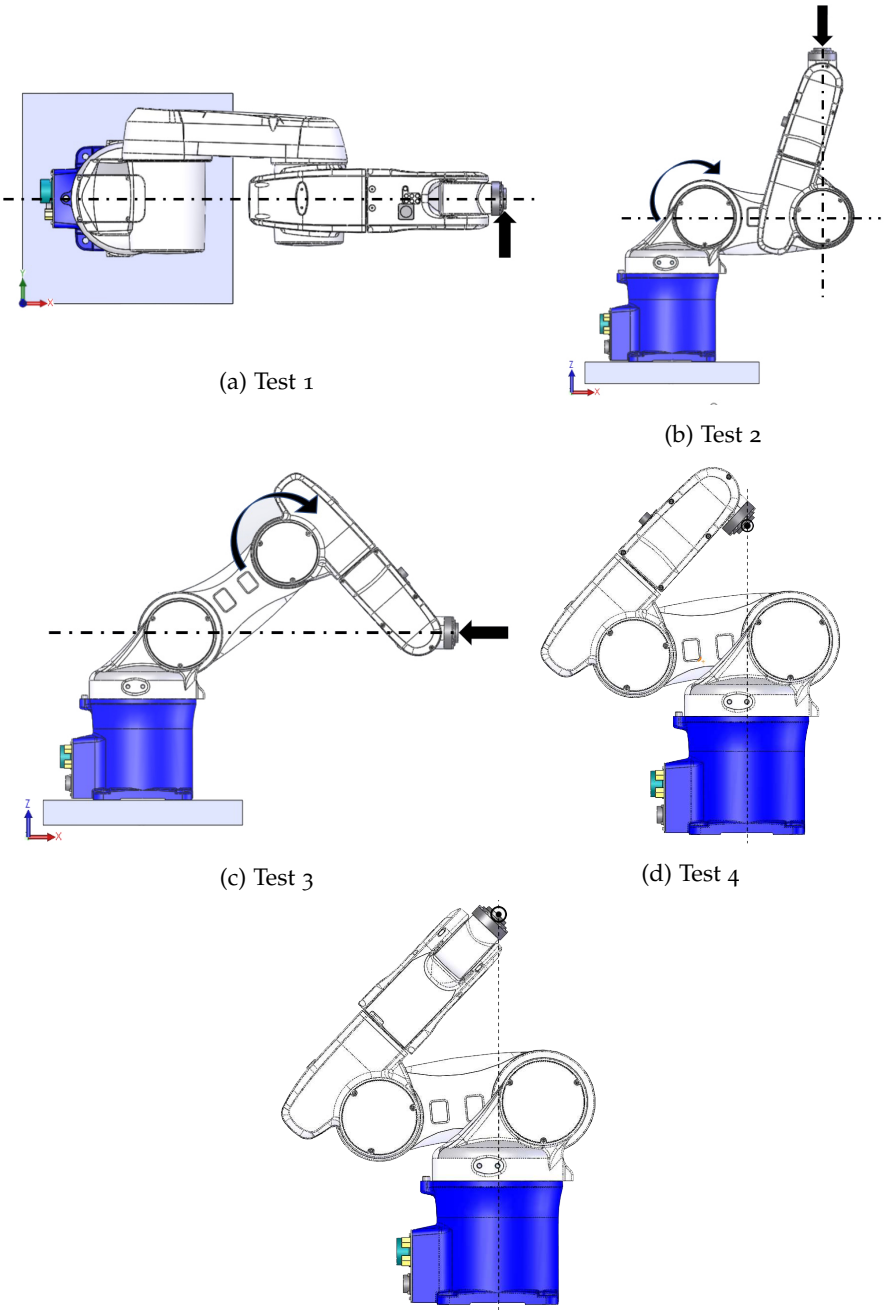
5.2.2 *Experimental results*

Figures 53, 55, 57, 59, 60 show the results of the modal experiments.

Figure 53 shows the overlay of FRFs measured in configuration 1. The modulus plot highlights the presence of the fundamental mode at about 13 Hz, and the phase plot corroborates this fact, showing large phase changes at the same frequency.

The identification of the frequencies, carried out with the algorithm *Quick Fit* of ModalVIEW, made it possible to find a natural frequency of 13.0 Hz, with a viscous damping ratio [126] of 2.3%. The corresponding mode of vibration, shown in Figure 54, is dominated by displacements in the $x-y$ plane caused by the rotation of joint 1, and the contribution of the other joints and of link flexibility to this mode appears negligible. Therefore, the identified frequency is suited to identify the stiffness of joint 1.

The results of the modal tests carried out in order to identify the compliance of joint 2 are presented in Figures 55 and 56. The overlay of the measured FRFs shows the first peak at about 18 Hz. Modal analysis made it possible to identify a mode of vibration with natural frequency 17.5 Hz and viscous damping ratio of 1.9%. The shape of this mode of vibration (Figure 56) is characterized by displacements



(a) Test 1

(b) Test 2

(c) Test 3

(d) Test 4

(e) Test 5

Figure 52: Robot configurations used to find the modal behavior of robot joints.

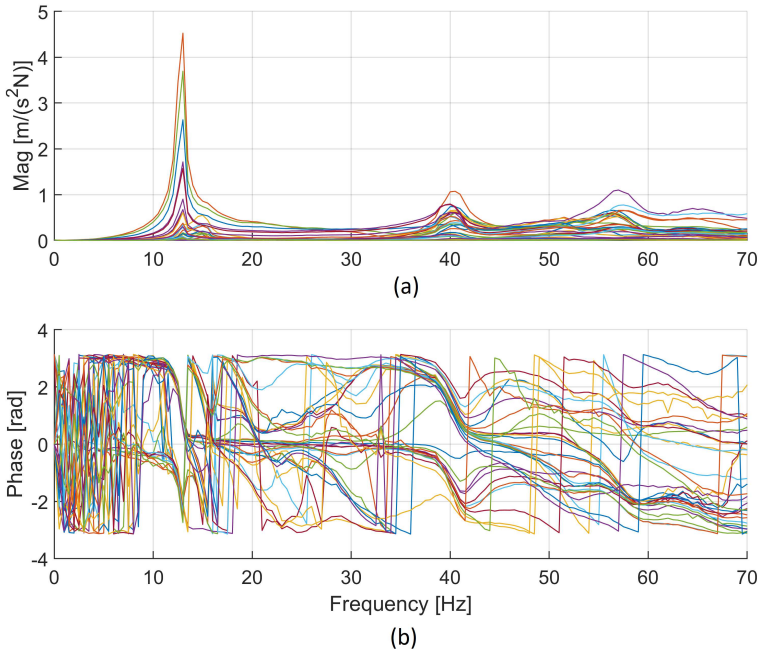


Figure 53: FRFs measured in the first test configuration (modulus (a) and phase (b)).

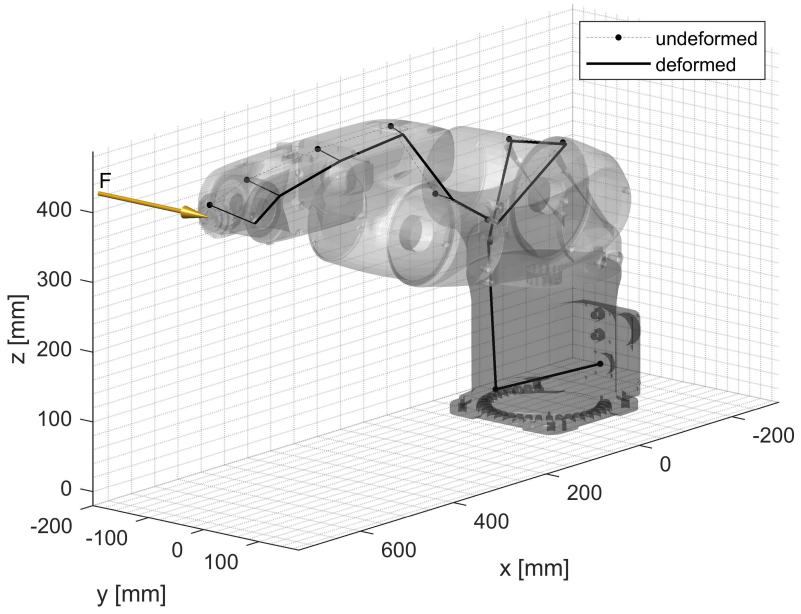


Figure 54: Mode at 13.0 Hz for the identification of joint 1 compliance.

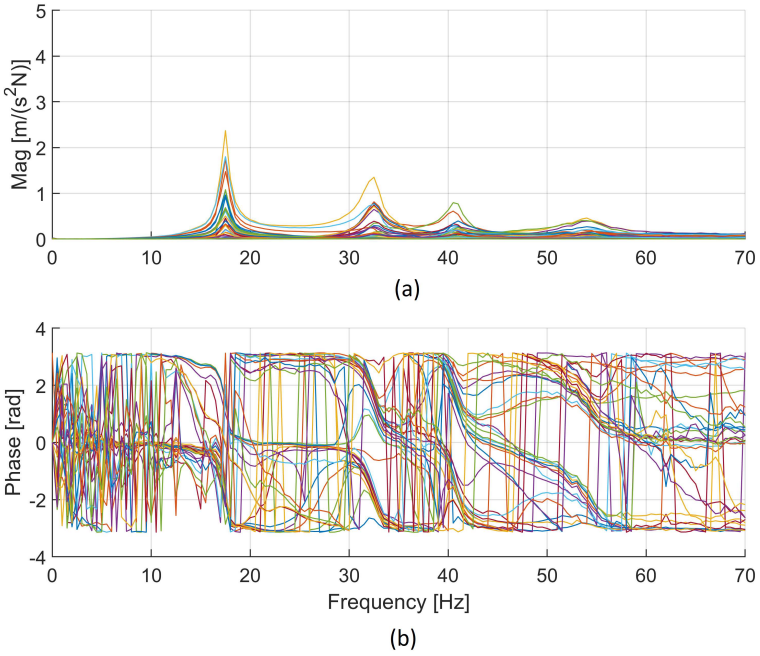


Figure 55: FRFs measured in the first test configuration (modulus (a) and phase (b)).

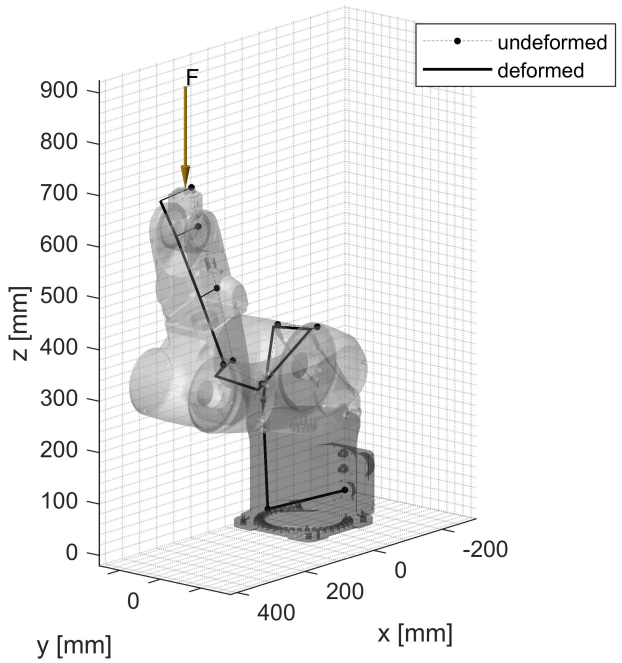


Figure 56: Mode at 17.5 Hz for the identification of joint 2 compliance.

dominated by the rotation of joint 2, therefore it is suited to identify the compliance of this joint.

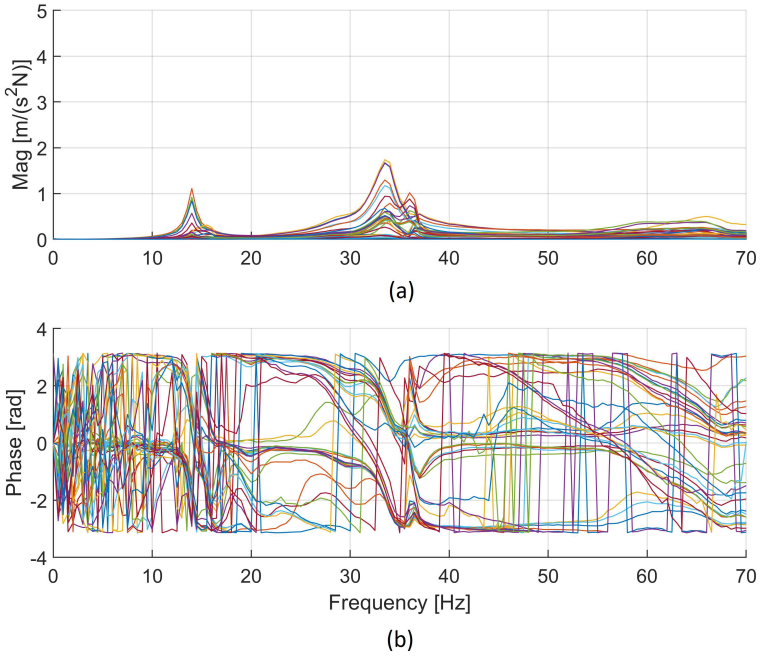


Figure 57: FRFs measured in the first test configuration (modulus (a) and phase (b)).

Figure 57 shows the FRFs measured in the third configuration, in order to identify the compliance of joint 3. Two resonance peaks appear. The first is at about 14 Hz and the second, which is the highest, is at about 33 Hz. Modal analysis results show that the first peak (the minor) corresponds to a mode dominated by the compliance of joint 2; this mode was excited since the hammer force did not intersect exactly the axis of joint 2. The mode at 33.5 Hz (with viscous damping ratio of 2.7%) is dominated by joint 3 compliance and is suited to identify the compliance of this joint (Figure 58).

The identification of the stiffness of joints 4 and 5 is more difficult: the lower moments of inertia greatly increase the natural frequencies of the last two joints, making it possible to reach high values, in which other aspects, different from the mechanical points of view, arise. Moreover, the low lever arms that can be applied to these joints result in low torques at the actuators, thus providing a low magnitude compared to other resonance peaks.

However, two possible resonance behaviors can be found in the FRFs of joints 4 (Figure 59) and joint 5 (Figure 60). The first is at about 88 Hz (damping of 9.4%) and is related to the compliance of joint 5: in Figure 60 a small peak in magnitude and a big change in phase can be found.

The second resonance is at about 92.5 Hz (damping of 2.7%) and is related to the compliance of joint 4: a rapid change of phase can be found in Figure 59

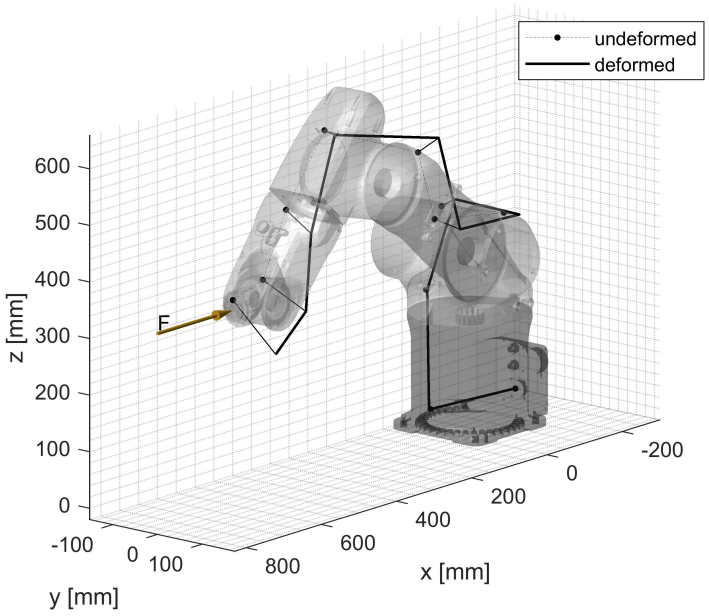


Figure 58: Mode at 33.5 Hz for the identification of joint 3 compliance.

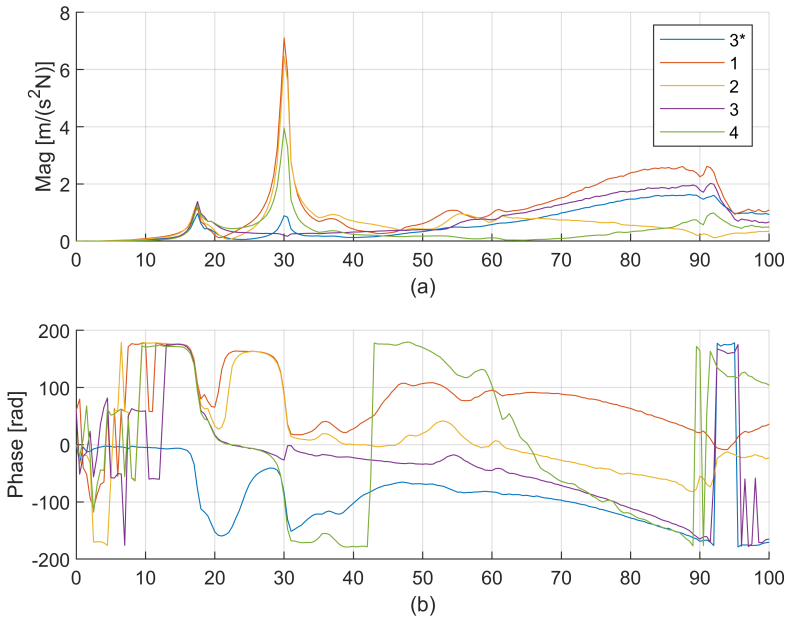


Figure 59: FRFs measured in the first test configuration (modulus (a) and phase (b)). Only the Y data retrieved from accelerometers 1, 2, 3, 3^* and 4 is shown.

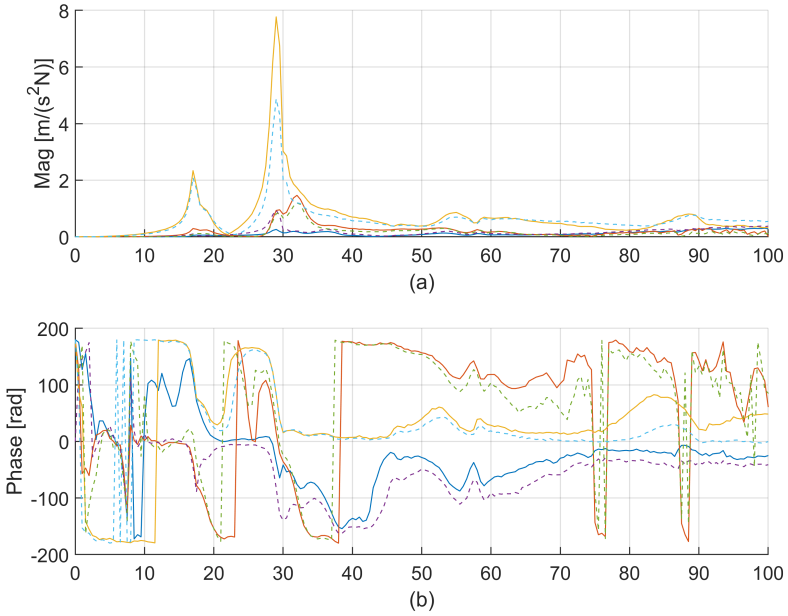


Figure 60: FRFs measured in the first test configuration (modulus (a) and phase (b)). Only the data retrieved from accelerometers 1 (continuous lines) and 2 (dashed lines) is shown.

Due to the low magnitudes of the last two resonances, it is impossible to clearly show the modes of joint 4 and 5. The presence of resonance at those close frequencies has been provided by studying the FRFs.

It is worth noticing that the damping ratios of the selected modes are similar; this is a good clue that highlights that these modes chiefly involve the joints, which have similar transmissions and servos that are characterized by similar dissipation phenomena. The only difference can be seen in joint 5: a possible solution to this high damping is due to the different type of transmission (using a flexible belt). Unfortunately, this hypothesis can be confirmed only after a complete disassemble of the robot.

The stiffnesses identified with the above-mentioned method are summarized in Table 13; the first three joints have similar stiffnesses, in agreement with literature values [121], while the last two joints are more compliant.

To conclude, in a static point of view, joints 4 and 5 cannot be neglected, since their low stiffness can produce high deformations. However, the low lever arms that the forces on the end effector can apply result in low deformations.

From a vibratory point of view, fortunately, the last two joints can be neglected: their high frequencies show that the most important joints in the modal behavior are the first three. In a real world appli-

Joint	Natural frequency [Hz]	Moment of inertia $[kg \cdot m^2]$	Stiffness $[Nm/rad]$	Compliance $[rad/(Nm)]$
1	13.0	1.990	12913	$7.74\ 10^{-5}$
2	17.5	0.809	9738	$1.03\ 10^{-4}$
3	33.5	0.229	10240	$9.76\ 10^{-5}$
4	92.5	0.005	1706	$5.86\ 10^{-4}$
5	88.0	0.001	336	$2.97\ 10^{-3}$

Table 13: Identified parameters.

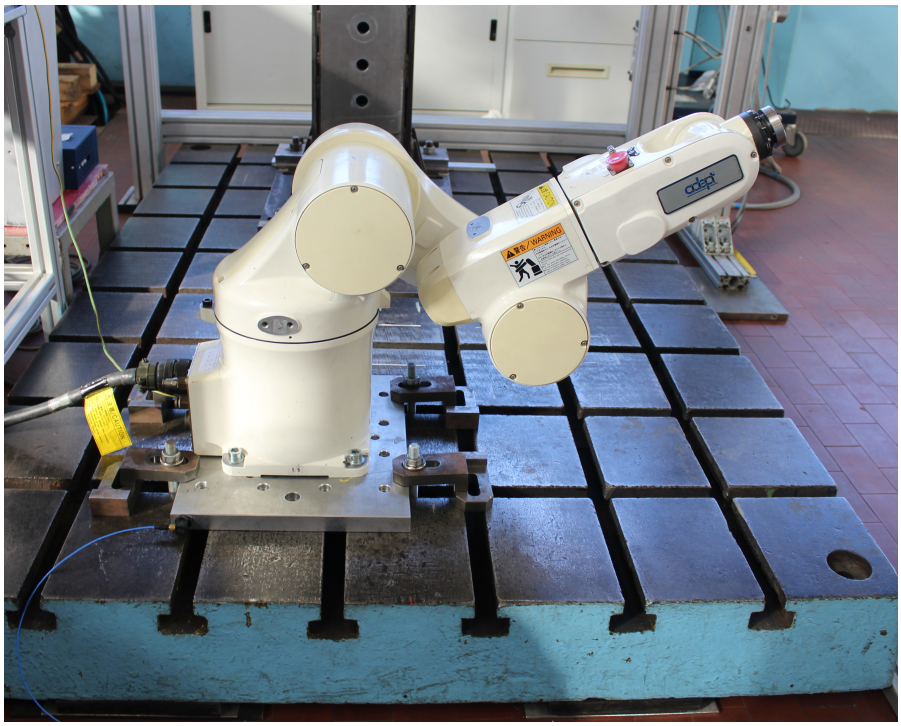


Figure 61: The robot in the validation configuration.

cations, the high frequency forces with high amplitude are unlikely to appear.

5.2.3 *Validation*

In order to assess the validity of the identification method, the stiffness values of Table 13 were implemented in a dynamic model of the robot and the first three natural frequencies of the robot were calculated in a configuration (validation configuration in Table 12) different from the configurations used for identification. This configuration is shown in Figure 61.

Natural frequency	Experimental x excitation [Hz]	Experimental y excitation [Hz]	Experimental z excitation [Hz]	Numerical [Hz]
1	16.0	16.0	16.0	15.0
2	-	18.0	17.5	16.5
3	37.0	36.5	35.5	39.0

Table 14: Validation tests.

Then, the robot was modally tested in the validation configuration and the first natural frequencies were experimentally identified. Hammer excitation was applied on the end-effector along three orthogonal directions of the end-effector coordinate system. It is worth noticing that the validation configuration was chosen in order to obtain modes of vibration that involved the compliance of various joints simultaneously.

The comparison between numerical and experimental values is shown in Table 14. The three testing conditions (with different directions of excitation) essentially lead to the identification of the same natural frequencies. These frequencies are rather close to the numerical ones, with a maximum error of about 3 Hz in the natural frequency of the third mode.

5.3 COMPLIANCE OF THE ROBOT ARM: THE MOZZI AXIS

When the robot has an end effector loaded by a static or dynamic force, the deformation of the loaded end can be studied considering the motion of a rigid body fixed to the loaded end of the structure. The basic concepts of rigid body mechanics can be adopted for describing this motion.

In general, the rigid motion of a body is represented in every instant by rotation about and translation along a specific axis, called the Mozzi axis [127][3] or the instantaneous screw axis. The direction of the Mozzi axis coincides with the instantaneous direction of the angular velocity vector.

The concept of the Mozzi axis has been further developed during the years [128, 129, 130] and has been applied to different research fields, from multibody dynamics [131] to vehicle dynamics [132, 133, 134] and vibration control [135]. Moreover, in robotics the Mozzi axis has been applied for decomposing the stiffness matrix of the structure [136] and for finding its properties [137, 138].

3 Its name comes from Giulio Giuseppe Mozzi del Garbo, an Italian mathematician who published in 1763 a book [127] in which he stated that a generic differential spatial rigid motion can be considered a helical motion around a line

It is clear that the Mozzi axis has a high potential in terms of feasibility. In this research, the concept of the Mozzi axis will be used [90] in a different way with respect to previous works: Mozzi axis will represent the rotation and translation axis of the compliant motion of the end-effector caused by an external load. This is particularly useful in the framework of this research because it allows, at a glance, to see the direction of most compliance of the end effector. This direction, along with the amplitude of the displacement, can provide a useful information regarding the best placement of the workpiece with respect to the end effector. It has to be noticed that the Mozzi axis is instantaneous, which means that it can be used in a dynamic analysis.

For this part of the research, since it has been proved that joints 4 and 5 are negligible from a vibration point of view, only the first three joints will be considered.

5.3.1 *Mathematical model*

To achieve my objective, the basic concepts of Mozzi or screw axis are sufficient. In Figure 62, a serial six-joint robot is represented. A fixed reference frame is established, whose origin and axes are, respectively, O_0 and $x_0y_0z_0$.

Joint rotations are fixed but compliance is allowed in the first three joints. Point Q is the center of the wrist. Since wrist joints negligible, link 3, the wrist and the end-effector form a unique rigid body, which is used for Mozzi axis analysis. Point O_t is the generic point in which a generic force is applied.

Thanks to the equation of rigid body velocity, the linear velocity $\vec{v}_P$ of a point P belonging to the rigid body can be calculated:

$$\vec{v}_P = \vec{v}_Q + \vec{\omega} \times \overrightarrow{QP} \tag{36}$$

where $\vec{v}_Q$ is the linear velocity of a reference point belonging to the rigid body (e.g. wrist center Q), $\vec{\omega}$ is the angular velocity vector, and $\overrightarrow{QP}$ is the vector from Q to P.

In the same way, linear velocity of point P can be expressed as a movement along the Mozzi axis (translation and rotation):

$$\vec{v}_P = k\vec{\omega} + \vec{\omega} \times \overrightarrow{MP} \tag{37}$$

where k is a scalar and M a point on the Mozzi axis. It is possible to merge Equations 36 and 37, obtaining as a result:

$$\vec{v}_Q + \vec{\omega} \times \overrightarrow{QM} = k\vec{\omega} \tag{38}$$

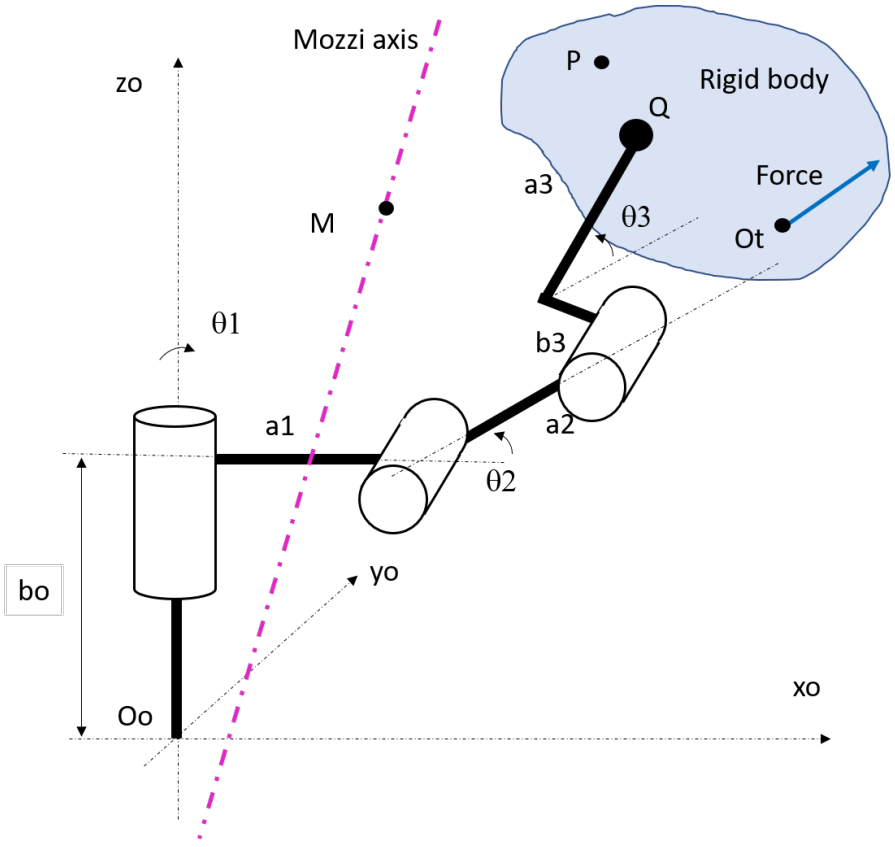


Figure 62: Kinematic model of the tested robot.

Due to the theory of Mozzi axis, linear velocity $\vec{v}_Q$ can be decomposed into two parts: the first parallel to angular velocity $\vec{\omega}$ (the translation along Mozzi axis), $\vec{v}_{Q\parallel}$, and the second perpendicular to angular velocity $\vec{\omega}$ (the rotation about Mozzi axis), $\vec{v}_{Q\perp}$. The equations hold:

$$\vec{v}_Q = \vec{v}_{Q\perp} + \vec{v}_{Q\parallel} \tag{39}$$

$$\vec{v}_{Q\parallel} = k\vec{\omega} = \vec{\omega} \cdot \vec{v}_Q \frac{\vec{\omega}}{|\omega|} \tag{40}$$

$$\vec{v}_{Q\perp} + \vec{\omega} \times \overrightarrow{QM} = 0 \tag{41}$$

$$\vec{\omega} \cdot \vec{v}_{Q\perp} = 0 \tag{42}$$

The system of Equations 41 and 42 holds only one unknown variable, $\overrightarrow{QM}$. There are infinite solutions for $\overrightarrow{QM}$, because the the angular velocity matrix is skew-symmetric [139]. However, by using an arbitrary scalar parameter μ it is possible to calculate all the possible solutions of $\overrightarrow{QM}$:

$$\overrightarrow{QM} = \frac{\vec{\omega} \times \vec{v}_{Q\perp}}{\omega^2} + \mu\vec{\omega} \tag{43}$$

From this solution, it is possible to define every point of the Mozzi axis with respect to the origin O_0:

$$\overrightarrow{O_0M} = \overrightarrow{O_0Q} + \overrightarrow{QM} \tag{44}$$

and, if all the vectors are projected on the fixed coordinate system, these scalar equations holds:

$$\begin{Bmatrix} O_0M_x \\ O_0M_y \\ O_0M_z \end{Bmatrix} = \begin{Bmatrix} O_0Q_x \\ O_0Q_y \\ O_0Q_z \end{Bmatrix} + \begin{Bmatrix} (\omega_x\mu)/|\omega| - (\omega_z v_{Qy} + \omega_y v_{Qz})/\omega^2 \\ (\omega_y\mu)/|\omega| - (\omega_z v_{Qx} + \omega_x v_{Qz})/\omega^2 \\ (\omega_z\mu)/|\omega| - (\omega_y v_{Qx} + \omega_x v_{Qy})/\omega^2 \end{Bmatrix} \tag{45}$$

This equation can also be written in terms of differential displacements (dQ_x, dQ_y, dQ_z) and rotations $(d\theta_x, d\theta_y, d\theta_z)$:

$$\begin{Bmatrix} O_0M_x \\ O_0M_y \\ O_0M_z \end{Bmatrix} = \begin{Bmatrix} O_0Q_x \\ O_0Q_y \\ O_0Q_z \end{Bmatrix} + \begin{Bmatrix} (d\theta_x\mu)/|d\theta| - (d\theta_z dQ_y + d\theta_y dQ_z)/d\theta^2 \\ (d\theta_y\mu)/|d\theta| - (d\theta_z dQ_x + d\theta_x dQ_z)/d\theta^2 \\ (d\theta_z\mu)/|d\theta| - (d\theta_y dQ_x + d\theta_x dQ_y)/d\theta^2 \end{Bmatrix}$$

(46)

By using the Jacobian matrix $\mathbf{J}$ and the stiffness matrix $\mathbf{K}$ it is possible to correlate the force vector $\{F\}$ (which includes both forces and torques applied to the end effector) to the deformation vector $\{\Delta x\}$ of the end effector:

$$\{\Delta x\} = \mathbf{J}\mathbf{K}^{-1}\mathbf{J}^{T}\{F\} \tag{47}$$

The representation of the compliance properties of the robot arm by means of the Mozzi axis has some advantages with respect to the representation based on the Cartesian stiffness matrix. First, the Mozzi axis gives a more intuitive and geometric interpretation of the phenomenon. Second, the Mozzi axis also suggests the origin of the compliance, since its proximity to a joint axis identifies this axis as the most compliant.

5.3.2 *Implementation of the Mozzi axis approach*

To show the potentialities of the Mozzi axis approach, multiple simulations, in two different scenarios, A and B, were performed. Both A and B configurations were excited by grinding forces along x_0, y_0, and z_0 axes. The two configurations differed in the orientation of the grinding equipment. From a generic point of view, these configurations show how the Mozzi axis changes with a different position of the grinding forces with the same robot joint configuration. Finally, a real case machining scenario [114] was simulated (Figure 69).

In Figures 63, 65, and 68, the robot is excited by a machining force that lies in the vertical plane perpendicular to the axes of joints 2 and 3. The rotation vector of the end-effector is parallel to the axes of joints 2 and 3, therefore the Mozzi axis has the same direction and passes at a distance from the robot base that depends on the compliances of joints 2 and 3. No translation along the Mozzi axis takes place. It is worth noticing that, since in Figure 63 the force has a small lever arm with respect to joint 2, the motion of the end-effector is dominated by compliance of joint 3 and the Mozzi axis passes very close to the center of this joint. In Figures 65 and 68, the vertical force exerts moments about both joint 2 and 3 and the Mozzi axis crosses the robot at an intermediate point.

In Figures 64 and 67, the robot is excited by machining forces parallel to the y_0 axis. Since this axis is parallel to the joint 2 and 3 axes, the only joint that can comply to the machining force is joint 1, therefore the Mozzi axis is coincident with the axis of joint 1.

Figure 66 shows how the compliance of all the joints can combine together when excited by an eccentric force. Force $\vec{F}$ is applied parallel to the x_0 axis on a plane perpendicular to the axes of joints 2 and

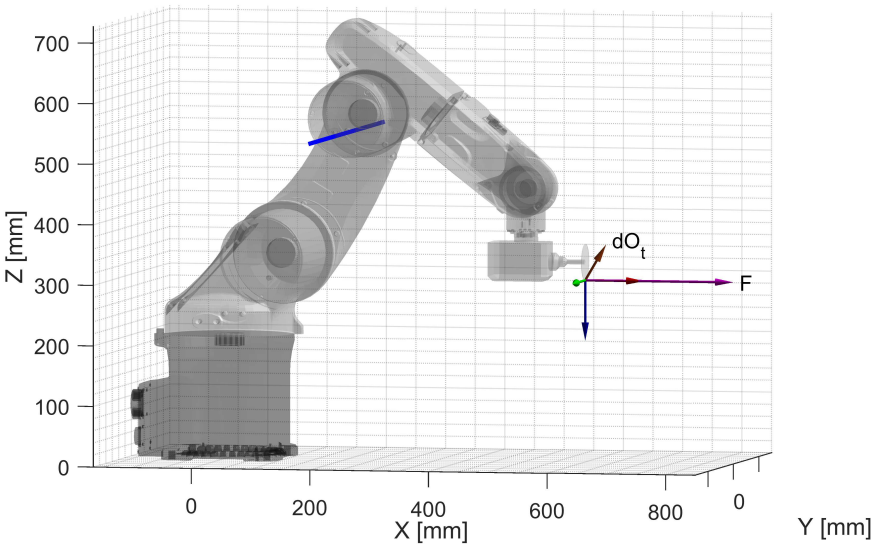


Figure 63: Configuration A with force $\vec{F}$ applied parallel to x_0 axis.

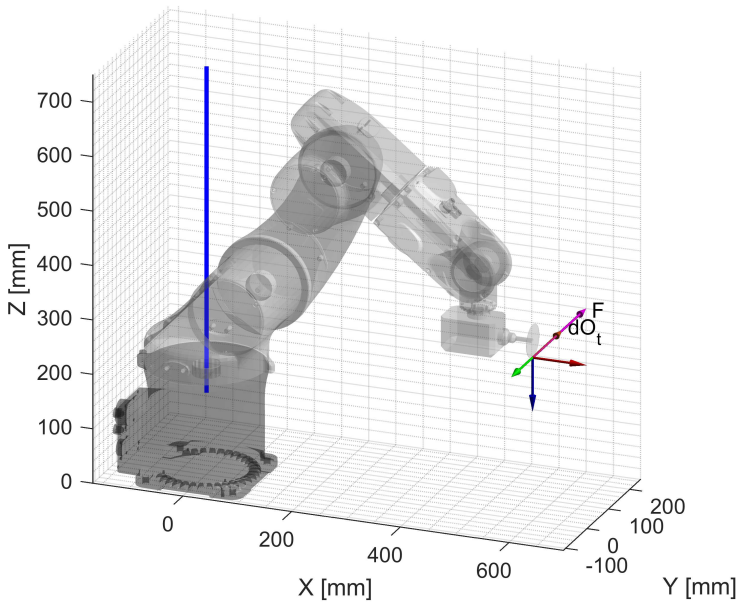


Figure 64: Configuration A with force $\vec{F}$ applied parallel to y_0 axis.

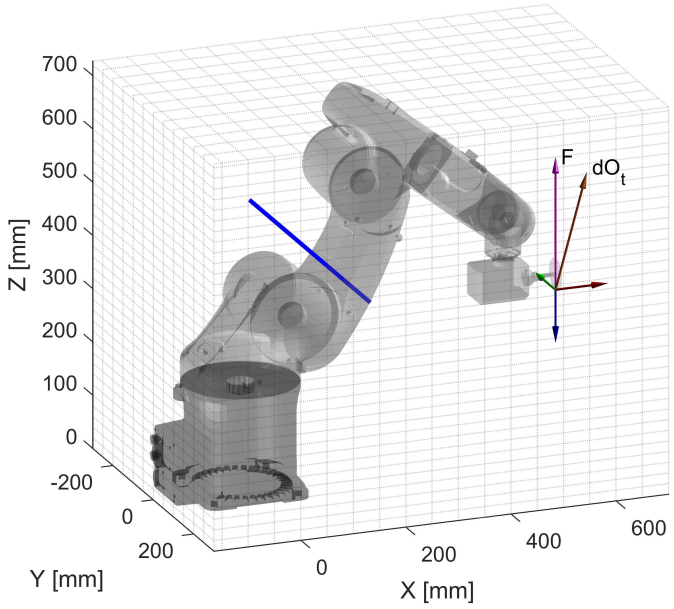


Figure 65: Configuration A with force $\vec{F}$ applied parallel to z_0 axis, bottom view.

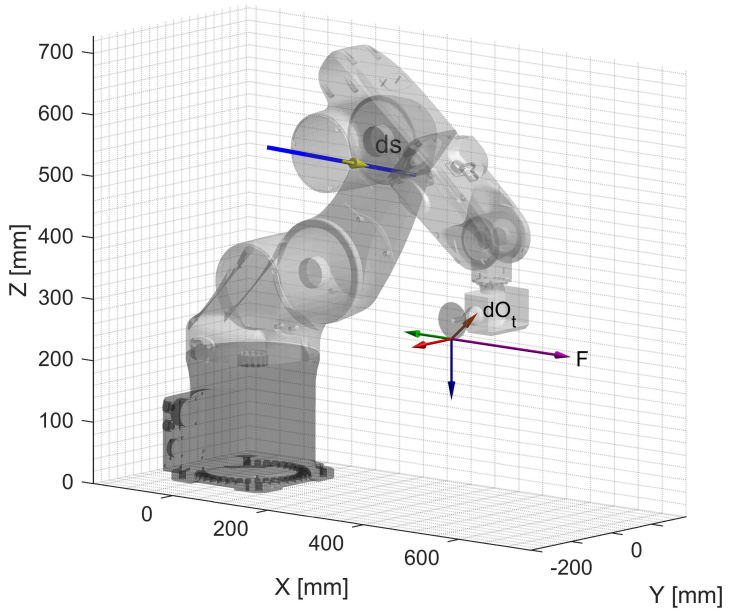


Figure 66: Configuration B with force $\vec{F}$ applied parallel to x_0 axis.

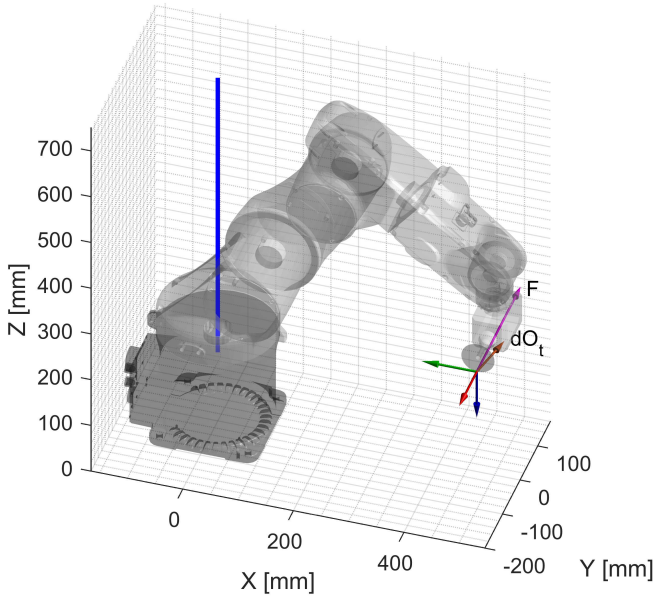


Figure 67: Configuration B with force $\vec{F}$ applied parallel to y_0 axis.

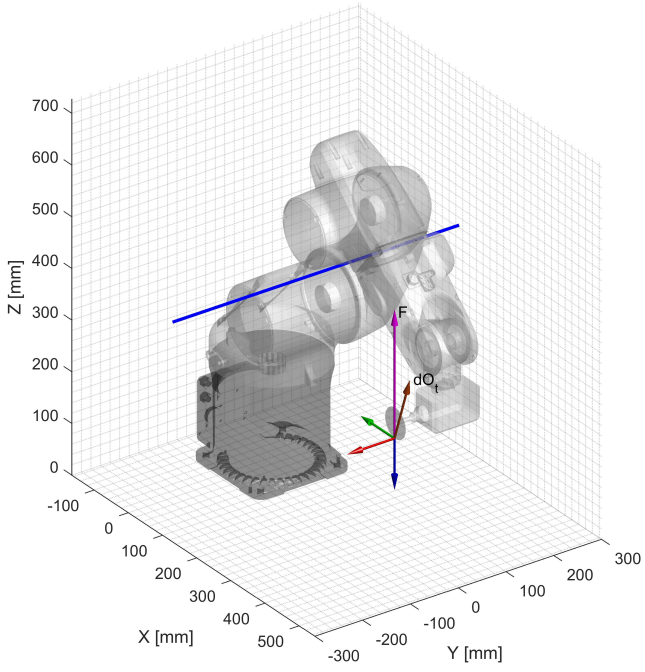


Figure 68: Configuration B with force $\vec{F}$ applied parallel to z_0 axis.

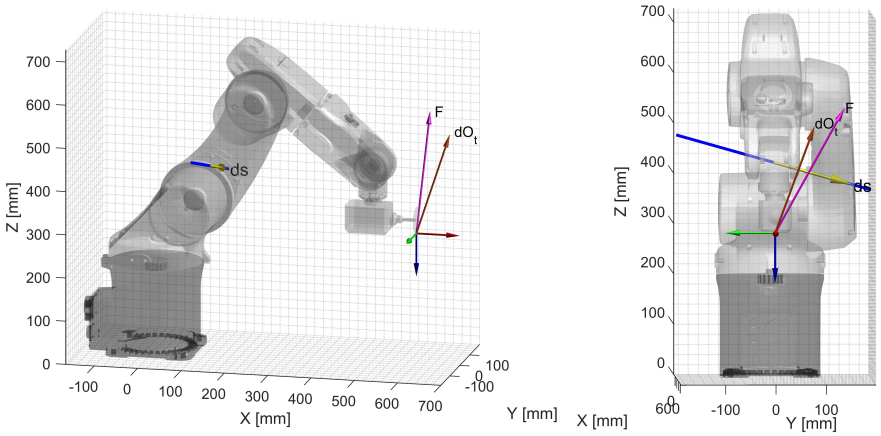


Figure 69: Configuration A with force $\vec{F}$ applied on a plane parallel to $y_0 - z_0$ plane, two different points of view.

3, which does not intersect the axis of joint 1. Thus, it excites all the joints and the position of the Mozzi axis results from the combination of all joint compliances. The Mozzi axis is tilted with respect to plane $x_0 - y_0$, and there is a displacement $\overrightarrow{ds}$ along the axis as well.

In Figure 69, a real scenario simulation is shown. Force $\vec{F}$ lies on a plane parallel to the $y_0 - z_0$ plane and its direction is defined by the downmilling process described in [114]. In this case, the Mozzi axis has a location similar to the one in Figure 66: the main contribution to the end-effector movement is provided by the compliance of joints 2 and 3, but a small component of force $\vec{F}$ along the y_0 axis results in a non-negligible compliance of joint 1. It is interesting to notice how the high lever arm with respect to joint 1 results in a significant compliance even with a small exciting force.

Another important aspect that can be highlighted by the seven figures is the direction of the Cartesian movement $\overrightarrow{dO_t}$ of the point of application of the force. This direction is perpendicular to the Mozzi axis when displacement $\overrightarrow{ds}$ along the axis is null and a combination of both rotational and sliding displacements in the general case. Moreover, the direction is not related to the direction of force $\vec{F}$, but it depends on the compliance of the joints. The only relation that can be found between the two vectors is that this movement $\overrightarrow{dO_t}$ is placed within the semispace defined by force vector $\vec{F}$.

Starting from this approach, the design of the end effector can be performed by taking into account the movement of the end effector with respect to the direction of the deburring forces. The configuration of the robot (considering the redundant axis, coincident with the spindle axis) can be adapted in such a way that the resulting compliance direction (described by the Mozzi axis) is allowed by our task. For example, it could be possible to design the end effector in such a way that the main direction of compliance is coincident with the

burr direction: in this way, no overshoots in the direction normal to the surface would appear, resulting in a good surface finishing.

5.4 UNDER-ACTUATED SYSTEMS: CAN THEY BE A GOOD SOLUTION TO THE VIBRATORY PROBLEM?

Under-actuated robots are well-known solutions for applications where the weight of the manipulator or the production costs are priority features. By applying proper design choices it is possible to perform the same task of fully actuated systems with fewer actuators.

However, there are some drawbacks to using under-actuated systems: while a fully-actuated robot can perform any joint trajectories (within the boundaries), for an under-actuated system, not all joint trajectories are attainable. This means that the controllability of the system can be lost.

Fortunately, if a system is *differentially flat*, simple point-to-point movements can be executed [140, 141]. This aspect is crucial in industrial applications where most of the tasks are performed by simple movements.

It is possible to demonstrate that for an n-DOF arm system a proper design can achieve differential flatness [142, 143, 144], so adopting these design rules should allow applying under-actuated robots in an industrial environment. However, the uncertainty of the movement between the start and end positions could be a problem in certain applications, e.g. where the environment is cluttered with obstacles.

The aim of this Section is to evaluate how an under-actuated system can be compared to a fully-actuated system. As an example, I provide a comparison between a 3 DOF under-actuated planar manipulator with 2 active joints followed by one passive joint and a 2 DOF fully actuated system whose second link is equivalent, in term of length and mass distribution, to the sum of the last two links of the under-actuated system.

The comparison will consider three aspects: trajectories, actuators torques, and vibrations. These three factors can be crucial in the design of a system: the trajectories can be important in a cluttered environment, actuator torques matter in the design of lightweight structures and reduction of vibration is crucial to gain functionality in applications.

5.4.1 *Mathematical and simulation models*

The under-actuated serial robot to be studied in this paper is made of three rotational joints, the first of which can rotate around a fixed point. The first two joints are active, driven by rotational actuators, while the last one is passive and equipped with a torsional spring whose stiffness is k_3 (Figure 70a) [145]. The comparable fully-actuated

robot is made of two active rotational joints (Figure 70b), its first link is inertial equivalent to the first link of the under-actuated system; its second link is equivalent to the sum of the second and third link of the under-actuated system. In other words, fixing the under-actuated third joint to the $q_3 = 0$ position results in the fully-actuated system of Figure 70b. In this work all the parameters of the fully actuated system will be labeled by an apostrophe to distinguish them from the parameters of the under-actuated system.

The dynamic model of the planar manipulator can be developed based on [146] using the Lagrange equations:

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{\mathbf{q}}}\right) - \frac{\delta L}{\delta \mathbf{q}} = \boldsymbol{\tau} \tag{48}$$

where L is the Lagrangian, $\mathbf{q}$ is the vector of joint variables and $\boldsymbol{\tau}$ is the vector of joint torques.

Into the kinetic energy formulation only the link movement is considered. Motor's kinetic energy should be considered within the dissertation [146], but in most of the industrial robots it is difficult or impossible to obtain the inertial data of the actuators from experiments; as a result, motor are included in the inertial properties of the corresponding link. Kinetic energy T can then be calculated as the sum of link's kinetic energy T_i:

$$T_i = \frac{1}{2} m_i \dot{\mathbf{q}}^T \mathbf{J}_P^{(l_i)T} \mathbf{J}_P^{(l_i)} \dot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{J}_O^{(l_i)T} \mathbf{R}_i \mathbf{I}_i \mathbf{R}_i^T \mathbf{J}_O^{(l_i)} \dot{\mathbf{q}} \quad \rightarrow \quad T = \sum_{i=1}^{n} T_i \tag{49}$$

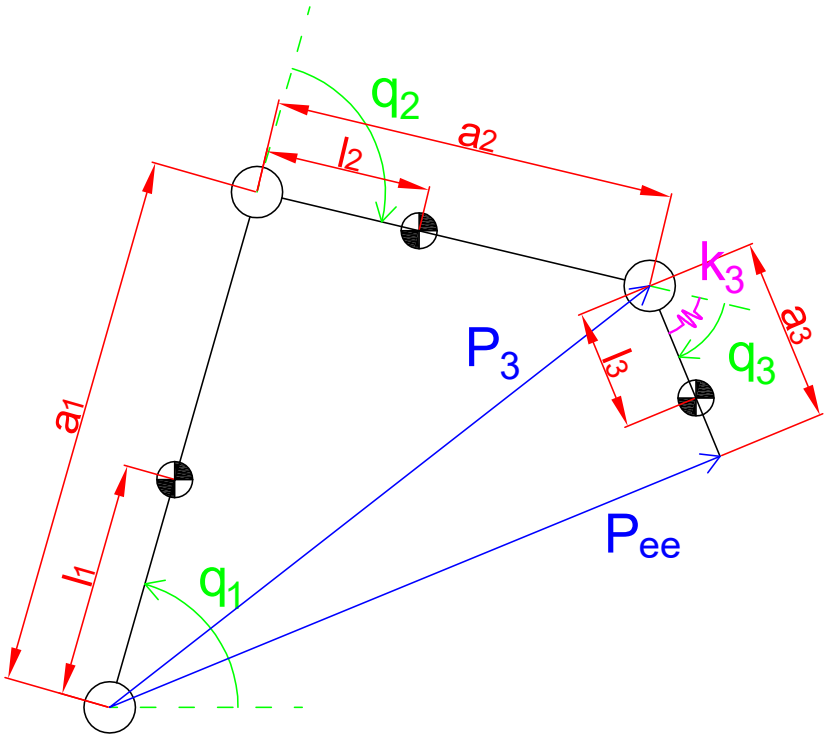
where m_i and $\mathbf{I}_i$ are the mass and inertia matrix of i-th link with respect to their center of mass, $\mathbf{R}_i$ is the rotational matrix that transforms origin frame to the i-th link frame and $\mathbf{J}_P^{(l_i)}$ and $\mathbf{J}_O^{(l_i)}$ are the partial Jacobians calculated as follows:

$$\mathbf{J}_P^{(l_i)} = \begin{bmatrix} J_{P1}^{(l_i)} \cdots J_{Pi}^{(l_i)} \; 0 \cdots 0 \end{bmatrix} \quad , \quad J_{Pj}^{(l_i)} = \mathbf{z}_j \times \left(\mathbf{p}_{l_i} - \mathbf{p}_j \right) \tag{50}$$

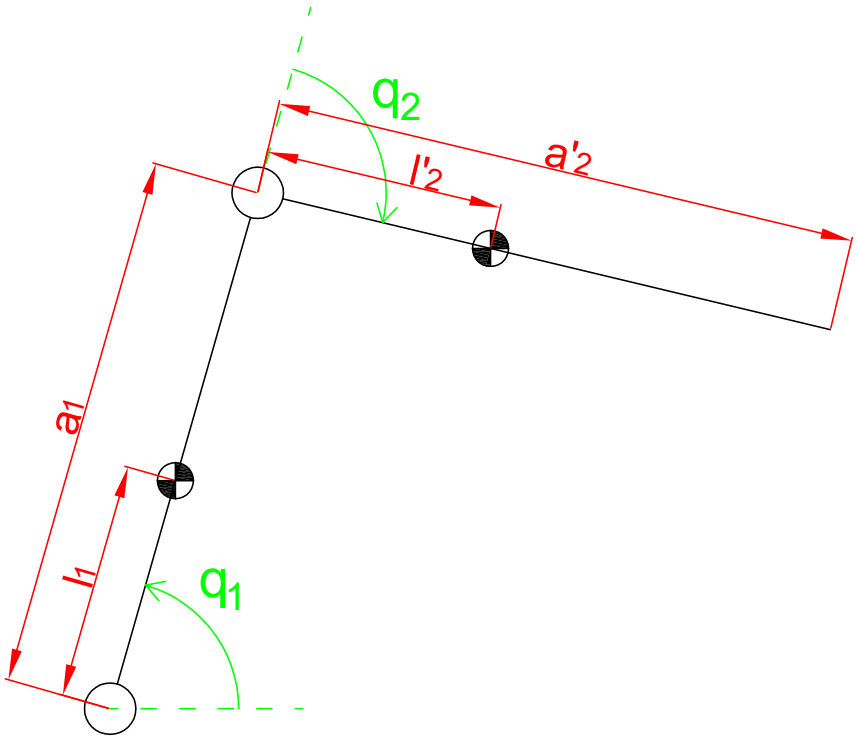
$$\mathbf{J}_O^{(l_i)} = \begin{bmatrix} J_{O1}^{(l_i)} \cdots J_{Oi}^{(l_i)} \; 0 \cdots 0 \end{bmatrix} \quad , \quad J_{Oj}^{(l_i)} = \mathbf{z}_j \tag{51}$$

where $\mathbf{z}_j$ is j-th joint's axis unit vector, $\mathbf{p}_{l_i}$ defines i-th link's center of mass coordinates and $\mathbf{p}_j$ defines j-th link's origin coordinates.

While joints are considered to be compliant with a specific stiffness k_i, links are considered rigid. This is reasonable since it has been demonstrated that joint's stiffness is way lower than link's stiffness, thus providing higher deformations of the structure [117].



(a) Under-actuated robot



(b) Fully-actuated robot

Figure 70: Mathematical models of the considered robots.

Potential energy U is:

$$\begin{aligned} U_{gi} &= m_i \mathbf{g} \mathbf{p}_{l_i} \\ U_{ki} &= \frac{1}{2} k_i \mathbf{\Delta q} \end{aligned} \quad \rightarrow \quad U = \sum_{i=1}^{n} (U_{gi} + U_{ki}) \tag{52}$$

Joint stiffness is identified by the mechanical connection between motor and gear train for the active joints and by the torsional spring k_3 for the passive joint. This means that $\mathbf{\Delta q}$ identifies the position error for the active joints and is equal to q_3 for the passive joint. Since position errors for the active joints are very small and are compensated by the control system, in a movement their stiffness can be neglected. However, in a vibratory study this is not true: the external forces usually create small movements, and these movements cannot be compensated by robot controller.

Developing Equation 48 results in the state-space equations:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} \tag{53}$$

where the coefficients of the matrices are shown in Appendix A.2. For the under-actuated system, last link's inertial properties are driven by specific requirements [142, 143, 144] that can ensure controllability as described in Section 5.4.1. In particular, it has to be that last link's center of mass is to be coincident with last joint's rotation axis ($l_3 = 0$).

All the previous equations can be used to define the mathematical model of both the under-actuated and fully-actuated models. However, the former is described by three equations, while the latter is described by only two equations. This aspect clearly shows why the 3 DOF system is under-actuated: while there are three joint values (q_1, q_2 and q_3), only two actuators are used (τ_1 and τ_2).

Natural frequencies

From equation 53 it is possible to calculate the natural frequencies of the two systems. The damping matrix $\mathbf{C}$ is usually not null, thus the calculation of the natural frequencies is not easy to perform: one solution is described in [147] by using the method shown in [148]. However, how it will be described in Section 5.4.2, $\mathbf{C}$ contribution can be neglected, thus the natural frequencies ω can be calculated easily from the eigenvalue problem:

$$\det(\mathbf{K} - \omega^2 \mathbf{M}) = 0 \tag{54}$$

As stated in Appendix A.2, inertia matrix $\mathbf{M}$ is configuration dependent. In particular, $\mathbf{M}$ depends on q_2 and $\mathbf{K}$ is constant, so the natural frequencies will be a function of q_2.

Diffeomorphism

In general, for an under-actuated robot, not all outputs can be controlled arbitrarily. However, by using the differential flatness paradigm it is possible to define some outputs, called *flat outputs*, that can be controlled arbitrarily. Using these flat outputs, it is possible to determine the system states. This operation is called *diffeomorphism*.

Agrawal et al. [142, 143, 144] have already presented the design requirements that can make an n-DOF under-actuated planar manipulator differentially flat. In particular, as described in Section 5.4.1, the center of mass of the last link has to be placed coincident with last joint's axis ($l_3 = 0$). Moreover, a torsional spring has to be installed on the last joint [149, 150]: otherwise, the system would not be able to preserve its controllability [142].

Starting from the third state-space equation of Equation 53, since $\tau_3 = 0$ due to the lack of an active actuator:

$$M_{31}\ddot{q}_1 + M_{32}\ddot{q}_2 + M_{33}\ddot{q}_3 + k_3 q_3 = 0 \quad \rightarrow \quad q_3 = -\frac{I_3\,(\ddot{q}_1 + \ddot{q}_2 + \ddot{q}_3)}{k_3} \tag{55}$$

From this equation it is possible to define the following diffeomorphism [145] between the state inputs and the flat outputs:

$$\mathbf{y} = [y_1, y_2] = [q_1, q_1 + q_2 + q_3] \tag{56}$$

That bounds the state

$$\begin{cases} q_1 = y_1 \\ q_2 = y_2 - y_1 + \frac{I_3 \ddot{y}_2}{k_3} \\ q_3 = -\frac{I_3 \ddot{y}_2}{k_3} \end{cases} \rightarrow \begin{cases} \dot{q} = \dot{y}_1 \\ \dot{q}_2 = \dot{y}_2 - \dot{y}_1 + \frac{I_3 y_2^{(3)}}{k_3} \\ \dot{q}_3 = -\frac{I_3 y_2^{(3)}}{k_3} \end{cases} \rightarrow \begin{cases} \ddot{q} = \ddot{y}_1 \\ \ddot{q}_2 = \ddot{y}_2 - \ddot{y}_1 + \frac{I_3 y_2^{(4)}}{k_3} \\ \ddot{q}_3 = -\frac{I_3 y_2^{(4)}}{k_3} \end{cases} \tag{57}$$

By applying the diffeomorphism to the state space equations:

$$\mathbf{M}'(\mathbf{y}) \begin{pmatrix} \ddot{y}_1 \\ y_2^{(4)} \end{pmatrix} + \mathbf{C}'(\dot{y}_1, \dot{y}_2, \ddot{y}_2, y_2^{(3)}) \begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \ddot{y}_2 \\ y_2^{(3)} \end{pmatrix} + \mathbf{K}' \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \mathbf{G}(\mathbf{y}) = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} \tag{58}$$

where the coefficients of the matrix $\mathbf{M}' \in \mathbb{R}^{2\times 2}$ are:

$$\begin{cases} M'_{11} = I_1 + a_1^2(m_2 + m_3) + l_1^2 m_1 + j_1 \\ M'_{12} = \frac{I_3}{k_3}\left(I_2 + m_3 a_2^2 + m_2 l_2^2 + j_1\right) \\ M'_{21} = j_1 \\ M'_{22} = \frac{I_3}{k_3}\left(I_2 + m_3 a_2^2 + m_2 l_2^2\right) \end{cases} \tag{59}$$

the term $\mathbf{C}' \in \mathbb{R}^{2\times 4}$ is:

$$\begin{cases} C'_{11} = C'_{21} = j_2 \dot{y}_1 \\ C'_{12} = -j_2\left(\dot{y}_2 + y_2^{(3)} I_3/k_3\right) \\ C'_{22} = C'_{24} = 0 \\ C'_{13} = I_2 + I_3 + m_3 a_2^2 + m_2 l_2^2 + j_1 \\ C'_{23} = I_2 + I_3 + m_3 a_2^2 + m_2 l_2^2 \\ C'_{14} = -j_2 I_3/k_3\left(\dot{y}_2 + y_2^{(3)} I_3/k_3\right) \end{cases} \tag{60}$$

the new stiffness matrix $\mathbf{K}'$ is null due to the lack of torsional springs on the first two joints, whereas the term $\mathbf{G}' \in \mathbb{R}^{2\times 1}$ is:

$$\mathbf{G}' = \begin{bmatrix} g\left[(a_1 m_2 + a_1 m_3 + l_1 m_1)\cos(y_1) + (a_2 m_3 + l_2 m_2)\cos(y_2 + I_3 \ddot{y}_2/k_3)\right] \\ g(a_2 m_3 + l_2 m_2)\cos(y_2 + I_3 \ddot{y}_2/k_3) \end{bmatrix} \tag{61}$$

with $j_1 = a_1 \cos(y_2 - y_1 + I_3 \ddot{y}_2/k_3)(a_2 m_3 + l_2 m_2)$ and $j_2 = a_1 \sin(y_2 - y_1 + I_3 \ddot{y}_2/k_3)(a_2 m_3 + l_2 m_2)$.

Since the form of the flat inputs $\mathbf{y}$ can be chosen arbitrarily, the system is fully controllable and can follow any output trajectory. The trajectory depends on the starting and ending positions defined by a defined configuration in joint space $\mathbf{q}$, but PTP movements can be planned within the flat input space and result in flat output space.

Simulation models

The two models to be compared are considered to be made by rigid links, and each link can be simplified as a simple rod with distributed mass. The inertial properties of the two systems are calculated as shown in Table 15. It is important to notice that third link inertia I_3 is calculated about center (since $l_3 = 0$ and the mass is uniformly distributed along the rod) while I_1 and I_2 are calculated about the end connected to the corresponding joint.

The comparable 2 DOF system is equal to the 3 DOF system with the third joint fixed to a null position ($q_3 = 0$), thus the new second link inertia I_2' can be easily calculated using Huygens-Steiner theorem.

5.4.2 *Model comparison*

A designer may be interested in finding if an under-actuated robot suits better the design needs than a fully actuated robot. Apart from robot design (whose parameters can be optimized as shown in [149]), it can be interesting to know how two dynamically comparable robots differ considering specific aspects.

To find out the different behaviors of the two systems, a comparison between them has been performed. The main focuses of the comparison are the Cartesian trajectory of the end-effector (placed on the end of the last link), the actuator torques and the robot vibrations.

End-effector trajectories and actuator torques

Differentially flat under-actuated robots can perform PTP movements, but the trajectory that the robot arm is going to perform is uncontrollable. In fact, after the design of the flat inputs $\mathbf{y}$ the system is able to follow the output trajectory, but the joint values $\mathbf{q}$ depends solely on the flat inputs. On the other hand, a fully actuated robot can be controlled by defining specific joint functions $\mathbf{q}$.

Let's consider a representative PTP movement from position A to position B. Both these positions can be defined in Cartesian or joint space. In the first case, inverse kinematics has to be performed in A and B to find out joint values $\mathbf{q}_A = [q_{1A}, q_{2A}, 0]$ and $\mathbf{q}_B = [q_{1B}, q_{2B}, 0]$. These joint values can then be used to find the boundary conditions for the flat inputs of the under-actuated robot. It is important to notice that, since this is a PTP movement, the robot will move from A and will arrive in B both with null velocities, so the third joint value q_3 will be null in both locations. As a result, at these locations the two systems are equivalent from a kinematic point of view ($q_{1A} = q_{1A}'$, $q_{1B} = q_{1B}'$, $q_{2A} = q_{2A}'$ and $q_{2B} = q_{2B}'$).

For our specific example, in Table 16 the poses are outlined, while in Table 17 the inertial parameters are listed.

The path followed by an under-actuated robot depends on the movement time T_f: the flat functions that interpolate the boundary conditions depend on the overall movement time. Without considering actuator torques limits, different movement times have been studied.

The mathematical models have been implemented within a Matlab® script. The results with $T_f = \{0.5, 2, 5\}[s]$, are shown in Figures 71, 72, 73 and 74. It is clear how high movement times result in very similar trajectories and behavior: in Figures 72 and 73 the passive joint

Parameter	**Formulation**
3 DOF System	
l_1	$a_1/2$
I_1	$m_1 a_1^2/3$
l_2	$a_2/2$
I_2	$m_2 a_2^2/3$
l_3	0
I_3	$m_3 a_3^2/12$
2 DOF System	
l_1	$a_1/2$
I_1	$m_1 a_1^2/3$
l_2'	$(a_2 m_3 + l_2 m_2)/(m_2 + m_3)$
I_2'	$I_2 + I_3 + m_3 a_2^2$

Table 15: Inertial parameter formulation

	Value [°]
q_{1A}, q_{1A}'	0
q_{2A}, q_{2A}'	-120
q_{3A}	0
q_{1B}, q_{1B}'	90
q_{2B}, q_{2B}'	30
q_{3B}	0

Table 16: Parameters defining the PTP movement

	Value	Unit
m_1	3	kg
m_2	2	kg
m_3	3	kg
a_1	0.5	m
a_2	0.5	m
a_3	0.4	m
k_1	10^4	Nm/rad
k_2	10^4	Nm/rad
k_3	2	Nm/rad

Table 17: Inertial parameters of the manipulator

fluctuates around the null position by small values, and the torque effort required for the actuators is very similar. However, with very low movement times, the under-actuated system seems to be not very reliable: joint torques are high compared to the fully actuated system, and the passive joint swings by high values around the equilibrium position (Figure 71). This is due to the high accelerations and speeds required by the flat outputs to perform the trajectory in so little time and fully stop at position B.

The information provided by the graphs in Figures 71, 72 and 73 clearly reflects in Figure 74: with low movement times (such as $T_f = 0.5s$) the trajectory of the end-effector of the under-actuated robot is not simple, tangling in the middle. However, even small increases in the movement times (e.g. $T_f = 0.8s$) the trajectory is way more reliable.

The real advantage of the under-actuated system with respect to the fully-actuated system is the flexibility: even if the fully-actuated system provides a more reliable trajectory, the under-actuated system, with the additional degree of freedom, is able to move around any obstacles with less effort. This important aspect can be integrated in the system simply by unlocking the last joint and installing a torsional spring.

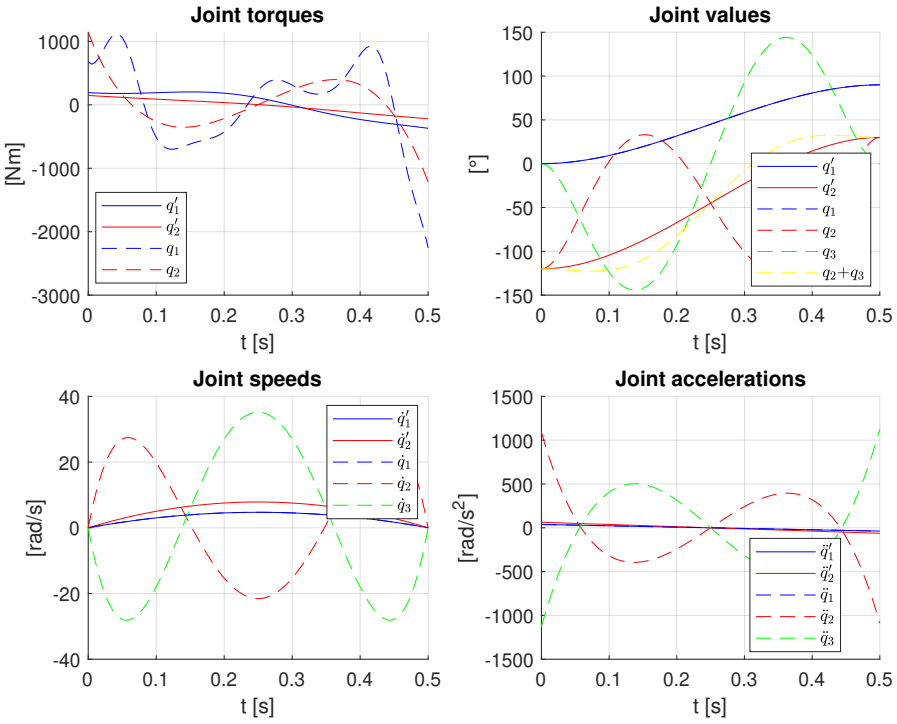


Figure 71: Simulation results with $T_f = 0.5s$.

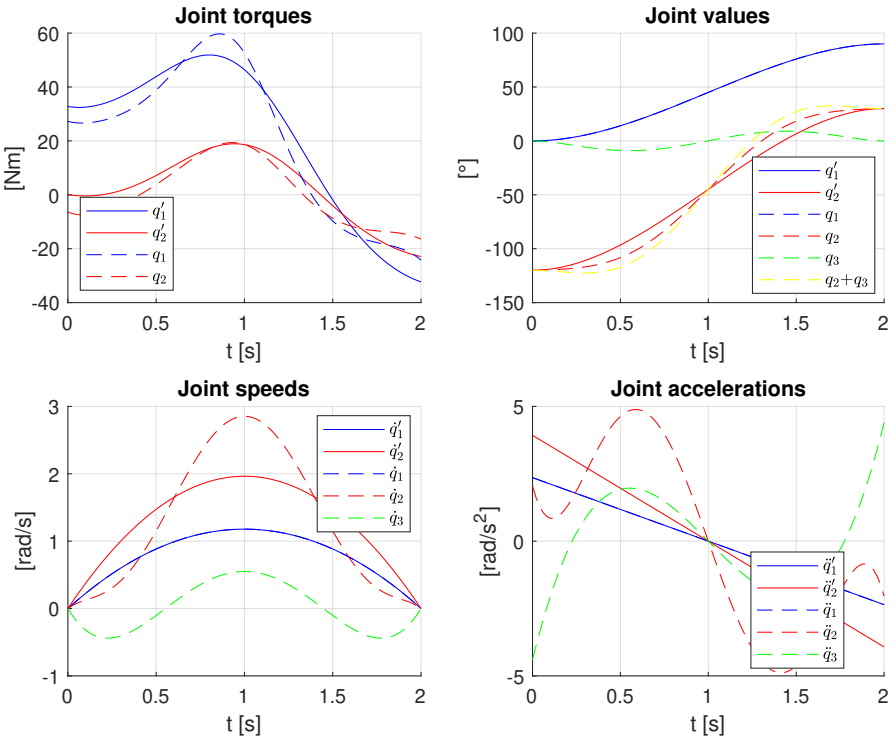


Figure 72: Simulation results with $T_f = 2s$.

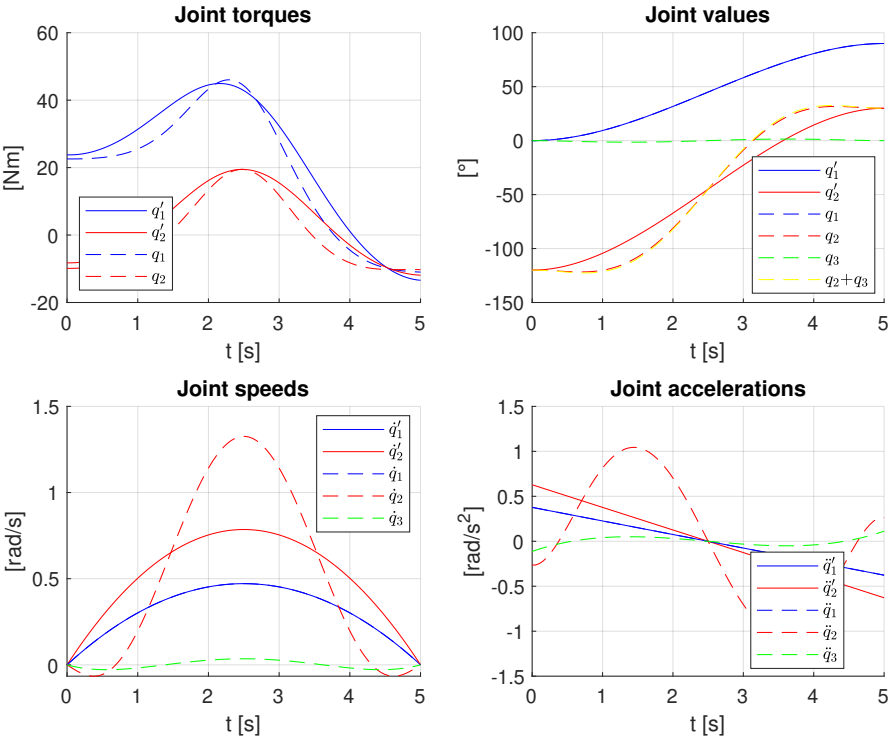


Figure 73: Simulation results with $T_f = 5s$.

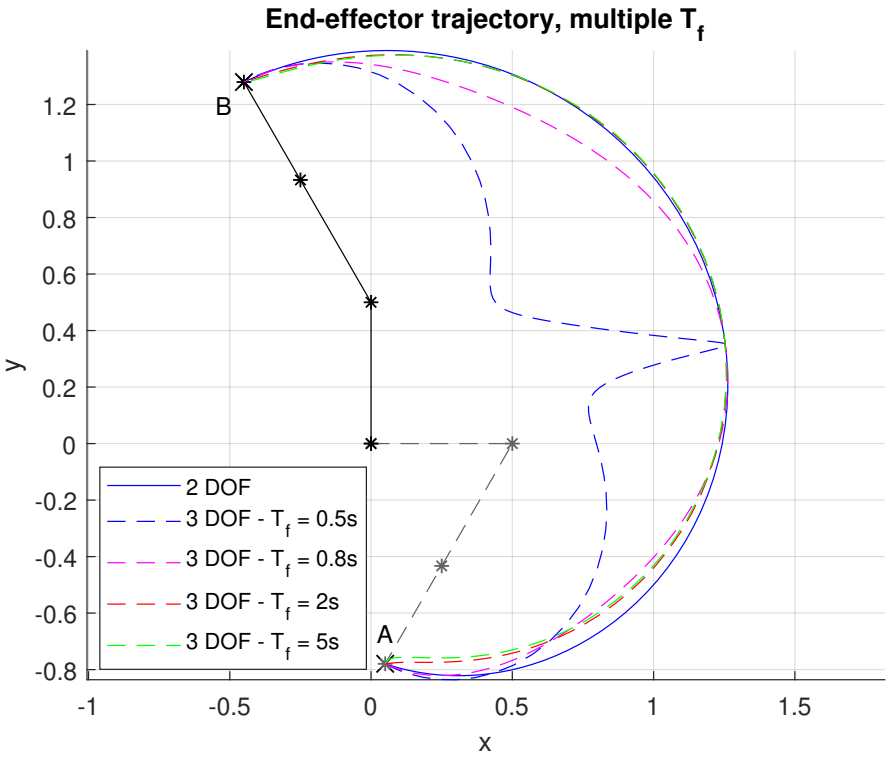


Figure 74: End-effector trajectory depends on the movement time.

Robot vibrations

Following the method described in Section 5.4.1, a comparison between the under-actuated and the fully actuated system has been performed.

As stated before, matrix $\mathbf{M}$ depends on the second joint value q_2, so, following Equation 54, also the natural frequencies ω depend on q_2. The dependency of these frequencies with respect to q_2 is shown in Figure 75.

Analyzing the figure, it is clear how the under-actuated system has a drawback: an additional resonance frequency is added (green in Figure 75). This frequency is nearly configuration independent since the passive joint is way less stiff than the active ones, so its value is nearly the same as a single degree of freedom structure with the same properties:

$$\omega_{1,3DOF} \approx \sqrt{\frac{k_3}{I_3}} \tag{62}$$

As a result, this value depends on design choices: with the same inertia I_3, spring stiffness k_3 drives the natural frequency of the system. To change this value, it is easier to change the stiffness of the spring, since the tools on the end effector have a certain mass that cannot be reduced to increase natural frequency value. However, increasing k_3

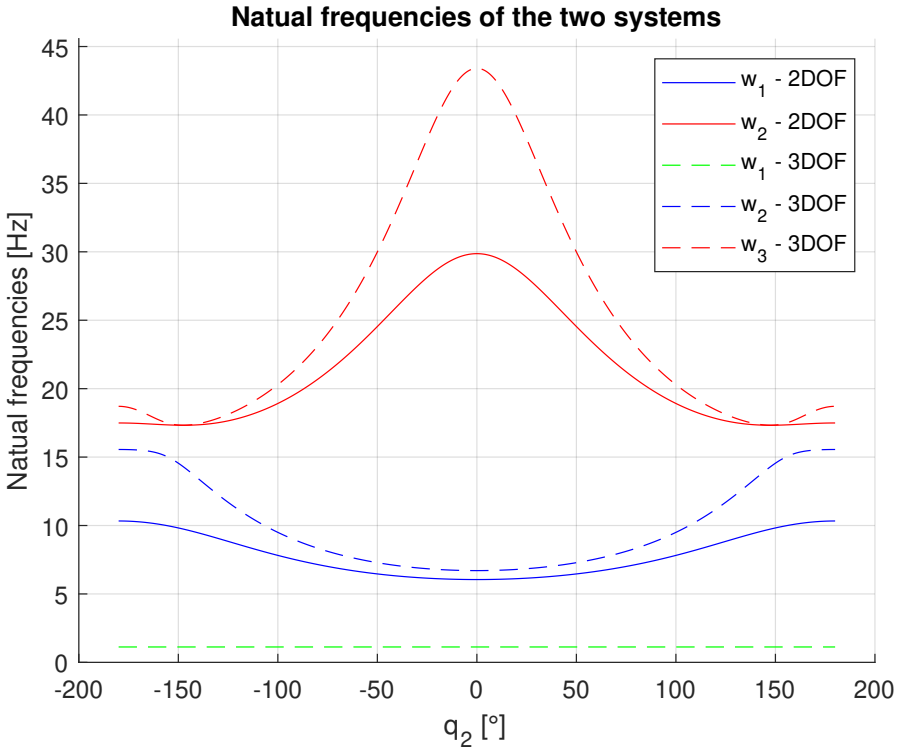


Figure 75: Dependency of the natural frequencies with respect to q_2.

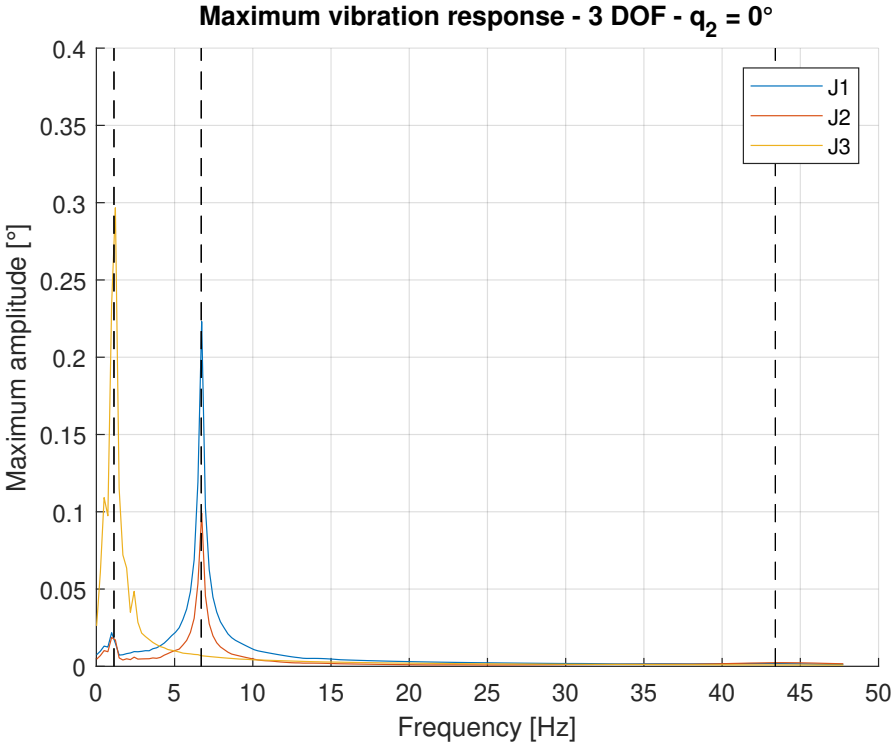
to increase the frequency has a drawback: the flexibility of the under-actuated system would be reduced since the last link would need higher accelerations to provide the same angle value.

To create Figures 76 and 77, an exciting force F with varying frequency and unit amplitude is applied to the end effector of the robot ($F = \sin(\omega t)$). This force is transferred within joint space using the Jacobian matrix and used within the state space Equation 53. The amplitude of the response after the initial transient is stored and then shown in the figures.

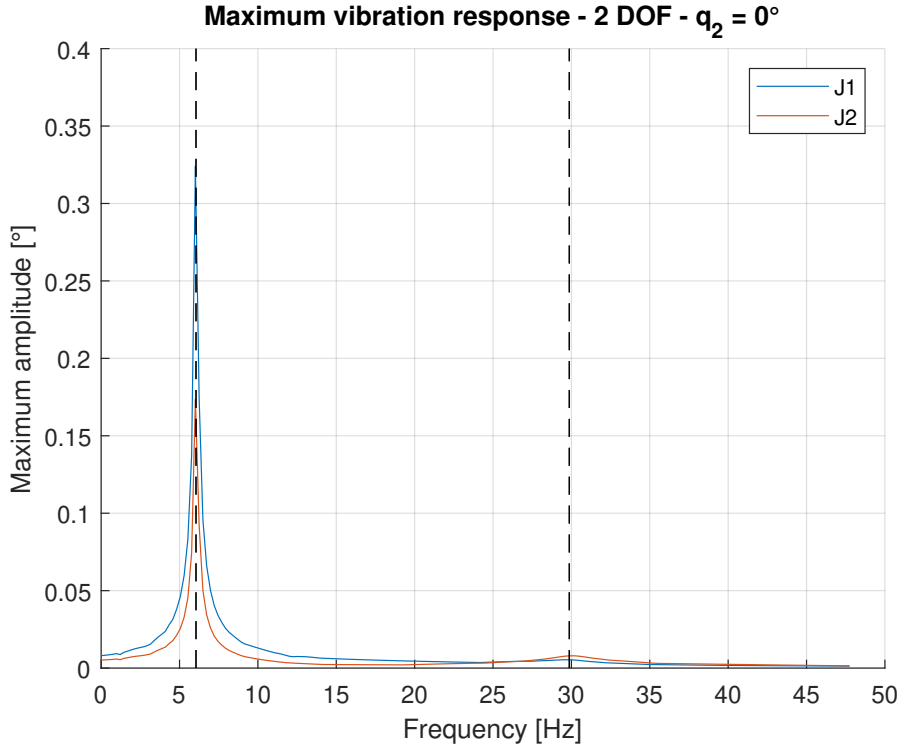
While the additional frequency is mainly related to the motion of the third link, the coupling with the active joints provide a small additional motion to these links (Figures 76a and 77a).

An under-actuated system provides two important benefits: the values of the other two resonance frequencies increase, providing a wider range of frequencies in which the robot can operate before encountering a resonance on the first two joints (Figure 75); moreover, the response at these frequencies is lowered by a considerable factor: in Figures 76 and 77 the peaks are reduced up to 60%.

As can be seen in Figures 76 and 77, the contribution of the damping matrix $\mathbf{C}$ in the identification of the natural frequencies is negligible: the peaks of response calculated iterating the state-space equations around the equilibrium configuration are nearly coincident with the natural frequencies (dashed lines) calculated with Equation 54.

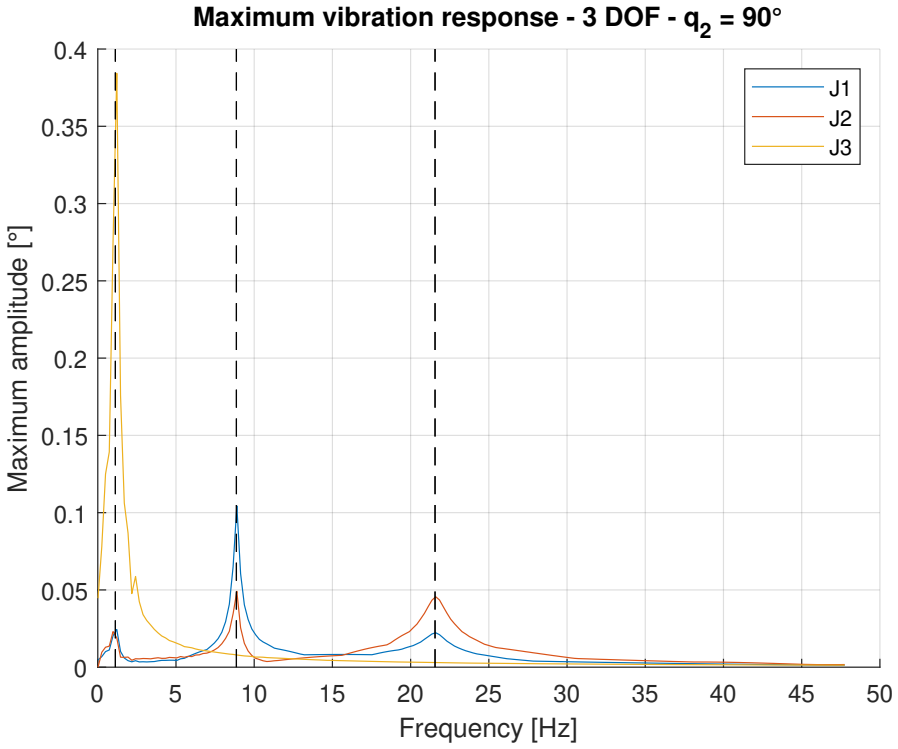


(a) Under-actuated robot

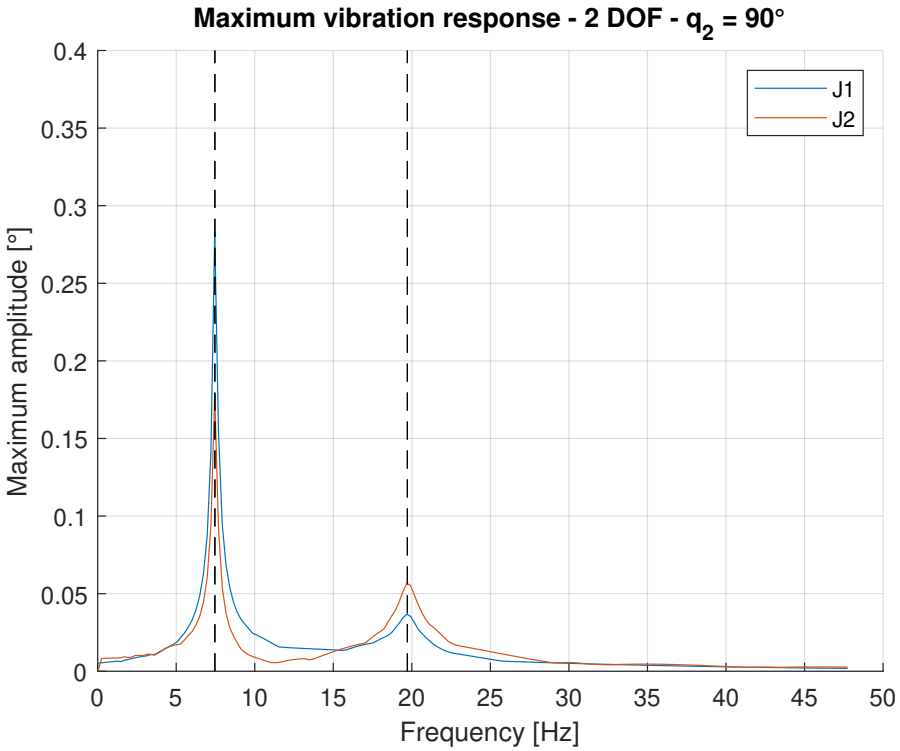


(b) Fully-actuated robot

Figure 76: Response amplitude at the end-effector of the considered robot while exited by continuous forces at different frequencies. Dashed black lines represent the natural frequencies calculated with Equation 54 ($q_1 = q_2 = q_3 = 0$). For clarity, J3 values are reduced by a factor of $3 \cdot 10^2$.



(a) Under-actuated robot



(b) Fully-actuated robot

Figure 77: Response amplitude at the end-effector of the considered robot while exited by continuous forces at different frequencies. Dashed black lines represent the natural frequencies calculated with Equation 54 ($q_1 = q_3 = 0$, $q_2 = 90°$). For clarity, J3 values are reduced by a factor of $3 \cdot 10^2$.

It is also interesting to notice how the configuration influence the amplitude of the response: in Figure 77 ($q_2 = 90^\circ$) the last peak is higher than the last one of Figure 76. In the same way, however, the middle peak of Figure 77 is lower than the corresponding one of Figure 76.

None of the systems is definitely better than the other: while the two degrees of freedom robot provides a more reliable trajectory, the three degrees of freedom manipulator is more flexible and can adapt to the environment without adding relevant masses to the structure; however, the torque required by the under-actuated system can be much higher for low movement times. While the two degrees of freedom robot has only two natural frequencies, the under-actuated system increases the resonance frequencies of the active joints while, at the same time, reducing the amplitude of the response at resonance. The additional resonance frequency can be suitably chosen by design, as it is mostly related to spring stiffness and link inertia. This choice requires to define a trade-off between flexibility and resonance frequency value.

Since none of the two systems is the best in an absolute sense, a hybrid under-actuated system could be considered, in which the third link can be fixed on-demand, so that it could be possible to switch between the under-actuated and the fully-actuated systems depending on the operative conditions. In this way, it would be possible to take advantage of all the features of both the systems.

This hybrid system is not part of this dissertations, and could be developed in future works.

6

CONCLUSIONS

The main contribution of this thesis consists in the introduction in a new set of tools and algorithms that can optimize the design of complex work cells in which redundant tasks are performed. Not only this works applies to the initial background, the robotic deburring, in which the high machining forces and the redundancy of the task make it difficult to perform the operations with a serial industrial robot, but also it applies to collaborative robotics, in which the robots are usually kinematically redundant and the performances of the robot are limited due to safety reasons. By using these tools, the design of the work cell can become a less time-consuming activity: while right now most of the robotic deburring paths are registered by hand in a real work cell, this off-line method can provide a complete path through which the robot will pass to complete all the tasks in parallel with other activities, like the assemble of the work cell. The only step to be performed in a real work cell is the calibration of the real workpiece to adapt the path obtained from the off-line design. This achievement will save hours of the first set-up of the work cell and will reduce the wasted time in the set-up of new products.

While in the first Chapter a first introduction is outlined, in the second Chapter a general idea of the robotic deburring has been provided. It has been shown that multiple grinding tools exist and can be used in multiple (and different) applications. This variety of purposes leads to a clustering of the tasks that has to be taken into account in the optimization process of the work cell. Moreover, different grinding wheels can collide with the environment in different ways.

In the third Chapter, the optimization of the movement between the working location is considered. Two possible solutions have been proposed: a graph method, solved with the Dijkstra algorithm, that uses a net of safe via points to move the robot from one location to another; an adaptive method, that iteratively build a safe movement between two positions using a collision detection algorithm developed for this thesis. Both these methods are equally powerful and computationally efficient, but their fields of application are quite different: while the graph method is more suitable in a static environment with lots of working positions, the adaptive method is more suitable in flexible applications, in which the stating positions can vary during the process. From this optimization, a list of movement times required to move between all the possible combinations of positions is obtained.

This list is to be used in the optimization step proposed in the fourth Chapter. This Chapter presents an extension of the Traveling

Salesman Problem that aims at improving its flexibility and capability. In this way, it is possible to cluster the working positions, constrain some connections and also choose the best constrains to apply to obtain a good solution in very little time.

Finally, in the fifth Chapter, the dynamic of the industrial robot is considered. This is a crucial part because if the robot is not able to perform the deburring process with a good finishing the work cell will require more time to complete the task, thus wasting a lot of time. After the modal testing of the industrial robot located in the Robotics and Automation Labs of the Department of Industrial Engineering of the University of Padova, the results show that the joints that are the most responsible for the vibration of the structure are the first three. This means that, from a vibration point of view, the robot can be simplified by a 3 DOF system in space or by a 2 DOF manipulator that lies on a plane moved by the first joint. Starting from this result, a study on the under-actuated robots has been performed during my period abroad at the Robotics And Rehabilitation (RoAR) Lab of Columbia University (New York) under the supervision of the Prof. Sunil K. Agrawal. The study has focused on the comparison of a 3 DOF under-actuated planar manipulator with a 2 DOF dynamically equivalent fully actuated robot. The results show that, even if the under-actuated robots are not very reliable in terms of trajectory, their vibratory behaviour is very promising, thus leading to a possible future development of hybrid under-fully-actuated systems, in which the passive joints can be locked on demand to provide the best behaviour in terms of trajectory planning and dynamics.

This research can be seen as a starting point in the development of future complex applications, such as, as stated before, hybrid under-fully-actuated systems. In the field of the optimization of the movement and task order, the TSP could be further analysed to introduce some constraints that simulate a specific task order. Finally, the movement between the locations could be further optimized by merging the graph and adaptive methods: an initial movement could be defined using the graph algorithm, and then reducing it by making it pass closer to the object by using the collision avoidance algorithm.

APPENDIX A

A.1 COLLISION DETECTION ALGORITHM

Since the geometry of the objects can be very complex, it is easier to encapsulate them inside bounding volumes [102]. These volumes can be simple spheres, Oriented Bounding Boxes (OBBs) [29], Sphere Swept Volumes (SSVs) [27] and many more. Usually, the tighter the volume is to the object, the slower is the collision detection algorithm [27]. This is due to the complexity (and the number) of the elements that define the volumes: usually to define tighter volumes are required more components, e.g. a tighter discretely-oriented polytope (k-DOP) requires several bounding planes [151].

To evaluate the possible collision between the robot and the environment, I decided to encapsulate the links inside SSVs. To reduce false positive collisions, several SSVs can be used for each single link. All the other objects are encapsulated inside OBBs. So, the first problem is to calculate the collision between an OBB and an SSV.

Each OBB is defined by the intersection of six half-spaces. The boundary of each half-space is described by a plane with the equation:

$$a_i x + b_i y + c_i z + \lambda_i = 0 \tag{63}$$

where a_i, b_i and c_i are the components of the unit vector normal to the $i-th$ plane, pointing towards the restricted volume, while λ_i is the signed distance of the origin of the reference frame from the plane. By putting such parameters in a vector $v_{p,i} = [a_i, b_i, c_i, \lambda_i]$, the signed distance from the $i-th$ plane of a generic point $P = [x_P, y_P, z_P, 1]^T$ is given by:

$$l_{P,i} = v_{p,i} \cdot P \tag{64}$$

where $l_{P,i} \geqslant 0$ for all the six half-spaces when the point P falls within the OBB.

The geometry of an SSV (Figure 78) is defined by the two points P_1 and P_2 (the centres of the the end-cap spheres) and the radius R. Let $l_{P_1,i}$ and $l_{P_2,i}$ be the distances of P_1 and P_2 from the $i-th$ plane. If both $l_{P_1,i}$ and $l_{P_2,i}$ are lower than $-R$, then there is no collision between the SSV and the plane, since both spheres fall outside the restricted volume. Otherwise, the plane intersects the SSV.

Provided that an OBB is bounded by six planes, the collision between an SSV and an OBB can be computed by calculating the distances of the two points P_1 and P_2 from the six planes, and checking if at least one of them is greater than $-R$.

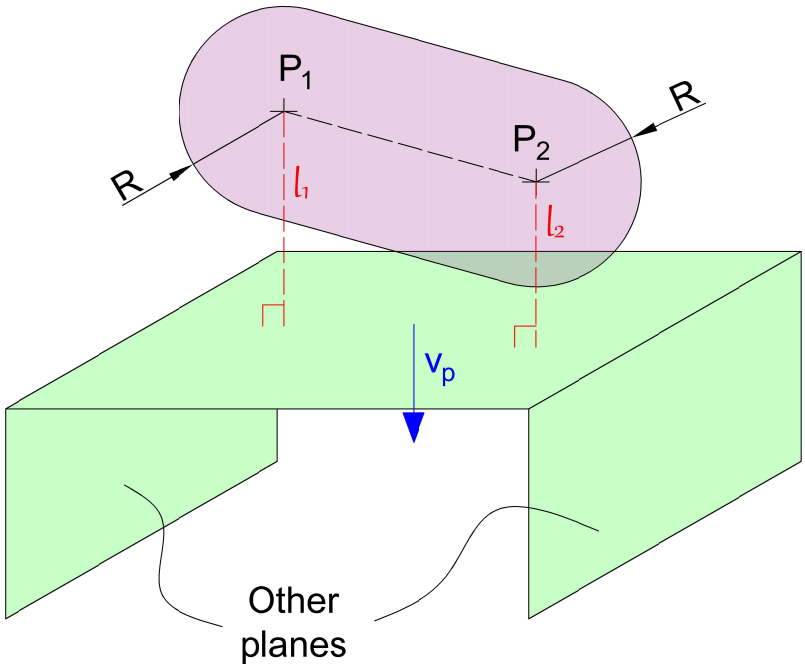


Figure 78: A simple scheme of the parameters used in the estimation of the collision between SSV and half-space

Let n be the number of OBBs in the workspace, and m be the total number of SSVs used to encapsulate robot links. A single matrix calculation can be performed to evaluate all possible interactions:

$$\mathbf{L} = \mathbf{V} \cdot \mathbf{P} \tag{65}$$

where $\mathbf{V}$ is the $6n \times 4$ matrix whose rows contain the parameters of the $6n$ planes bounding the OBBs; $\mathbf{P}$ is the $4 \times 2m$ matrix whose columns contain the coordinates of the points defining the SSVs; $\mathbf{L}$ is the $6n \times 2m$ matrix yielding the distances of all points from all planes. If any element of $\mathbf{L}$ fails to fulfil the condition on the minimum distance between the sphere center and the plane, the collision between OBB and SSV is detected.

Using multiple SSVs may not be enough to avoid false-positive collision detections: as can be seen in Figure 79, the SSVs used in the simplification of the structure result in exceeding volumes. Whilst some exceeding volumes are needed for safety reasons, some of them overly extend the safe volume (green in Figure 79) or simply cannot envelope tighter the robot (blue).

As a result, when the collision between one SSV and one OBB is detected, a more precise (and more demanding) calculation is performed: the whole points cloud defining the link encapsulated by the SSV is compared to the six planes bounding the OBB, to check if any of the points falls within the OBB ($l_{P,i} \geqslant 0$).

Overall, it is computationally more efficient to use SSVs rather than the whole point cloud to perform the collision test (Table 18). With

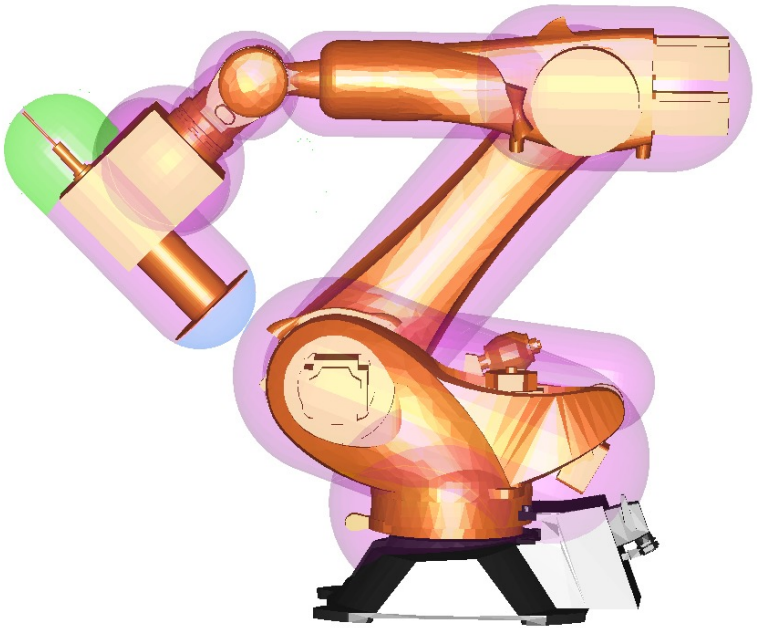


Figure 79: SSVs (purple) surrounding a KUKA KR180 R2500 industrial robot. Some parts of the SSVs can lead to false-positive collision detection due to a bad encapsulation (green) or impossible tighter envelopment (blue).

a path made of 2200 control points, the SSV variant took 0,311s to perform the test, while with the full point cloud the computational time increases to 1,837s (in this example, the point cloud consisted of 50.798 points, while only 12 SSVs were used to enclose the robot links). The PC used for the simulation is powered by a Intel Core i7-2700K, 16 GB of RAM and 64 bit Windows 10 Pro 1803 version.

	Computational time
Full point cloud evaluation	1.837s
Proposed algorithm	0.311s
Improvement	83.07%

Table 18: The proposed algorithm of collision detection is more efficient than a direct collision evaluation of the point cloud of the links (2200 control points).

$$\mathbf{M}_{1\dots3,1}(\mathbf{q}) = \begin{bmatrix} I_1 + I_2 + I_3 + a_1^2(m_2 + m_3) + a_2^2 m_3 + l_1^2 m_1 + l_2^2 m_2 + 2j_1 \\ I_2 + I_3 + a_2^2 m_3 + l_2^2 m_2 + j_1 \\ I_3 \end{bmatrix}$$

$$\mathbf{M}_{1\dots3,2\dots3}(\mathbf{q}) = \begin{bmatrix} I_2 + I_3 + a_2^2 m_3 + l_2^2 m_2 + j_1 & I_3 \\ I_2 + I_3 + a_2^2 m_3 + l_2^2 m_2 & I_3 \\ I_3 & I_3 \end{bmatrix} \tag{66}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -j_2 \dot{q}_2 & -j_2(\dot{q}_1 + \dot{q}_2) & 0 \\ j_2 \dot{q}_1 & 0 & 0 \\ 0 & 0 & k_3 \end{bmatrix} \tag{67}$$

For the trajectory calculation (since the stiffness of the joint is around the designed joint position, but no position error is considered):

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_3 \end{bmatrix} \tag{68}$$

For the vibration analysis:

$$\mathbf{K} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix} \tag{69}$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} g\left[(a_1 m_2 + a_1 m_3 + l_1 m_1)\cos(q_1) + (a_2 m_3 + l_2 m_2)\cos(q_1 + q_2)\right] \\ g(a_2 m_3 + l_2 m_2)\cos(q_1 + q_2) \end{bmatrix} \tag{70}$$

with $j_1 = a_1 \cos(q_2)(a_2 m_3 + l_2 m_2)$ and $j_2 = a_1 \sin(q_2)(a_2 m_3 + l_2 m_2)$.

BIBLIOGRAPHY

[1] International Federation of Robotics, "World robotics 2019 preview," 2019.

[2] A. Allahverdi and H. Soroush, "The significance of reducing setup times/setup costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 978–984, 2008.

[3] J. Heilala, J. Montonen, and O. Väätäinen, "Life cycle and unit-cost analysis for modular reconfigurable flexible light assembly systems," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 222, no. 10, pp. 1289–1299, 2008.

[4] J. P. Shewchuk, "Worker allocation in lean u-shaped production lines," *International Journal of Production Research*, vol. 46, no. 13, pp. 3485–3502, 2008.

[5] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pp. 1–8, VDE, 2010.

[6] G. Boothroyd, *Assembly automation and product design*. CRC Press, 2005.

[7] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, no. 3, pp. 201–212, 1990.

[8] T. Yoshikawa, "Manipulability and redundancy control of robotic mechanisms," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1004–1009, IEEE, 1985.

[9] R. Dubey and J. Luh, "Redundant robot control for higher flexibility," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 1066–1072, IEEE, 1987.

[10] J. M. Hollerbach, "Optimum kinematic design for a seven degree of freedom manipulator," in *Robotics research: The second international symposium*, pp. 215–222, Cambridge, MIT Press, 1985.

[11] A. A. Ata, "Optimal trajectory planning of manipulators: a review," *Journal of Engineering Science and Technology*, vol. 2, no. 1, pp. 32–54, 2007.

[12] G. S. Chirikjian and J. W. Burdick, "An obstacle avoidance algorithm for hyper-redundant manipulators," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 625–631, IEEE, 1990.

[13] G. S. Chirikjian and J. W. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 343–354, 1994.

[14] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *The international journal of robotics research*, vol. 4, no. 3, pp. 109–117, 1985.

[15] Y. Zhang and J. Wang, "Obstacle avoidance for kinematically redundant manipulators using a dual neural network," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 752–759, 2004.

[16] J. Wang, Q. Hu, and D. Jiang, "A lagrangian network for kinematic control of redundant robot manipulators," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1123–1132, 1999.

[17] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 1, pp. 147–154, 2001.

[18] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[19] N. Kumar, V. Panwar, N. Sukavanam, S. Sharma, and J.-H. Borm, "Neural network-based nonlinear tracking control of kinematically redundant robot manipulators," *Mathematical and Computer Modelling*, vol. 53, no. 9-10, pp. 1889–1901, 2011.

[20] R. Dubey and J. Y. Luh, "Redundant robot control using task based performance measures," *Journal of robotic systems*, vol. 5, no. 5, pp. 409–432, 1988.

[21] D. R. Baker and C. W. Wampler, "On the inverse kinematics of redundant manipulators," *The International journal of robotics research*, vol. 7, no. 2, pp. 3–21, 1988.

[22] Y. Wang and P. Artemiadis, "Closed-form inverse kinematic solution for anthropomorphic motion in redundant robot arms," *Adv Robot Autom*, vol. 2, no. 110, p. 2, 2013.

[23] R. V. Dubey, J. A. Euler, and S. M. Babcock, "An efficient gradient projection optimization scheme for a seven-degree-of-freedom redundant robot with spherical wrist," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 28–36, IEEE, 1988.

[24] C. Urrea and J. Kern, "Modeling, simulation and control of a redundant scara-type manipulator robot," *International Journal of Advanced Robotic Systems*, vol. 9, no. 2, p. 58, 2012.

[25] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[26] M. Bottin, G. Boschetti, and G. Rosati, "A novel collision avoidance method for serial robots," in *IFToMM Symposium on Mechanism Design for Robotics*, pp. 293–301, Springer, 2018.

[27] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," tech. rep., Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[28] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 4, pp. 3719–3726, IEEE, 2000.

[29] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 171–180, ACM, 1996.

[30] C. Park, "Self-collision detection & avoidance algorithm for a robot manipulator," *International Journal of Engineering and Innovative Technology*, vol. 5, no. 4, pp. 139–142, 2015.

[31] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[32] M. I. Ribeiro, "Obstacle avoidance," *Instituto de Sistemas e Robótica, Instituto Superio Técnico*, p. 1, 2005.

[33] S. Redon, A. Kheddar, and S. Coquillart, "An algebraic solution to the problem of collision detection for rigid polyhedral objects," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4, pp. 3733–3738, IEEE, 2000.

[34] C. Rodriguez-Garavito, A. A. Patiño-Forero, and G. Camacho-Munoz, "Collision detector for industrial robot manipulators," in *The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications*, pp. 187–196, Springer, 2018.

[35] M. Ratiu and M. A. Prichici, "Industrial robot trajectory optimization-a review," in *MATEC Web of Conferences*, vol. 126, p. 02005, EDP Sciences, 2017.

[36] S. Hrabar, "3d path planning and stereo-based obstacle avoidance for rotorcraft uavs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 807–814, IEEE, 2008.

[37] C. Carbone, U. Ciniglio, F. Corraro, and S. Luongo, "A novel 3d geometric algorithm for aircraft autonomous collision avoidance," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 1580–1585, IEEE, 2006.

[38] J. Barraquand and J.-C. Latombe, "A monte-carlo algorithm for path planning with many degrees of freedom," in *Proceedings., IEEE International Conference on Robotics and Automation*, pp. 1712–1717, IEEE, 1990.

[39] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[40] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[41] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 477–482, IEEE, 2007.

[42] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1874–1879, IEEE, 2006.

[43] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[44] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3d path planning algorithms," *Journal of Control Science and Engineering*, vol. 2016, p. 5, 2016.

[45] S. Seereeram and J. T. Wen, "A global approach to path planning for redundant manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 152–160, 1995.

[46] O. P. Agrawal and Y. Xu, "On the global optimum path planning for redundant space manipulators," *IEEE transactions on systems, man, and cybernetics*, vol. 24, no. 9, pp. 1306–1316, 1994.

[47] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[48] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[49] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[50] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 403–410, 1988.

[51] L. Tian and C. Collins, "Motion planning for redundant manipulators using a floating point genetic algorithm," *Journal of Intelligent & Robotic Systems*, vol. 38, no. 3, pp. 297–312, 2003.

[52] N. C. N. Doan and W. Lin, "Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6r articulated robots," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 233–242, 2017.

[53] J. Pamanes-García, E. Cuan-Durón, and S. Zeghloul, "Single and multi-objective optimization of path placement for redundant robotic manipulators," *INGENIERÍA Investigación y Tecnología*, vol. 9, no. 3, pp. 231–257, 2008.

[54] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: a survey," *Operations Research*, vol. 16, no. 3, pp. 538–558, 1968.

[55] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations research*, vol. 11, no. 6, pp. 972–989, 1963.

[56] M. Bellmore and J. C. Malone, "Pathology of traveling-salesman subtour-elimination algorithms," *Operations Research*, vol. 19, no. 2, pp. 278–307, 1971.

[57] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *biosystems*, vol. 43, no. 2, pp. 73–81, 1997.

[58] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 61–63, 1962.

[59] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," in *Encyclopedia of operations research and management science*, pp. 1573–1578, Springer, 2013.

[60] J.-F. Petiot, P. Chedmail, and J.-Y. Hascoët, "Contribution to the scheduling of trajectories in robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 3, pp. 237–251, 1998.

[61] Y. Edan, T. Flash, U. M. Peiper, I. Shmulevich, and Y. Sarig, "Near-minimum-time task planning for fruit-picking robots," *IEEE transactions on robotics and automation*, vol. 7, no. 1, pp. 48–56, 1991.

[62] R. Kumar and Z. Luo, "Optimizing the operation sequence of a chip placement machine using tsp model," *IEEE Transactions on Electronics Packaging Manufacturing*, vol. 26, no. 1, pp. 14–21, 2003.

[63] D. CHAN and D. Mercier, "Ic insertion: an application of the travelling salesman problem," *The International Journal of Production Research*, vol. 27, no. 10, pp. 1837–1841, 1989.

[64] J. Balakrishnan and P. D. Jog, "Manufacturing cell formation using similarity coefficients and a parallel genetic tsp algorithm: Formulation and comparison," *Mathematical and Computer Modelling*, vol. 21, no. 12, pp. 61–73, 1995.

[65] J. Mareczek, M. Buss, and G. Schmidt, "Robust global stabilization of the underactuated 2-dof manipulator r2d1," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 3, pp. 2640–2645, IEEE, 1998.

[66] J. Mareczek, M. Buss, and G. Schmidt, "Robust control of a nonholonomic underactuated scara robot," in *Progress in system and robot analysis and control design*, pp. 381–396, Springer, 1999.

[67] S. Abdolshah, D. Zanotto, G. Rosati, and S. K. Agrawal, "Optimizing stiffness and dexterity of planar adaptive cable-driven parallel robots," *Journal of Mechanisms and Robotics*, vol. 9, no. 3, p. 031004, 2017.

[68] G. Rosati, M. Faccio, A. Carli, and A. Rossi, "Fully flexible assembly systems (f-fas): a new concept in flexible automation," *Assembly Automation*, vol. 33, no. 1, pp. 8–21, 2013.

[69] M. Bergerman, Y. Xu, and Y.-H. Liu, "Control of cooperative underactuated manipulators: a robustness comparison study,"

in *1st Workshop on Robotics and Mechatronics, Hong Kong. Citeseer*, pp. 279–286, Citeseer, 1998.

[70] A. A. Siqueira and M. H. Terra, "Nonlinear and markovian h/-sub/spl infin//controls of underactuated manipulators," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 6, pp. 811–826, 2004.

[71] C. Aguilar-lbÅAñez and H. Sira-Ramirez, "Pd control for active vibration damping in an underactuated nonlinear system," *Asian Journal of Control*, vol. 4, no. 4, pp. 502–508, 2002.

[72] S. S. Nudehi and R. Mukherjee, "Enhancing controllability and observability in underactuated and undersensed systems through switching: Application to vibration control," *Journal of dynamic systems, measurement, and control*, vol. 126, no. 4, pp. 790–799, 2004.

[73] International Federation of Robotics, "World robotics 2018 executive summary industrial robots," 2018.

[74] International Federation of Robotics, "World robotics 2018 executive summary service robots," 2018.

[75] J. Edward, C. W. Wannasuphoprasit, and M. A. Peshkin, "Cobots: Robots for collaboration with human operators," in *International Mechanical Engineering Congress and Exposition, Atlanta*, Citeseer, 1996.

[76] L. Ventures, "Industrial robotics outlook: 2015-2025," 2019.

[77] ISO, "Robots for industrial environments safety requirements-part 1," ISO ISO 10218-1, International Organization for Standardization, Geneva, Switzerland, 2011.

[78] ISO, "Robots for industrial environments safety requirements-part 2," ISO ISO 10218-2, International Organization for Standardization, Geneva, Switzerland, 2011.

[79] kuka, "http://www.kuka-robotics.com/," 2019.

[80] ABB, "http://www.abb.com/," 2019.

[81] R. robotics, "https://www.rethinkrobotics.com/," 2019.

[82] U. robots, "https://www.universal-robots.com/," 2019.

[83] Fanuc, "http://www.fanuc.eu/uk/en," 2019.

[84] Comau, "https://www.comau.com/," 2019.

[85] G. Rosati, G. Boschetti, A. Biondi, and A. Rossi, "On-line dimensional measurement of small components on the eyeglasses assembly line," *Optics and Lasers in Engineering*, vol. 47, no. 3, pp. 320–328, 2009.

[86] G. Rosati, M. Faccio, L. Barbazza, and A. Rossi, "Hybrid fexible assembly systems (h-fas): bridging the gap between traditional and fully flexible assembly systems," *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 5-8, pp. 1289–1301, 2015.

[87] C. Finetto, G. Rosati, M. Faccio, and A. Rossi, "Implementation framework for a fully flexible assembly system (f-fas)," *Assembly Automation*, vol. 35, no. 1, pp. 114–121, 2015.

[88] M. Bottin, G. Rosati, and G. Boschetti, "Fixed point calibration of an industrial robot," in *18th International Conference of the European Society for Precision Engineering and Nanotechnology, EUSPEN 2018*, pp. 215–216, 2018.

[89] M. Faccio, M. Bottin, and G. Rosati, "Collaborative and traditional robotic assembly: a comparison model," *The International Journal of Advanced Manufacturing Technology*, vol. 102, no. 5-8, pp. 1355–1372, 2019.

[90] A. Doria, S. Cocuzza, N. Comand, M. Bottin, and A. Rossi, "Analysis of the compliance properties of an industrial robot with the mozzi axis approach," *Robotics*, vol. 8, no. 3, p. 80, 2019.

[91] L. Gillespie, "Deburring precision miniature parts," *Precision Engineering*, vol. 1, no. 4, pp. 189–198, 1979.

[92] M. Her and H. Kazerooni, "Automated robotic deburring of parts using compliance control," *Journal of dynamic systems, measurement, and control*, vol. 113, no. 1, pp. 60–66, 1991.

[93] L. K. Gillespie, *Deburring and edge finishing handbook*. Society of Manufacturing Engineers, 1999.

[94] S. Malkin and C. Guo, *Grinding technology: theory and application of machining with abrasives*. Industrial Press Inc., 2008.

[95] I. D. Marinescu, M. P. Hitchiner, E. Uhlmann, W. B. Rowe, and I. Inasaki, *Handbook of machining with grinding wheels*. CRC Press, 2016.

[96] ISO, "Industrial automation systems and integration - product data representation and exchange - part 1: Overview and fundamental principles," ISO ISO 10303-1, International Organization for Standardization, Geneva, Switzerland, 1994.

[97] I. G. E. Specification, "Digital representation for communication of product definition data," *ANSI Y14. 26M-1981, published by: The American Society of Mechanical Engineers*, vol. 345.

[98] L. Roscoe *et al.*, "Stereolithography interface specification," *America-3D Systems Inc*, vol. 27, 1988.

[99] W. Khalil and J. Kleinfinger, "A new geometric notation for open and closed-loop robots," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1174–1179, IEEE, 1986.

[100] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.

[101] E. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 829–858, 2016.

[102] C. Ericson, *Real-time collision detection*. CRC Press, 2004.

[103] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[104] E. Plaku and L. E. Kavraki, "Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning," in *Algorithmic Foundation of Robotics VII*, pp. 3–18, Springer, 2008.

[105] G. Boschetti, "A picking strategy for circular conveyor tracking," *Journal of Intelligent & Robotic Systems*, vol. 81, no. 2, pp. 241–255, 2016.

[106] Z. Shiller, "Off-line and on-line trajectory planning," in *Motion and Operation Planning of Robotic Systems*, pp. 29–62, Springer, 2015.

[107] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of applied mechanics*, vol. 77, no. 2, pp. 215–221, 1955.

[108] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[109] G. Rosati, M. Faccio, C. Finetto, and A. Carli, "Modelling and optimization of fully flexible assembly systems (f-fas)," *Assembly Automation*, vol. 33, no. 2, pp. 165–174, 2013.

[110] MATLAB, "Traveling salesman problem: Problem-based."

[111] MATLAB, "Traveling salesman problem: Solver-based."

[112] MATLAB, "Mixed-integer linear programming (milp)."

[113] Y. Chen and F. Dong, "Robot machining: recent development and future research issues," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 9-12, pp. 1489–1497, 2013.

[114] Z. Pan, H. Zhang, Z. Zhu, and J. Wang, "Chatter analysis of robotic machining process," *Journal of materials processing technology*, vol. 173, no. 3, pp. 301–309, 2006.

[115] A. Gasparetto, "Eigenvalue analysis of mode-coupling chatter for machine-tool stabilization," *Journal of Vibration and Control*, vol. 7, no. 2, pp. 181–197, 2001.

[116] A. Gasparetto, "A system theory approach to mode coupling chatter in machining," *Journal of dynamic systems, measurement, and control*, vol. 120, no. 4, pp. 545–547, 1998.

[117] H. Zhang, J. Wang, G. Zhang, Z. Gan, Z. Pan, H. Cui, and Z. Zhu, "Machining with flexible manipulator: toward improving robotic machining performance," in *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, pp. 1127–1132, IEEE, 2005.

[118] C. Dumas, S. Caro, S. Garnier, and B. Furet, "Joint stiffness identification of six-revolute industrial serial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 4, pp. 881–888, 2011.

[119] E. Abele, M. Weigold, and S. Rothenbücher, "Modeling and identification of an industrial robot for machining applications," *CIRP annals*, vol. 56, no. 1, pp. 387–390, 2007.

[120] G. Alici and B. Shirinzadeh, "Enhanced stiffness modeling, identification and characterization for robot manipulators," *IEEE transactions on robotics*, vol. 21, no. 4, pp. 554–564, 2005.

[121] F. Rafieian, Z. Liu, and B. Hazel, "Dynamic model and modal testing for vibration analysis of robotic grinding process with a 6dof flexible-joint manipulator," in *2009 International Conference on Mechatronics and Automation*, pp. 2793–2798, IEEE, 2009.

[122] G. Carbone, "Stiffness analysis and experimental validation of robotic systems," *Frontiers of Mechanical Engineering*, vol. 6, no. 2, pp. 182–196, 2011.

[123] H. Ni, C. Zhang, T. Hu, T. Wang, Q. Chen, and C. Chen, "A dynamic parameter identification method of industrial robots considering joint elasticity," *International Journal of Advanced Robotic Systems*, vol. 16, no. 1, p. 1729881418825217, 2019.

[124] D. J. Ewins, *Modal testing: theory and practice*, vol. 15. Research studies press Letchworth, 1984.

[125] N. M. M. Maia and J. M. M. e Silva, *Theoretical and experimental modal analysis*. Research Studies Press, 1997.

[126] D. J. Inman and R. C. Singh, *Engineering vibration*, vol. 3. Prentice Hall Englewood Cliffs, NJ, 1994.

[127] G. M. del Garbo, "Discorso matematico sopra il rotamento dei corpi," 1763.

[128] R. Marcolongo, *Notizie sul Discorso Matematico e sulla vita di Giulio Mozzi*. 1905.

[129] A. A. Shabana, *Dynamics of multibody systems*. Cambridge university press, 2013.

[130] R. S. Ball, *A Treatise on the Theory of Screws*. Cambridge university press, 1998.

[131] H. Lipkin and J. Duffy, "Sir robert stawell ball and methodologies of modern screw theory," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 216, no. 1, pp. 1–11, 2002.

[132] X. Wu, Y. Lu, X. Duan, D. Zhang, and W. Deng, "Design and dof analysis of a novel compliant parallel mechanism for large load," *Sensors*, vol. 19, no. 4, p. 828, 2019.

[133] V. Cossalter and A. Doria, "Analysis of motorcycle slalom manoeuvres using the mozzi axis concept," *Vehicle System Dynamics*, vol. 42, no. 3, pp. 175–194, 2004.

[134] V. Cossalter and A. Doria, "Instantaneous screw axis of two-wheeled vehicles in typical manoeuvres," *Vehicle System Dynamics*, vol. 44, no. sup1, pp. 669–678, 2006.

[135] A. Doria and L. Taraborrelli, "The twist axis of frames with particular application to motorcycles," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 230, no. 17, pp. 3026–3039, 2016.

[136] P. Blanchet and H. Lipkin, "Dual properties for vibration analysis via screw theory," in *Proc. ASME Design Engineering Technical Conference, Atlanta, Georgia, DETC98/MECH-5868*, Citeseer, 1998.

[137] T. Patterson and H. Lipkin, "Structure of robot compliance," *Journal of Mechanical Design*, vol. 115, no. 3, pp. 576–580, 1993.

[138] C. Chen, F. Peng, R. Yan, Y. Li, D. Wei, Z. Fan, X. Tang, and Z. Zhu, "Stiffness performance index based posture and feed orientation optimization in robotic milling process," *Robotics and Computer-Integrated Manufacturing*, vol. 55, pp. 29–40, 2019.

[139] L.-W. Tsai, *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons, 1999.

[140] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[141] H. Sira-Ramirez and S. K. Agrawal, *Differentially flat systems*. Crc Press, 2004.

[142] J. Franch, S. K. Agrawal, and V. Sangwan, "Differential flatness of a class of n-dof planar manipulators driven by 1 or 2 actuators," *IEEE transactions on automatic control*, vol. 55, no. 2, pp. 548–554, 2010.

[143] J. Franch, A. Reyes, and S. K. Agrawal, "Differential flatness of a class of n-dof planar manipulators driven by an arbitrary number of actuators," in *2013 European Control Conference (ECC)*, pp. 161–166, IEEE, 2013.

[144] S. K. Agrawal and V. Sangwan, "Design of under-actuated open-chain planar robots for repetitive cyclic motions," in *ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1057–1066, American Society of Mechanical Engineers, 2006.

[145] S. K. Agrawal and V. Sangwan, "Differentially flat designs of underactuated open-chain planar robots," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1445–1451, 2008.

[146] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[147] K. A. Foss, "Coordinates which uncouple the equations of motion of damped linear dynamic systems," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE AEROELASTIC AND STRUCTURES RESEARCH LAB, 1956.

[148] R. A. Frazer, W. J. Duncan, A. R. Collar, *et al.*, *Elementary matrices and some applications to dynamics and differential equations*, vol. 1963. Cambridge University Press Cambridge, 1938.

[149] L. Barbazza, D. Zanotto, G. Rosati, and S. K. Agrawal, "Design and optimal control of an underactuated cable-driven micro–macro robot," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 896–903, 2017.

[150] D. Zanotto, G. Rosati, and S. K. Agrawal, "Modeling and control of a 3-dof pendulum-like manipulator," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3964–3969, IEEE, 2011.

[151] J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-dops," *IEEE Transactions on Visualization & Computer Graphics*, no. 1, pp. 21–36, 1998.

Ma sai che ti vedrei bene come professore?

— I. Callegari
(sperando che questo augurio si avveri)

RINGRAZIAMENTI

Tre anni possono cambiarti la vita. E se mi guardo indietro, non posso che confermare questa tesi. Questo dottorato mi ha insegnato tanto, mi ha fatto maturare tanto e mi ha sicuramente reso una persona migliore. E non sarebbe stato possibile senza le fantastiche persone che mi sono state accanto.

Prima di tutto, la mia cara mor-moglie Federica, che mi ha dovuto sopportare in prima persona dal primissimo giorno di dottorato fino all'ultimo, nella gioia e nel dolore, nella salute e nella malattia (ecc, ecc.). Senza di lei questi tre anni sarebbero stati molto piú difficili!

Ringrazio poi la mia famiglia, mamma, papá, sorellina (che studiando meno ha avuto un contratto a tempo indeterminato prima di me, beata lei!), nonni e suoceri, perché se non era per loro tanti traguardi non sarei mai stato in grado di raggiungerli, sia prima che durante il dottorato.

Grazie al Prof. Giulio Rosati, che mi ha convinto ad intraprendere questa strada, mi ha fatto appassionare al mondo universitario e sta facendo (ancora) tanto per me. Grazie anche ai Prof. Rossi e Cocuzza per il supporto!

Un grande abbraccio a tutti i vecchi e nuovi amici: le Sbolfe, i compari da Maser, "i Ceschi", i mitici di Trento, i colleghi d'ufficio (ormai sono piú amici che colleghi), i nerdoni di Treviso (prima o poi riprenderemo D&D!), le due bellissime coppie del Corso Fidanzati e i fantastici compagni di viaggio di New York. Non dimentichiamoci poi dei colleghi di Smartworld, che da un anno e mezzo hanno deciso di includermi nella loro particolare famigliola, e di rimando anche il gruppo accanito di *fan* dal quale sono nate alcune belle amicizie a distanza.

A huge "Thank You" to Prof. Sunil K. Agrawal and the entire staff of the ROaR Lab of the Columbia University, who have given me the opportunity to work in such a different and friendly place at the opposite side of the world.

E infine grazie a chi non ho citato, che non se la prenda con me ma con la mia pessima memoria!