Optimal motion planning in redundant robotic systems for automated composite lay-up process

Jiuchun Gao

UNIVERSITE BRETAGNE SCIENCES LOIRE POUR L'INGENIEUR

CENTRALE NANTES

THESE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 602
Sciences pour l'Ingénieur
Spécialité : Génie mécanique

Par

Jiuchun GAO

Optimal Motion Planning in Redundant Robotic Systems for Automated Composite Lay-up Process

Thèse présentée et soutenue à Amphithéâtre S, Ecole Centrale de Nantes, le 29 juin 2018
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Gabriel Abba	Professeur des universités, Université de Lorraine
Hélène Chanal	Maître de conférences HDR, Institut français de mécanique avancée, Aubière

Composition du Jury :

Président	
Benoît Furet	Professeur des universités, Université de Nantes
Gabriel Abba	Professeur des universités, Université de Lorraine
Hélène Chanal	Maître de conférences HDR, Institut français de mécanique avancée, Aubière
Sylvain Miossec	Maître de conférences, IUT Bourges
Directeur de thèse	
Anatol Pashkevich	Professeur, IMT-Atlantique
Co-directeur	
Stéphane Caro	Chargé de recherche-HDR, CNRS
Invité	
Claire Dumas	Ingénieur de recherche, Daher Aerospace

ACKNOWLEDGEMENTS

Before beginning this manuscript, I would like to express my gratefulness to people who have guided and helped me in my research and who have provided all possible assistance throughout the time of working on the thesis.

First and foremost I am sincerely thankful to my supervisor, Professor Anatol Pashkevich, for his continuous support of my Ph.D study and research. It is difficult to assess his contribution in my work. He always directed my work in a right way, gave invaluable advices, and spent plenty of time to improve my thesis quality and the research significance. He taught me to write high-quality scientific papers and carry out independent research.

Special thanks also goes to my another supervisor, Dr. Stéphane Caro, for his insightful comments, patience and encouragement. His guidance helped me in all the time of my study and research. It is also an honor for me to thank the members of the "Comité de suivi" Dr. Sylvain MIOSSEC and Dr. Mathieu RITOU for their help and precious suggestions.

I am grateful for all conversations and discussions with my colleague Dr. Alexandr Klimchik who always gave reasonable answers and high quality advices when I met difficulties during my first year. In addition, I thank Mr. Divya Shah for his significant contribution to the simulations.

I also would like to thank our industrial partner from CETIM Alain Lemasçon, Benoît Courtemanche and Julien Lefort. They provided me technological advices and gave the opportunity to evaluate experimentally my theoretical work on a real robotic platform.

The last and most important thank goes to my family, who always supported me in all my pursuits. I would like to thank them for all their love and encouragement.

This work was funded by the China Scholarship Council [grant numbers 201404490018].

ABSTRACT AND KEYWORDS

Abstract: The thesis deals with the optimal motion planning in redundant robotic systems for automation of the composite lay-up processes. The primary goal is to improve the lay-up workcell productivity by developing a novel methodology of optimizing coordinated motions of the robotic manipulator, workpiece positioner and workspace extension unit, which ensure the shortest processing time and smooth movements of all mechanical components. In contrast to the previous works, the proposed methodology provides high computational efficiency and also takes into account both the technological constraints and the robotic system constraints, which describe capacities of the actuators and are expressed by the maximum allowable velocities and accelerations in the actuated joints. The developed technique is based on conversion of the original continuous problem into a combinatorial one, where all possible configurations of the mechanical components are represented as a directed multi-layer graph and the desired time-optimal motion is generated using dynamic programming principle for searching the shortest path on the graph satisfying the smoothness constraints. It is also proposed an enhancement of this technique by dividing the optimization procedure in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with large discretization step. Then, the same technique is applied in the local search space, which is created with smaller step in the neighborhood of the obtained trajectory. Alternatively, the second stage may implement a straightforward smoothing of the redundant variable profiles. The advantages of the developed methodology are confirmed by industrial implementation on the factory floor that deals with manufacturing of the high-pressure vessel.

Keywords: Redundant robotic system, Motion planning, Time-optimal trajectory, Dynamic programming, Automated composite lay-up

CONTENTS

GENERAL INTRODUCTION

Motivation. At present, composite materials are increasingly used in engineering practice. Compared to traditional ones, they have good strength-to-weight ratio, durability, flexibility of shaping and corrosion resistance (Christensen, 2012, Jones, 1998, Kaw, 2005, Lubin, 2013, Peters, 2013). For these reasons, they are extremely attractive for fabrication of large dimensional parts in aerospace, automotive and marine industries (Gay, 2014, Marsh, 2011).

A conventional way of manufacturing such composite components is based on the manual lay-up process. However, this labor-intensive procedure is quite slow, expensive and does not allow achieving required repeatability (Marsh, 2011). An alternative solution implementing such process is usually based on automated lay-up of the composite tape or fiber, which are placed layer-by-layer to ensure full coverage of the relevant surface. Compared to manual techniques, such automated lay-up processes ensure higher productivity, repeatability and better product quality.

The automated lay-up processes can be implemented by using either specifically designed machines or robotic systems, which in this application are usually redundant. In the first case, general CNC machines equipped with specific lay-up head are used. They have no essential limitations on the workpiece size, but usually are quite expensive and require large work floor areas (Gallet-Hamlyn, 2011, Rabeneck, 2010). Compared to the process-dedicated machines, the robotic systems composed of a 6-axis serial robot (mounted on a linear track/gantry), an actuated positioner and a dedicated technological tool are relatively cheap and flexible allowing changing the product type easily (Krombholz et al., 2012, Shirinzadeh et al., 2004). For these reasons, the robotic lay-up systems are increasingly used in practice. However, the manufacturing process planning and manipulator motion programming for such robotic systems is not trivial. The main difficulty here arises because the kinematic redundancy of the robotic system, which usually contains two extra degrees of freedom provided by the rotational positioner and the translational unit (linear track or gantry). This redundancy does not allow generating the manipulator trajectories straightforwardly and also complicates robot programming, which is still a challenge in productivity improvement for the automated lay-up process. On the other hand, the redundancy gives user some flexibility in motion planning and optimal utilization of the actuator capabilities allowing generating faster trajectories.

In robotic literature, the issue of the kinematic redundancy resolution has been studied for several decades. A conventional way is based on the pseudo inverse of the kinematic Jacobian (Flacco and De Luca, 2015, Kazerounian and Nedungadi, 1988, Wu et al., 2000, Buss, 2004). It provides a unique solution for the differential kinematic equations in the sense of least squares corresponding to the smallest Euclidean norm of the displacement vector in the joint space. Some other approaches (Tabarah et al., 1994, Gan et al., 2013, Gan et al., 2012) for non-complex shape tasks are based on coordinating of robot/positioner motions, where the motions of the "master" are assigned firstly and the conjugate trajectories of the "slave" are then determined. For complex shape workpiece taking into account constraints imposed by actuators, several alternative optimization based methods have been developed for the spot-welding application (Gueta et al., 2011a, Gueta et al., 2011b). They are based on converting the original problem into a discrete form, where the robotic manipulator and the positioner joint spaces are discretized and desired trajectory can be represented as a path on the corresponding graph. Slightly different approach was proposed in (Dolgui and Pashkevich, 2009, Pashkevich et al., 2004) for the laser cutting applications. It minimizes the oscillation in actuator

velocities when assuming the tool speed is constant. Hence, the known techniques cannot be straightforwardly applied to the considered lay-up technology where the robot tool speed should be as high as possible. Besides, required functionality is not included in existing commercial software packages (such as Robcad, Robotmaster, CATFiber, etc) allowing generating certain coordinated motions for the redundant robotic systems. For these reasons, the problem of optimal motion planning for robotic lay-up systems becomes very important and corresponds to current needs of the composite product fabrication technology.

Thesis goal and research problems. The thesis focuses on *the optimal motion planning in redundant robotic systems allowing improving the productivity of the automated composite lay-up workcell*. Special attention is paid to the motion coordination of the robotic manipulator, workpiece positioner and workspace extension unit, which ensures the shortest manufacturing time and smooth movements of all mechanical components. To achieve this goal, the following tasks have to be solved:

Task #1:

> Analysis of existing systems for composite product manufacturing, detecting their weak points and comparative study of known methods for the motion planning in redundant robotic systems.

Task #2:

> Creating kinematic model of a typical redundant robotic system used in the composite lay-up technology and formalization of the related optimal motion planning problem.

Task #3:

> Development of the method for the optimal coordinated motion planning in the redundant robotic system composed of the robotic manipulator, workpiece positioner and workspace extension unit, which allows minimizing the total motion time and ensuring smooth movements of all mechanical components.

Task #4:

> Application of the developed algorithms to real industrial problems, development of the robot control programs implementing generated time-optimal trajectories, simulation of the coordinated movements in 3D environment, and experimental validation of the developed techniques on the factory floor.

Thesis organization. To solve the above defined tasks, the thesis is organized as follows:

Chapter 1 presents the state of the art in the field of robot based composite lay-up technology and describes existing techniques of motion planning in robotic systems with kinematic redundancies. The main purpose of this chapter is to detect the weak points of existing methods and justify necessity of developing of new ones, which allow defining the thesis main goal and related research problems.

Chapter 2 focuses on the kinematic modeling of a typical redundant robotic system used in the composite lay-up technology and formalization of the related optimal motion planning problem. The considered problem is presented as finding of the time-optimal trajectory in robotic system joint space

under specific constraints related to both the robotic system and technological task. These constraints allow taking into account limitations of the robotic system (joint limits, maximum joint velocities/accelerations), practical requirements for the manipulator postures (via singularities and collision constraints) as well as the Cartesian path constraints issued from the composite lay-up process. The main purpose of this chapter is to formalize the considered technological problem and to present it in the form common for the mathematical optimization theory.

Chapter 3 is devoted to the development of a new motion planning method for the redundant robotic systems utilized in the automated composite lay-up process. It is based on the transforming the original problem into a combinatorial one and applying the dynamic programming principle. First, the task graph is created by discretizing the redundant variables and sequentially applying to all task locations the direct kinematics of the positioner (and workspace extension unit) as well as inverse kinematics of the robot. Then, the developed algorithm based on dynamic programming generates the time-optimal motion taking into account all relevant constraints. To reduce the computing time, the time-optimal motion planning is divided in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with large discretization step. Then, the same technique is applied in the local search space, which is created with smaller step in the neighborhood of the obtained trajectory. Alternatively, the second stage may implement a straightforward smoothing of the redundant variable profiles. Efficiency of them and advantages compared to the conventional techniques are confirmed by several case studies.

Chapter 4 deals with industrial implementation of the developed motion planning method on the factory floor (for manufacturing of a high-pressure vessel). Using the proposed method, there were generated time-optimal smooth motions for the manipulator and positioner of the SPIDE-TP robotic platform allowing speeding up the thermoplastic tape lay-up process. Correctness of these motions was verified in 3D simulation environment of CATIA software package. Besides, it was created a program in KRL language implementing these motions and ensuring coordinated control of KUKA KR210 robot and AFPT workpiece positioner. There are presented results of industrial experiments carried out with this program, which confirmed smoothness of the manipulator/positioner movements and essential reduction of the time required for the thermoplastic tape lay-up process.

Finally, Conclusion summarizes the main contributions of the thesis and defines perspectives for future research work.

Main contributions of the thesis. Theoretical contributions of this thesis are in the area of redundancy resolution and time-optimal motion planning for robotic systems composed of the robotic manipulator, workpiece positioner and workspace extension unit. The most essential results can be summarized as follows:

(i) *Formalization of the optimal motion planning problem for typical robotic lay-up platforms*. The proposed approach presents the considered problem as finding of the time-optimal trajectory in the joint space of the robotic system under specific constraints related to the technological task. The latter include both conventional kinematic constraints (joint limits, maximum joint velocities/accelerations), allowable distances to the singularities and obstacles as well as the Cartesian path constraints issued from the composite lay-up process.

(ii) *A new method of coordinated time-optimal motion planning for redundant robotic systems ensuring smooth movements of all mechanical components*. It is based on transformation of the original optimization problem into a combinatorial one and application of the dynamic programming principle. At the first stage, the redundant variables are discretized and the task graph is created by sequentially applying to all task locations the direct kinematics of the positioner (and workspace extension unit) as well as inverse kinematics of the robot. Then, the developed optimization algorithm based on dynamic programming generates the time-optimal motion taking into account all constraints related to the system kinematics and considered technology. Efficiency of the developed method and its advantages compared to the conventional techniques are confirmed by several case studies.

(iii) *Enhancement strategies for the developed motion planning method.* To reduce the computing time required for the motion planning, it was proposed to avoid combinatorial search in high-dimensional space. To achieve this target, the time-optimal motion planning is divided in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with relatively large discretization step. Then, the same technique is applied in the local search space, which is created with small discretization step in the neighborhood of the trajectory obtained at the previous stage. As an alternative, the second stage may implement a straightforward smoothing of the redundant variable profiles based on the spline approximation. As follows from relevant study, the proposed enhancement strategies allow essentially reducing the computing time, down to the level acceptable in engineering practice.

(iv) *Application of the developed optimal motion planning method to real industrial problems.* Using the thesis results, there were generated time-optimal smooth motions for the manipulator and positioner of the SPIDE-TP robotic platform allowing speeding up the thermoplastic tape lay-up process for manufacturing of the composite high-pressure vessel. For these motions, it was created a program in KRL language ensuring coordinated control of KUKA KR210 robot and AFPT workpiece positioner. Experimental evaluation of the motions described by this program confirmed smoothness of the manipulator/positioner movements and essential reduction of the time required for the thermoplastic tape lay-up process.

Dissemination of research results. The main results obtained in this thesis have been published in four works and have been presented in three international conferences. Among them, there is a paper in an international journal (Mechanism and Machine Theory) and three papers in international conference proceedings (EUCOMES'2016 - European Conference on Mechanism Science of IFTOMM, ICMIT'2017 - International Conference on Manufacturing and Industrial Technologies, and ICOME'2017 - International Conference on Mechanical Engineering).

The scientific contribution of this thesis have been developed and validated experimentally in the framework of strong cooperation with R&D Department of Polymer and Composite of the technical center CETIM (Centre Technique des Industries Mécaniques), Nantes.

FIGURES

TABLES

CHAPTER 1 ROBOT-BASED FIBER REINFORCEMENT TECHNOLOGY IN INDUSTRY

This chapter presents the state of the art in the field of robot based composite lay-up technology and describes existing techniques of motion planning in robotic systems with kinematic redundancies. The main purpose of this chapter is to detect the weak points of existing methods and justify necessity of developing of new ones, which allow defining the thesis main goal and related research problems.

1.1 COMPOSITES PARTS FABRICATION TECHNIQUES

Composite materials are being increasingly used for primary structures in industrial, aerospace and marine structures because of their outstanding properties and features. Among the existing composites parts fabrication techniques, automated composite lay-up is attractive since it suits the workpiece with arbitrary surfaces. For this process, a robotic system (an industrial robot collaborating with an actuated positioner) replaces the general CNC machines with the advantages of no limitations on the object size and geometry. This section presents the utilizations of the composite materials in industry and the automation of fabrication.

1.1.1 Composite Materials in Manufacturing

Composite materials are combinations of at least two constituent materials with significantly different physical or chemical properties. One of the material is called the reinforcing phase that is usually in the form of fibers (continuous or short), sheets, or particles, and is embedded in the other material called the matrix phase (Lubin, 2013, Peters, 2013). Usually, continuous fiber composites are stronger and stiffer than particulate composites (Campbell, 2010). Figure 1 briefly shows how typical continuous fiber composites are made.

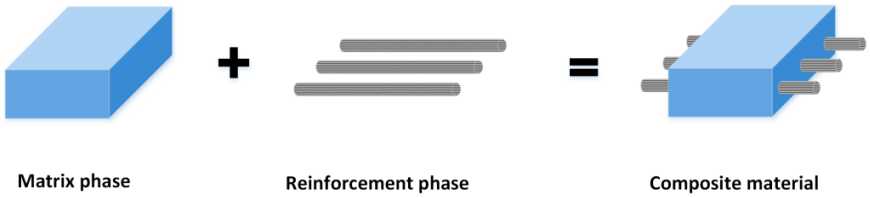


Figure 1 Components of typical fiber reinforced composite material

The two constituents are a reinforcing and a matrix. The reinforcing phase provides the strength and stiffness. In most cases, the reinforcement is harder, stronger, and stiffer than the matrix (Campbell, 2010). The common constitutes of fiber reinforced composites are shown in Table 1.

Table 1 Constitutes of fiber reinforced composite materials

Reinforcing phase	Matrix phase	Interface
Glass Carbon Ceramic Metallic …	Polymer Metals Ceramics …	Bonding Surface

Such new combined materials usually have characteristics different from the individual components. The individual components remain separate and distinct within the finished structure, and the new combined materials have improved properties over the individual materials (Gay, 2014, Christensen, 2012, Dirk et al., 2012, Jones, 1998, Kaw, 2005, Lubin, 2013, Marsh, 2011, Peters, 2013, Rabeneck, 2010, Shama Rao et al., 2014). Generally, the matrix phase is the main constituent of composite

materials (may be up to 70% by volume) mainly responsible for its overall mechanical properties. Accordingly, with the different types of matrix phase, the composite materials may have very different properties, see Table 2.

Table 2 Composite classifications on the basis of matrix phase

	Polymer matrix composite	Ceramic matrix composite	Metal matrix composite
Reinforcing	Glass, carbon, graphite and aramid	Carbon and aluminum oxide	Graphite, alumina, silicon carbide…
Main Properties	High strength-to-weight ratio High stiffness-to-weight ratio	Resistance to high temperature and corrosive environment	High modulus of elasticity, ductility, and resistance to elevate temperature
Applications	Automotive, naval, aeronautical and aerospace…	Jet and automobile engines…	Satellite, missile, space structures…

Polymer matrix composites (PMCs) have established themselves as engineering structural materials, which are prominent class of composites compared to other composite materials in commercial applications. The property of high strength (stiffness)-to-weight ratio makes PMCs more attractive than conventional materials in manufacturing, see Figure 2.

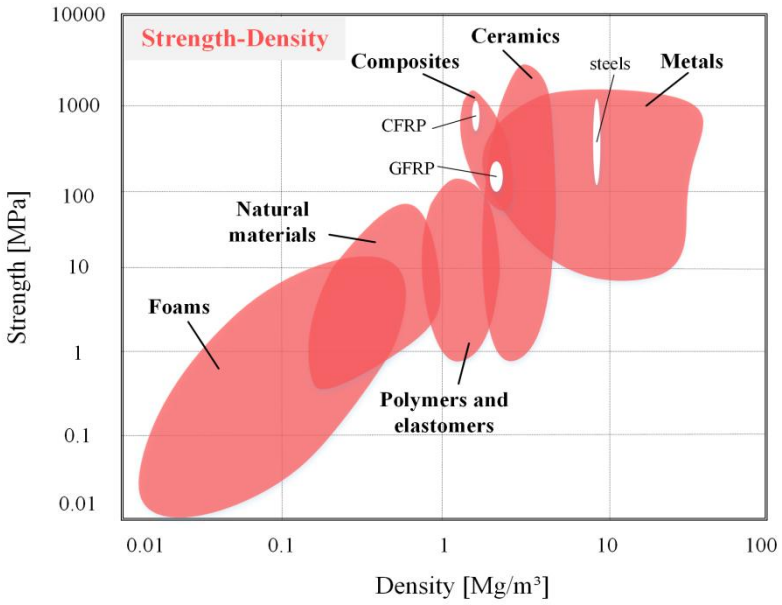


Figure 2 Strength plotted against density (yield strength for metals and polymers; compressive strength for ceramics, tear strength for elastomers and tensile strength for composites) (Ashby and Cebon, 1993)

Glass fiber reinforced polymers represent the largest class of PMCs (Chawla, 2012). An important application is in the use of manufacturing pressure vessels (~200kPa), see Figure 3. Heavy steel cylinders usually result in a reduced payload, which are desired to be replaced by much lighter PMC cylinders. Other common applications involve pipes for transportation of water or petroleum. In the offshore industry, they are widely used for risers, stress joints and fluid handling since PMCs offer

many advantages over metal because of their high specific strength and stiffness, good durability, low thermal conductivity and good corrosion resistance (Pham et al., 2016, Salama et al., 2002, Tamarelle and Sparks, 1987, Garoushi, 2018).

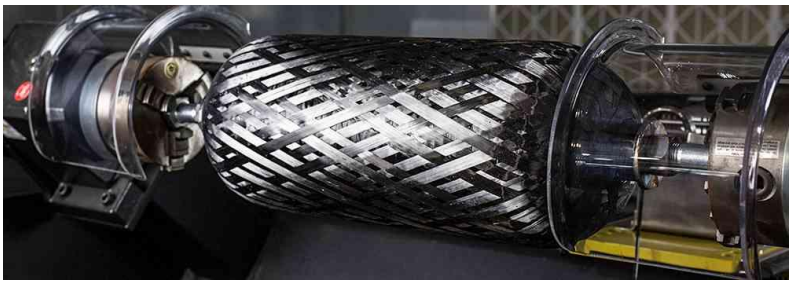


Figure 3 Composite materials are used to make pressure tanks
©MATERIA (http://www.materia-inc.com/)

Carbon fiber reinforced PMCs are the most important structural composites; especially in the aerospace field (Chawla, 2012, Nicolais et al., 2011, Deborah, 2010, Pilato and Michno, 1994). For example, the Boeing 787 and Airbus 350 make great use of composite materials in its airframe and primary structure (an airframe comprising more than 50% carbon fiber reinforced epoxy and other composites) (Hale, 2006, Marsh, 2007), as is shown in Figure 4. By using the composites, it obviously saves the weight (on average of 20% compared to more conventional aluminum designs), fuel and extends range of flying. In addition, such large airplanes, flying at high altitudes, have pressurized cabins. The limit of pressurization depends on the strength of the fuselage material. A fuselage made of PMCs can withstand higher pressure (corresponding to 1800 m altitude) than one made of aluminum (corresponding to 2400 m) (Chawla, 2012).

Figure 4 Composite materials used throughout the body of the Boeing 787
©BOEING (https://www.comsol.com)

In wind energy industry, the rotor blade of a wind turbine has the shape of an aerofoil like the wing of an airplane. The material for the skins of the aerofoil needs to be strong, stiff, but light. These requirements lead to fiber reinforced polymer composites as the optimum materials. Goubalt and Mayes (Goubalt and Mayes, 1996) compared composite materials to steel and aluminum to be used for the primary structures of a patrol craft. The corresponding studies find that the structural weight of a patrol craft made of glass-reinforced plastic materials is 10% lighter than an aluminum one and 36% lighter than a steel one of similar size (Mäkinen et al., 1998, Mouritz et al., 2001, Goubalt and Mayes,

1996). A prediction is made that the cost of operating a composite craft will be less than for a steel design because of the maintenance reduction (less corrosion) and lower fuel consumption. Additionally, calculated life cycle costs of a composite craft are 7% less than that of a steel one of the same size.

Non-polymer matrix composites here include metal matrix composites and ceramic matrix composites. *Metal matrix composites* (MMCs) usually use aluminum, titanium or magnesium alloys as continuous matrix, with high modulus of elasticity (Pank and Jackson, 1993). In the Hubble telescope, pitch based continuous carbon fiber reinforced aluminum was used for waveguide booms because this composite is very light, has a high elastic modulus and has a low coefficient of thermal expansion. In the US Trident missile, beryllium has been replaced by SiC*p*/Al composite (Chawla, 2012). Metal matrix composites can be tailored to have optimal thermal and physical properties to meet the requirements of electronic packaging systems (e.g., cores, substrates, carriers, and housings). Continuous boron fiber reinforced aluminum composites made by diffusion bonding have been used as heat sinks in chip carrier multilayer boards.

Ceramic matrix composites (CMCs) in general have a very attractive property, i.e., resistance to high temperature (800°C~1650°C) and corrosive environment. CMCs are more commonly used in heat engines, components requiring resistance to aggressive environments, special electronic/electrical applications, energy conversion, and military systems.

In general, because metal and ceramic matrix composites require very high temperatures and sometimes high pressures for processing, they are normally much more expensive than polymer matrix composites. For this reason, compared to PMCs, the applications of CMCs and MMCs are relatively limited (CHUNG, 2003, Campbell, 2010). In this work, the discussed composite material is common high-performance polymer matrix composites that are widely used in industry.

1.1.2 Automation of Composite Products Manufacturing

In industry, there are several techniques originally developed for making fiber reinforced polymer matrix composites. Since the matrix type affects processing, fabrications for PMCs can be classified into two categories: thermo-set composite processing and thermoplastic composite processing (see Table 3). A thermo-set is a resin with low-viscosity that reacts and cures during processing, forming an intractable solid. A thermoplastic is a high-viscosity resin that is processed by heating it above its melting temperature. Because a thermo-set resin sets up and cures during processing, it cannot be reprocessed by reheating. By comparison, a thermoplastic can be reheated above its melting temperature for additional processing (Campbell, 2010, Chawla, 2012, Park and Seo, 2015, Bratukhin and Bogolyubov, 2012). Figure 5 shows the main manufacturing processes for the both.

Lay-up is the most common polymer processing techniques. It can be done either manually (known as hand lay-up) or by automated devices. Fiber reinforcements can be laid onto a mold and the resin (unsaturated polyester is one of the most common) is applied with a brusher or a roller. The schematic is shown in Figure 5a. Hand lay-up is suitable for making a wide variety of composites products from very small to very large, however, it is usually time consuming and labor intensive, which leads to high costs (Elkington et al., 2015). The labor cost associate with creating composites takes a large portion of the total manufacturing cost (Lindbäck et al., 2012, Frketic et al., 2017). A highly trained

technician creates around 1kg composite material per hour, but even the best can make a mistake (Sloan, 2008). Human error can introduce voids and irregularities into the composite part during production, which negatively affects mechanical properties, making parts unusable for standard operation. For those reasons, manufacturers are transitioning to automated lay-up to improve production volume per mold and cost efficiency (Grimshaw et al., 2001, Skinner, 2006).

Table 3 Classification of PMCs processes

	Thermo-set composites processing	Thermoplastic composites processing
Continuous-fiber composites	Lay-up (manual/automated) Filament winding Resin transfer molding (RTM) Pultrusion	Lay-up (manual/automated) Thermoforming Compression molding
Short-fiber composites	Spray-up Injection molding Compression molding Resin transfer molding (RTM)	Injection molding Thermoplastic compression molding

Spray-up is quite similar to hand lay-up in its suitability for making boats, tanks, transportation components, and tub/shower units in a large variety of shapes and sizes. Frequently, resin and fibers (chopped) are fed and sprayed together onto the mold surface through a chopper gun (Chawla, 2012). The schematic is shown in Figure 5b. This process uses simple, low cost tooling and simple processing. Portable equipment permits on-site fabrication with virtually no part size limitations. Similar to hand lay-up, the production volume per mold is not high, but it also could be improved by automation.

Vacuum bag molding is designed to improve the mechanical properties of multiple layers of fiber reinforcement bonded with a resin after hand lay-ups (resin rich problem). It can be used with wet laminating and pre-impregnated composites molding. In *wet laminating* case, the fiber reinforcement is saturated using hand lay-up, and then a vacuum bag is mounted on the mold to force out trapped air and excess resin, compact the laminate. In the case of *pre-impregnated composites molding*, the pre-impregnated material is laid up on the mold, a vacuum bag is mounted and the mold is heated or the mold is placed in an autoclave that applies both heat and external pressure, adding to the force of atmospheric pressure. The schematic of this process is shown in Figure 5c. The procedure of pre-impregnating, vacuum bagging and autoclave curing is most often used to create advanced composite aircraft and military products (Chawla, 2012, Mallick, 2007).

Compression molding is a process in which the preheated materials are placed over a mold that mounted in a hydraulic or mechanical molding press, then, two halves are closed and pressure is applied. After the hydraulic press releases, the finished piece will be ejected out of the mold. The schematic is shown in Figure 5d. Compression molding enables part design flexibility and features such as inserts, ribs, bosses and attachments. Good surface finishes are obtainable, contributing to lower part finishing cost. Subsequent trimming and machining operations are minimized in compression molding and labor costs are low (Mallick, 2007).

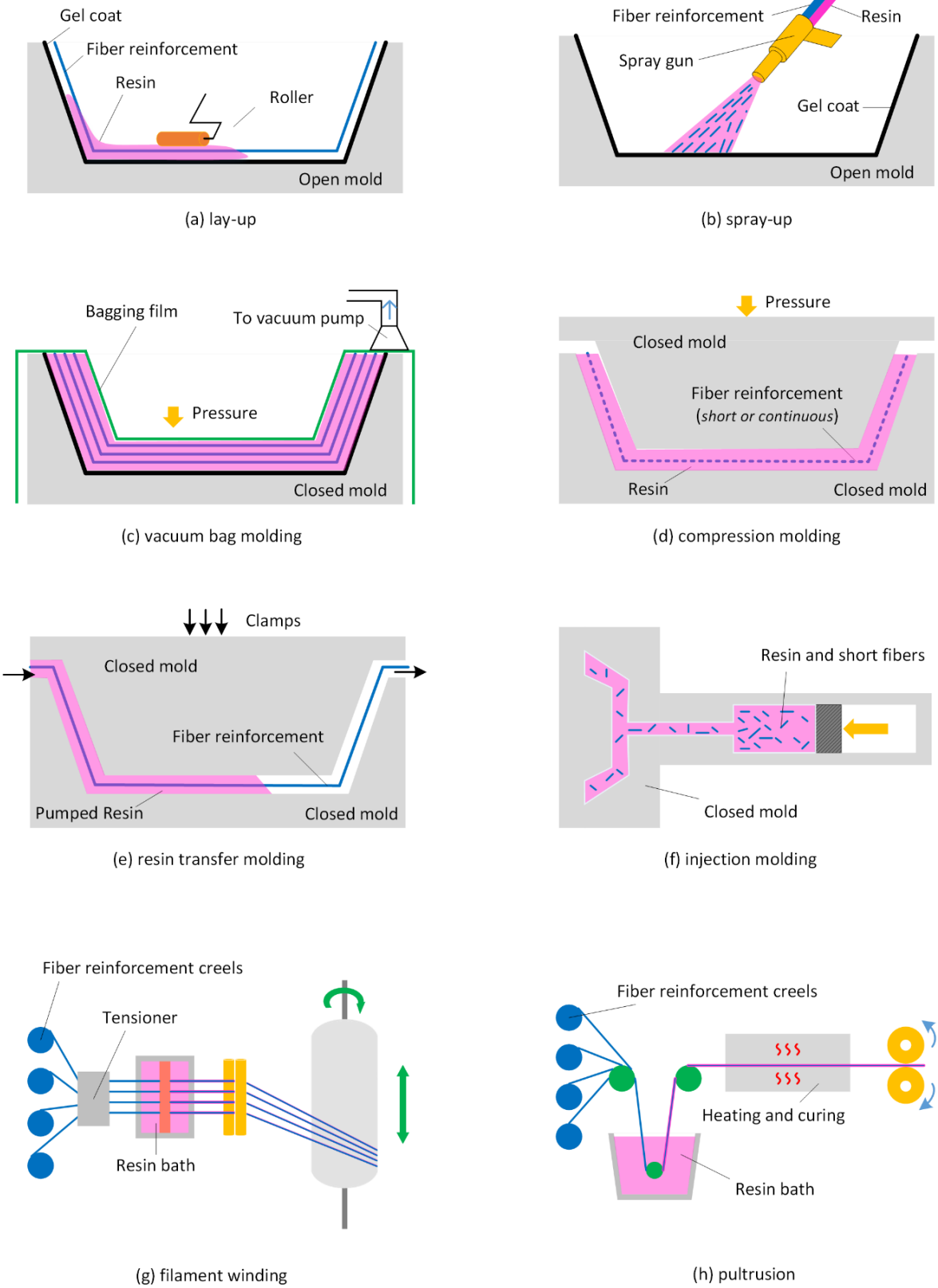


Figure 5 Common PMCs manufacturing processes in industry

Resin transfer molding (RTM), also called liquid molding, is an intermediate volume molding process. In this method, dry reinforcement material is laid inside the mold, the mold is clamped, and resin is pumped in (through injection ports) under pressure. The schematic is shown in Figure 5e. This

process produces complex parts with smooth finishes on all exposed surfaces. The process can be simple or highly automated–and cycle times are speedy. The automotive industry uses RTM to be a cost-effective process for large scale processing, e.g., composite parts of Dodge Viper automobile are made by RTM (Chawla, 2012).

Injection molding is a process in which the resin with short fibers is forced into a heated chamber, and then pushed through a nozzle at the end of the barrel that is pressed against the mold. Once the part inside the mold cools completely, the mold opens, and the part is ejected. The schematic is shown in Figure 5f. An extension of this process is reinforced reaction injection molding (RRIM), in which two (or more) resins are heated separately, combined with short fibers, and then injected into a mold under high pressure and compressed. The types of products that are processed by this technique are wide and varied, ranging from large automotive parts to tiny gears (Chawla, 2012).

Pultrusion forms composites into long, straight shapes like rods or bars. Continuous strands of reinforcement are pulled through a resin bath to saturate them, and then pulled through heated steel molds that sculpt the composites into continuous lengths. The schematic is shown in Figure 5h. It is a continuous process and can be readily automated. Labor costs are low and finished products are very strong. Pultrusion is utilized to make products such as beams, channels, pipes, tubing, fishing rods and golf club shafts (www.compositeslab.com).

Filament winding is an automated process commonly used for manufacturing axial symmetric structures by winding continuous filaments under tension around a rotating mandrel (Skinner, 2006), as shown in Figure 5g and Figure 6. Filament winding may use either dry fiber passed through a resin bath (wet winding) or pre-impregnated materials (dry winding) for production (Abdalla et al., 2007). After the composite part reach the desired thickness, the mandrel then can be cured and removed from the part, leaving the hollow final product. For some products, the mandrel is a permanent part of the finished product as a barrier to protect the composite from the fluid/gas to be stored. Filament winding is well suited to industrial automation. The controlled variables for winding are fiber type, winding angle, tow or bandwidth and thickness of the fiber bundle. The winding angle has an effect on the properties of the final product. A high angle (hoop) will provide circumferential strength, while lower angle patterns (polar or helical) will provide greater longitudinal/axial tensile strength (Laval, 2006, Fleischer and Schaedel, 2013, Hernandez-Moreno et al., 2008, Rousseau et al., 1999). Products currently being produced using this technique range from pipes, oars, bicycle forks, power and transmission poles, pressure vessels to missile casings, aircraft fuselages and lamp posts and yacht masts.

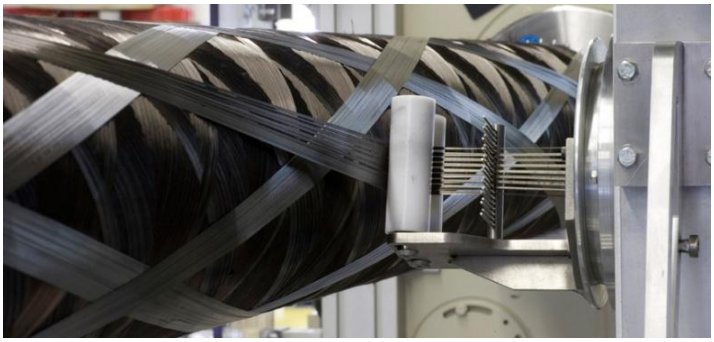


Figure 6 Typical filament winding process
© CONNOVA (www.connova.com/)

The above presented processes are all used for polymer matrix composite product manufacturing, and their properties are briefly shown in Table 4. It is obvious that, compared to the others, lay-up technique can be applied in a wider range of products, from very small to very large, and it well suits complex and large dimensional components fabrication especially in aerospace fields. However, the production rate and the repeatability of hand lay-up are quite low, which cannot meet the requirements of mass production volumes in practice. For these reasons, to gain the productivity, automation of lay-up techniques is becoming necessary (Shama Rao et al., 2014, Mallick, 2007, Olsen and Craig, 1993).

Table 4 Properties of existing PMCs manufacturing process

Processes	Properties
Lay-up	Suitable for a wide variety of products No complicated equipment required (low cost on tooling)
Spray-up	Only with short fibers
Vacuum bag molding	An extra process adds cost both in labor and in bagging materials
Compression molding Resin transfer molding Injection molding	Matched complicated tooling is always required
Pultrusion Filament winding	Limited to component shapes (constant cross-section and cylindrical, respectively)

As follows from the literature analysis, at present, there are two main mechanized lay-up techniques used in industry: automated tape laying (ATL) and automated fiber placement (AFP) (Frketic et al., 2017). These two techniques use a computer-aided design and/or computer-aided manufacturing model to build up a specific structure, in a layer by layer process (layer of laminated tapes or tows, resin-impregnated continuous fibers) (Dirk et al., 2012, Groover, 2007). In an automated lay-up process (ATL or AFP), the product geometry is programmed into control system and commonly the molds themselves are made of composite materials.

Automated tape laying (ATL) is a technique mainly used to produce composite structures for large noncomplex (flat or single curvature) parts (Beakou et al., 2011, Sloan, 2008). ATL machines lay a resin pre-impregnated tape or continuous fabric strips with widths ranging from 75 to 300 mm onto a flat surface in various orientations (Sloan, 2008, Dirk et al., 2012), as shown in Figure 7. The tool of the lay-up machine usually consists of spools of tape, a winder, winder guides, a compaction roller, tape cutter and a positioner sensor. The first step to produce composite is depositing a starting amount of pre-impregnated tape onto the mold using a soft silicone roller. Then, the machine deposits the tape according to tool paths defining the part geometry. At the end of the lay-up, the tape is cut automatically by rotating pinching blades. Low-rail gantry platforms are also available in medium and large size ranges that can be matched to customer part size and floor space requirements. The machines feature tape-deposit speeds of up to 800 mm/s and a high degree of placement accuracies.

Figure 7 Typical automated tape laying system
© MTORRES (www.mtorres.com/)

This process is realized by computer aided design (CAD). Using the CAD system, the product to be manufactured is developed mathematically onto a surface, which is further broken down into layers to be fabricated by laying tapes side by side. Control software is used to place strips in each layer via a series of numerical control steps to develop the final product shape. Such equipments were initially found in defense related applications, e.g., the wing skin panels of F-22 Raptor fighter jet. With Boeing 787 and Airbus 380 and other advanced aircrafts, automated processing techniques have moved into civilian aircraft construction as well, i.e., make aircraft parts such as wing stringers, spars, skins and elevators, tail skins and horizontal planes, engine cowls, fuselage skins and belly fairings (Chawla, 2012). Boeing uses the tape laying process wherein strips of carbon fiber are laid on a spinning mandrel by multiple robotic laying heads. Layer upon layer of pre-impregnated strips are laid on the spinning mandrel until desired shape and thickness is obtained. This is followed by curing for about 2 h in an autoclave at around 250 °C (Chawla, 2012). Figure 8 shows the Boeing 787's unique one-piece composite barrel construction processed with ATL.

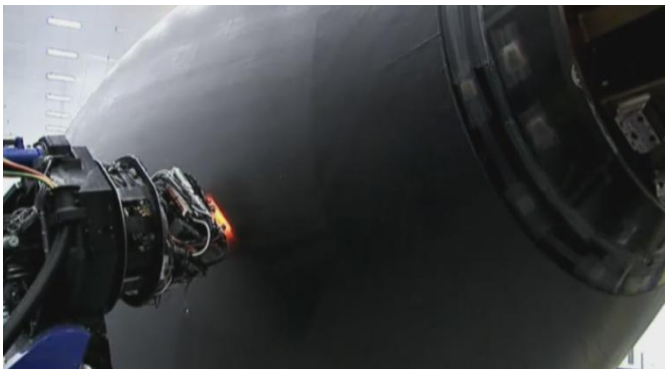


Figure 8 ATL for Boeing787's unique one-piece composite barrel construction
©BOEING (https://www.boeing.com)

Automated fiber placement (AFP) is a technique much similar to ATL and it is employed when producing large composite structures with much complex geometries (Marsh, 2011). That is because AFP places a number of narrow pre-impregnated fiber tows (2 to 32 tows) that can be steered over sharply curved surfaces whereas wider tapes cannot be so placed without wrinkling some of the fibers (Marsh, 2011, Blom et al., 2009). This difference is shown in Figure 9. Each tow here is normally driven individually and can be clamped, cut and restarted during processing with low tow tensions. This enables lay-up over complex shapes and tows steering, and is suitable for example in structures such as fuselage sections with window cut-outs, or wing skins with numerous pad-ups and valleys (Dirk et al., 2012, Izco et al., 2006, DeVlieg et al., 2007, Denkena et al., 2016), see Figure 10. Similar

to ATL, the tows here can be laid in any orientations and positions so that laminate can be tailored to deliver the strength and stiffness required by the designers at various parts of the structure.

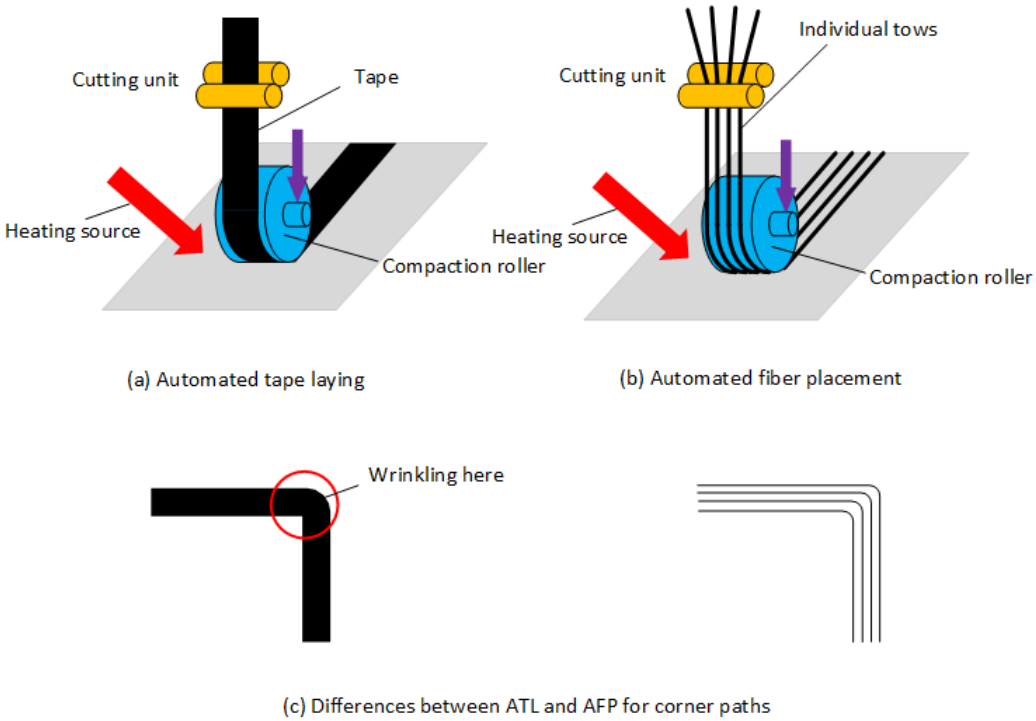


Figure 9 Schematic of automated tape laying and automated fiber placement

Low-rail gantry platforms are also available here in medium and large size ranges that can be matched to customer part size and floor space requirements. Compared to the ATL, the processing speed of AFP is slightly slower (around half of ATL) (Dirk et al., 2012), but AFP can place material more effectively over contoured surfaces, it will be key to high-volume production of composite structures with complex shapes (Marsh, 2011).

Figure 10 Automated fiber placement process for fuselage section producing with window cut-outs
© MTORRES (www.mtorres.com/)

As follows from the literature, the above presented automated lay-up techniques (ATL and AFP) are becoming industry standards for producing fiber reinforced polymer matrix composites. AFP improves on ATL by allowing direct lay-up of more complex components. In addition, material wastage rates are reduced and productivity for complex parts is even higher due to the unique operating per tow and steering capabilities. The future for AFP probably lies with machines that are considerably faster. Increases in manufactured part size and complexity, together with the high rates at which the industry

needs to fabricate composite parts, have demanded the fiber placement at 0.85m/s and more (Marsh, 2011).

1.1.3 Industrial Systems for AFP Based Composite Manufacturing

The automated fiber placement process was firstly realized in practice by using specially designed CNC-controlled machines, and some of the AFP system suppliers are companies rooted in CNC machine tools, e.g. Cincinnati, Ingersoll and MTorres. A specific placement head can be driven with at least 6 axes of motion to lay the fiber reinforcements. These machines are usually delivered under different architectures, i.e. gantry type, column type and cantilever type.

A well-known system is the VIPER™ platform made by Cincinnati (USA), as is shown in Figure 11a. With 7 axes of motion, this device particularly suited to flat and highly contoured structures such as cowls, ducts, fuselage panels and barrels, bulkheads, wings, payload adaptors, fan blades, spars, frames and stringers. The latest VIPER 6000 series AFP machines can produce fuselage panels up to 6.3 m in diameter. These VIPER 6000s machines are being used in Boeing for the barrel sections of its B787 fuselages (Marsh, 2011, Marsh, 2007).

(a) (b)

Figure 11 (a) Cincinnati VIPER™ platform for automated fiber placement, © Cincinnati VIPER™
(b) TORRESFIBERLAYUP machine for automated fiber placement, © MTorres.

Another system for fiber placement is the TORRESFIBERLAYUP machine developed by MTorres (Spain), see Figure 11b. Similar to VIPER™ platform, this dedicated machine is based on a general CNC machine equipped with a placement head. It is being used for building the Airbus A350 XWB wing front spar. Electroimpact (USA) produced a cell built around a control architecture that permitted the use of two CNC-controlled machines equipped modular AFP heads running simultaneously or independently (Flynn et al., 2011, Flynn et al., 2010). It is employed for the manufacture of large primary aircraft structures.

Compared to the CNC-controlled fiber placement machines, the robotic lay-up systems are economically attractive and flexible, allowing changing the product type easily (Rabeneck, 2010, Gallet-Hamlyn, 2011, Dirk et al., 2012). In comparison to their sizes, they can provide a large working area. These robotic lay-up systems are usually composed of a 6-axis serial robotic manipulator equipped with a specific fiber placement end-effector, an actuated workpiece positioner and a workspace extension unit (linear track/gantry). The actuated positioner can have one or two degrees of freedoms. The product to be manufactured is mounted on the positioner flange that adjusts its posture

by rotating the positioner axis. In practice, a one-axis positioner can be sufficient if the product is not too large and its shape is simple. In case of large dimensional product, the linear rail unit is activated to increase the robot workspace.

In the robotic lay-up system developed by Coriolis Composite (France), a placement head is drove by a standard 6-axis poly-articulated robot (located on a rail) for laying and the mold is held by an actuated positioner, see Figure 12. The set composed of 8 axes of motion effectively meet the specifications of complex composite part processing. For feeding the fibers, a creel (or bobbin cabinet) situated at the foot of the robot provides all the necessary functions for unwinding the bobbins at high speed with low tension and enables swift ergonomic loading and unloading of the bobbins. The pipes individually feed each fiber from the creel to the compact light head avoiding all risks of twisting or damage to the fiber, while maintaining a low tension (www.coriolis-composites.com). The Coriolis AFP system is also used to develop structural components in thermo-set and thermoplastic composites as well dry fiber performs in National Aerospace Laboratory of Netherlands.

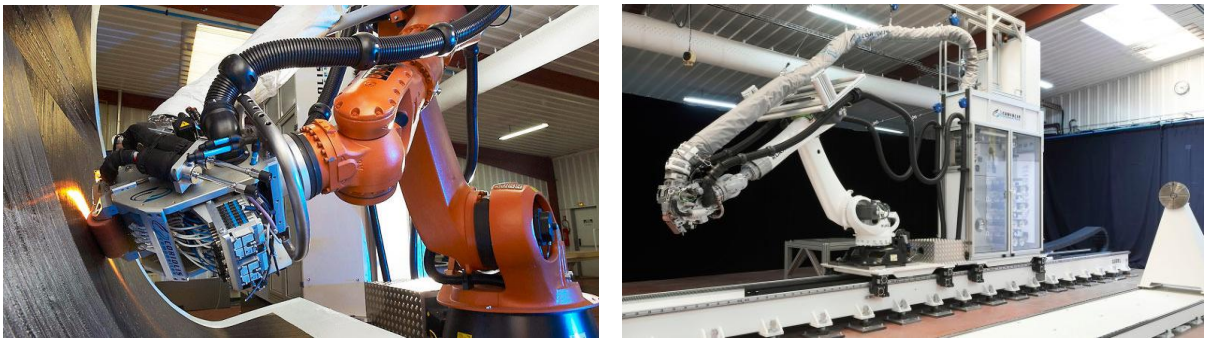


Figure 12 Robot-based AFP system developed by Coriolis Composite
© Coriolis Composite (www.coriolis-composite.com)

Electroimpact (USA) developed a modular placement head that can be mounted on the robot flange, it avoids using the long tow tube back to the refrigerated "creel house" attached to a major structural element mounted on a linear axis (Flynn et al., 2011, Rower, 2010, Rudberg et al., 2011), see Figure 13a. Such architecture is also used by NASA for producing composite parts for the agency's aeronautics and space exploration programs, see Figure 13b. Similar architecture is also launched by Automated Dynamics (USA).

(a) (b)

Figure 13 (a) Robot-based AFP system with modular head developed by Electroimpact, ©Electroimpact
(b) Robot-based AFP system in the Composites Technology Center of NASA, ©NASA

Mikrosam (Macedonia) is launching a system that integrates AFP head for complex parts and ATL head for flat molds into a single robotic cell for custom development of structural composites. It allows fast transition from automated fiber placement to automated tape laying and vice versa, by simply changing the head.

Current challenges still exist in design of robot based AFP, although such robotic systems have been widely employed in practice with good flexibility and adaptability. The main challenges can be related to several following terms:

- *Improvement of the fiber feeding system.* There are two representative designs of the fiber feeding system. 1) A "creel house" is situated at the foot of the robot and pipes individually feed each fiber from the creel to the head. It is developed by Coriolis Composite, see Figure 12b. 2) A modular head is made by embedding all the fiber creels into the head. This design is delivered by Electroimpact, see Figure 13a. For the former, feeding fibers from the creel to the head limits the robot movements; and in the latter case, the increased head dimension and weight by modular designing require to use heavy payload robot (relatively slower and less accurate) (Gallet-Hamlyn, 2011). The design of feeding system might be future optimized.
- *Automation of robot programming.* Productivity improvement can be expected from improved robot programming (Dirk et al., 2012). Since the robotic system usually contains two external axes of motion from rotational positioner and linear rail track, the system is kinematically redundant with respect to the fiber placement task. These two *redundancies* create some difficulties and complicate the robot programming, but it also gives some space for optimizing the robot and positioner movement to improve the productivity, for example, generating of the fastest trajectory by using the redundant degrees of freedom.

In the thesis, the challenge of productivity improvement via generating time-optimal motion at the stages of manufacturing process planning is mainly concerned and the problem of utilizing the redundant motion axes in best way is solved by generating the fastest lay-up motion for the redundant robotic system.

1.2 REDUNDANCY RESOLUTION IN ROBOTIC SYSTEMS

In order to analyze the redundancy problem, let us model the redundant robotic system with a set of links connected by rotational/translational joints. The configurations of joints are described by single scalar angle values. The complete configuration of the robotic system is specified by 8×1 vector $\mathbf{q}=[q_L, q_P, \mathbf{q}_R]^T$ where q_L describes the configuration coordinate of the linear rail track; q_P describes the joint coordinate of the positioner; $\mathbf{q}_R=(q_1,q_2,q_3,q_4,q_5,q_6)^T$ describes the angles of the robot joints. A 6-dimensional lay-up task location can be described as 6×1 vector $\boldsymbol{\chi}=[x, y, z, \alpha, \beta, \gamma]^T$, where $(x,y,z)^T$ is the position coordinate relative to the world frame and $(\alpha,\beta,\gamma)^T$ is the Z-Y-X Euler angles representing the orientation. Thus, a specific posture of the end-effector can be represented in two ways, i.e. $\mathbf{q}$ from joint space and $\boldsymbol{\chi}$ from task space.

The direct kinematic transformation is the mapping from the joint space to the task space. A point in joint space represents a unique location of the end-effector referred to the robot base frame in task space. The task kinematics can be described by $\boldsymbol{\chi}=g(\mathbf{q})$ where $g(\cdot)$ is the geometric transformation function. The standard Denavit-Hartenberg (DH) technique can be used to build the geometric model

of the robotic system (Hartenberg and Denavit, 1955), and the locations of the end-effector are presented as 4×4 homogeneous transformation matrices.

The inverse kinematic transformation is the mapping from the task space to the joint space. Compared to the direct kinematics, the inverse one is not trivial since the solution is not unique and cannot be expressed in a closed form. For this reason, special types of manipulator architecture are used to satisfy the Peiper condition (Peiper, 1968) that ensures the closed form solution. Otherwise, there are also some numerical techniques to deal with this problem (Pashkevich, 1997, Husty et al., 2007). In this thesis, the serial robot with last three intersecting axes is employed, which corresponds to the majority of industrial applications. In order to obtain a unique solution for the robot, the task kinematics referred to robot base can be described by $\mathbf{q}_R = g^{-1}(\boldsymbol{\chi},\mu)$ where μ is the configuration index that determines the posture of the robot shoulder, elbow and wrist. However, for the whole system with redundancies, it is still impossible to get a unique solution for $\mathbf{q} = g^{-1}(\boldsymbol{\chi},\mu)$ because the dimension of $\mathbf{q}$ is higher than that of $\boldsymbol{\chi}$.

For automated lay-up processes, the architecture of 6-axis robot plus 1-axis positioner and 1-axis workspace extensioner essentially complicates the preparation stage of robot programming since a location on the product can be reached with infinite numbers of $\mathbf{q}$. To generate the desired motion profile for a given task (lay-up path that is predefined as an augmented line), it is required decomposing the path in task space into robot motion and positioner motion in joint space. In literature, there are several works dealing with the redundancy resolution problems. Some representative approaches are presented as follows.

1.2.1 Redundancy Resolution via Generalized Inverse of the Kinematic Jacobian

The existence of the redundant motion axes leads to the insolvability of the system inverse kinematics, i.e. $\mathbf{q} = g^{-1}(\boldsymbol{\chi},\mu)$, that is because the dimension of $\mathbf{q}$ is higher than that of $\boldsymbol{\chi}$. In order to generate unique solution for $\mathbf{q} = g^{-1}(\boldsymbol{\chi},\mu)$, a redundancy resolution technique based on the generalized inverse of the kinematic Jacobian can be found in literature (Kazerounian and Nedungadi, 1988, Buss, 2004, Andres et al., 2012, Flacco and De Luca, 2015, Wang et al., 2010, Fahimi, 2008, Patel and Shadpey, 2005, Chiaverini et al., 2016, Siciliano, 1990, Fernandez and Cook, 1988, Nenchev, 1989). The most widely known generalized inverse of the kinematic Jacobian used for solving this problem is the Moore-Penrose pseudo-inverse.

The Moore-Penrose pseudo-inverse was proposed as a general way to find the solution to the equation $\mathbf{b} = \mathbf{A} \cdot \mathbf{x}$ where $\mathbf{b} \in R^m; \mathbf{x} \in R^n; \mathbf{A} \in R^{m \times n}$. In the case of $m < n$, a unique solution can be obtained with the form $\mathbf{x} = \mathbf{A}^+ \cdot \mathbf{b}$, where $\mathbf{A}^+ = \mathbf{A}^T \cdot (\mathbf{A} \cdot \mathbf{A}^T)^{-1}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ (Eldén, 1982, Ben-Israel and Greville, 2003). This particular solution has the property of minimizing the Euclidean norm of $\mathbf{x}$, i.e. $\|\mathbf{x}\| := \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \rightarrow \min$ (Buss, 2004).

To apply this technique, let $\mathbf{q} \in R^8$ be the joint coordinates of the considered redundant robotic system and $\boldsymbol{\chi} \in R^6$ be the Cartesian coordinates describing the 6-dimensional lay-up task. The task kinematics is given by the direct transformation $\boldsymbol{\chi} = g(\mathbf{q})$. By differentiating it with respect to time, the first-order kinematic can be expressed as follows:

$$\frac{d\boldsymbol{\chi}}{dt} = \mathbf{J}(\mathbf{q}) \cdot \frac{d\mathbf{q}}{dt} \tag{1}$$

where $\mathbf{J} \in R^{6\times 8}$ is the Jacobian matrix. It should be noticed that since the system here is redundant, the Jacobian matrix is not square. Then, from the above equations, the rates of displacement $d\mathbf{q}$ can be obtained by using the pseudo-inverse of Jacobian as follows

$$d\mathbf{q} = \mathbf{J}^{+}(\mathbf{q}) \cdot d\boldsymbol{\chi} \tag{2}$$

where $\mathbf{J}^{+} = \mathbf{J}^{T} \cdot (\mathbf{J} \cdot \mathbf{J}^{T})^{-1}$ (Whitney, 1969, Tucker and Perreira, 1987). Then, by defining $d\mathbf{q} = \mathbf{q}_c - \mathbf{q}_0$ and $d\boldsymbol{\chi} = \boldsymbol{\chi}_c - \boldsymbol{\chi}_0$ where $\mathbf{q}_0, \boldsymbol{\chi}_0$ and $\mathbf{q}_c, \boldsymbol{\chi}_c$ respectively represent the previous and the current system configurations and the corresponding locations in task space, the joint coordinates for each task location can be sequentially computed by using the following equation

$$\mathbf{q}_c \approx \mathbf{q}_0 + \mathbf{J}^{+} \cdot (\boldsymbol{\chi}_c - \boldsymbol{\chi}_0) \tag{3}$$

It should be mentioned that the initial $\mathbf{q}_0, \boldsymbol{\chi}_0$ have to be pre-determined.

Application Example. To verify the properties of solution from Moore-Penrose pseudo-inverse, let us apply it to a redundant robotic system composed of a two-axis planar robot and one-axis positioner. The geometrical parameters of the systems are given in Figure 14. This system possesses a 1-dof redundancy with respect to the planar task. The ellipsis-type lay-up path was described by a set of task locations, i.e. $\mathbf{p}_i = (x_i, y_i)^T$ where $i = 1,2,\ldots 100$.

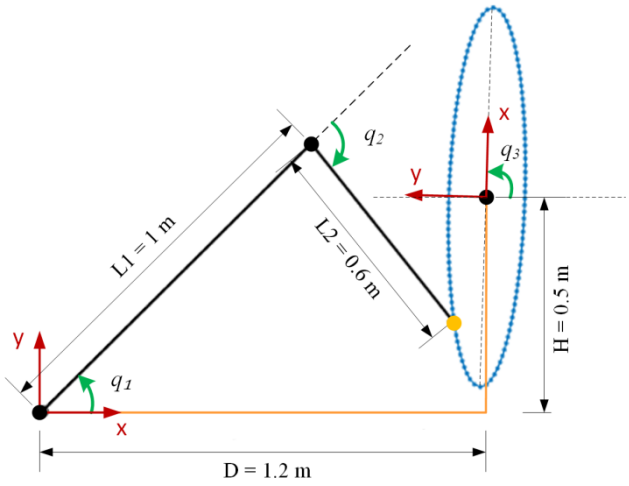


Figure 14 Redundant robotic system for application example

From the geometry of the robotic system, the position of the end-effector on the workpiece surface is written as the following function of joint parameters

$$\begin{cases} x = H\sin q_3 + D\cos q_3 + L_1\cos(q_1 - q_3) + L_2\cos(q_1 + q_2 - q_3) \\ y = H\cos q_3 - D\sin q_3 - L_1\sin(q_3 - q_1) - L_2\sin(q_3 - q_1 - q_2) \end{cases} \tag{4}$$

that is the direct kinematics of the robotic system. Then, by differentiating those equations, the Jacobian matrix is written as follows

$$\mathbf{J}=\begin{bmatrix} -L_1 S(q_1-q_3)-L_2 S(q_1+q_2-q_3) & -L_2 S(q_1+q_2-q_3) & L_1 S(q_1-q_3)+L_2 S(q_1+q_2-q_3)-DSq_3+HCq_3 \\ L_1 C(q_3-q_1)+L_2 C(q_3-q_2-q_1) & L_2 C(q_3-q_2-q_1) & -L_1 C(q_3-q_1)-L_2 C(q_3-q_2-q_1)-DCq_3-HCq_3 \end{bmatrix} \quad (5)$$

where *C* and *S* are shorthand for *cos* and *sin.*

By applying the equations (2) and (3) presented above, a particular solution with the minimization of $\|d\mathbf{q}\|$ is quickly obtained. However, when implementing the solution in Matlab software, the generated motion leads to collisions (see Figure 15) that are completely not acceptable.

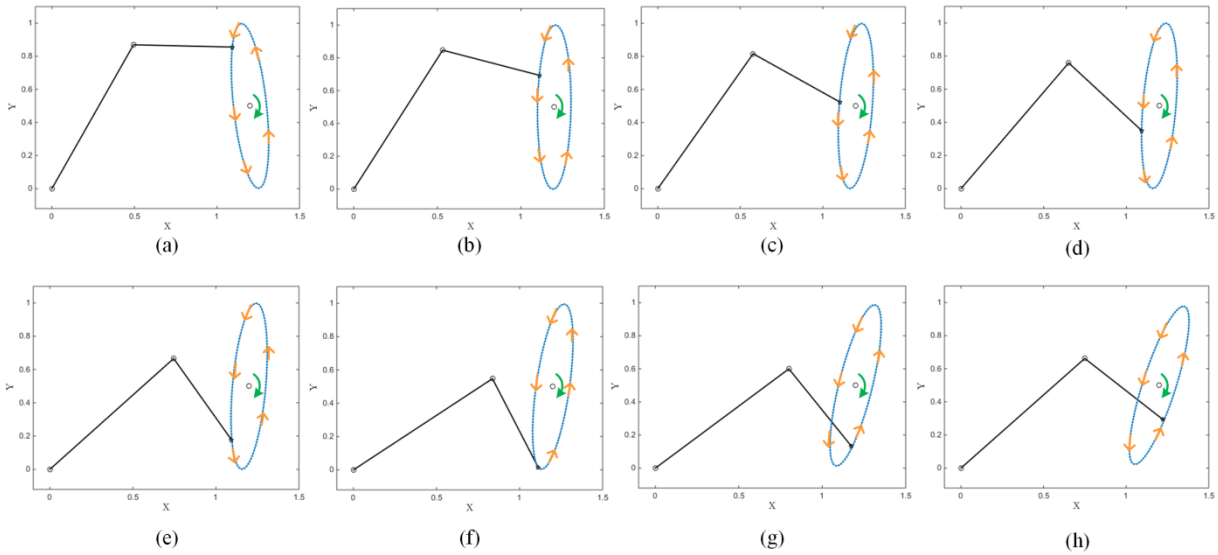


Figure 15 Simulation of the solution from pseudo-inverse for planar redundant robotic system: (a) ~ (h)

From the above application example, it is noticed that the redundancy resolution technique via generalized inverse of Jacobian can quickly generate a solution with the minimization of instantaneous power, since the norm of joint velocities relate to the energy consumption. However, it must be mentioned that the solution from this approach usually leads to unreasonable manipulator behavior (Padula and Perdereau, 2011, Klein and Huang, 1983), i.e. collisions between system components in the above example. Besides, this pure mathematical technique does not take into account the velocity and acceleration constraints of actuators. For these reasons, the technique based on generalized inverse of Jacobian cannot be straightforwardly applied for our technical problem (automated lay-up process).

1.2.2 Redundancy Resolution via Coordinating of Robot/Positioner Motions

Another idea for solving the inverse kinematics of the redundant system is to geometrically decompose the Cartesian path into a robot motion and a positioner motion. Related research have been carried out to develop robot-positioner coordinated motion planning since the middle of 1990s (Holmes et al., 1986, Jouaneh and Dornfeld, 1988, Jouaneh et al., 1990a, Jouaneh et al., 1990b, Ahmad and Luo, 1989, Alford and Belyeu, 1984).

A method of trajectory planning for a robot and a positioning table coordinated motion was proposed by Jouaneh (Jouaneh and Dornfeld, 1988) in welding applications. The planar tool motion

was resolved among a 5-axis manipulator and a 2-axis table, by using two geometrically defined parameters a_i and b_i representing the fraction of the incremental motion in the x and y axis directions, respectively moved by the robot in the i^{th} intervals along the welding path. Then, depending on the given path information, Jouaneh (Jouaneh et al., 1990b, Jouaneh et al., 1990a) extended this approach to two strategies for the motion coordination problem. The first one is driving two devices in opposite direction for simple path shapes, e.g. a straight line or a gentle curve. The second strategy that is used for cornered-paths, resolves the path with sharp corners into smooth ones. By this way, the trajectories for both of the table and the robot can be obtained in an off-line mode.

Another way of coordinated motion planning for redundant robotic systems (Tabarah et al., 1994, Gan et al., 2013, Gan et al., 2012) is based on the idea of "master-slave", where the trajectories of the "master" manipulator are assigned firstly, and the corresponding conjugate trajectories of the "slave" are then determined. This idea is simple and computationally efficient, but assigning the master trajectory is not trivial, especially for complex shape objects. Besides, here it is not possible to take into account the actuator constraints in an explicit way.

Application Example. To verify the properties of solution from motion coordination techniques, let us apply it to the same redundant robotic system that is already used in previous example. The geometrical parameters of the systems can be found in Figure 14. The ellipse-type lay-up path was described by a set of task locations, i.e. $\mathbf{p}_i = (x_i, y_i)^T$ where $i = 1,2,\ldots 100$.

The kinematics of the robotic system here is separately described by robot and positioner. From the geometry of the robot, the position of the end-effector with respect to the robot base frame (also selected as the world frame) is written as the following function of joint parameters

$$\begin{cases} x = L_1 \cos q_1 + L_2 \cos(q_1 + q_2) \\ y = L_1 \sin q_1 + L_2 \sin(q_1 + q_2) \end{cases} \tag{6}$$

that is the direct kinematics of the robot. Accordingly, its inverse kinematics is written as follows

$$\begin{cases} q_1 = \mathrm{a}\tan 2(y,x) - \mathrm{a}\tan 2(L_2 \sin q_2,\, L_2 \cos q_2 + L_1) \\ q_2 = \mu \cdot \mathrm{a}\cos\left(\dfrac{x^2 + y^2 - L_1^2 - L_2^2}{2 L_1 L_2}\right) \end{cases} \tag{7}$$

where $\mu = \pm 1$ and $\left|x^2 + y^2 - L_1^2 - L_2^2\right| / 2 L_1 L_2 \le 1$. Also, the perspective of the positioner, the task points can be described as follows

$$\begin{cases} x_i' = x_i \cos q_3 - y_i \sin q_3 + D \\ y_i' = x_i \sin q_3 + y_i \cos q_3 + H \end{cases} \tag{8}$$

that is the direct kinematics of the positioner.

Table 5 Joint limits and velocity limits of the redundant robotic system

Joint limits	Velocity limits
$-180° \le q_1 \le 180°$	$-20° \cdot s^{-1} \le \dot{q}_1 \le 20° \cdot s^{-1}$
$-180° \le q_2 \le 180°$	$-30° \cdot s^{-1} \le \dot{q}_2 \le 30° \cdot s^{-1}$
$-180° \le q_3 \le 180°$	$-40° \cdot s^{-1} \le \dot{q}_3 \le 40° \cdot s^{-1}$

To generate the coordinated motion, let us assign the positioner motion firstly as determining the robot motion is relatively harder. Here, the positioner actuator is simply driven at the maximum speed (see Table 6) within its pre-determined range, and the robot coordinates with the positioner. The positioner motion is described as $q_3(t_i)$ where $i = 1,2,\ldots 100$. Then, by applying the direct kinematics of the positioner and the inverse kinematics of the robot sequentially, corresponding $\mathbf{q}(t_i)$ can be obtained. The solution implemented in Matlab is shown in Figure 16.

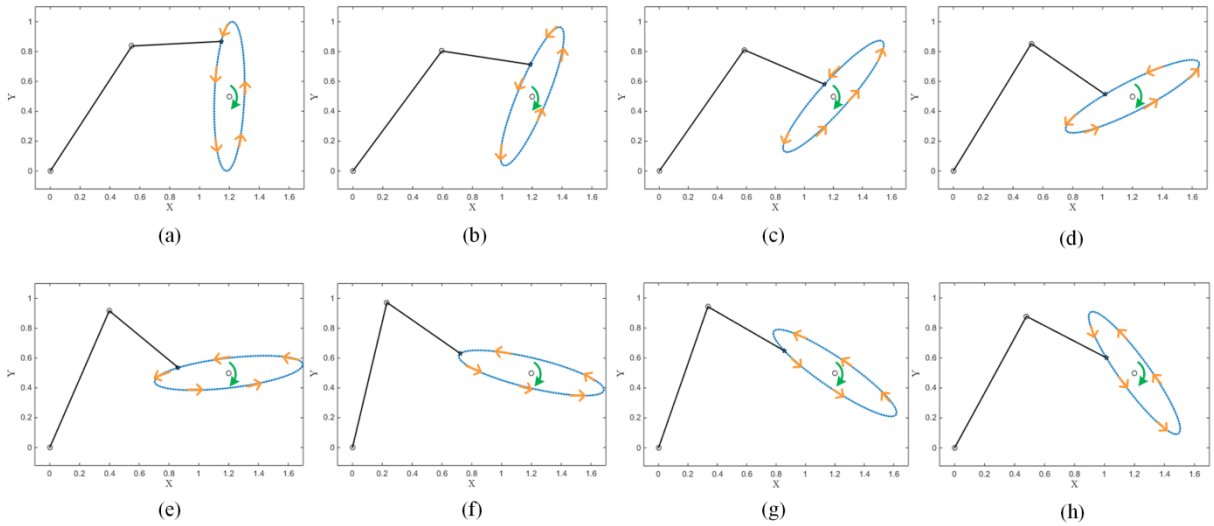


Figure 16 Simulation of the solution from motion coordination for planar redundant robotic system: (a) ~ (h)

From the above application example, it is found that robot and positioner motion generated via geometrically decomposing the Cartesian path shows a reasonable system behavior. Compared to the previous solution from generalized inverse of Jacobian, it avoids the collisions since these physical constrains can be considered when assigning trajectories for the two devices (in this example, if the rotational range of the positioner is limited to a small value, collisions will appear). However, it should be mentioned that for automated lay-up process, it is not always easy to decompose the placement path into a robot motion and a positioner motion since the product shape may be irregular and complex. Besides, this idea does not allow taking into account the velocity and acceleration constraints of actuators. For these reasons, the techniques based on this idea are not suitable for the considered applications.

1.2.3 Optimization Based Techniques for Redundancy Resolution

Besides the previously presented techniques, optimization-based methods are also frequently used for the redundancy resolution problem. This scheme tries to fully exploit the kinematic redundancy of the system in order to improve robot motion performance by using some criteria, e.g. minimizing the positioning error, obtaining smooth solutions in joint space, minimizing the displacements of the actuators, obtaining collision-free joint paths, etc. (Za'er et al., 2002). Several related work can be found in literature, and briefly shown in Table 6.

By reviewing the techniques shown above, it can be found that most of such techniques are based on heuristic search algorithms (mostly genetic algorithms, GAs) with the objective of minimizing the

positioning errors. In these approaches, the difficulties of solving inverse kinematics of the redundant system are avoided. In order to analyze the optimization-based techniques, let us briefly classify them into two categories: 1) for *point-to-point* motion generation; 2) for *continuous-path* generation.

Table 6 Main optimization-based techniques for redundancy resolution in literature

Publication	Method, objective function, constraints
(Parker et al., 1989)	GA; Minimize positioning error and the maximum joint displacement; *Point-to-point* motion; applied to 4R robot for (*x*, *y*, *z*) control.
(Nearchou and Aspragathos, 1996, Nearchou, 1998, Nearchou and Aspragathos, 1997)	GA; Minimize positioning error; Collision-free constraint; *Point-to-point* motion; applied to 7R robot for (*x*, *y*) control.
(Za'er et al., 2002)	Continuous GA; Minimizing accumulative deviation between the generated and the desired paths; *Continuous-path* motion; applied to 3R robot for (*x*, *y*) control / 6R robot for (*x*, *y*, *z*, *a*, *b*, *c*).
(Ata and Myo, 2005, Ata and Myo, 2006)	Generalized pattern search; Minimizing positioning error, joint displacement; *Continuous-path* motion; applied to 3R robot for (*x*, *y*) control;
(Pires et al., 2007, Pires and Machado, 2000a, Pires and Machado, 2000b)	GA; Minimizing positioning error, joint displacement, energy, etc.; *Point-to-point* motion; applied to 3R robot for (*x*, *y*) control.
(da Graça Marcos et al., 2009)	GA with closed-loop pseudo inverse; Minimizing positioning error, maximum joint displacement, etc.; *Point-to-point* motion; applied to 3R robot for (*x*, *y*) control.
(Menasri et al., 2015)	Bi-GA; Minimizing positioning error and maximizing the robot manipulability; Collision-free constraint; *Point-to-point* motion; applied to 3R robot for (*x*, *y*) control.
(Debout et al., 2011)	Optimizing redundant axis with Levenberg–Marquardt method; Minimizing tool path length and the curvature variation; *Continuous-path* motion; applied to 7R robot for (*x*, *y*, *z*, *a*, *b*, *c*) control.
(Gueta et al., 2009a, Gueta et al., 2008b)	Graph based search space representation; Minimizing total travelling time; *Point-to-point* motions; applied to 6R robot and 1R table for (*x*, *y*, *z*, *a*, *b*, *c*) control.
(Dolgui and Pashkevich, 2009, Dolgui and Pashkevich, 2006, Pashkevich et al., 2004)	Graph based search space representation; Minimizing joint displacements; *Continuous-path* motions; applied to 6R robot for (*x*, *y*, *z*, *a*, *b*) control.

To generate *point-to-point* motions, a representative approach of redundancy resolution was presented by Parker et al. (Parker et al., 1989). This GA based method minimizes the positioning error and the maximum joint displacement by applying following expression

$$\left\|\boldsymbol{\chi}_i - \boldsymbol{\chi}_f\right\| + k \cdot \max\{\left|\mathbf{q}_i - \mathbf{q}_f\right|\} \rightarrow \min \tag{9}$$

where $\boldsymbol{\chi}_i$ and $\boldsymbol{\chi}_f$ represent the initial and final Cartesian position of the end-effector; $\mathbf{q}_i$ and $\mathbf{q}_f$ are the initial and final joint coordinate of the robot; factor k scales the contributions from each term. In order to take into account the collision constraint, Nearchou et al. (Nearchou and Aspragathos, 1996, Nearchou, 1998, Nearchou and Aspragathos, 1997) presented a similar GA based approach while the geometries of obstacles are described in constraint function. It allows generating a point-to-point solution subject to a collision-free movement.

To generate joint trajectories for a *continuous-path* in task space, that is more interested in automated lay-up process, Za'er et al. (Za'er et al., 2002) proposed a technique based on continuous genetic algorithm taking into account the singularity-free constraints, which minimizes the accumulative deviation between the two paths given by the following formula

$$E = \sum_{i=1}^{n}\sum_{k=1}^{N}\left| P_{dc}(k,i) - P_{gc}(k,i)\right| \tag{10}$$

where n is the number of the task point on the path; N represents the number of the joint axis; P_{dc} is the desired Cartesian location and P_{gc} is the generated Cartesian location. The optimal solution of the problem is obtained when the deviation function, E, approaches zero.

Ata et al. (Ata and Myo, 2005) presented a multi-objectives optimization technique that is realized by implementing GA and GPS (generalized pattern search) to lead the end-effector along a straight path or a circle path. The main objective is to minimize the sum of the positioning error of the end-effector at each intermediate point, and the sub-objectives are expressed as joint displacement, velocities, etc. The objective function is expressed as follows

$$F = \sum_{i=1}^{m} w_i \cdot E_i \tag{11}$$

where w_i is the weighting factor to control the desired configuration which satisfy the constraint $\sum_{i}^{m} w_i = 1$, and m is the number of the objectives.

Application example. The case study presented above can be also used to test the GA-based technique (see Figure 14), and the geometry of the robotic system is described as Equation (4). The problem can be formulated as follows

$$\begin{cases} \text{find:} \\ \min \quad F = (1-\eta)\cdot\sum_{i=1}^{100}\left(\left|x_i - g_x(q_1^{(i)}, q_2^{(i)}, q_3^{(i)})\right| + \left|y_i - g_y(q_1^{(i)}, q_2^{(i)}, q_3^{(i)})\right|\right) + \eta\cdot\sum_{i=1}^{99}\sum_{j=1}^{3}\left|q_{(j)}^{(i+1)} - q_{(j)}^{(i)}\right| \\ s.t. \\ -180^\circ \le q_{1,2,3}^{(i)} \le 180^\circ \quad i = 1,2,\ldots 100 \\ \text{intersection} - \text{free} \end{cases} \tag{12}$$

where g_x and g_y are geometric model of the system, based on Equation (14); $\mathbf{p}_i = (x_i, y_i)^T$ is the i^{th} point on the Cartesian path, F can be set as the fitness value of genetic algorithm. Then, by applying the standard genetic algorithm, i.e. ga(·) in Matlab, with the setting parameters shown in Table 7 and randomly generated initial solution, a result for 300 decision variables is generated within around 10 hours (Matlab2014b in Win7SP1 with Intel® i5 @2.67GHz 2.67GHz). Figure 17a visualizes the progress of the algorithm, which is provided in the options of ga(·). It shows that the algorithm halted when reached ~3500 generations, with best fitness value F=14.6137. However, such straightforwardly obtained solution is not ideal. As is shown in Figure 17b, the positioner joint coordinate for 100 task points does not vary continuously which brings sharp jumps in the trajectories of the robotic system. This phenomenon is possibly caused by the randomly generated initial values. Another reason is that the motion smoothness is hardly expressed as a constraint during the path searching. Besides, this solution does not provide acceptable positioning accuracy.

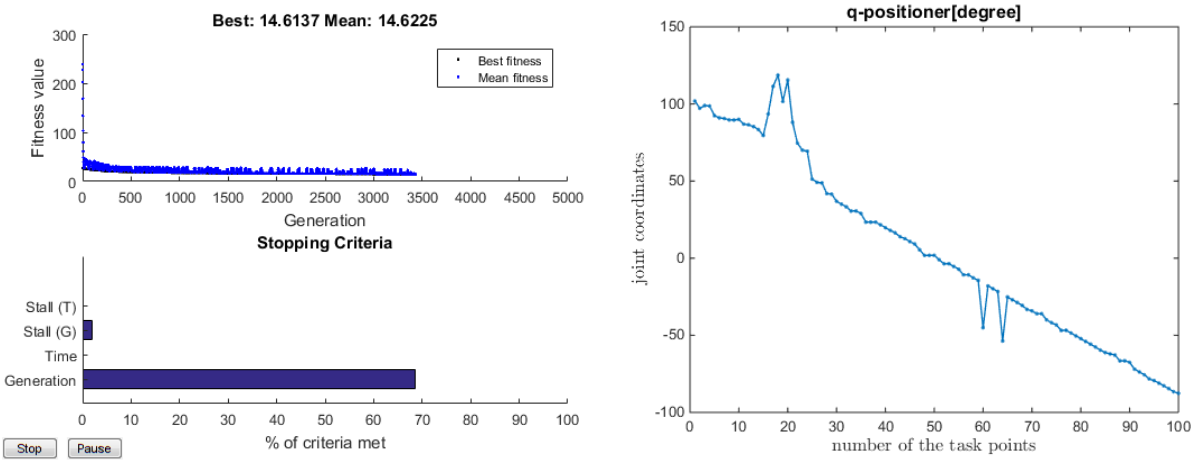
Table 7 Parameters setting of ga(·) function in Matlab

Population Size	200
Fitness scaling	Rank
Selection	Stochastic uniform
Elite count	10
Mutation	Gaussian (scale =1; shrink =1)
Crossover	Scattered function; Fraction =0.8
Migration	Forward (fraction= 0.2, Interval= 20)
Generation	MaxGenerations = 5000; MaxStallGenerations = 100

By analyzing several solutions obtained from GA-based technique, it can be found that, using such GA-based straightforward techniques, only the direct kinematic equation of the robotic system is required, and the difficulties related to the inverse kinematics does not exist here. Also, singular configurations and collision cases can be avoided by adding the constraint function. Obviously, it brings some simplicity for the redundancy resolution. However, these straightforward mathematical methods do not take into account properties of the systems kinematics, which leads to some unreasonable movements. Additionally, it has to be noticed that the result qualities are strongly related to the initial values of the variables. But, in automated lay-up processes, selection of these initial values is non-trivial since the Cartesian path in task space is usually complex. A reference test is done on a simple task, i.e. straight line, with simply estimated initial values, better solutions can be obtained. It proves that this straightforward GA based approach is more suitable for the tasks with non-complex Cartesian path.

Besides the above illustrated techniques, there are some other types of optimization based methods in literature. For example, a method was developed for fiber placement process with a specific designed 7-axis robot (Debout et al., 2011). The redundancy resolution is performed by optimizing one axis control value all along the tool path using a least square optimization algorithm. And then, the other 6 axes values can be computed analytically using inverse kinematics. The technique generates a local solution in the objective of fastest tool path after several iterations and evaluations of an objective function which takes into account the variations of several axes. Obviously, compared to the previous optimization-based techniques, this approach ensures exactly positioning, and also contributes to the trajectory smoothness in the configuration space and manufacturing efficiency by

minimizing variations of the joint coordinates. But, it does not allow using full capacities of all the actuators in the robotic system.



(a) (b)

Figure 17 (a) visualization of the GA progress for considered application example; (b) Joint coordinates of the positioner from GA based solution.

Another efficient strategy is based on converting the problem into a discrete one, where the robotic manipulator and the positioner joint spaces are discretized and the desired trajectory is represented as the shortest path on the corresponding graph. One of such techniques was developed by Gueta (Gueta et al., 2011a, Gueta et al., 2009b, Gueta et al., 2008a, Gueta et al., 2009a, Gueta et al., 2011b, Gueta et al., 2008b, Gueta et al., 2017) to generate a set of *point-to-point* motions for a robot and a rotary table in the objective of minimum travelling time in multi-goal applications. Slightly different approach was proposed in (Dolgui and Pashkevich, 2009, Pashkevich et al., 2004, Dolgui and Pashkevich, 2006) for laser cutting applications with *continuous-path* task in Cartesian space. In this case, it minimizes the oscillations in actuator velocities, taking into account the collision constraint when assuming the tool speed is constant.

Table 8 Summary of the related works for motion coordination redundant robotic system

Methods	Properties and applications
Pseudo inverse of the kinematic Jacobian	Singular configuration may be generated; Constraints are hardly applied; Cannot be straightforwardly used in industry.
Robot/Positioner **motion coordination**	Cannot use the redundancy optimally; Can be used in the case of simple path task. (e.g. some arc-welding applications, etc.)
Optimization-based technique (GA based straightforwardly search)	Unreasonable movements may be generated; Might be used for pick and place applications; Might be used in the case of simple path task.
Optimization-based technique (Graph based search space representation)	Can be used for continuous-path task; (e.g. laser cutting, arc-welding, etc.) Can be used for multi-goal task; (e.g. spot-welding, etc.)

By reviewing all the techniques illustrated above, a small summary for the redundancy resolution techniques is presented in Table 8. It shows that only the graph based search space representation can be applied on the automated lay-up processes since it allows generating trajectories for tasks with complex path in task space. However, the aforementioned techniques considering graph based search space representation were respectively designed for multi-goal tasks and continuous-path tasks assuming constant tool speed cases. They do not allow taking into account the kinematic constraints of the robotic system completely, i.e. velocity and acceleration constraints of all the actuators. In addition, they cannot meet the demands of high-speed automated lay-up processes since they do not generate time-optimal motions for a continuous-path task.

To our knowledge, there are no techniques directly addressing the problem of the time-optimal motion planning for robotic lay-up system. It will be very significant to develop a comprehensive methodology to achieve time-optimal smooth motion generation in redundant robotic system for the robotic automated lay-up applications, which is in the focus of the thesis.

1.3 ROBOTIC CELL DESIGN AND PROGRAMMING FOR COMPOSITE PRODUCT MANUFACTURING

In order to implement the robot-based automated lay-up process on the factory floor, CAD/CAM software is usually used for both designing a composite product, creation of the robotic system and programming movements of relevant equipment (robot, positioner and linear unit). The CAD software creates models and assemblies that are used by the CAM portion to generate tool paths that drive the robotic system to turn the designs into physical parts. This section provides some details on the manufacturing process preparation as well as robotic system design and programming using modern software tools. It also highlights some difficulties arising here, which are resolved in the following chapters of the thesis.

1.3.1 Manufacturing Process Planning for Robotic Lay-up Applications

Computer aided manufacturing process planning is the systematic determination of manufacturing methods and operations details by which parts can be produced economically and efficiently from raw materials to finished products (Leondes, 2000). According to (ElMaraghy and Nassehi, 2014), the main focuses of this procedure are the optimal selection of equipment, proper tuning of the process parameters and creation of numerical control codes. Usually, the manufacturing process planning involves a series of key steps that can be itemized as follows (Alting and Zhang, 1989, Bagge, 2014): (i) interpretation of the product design data; (ii) selection of the process; (iii) selection of equipments and fixtures; (iv) generation of movements ensuring desired process; (v) creation of process sheets, including numerical control programs.

For the robotic lay-up processes (see Figure 18), the *manufacturing process preparation* stage includes the workpiece 3D modeling, generation of the required path for the technological tool and the path presentation in the form of sequence of the robot end-effector locations. In this work, it is assumed that the CAD model of the manufacturing task is known and was already created using dedicated computer aided design system. In particular, the desired lay-up path ensuring covering of the

workpiece by fiber reinforcements is generated as a 3D augmented curve on the surface of the product. This curve defines both positions and orientations of the technological tool while it is moving along the path. For further convenience, this augmented line is discretized and is presented as a sequence of location vectors (or homogeneous matrices) describing the postures of the technological tool with respect to the product. The density of the discretization is defined by the user, as well as the type of the sampling (regular or irregular).

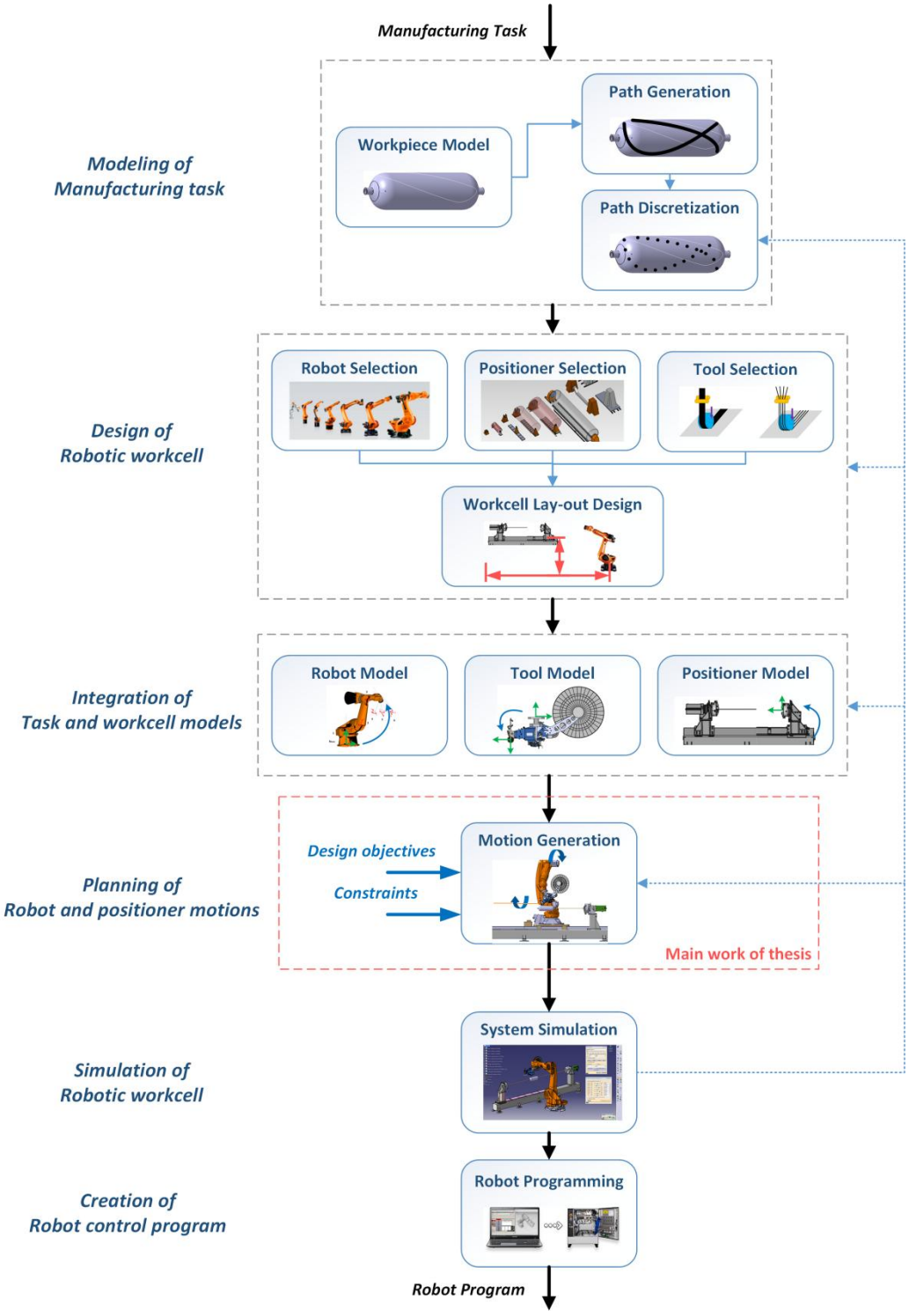


Figure 18 Manufacturing process planning for robotic lay-up processes

The next stage deals with the *design of robotic workcell*. It includes selection of the robotic system components (manipulator, positioner, workspace extension unit, technological tool, etc.) as well as their space arrangement. The components selection essentially depends on the product size and its geometry. In particular, an AFP head guiding multiple fibers is more suitable for complex shapes, while an ATL head is more appropriate for lay-up over flat surfaces (with high curvature radius). To manipulate the technological tool, which must be properly positioned and oriented with respect to the product, serial industrial robots with 6 degrees of freedom are usually used. Their selection is mainly based on the workspace size that must suit product geometry. Besides, the robot payload is also taken into account to ensure capacity of the tool handling. To implement the desired lay-up path that usually includes a number of quasi-circular segments, a rotational positioner is commonly employed. Its selection depends on the product size/weight. Besides, for some large dimensional products, a robot workspace extension unit (linear track, etc.) is used, which is selected to be in agreement with the product size. After selecting the workcell components, the designer decides on their mutual location that must ensure accessibility of the lay-up path avoiding collisions between the technological tool, manipulator, positioner and the workpiece. The latter procedure is also often referred to as the robotic cell layout design.

At the third stage, the geometric models of the workcell components and the given lay-up task are integrated. This operation requires direct/inverse kinematic models of all workcell components and relevant geometric presentation of the lay-up task, which must be included in a basic relation describing the closed kinematic loop “manipulator-tool-workpiece-positioner-manipulator”. It is clear that this kinematic relation includes some redundant variables and it must be treated as the principle constraint while planning the motions in the considered robotic systems. More details concerning the *integration of the task and workcell models* are given in Chapter 2.

At the fourth stage, *planning of robot and positioner motions*, the movements of the workcell components (robotic manipulator, positioner and the workspace extension unit) are determined taking into account the design objectives and technological/physical constraints. The latter include the lay-up path specification, physical limits of the actuators, geometrical limits on the actuating variables as well as some additional limitations describing minimum distances to singularities and collisions. It is obvious that the considered motion planning problem cannot be solved in a unique way because of kinematic redundancy, which gives some space for optimization of workcell components movements. This optimization is the main issue studied in this work; relevant contributions are presented in Chapter 3.

At the fifth stage, *robotic cell simulation*, the obtained optimal motion is carefully examined using industrial CAD/CAM packages (such as Robcad, DELMIA, etc.). This allows designer to verify accurately the manipulator/positioner joint limits, the collision constraints and also examine the lay-up path visually. If some problems are detected during the simulation, the designer can modify some settings in the motion planning algorithm and repeat the previous stages.

Finally, at the sixth stage, the verified optimal motion is described using the language of the robotic system controller. Relevant post-processors are usually integrated in the robotic simulation packages allowing the user *creating the control program*, where the motions of the manipulator, positioner and linear unit are presented as the sequence of linear, circular or spline-based segments.

It should be mentioned that the above described procedure of the robotic lay-up process planning is *iterative*. In particular, some steps can be repeated several times before arriving to the suitable result. Besides, there are a number of commercial software packages that can be applied to some of the process planning stages. Their brief review and applicability analysis is presented in the next sub-section.

1.3.2 Software Packages for Computer Aided Design of Industrial Robotic System

At present, there are a number of software packages on the market that help users to design robotic cells for some specific applications such as arc or spot welding, water-jet or laser cutting, painting, spraying, etc. They are available either as standalone applications or as a set of modules embedded in the universal CAD/CAM systems. Usually they offer the users a 3D interactive graphical simulation environment and some convenient tools for robotic cell layout design, manipulator motion planning, workcell simulation and offline programming. The most common of these packages are summarized in Table 9.

Table 9 Software packages for computer aided design of industrial robotic systems

Software Package	Company	Principle applications
Robcad	Tecnomatix www.siemens.com/tecnomatix	Welding, cutting, drilling, riveting, painting, spraying, etc.
DELMIA (IGRIP)	Dassault Systems www.3ds.com	Welding and material handling applications.
Robotmaster	Hypertherm www.robotmaster.com	Cutting, 3D machining, trimming, welding, polishing, dispensing, de-burring, painting, spraying, etc.
Workspace5	WAT Solutions www.workspace5.com	Welding; Waterjet cutting.
CATFiber for CATIA CADFiber for NX	Coriolis Composite www.coriolis-software.com	Fiber placement; Ultrasonic Trimming; Non-destructive testing.
FPM/FPS	Automated Dynamics www.automateddynamics.com	Fiber placement; Tape laying
KUKA.Sim	KUKA www.kuka.com	Welding, palletizing, coating, painting, machining.
RobotStudio	ABB http://new.abb.com	Application software for cutting, machining, welding and palletizing.

In early works devoted to this subject, computer aided design of robotic systems was mainly related to the workcell layout (Zhang and Fang, 2017). The primary attention to this subject was caused by its

high influence on the system productivity, since an ill-placed robot risks inefficient operation and even failure. To make a proper layout design with the aid of graphical simulation software, there are some critical issues highlighted by (Lueth, 1992) who mentioned that the well placed robot must reach all task points without collisions and ensure collision-free movement in their neighborhood. The simplest strategy to find optimum robot location is based on straightforward testing a 2D-grid of possible robot base positions (Barral, 2003). This test is usually performed either automatically or in an interactive mode, using relevant interface. More sophisticated modern software packages also allows user to generate collision-free movements, optimize the robot cycling time, simulate the manipulator motions, create and debug the robot control program, etc. Let us give some details in the most common software packages for computer aided design of the robotic systems.

One of the most popular commercially available packages, Robcad (www.siemens.com/tecnomatix), is a robotic cell design, simulation and programming system, which provides a platform to handle the task throughout the various steps of the process planning. Robcad has a comprehensive library of standard robots, technological tools, machines and equipments from KUKA, ABB, FANUC, YASKAWA, etc. Its layout design tool allows displaying graphically the robot envelope and indicating whether the task points are reachable or not (showing them in red or green). Using such environment, the user can easily modify the robot placement in an interactive mode to ensure implementation of the given task. In some cases, a suitable robot location can be found automatically. In addition, Robcad is able to generate some typical motions taking into account specific features of the robot controller. With the RRS (realistic robot simulation) module, it offers extremely accurate cycling time calculation and online collision detection. At the final stage, Robcad OLP (offline programming) module transforms the obtained motion into the program codes suited to the relevant control system (over 200 types from KUKA KRC, ABB IRC, YASKAWA YRC, etc.).

It should be also mentioned that Robcad allows data exchange with the most mainstream CAD systems by supporting such data formats as JT, IGES, DXF, STL and STEP. This interoperability allows the manufacturing task created in a dedicated CAD environment to be easily imported. Besides, Robcad includes several process-specific modules such as Robcad Spot, Robcad Arc, Robcad Paint and Robcad Cut, which concentrate on the robot-based automation of spot/arc welding, painting, sand blasting, shot peening, flaming, thermal spraying, laser/water-jet/plasma cutting, sealing and gluing. For example, for the spraying process, Robcad Paint enables generating the robot path, verifying access to all areas, determining coverage parameters and thickness, creating and adjusting process triggers, simulating and downloading the optimized program to the shop floor. However, for the composite layup application studied in this work, there is no specific tool integrated in Robcad.

Another software package, Robotmaster (www.robotmaster.com), provides similar functionalities such as the workcell layout design, automated robot motion optimization, workspace simulation and control code generation. Its application areas include numerous metal processing technologies such as machining, cutting, trimming, polishing, de-burring and also painting, spraying, surface treatment, etc. For example, for the robot-based machining, the user can define the cutter shape, adjust the cutter diameter as well as the depth and the number of cuts. This adjustment is simplified with the modifiable screens that define the interaction, terminology and control setting. At the motion generation stage, the cutter orientations and manipulator configurations can be managed automatically to minimize the wrist rotation, maximize the robot reach and ensure the optimal milling trajectories. At the final stage, the desired robot motion is simulated and programmed automatically. In addition, Robotmaster contains an extensive set of programming and simulation tools for external axes such as rails and rotaries,

which allows user to design workcells with robots mounted on linear tracks/gantries or with workpieces mounted on actuated positioners, where up to eight axes are controlled simultaneously. However, Robotmaster cannot address the entire composite layup process considered in this work, and it does not allow generating the desired optimal motions for the technologies studied here.

Among the robotic CAD systems, it is also worth mentioning DELMIA IGRIP (www.3ds.com) from Dassault. It deals with robotic arc welding and allows generating automatically a manipulator end-effector path based on the welding seam geometry and kinematics of the welding positioner, which ensures workpiece optimal orientation with respect to gravity. Its Arc Weld Macro Programming (AMP) module allows the user to generate robot trajectories for complex welding seams, validate them in simulation environment, create the control program, download it into the robot controller and debug the program on the shop floor. Other similar software packages widely used in industry are Workspace5, KUKA.SIM and ABB RobotStudio. The latter two developed by the primary robot manufacturers provide very realistic simulation and programming environment while using their own equipment, but their functionalities for different technological processes are rather limited (Hasenjaeger, 2013).

For the technology considered in this work, the robot-based composite laying, there is very small number of software systems supporting design and offline programming of robotic cells. One of the most known is CATFiber integrated in CATIA/DELMIA (www.coriolis-composites.com). It allows user to design composite parts in CATIA and simulate automated fiber placement cells with DELMIA offline programming module. Within this package, it is possible to define the desired fiber propagation pattern (parallel, geodesic, etc.) and to set the required fiber placement parameters (steering radius, angular deviation, minimum tape length and fibers overlapping). Similar to other robotic CAD systems, DELMIA includes modules for the workcell components selection (robot, linear track/gantry, technological tool and workpiece positioner from ABB or KUKA), robotic cell layout design, workcell motion planning, simulation and offline programming. However, handling redundant degrees of freedom caused by linear unit and positioner creates some difficulties and requires essential effort of the user who improves the trajectory interactively. For non-CATIA users, Coriolis Composite also provides a package CADFiber directly interfaced with SIEMENSNX. Another software package devoted to fiber placement and tape laying processes, FPM/FPS suite in conjunction with SolidWorks, was developed by Automated Dynamics but the latter has very limited functionality for the kinematic redundancy resolution and optimal motion planning for the layup process.

Hence, there still exists a gap between capabilities of the commercial robotic software packages and requirements of the particular technology. There is no comprehensive robotic CAD system on the market that implement the complete process planning procedures for the robot-based lay-up processes, despite some common functionalities such as workcell layout design, collision check and cycle time evaluation that are available in all above mentioned software. In our case, the robotic system contains one or two redundant degrees of freedom, which makes the motion planning very complicated by using commercial robotic CAD systems. In particular, the existing robotic software packages are not able to generate automatically optimal manipulator/positioner motions desired in the automated layup process. So, in practice, the redundancy resolution is usually performed in an interactive mode (Pan et al., 2012). For these reasons, this thesis is aimed at developing optimal motion planning methods for robot-based layup processes, which should allow generating time-optimal movements of the robot, linear track and positioner using acceptable computing efforts.

1.4 THESIS GOAL AND RESEARCH PROBLEMS

In automated manufacturing of polymer matrix composite products utilization of robotic system is economically attractive since they allow easily changing the product type. Typical robotic systems for such applications are kinematically redundant. Usually they are composed of a 6-axis manipulator, a 1-axis positioner and a 1-axis workspace extension unit (a linear track or gantry), which yields two extra degrees of freedom with respect to the considered technological task. This redundancy provides some convenience for the robotic cell programming but it also creates some difficulties in the motion planning for the mechanical components. On the other hand, the redundancy gives some space for increasing the workcell productivity by optimal coordination of their movements and using full capacities of the system actuators.

In the considered technology, a manipulator usually implements the lay-up process that requires smooth continuous motion of the fiber (tape) laying head along the specified path with the maximum achievable speed. In literature, there are known several motion planning techniques for similar redundant systems but they assume that the manipulator end-effector speed is constant. Hence, they cannot be applied directly for the considered processes where the laying head speed can vary in some degree and minimization of the processing time is required. Besides, existing commercial CAD/CAM software packages cannot solve perfectly the problem of the optimal motion planning for the lay-up process. To our knowledge, at present there are no techniques or software products, which directly solve the problem of the time-optimal motion planning for robotic systems in the lay-up applications.

For the above mentioned reasons, the thesis focuses on **the optimal motion planning in redundant robotic systems allowing improving productivity of the automated composite lay-up workcell**. Special attention is paid to the motion coordination of the robotic manipulator, workpiece positioner and workspace extension unit, which ensures the shortest manufacturing time and smooth movements of all mechanical components. To achieve this goal, the following tasks have to be solved:

Task #1:

> Analysis of existing systems for composite product manufacturing, detecting their weak points and comparative study of known methods for the motion planning in redundant robotic systems.

Task #2:

> Modeling of typical redundant robotic system used in the composite lay-up technology and formalization of the related optimal motion planning problems.

Task #3:

> Development of the optimization algorithms for coordinated motion planning in the redundant robotic system composed of the robotic manipulator, workpiece positioner and workspace extension unit, which allow minimizing the total motion time and ensuring smooth movements of all mechanical components.

Task #4:

> Application of the developed algorithms to real industrial problems, development of the robot control programs implemented generated time-optimal trajectories, simulation of the coordinated movements in 3D environment and experimental validation of the developed techniques on the factory floor.

To solve these tasks, the remainder of the work is organized as follows. Chapter 2 focuses on the kinematic modeling of a typical redundant robotic system used in the composite lay-up technology and formalization of the related optimal motion planning problem. Chapter 3 is devoted to the development of a new motion planning method for the redundant robotic systems utilized in the considered process. Chapter 4 deals with industrial implementation of the developed method on the factory floor (for manufacturing of a high-pressure vessel).

CHAPTER 2 ROBOTIC LAY-UP SYSTEM MODEL AND MOTION GENERATION PROBLEM FORMALIZATION

This chapter focuses on the kinematic modeling of a typical redundant robotic system used in the composite lay-up technology and formalization of the related optimal motion planning problem. The considered problem is presented as finding of the time-optimal trajectory in robotic system joint space under specific constraints related to both the robotic system and technological task. These constraints allow taking into account limitations of the robotic system (joint limits, maximum joint velocities/accelerations), practical requirements for the manipulator postures (via singularities and collision constraints) as well as the Cartesian path constraints issued from the composite lay-up process. The main purpose of this chapter is to formalize the considered technological problem and to present it in the form common for the mathematical optimization theory.

2.1 LAY-UP TASK DESCRIPTION AND ITS CREATION IN CAD SYSTEM

Robotic offline programming for automated lay-up process starts from a 3D CAD model of the workpiece that is to be covered with fiber reinforcements. There are various types of CAD files, but most offline programming software packages are capable of converting other types of CAD files to a compatible one (Pan et al., 2012). For example, in Figure 19, a 3D model is shown in FPM© that is developed by AUTOMATED DYNAMICS®.

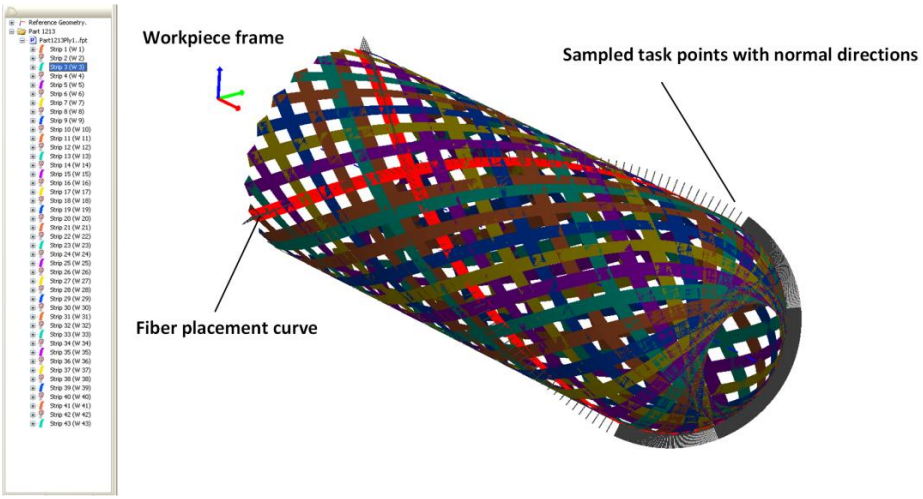


Figure 19 Typical fiber placement curves defined in 3D CAD model (SolidWorks), with position and normal direction relative to the workpiece frame ©AUTOMATED DYNAMICS® (www.automateddynamics.com).

This software works in conjunction with SolidWorks. The user can develop the part surface in SolidWorks directly or import the surface geometry as an IGES, STEP, or other CAD translation. Such universal CAD/CAM designs allow users to create the lay-up curve data files and extract position as well as orientation tags directly. The layup task can be also described in CATIA, which is also able to operate with surfaces and augmented lines defining both positions and orientations of the technological tool for the considered task.

2.1.1 Representation of the Lay-up Task in the Form of the Sequence of 4×4 matrices

Let us assume that the desired lay-up curve with respect to workpiece frame, along which the placement head is to be moved, is imported from the composite part CAD system and described by a 3D-augmented line (as the typical task shown in Figure 19). This augmented curve is discretized in n segments and can be expressed as follows

$$\mathbf{C} = \left\{ \mathbf{p}_i, \hat{\mathbf{a}}_i \;\middle|\; i = 1,2,\ldots n; \right\} \tag{13}$$

where $\mathbf{p}_i = (x_i, y_i, z_i)^T$ defines the Cartesian coordinates of each sampled task point on the lay-up curve and the unit vector $\hat{\mathbf{a}}_i = (\hat{a}_{xi}, \hat{a}_{yi}, \hat{a}_{zi})^T$ defines the normal direction of each task location outside the workpiece surface.

To describe the spatial location of each task point, let us define the displacement along the Cartesian path

$$\Delta \mathbf{p}_i = \mathbf{p}_{i+1} - \mathbf{p}_i; \quad i = 1,2,...n-1 \tag{14}$$

and derive the unit vector of roller axis by following equation

$$\hat{\mathbf{s}}_i = \frac{\hat{\mathbf{a}}_i \times \Delta \mathbf{p}_i}{\left\| \hat{\mathbf{a}}_i \times \Delta \mathbf{p}_i \right\|}; \quad i = 1,2,...n-1 \tag{15}$$

Then, the traveling direction of the placement head on each task location is described as the following vector product

$$\hat{\mathbf{n}}_i = \hat{\mathbf{s}}_i \times \hat{\mathbf{a}}_i; \quad i = 1,2,...n-1 \tag{16}$$

Thus, each task node can be associated with a Cartesian frame in which the frame origin is located at the path point $\mathbf{p}_i$; X-axis is directed along the path and coincides with the path tangent; the Z-axis is along norm direction; the Y-axis perpendicular to the axes X and Z so that these three axes form a right-handed coordinate frame (see Figure 20), i.e. $F_{task}^{(i)}; i = 1,2,...n-1$. The discretized task locations are described by homogeneous transformation matrix from workpiece frame F_W to $F_{task}^{(i)}$

$${}^{W}\mathbf{T}_{task}^{(i)} = \left(\begin{array}{ccc|c} \hat{\mathbf{n}}_i & \hat{\mathbf{s}}_i & \hat{\mathbf{a}}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{array} \right)_{4\times4}; \quad i = 1,2,...n-1 \tag{17}$$

Using the above definition, the lay-up task can be presented as a set of locations that should be visited sequentially by the robot end-effector. Since each location is already described as a 4×4 homogenous transformation matrix, the considered task is formalized as follows

$${}^{W}\mathbf{T}_{task}^{(1)} \rightarrow {}^{W}\mathbf{T}_{task}^{(2)} \rightarrow ... \rightarrow {}^{W}\mathbf{T}_{task}^{(i)} \rightarrow ... \rightarrow {}^{W}\mathbf{T}_{task}^{(n)}; \quad i = 1,2,...n \tag{18}$$

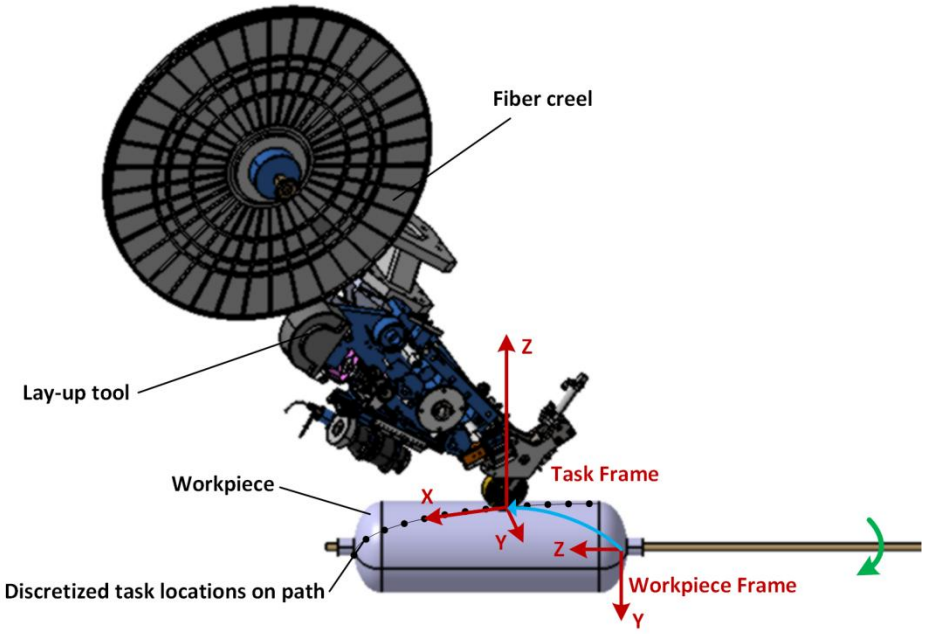


Figure 20 Definition of task frames with respect to workpiece frame

2.1.2 Conversion of the Lay-up Task to the Sequence of 6×1 vectors

From the Equation (18), the task locations (also known as the tool locations) are described by the way of 4×4 homogenous transformation matrices. However, it should be mentioned that, to control the industrial manipulator in practice, an equivalent expression of the end-effector location is more frequently used, i.e. 6×1 vector $(x,y,z,\alpha,\beta,\gamma)^T$. These vectors include three position coordinates and three orientation angles. It can be directly computed from the known homogeneous transformation matrices $(\hat{\mathbf{n}}_i,\hat{\mathbf{s}}_i,\hat{\mathbf{a}}_i)_{3\times3}$ (Craig, 2005)

$$ {}^{W}\mathbf{T}_{task}^{(i)} = \left(\begin{array}{ccc|c} \hat{\mathbf{n}}_i & \hat{\mathbf{s}}_i & \hat{\mathbf{a}}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{array}\right)_{4\times4} = \left(\begin{array}{c|c} \mathbf{R}(\boldsymbol{\varphi}_i) & \mathbf{p}_i \\ \mathbf{0} & 1 \end{array}\right)_{4\times4} ; \quad i=1,2,\ldots n \tag{19} $$

where $\boldsymbol{\varphi}_i$ is the vector of orientation angles $(\alpha,\beta,\gamma)^T$ and $\mathbf{R}(\cdot)$ is the 3×3 orthogonal matrix which defines the rotation from F_W to $F_{task}^{(i)}$. There are many ways to define the orientation angles. In this work, we use the definition in most of the commercial industrial robot, where the angles are defined as Z-Y-X Euler angles.

It has to be stressed that the task is presented uniquely by the sequences of frames and 4×4 matrices, whereas the corresponding 6×1 vectors might be non-unique. To compute the angle values from the matrices, the scalar equations can be extracted. For example, the angle β_i can be obtained from the following expressions

$$ {}^{w}\mathbf{T}_{task}^{(i)} = \mathbf{r}_{4\times4} = \left(\begin{array}{ccc|c} \cos\alpha_i\cos\beta_i & \cos\alpha_i\sin\beta_i\sin\gamma_i-\sin\alpha_i\cos\gamma_i & \cos\alpha_i\sin\beta_i\cos\gamma_i+\sin\alpha_i\sin\gamma_i & x_i \\ \sin\alpha_i\sin\beta_i & \sin\alpha_i\sin\beta_i\sin\gamma_i+\cos\alpha_i\cos\gamma_i & \sin\alpha_i\sin\beta_i\cos\gamma_i-\cos\alpha_i\sin\gamma_i & y_i \\ -\sin\beta_i & \cos\beta_i\sin\gamma_i & \cos\beta_i\cos\gamma_i & z_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{20} $$

and

$$ \beta_i = \operatorname{atan2}(-r_{3,1}, \pm\sqrt{r_{1,1}^2+r_{2,1}^2}) \tag{21} $$

In industrial robotics, the positive solution is usually used here, for which $\cos\beta_i \geq 0$. And then, α_i and γ_i can be easily obtained if $\cos\beta_i > 0$ (Craig, 2005). In the case of $\cos\beta_i = 0$, known as the singular cases, it leads to infinite number of solutions. However, in practice, two possible conventions are usually recommended to ensure unique solution (Nof, 1999, Niku, 2001, Groover, 2007).

- Utilizing previously-used α_i and γ_i;
- Using the present $\alpha_i \pm \gamma_i$ and the previous $\alpha_{i-1} \pm \gamma_{i-1}$. For example,

$$ \alpha_i := \alpha_{i-1} + \frac{(\alpha_i+\gamma_i)-(\alpha_{i-1}+\gamma_{i-1})}{2}; \quad \gamma_i := \gamma_{i-1} + \frac{(\alpha_i+\gamma_i)-(\alpha_{i-1}+\gamma_{i-1})}{2} \tag{22} $$

By this way, an equivalent expression of Equation (18) is obtained as a unique sequence of 6×1 vectors

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \alpha_1 \\ \beta_1 \\ \gamma_1 \end{bmatrix} \rightarrow \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ \alpha_2 \\ \beta_2 \\ \gamma_2 \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} x_i \\ y_i \\ z_i \\ \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} x_n \\ y_n \\ z_n \\ \alpha_n \\ \beta_n \\ \gamma_n \end{bmatrix}; \quad i = 1,2,\ldots n \tag{23}$$

Based on this one-to-one transformation between two representations, both the joint trajectory and Cartesian trajectory can be obtained during the motion generation for automated lay-up application.

2.2 ROBOTIC LAY-UP SYSTEM COMPONENTS AND THEIR MODELS

The considered robotic system includes two principle mechanisms: an industrial robot (a technological tool manipulator) and a positioner (a workpiece manipulator). It is assumed that their kinematics is known and spatial locations of the output frames depend on the vectors of actuated joint coordinates. Let us derive the kinematic models of these two mechanisms and then aggregate them with the model of the technological task.

2.2.1 Kinematic Model of Robotic Manipulator

The kinematic model of robotic manipulator defines relations between the actuated variables (joint coordinates) and spatial location of the technological tool (its position and orientation). In the frame of this work, the links of the robot are modeled as rigid bodies and the joints are assumed to provide pure rotation or translation.

In order to derive the kinematic model of an industrial robot, a universal DH-technique proposed by Denavit-Hartenberg (Hartenberg and Denavit, 1955) can be applied. However, for industrial robots with standard architectures considered in this work, a more convenient description can be used in order to simplify equations and reduce the computing time required for the optimal motion generation algorithm. To obtain the desired model, let us define a number of frames attached to each link as shown in Figure 21:

F_{Rbase} *Robot base frame*. This frame is attached to the robot base and does not move. It can be considered the reference frame where the positions of all other link frames are described. In F_{Rbase}, X-Y plane describes the floor and Z-direction is chosen along axis #1.

F_1 *Frame* 1 is attached to the first link. The distance between origins of F_1 and F_{BASE} is L_0 in Z-direction. The X-direction is pointed along the link L_1, and it has the same direction of X_0 when $q_{R1} = 0$.

F_2 *Frame* 2 is attached to the second link. The origin locates on the rotational axis of q_{R2}; the X-direction coincides with link L_2; and the Y-direction is the same to that in F_1.

F_3 *Frame* 3 is attached to the third link. The origin locates on the rotational axis of q_{R3}; the X-direction is parallel to link L_3; and the Y-direction is the same to that in F_2.

F_4 *Frame* 4 origin is located at the wrist center since the axes 4, 5, 6 all intersect at the wrist center point and are mutually orthogonal. X-direction is parallel to link L_3, and Y as well as Z have the same directions to that in F_3 when $q_{R4}=0$.

F_5 *Frame* 5 origin is also located at the wrist center. Its Y-direction is along the rotational axis 5. When $q_{R5}=0$, it completely coincides to F_4.

F_6 *Frame* 6 is assigned to the wrist center point too. The X-direction is along the rotational axis 6. When $q_{R6}=0$, it completely coincides to F_5.

F_{Rtool} *Robot tool frame* has its origin at the tip of the tool (tool center point). The orientation is selected in such a way that its Z axis is identical to the tool axis direction and points out of the tool; its Y axis is parallel to Y_5 with the same direction. If the tool center point is moved, the tool frame is moved with it.

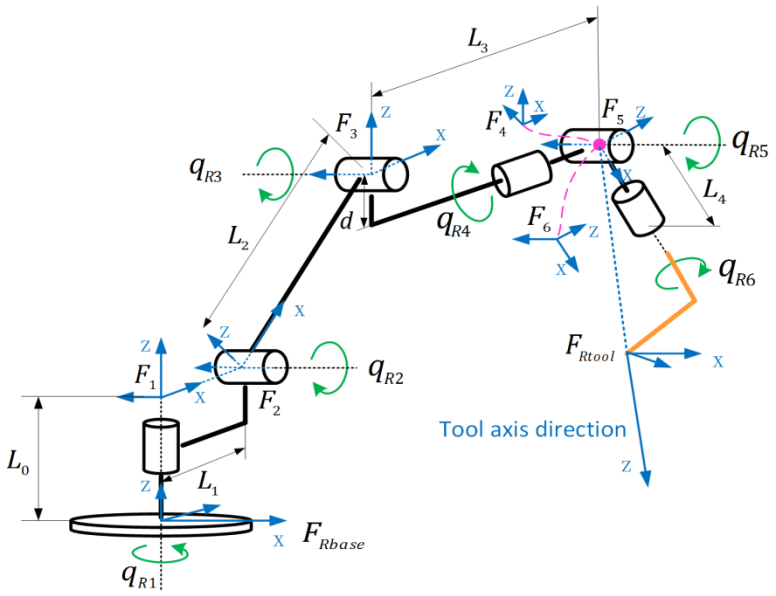


Figure 21 Architecture of standard 6-axis industrial robot

As follows from the above figure, the robot kinematic model should include the following parameters

$$\{L_0, L_1, L_2, L_3, L_4, d\} \tag{24}$$

where L_0 is the vertical distance between the origins of F_{Rbase} and F_1, as well as F_2; L_1 is the distance between q_{R1} axis and q_{R2} axis; L_2 it the length of the second link, i.e. the distance between F_2 origin and F_3 origin; d is an offset between the third link and q_{R3} axis; L_3 is the length of link 3; L_4 is the distance between the wrist center point and the mounting flange surface. This parameter is generally considered in the tool model.

In robot modeling, there are two basic types of models that are usually referred to as *direct* and *inverse* ones. The robot direct kinematic model describes the spatial location of mounting flange frame

F_6 with respect to F_{Rbase} as a function of the actuated joint coordinates. For the serial architecture presented above, the robot direct kinematic model can be expressed as a product of the 4×4 homogenous transformation matrices (Craig, 2005)

$$g_R(\mathbf{q}_R) = {}^{Rbase}\mathbf{T}_1(q_{R1})\cdot{}^{1}\mathbf{T}_2(q_{R2})\cdot{}^{2}\mathbf{T}_3(q_{R3})\cdot{}^{3}\mathbf{T}_4(q_{R4})\cdot{}^{4}\mathbf{T}_5(q_{R5})\cdot{}^{5}\mathbf{T}_6(q_{R6})\cdot{}^{6}\mathbf{T}_{Rtool} \tag{25}$$

that depends on the joint variables $q_{R1}, q_{R2}, \ldots, q_{R6}$. It should be mentioned that it is assumed here that the robot base frame is located at the intersection of the axis Z_1 and the floor level, but it can be easily relocated in practice using the relevant commands of robot programming languages (e.g. $ROBROOT in KRL language).

Besides, it is also assumed that the last frame is located to the TCP point (so called "tool center point") and orientation of the tool axes $X_{tool}, Y_{tool}, Z_{tool}$ is determined with the accordance of the technological requirements. This definition corresponds to general type of the 4x4 homogeneous matrix

$${}^{6}\mathbf{T}_{Rtool} = \left(\begin{array}{ccc|c} \cos A_t \cos B_t & \cos A_t \sin B_t \sin C_t - \sin A_t \cos C_t & \cos A_t \sin B_t \cos C_t + \sin A_t \sin C_t & X_t \\ \sin A_t \sin B_t & \sin A_t \sin B_t \sin C_t + \cos A_t \cos C_t & \sin A_t \sin B_t \cos C_t - \cos A_t \sin C_t & Y_t \\ -\sin B_t & \cos B_t \sin C_t & \cos B_t \cos C_t & Z_t \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{26}$$

that depends on six parameters $(X_t, Y_t, Z_t, A_t, B_t, C_t)$ describing position and orientation of F_{Rtool} with respect to F_6 attached to the mounting flange. It is worth mentioning that the parameter L_4 of the robot (see Figure 21) is usually included in the tool parameterization in order to avoid useless computations. In practice, these six parameters can be easily modified using dedicated commands of robot programming languages (e.g. $TOOL in KRL language).

For the remaining matrices, relevant expressions can be obtained in a conventional way, using the Denavit-Hartenberg technique where they are presented as a composition of elementary rotations and translations ${}^{i-1}\mathbf{T}_i(q_{Ri}) = \mathbf{R}_X(\alpha_{i-1})\cdot\mathbf{D}_X(a_{i-1})\cdot\mathbf{R}_Z(q_{Ri})\cdot\mathbf{D}_Z(d_i)$ depending on both the joint variables q_{Ri} and the parameters a_i, d_i, α_i describing the links/joints geometry. However, in this work, with the slight different definition of the frames, it is possible to achieve some simplification and to avoid unnecessary matrix multiplications. In particular, for the first joint that provides rotation about the vertical axis Z_1, the matrix ${}^{Rbase}\mathbf{T}_1(q_{R1})$ can be expressed as

$${}^{Rbase}\mathbf{T}_1 = \left(\begin{array}{ccc|c} \cos q_{R1} & -\sin q_{R1} & 0 & 0 \\ \sin q_{R1} & \cos q_{R1} & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

where the parameter L_0 takes into account the shift between the robot base frame F_{Rbase} and the frame of the first link F_1. Similarly, the second and the third joints can be described using the rotations around the horizontal axes Y_2 and Y_3, which yields the following expressions for the matrices ${}^{1}\mathbf{T}_2(q_{R2})$ and ${}^{2}\mathbf{T}_3(q_{R3})$

$$ {}^{1}\mathbf{T}_{2}=\left(\begin{array}{ccc|c}\cos q_{R2} & 0 & \sin q_{R2} & L_1\\ 0 & 1 & 0 & 0\\ -\sin q_{R2} & 0 & \cos q_{R2} & 0\\ \hline 0 & 0 & 0 & 1\end{array}\right) \qquad {}^{2}\mathbf{T}_{3}=\left(\begin{array}{ccc|c}\cos q_{R3} & 0 & \sin q_{R3} & L_2\\ 0 & 1 & 0 & 0\\ -\sin q_{R3} & 0 & \cos q_{R3} & 0\\ \hline 0 & 0 & 0 & 1\end{array}\right) $$

where the parameters L_1 and L_2 define the lengths of the manipulator links 1 and 2 along the axes X_1 and X_2. It is worth mentioning that in general case, the link parameters must include additional parameter describing the length in the direction Z_1 but it is reasonable to include it in the above presented parameter L_0 (it allows us to simplify the model and reduce unnecessary computations while keeping the model correctness).

For the remaining joint angles providing the tool orientation, the desired matrices can be derived in similar way. In particular, the matrix ${}^{3}\mathbf{T}_{4}(q_{R4})$ can be expressed using rotation about the axis X_4

$$ {}^{3}\mathbf{T}_{4}=\left(\begin{array}{ccc|c}1 & 0 & 0 & L_3\\ 0 & \cos q_{R4} & -\sin q_{R4} & 0\\ 0 & \sin q_{R4} & \cos q_{R4} & d\\ \hline 0 & 0 & 0 & 1\end{array}\right) $$

Also it includes translational parameters L_3 and d describing geometry of the link 3. For the remaining axis the desired matrices can be obtained via elementary rotations about the axes Y_5 and X_6:

$$ {}^{4}\mathbf{T}_{5}=\left(\begin{array}{ccc|c}\cos q_{R5} & 0 & \sin q_{R5} & 0\\ 0 & 1 & 0 & 0\\ -\sin q_{R5} & 0 & \cos q_{R5} & 0\\ \hline 0 & 0 & 0 & 1\end{array}\right) \qquad {}^{5}\mathbf{T}_{6}=\left(\begin{array}{ccc|c}1 & 0 & 0 & 0\\ 0 & \cos q_{R6} & -\sin q_{R6} & 0\\ 0 & \sin q_{R6} & \cos q_{R6} & 0\\ \hline 0 & 0 & 0 & 1\end{array}\right) $$

It is clear that the above expressions do not include translational parameters, because the origins of the frames 4, 5, 6 are located at the same points (so-called "robot wrist center"). It should be mentioned that the parameter L_4 defining the distance between the mounting flange and the wrist center is usually included in the matrix describing the tool geometry, i.e. the (1,4) element of ${}^{6}\mathbf{T}_{Rtool}$ is assumed to be equal to $X_t + L_4$. The latter also allows us to reduce computational efforts.

For further convenience, the matrix computations included in the direct kinematic model (24) were executed analytically which allowed us to present the product ${}^{Rbase}\mathbf{T}_{1}(q_{R1})\cdot{}^{1}\mathbf{T}_{2}(q_{R2})\cdots{}^{5}\mathbf{T}_{6}(q_{R6})$ in the following way

$$ {}^{Rbase}\mathbf{T}_{6}(\mathbf{q}_R)=\left(\begin{array}{ccc|c}\mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p}\\ \hline 0 & 0 & 0 & 1\end{array}\right) \tag{27} $$

where $\mathbf{q}_R=(q_{R1},q_{R2},q_{R3},q_{R4},q_{R5},q_{R6})^T$ is the vector of the robot joint variables and

$$ \mathbf{n}=\begin{pmatrix}C_5C_1C_{23}-S_5(S_1S_4+C_1C_4S_{23})\\ C_5S_1C_{23}+S_5(C_1S_4+S_1C_4S_{23})\\ -C_5S_{23}-C_4S_5C_{23}\end{pmatrix} \qquad \mathbf{s}=\begin{pmatrix}S_6(C_5(S_1S_4+C_1C_4S_{23})+S_5C_1C_{23})+C_6(S_1S_4-C_1S_4S_{23})\\ -S_6(C_5(C_1S_4-C_1S_4S_{23})+S_5C_1C_{23})+C_6(C_1C_4-S_1S_4S_{23})\\ C_6S_4C_{23}-S_6(S_5S_{23}-C_4C_5C_{23})\end{pmatrix} $$

$$\mathbf{a}=\begin{pmatrix} S_6(C_4S_1-C_4S_1S_{23})+C_6(C_5(S_4S_1+C_4S_1S_{23})+S_5C_1C_{23}) \\ -S_6(C_4C_1+S_4S_1S_{23})-C_6(C_5(S_4C_1-C_4S_1S_{23})-S_5S_1C_{23}) \\ -C_6(S_{23}S_5-C_{23}C_4C_5)-S_6S_4C_{23} \end{pmatrix} \quad \mathbf{p}=\begin{pmatrix} C_1(L_1+L_3\cdot C_{23}+d\cdot S_{23}+L_2\cdot C_2) \\ S_1(L_1+L_3\cdot C_{23}+d\cdot S_{23}+L_2\cdot C_2) \\ L_0-L_3\cdot S_{23}+d\cdot C_{23}-L_2\cdot S_2 \end{pmatrix}.$$

Here, usual robotic notations are used allowing achieving compact presentation: $C_i=\cos q_{Ri}; S_i=\sin q_{Ri}$ and $C_{23}=\cos(q_{R2}+q_{R3}); S_{23}=\sin(q_{R2}+q_{R3})$.

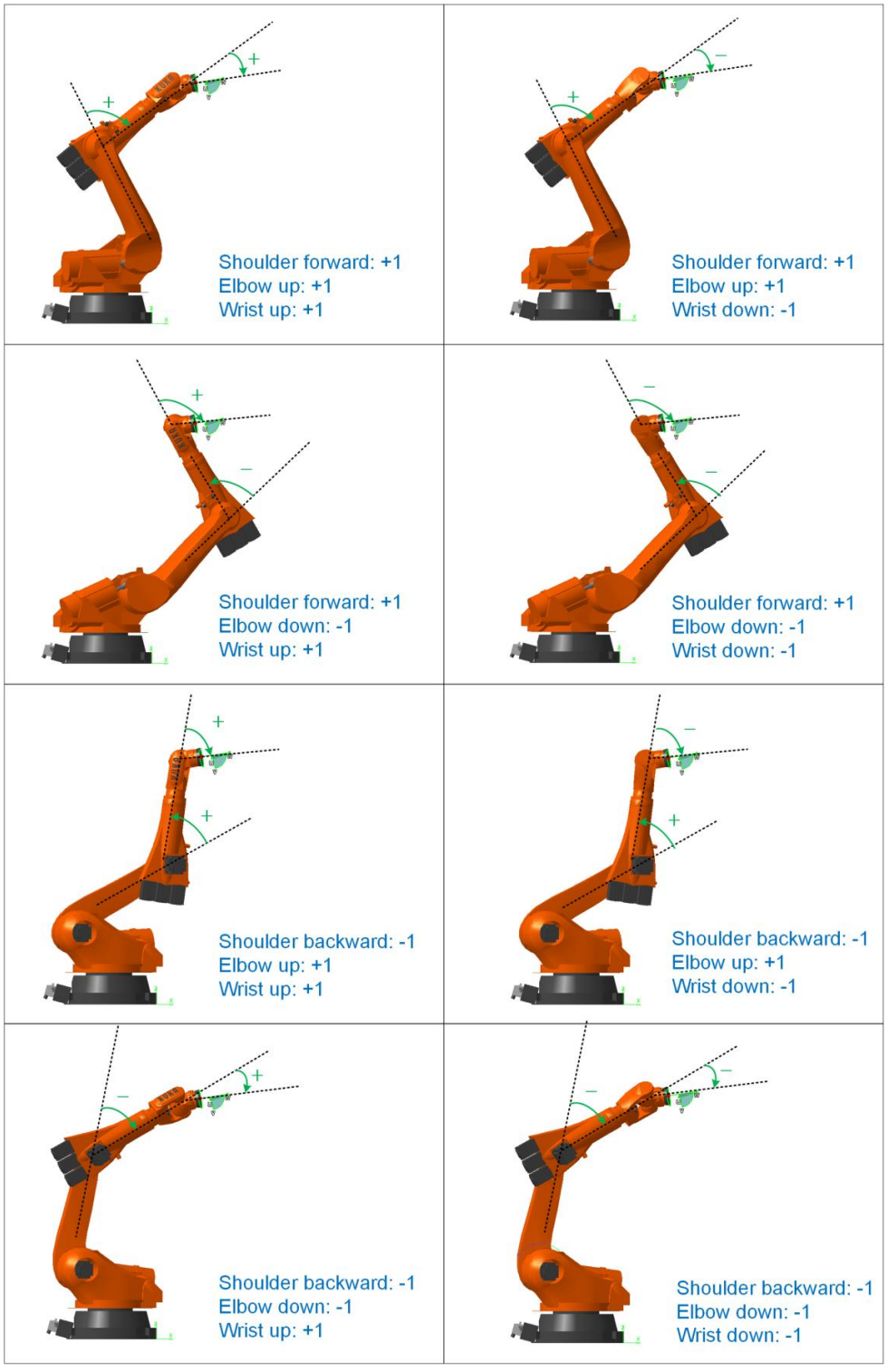


Figure 22 Different configuration of robotic manipulator for the same location of the end-effector

To generate desired robot motions, the developed algorithm (see Chapter 3) extensively utilizes the inverse kinematic transformations allowing us to compute the vector of the actuated joint variables $\mathbf{q}_R$ corresponding to the desired robot flange location described by a given homogeneous 4×4 matrix ${}^{Rbase}\mathbf{T}_6$. In robotics literature, this transformation is usually called the robot inverse kinematic model, and it will be denoted below as $g_R^{-1}(.)$. Compared to the direct transformation, the inverse one is not trivial since the solution is usually not unique and can be expressed in a closed form in some cases only. For this reason, special types of manipulator architecture are used in industry to satisfy the Pieper condition (Peiper, 1968) that ensures the closed-form solution. Otherwise, there are also some numerical techniques to deal with this problem (Pashkevich, 1997, Husty et al., 2007, Manocha and Canny, 1994) that may produce up to 16 different solutions for the inverse kinematics. In this work, the serial robot with last three intersecting axes is employed, which corresponds to the majority of industrial applications. For this architecture, for most of the allowable robot end-effector locations, there are 8 possible solutions as is shown in Figure 22. In industrial robot programming, these solutions are distinguished using so-called "*configuration index*". From geometric point of view, the manipulator configurations corresponding to the same tool location differ by the shoulder posture (forward/backward), the elbow posture (up/down) and the wrist posture (up/down). Algebraically, these configurations will be described by the binary vector $\boldsymbol{\mu}=(\mu_1,\mu_2,\mu_3)$ with the components ± 1 that will be used below for selection of particular solutions of relevant trigonometrical equations (see Table 10). To take into account the multiplicity of the inverse kinematic solutions, the corresponding inverse function will be denoted below as

$$\mathbf{q}_R = g_R^{-1}({}^{Rbase}\mathbf{T}_{Rtool}, \boldsymbol{\mu}) \tag{28}$$

where the 4×4 homogeneous matrix ${}^{Rbase}\mathbf{T}_{Rtool}$ defines the desired end-effector location; the configuration vector $\boldsymbol{\mu}=(\pm 1,\pm 1,\pm 1)$ describes the desired manipulator posture and $\mathbf{q}_R$ is the vector of the actuated joint coordinates. It is worth of mentioning that the vector $\boldsymbol{\mu}$ is an output of the robot direct kinematic model, which is determined by the coordinates of q_{R1}, q_{R3}, q_{R5} with the rules derived below.

Table 10 Definition of the configuration index for serial 6-axis manipulator.

Value	μ_1	μ_2	μ_3
1	The x-value of the intersection of the wrist axes, relative to F_1, is positive. (shoulder forward)	$q_3 \geq \phi$ (elbow up)	$q_5 \geq 0$ (wrist up)
-1	The x-value of the intersection of the wrist axes, relative to F_1, is negative. (shoulder backward)	$q_3 < \phi$ (elbow down)	$q_5 < 0$ (wrist down)
ϕ depends on the size of the offset between axis 3 and axis 4; $\phi = 0$ when $d = 0$			

To obtain the inverse kinematic solution for the considered manipulator (satisfying the Pieper condition), two classic approaches can be applied: *algebraic* and *geometric*. The first of them is based on decomposing the original six-dimensional task into several relatively simple plane geometry problems that may be easily treated analytically (Craig, 2005). The second approach is based on solving a set of trigonometric equations derived directly from Equation (25) describing the direct kinematics (Craig, 2005). There are some special techniques here allowing separating the unknown

variables (left- and right-hand side matrix multiplications, etc.). The most efficient of them is based on the matrix expression

$$^{Rbase}\mathbf{T}_6(\mathbf{q}_R) = {}^{Rbase}\mathbf{T}_{Rtool} \cdot ({}^{6}\mathbf{T}_{Rtool})^{-1} \tag{29}$$

which is equivalent to relocating the target frame $^{Rbase}\mathbf{T}_{Rtool}$ to the robot wrist center, where the frames F_4, F_5, F_6 are located. The latter allows us to separate the variables q_{R1}, q_{R2}, q_{R3} and q_{R4}, q_{R5}, q_{R6}. By this way, to simplify the computation in this work, a method combining the geometric approach and algebraic approach together is presented as follows.

Computing the angle for axis #1. The joint variable q_{R1} can be easily computed from algebraic equations straightforwardly derived from last column of the matrix expression (29)

$$\begin{aligned} p_x &= C_1(L_1 + L_3 \cdot C_{23} + d \cdot S_{23} + L_2 \cdot C_2) \\ p_y &= S_1(L_1 + L_3 \cdot C_{23} + d \cdot S_{23} + L_2 \cdot C_2) \end{aligned} \tag{30}$$

where p_x and p_y are the Cartesian coordinates of the wrist center point relative to the robot base frame F_{Rbase} that can be directly extracted from the last column of the matrix $^{Rbase}\mathbf{T}_6$, i.e. $p_x = \left[{}^{Rbase}\mathbf{T}_6\right]_{14}$ and $p_y = \left[{}^{Rbase}\mathbf{T}_6\right]_{24}$. The above system of trigonometric equations can be transformed into

$$C_1 = p_x/\rho; \quad S_1 = p_y/\rho \tag{31}$$

where $\rho = (L_1 + L_3 \cdot C_{23} + d \cdot S_{23} + L_2 \cdot C_2)$. This allows us to present the desired expression for q_{R1} in the form

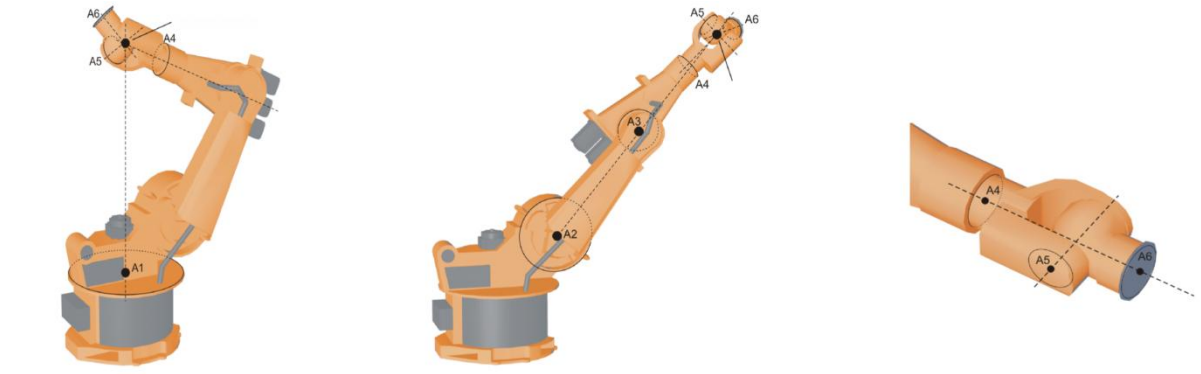
$$q_{R1} = \text{atan2}(S_1, C_1) = \text{atan2}(\mu_1 \cdot p_y, \mu_1 \cdot p_x) \tag{32}$$

where

$$\mu_1 = sign(L_1 + L_3 \cdot C_{23} + d \cdot S_{23} + L_2 \cdot C_2) \tag{33}$$

is the configuration index defining the manipulator posture with respect to the axis #1 (shoulder forward/backward). It is clear that μ_1 must be an input variable for the inverse kinematics and an output of the direct kinematics.

It should be mentioned that the above equations cannot be applied directly if $\rho = 0$, i.e. when the wrist center point is located on axis#1. In this case, the angle q_{R1} cannot be determined unambiguously and thus can take any value. In practice, it is named as "shoulder position singularity" (see Figure 23a). To avoid chaotic manipulator motions in the neighborhood of this singularity, industrial robot controllers may assign to q_{R1} either default or previous value. For example, in KRL language of KUKA, user can chose one of these two options by setting a special system variable: $ SINGUL_POS[1] that is equal to 0 for the default setting and is equal to 1 for the previous value.



(a) Shoulder position singularity (b) Elbow position singularity (c) Wrist position singularity

Figure 23 Three types of singular configurations in KUKA kinematic system, ©KUKA (KUKA, 2010)

Computing angles for axes #2 and #3. After obtaining the actuated angle for the axis #1, it is possible to solve equations for the wrist center point coordinates

$$\begin{aligned} p_x &= C_1(L_1 + L_3 \cdot C_{23} + d \cdot S_{23} + L_2 \cdot C_2) \\ p_y &= S_1(L_1 + L_3 \cdot C_{23} + d \cdot S_{23} + L_2 \cdot C_2) \\ p_z &= L_0 - L_3 \cdot S_{23} + d \cdot C_{23} - L_2 \cdot S_2 \end{aligned} \tag{34}$$

with respect to q_{R2} and q_{R3} assuming that C_1 and S_1 are already known. However, to simplify the final expressions, it is more convenient to apply here the geometric approach.

In order to find the desired angles, let us consider the manipulator projection to the vertical plane defined by the axis Z_1 and wrist center point as shown in Figure 24. For this projection, let us consider the first triangle with the edges L_2, $\sqrt{d^2 + L_3^2}$ and $\sqrt{(p_z - L_0)^2 + (\rho - L_1)^2}$ that allows us to compute the axis #3 angle (this triangle is highlighted in pink in Figure 24). Applying the law of cosines, one can get the following expression for the adjacent angle α at the vertex connecting the edges L_2 and $\sqrt{d^2 + L_3^2}$

$$\alpha = \arccos\left(\frac{(p_z - L_0)^2 + (\rho - L_1)^2 - (L_3^2 + d^2 + L_2^2)}{2L_2\sqrt{L_3^2 + d^2}}\right) \tag{35}$$

As follows from the figure, the desired angle q_{R3} can be easily obtained from the computed value α and the constant angle $\beta = \arctan(d/L_3)$ describing geometry of the third link, i.e. $q_{R3} = (\pi - \alpha) - |\beta|$ that yields the expression

$$q_{R3} = \arccos\left(\frac{(L_3^2 + d^2 + L_2^2) - (p_z - L_0)^2 - (\rho - L_1)^2}{2L_2\sqrt{L_3^2 + d^2}}\right) + \arctan\left(\frac{d}{L_3}\right) \tag{36}$$

It should be mentioned that here the value of d is negative, but it may be positive for some industrial robots. In case of $d > 0$, q_{R3} should be rewritten as $q_{R3} = (\pi - \alpha) + |\beta|$. Besides, compared to the standard DH convention, the positive directions of q_{R3} is opposite compared to the usual one, which leads to slightly different inverse kinematic expressions (this definition of q_{R3} corresponds to KUKA robots used for the implementation of the algorithms developed in this work).

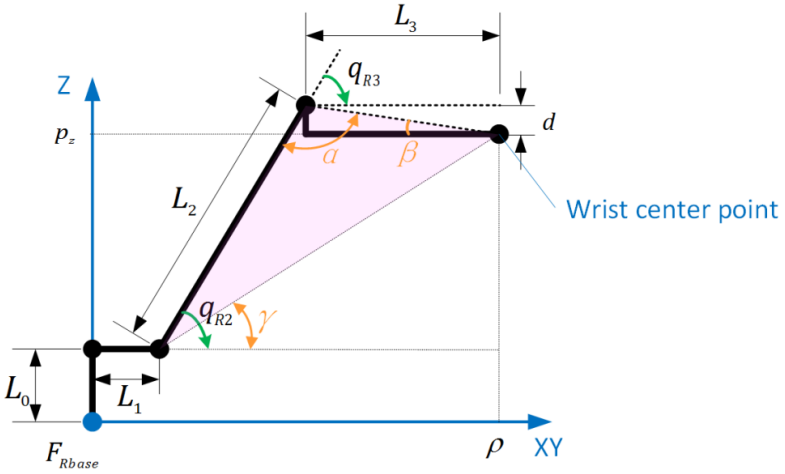


Figure 24 Projection of the manipulator links on the vertical plane (Configuration SHOULDER_FORWARD & ELBOW_UP)

It is also worth mentioning that the above expression for q_{R3} is valid for a particular manipulator configuration that usually is denoted as "ELBOW_UP". To take into account multiplicity of the configurations corresponding to the same location of the wrist center point (see Figure 29), the final expression for angle q_{R3} can be presented as

$$q_{R3} = \mu_2 \cdot \arccos\left(\frac{(L_3^2 + d^2 + L_2^2) - (p_z - L_0)^2 - (\rho - L_1)^2}{2L_2\sqrt{L_3^2 + d^2}}\right) + \arctan\left(\frac{d}{L_3}\right) \tag{37}$$

where $\mu_2 = \pm 1$ is the second configuration index that defines the manipulator elbow configuration (ELBOW UP/DOWN). Similar to μ_1, the configuration index μ_2 must be computed in the direct kinematics. Analyzing Figure 25, it is easy to prove that the second configuration index is expressed as

$$\mu_2 = sign(q_{R3} - \beta) \tag{38}$$

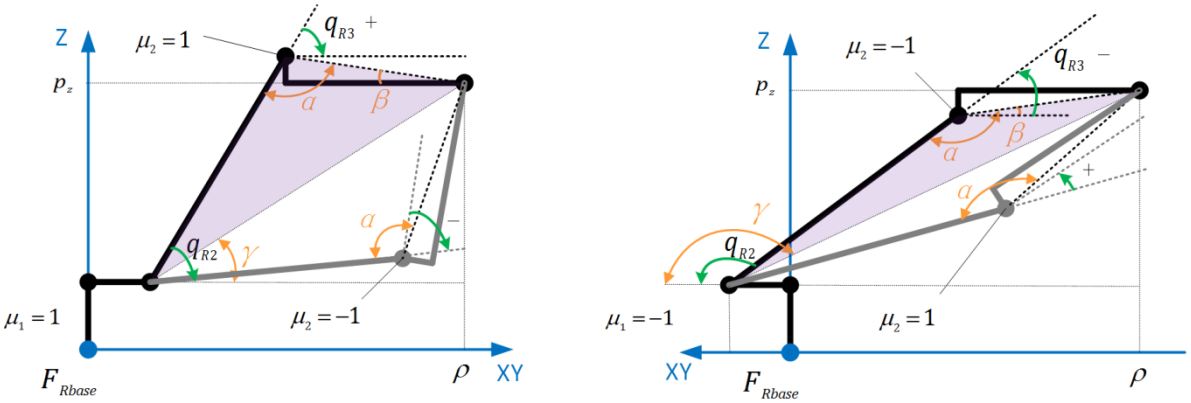


Figure 25 Multiple manipulator configurations for the same wrist center point position

To find the remaining angle q_{R2}, let us consider the second triangle that is highlighted in green in Figure 26, which allows us to compute the auxiliary angle γ. As follows from the figure, it can be computed from the expression

$$\gamma = \text{atan2}(p_z - L_0, \rho - L_1) \tag{39}$$

that may provide either positive or negative value. Besides, considering the third triangle (highlighted in blue in Figure 26) and assuming that q_{R3} is already known, one can compute the auxiliary angle θ using the expression

$$\theta = \text{atan2}\left(\sqrt{d^2+L_3^2}\cdot\sin(q_{R3}-\beta),\, L_2+\sqrt{d^2+L_3^2}\cdot\cos(q_{R3}-\beta)\right) \tag{40}$$

Then, using the angles γ and θ, the desired actuated angle can be expressed as $\theta+\gamma$. However, it is worth mentioning that here the positive direction of q_{R2} is opposite to the usual one, which leads to slightly different inverse kinematic expressions (this definition of q_{R2} corresponds to KUKA robots). Thus, $q_{2R}=-(\theta+\gamma)$ here and the final expression for q_{R2} is

$$q_{2R} = -\text{atan2}\left(\sqrt{d^2+L_3^2}\cdot S_{3\beta},\, L_2+\sqrt{d^2+L_3^2}\cdot C_{3\beta}\right) - \text{atan2}(p_z-L_0,\, \rho-L_1) \tag{41}$$

where $S_{3\beta}=\sin(q_{R3}-\beta)$ and $C_{3\beta}=\cos(q_{R3}-\beta)$.

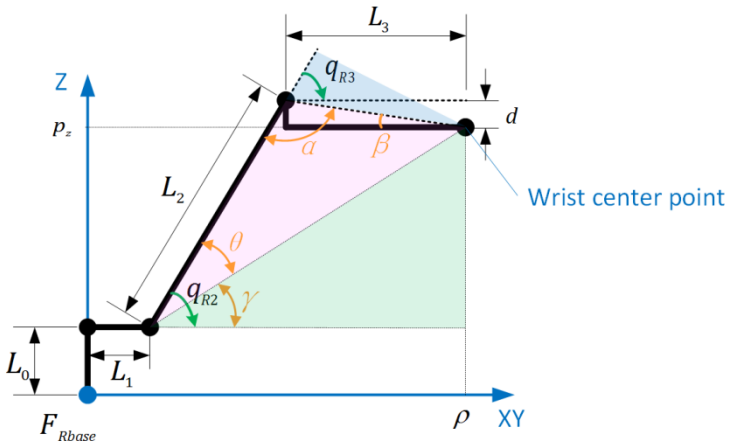


Figure 26 Triangles highlighted for computing the angle of axis #2

It should be mentioned that the above equations can be applied only on the condition that the inverse function $\arccos(.)$ in equation (37) exists and provides real value of α, i.e. if

$$\left|\frac{(p_z-L_0)^2+(\rho-L_1)^2-(L_3^2+d^2+L_2^2)}{2L_2\sqrt{L_3^2+d^2}}\right| \le 1 \tag{42}$$

Otherwise, the inverse kinematic transformation should provide an error message "*Too Far*" or "*Too Close*", because the desired wrist center point is out of the manipulator working area. It is also worth mentioning that if the above expression is equal to ± 1, the robot is at the border of its workspace (see Figure 23b). In this case, the wrist center point is located on the line connecting the centers of the axis #2 and axis #3, so independence of the configuration index μ_2, the θ angle is equal to either $\text{atan2}(d,L_3)$ or $\text{atan2}(d,L_3)\pm\pi$. In industrial robotics, the above mentioned phenomenon is categorized as "elbow singularity" and expected to be avoided by assigning the target points inside of the manipulator workspace (so-called reachable locations).

Computing angles for axes #4, #5 and #6. After computing the actuated angles for the axes #1, #2 and #3, it is possible to solve equations for the remaining angles q_{R4}, q_{R5} and q_{R6}. To derive relevant equations, the direct kinematic model (25) can be rewritten as

$$\left({}^{Rbase}\mathbf{T}_1(q_{R1})\cdot{}^{1}\mathbf{T}_2(q_{R2})\cdot{}^{2}\mathbf{T}_3(q_{R3})\right)^{-1}\cdot{}^{Rbase}\mathbf{T}_{Rtool}\cdot\left({}^{6}\mathbf{T}_{Rtool}\right)^{-1}={}^{3}\mathbf{T}_4(q_{R4})\cdot{}^{4}\mathbf{T}_5(q_{R5})\cdot{}^{5}\mathbf{T}_6(q_{R6}) \tag{43}$$

where the left hand-side is already known and the right hand-side contains the unknowns to be found. This equation can be also expressed in the scalar form using the following notations

$$\left({}^{Rbase}\mathbf{T}_1(q_{R1})\cdot{}^{1}\mathbf{T}_2(q_{R2})\cdot{}^{2}\mathbf{T}_3(q_{R3})\right)^{-1}\cdot{}^{Rbase}\mathbf{T}_{Rtool}\cdot\left({}^{6}\mathbf{T}_{Rtool}\right)^{-1} = \left(\begin{array}{cccc} n_x & s_x & a_x & 0 \\ n_y & s_y & a_y & 0 \\ n_z & s_z & a_z & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{44}$$

and

$${}^{3}\mathbf{T}_6(q_{R4},q_{R5},q_{R6}) = \left(\begin{array}{ccc|c} C_5 & S_5S_6 & S_5C_6 & 0 \\ S_4S_5 & C_4C_6-S_4C_5S_6 & -C_4S_6-S_4C_5C_6 & 0 \\ -C_4S_5 & S_4C_6+C_4C_5S_6 & -S_4S_6+C_4C_5C_6 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{45}$$

For the angle q_{R5}, let us consider three scalar equations provided by the first columns of the above matrices

$$n_x = C_5;\quad n_y = S_4S_5;\quad n_z = -C_4S_5 \tag{46}$$

which allow us to find explicitly cosine and sine of q_{R5}

$$\cos q_{R5} = n_x;\quad \sin q_{R5} = \pm\sqrt{n_y^2+n_z^2} \tag{47}$$

and easily compute the desired angle using the function $\mathrm{atan2}(.)$. However, it is necessary to take into account that two symmetric solutions are possible here, which geometrically correspond to different manipulator wrist configuration (WRIST UP/DOWN) and may be described algebraically by the third configuration index

$$\mu_3 = sign(q_{R5}) \tag{48}$$

Using this notation, the expression for the angle q_{R5} can be presented as

$$q_{R5} = \mu_3\cdot\mathrm{atan2}\left(\sqrt{n_y^2+n_z^2},n_x\right) \tag{49}$$

where $\mu_3 = \pm 1$ is provided either by the user or by the direct kinematics. It is worth mentioning that there is no computational problems (singularities) related to the angle q_{R5} because the situation $\mathrm{atan2}(0,0)$ is not possible due to the identity equation $n_x^2+n_y^2+n_z^2 \equiv 1$.

For the angles q_{R4} and q_{R6}, it is easy to write similar scalar equations provided by the first lines and columns of the matrices (45) and (46)

$$n_y = S_4S_5;\quad n_z = -C_4S_5 \tag{50}$$

and

$$s_x = S_5 S_6; \quad a_x = S_5 C_6 \tag{51}$$

which contain already computed value S_5. Assuming that $S_5 \neq 0$, the desired angles q_{R4} and q_{R6} may be found using expressions

$$\begin{aligned} q_{R4} &= \text{atan2}\left(\mu_3 \cdot n_y, \; -\mu_3 \cdot n_z \right) \\ q_{R6} &= \text{atan2}\left(\mu_3 \cdot s_x, \quad \mu_3 \cdot a_x \right) \end{aligned} \tag{52}$$

where the sign of S_5 is taken into account via the configuration index μ_3.

However, in case of $S_5 = 0$, when the axes #4 and #6 are parallel, the above formulas degenerate leading to uncertainty $\text{atan2}(0,0)$ in expressions for q_{R4} and q_{R6}. Relevant manipulator configurations correspond to so-called the "wrist singularity" for which $q_{R5} = 0°$ or $q_{R5} = \pm\pi$ (it is clear that the second case is not possible in practice, see Figure 23c). For the feasible case when $q_{R5} = 0°$, the matrix (45) is reduced to

$${}^3\mathbf{T}_6(q_{R4}, 0, q_{R6}) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & C_4C_6 - S_4S_6 & -C_4S_6 - S_4C_6 & 0 \\ 0 & S_4C_6 + C_4S_6 & -S_4S_6 + C_4C_6 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right) \tag{53}$$

and yields the following scalar equations

$$\begin{aligned} s_y &= C_4C_6 - S_4S_6 \equiv \cos(q_{R4} + q_{R6}) \\ s_z &= S_4C_6 + C_4S_6 \equiv \sin(q_{R4} + q_{R6}) \end{aligned} \tag{54}$$

that provide infinite number of solutions for (q_{R4}, q_{R6}) satisfying the equality

$$q_{R4} + q_{R6} = \text{atan2}\,(s_z, s_y) \tag{55}$$

It should be noted that here $s_x = 0$, so the remaining components of the second column satisfy the identity equation $s_y^2 + s_z^2 \equiv 1$, which eliminates the uncertainty $\text{atan2}(0,0)$ in (56). Similarly, the sum $q_{R4} + q_{R6}$ can be computed using other pairs of the orientation matrix elements, such as (a_y, a_z), (s_y, a_y) and (s_z, a_z). In practice, the ambiguity related to the wrist singularity is solved by assigning either the default or previous value to one of the angles, q_{R4} or q_{R6}. For example, in KRL language of KUKA, user can chose one of these two options by setting a special system variable: $ SINGUL_POS[3] that is equal to 0 for the default setting and is equal to 1 for the previous value. Another solution is sharing the required rotation between axis #4 and axis #6 similar to equation (22) in the previous section.

Therefore, in the following sections the considered manipulator can be described using either its direct kinematic transformation $[{}^{Rbase}\mathbf{T}_{Rtool}, \boldsymbol{\mu}] = g_R(\mathbf{q}_R)$ or inverse kinematic transformation $[\mathbf{q}_R] = g_R^{-1}({}^{Rbase}\mathbf{T}_{Rtool}, \boldsymbol{\mu})$ where $\boldsymbol{\mu} = (\pm 1, \pm 1, \pm 1)$ the vector of the configuration indices and both of the functions $g_R(.)$ and $g_R^{-1}(.)$ are expressed in closed-form.

2.2.2 Kinematic Model of Actuated Positioner

An actuated positioner is a machine employed to adjust the workpiece posture with respect to the robot in order to ensure accessibility of the given task locations or satisfy some technological requirements. For example, in arc-welding application, the positioner allows the welding tool to be oriented almost vertically while keeping the welding seam nearly horizontally. In automated lay-up process studied in this work, the actuated positioners are also utilized to improve access to the task locations, but they provide some additional opportunities for increasing the lay-up process speed by combining motions of the robot and positioner.

Basically, an actuated positioner is a workpiece hold structure that is driven by the robot controller. The simplest architecture contains a head and a tailstock with a servo drive (so-called one-axis positioner). There are varieties of one-axis positioners that differ in sizes and geometries adopted to the work to be done (see Figure 27). Some smaller types are suitable for light objects while others are capable of handling workpiece weighing many metric tons. For example, KUKA positioner presented in Figure 31 are capable to manipulate objects weighting from 250 kg to 4 tons.

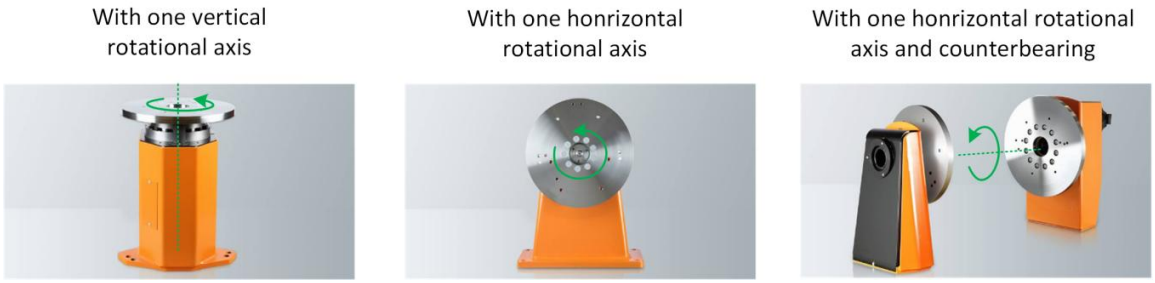


Figure 27 Typical one-axis positioners from KUKA (www.kuka.com)

The two-axis positioners are usually utilized if it is necessary to orientate the component with respect to the vertical axis. They are frequently used in arc-welding applications since one-axis positioners are not capable of providing the full weld orientation with respect to the gravity. Some examples of two-axis positoners from KUKA are presented in Figure 28.

Figure 28 Typical two-axis positioners from KUKA (www.kuka.com)

For large-scale workpieces, such as engine cowls, airplane fuselage, etc., the robotic manipulator is mounted on the linear tracks or gantries providing additional translational axis and enlarging the robot workspace. Some examples of these equipments from KUKA are presented in Figure 29. It is clear that physically these mechanical components should be included in the model of the robotic manipulator leading to increase of the actuated axis number up to seven (and obviously causing kinematic redundancy). However, in the frame of this work, the linear tracks and gantries are modeled

separately, similar to the workpiece positioners, allowing considering the robot as a non-redundant mechanism with finite number of the inverse kinematic solutions.

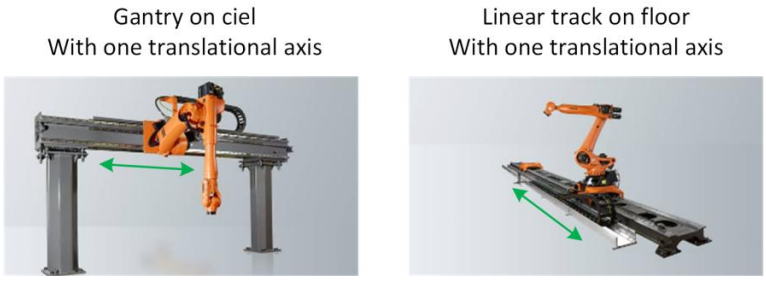


Figure 29 Typical linear track and gantry from KUKA (www.kuka.com)

For the technological process studied in this work (automated lay-up), the workpieces usually are shaped as solids of revolution. Typical examples include pressure vessels, engine cowls, airplane components, etc. For this reason, a one-axis positioner with horizontal rotation is usually sufficient for this application. Here, the positioner adjusts the orientation of the product in order to achieve the desired tool orientation and also ensures the workpiece rotation required for the lay-up. In some specific cases where the product is long or large dimensional, the robotic manipulator is mounted on the linear track/gantry while the workpiece is manipulated using a one-axis positioner. The two-axis positoners are not common for the considered technology.

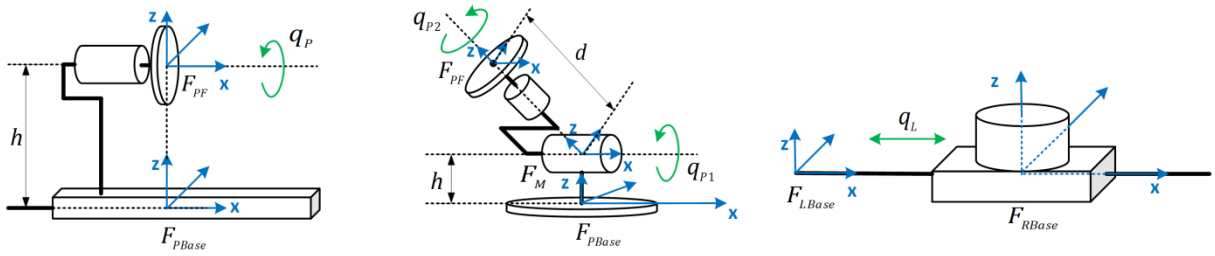
Similar to the robot model presented above, the kinematic model of the positioner defines relations between the actuated variables (joint coordinates) and spatial location of the workpiece mounted on flange (its position and orientation). In the frame of this work, the components of the positioner are modeled as rigid bodies and the joints are assumed to provide pure rotation or translation. To obtain the desired model, let us define a number of frames as shown in Figure 30:

F_{Pbase} *Positioner base frame*. This frame is attached to the positioner base and does not move. It can be considered the reference frame where the positions of all other link frames are described. In F_{Pbase}, X-Y plane describes the floor and Z-axis is vertical and outward the floor. The definitions for one-axis positioner and two-axis positioner are the same, as is shown in Figure 30(a)(b).

F_M *Positioner intermediate frame.* This frame is attached to the first rotational axis of the two-axis positioner, as is shown in Figure 30(b). The distance between origins of F_M and F_{Pbase} is h. The X-axis of this frame has the same direction to that of F_{Pbase} and their Z directions coincide when $q_{P1}=0$.

F_{PF} *Positioner flange frame*. The origin of this frame is located at the positioner mounting flange center. Its Y-Z plane describes the flange surface and X-direction is along the rotational axis in the model of one-axis positioner (see Figure 30(a)). In the model of two-axis positioner, the X-Y plane describes the flange surface and Z-direction is along the rotational axis (see Figure 30(b)).

F_{Lbase} *Linear unit base frame*. This frame is attached to the linear track/gantry base and does not move (see Figure 30(c)). It completely coincides with the robot base frame when $q_L=0$. Usually, for computational convenience, the world frame is also located at this position.



(a) One-axis positioner (b) Two-axis positioner (c) Linear track

Figure 30 Kinematics of typical one-axis positioner, two-axis positioner and linear track

As follows from the above figures, the positioner kinematic model should include two main parameters $\{h,d\}$, where h is the vertical distance from the positioner base from to first rotational axis and d is the distance between the frames F_M and F_{PF} (for two-axis positioner only).

The direct kinematic model of the positioner describes the spatial location of workpiece frame F_W with respect to its base frame F_{Pbase} as a function of the actuated coordinates. For the *one-axis positioner* presented in Figure 34a, the direct kinematic model can be expressed as a product of the following 4×4 homogenous transformation matrices

$$g_P(q_P) = {}^{Pbase}\mathbf{T}_{PF}(q_P) \cdot {}^{PF}\mathbf{T}_W \tag{56}$$

where

$${}^{PBase}\mathbf{T}_{PF}(q_P) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos q_P & -\sin q_P & 0 \\ 0 & \sin q_P & \cos q_P & h \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

and ${}^{PF}\mathbf{T}_W$ is a constant 4×4 homogeneous matrix that depends on six parameters $(X_w, Y_w, Z_w, A_w, B_w, C_w)$ describing position and orientation of the workpiece frame F_W with respect to the mounting flange frame F_{PF}. Using this parameterization, the matrix ${}^{PF}\mathbf{T}_W$ may be presented as

$${}^{PF}\mathbf{T}_W = \left(\begin{array}{ccc|c} \cos A_w \cos B_w & \cos A_w \sin B_w \sin C_w - \sin A_w \cos C_w & \cos A_w \sin B_w \cos C_w + \sin A_w \sin C_w & X_w \\ \sin A_w \sin B_w & \sin A_w \sin B_w \sin C_w + \cos A_w \cos C_w & \sin A_w \sin B_w \cos C_w - \cos A_w \sin C_w & Y_w \\ -\sin B_w & \cos B_w \sin C_w & \cos B_w \cos C_w & Z_w \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{57}$$

For the *two-axis positioner*, the direct kinematic model is expressed as the product of three 4×4 homogenous transformation matrices

$$g_P(\mathbf{q}_P) = {}^{Pbase}\mathbf{T}_M(q_{P1}) \cdot {}^{M}\mathbf{T}_{PF}(q_{P2}) \cdot {}^{PF}\mathbf{T}_W \tag{58}$$

where

$${}^{PBase}\mathbf{T}_M(q_{P1}) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos q_{P1} & -\sin q_{P1} & 0 \\ 0 & \sin q_{P1} & \cos q_{P1} & h \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad {}^{M}\mathbf{T}_{PF}(q_{P2}) = \left(\begin{array}{ccc|c} \cos q_{P2} & -\sin q_{P2} & 0 & 0 \\ \sin q_{P2} & \cos q_{P2} & 0 & 0 \\ 0 & 0 & 1 & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

depending on the actuated coordinates (q_{P1}, q_{P2}). Similarly to the above case, ${}^{PF}\mathbf{T}_{W}$ is the constant matrix describing the workpiece location with respect to the mounting flange.

For the *linear track or gantry*, the direct kinematic model is expressed similar to the one-axis positioner case, i.e. as a product of two 4×4 homogenous transformation matrices

$$g_L(q_L) = {}^{Lbase}\mathbf{T}_{LF}(q_L) \cdot {}^{LF}\mathbf{T}_{Rbase} \tag{59}$$

where

$${}^{Lbase}\mathbf{T}_{Rbase}(q_L) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & q_L \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

and ${}^{LF}\mathbf{T}_{Rbase}$ is the constant matrix describing the robot base location with respect to the mounting flange of the linear unit.

It is clear that the inverse kinematic model of the positioner cannot be presented in systematic way because of lack of degrees of freedom. However, the motion planning and optimization technique developed below does not need the inverse transformation for the positioner or linear track/gantry (only inverse kinematics of the robotic manipulator is used extensively). Nevertheless, it is worth mentioning that for some other applications, such as arc welding etc., the positioner inverse kinematics may be solved in a reduced form assuming that it is necessary to ensure the desired workpiece orientation with respect to the gravity only (Pashkevich et al., 2003).

2.2.3 Integrated Kinematic Model of the Robotic Lay-up System

Using the robot and positioner kinematic models presented above, the lay-up task locations can be described in two ways, with respect to the positioner base frame F_{PBase} and the robot base frame F_{RBase}. After equating these two presentations with respect to the world frame F_{World}, an integrated model of the lay-up task can be derived. The latter is considered as a principle constraint for the optimization problem studied in the following sections. This constraint describes coupled motions of the positioner and the robot, which guarantees that all the task points are visited by the robot end-effector (it is clear that the solution is not unique due to the redundant degrees of freedom). It should be also mentioned that the location of the world frame is arbitrary in general cases and depends on the user references. In this work, for computational convenience, it is assumed that the world frame coincides with the base frame of the linear unit F_{LBase} (see Figure 31), i.e. ${}^{World}\mathbf{T}_{Lbase} = \mathbf{I}$.

Using definitions presented in Figure 31, the desired tool locations can be expressed in two different ways. From the robot perspective, the homogeneous 4×4 matrix describing the tool location may be computed as ${}^{World}\mathbf{T}_{Rtool} = {}^{World}\mathbf{T}_{Lbase} \cdot {}^{Lbase}\mathbf{T}_{Rbase} \cdot {}^{Rbase}\mathbf{T}_{Rtool}$, which after substitution the direct kinematic equations for the robot $g_R(.)$ and linear unit $g_L(.)$ is rewritten as

$${}^{World}\mathbf{T}_{Rtool} = {}^{World}\mathbf{T}_{Lbase} \cdot g_L(q_L) \cdot g_R(\mathbf{q}_R) \tag{60}$$

Further, to express the task locations, it is necessary to take into account the frame alignment conditions:

- The X-axes of the tool and task frames coincide, i.e. $X_{tool} = X_{task}$;
- The Z-axes of the tool and task frames are opposite, i.e. $Z_{tool} = -Z_{task}$

that are described by the following homogeneous matrix

$$ {}^{Rtool}\mathbf{T}_{task} = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{61} $$

The latter allows us to obtain the final expression for the given task locations as functions of actuated coordinates of the robot and the linear unit

$$ {}^{World}\mathbf{T}^{(i)}_{Rtask} = {}^{World}\mathbf{T}_{Lbase} \cdot g_L(q_L^{(i)}) \cdot g_R(\mathbf{q}_R^{(i)}) \cdot {}^{Rtool}\mathbf{T}_{task}; \qquad i = 1,2,\ldots n \tag{62} $$

Similarly, *from the positioner perspective*, the workpiece location with respect to the world frame can be computed as ${}^{World}\mathbf{T}_W = {}^{World}\mathbf{T}_{Pbase} \cdot {}^{Pbase}\mathbf{T}_{PF} \cdot {}^{PF}\mathbf{T}_W$, which after substitution the direct kinematic equations for the positioner $g_P(.)$ allows us to express the task locations in the following way

$$ {}^{World}\mathbf{T}^{(i)}_{Ptask} = {}^{World}\mathbf{T}_{Pbase} \cdot g_P(q_P^{(i)}) \cdot {}^{W}\mathbf{T}^{(i)}_{task}; \qquad i = 1,2,\ldots n \tag{63} $$

where the matrix ${}^{W}\mathbf{T}^{(i)}_{task}$ describes the i^{th} task point in the workpiece frame F_W.

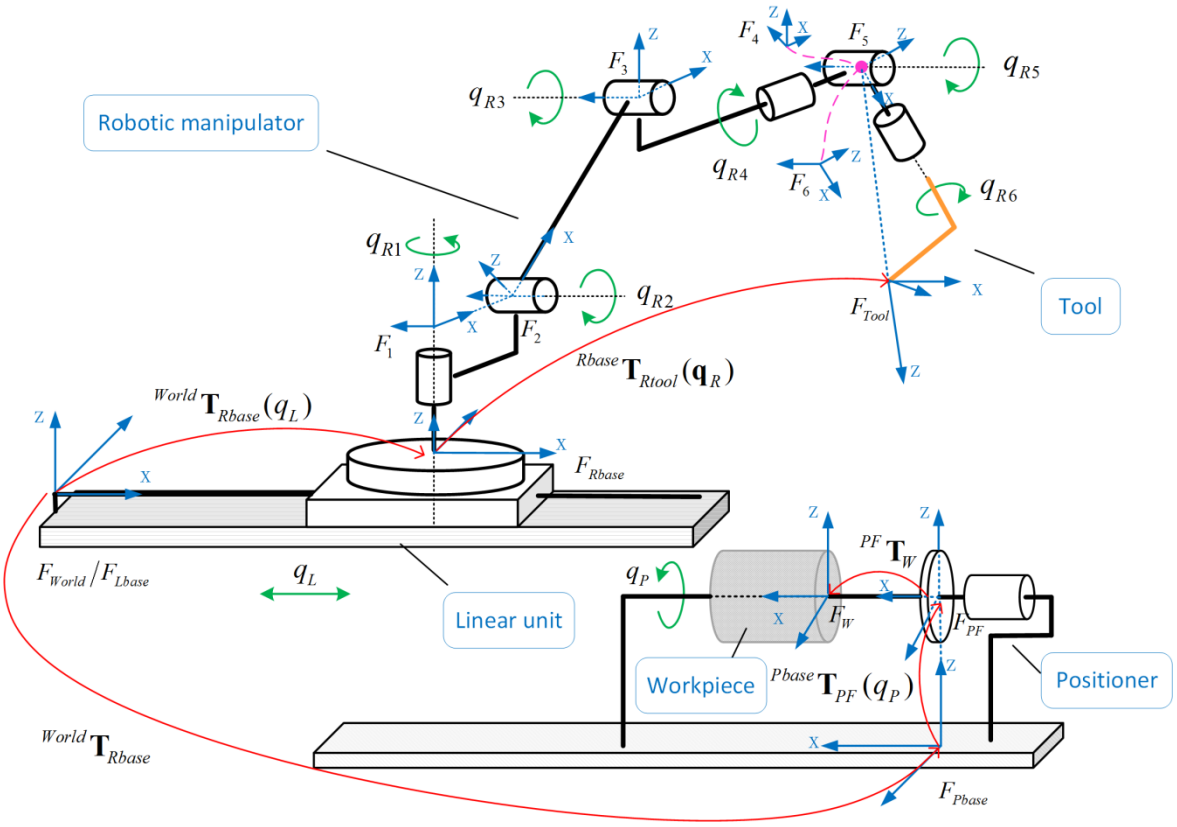


Figure 31 Integration of the robot and the positioner kinematic models

After equating these two presentations (62) and (63), the desired constraints associated with the technological task may be presented in the following form

$$ {}^{World}\mathbf{T}_{Lbase}\cdot g_L(q_L^{(i)})\cdot g_R(\mathbf{q}_R^{(i)})\cdot {}^{Rtool}\mathbf{T}_{task}={}^{World}\mathbf{T}_{Pbase}\cdot g_P(q_P^{(i)})\cdot {}^{W}\mathbf{T}_{task}^{(i)};\qquad i=1,2,...n \tag{64} $$

that may be also rewritten as

$$ g_R(\mathbf{q}_R^{(i)})=\left[{}^{World}\mathbf{T}_{Lbase}\cdot g_L(q_L^{(i)})\right]^{-1}\cdot {}^{World}\mathbf{T}_{Pbase}\cdot g_P(q_P^{(i)})\cdot {}^{W}\mathbf{T}_{task}^{(i)}\cdot\left[{}^{Rtool}\mathbf{T}_{task}\right]^{-1};\qquad i=1,2,...n \tag{65} $$

This equation describes the relations between the actuated coordinates of the robot and the coordinates of the positioner and linear track, which ensure implementation of the given path. In the following section, for the computational convenience, the constraints (65) are presented in slightly different form

$$ \mathbf{q}_R^{(i)}=g_R^{-1}\left(\left[{}^{World}\mathbf{T}_{Lbase}\cdot g_L(q_L^{(i)})\right]^{-1}\cdot {}^{World}\mathbf{T}_{Pbase}\cdot g_P(q_P^{(i)})\cdot {}^{W}\mathbf{T}_{task}^{(i)}\cdot\left[{}^{Rtool}\mathbf{T}_{task}\right]^{-1},\boldsymbol{\mu}\right);\qquad i=1,2,...n \tag{66} $$

that is based on the robot inverse kinematic transformation $g_R^{-1}(.)$, which is not unique and depends on the manipulator configuration index $\boldsymbol{\mu}$.

It should be noted that the obtained equations (65) may be also treated as integrated kinematic model of the lay-up task, which produce at each task point 6 independent scalar constraints related to the position and orientation. It is clear that among 16 scalar relations produced by straightforward equating of 4×4 homogeneous matrices, there are 4 identity equations coming from the last line and 9 dependencies caused by properties of the 3×3 orthogonal sub-matrices describing rotations. On the other side, the obtained model includes 8 variables (6 actuated coordinates of the robot and 2 actuated coordinates of the positioner and the linear track). This redundancy causes multiplicity of the robot/positioner motions implementing the given technological task. Hence, it provides us some space for the motion optimization in the considered robotic workcell.

2.3 MOTION GENERATION IN ROBOTIC LAY-UP SYSTEM

Motion generation in robotic system is defined as the planning of robot/positioner motions while taking into account practical constraints (Brock, 2000). It includes generation of the time profile for Cartesian position/orientation or the time profiles for all actuated coordinates, together with relevant velocities and accelerations. General scheme for this process is presented in Figure 32. It deals with creating inputs for the robot controller allowing executing the desired smooth movements.

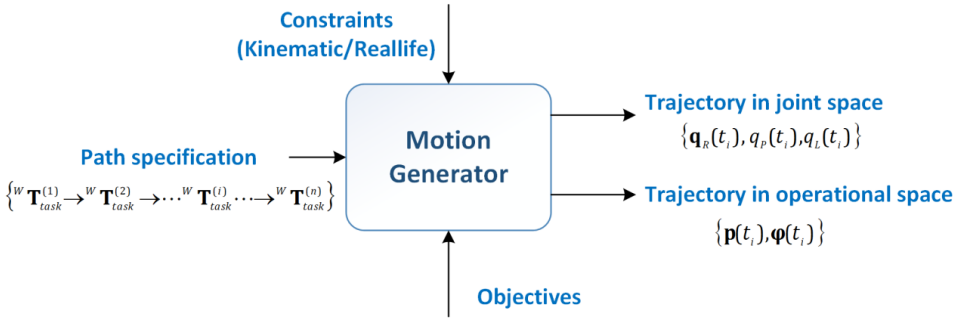


Figure 32 Schematic of motion generation in robotic lay-up system

In robotic lay-up system, the desired path is pre-defined in the workpiece coordinate system as the sequence of 4×4 homogenous matrices (18). The motion generator must produce a sequence of successive joint configurations $(\mathbf{q}_R^{(i)}, q_P^{(i)}, q_L^{(i)}); i = 1,2,...n$ for the robot, the positioner and the linear unit, which ensures that the robot end-effector strictly follows the desired path. It is clear that because of kinematic redundancy, there is no unique trajectory in the joint space corresponding to the given task. For this reason, there arises the problem of optimal redundancy resolution that is addressed in this work.

2.3.1 Formalization of Motion Planning Problem for Robotic Lay-up System

To utilize the redundancy in the best way for the considered lay-up application, it is reasonable to partition the desired motion among the robotic manipulator, the positioner and the linear unit in such a way that the technological tool passes the given path smoothly and as fast as possible. It is obvious that for technological reasons the lay-up speed should be limited, but in practice this limit is usually much higher than the maximum velocity of the end-effector relative to the workpiece that can be achieved in typical industrial robotic cells. For this reason, the principle objective in the motion planning below will be the total processing time (or the tool travelling time along the given path). The secondary objective is the smoothness of the time profiles for all actuated coordinates. It is worth mentioning that such approach to the redundancy resolution essentially differs from the conventional ones where the processing speed is given and the motion planning is targeted at minimizing the total joint displacement or the coordinate ranges associated with the desired path (Dolgui and Pashkevich, 2009, Dolgui and Pashkevich, 2006, Pashkevich et al., 2004).

To present this problem in a more formal way, let us introduce the functions $\mathbf{q}_R(t)$, $q_P(t)$ and $q_L(t)$ that describe the robot, positioner and linear unit motions on the time interval $t \in [0,T]$. In addition, let us define the time instances $\{t_1, t_2, ... t_n\}$ describing the time-points at which the robot end-effector visits the task frames "${}^W\mathbf{T}_{task}^{(1)} \rightarrow {}^W\mathbf{T}_{task}^{(2)} \rightarrow ... \rightarrow {}^W\mathbf{T}_{task}^{(n)}$", where obviously $t_1 = 0$, $t_n = T$. Using this notation, the problem can be presented as minimization of the total travelling time

$$T \rightarrow \min_{\mathbf{q}_R(t),\, q_P(t), q_L(t)} \tag{67}$$

over the set of continuous functions $\mathbf{q}_R(t)$, $q_P(t)$, $q_L(t)$ that satisfy the Cartesian path constraints imposed by equation (64) at all considered time instants

$${}^{World}\mathbf{T}_{Lbase} \cdot g_L(q_L(t_i)) \cdot g_R(\mathbf{q}_R(t_i)) \cdot {}^{Rtool}\mathbf{T}_{task} = {}^{World}\mathbf{T}_{Pbase} \cdot g_P(q_P(t_i)) \cdot {}^W\mathbf{T}_{task}^{(i)}; \qquad i = 1, 2, ... n \tag{68}$$

In addition, some specific constraints describing kinematic and dynamic capacities of the robot, positioner and the linear unit must be also taken into account. Usually, they can be extracted from the manufacturer specifications and presented as the set of the following inequalities

$$\mathbf{q}_R(t_i) \in [\mathbf{q}_R^{\min}, \mathbf{q}_R^{\max}], \quad q_P(t_i) \in [q_P^{\min}, q_P^{\max}], \quad q_L(t_i) \in [q_L^{\min}, q_L^{\max}] \tag{69}$$

describing mechanical constraints (so-called "joint limits") and the actuator capability presented by the maximum allowable speed in the actuating joints

$$\dot{\mathbf{q}}_R(t_i) \in [\dot{\mathbf{q}}_R^{\min}, \dot{\mathbf{q}}_R^{\max}], \quad \dot{q}_P(t_i) \in [\dot{q}_P^{\min}, \dot{q}_P^{\max}], \quad \dot{q}_L(t_i) \in [\dot{q}_L^{\min}, \dot{q}_L^{\max}] \tag{70}$$

and maximum allowable acceleration

$$\ddot{\mathbf{q}}_R(t_i) \in [\ddot{\mathbf{q}}_R^{\min}, \ddot{\mathbf{q}}_R^{\max}], \quad \ddot{q}_P(t_i) \in [\ddot{q}_P^{\min}, \ddot{q}_P^{\max}], \quad \ddot{q}_L(t_i) \in [\ddot{q}_L^{\min}, \ddot{q}_L^{\max}] \tag{71}$$

To give an idea of typical values of the above constraints, Table 11 presents the joint limits, velocities and accelerations for robot KUKA KR210 R3100 that was used at the implementation stage of this work. It should be noted that the maximum accelerations are not usually provided in a straightforward way but they are expressed via the acceleration time (required to achieve maximum speed starting from zero). For the above mentioned robot, the acceleration time is fixed in the controller and it is equal to 0.25 sec. For the actuated positioner and linear track used in our experiments, the maximum velocities are equal to 142°/s and 1.96m/s respectively, while the acceleration time is 0.5 sec.

Table 11 Kinematic and dynamic constraints describing capabilities of robot KUKA KR210

AXIS	Joint limits $\mathbf{q}_R^{\max}$	Maximum joint velocity $\dot{\mathbf{q}}_R^{\max}$	Maximum joint acceleration $\ddot{\mathbf{q}}_R^{\max}$
1	185° to -185°	105°/s	105°s^{-1}/0.25s
2	-5° to -140°	101°/s	101°s^{-1}/0.25s
3	155° to -120°	107°/s	107°s^{-1}/0.25s
4	350° to -350°	136°/s	136°s^{-1}/0.25s
5	122.5° to -122.5°	129°/s	129°s^{-1}/0.25s
6	350° to -350°	206°/s	206°s^{-1}/0.25s

Hence, the considered motion planning problem can be presented as finding eight smooth (bounded with their derivatives) functions $\mathbf{q}_R(t)$, $q_P(t)$ and $q_L(t)$ on $t \in [0,T]$ describing motions of the robot manipulator, positioner and linear unit. These functions must not only suit some boundary conditions at the initial and final points $t_1 = 0$ and $t_n = T$, but must also satisfy the Cartesian path constraints (68) at each intermediate time instant $t_2, t_3, \ldots t_{n-1}$.

From general point of view, this problem can be categorized as the optimization in function space with free end-time and specific constraints at the boundary and intermediate points. In literature, there are several techniques for the problems of such type. They include classical calculus of variations (with *Euler-Lagrange* equation), *Pontryagin*'s maximum principle, and *Bellman*'s dynamic programming (continuous-time version) (Sasane, 2016, Bertsekas et al., 1995). Classical calculus of variations is a traditional approach for maximizing or minimizing numerical objectives over the function space (Gelfand and Silverman, 2000). However, this method assumes that the optimization is performed in an open space, i.e. the unknown functions as well as their derivatives are unconstrained. For this reason, it cannot be applied because of the numerous inequality constraints imposed on the unknown functions (joint limits, velocity and acceleration limits).

The second technique, the maximum principle, is widely used in optimal control theory because of its capability of maximizing numerical objectives over the closed function space (Pontryagin, 1987). The considered problem can be rather easily converted into the relevant form by treating the second derivatives of $\mathbf{q}_R(t)$, $q_P(t)$ and $q_L(t)$ as the control inputs. The latter corresponds to the control system with double saturated integrators, which sequentially compute velocities and joint variables from the

accelerations. For this system, the maximum principle allows rather efficiently solve the time-optimal control problem if the additional constraints are applied at the initial and final points only (classical boundary conditions). However in our case, there are a number of specific path constraints applied at the intermediate points that do not allow easily solving corresponding differential equations and maximizing the Hamiltonian function.

The third technique, Bellman's dynamic programming (Bellman, 2013), relies on the principle that if the optimal trajectory is divided into the sub-arcs, then any one of them is also optimal for the corresponding sub-problem. In the case of the continuous-time, this idea leads to the so called *Hamilton–Jacobi–Bellman* equation in partial derivatives that it is not easy to solve for our technical problem taking into account all above mentioned constraints (joint limits, velocity and acceleration limits, and path constraints). On the other hand, discrete-time formulation with sampling at the points $t_1, t_2, \dots t_n$ looks attractive for numerical solution and will be explored in the following chapters. To author's knowledge, at present there is no known technique that can be straightforwardly applied to the problem formulated in this section.

2.3.2 Additional Constraints from the Actual Robotic System

In practice, in addition to the above mentioned constraints (joint limits, velocity and acceleration limits, and path constraints), there are some other important limitations to be taken into account while planning optimal motions in considered robotic cell. First of all, it is necessary to avoid collisions between the manipulator and other workcell components. Besides, it is reasonable to avoid movements in the neighborhood of the manipulator singular configurations, where it is difficult to control the robot motions precisely. Let us present relevant techniques allowing excluding some potential manipulator configurations taking into account these limitations.

In literature, there are several schemes addressing collision detection in path planning of industrial robots. A common one is based on approximating the actual robot geometry by certain simple structures that completely enclose the robot. For example, in some works (Chang et al., 1990, Ennen et al., 2016, Liu and Tomizuka, 2016), each robot link is replaced by a simple cylinder with the radius equal to the maximum distance between the cylinder axis and the link surface. A collision is detected if the minimal distance between the cylinder axes is smaller than the sum of the related radiuses. This idea is illustrated in Figure 33 where two components are described by the cylinders with the axes $\mathbf{P}_n\mathbf{P}_{n+1}$ and $\mathbf{P}_m\mathbf{P}_{m+1}$ and with the radiuses r_n and r_m respectively. To ensure that there is no collision between them, the following sufficient condition should be satisfied

$$dist\left(\mathbf{P}_n\mathbf{P}_{n+1}, \mathbf{P}_m\mathbf{P}_{m+1}\right) > r_n + r_m \tag{72}$$

where $dist(.)$ is the function for calculating minimum Euclidean distances between the cylinder axes. It is clear that this technique should be applied to detect collisions between the robot and positioner, as well as between the technological tool and workpiece, positioner and the robot. However, it is not suitable for detection of the robot self-collisions.

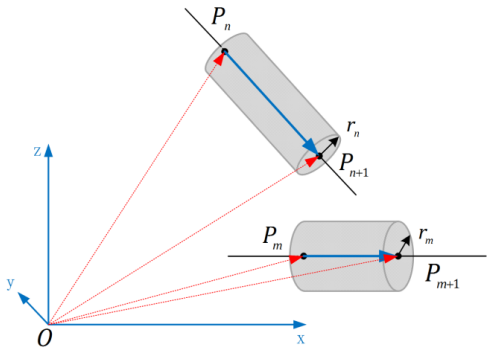


Figure 33 Collision detection based on intersection of two cylinders

In the above expression (72), the function $dist(.)$ can be implemented using the robust technique proposed in Eberly's work (Eberly, 2015). To present it, let us parameterize the cylinder axes as $\mathbf{s}_n(\lambda_1)=\mathbf{P}_n+\lambda_1\cdot(\mathbf{P}_{n+1}-\mathbf{P}_n)$ and $\mathbf{s}_m(\lambda_2)=\mathbf{P}_m+\lambda_2\cdot(\mathbf{P}_{m+1}-\mathbf{P}_m)$ where $\lambda_{1,2}\in[0,1]$. Then the distance between two points on the axes is expressed as follows

$$\Delta\mathbf{s}(\lambda_1,\lambda_2)=\left\|(\mathbf{P}_n-\mathbf{P}_m)+\lambda_1\cdot(\mathbf{P}_{n+1}-\mathbf{P}_n)-\lambda_2\cdot(\mathbf{P}_{m+1}-\mathbf{P}_m)\right\|, \tag{73}$$

which allows us to present the distance between the cylinder axes as the solution of the following quadratic optimization problem over the closed set $(\lambda_1,\lambda_2)\in[0,1]\times[0,1]$

$$\Delta\mathbf{s}^2=\lambda_1^2\cdot\mathbf{e}_n^2+\lambda_2^2\cdot\mathbf{e}_m^2-2\lambda_1\lambda_2\cdot\mathbf{e}_n\cdot\mathbf{e}_m+2\lambda_1\cdot\mathbf{e}_{mn}\cdot\mathbf{e}_n-2\lambda_2\cdot\mathbf{e}_{mn}\cdot\mathbf{e}_m+\mathbf{e}_{mn}^2\ \rightarrow\ \min_{\lambda_1,\lambda_2} \tag{74}$$

where $\mathbf{e}_{mn}=\mathbf{P}_n-\mathbf{P}_m$, $\mathbf{e}_m=\mathbf{P}_{m+1}-\mathbf{P}_m$, $\mathbf{e}_n=\mathbf{P}_{n+1}-\mathbf{P}_n$. Since $\Delta\mathbf{s}^2(\lambda_1,\lambda_2)$ is a continuously differentiable function, the minimum occurs either at the boundary of the square $(\lambda_1,\lambda_2)\in[0,1]\times[0,1]$ or at an interior point where the gradient is equal to zero, i.e.

$$\begin{aligned}\frac{\partial\Delta s^2}{\partial\lambda_1}&=2\lambda_1\cdot\mathbf{e}_n^2-2\lambda_2\cdot\mathbf{e}_n\cdot\mathbf{e}_m+2\cdot\mathbf{e}_{mn}\cdot\mathbf{e}_n=0\\ \frac{\partial\Delta s^2}{\partial\lambda_2}&=2\lambda_2\cdot\mathbf{e}_m^2-2\lambda_1\cdot\mathbf{e}_n\cdot\mathbf{e}_m-2\cdot\mathbf{e}_{mn}\cdot\mathbf{e}_m=0\end{aligned} \tag{75}$$

which yields

$$\hat{\lambda}_1=\frac{(\mathbf{e}_n\cdot\mathbf{e}_m)\cdot(\mathbf{e}_m\mathbf{e}_{mn})-\mathbf{e}_m^2\cdot(\mathbf{e}_n\cdot\mathbf{e}_{mn})}{\mathbf{e}_n^2\cdot\mathbf{e}_m^2-(\mathbf{e}_n\cdot\mathbf{e}_m)^2};\qquad \hat{\lambda}_2=\frac{\mathbf{e}_n^2\cdot(\mathbf{e}_m\cdot\mathbf{e}_{mn})-(\mathbf{e}_n\cdot\mathbf{e}_m)\cdot(\mathbf{e}_n\cdot\mathbf{e}_{mn})}{\mathbf{e}_n^2\cdot\mathbf{e}_m^2-(\mathbf{e}_n\cdot\mathbf{e}_m)^2} \tag{76}$$

provided that $\mathbf{e}_n^2\cdot\mathbf{e}_m^2-(\mathbf{e}_n\cdot\mathbf{e}_m)^2\neq0$. It is clear that the condition $(\hat{\lambda}_1,\hat{\lambda}_2)\in[0,1]\times[0,1]$ must be verified to be sure that the obtained point can be accepted for computing the desired distance.

Otherwise, the minimum of $\Delta\mathbf{s}^2(\lambda_1,\lambda_2)$ occurs on the boundary of $[0,1]\times[0,1]$. Since here the level curves of the objective function are ellipses, the minimum value of $\Delta\mathbf{s}^2$ corresponds to the situation when the relevant ellipse just touches the square edge. Hence, four possible cases $(\hat{\lambda}_1,0)$, $(\hat{\lambda}_1,1)$, $(0,\hat{\lambda}_2)$ and $(1,\hat{\lambda}_2)$ should be considered where $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are computed assuming that only one corresponding derivate is equal to zero. The latter yield the following critical points

$$\begin{array}{ll} \hat{\lambda}_1 = -\mathbf{e}_{mn}\cdot\mathbf{e}_n/\mathbf{e}_n^2 & \hat{\lambda}_2 = 0 \\ \hat{\lambda}_1 = (\mathbf{e}_n\cdot\mathbf{e}_m - \mathbf{e}_{mn}\cdot\mathbf{e}_n)/\mathbf{e}_n^2 & \hat{\lambda}_2 = 1 \\ \hat{\lambda}_1 = 0 & \hat{\lambda}_2 = \mathbf{e}_{mn}\cdot\mathbf{e}_m/\mathbf{e}_m^2 \\ \hat{\lambda}_1 = 1 & \hat{\lambda}_2 = (\mathbf{e}_n\cdot\mathbf{e}_m + \mathbf{e}_{mn}\cdot\mathbf{e}_m)/\mathbf{e}_m^2 \end{array} \quad (77)$$

that should be verified while finding the global minimum on the square $[0,1]\times[0,1]$. In addition, the square corners $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ should be also evaluated, which finally leads to 9 separate cases to be considered in this optimization problem.

It should be also noted that equations (76) can be applied only when $\mathbf{e}_n^2\cdot\mathbf{e}_m^2 - (\mathbf{e}_n\cdot\mathbf{e}_m)^2 \neq 0$, i.e. if the vectors $\mathbf{e}_n$ and $\mathbf{e}_m$ are non-collinear. In the opposite case, when the cylinder axes $\mathbf{P}_n\mathbf{P}_{n+1}$ and $\mathbf{P}_m\mathbf{P}_{m+1}$ are parallel, the equations (75) are linear dependent and produce the following set of critical points to be verified

$$\begin{array}{ll} \hat{\lambda}_1 \in \{0,1\} & \hat{\lambda}_2 = (\lambda_1\cdot\mathbf{e}_n^2 + \mathbf{e}_{mn}\cdot\mathbf{e}_n)/\mathbf{e}_n\cdot\mathbf{e}_m \\ \hat{\lambda}_1 = (\mathbf{e}_{mn}\cdot\mathbf{e}_m - \lambda_2\cdot\mathbf{e}_m^2)/\cdot\mathbf{e}_n\cdot\mathbf{e}_m & \hat{\lambda}_2 \in \{0,1\} \end{array} \quad (78)$$

If any of such $(\hat{\lambda}_1,\hat{\lambda}_2)$ belongs to the square $[0,1]\times[0,1]$, it can be used for computing the desired minimum distance. Otherwise, it is necessary to verify the square corners $\{0,1\}\times\{0,1\}$ and select the minimum of 4 corresponding distances. Therefore, the above presented expressions allow us to compute the distance between the cylinder axes required for the collision detection in the robotic cell.

In order to apply this collision detection technique to the considered task, a proper strategy for selecting pairs of cylinders is also important. Here, because of the existence of the joint limits, it is impossible to have collisions inside the robot (between links) or inside the positioner. Besides, for this task, the collision check between the workpiece and the end-effector is meaningless. So, only the following pairs should be tested: 1) robot forearm/workpiece; 2) robot forearm/positioner shaft; 3) robot forearm/tool part B; 4) tool part A/positioner shaft; 5) tool part B/workpiece; 6) tool part B/positioner shaft. The latter corresponds to the robot description presented in Figure 34 that is based on the approximations listed below:

- Robot forearm: approximated by cylinder 1 with center axis $\mathbf{P}_{C1}^1\mathbf{P}_{C1}^2$ and radius r_1;
- Tool part A: approximated by cylinder 2 with center axis $\mathbf{P}_{C2}^1\mathbf{P}_{C2}^2$ and radius r_2;
- Tool part B: approximated by cylinder 3 with center axis $\mathbf{P}_{C3}^1\mathbf{P}_{C3}^2$ and radius r_3;
- Workpiece: approximated by cylinder 4 with center axis $\mathbf{P}_{C4}^1\mathbf{P}_{C4}^2$ and radius r_4;
- Positioner shaft: approximated by cylinder 5 with center axis $\mathbf{P}_{C5}^1\mathbf{P}_{C5}^2$ and radius r_5.

Using these notations, the desired collision-free condition may be presented as follows

$$\begin{cases} dist\left(\mathbf{P}_{C1}^1\mathbf{P}_{C1}^2,\ \mathbf{P}_{C3}^1\mathbf{P}_{C3}^2\right) > r_1 + r_3 \\ dist\left(\mathbf{P}_{C1}^1\mathbf{P}_{C1}^2,\ \mathbf{P}_{C4}^1\mathbf{P}_{C4}^2\right) > r_1 + r_4 \\ dist\left(\mathbf{P}_{C1}^1\mathbf{P}_{C1}^2,\ \mathbf{P}_{C5}^1\mathbf{P}_{C5}^2\right) > r_1 + r_5 \\ dist\left(\mathbf{P}_{C2}^1\mathbf{P}_{C2}^2,\ \mathbf{P}_{C5}^1\mathbf{P}_{C5}^2\right) > r_2 + r_5 \\ dist\left(\mathbf{P}_{C3}^1\mathbf{P}_{C3}^2,\ \mathbf{P}_{C4}^1\mathbf{P}_{C4}^2\right) > r_3 + r_4 \\ dist\left(\mathbf{P}_{C3}^1\mathbf{P}_{C3}^2,\ \mathbf{P}_{C5}^1\mathbf{P}_{C5}^2\right) > r_3 + r_5 \end{cases} \quad (79)$$

However, it should be mentioned that the above set of inequalities is only a sufficient but not necessary condition for no-collision configurations of the considered robotic cell. Nevertheless, it is rather simple and allows easily reducing the set of admissible candidates in the considered motion planning problem. It is clear that in practice it is necessary to verify the remaining configurations carefully using more precise approximation of the workcell components.

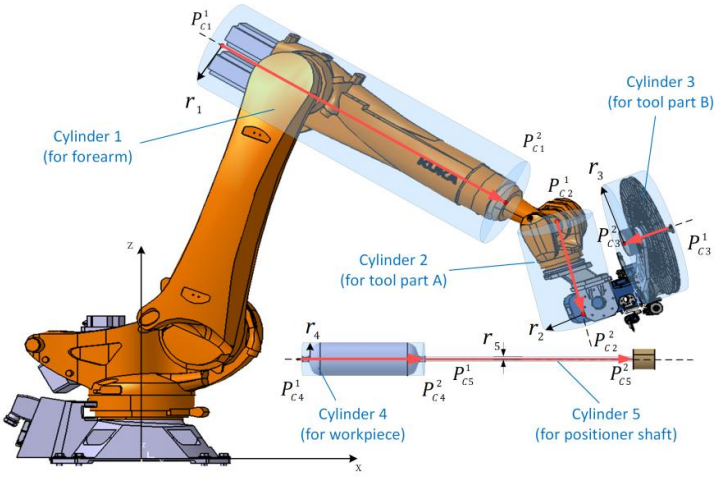


Figure 34 Cylinder-based descriptions of the workcell components for collision detection

To detect possible collisions taking into account real geometry of robot, technological tool, workpiece and positioner, some standard functions provided by commercial CAD/CAM software can be used. In CAD/CAM environments, all components of the robotic cell can be created by 3D modelers and there are several schemes to represent these objects. The simplest ones are the wireframe representation that operates with edges and vertices only. Another scheme is the surface representation where a solid is described by its boundaries (using planar faces, NURBS surfaces, triangular meshes etc.). The most advanced technique widely used in modern CAD system is usually called solid modeling and allows presenting all the object geometry, topology and physical properties in the form of a data structure. It is based on the combination of different methods, such as parameterized primitive instancing, spatial occupancy enumeration, sweeping etc. For example, solids may be defined via finite number of regularized Boolean operations on the primitives or by means of primitive sweeping along a certain space trajectory. However, in this work we will use a simple triangular mesh representation allowing essentially speed-up the collision test required for the motion planning for the considered robotic system. Modern commercial CAD systems usually include integrated functions providing transformations of internal representation of the robotic cell into the set of triangle primitives describing the surfaces of workpiece as well as the components of the robot, technological tool and positioner (see Figure 35). Hence, the desired collision test between workcell components is reduced to numerous checks of *triangle-to-triangle* intersections.

In literature, there are several techniques for checking *triangle-to-triangle* intersections, and the most common of them was developed by *Möller* (Möller, 1997). To present this technique, let us describe two triangles T_1 and T_2 by their vertices $\mathbf{V}_1^1$, $\mathbf{V}_2^1$, $\mathbf{V}_3^1$ and $\mathbf{V}_1^2$, $\mathbf{V}_2^2$, $\mathbf{V}_3^2$ respectively, and denote the planes containing the triangles as S_1 and S_2. Corresponding equations of these planes may be written in the following form

$$\mathbf{n}_i \cdot \mathbf{P} - \mathbf{n}_i \cdot \mathbf{V}_1^i = 0; \qquad i = 1, 2. \tag{80}$$

where $\mathbf{n}_i = (\mathbf{V}_2^i - \mathbf{V}_1^i) \times (\mathbf{V}_3^i - \mathbf{V}_1^i)$ and $\mathbf{P}$ denotes the point belonging to the considered plane. Using this notation, one can compute the signed distances from the vertices $\mathbf{V}_1^1$, $\mathbf{V}_2^1$, $\mathbf{V}_3^1$ of the first triangle to the plane S_2:

$$d_i^{S_2} = \mathbf{n}_2 \cdot \mathbf{V}_i^1 - \mathbf{n}_2 \cdot \mathbf{V}_1^2; \qquad i = 1,2,3. \tag{81}$$

It is clear that if three computed values $d_i^{S_2}$ are either all positive or all negative, then the triangle T_1 is located on one side of S_2, which guarantees that there is no intersection between the triangles T_1 and T_2. Similar test should be applied to the triangle T_2 and the plane S_1. This early test applied at the first step allows us avoiding a lot of computations related to the detailed analysis presented below.

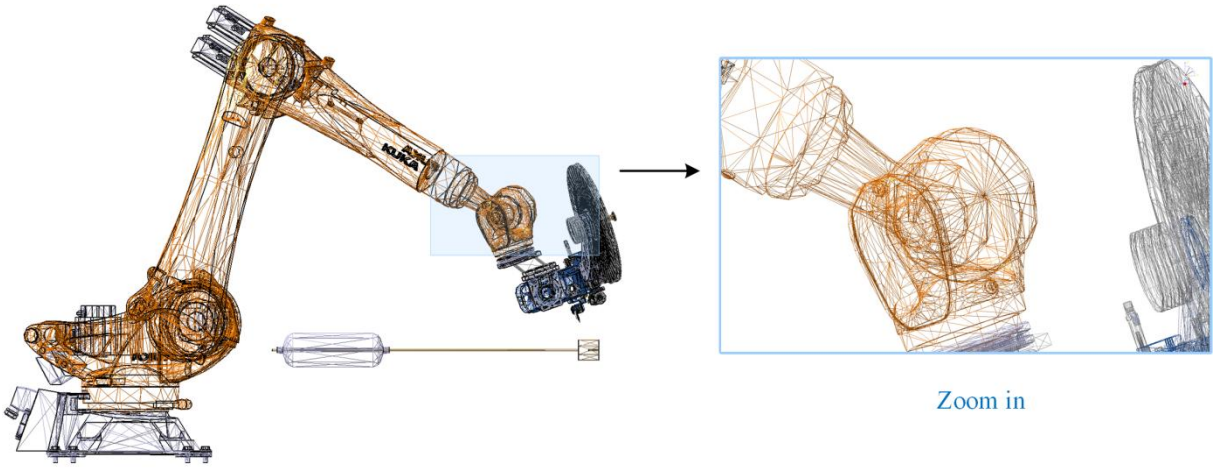


Figure 35 Triangle-based description of the workcell components for collision detection

The second step deals with analysis of the case when all $d_i^{S_2} = 0$, which means that the triangles T_1 and T_2 are coplanar (in practice, the above equation should be replaced by the inequality $|d_i^{S_2}| < \varepsilon$). In this case, the problem is reduced to a simple 2D *triangle-triangle* overlap test. First, the edges of T_1 should be checked for intersection with the edges of T_2 (or vice versa) that can be easily implemented using the above presented technique computing the distance between the line segments (73). It is clear that an alternative technique can be also applied that is based on 2D computer geometry where the line segments intersection is detected if both segment ends lie on the opposite sides of the corresponding infinite line (Ericson, 2004). For instance, this approach yields the following inequalities for detection intersection of the edges $\mathbf{V}_1^1\mathbf{V}_2^1$ and $\mathbf{V}_1^2\mathbf{V}_2^2$

$$\left((\mathbf{V}_1^2 - \mathbf{V}_1^1) \cdot \mathbf{s}_1\right) \cdot \left((\mathbf{V}_2^2 - \mathbf{V}_1^1) \cdot \mathbf{s}_1\right) < 0 \quad \vee \quad \left((\mathbf{V}_1^1 - \mathbf{V}_1^2) \cdot \mathbf{s}_2\right) \cdot \left((\mathbf{V}_2^1 - \mathbf{V}_1^2) \cdot \mathbf{s}_2\right) < 0 \tag{82}$$

where $\mathbf{s}_1 = (\mathbf{v}_1 \times \mathbf{v}_2) \times \mathbf{v}_1$, $\mathbf{s}_2 = (\mathbf{v}_1 \times \mathbf{v}_2) \times \mathbf{v}_2$, $\mathbf{v}_1 = \mathbf{V}_1^1 - \mathbf{V}_2^1$ and $\mathbf{v}_2 = \mathbf{V}_1^2 - \mathbf{V}_2^2$. In case that there is no intersections between the triangle edges, it is necessary to check whether the triangle T_1 is located inside of the triangle T_2 or not (and vice versa). In order to do this, all vertices of T_1 can be checked if they all lie in the interior of T_2. For example, a condition for the vertex $\mathbf{V}_i^1$ to be inside of the triangle T_2 is expressed as

$$\mathbf{w}_1 \cdot \mathbf{w}_2 \geq 0 \quad \wedge \quad \mathbf{w}_2 \cdot \mathbf{w}_3 \geq 0 \tag{83}$$

where $\mathbf{w}_1=(\mathbf{V}_1^2-\mathbf{V}_i^1)\times(\mathbf{V}_2^2-\mathbf{V}_i^1)$, $\mathbf{w}_2=(\mathbf{V}_2^2-\mathbf{V}_i^1)\times(\mathbf{V}_3^2-\mathbf{V}_i^1)$ and $\mathbf{w}_3=(\mathbf{V}_3^2-\mathbf{V}_i^1)\times(\mathbf{V}_1^2-\mathbf{V}_i^1)$. If all three $\mathbf{V}_1^1$, $\mathbf{V}_2^1$, $\mathbf{V}_3^1$ are inside the T_2, T_1 is inside of T_2, i.e. the triangles intersect (similar test should be applied to $\mathbf{V}_1^2$, $\mathbf{V}_2^2$, $\mathbf{V}_3^2$).

The third step deals with the case when some of the distances $d_i^{S_2}$ have opposite signs. Geometrically, it means that both planes S_1 and S_2 intersect corresponding triangles T_2 and T_1 (Figure 40). It is clear that such intersections are the segments of the common line of the planes S_1 and S_2, which is directed along the vector $\mathbf{n}_1\times\mathbf{n}_2$. If these two segments overlap, the triangles intersect as well. To derive relevant test, let us describe this common line by a parametric equation $\mathbf{P}=\mathbf{P}_0+t\cdot(\mathbf{n}_1\times\mathbf{n}_2)$, $t\in\Re$ where $\mathbf{P}_0$ is one of the points of this line that can be easily found. Then it is necessary to find four parameters $\{t_1,t_2\}$ and $\{t_3,t_4\}$ defining the above mentioned segments $\mathbf{A}_1\mathbf{A}_2$ and $\mathbf{A}_3\mathbf{A}_4$. For example, if the parameter t_1 corresponding to the cross point of the edge $\mathbf{V}_1^1\mathbf{V}_2^1$ and the common $\mathbf{P}$ (see Figure 36) can be computed as

$$t_1=p_1^{S_1}+\frac{d_1^{S_1}}{d_1^{S_1}-d_2^{S_1}}\cdot(p_2^{S_1}-p_1^{S_1}) \tag{84}$$

where $p_i^{S_1}=(\mathbf{n}_1\times\mathbf{n}_2)\cdot(\mathbf{V}_i^1-\mathbf{P}_0)$. After computing $t_1\ldots t_4$, the problem is reduced to a simple 1D *segment-segment* overlap test. It yields the following inequalities for the intersection detection

$$\max\{t_1,t_2\}\geq\min\{t_3,t_4\}\ \wedge\ \max\{t_3,t_4\}\geq\min\{t_1,t_2\} \tag{85}$$

Hence, the three simple tests presented above allow us to detect intersections between the triangles belonging to the surfaces of the workcell components. However, creation of the relevant sets of the triangles is not a trivial task.

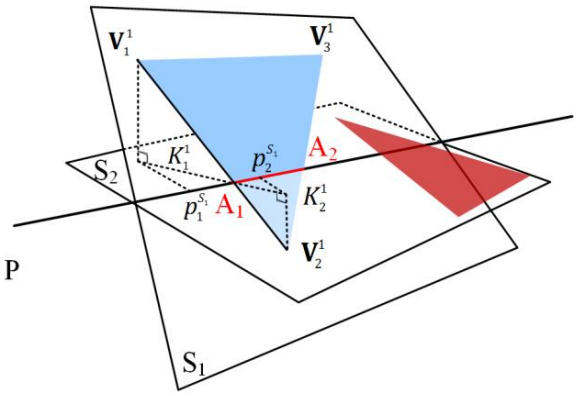


Figure 36 Situation of two triangle planes intersecting at a straight line

In some commercial CAD/CAM software packages, for example CATIA used in this work, the collision detection function can be straightforwardly invoked in "DMU Kinematics" workbench. Using the interface of the "Check Clash" module, the user can define the type of collisions (clearance, contact or clash) to be tested between any selected pair of workcell components (see Figure 37). Then, after activating the sensors on the "Kinematics Simulation" panel, the relevant collision data can be recorded into a ".xls" or ".txt" file. This procedure is implemented by checking all triangle pairs belonging to the selected objects. However, as follows from our experience, using a very fine triangular mesh precisely describing the mechanical components leads to a very high computational time. For example, for the application example considered in Chapter 4, it took more than 3 hours to

test collisions for 10^5 manipulator configurations if the triangulation accuracy was set to 0.2 mm (number of the triangles is over 5.3×10^5, processor Intel® i5 @ 1.70GHz 2.40GHz, 8G memory). On the other side, setting triangulation accuracy to 4.0 mm yielded about 3.5×10^5 triangles and allowed reducing the computing time down to 2 hours. Hence, the collision test is one of the most time-consuming procedures in the motion planning process studied in this work and it deserves some special efforts of speeding up.

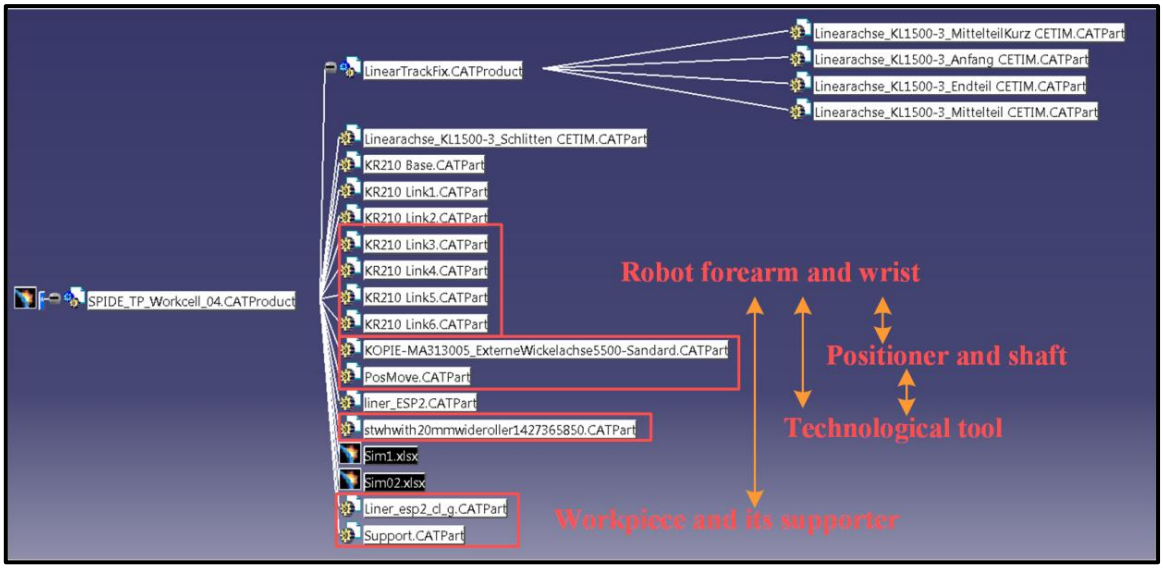


Figure 37 Structural presentation of the robotic lay-up workcell in CATIA, and component pairs selected for collision detection

To integrate the collision constraint into the in the considered motion planning problem, let us introduce a binary function $cols(.)\in\{0,1\}$, which is equal to zero if there is no collision detected. This allows us to express the non-collision requirement for the desired motions in the robotic system in a compact form

$$cols\left(\mathbf{q}_R(t), q_P(t), q_L(t)\right)=0;\ \forall t\in[0,T] \tag{86}$$

where $\mathbf{q}_R, q_P, q_L$ are the joint variables describing current configurations of the manipulator, positioner and linear unit.

In practice, in addition to collision avoidance, the optimal manipulator motions should be rather far from **singular configurations** where robot loses some of its degrees of freedom and its kinematic control is difficult. In particular, in the neighborhood of a singular location, an enormous change of the joint coordinates is required for a requested normal change of the robot posture. Generally, the manipulator singular configurations can be identified by studying its Jacobian matrix that becomes rank-deficient. For the considered problem of the optimal motion planning, it is necessary to introduce some numerical indices allowing evaluating how close is the manipulator to a singular configuration.

The simplest quantitative measure of this type (so-called manipulability measure) was proposed by Yoshikawa (Yoshikawa, 1985) and is expressed as follows

$$w=\sqrt{\det\left(\mathbf{J}(\mathbf{q}_R)\cdot\mathbf{J}^T(\mathbf{q}_R)\right)} \tag{87}$$

where $\mathbf{J}(\mathbf{q}_R)$ is the manipulator kinematic Jacobian corresponding to the joint variables $\mathbf{q}_R$. It is clear that $w=0$ if the configuration is singular. Geometrically, this measure is proportional to the volume of the manipulability ellipsoid that represents an ability of manipulator to move its end-effector in various directions. However, this index can be hardly used for the considered problem because it provides false evaluation of the manipulator capabilities in the case when the ellipsoid is stretched, i.e. some axes are much longer than other ones but the volume is high enough. Kinematically, it means that the end-effector can easily move at higher speed in the direction of the ellipsoid major axis, while only lower speed is achievable in direction of the minor axes (Huo and Baron, 2008).

For this reason, it is logical to utilize another index that can be induced from the manipulability ellipsoid, which is the Jacobian condition number (Salisbury and Craig, 1982). This index is defined as the ratio of the maximum and minimum of the ellipsoid axes and can be expressed via the singular values σ_i of the Jacobian matrix

$$cond\left(\mathbf{J}(\mathbf{q}_R)\right)=\frac{\max\sigma_i}{\min\sigma_i} \tag{88}$$

Corresponding singular values $\sigma_i \geq 0$ are obtained from the singular value decomposition (SVD) of matrix $\mathbf{J}(\mathbf{q}_R)$ as follows

$$\mathbf{J}(\mathbf{q}_R)=\mathbf{U}\cdot\mathbf{S}\cdot\mathbf{V}^T;\quad \mathbf{S}\equiv\mathrm{diag}\left(\sigma_1,\sigma_2,\ldots\sigma_6\right) \tag{89}$$

where $\mathbf{U}$ and $\mathbf{V}$ are 6×6 orthogonal matrices, and $\mathbf{S}$ is a 6×6 diagonal matrix of the singular values. It can be easily proved that if the manipulator is in a singular state, then at least one of its singular value is equal to zero, i.e. $\sigma_i=0$. The corresponding column of $\mathbf{U}$, $\mathbf{u}_i$ is referred to as the singular direction. In this state, the motion along the singular directions is not possible. So, the condition number can be treated as an index of the ellipsoid directional uniformity. When its value is close to 1, the ellipsoid is close to a sphere and the manipulator configuration is evaluated as well-conditioned. If the minor axis of the ellipsoid is close to zero (i.e. $\min\sigma_i\approx 0$), the condition number becomes high and this state is evaluated as ill-conditioned. So, the condition number can be used in our research as the distance criterion to avoid manipulator's singularities in the motion planning. Mathematically, relevant constraint can be presented in the form of the following inequality

$$cond\left(\mathbf{J}(\mathbf{q}_R(t_i))\right)<C_{\max};\quad \forall t\in[0,T] \tag{90}$$

where $C_{\max}$ is the user defined value that is about 5 for the application example studied in this work. It should be mentioned that in this work, in order to take into account the non-homogeneity of $\{\sigma_1,\sigma_2,\ldots\sigma_6\}$, the definition of the condition number was slightly modified. In particular, the ratio (88) was computed separately for the translational $\{\sigma_1,\sigma_2,\sigma_3\}$ and rotational $\{\sigma_4,\sigma_5,\sigma_6\}$ components and the maximum of them was used as the condition number.

Summarizing this section, it is necessary to propose a practical strategy of verifications of the above collision and singularity constraints. To minimize overall computing time required for both the data preparation and motion planning itself, first it is reasonable to verify the configuration candidates for the singularities (see Figure 38). As it was mentioned above, it is a rather fast procedure that allows us reducing the size of the initial set. Further, it is logical to apply the "rough" collision detection test based on the cylinder-cylinder intersections that also does not require high computational expenses. And finally, the "fine" collision test should be applied that is based on the triangle-triangle

intersections, which provides precise evaluations of the possible configurations analyzed at the motion planning stage. It is clear that such sequence of tests, where many configurations are already eliminated before applying the time-consuming "fine" collision test, allows us avoiding some unnecessary computations.

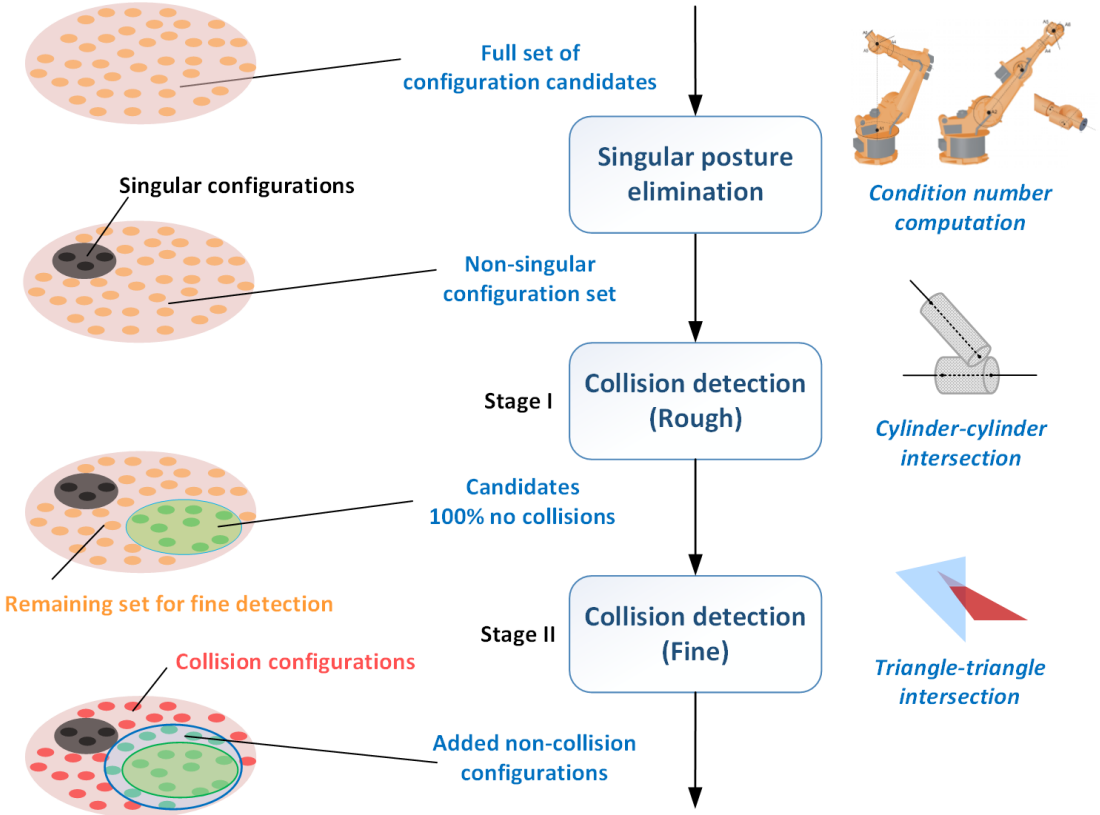


Figure 38 Strategy of verifications on the additional constraints

Hence, after integrating the additional constraints presented above, the considered optimization problem related to the motion planning can be finally formulated as follows

$$
\begin{cases}
\textit{find} \\
\quad \left\{ \mathbf{q}_R(t_i);\ \ q_P(t_i);\ \ q_L(t_i) \,\middle|\, i = 1,2,\ldots n \right\} \\
\textit{such that} \\
\quad T \to \min \\
\textit{s.t.} \\
\quad {}^{World}\mathbf{T}_{Lbase} \cdot g_L(q_L(t_i)) \cdot g_R(\mathbf{q}_R(t_i)) \cdot {}^{Rtool}\mathbf{T}_{task} = {}^{World}\mathbf{T}_{Pbase} \cdot g_P(q_P(t_i)) \cdot {}^{W}\mathbf{T}_{task}^{(i)} \\
\\
\quad \mathbf{q}_{R,P,L}^{\min} \le \mathbf{q}_{R,P,L}(t_i) \le \mathbf{q}_{R,P,L}^{\max} \\
\quad \dot{\mathbf{q}}_{R,P,L}^{\min} \le \dot{\mathbf{q}}_{R,P,L}(t_i) \le \dot{\mathbf{q}}_{R,P,L}^{\max} \\
\quad \ddot{\mathbf{q}}_{R,P,L}^{\min} \le \ddot{\mathbf{q}}_{R,P,L}(t_i) \le \ddot{\mathbf{q}}_{R,P,L}^{\max} \\
\\
\quad cond\left(\mathbf{J}(\mathbf{q}_R(t_i)) \right) < C_{\max} \\
\quad cols\left(\mathbf{q}_R(t_i), q_P(t_i), q_L(t_i) \right) = 0 \\
\textit{where} \\
\quad t_1 = 0,\ t_n = T; \qquad i = 1,2,\ldots n
\end{cases}
\tag{91}
$$

and it aims at finding 8 continuous functions, i.e. $\mathbf{q}_R(t)$, $q_P(t)$ and $q_L(t)$ describing the robot manipulator, positioner and linear unit motions. Solution of this problem will be presented in the next section.

2.4 SUMMARY

This chapter is devoted to the kinematic modeling of a typical redundant robotic system used in the composite lay-up technology and formalization of the related optimal motion planning problem. The main contribution is presenting the considered technological problem in a pure mathematical way, as finding of the time-optimal trajectory in the joint space of the robotic system under specific constraints related to the composite lay-up task. The latter include both conventional kinematic constraints (joint limits, maximum joint velocities/accelerations), allowable distances to the singularities and obstacles as well as the Cartesian path constraints issued from the composite lay-up process.

In more details, the contribution of Chapter 2 can be summarized as follows:

(i) *Formal description of the technological task* and its representation in the form of the sequence of homogenous transformation matrices (or the location vector sequence) describing desired position and orientation of the robot end-effector for the considered lay-up process.

(ii) *Integrated kinematic model* of the redundant robotic lay-up system incorporating description both the workcell mechanical components (robotic manipulator, workpiece positioner, workspace extensioner, technological tool) and the desired technological task.

(iii) *Formalization of the motion planning problem* for robotic lay-up system and presenting it in the form of optimization in the function space under the specific constraints describing the technological task, capabilities of the robotic manipulator (joint limits, maximum velocities and accelerations in joints) and the requirements for the manipulator postures expressed via the singularity and collision constraints.

For the optimization problem derived in this chapter, no technique exists in literature that can be directly applied because this problem is highly non-linear and includes very specific constraints arising from the corresponding technology. For these reasons, a new method dedicated to motion planning in the redundant robotic system for automated lay-up process will be developed in the following chapter.

CHAPTER 3 MOTION PLANNING FOR ROBOTIC LAY-UP SYSTEM

This chapter is devoted to the development of a new motion planning method for the redundant robotic systems utilized in the automated composite lay-up process. It is based on the transforming the original problem into a combinatorial one and applying the dynamic programming principle. First, the task graph is created by discretizing the redundant variables and sequentially applying to all task locations the direct kinematics of the positioner (and workspace extension unit) as well as inverse kinematics of the robot. Then, the developed algorithm based on dynamic programming generates the time-optimal motion taking into account all relevant constraints. To reduce the computing time, the time-optimal motion planning is divided in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with large discretization step. Then, the same technique is applied in the local search space, which is created with smaller step in the neighborhood of the obtained trajectory. Alternatively, the second stage may implement a straightforward smoothing of the redundant variable profiles. Efficiency of them and advantages compared to the conventional techniques are confirmed by several case studies.

3.1 PROBLEM TRANSFORMATION INTO DISCRETE FORM

To solve the motion planning problem (91) presented in Chapter 2, which deals with optimization in function space with specific constraints at the boundary and intermediate points, it is reasonable to transform it from the continuous form into a discrete one. This transformation allows us to apply some combinatorial optimization techniques in order to find the time-optimal trajectories describing the desired motions.

For the considered robotic system containing a 6-axis robot, 1-axis positioner and a 1-axis linear track/gantry (optional), there are one or two redundant variables with respect to the given lay-up task that is defined as a sequence of the robot end-effector locations in 6-dimensional space. Let us consider these cases separately.

3.1.1 Search Space Discretization for One Redundant Variable

It is clear that for the one redundant variable case, any of the workcell joint coordinates can be treated as the redundant one, but it is more convenient to consider the positioner joint angle q_P as the redundant variable. This allows us, after solving the positioner direct kinematics for any given q_P, to apply straightforwardly the manipulator inverse kinematic function $g_R^{-1}({}^{Rbase}\mathbf{T}_{Rtool}, \boldsymbol{\mu})$ and find corresponding joint coordinate vectors $\mathbf{q}_R$ for the robot (which is not unique because of multiplicity of possible configurations defined by the parameter $\boldsymbol{\mu}$, see Section 2.2.1). This approach permits to take into account explicitly the equality constraints presented in (68) and substantially reduce the search space dimension.

To present the problem in a discrete way, let us sample the allowable domain of the redundant variable $q_P \in [q_P^{\min}, q_P^{\max}]$ with the step Δq_P

$$\left\{ q_P^{(k)} = q_P^{\min} + \Delta q_P \cdot (k-1) \;\middle|\; k = 1,2,\ldots m \right\} \tag{92}$$

where $m = (q_P^{\max} - q_P^{\min})/\Delta q_P + 1$. Then, applying sequentially the positioner direct kinematics and the manipulator inverse kinematics in accordance with (56) and (28), one can get a set of possible configuration states for the robotic system. For the full mapping from the task space to joint space, let us also take into account the configuration index $\boldsymbol{\mu}$ that corresponds to the manipulator posture, and specifies the shoulder, elbow and wrist configurations. Using these notations and assumptions, the set of the manipulator configurations corresponding to the given task and sampled redundant variable $q_P^{(k)}, k = 1,2,\ldots m$ can be presented as follows:

$$\left\{ \mathbf{q}_R^{(k)}(t_i) = g_R^{-1}\left(\left[{}^{World}\mathbf{T}_{Rbase} \right]^{-1} \cdot {}^{World}\mathbf{T}_{Pbase} \cdot g_P(q_P^{(k)}(t_i)) \cdot {}^{W}\mathbf{T}_{task}^{(i)} \cdot \left[{}^{Rtool}\mathbf{T}_{task} \right]^{1}, \boldsymbol{\mu} \right) \;\middle|\; k = 1,2,\ldots m;\; i = 1,2,\ldots n \right\} \tag{93}$$

where t_i specifies the unknown time instant corresponding to the task location ${}^{W}\mathbf{T}_{task}^{(i)}$ and the functions $g_P(.)$ and $g_R^{-1}(.)$ denote the positioner direct kinematics and the manipulator inverse kinematics respectively. Therefore, for each task location we can generate a number of configuration states $\mathbf{L}_{task}^{(k,i)}$

$$\left\{ {}^{W}\mathbf{T}_{task}^{(i)} \;\middle|\; i = 1,2,\ldots n \right\} \;\rightarrow\; \left\{ \mathbf{L}_{task}^{(k,i)} \;\middle|\; k = 1,2,\ldots m;\; i = 1,2,\ldots n \right\} \tag{94}$$

where $\mathbf{L}_{task}^{(k,i)}=\left(\mathbf{q}_R^{(k)}(t_i),q_P^{(k)}(t_i)\right)$ will be further referred to as the task location cell. This procedure is illustrated in Figure 39.

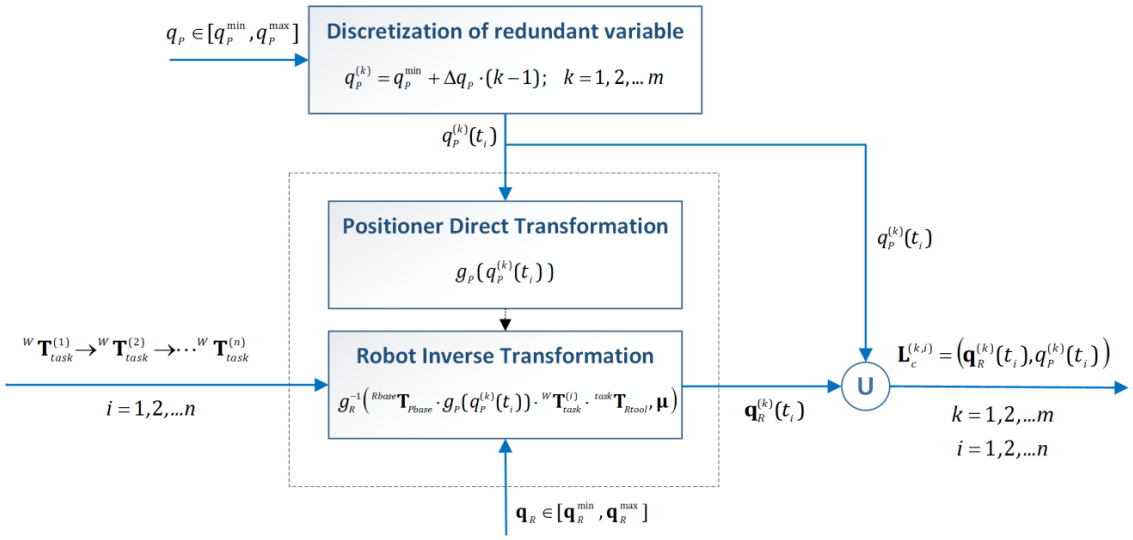


Figure 39 Transformation of the motion planning problem into discrete form (case of one redundant variable)

Taking into account that the task locations are strictly ordered in time, the original sequence of ${}^W\mathbf{T}_{task}^{(i)}$ described in (18) may be converted into the directed graph presented in Figure 40. It should be noted that some of the configurations generated using (93) must be excluded from the graph because of the collision constraints violation or exceeding of the actuator joint limits (69). Besides, from an engineering point of view, it is prudent to avoid some configurations that are very close to the manipulator singular postures. These cases correspond to the "inadmissible nodes" in the graph (see Figure 40), which are not connected to any of the neighbors. It is clear that due to time-irreversibility, the allowable connections between the graph nodes are limited to the subsequent configuration states $\mathbf{L}_{task}^{(k',i)} \rightarrow \mathbf{L}_{task}^{(k'',i+1)}$ and the edge weights correspond to the minimum travelling time between relevant configurations that are restricted by the maximum actuator velocities and accelerations expressed by the constraints (70) and (71).

To generate the above graph, it can be applied specially developed algorithm presented below. It is described in generic programming language, but it has been tested using the MATLAB and C++ environment. The algorithm input is a sequence of 4×4 location matrices $Task(i)$ corresponding to ${}^W\mathbf{T}_{task}^{(i)}$. The upper and lower limits of the redundant variable are defined as $q_P^{\max}$ and $q_P^{\min}$ respectively. The discretization density m determines the number of the discrete values of the redundant variable. The algorithm operates with the positioner direct kinematic function $g_P(.)$ and the robot inverse kinematic function to transform the task locations $Task(i)$ into the joint space. The procedure is composed of two basic steps. At the first step, the redundant variable is discretized in the interval $[q_P^{\min}, q_P^{\max}]$ by implementing formula (92), and $m \times n$ matrix $\{q_P(k,i) \,|\, i=1,2,\ldots n;\ k=1,2,\ldots m\}$ is obtained. At the second step, the functions $g_P(.)$ and $g_R^{-1}(.)$ are applied sequentially, and the robot configuration states corresponding to $\{q_P(k,i) \,|\, \forall i; \forall k\}$ are computed. It should be mentioned that in the algorithm description the configuration index $\boldsymbol{\mu}$ is treated as a constant input parameter, while in practice the step (2) should be re-executed for each possible $\boldsymbol{\mu}$. The sub-step (2a) checks the solution of the robot inverse kinematics for each $\{\mathbf{q}_R(k,i) \,|\, \forall i; \forall k\}$ and applies the collision test function for each configuration state $\{q_P(k,i), \mathbf{q}_R(k,i) \,|\, \forall i; \forall k\}$. Finally, the task graph is generated with nodes $L(k,i) = \left(q_P(k,i), \mathbf{q}_R(k,i)\right)$. If a collision is detected or the function $g_R^{-1}(.)$ returns the "*null*" denoting

that the robot inverse kinematics is not solvable (because of the joint limits violation, etc.), the "*null*" value is assigned to the current node (it is marked as "inadmissible" one on Figure 40). The "*null*" value is assigned if the robot inverse kinematic solution does not exist or the configuration is too close to one of the singularities.

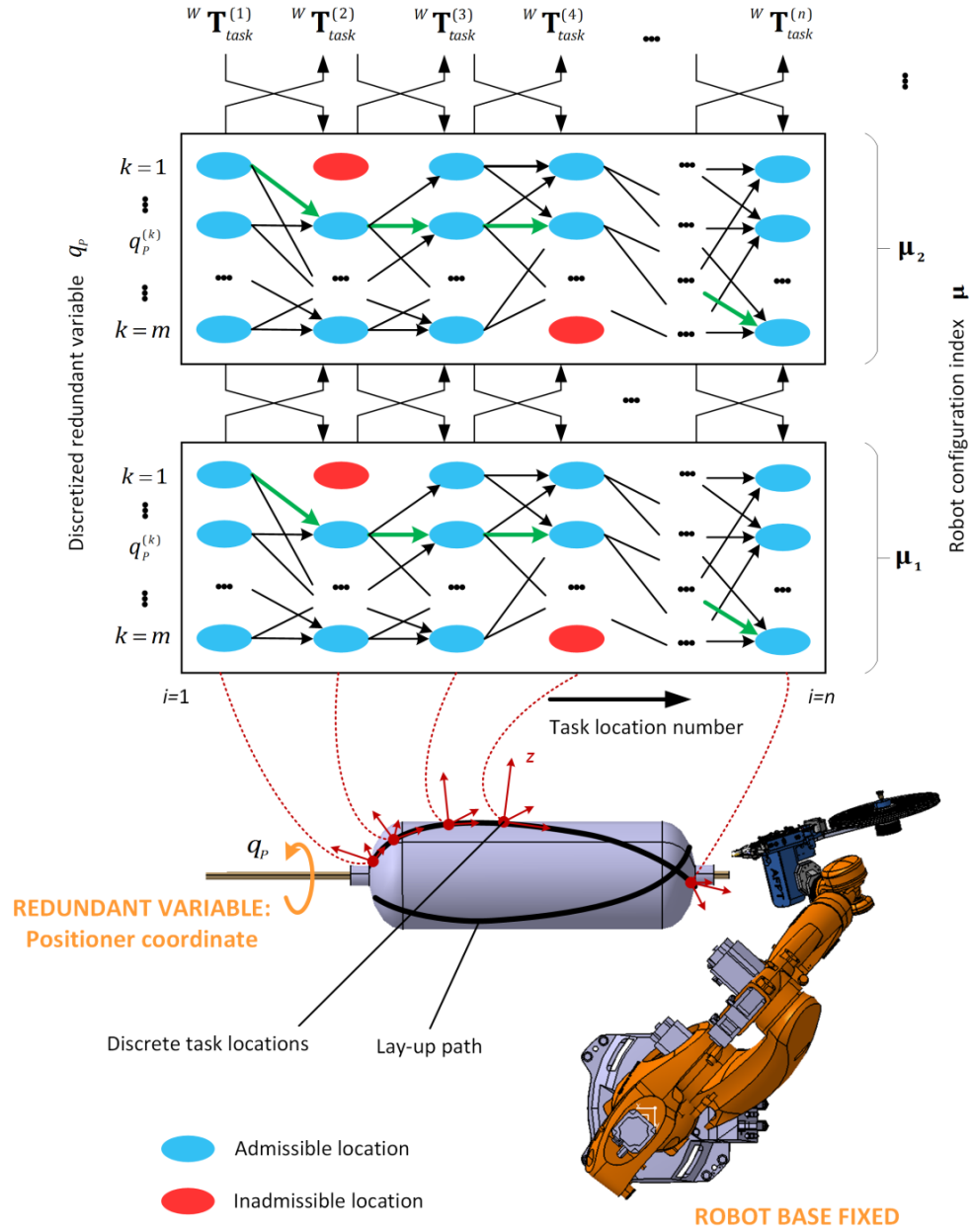


Figure 40 Directed graph describing the motion planning problem with one redundant variable

Using this presentation of the search space, the considered problem can be transformed to the shortest path searching on the directed graph shown in Figure 40, where each column corresponds to the same task location ${}^{W}\mathbf{T}_{task}^{(i)}$ and each row represents the same value of the positioner joint coordinate $q_P^{(k)}$. In accordance with the physical sense, the initial and final path nodes must belong to the sets $\{\mathbf{L}_{task}^{(k_1,1)}, \forall k_1\}$ and $\{\mathbf{L}_{task}^{(k_n,n)}, \forall k_n\}$ respectively. With this notation, the desired solution can be represented as the sequence of the nodes

$$\{\mathbf{L}_{task}^{(k_1,1)}\} \rightarrow \{\mathbf{L}_{task}^{(k_2,2)}\} \rightarrow \cdots \rightarrow \{\mathbf{L}_{task}^{(k_n,n)}\} \tag{95}$$

corresponding to the robot and positioner configuration states $\left(\mathbf{q}_R^{(k)}(t_i), q_P^{(k)}(t_i)\right)$, which ensures the shortest travelling time under the velocity/acceleration constraints of the actuator.

Algorithm 1a: ***Search space generation*** **$G_1(.)$**

Input**:**	Task location matrices $\{Task(i)\mid i=1,2,...n\}$
	Upper limits of redundant variable $\{q_P^{\max}(i)\mid i=1,2,...n\}$
	Lower limits of redundant variable $\{q_P^{\min}(i)\mid i=1,2,...n\}$
	Discretization density m
	Robot configuration index $\boldsymbol{\mu}$
Output**:**	Matrix of locations $\{L(k,i)\mid k=1,2,...m; i=1,2,...n\}$
Notations**:**	q_P, $\mathbf{q}_R$ - positioner and robot joint coordinates
	T_P, T_R - Matrices of robot tool and positioner flange in local frames
Invoked functions**:**	Robot inverse kinematics $g_R^{-1}(.)$ in local frame
	Positioner direct kinematics $g_P(.)$ in local frame
	Transformation from robot base to positioner base $Trans(.)$
	Collision test function $cols(.)$
	Condition number calculation $cond(.)$

(1) *Redundant variable discretization*

For $i=1$ to n **do**

$\Delta q_P(i) := (q_P^{\max}(i) - q_P^{\min}(i))/(m-1);$

For $k=1$ to m **do**

$q_P(k,i) := q_P^{\min}(i) + (k-1)\cdot\Delta q_P(i);$

(2) *Location matrix creation*

For $i=1$ to n **do**

For $k=1$ to m **do**

(a) $T_P := g_P(q_P(k,i))\cdot Task(i)$;

$T_R := Trans(T_P)$;

$\mathbf{q}_R(k,i) := g_R^{-1}(T_R, \boldsymbol{\mu})$;

(b) **If** $(\mathbf{q}_R(k,i) = null) \| (cols(q_P(k,i), \mathbf{q}_R(k,i)) = 1) \| (cond(\mathbf{q}_R(k,i)) > C_{\max})$

$L(k,i) := null;$

else

$L(k,i) := \{q_P(k,i), \mathbf{q}_R(k,i)\};$

In accordance with the actuator constraints, the distance between subsequent graph nodes can be evaluated as the displacement time for the slowest joint

$$dist\left(\mathbf{L}_{task}^{(k_i,i)}, \mathbf{L}_{task}^{(k_{i+1},i+1)}\right) = \max_{j=0,...6}\left(\frac{\left|q_{j,i}^{(k_i)} - q_{j,i+1}^{(k_{i+1})}\right|}{\dot{q}_j^{\max}}\right) \quad (96)$$

where $j=0$ corresponds to the positioner joint variable q_P and $j=1,2,...6$ corresponds to the robot joint coordinates. The latter allows us to present the objective function (total motion time) as the sum of the edge weights

$$T_{total} = \sum_{i=1}^{n-1} dist\left(\mathbf{L}_{task}^{(k_i,i)}, \mathbf{L}_{task}^{(k_{i+1},i+1)}\right) \tag{97}$$

that depends on the indices $k_1, k_2, \ldots k_n$.

To justify the above expressions, let us consider in details the motion commands implemented in typical robot controllers. Usually, they are based on the trapezoidal velocity profiles and coordinated actuation of all joints, as shown in Figure 41. For the point-to-point motions, these profiles are generated to ensure the minimum time movement between two configurations using full capacities of actuators. Corresponding parameters of trapezoid T_i and τ_i are computed using expressions

$$T_i = \Delta q_i / \dot{q}_i^{\max}; \qquad \tau_i = \dot{q}_i^{\max} / \ddot{q}_i^{\max} \tag{98}$$

where Δq_i is the required joint displacement, $\dot{q}_i^{\max}$ and $\ddot{q}_i^{\max}$ are the maximum joint velocity and acceleration respectively. If the displacement is small enough (it is detected using the inequality $T_i < \tau_i$), the trapezoid is reduced to a triangle that is also described by the same parameters T_i and τ_i but they are calculated as

$$\tau_i = \sqrt{\Delta q_i / \ddot{q}_i^{\max}}; \qquad T_i = \tau_i \tag{99}$$

It is clear that in general case, the minimum displacement times may be different for different manipulator axes. So, to synchronize the movements of all joints, the robot controller adapts the desired motion to the slowest actuator, i.e. it selects the longest trapezoid and scales the remaining ones to ensure equal displacement time for all of them. This procedure is illustrated in Figure 41 where the second trapezoid is selected as the reference and the others are scaled ensuring the required joint displacement.

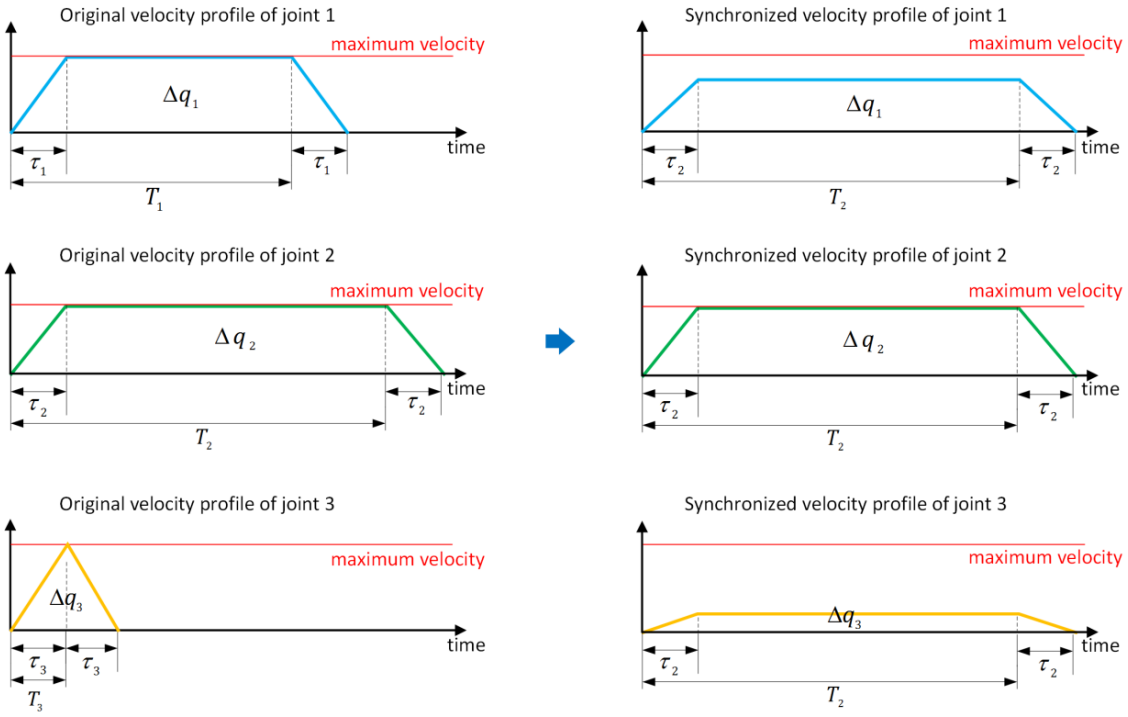


Figure 41 Motion synchronization in a typical robot controller

Using the longest trapezoid parameters $T=\max\{T_1,T_2,...\}$ and $\tau=\max\{\tau_1,\tau_2,...\}$, the synchronized motion of all joint axes is expressed as follows

$$q_i(t)=q_i^{start}+u(t)\cdot(q_i^{target}-q_i^{start}) \qquad \forall i=1,2,...N \tag{100}$$

where $u(t)\in[0,1]$ is the normalized timing law function defined on the interval $t\in[0,T+\tau]$, such that $u(0)=0$, $u(T+\tau)=1$ and $\dot{u}(0)=\dot{u}(T+\tau)=0$. In the trapezoid case, this function is generated as

$$u(t)=\begin{cases} at^2/2, & t\in[0,\tau] \\ u(\tau)+v(t-\tau), & t\in[\tau,T] \\ u(T)+v(t-T)-a(t-T)^2/2, & t\in[T,T+\tau] \end{cases} \tag{101}$$

where $v=1/T$ and $a=1/(T\tau)$. In triangle case when $T=\tau$, the above expression should be written as follows

$$u(t)=\begin{cases} at^2/2, & t\in[0,\tau] \\ u(\tau)+v(t-\tau)-a(t-\tau)^2/2, & t\in[\tau,2\tau] \end{cases} \tag{102}$$

Therefore, for a single point-to-point displacement with zero velocities at the beginning and the end (so-called start-stop mode), the robot controller generates trajectories with the motion time $T+\tau$. However for the considered task that includes multiple point-to-point segments, it is reasonable to utilize another functionality provided by most of the robot controllers that is usually called the "continuous motion" mode (or "approximated positioning" in KUKA robot controllers). It is based on the superposition of the velocity profiles for the consecutive segments where the acceleration phase of the current segment is combined with the deceleration phase of the previous one, as shown in Figure 42. In this case, the motion time for each segment is equal to T which justify expressions (96) and (97) that are used above in the objective function to be minimized in the motion planning for the considered lay-up task. It should be also mentioned that in practice, it is reasonable to avoid triangular profiles that appear if the sampling of the lay-up path is too fine. For such profiles, the motion time should be computed using equations (99) that are not in agreement with (96) and (97) but here it is not possible to use full capacities of the actuators (such as the maximum speed), so the total motion time is obviously over the minimum value limited by acceleration/velocity constraints.

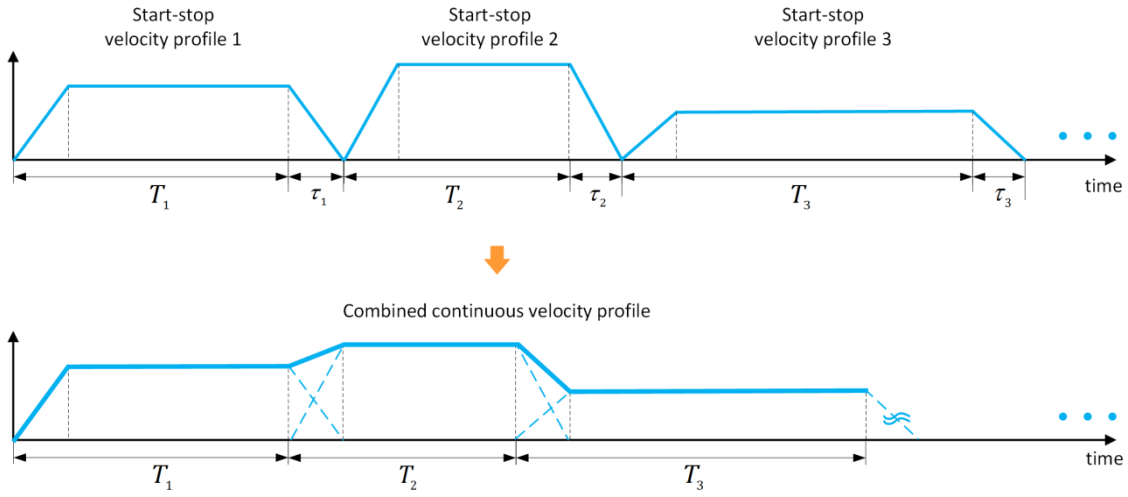


Figure 42 Superposition of the velocity profiles for continuous motions

It should be also noted that the applied method of edge weights computing for the directed graph presented in Figure 40 takes into account the velocity constraints, but the acceleration constraints must be examined for each considered path separately. This test can be implemented by applying a formula, which is based on the second order approximation of the corresponding functions $\mathbf{q}_R(t)$ and $q_P(t)$ on the time interval $t \in [t_{i-1},\ t_{i+1}]$. To derive this formula, let us consider a scalar function $q(t)$ defined at three sequential time instants t_{i-1}, t_i, t_{i+1} by its values q_{i-1}, q_i, q_{i+1} and approximate its acceleration by the second order polynomial $q(t) = f_i + f_i'(t-t_i) + \frac{1}{2} f_i''(t-t_i)^2$. Then, after solution of equations $q(t_{i-1}) = q_{i-1}$, $q(t_i) = q_i$ and $q(t_{i+1}) = q_{i+1}$ with respect to the coefficients f_i, f_i' and f_i'' one can obtain the following expression for the acceleration $f_i'' = 2\left((q_{i+1} - q_i)\Delta t_i - (q_i - q_{i-1})\Delta t_{i+1}\right) / \Delta t_{i+1}\Delta t_i (\Delta t_{i+1} + \Delta t_i)$ where $\Delta t_i = t_i - t_{i-1}$ and $\Delta t_{i+1} = t_{i+1} - t_i$. The latter allows us to present the acceleration constraints on the desired trajectory of the considered robotic system in the following form

$$\frac{2\left|\Delta t_i \left(q_{j,i+1}^{(k_{i+1})} - q_{j,i}^{(k_i)}\right) - \Delta t_{i+1}\left(q_{j,i}^{(k_i)} - q_{j,i-1}^{(k_{i-1})}\right)\right|}{\Delta t_{i+1}\Delta t_i \left(\Delta t_{i+1} + \Delta t_i\right)} \le \ddot{q}_j^{\max} \tag{103}$$

and where Δt_i and Δt_{i+1} correspond to the edge weights of the above presented graph which were defined as the traveling times between successive task locations, i.e. $\Delta t_i = dist\left(\mathbf{L}_{task}^{(k_i,i)}, \mathbf{L}_{task}^{(k_{i-1},i-1)}\right)$ and $\Delta t_{i+1} = dist\left(\mathbf{L}_{task}^{(k_i,i)}, \mathbf{L}_{task}^{(k_{i+1},i+1)}\right)$.

3.1.2 Search Space Discretization with Two Redundant Variables

For the two redundant variables case, where the linear track/gantry is activated together with the positioner, the above presented methodology can be generalized in the following way. Similarly to the previous case, let us treat the robot joint angles $\mathbf{q}_R$ as non-redundant variables and the positioner joint angle $q_P \in [q_P^{\min}, q_P^{\max}]$ as well as the linear track/gantry coordinate $q_L \in [q_L^{\min}, q_L^{\max}]$ as the redundant ones. Their sampling with the steps Δq_P and Δq_L respectively produces the following set of the redundant coordinates

$$\left\{ q_P^{(k_1)} = q_P^{\min} + \Delta q_P \cdot (k_1 - 1) \,\middle|\, k_1 = 1, 2, \ldots m_1 \right\} \ \times \ \left\{ q_L^{(k_2)} = q_L^{\min} + \Delta q_L \cdot (k_2 - 1) \,\middle|\, k_2 = 1, 2, \ldots m_2 \right\} \tag{104}$$

where $m_1 = (q_P^{\max} - q_P^{\min}) / \Delta q_P + 1$ and $m_2 = (q_L^{\max} - q_L^{\min}) / \Delta q_L + 1$. Then, applying sequentially the direct kinematics for the positioner and linear track/gantry as well as the manipulator inverse kinematics in accordance with (56), (59) and (28), one can get a set of possible configuration states for the robotic system. This set of the manipulator configurations corresponding to the given task and sampled redundant variables can be expressed as follows

$$\left\{ \begin{array}{c} \mathbf{q}_R^{(k_1,k_2)}(t_i) = g_R^{-1}\left(\left[{}^{World}\mathbf{T}_{Lbase} \cdot g_L(q_L^{(k_2)}(t_i))\right]^{-1} \cdot {}^{World}\mathbf{T}_{Pbase} \cdot g_P(q_P^{(k_1)}(t_i)) \cdot {}^{W}\mathbf{T}_{task}^{(i)} \cdot \left[{}^{Rtool}\mathbf{T}_{task}\right]^{-1}, \boldsymbol{\mu}\right) \\ i = 1, 2, \ldots n; \qquad (k_1, k_2) \in [1, m_1] \times [1, m_2] \end{array} \right\} \tag{105}$$

where the functions $g_L(.)$, $g_P(.)$ and $g_R^{-1}(.)$ denote the direct kinematics of the linear track/gantry, direct kinematics of the positioner and the inverse kinematics of the manipulator respectively. Therefore, for each task location we can generate a number of configuration states $\mathbf{L}_{task}^{(k_1,k_2,i)}$

$$\left\{ {}^{W}\mathbf{T}_{task}^{(i)} \,\middle|\, i = 1, 2, \ldots n \right\} \ \rightarrow \ \left\{ \mathbf{L}_{task}^{(k_1,k_2,i)} \,\middle|\, k_1 = 1, 2, \ldots m_1;\ k_2 = 1, 2, \ldots m_2;\ \ i = 1, 2, \ldots n \right\} \tag{106}$$

where the arrays $\mathbf{L}_{task}^{(k_1,k_2,i)}=\left(\mathbf{q}_R^{(k_1,k_2)}(t_i),q_P^{(k_1)}(t_i),q_L^{(k_2)}(t_i)\right)$ will be further referred to as the task location cells. This procedure is illustrated in Figure 42.

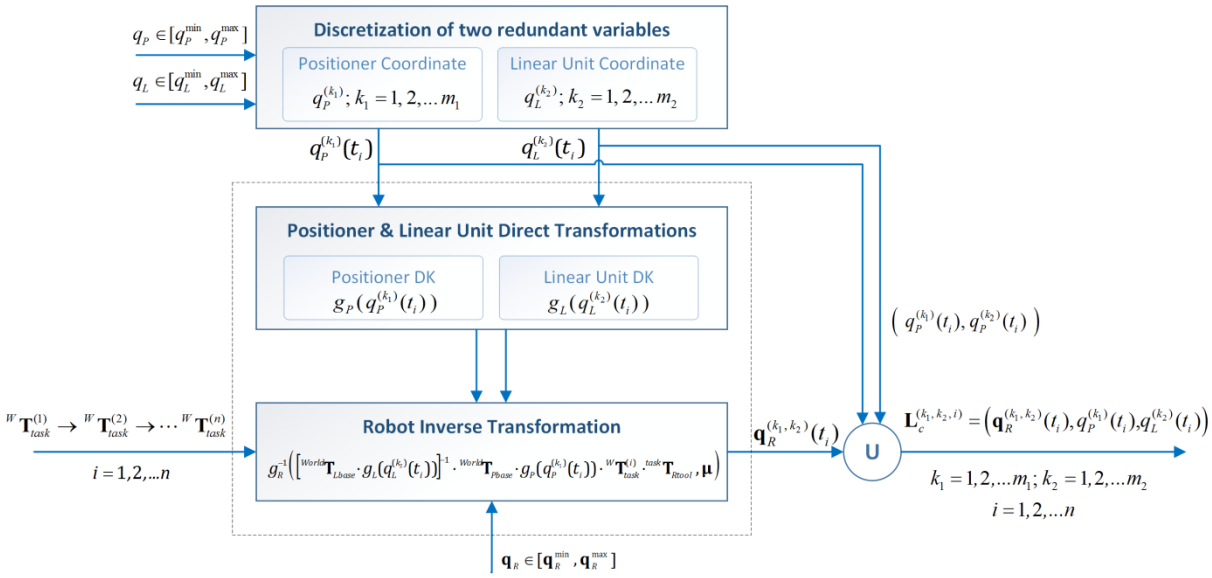


Figure 43 Transformation of the motion planning problem into discrete form (case of two redundant variables)

Using similar idea as above, the original sequence of the task locations ${}^{W}\mathbf{T}_{task}^{(i)}$ may be converted into the 3D directed graph shown in Figure 43. In contrast to the one redundant variable case where each task location corresponds to a column (see Figure 40), here each Cartesian task location is presented by a set of nodes located on the plane $(k_1,k_2)\in[1,m_1]\times[1,m_2]$. In this graph, the desired motion of the workcell components can be presented as a shortest path

$$\{\mathbf{L}_{task}^{(k_1^1,k_2^1,1)}\}\rightarrow\{\mathbf{L}_{task}^{(k_1^2,k_2^2,2)}\}\rightarrow\cdots\rightarrow\{\mathbf{L}_{task}^{(k_1^n,k_2^n,n)}\} \tag{107}$$

that sequentially connects the nodes belonging to the neighbor planes (exactly one node for each plane). For this optimization problem, the objective function, the edge weights, and all constraints are similar to ones corresponding to the case of one redundant variable.

To generate the 3D graph, it can be applied specially developed algorithm (Algorithm 1b) presented below. Its input includes is a sequence of 4×4 location matrices $Task(i)$, the discretization densities m_1 and m_2, and the upper and lower limits of the redundant variables denoted as $q_P^{\max}$, $q_P^{\min}$ and $q_L^{\max}$, $q_L^{\min}$ respectively. The algorithm transforms the task locations $Task(i)$ into the joint space by operating the functions of robot inverse kinematic $g_R^{-1}(.)$, positioner direct kinematic $g_P(.)$ and linear unit direct kinematic $g_L(.)$. This procedure also contains two steps. Firstly, the redundant variables are discretized in the intervals $[q_P^{\min},q_P^{\max}]$ and $[q_L^{\min},q_L^{\max}]$ by implementing (99), and $m_1\times m_2\times n$ matrix $\{q_P(k_1,k_2,i),q_L(k_1,k_2,i)\,|\,i=1,2,...n;\ k_1=1,2,...m_1;\ k_2=1,2,...m_2\}$ is obtained. Then, at the second step, $g_P(.)$, $g_L(.)$ and $g_R^{-1}(.)$ are applied sequentially, and the robot configuration states corresponding to $\{q_P(k_1,k_2,i),q_L(k_1,k_2,i)\,|\,\forall i;\ \forall k_1;\forall k_2\}$ are computed. After checking with the inverse kinematic solvability, collision test and the distance to singularities, the task graph is finally generated with nodes $L(k_1,k_2,i)=\left(q_P(k_1,k_2,i),q_L(k_1,k_2,i),\mathbf{q}_R(k_1,k_2,i)\right)$.

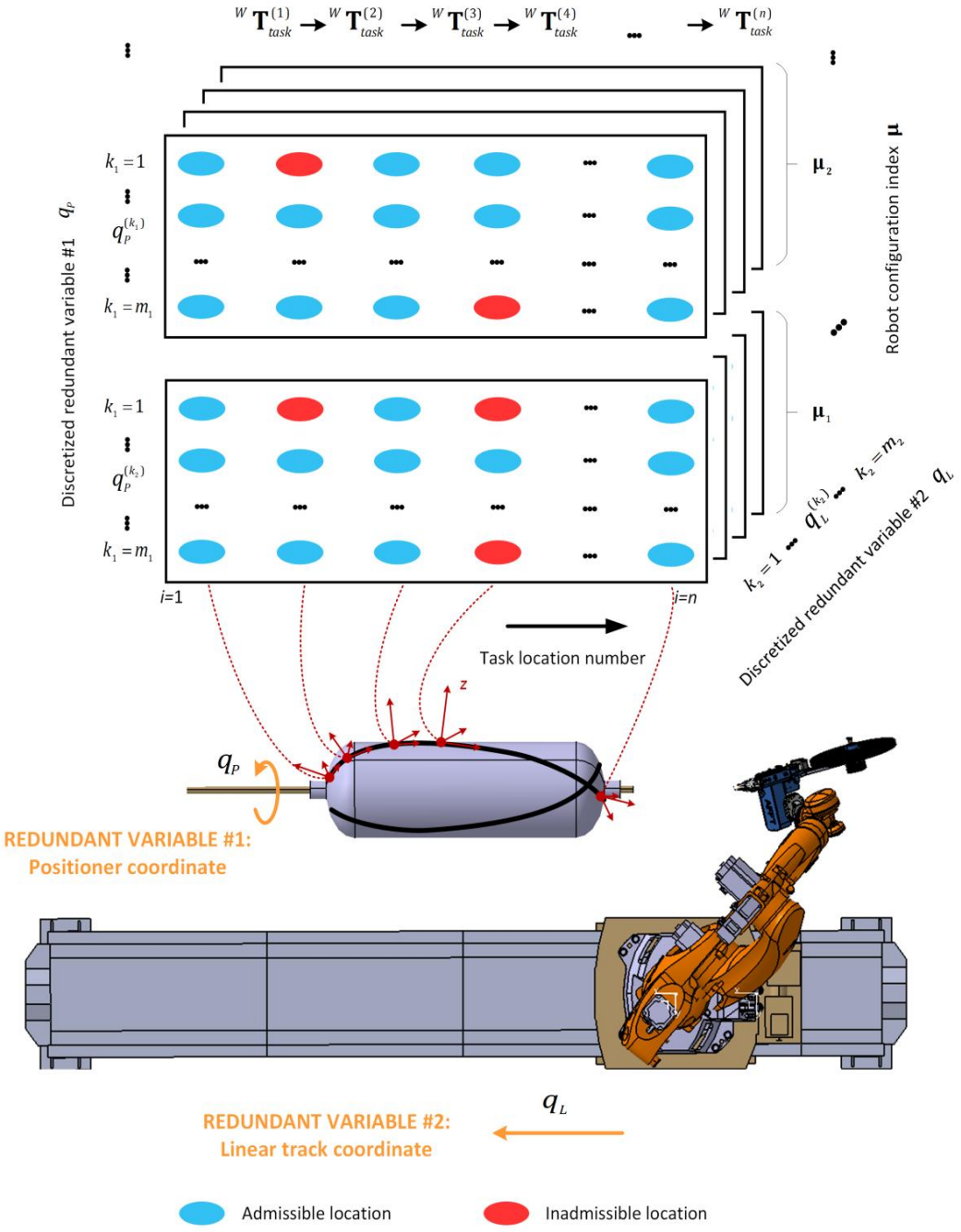


Figure 44 Directed graph describing the motion planning problem with two redundant variables

Algorithm 1b: ***Search space generation*** **$G_2(.)$**

***Input*:**	Task location matrices $\{Task(i)\mid i=1,2,...n\}$
	Upper limits of positioner coordinate $\{q_P^{\max}(i)\mid i=1,2,...n\}$
	Lower limits of positioner coordinate $\{q_P^{\min}(i)\mid i=1,2,...n\}$
	Upper limits of linear track coordinate $\{q_L^{\max}(i)\mid i=1,2,...n\}$
	Lower limits of linear track coordinate $\{q_L^{\min}(i)\mid i=1,2,...n\}$
	Discretization density for positioner coordinate m_1
	Discretization density for linear track coordinate m_2
	Robot configuration index $\boldsymbol{\mu}$
***Output*:**	Matrix of locations $\{L(k_1,k_2,i)\mid k_1=1,2,...m_1; k_2=1,2,...m_2; i=1,2,...n\}$
***Notations*:**	q_L , q_P, $\mathbf{q}_R$ - Robot location, positioner and robot joint coordinates
	T_P, T_R - Matrices of robot tool and positioner flange in local frames
***Invoked functions*:**	Robot inverse kinematics $g_R^{-1}(.)$ in local frame
	Positioner direct kinematics $g_P(.)$ in local frame
	Transformation from robot base to positioner base $Trans(.)$
	Collision test function $cols(.)$
	Condition number calculation $cond(.)$

(1) *Positioner and linear unit coordinates discretization*

For $i=1$ to n **do**

$\Delta q_P(i) := (q_P^{\max}(i) - q_P^{\min}(i))/(n-1);$

$\Delta q_L(i) := (q_L^{\max}(i) - q_L^{\min}(i))/(n-1);$

For $k_1=1$ to m_1 **do**

For $k_2=1$ to m_2 **do**

$q_P(k_1,k_2,i) := q_P^{\min}(i) + (k_1-1)\cdot\Delta q_P(i);$

$q_L(k_1,k_2,i) := q_L^{\min}(i) + (k_2-1)\cdot\Delta q_L(i);$

(2) *Location matrix creation*

For $i=1$ to n **do**

For $k_1=1$ to m_1 **do**

For $k_2=1$ to m_2 **do**

(a) $\mathbf{T}_{task}^P := g_P(q_P(k_1,k_2,i))\cdot Task(i)$;

$\mathbf{T}_{task}^R := Trans(q_L(k_1,k_2,i))\cdot\mathbf{T}_{task}^P;$

$\mathbf{q}_R(k_1,k_2,i) := g_R^{-1}(\mathbf{T}_{task}^R, \boldsymbol{\mu});$

(b) **If** $(\mathbf{q}_R(k_1,k_2,i) = null) \,\|\, (cols(q_P(k_1,k_2,i), q_L(k_1,k_2,i), \mathbf{q}_R(k_1,k_2,i)) = 1) \,\|\, (cond(\mathbf{q}_R(k_1,k_2,i)) > C_{\max})$

$L(k_1,k_2,i) := null;$

else

$L(k_1,k_2,i) := \{q_P(k_1,k_2,i), q_L(k_1,k_2,i), \mathbf{q}_R(k_1,k_2,i)\};$

Therefore, using sampling of the redundant variables, the original continuous problem is transformed into the discrete form that already incorporates the collision and singularity constraints. It allows us to apply further some combinatorial optimization techniques that should be modified to take into account the remaining velocity/acceleration constraints describing capacities of the actuators. This issue is in the focus of the next section.

3.2 MOTION PLANNING METHOD BASED ON COMBINATORIAL OPTIMIZATION

Using the graph generation approach presented in Section 3.1, the original optimization problem in function space (91) is converted into a discrete one, which is aimed at searching of the shortest path connecting the subsets of the nodes corresponding to the given task locations. These subsets are grouped in columns for the one redundant variable and in planes for the two redundant variables. The path should visit exactly one node in the subsets for each task location.

3.2.1 Straightforward Approaches and Related Difficulties

The classical shortest path problem is to find a path between 2 vertices in an undirected/directed graph such that the total sum of the edges weights is the minimum. It is clear that the generated graph for the considered problem is a directed one because of the time irreversibility. In the simplest case, when all edge weights were equal, this problem could be solved easily using the *Breadth-first search.* However, for the considered problem the equal-weight assumption is not valid because the robot travelling times between subsequent task locations are usually different. So, the considered problem can be categorized as finding the shortest path in a directed graph with non-equal positive weights.

In literature, there are several approaches that address the general shortest path problem in a directed graph. One of the most common of them is the *Dijkstra*'s algorithm (Dijkstra, 1959). It can be used for finding the shortest paths from a single departure vertex to a single destination vertex. To apply the *Dijkstra*'s algorithm to the considered engineering problem, it is necessary to create two additional virtual vertices to the generated graph at the beginning and the end as shown in Figure 43 where the weights of the new edges are assumed to be equal to each other (Gueta et al., 2009b, Gueta et al., 2008b, Dolgui and Pashkevich, 2006). Using such technique, the desired solution is presented as the shortest path starting from the virtual beginning vertex to the virtual end one. It is clear that for the considered graph the shortest path should pass through exactly one vertex in each column.

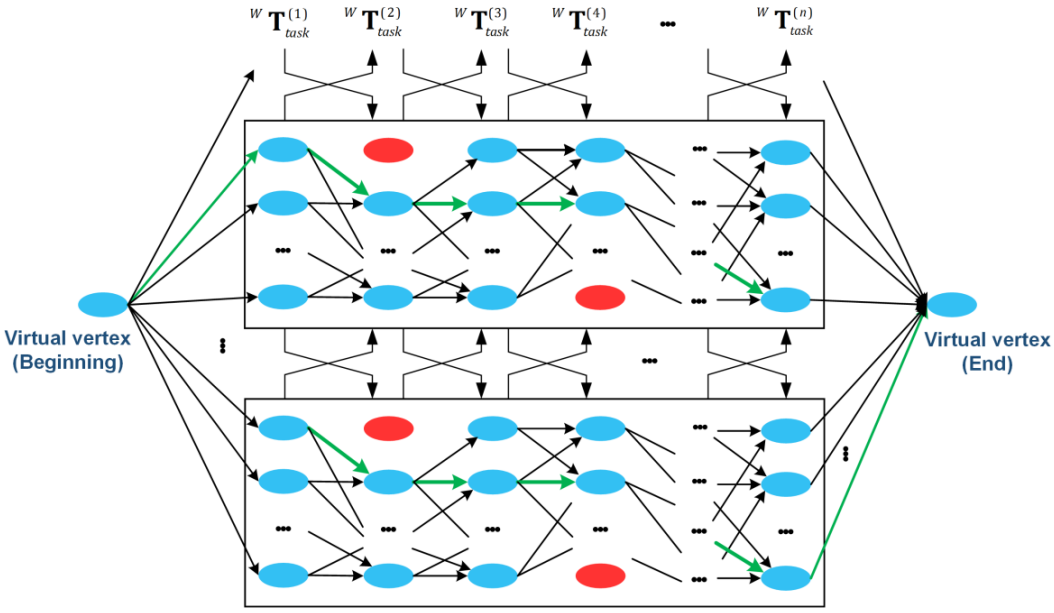


Figure 45 Directed graph with two virtual vertices for the motion planning problem

Using the *Dijkstra*'s algorithm, it is possible to compute the shortest distance (robot travelling time here) between the initial vertex and any other vertices in a graph. To implement this algorithm, it is initially assumed that a tentative distance value is assigned to every vertex (zero for the initial one and to infinity for the others). First, the initial vertex is considered as the current one, while others are marked as "unvisited". At the following steps, all neighbors of the current vertex are analyzed and their tentative distances are calculated by summing the current vertex distance and the weight of the edge connecting the current vertex with the corresponding neighbor. Then, the smaller value is selected and considered further as the distance of the current vertex, which is marked as a "visited" one. This procedure is repeated until all the vertices have been visited. Generally, the idea of the *Dijkstra*'s algorithm is to pick the unvisited vertex with the lowest distance, calculate the distance through it to each unvisited neighbor and to exclude longer distances when making an update (Cormen, 2009).

It should be mentioned that the standard *Dijkstra*'s algorithm uses specific structure of the input data, usually in the form "a sparse matrix" (see Matlab function "*graphshortestpath*" for instance). Relevant transformation of the original data describing our engineering problem into the format required for the *Dijkstra*'s algorithm requires non-negligible amount of computing time. For example, even for the simplest workcell consisting of a 2-axis manipulator and a 1-axis positioner considered in this work as a benchmark example, it takes several hours to transform the data of a 100×360 two dimensional directed graph into the desired format (processor Intel® i5 1.70 GHz 2.40 GHz, 8G memory). In contrast, the shortest path computation for this graph using the *Dijkstra*'s algorithm requires several seconds only.

Another known technique that solves the shortest path problem is the *Bellman Ford*'s algorithm (Bellman, 1958, Ford Jr, 1956). Similar to the algorithm of *Dijkstra*, it is also based on the relaxation principle where a tentative distance value is gradually replaced by the minimum of its old value and the length of a newly found path until eventually reaching the optimum solution. However, *Dijkstra*'s algorithm uses a priority queue to select the closest vertex that has not been "visited", and relaxes all of its outgoing edges. By contrast, the *Bellman–Ford*'s algorithm does this relaxation on all the possible edges. In each of these repetitions, the number of vertices with optimum distances grows, and eventually all vertices will have their right distances. However, it brings the difficulties similar to that when applying *Dijkstra*'s algorithm, i.e. too much computational effort for data transformation and preparation.

For the considered engineering problem, motion planning for redundant robotic systems, straightforward application of the known techniques has essential limitations. The first of them has been already mentioned above and is related to the high computational efforts for input data transformation into required structure. The second one comes out when the designer tries to take into account the acceleration constraints expressed by inequalities (71), which are quite important here for technological reasons. In fact, the conventional shortest path algorithms focus on simple minimization of the edge weight sum, while ignoring the smooth motion requirement that comes from robotics and is outside of the classical directed graph with weighted edges.

On the other hand, the data structure describing the discretized motion planning problem includes additional information. In particular, each vertex of the corresponding graph represents the task location containing all joint coordinates of the robotic manipulator, positioner and linear track/gantry that ensure desired positioning of the robot end-effector. These vertex attributes allow us to evaluate

the accelerations of all robotic system components for any considered trajectory, which gives some perspectives for enhancement of the shortest path techniques on the directed graphs adapted to the engineering problem studied in this thesis.

3.2.2 Motion Generation using Dynamic Programming Technique

As follows from the previous section, it is not reasonable to apply conventional shortest path algorithms (*Dijkstra*'s, *Bellman-Ford*'s, etc.) to find optimal motions for the considered robotic system because of high computational expenses and neglecting some technical constraints. On the other side, the directed graph related to the engineering problem studied here has very specific matrix structure where the vertices are naturally grouped in planes or columns and the edges connect the neighboring planes/columns only. Besides, the desired optimal path must include a single vertex from each plane/column starting from the first and ending at the last one. These specific properties of the directed graph motivate us to develop a problem-oriented optimization technique presented below.

Because of inherited matrix structure of the data describing robotic lay-up process and an additive objective, it is reasonable to apply here dynamic programming. Generally, it is a method for solving a complex problem by breaking it down into a set of sub-problems of smaller sizes, solving each of those sub-problems only once, and storing their results (Bellman, 1954). Next time the same sub-problem appears, instead of computing its solution again, one simply indexes the previously computed solution, thereby saving computation time at the expense of a slight expenditure in storage space. A dynamic programming algorithm will examine the previously solved sub-problems and will combine their solutions to give the best solution for the given problem.

To apply the dynamic programming, the problem must possess two key properties (Bertsekas et al., 1995). The first one is called optimal substructure that means the solution to an optimization problem can be obtained by the combination of optimal problems to its sub-problems, and these optimal substructures are able to be formulated in a recursive form. Here, the desired solution of the considered shortest path problem is already represented as the sum of sub-solutions described in (97), which allows us to write it in a type of recursion in future. Secondly, the sub-problems should be overlapping, which means any recursive algorithm solving the problem should solve the same sub-problems repeatedly, rather than generating new sub-problems. In the considered problem, the sub-problems are always the paths from the initial plane/column to the current one, and it is impossible to generating new sub-problems during the recursion.

To present the developed algorithm based on dynamic programming for the case of one redundant variable, let us denote $d_{k,i}$ as the length of the shortest path connecting one of the initial vertices $\{\mathbf{L}_{task}^{(k1,1)}, \forall k_1\}$to the current vertex $\{\mathbf{L}_{task}^{(k,i)}\}$. Then, taking into account the additivity of the objective (97), the shortest path for the nodes corresponding to the next task point $\{\mathbf{L}_{task}^{(k,i+1)}, \forall k\}$ can be found by combining the optimal solutions for the previous column $\{\mathbf{L}_{task}^{(k',i)}, \forall k'\}$ and the distances between the nodes with the indices i and $i+1$.The latter corresponds to the formula

$$d_{k,i+1} = \min_{k'} \left\{ d_{k',i} + dist\left(\mathbf{L}_{task}^{(k,i+1)}, \mathbf{L}_{task}^{(k',i)} \right) \right\} \tag{108}$$

that is applied sequentially starting from the second task point, i.e. for $i = 1,2,\ldots n-1$. At the final step, after selection of the minimum lengths $\{d_{k,n}, \forall k\}$ corresponding to the end task point and applying the

backtracking, one can get the desired optimal path. Therefore, the desired solution for n sets $\{\mathbf{L}_{task}^{(k1,1)}\}\rightarrow\{\mathbf{L}_{task}^{(k2,2)}\}\rightarrow\ldots\{\mathbf{L}_{task}^{(kn,n)}\}$ is obtained by increasing sequentially the i-index, computing $d_{k,i+1}$, and finding the minimum value among $d_{kn,n}$. The desired path is described by the recorded indices $\{k_1, k_2, \ldots k_n\}$.

It is clear that the above technique presented for a single redundant variable can be easily generalized to the case of two redundant variables, where the task graph vertices are grouped in planes instead of columns. The latter requires slight modification of indices leading to presenting each task point in the Cartesian space by the following set of locations in the configuration space $\{\mathbf{L}_{task}^{(k_1,k_2,i)};\ i=const\}$ where $k_1=1,2,\ldots m_1$ and $k_2=1,2,\ldots m_2$. This allows us to present the algorithm basic expression in a similar way

$$d_{k_1,k_2,i+1}=\min_{k_1',k_2'}\left\{d_{k_1',k_2',i}+dist\left(\mathbf{L}_{task}^{(k_1,k_2,i+1)},\mathbf{L}_{task}^{(k_1',k_2',i)}\right)\right\} \tag{109}$$

where $d_{k_1,k_2,i+1}$ denotes the length of the shortest path connecting one of the initial vertices $\{\mathbf{L}_{task}^{(k_1^{(1)},k_2^{(1)},1)},\forall k_1^{(1)},\forall k_2^{(1)}\}$ to the current vertex $\{\mathbf{L}_{task}^{(k_1',k_2',i)}\}$.

In more details, an outline of the developed algorithm for the case of one redundant variable is presented below (see Algorithm 2). Here, the input is the matrix of the locations $\{L(k,i)\mid\ i=1,2,\ldots n;\ k=1,2,\ldots m\}$, which contains information on the configuration states satisfying the equality constraint (68), the collision constraint (86) and the singularity constraint (90). The algorithm operates with two tables $D(k,i)$ and $P(k,i)$ that include the minimum distances for the sub-problem of lower size (for the path $1\rightarrow i$) and the pointers to the previous locations respectively. The procedure is composed of four basic steps. The step (1) initializes the distance and pointer matrices by defining their first columns. In step (2), the recursive formula (108) is implemented. The computing starts from the second column and tries all possible connections between the nodes in the current column $\{L(k,i),\forall k\}$ and the previous one $\{L(j,i-1),\forall j\}$. For the admissible configuration states, the acceleration constraints are examined using the expression (103) for each candidate path connecting the nodes with the indices i, $i-1$ and $i-2$. The acceleration constraint test is included in sub-step (2a) where $accl(\cdot)=0$ indicates that the current path satisfies this constraint, and it begins from the third column. It should be mentioned that the function $accl(.)$ requires three inputs $L(*,i)$, $L(*,i-1)$, $L(*,i-2)$, according to the expression (103), and the location $L(*,i-2)$ is determined using the pointer $P(j,i\text{-}1)$ to the previous location in the current path. Then, sub-step (2b) finds the minimum path from the current node $L(k,i)$ to the first column $\{L(j,1),\forall j\}$ and records the reference to $\{L(j,i-1),\forall j\}$ into the pointer matrix. In steps (3) and (4), the optimal solution is finally obtained and corresponding path is extracted by means of the backtracking.

To demonstrate the efficiency of the proposed methodology, let us apply it to the benchmark example that deals with a redundant robotic system composed of a 2-axis planar manipulator and a 1-axis positioner, which is described in details in Subsection 1.2.1 (see Figure 14). This system possesses a 1-dof redundancy with respect to the ellipsis-type planar lay-up task that was sampled in 100 segments uniformly. The segment nodes should be visited sequentially by the end-effector in minimum time by using in the best way the motion capabilities of both the robot and the positioner, which are described by the constraints presented in Table 12. The developed algorithm was implemented and examined using Matlab in Win7SP1 with Intel® i5 @2.67 GHz 2.67 GHz.

Algorithm 2: ***DP-based Path planning*** **DP(.)**

***Input*:**	Matrix of locations $\{L(k,i)\,\vert\, i=1,2,...n; k=1,2,...m\}$
***Output*:**	Minimum path length $D_{\min}$
	Optimal path indices $\{k^0(i)\,\vert\, i=1,2,...n\}$
***Notations*:**	$D(k,i)$, $P(k,i)$ - distance and pointer $m\times n$ matrices
***Invoked functions*:**	Distance between nodes $dist(L(k_1,i_1),L(k_2,i_2))$
	Acceleration test for nodes $accl\ (L(k,i),L(j,i-1),L(P(j,i\text{-}1),i-2))$

(1) *Initialization for the search space*

Set $D(k,1):=0;\ P(k,1):=null,\ \forall k=1,2,...m$

(2) *Path searching*

For $i=2$ to n **do**

For $k=1$ to m **do**

(a) **For** $j=1$ to m **do**

If $(L(k,i)\neq null)\&(L(j,i-1)\neq null)$

If $(i=2)\,\|\,(accl\ (L(k,i),L(j,i-1),L(P(j,i-1),i-2))=0)$

$r(j):=D(j,i-1)+dist(L(k,i),L(j,i-1))$;

else

$r(j):=inf$;

(b) **Set** $D(k,i):=\min_j(\{r(j)\,\vert\, j=1,2,...m\})$;

$P(k,i):=\arg\min_j(\{r(j)\,\vert\, j=1,2,...m\})$;

(3) *Selection of the shortest path*

Set $D_{\min}:=\min_k(\{D(k,n)\,\vert\, k=1,2,...\ m\})$;

$k^0(n):=\arg\min_k(\{r(j)\,\vert\, j=1,2,...m\})$;

(4) *Backtracking*

For $i=n$ to 2

Set $k^0(i-1):=P(k^0(i),i)$

Table 12 Kinematic constraints of redundant robotic system for the benchmark example

Joint limits	Maximum velocity	Maximum acceleration
$-180°\le q_P\le 180°$	$-50°/s\le \dot{q}_P\le 50°/s$	$-200°/s^2\le \ddot{q}_P\le 200°/s^2$
$-180°\le q_{R1}\le 180°$	$-10°/s\le \dot{q}_{R1}\le 10°/s$	$-40°/s^2\le \ddot{q}_{R1}\le 40°/s^2$
$-180°\le q_{R2}\le 180°$	$-30°/s\le \dot{q}_{R2}\le 30°/s$	$-120°/s^2\le \ddot{q}_{R2}\le 120°/s^2$

As follows from a simulation study, the developed algorithm is able to find the desired time-optimal motions but the computing time is not negligible. In particular, it takes about 10 minutes to find the optimal solution for the simple benchmark problem. It should be stressed that in contrast to the conventional techniques, this algorithm can generate desired trajectories taking into account the acceleration constraints. Nevertheless, since the inequalities (103) are checked for each candidate path during searching, the computation time may be still too high for real life lay-up applications where 6-

axis robots are usually used and the number of the task points is essentially larger. This motivates us to improve the developed algorithm further.

3.3 ENHANCEMENT OF THE MOTION PLANNING ALGORITHM

As follows from the simulation study, the motion planning algorithm presented in the previous section should be enhanced to be applied to real industrial problems. In particular, to obtain the desired time-optimal smooth motions the discretization should be very fine, which causes quite high computational efforts. For this reason, this section presents two strategies allowing finding reasonable balance between the algorithm running time and smoothness of the obtained trajectories. In both of them, the basic idea is to apply the rough discretization first in order to generate an initial solution, which is locally improved further using either smaller discretization step or approximation of the redundant variable profiles by polynomial functions.

3.3.1 First Strategy: Progressive reduction of the discretization step

To reduce the computing time for real industrial problems, let us apply the following heuristic technique: solve the problem applying several times the same optimization routine (Algorithm 2) in different search spaces. It is reasonable to start with a rather big discretization step Δq_p (stage I), and further improve the initial solution in its neighborhood using relatively small Δq_p (stage II). In this technique, in spite of the repetition of the basic optimization routine, both stages operate with the search space of smaller size compared to the straightforward optimization for small Δq_p in the full-range space, which allows us essentially decreasing the amount of the computations. The details of this strategy are presented in Figure 46.

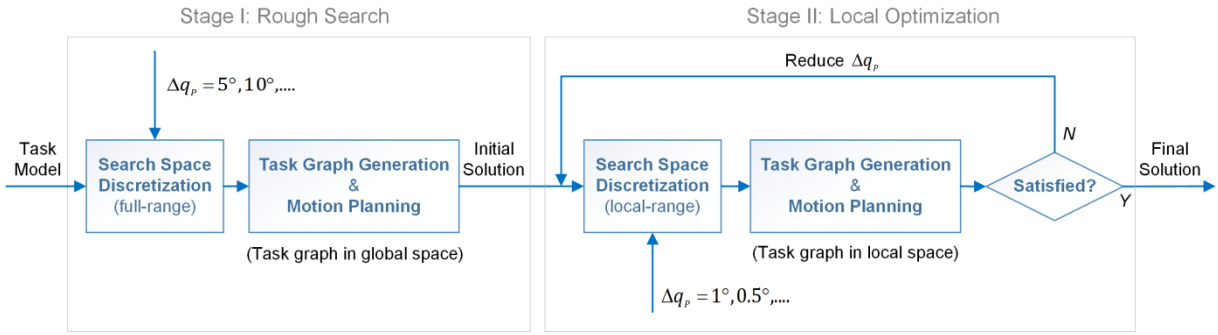


Figure 46 Combination of the rough search and the local optimization

The stage I, named here as the *global rough search*, is based on the full-range discretization of the redundant variable with relatively big increment step (compared to that in the fine search algorithm presented before). By applying the dynamic programming principle expressed by (108), an initial solution can be obtained quite rapidly (it takes about one minute for the benchmark example that deals with the two-axis robot and one-axis positioner, 100 task locations). It should be mentioned that the motion obtained at this stage may be not satisfactory from the engineering point of view since the discretization step is large. So, it is reasonable to optimize the obtained solution further.

At the stage II (named as the *iterative local fine search*), a secondary discretization with smaller step is applied in the neighborhood of the initial solution obtained at the stage I. The neighborhood size is defined by the user (see Figure 47). It is clear that this local discretization produces the search space of relatively small size (compared to straightforward approach) allowing us quickly generate an optimal solution that is obviously better than the initial one. For instance, for the above mentioned benchmark example, it takes about two minutes in total (stages I and II) while the one-stage technique (see subsection 3.2.2) requires more than 10 minutes to obtain a similar result. It should be mentioned that the stage II can be repeated several times, sequentially reducing the discretization step.

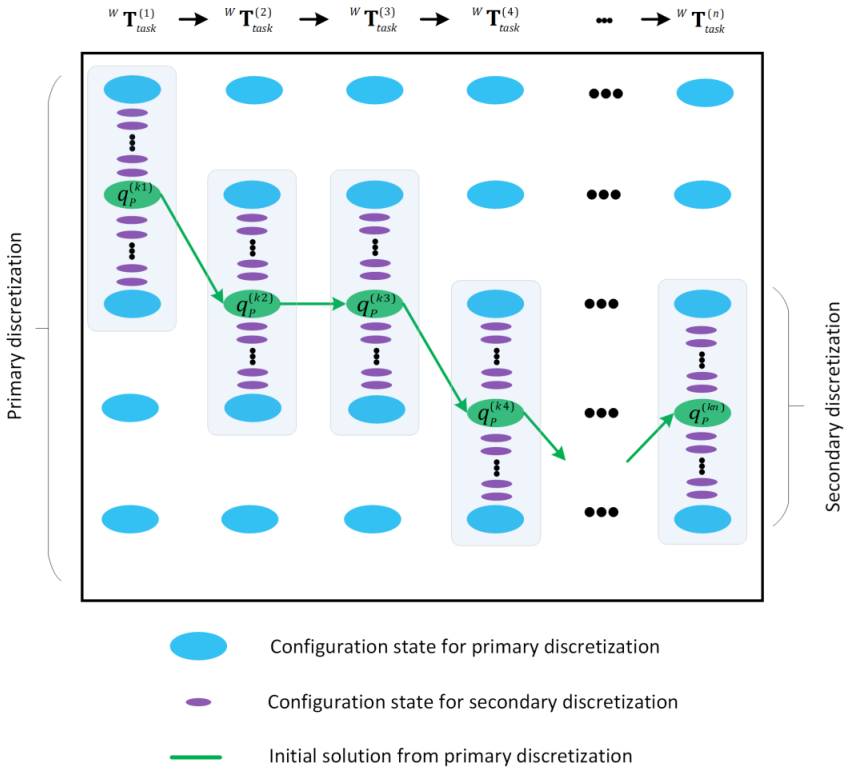


Figure 47 Search space evolution for the two-stage algorithm

An outline of the two-stage algorithm is presented below (see Algorithm 3). The input includes the array of the task location matrices $\{Task(i)\,|\,i=1,2,...n\}$, the boundary limits of the redundant variable $q_P^{\max}$ and $q_P^{\min}$, the neighborhood size Rg_P for the stage II, and the discretization densities m and m' for both stages. The output contains the minimum path length $D'_{\min}$ and optimal path indices $\{k'^0(i)\,|\,i=1,2,...n\}$. At the first stage, the $m\times n$ task graph $\{L(k,i)\,|\,k=1,2,...m;\,i=1,2,...n\}$ is generated using the full-range discretization with the density m (step Ia). The initial solution $\{k^0(i)\,|\,i=1,2,...n\}$ is obtained by applying the path planning function $DP(.)$ based on Algorithm 2 (steps Ib, Ic). At the second stage, the discretization domain is redefined (step IIa), and a new graph $\{L'(k',i)\,|\,k'=1,2,...m';\,i=1,2,...n\}$ of the size $m'\times n$ is generated in the neighborhood of the initial solution (step IIb). After applying again the motion planning function $DP(.)$ (step IIc), the optimal solution of the length $D'_{\min}$ and corresponding path indices $\{k'^0(i)\}$ are obtained. The optimal trajectory is extracted at the step (IId).

Algorithm 3: ***Two-stage path planning***

***Input*:**	Task location matrices $\{Task(i)\mid i=1,2,...n\}$
	Upper limit of redundant variable $q_P^{\max}$
	Lower limit of redundant variable $q_P^{\min}$
	Neighborhood size Rg_P
	Primary discretization density m
	Secondary discretization density m'
***Output*:**	Minimum path length $D'_{\min}$
	Optimal path indices $\{k'^0(i)\mid i=1,2,...n\}$
***Notations*:**	$\{L(k,i)\}$ - $m\times n$ matrix of locations for the first stage
	$\{L'(k',i)\}$ - $m'\times n$ matrix of locations for the second stage
***Invoked functions*:**	Graph generation $G(.)$; (*algorithm 1*)
	Path planning $DP(.)$; (*algorithm 2*)

Stage I:

(a) *Generate an initial graph using rough discretization in full domain of* q_P

Set $\{L(k,i)\mid i=1,2,...n;\ k=1,2,...m\} := G\left(\{Task(i)\mid i=1,2,...n\},\ [q_P^{\min}, q_P^{\max}],\ m\right)$;

(b) *Apply algorithm DP*(.)

Set $\{\{k^0(i)\mid i=1,2,...n\};\ D_{\min}\} := DP\left(\{L(k,i)\mid i=1,2,...n;\ k=1,2,...m\}\right)$;

(c) *Extract initial trajectory*

For $i=1$ to n **do**

$InitTraj(i) = L(k^0(i),i)$;

Stage II:

(a) *Redefine the discretization domain*

For $i=1$ to n **do**

$q_P'^{\min}(i) = \max\{(InitTraj(i).q_P - Rg_P),\ q_P^{\min}\}$;

$q_P'^{\max}(i) = \min\{(InitTraj(i).q_P + Rg_P),\ q_P^{\max}\}$;

(b) *Generate local graph using fine discretization in local domain of* q_P

Set

$\{L'(k',i)\mid k'=1,2,...m';\ i=1,2,...n\} = G\left(\{Task(i)\mid i=1,2,...n\},\ \{[q_P'^{\min}(i), q_P'^{\max}(i)]\mid i=1,2,...n\},\ m'\right)$;

(c) *Apply algorithm DP*(.) again

Set $\{\{k'^0(i)\mid i=1,2,...n\};\ D'_{\min}\} := DP\left(\{L'(k',i)\mid i=1,2,...n;\ k'=1,2,...m'\}\right)$;

(d) *Extract optimal trajectory*

For $i=1$ to n **do**

$OptTraj(i) = L(k'^0(i),i)$;

3.3.2 Second Strategy: Smoothing the redundant variable profiles

An alternative way to speed up the motion generation algorithm is to replace the stage II in the first strategy by simple smoothing the curve $q_P(t)$ corresponding to the redundant variable without changing the time instants obtained from the stage I (if it is acceptable from technological point of view). This can be achieved by applying the polynomial approximation to the function defined by the nodes $\{q_P(i), t(i)\mid i=1,2,...n\}$. From our experience, the 3rd order polynomials are quite satisfactory for this procedure, which generates modified values of the redundant variable $\{q'_P(i)\mid i=1,2,...n\}$ ensuring better profile of the curve $q'_P(t)$. Then, in order to guarantee that the task locations are exactly visited

by the end-effector, the robot joint coordinates should be also regenerated using the new values of the redundant variable $\{q'_P(i) \mid i=1,2,\dots n\}$ and sequentially reapplying the positioner direct kinematic and robot inverse kinematic transformations (see subsection 2.2.1 and subsection 2.2.2). This procedure is shown in Figure 48. It is worth mentioning that the smoothing-based technique is very time-efficient because it excludes the optimization at the stage II. However, in most cases studied by the authors the two-stage technique including the local optimization provided better results.

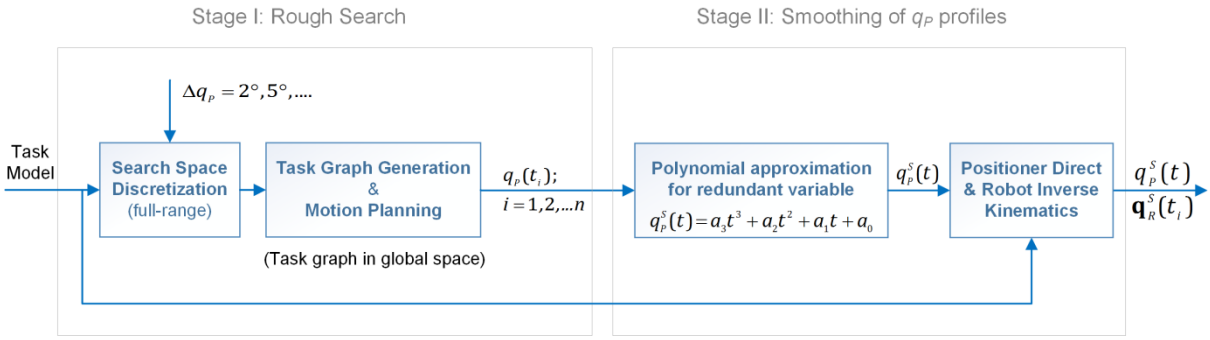


Figure 48 Combination of the rough search and smoothing of the redundant variable profiles

Algorithm 4 (Stage II only): *Smoothing of redundant variables*

***Input*:**	Task location matrices $\{Task(i) \mid i=1,2,\dots n\}$
	Initial trajectory from Stage I $\{InitTraj\ (i) \mid i=1,2,\dots n\}$
***Output*:**	Smoothed trajectory $\{q_P^S\ (i), \mathbf{q}_R^S(i) \mid i=1,2,\dots n\}$
***Notations*:**	q_P^S, $\mathbf{q}_R^S$ - Smoothed positioner and robot joint coordinates
	T_P, T_R - Matrices of robot tool and positioner flange in local frames
	order - Polynomial order
***Invoked functions*:**	Polynomial approximation $AP(.)$
	Robot inverse kinematics $g_R^{-1}(.)$ in local frame
	Positioner direct kinematics $g_P\ (.)$ in local frame
	Transformation from robot base to positioner base $Trans\ (.)$

Stage I:

-

Stage II:

(a) *Polynomial approximation of the redundant variable profile*

Set $\{q_P^S(i) \mid i=1,2,\dots n\} = AP\left(\{InitTraj\ (i).q_P \mid i=1,2,\dots n\},\ \{InitTraj\ (i).t \mid i=1,2,\dots n\},\ \text{order}\right);$

(b) *Recompute the robot trajectory using smoothed redundant variable profile*

For $i=1$ to n **do**

$T_P := g_P\left(q_P^S(i)\right) \cdot Task(i)$;

$T_R := Trans\,(T_P)$;

$\mathbf{q}_R^S(i) := g_R^{-1}(T_R, \boldsymbol{\mu})$;

An algorithm outline for the second strategy is presented below (see Algorithm 4). It is obvious that the Stage I of this algorithm is exactly the same to that in previous subsection, so the Stage II dealing with the smoothing of the redundant variables is presented only. The input includes the task location

matrices $\{Task(i)|\, i=1,2,...n\}$, and the initial trajectory obtained from the Stage I (where the redundant variable profiles will be used). The output contains the smoothed trajectories of redundant variables and the robot joint coordinates $\{q_P^s(i), \mathbf{q}_R^s(i)|\, i=1,2,...n\}$. A polynomial fitting is firstly applied to the initial trajectory of the redundant variable at the step IIa. Then, the smoothed $\{q_P^s(i)|\, i=1,2,...n\}$ is used to update the robot trajectory taking into account the positioner direct kinematics and robot inverse kinematics (step II).

3.3.3 Performance Evaluation of the Enhanced Algorithms

In order to estimate computational efficiency of the developed algorithms and evaluate smoothness of the obtained motions, there were carried out a number of computational experiments that deal with the benchmark example adopted in this work (robotic system composed of 2-axis manipulator and 1-axis positioner, 100 task locations). In these experiments, several algorithms were compared, including the conventional one (based on *Dijkstra*'s shortest path), the DP-based algorithm with fine discretization and also enhanced algorithms with local fine optimization or with smoothing at the second stage. The computing time was estimated in the Matlab R2014b environment running in Win7SP1 (with Intel® i5 CPU 2.67GHz 2.67GHz, 8G memory). It is clear that for C++ implementation the computing time is essentially smaller but it does not influence on the relation between the computing times for different algorithms.

Table 13 Comparison of the conventional and developed algorithms using the benchmark example (Robot Motion Time vs. Computing Time)

Discretization step	Conventional technique (Dijkstra's shortest path) *without acc. constraints*	One-stage DP (global fine search) *with acc.constraints*	Two-stage DP with local search *with acc. constraints*	Two-stage DP with smoothing *with acc. constraints*
$\Delta q_P = 3.0°$ (1.2×10^4 vts.)	T_{motion}= 7.63 sec $T_{comp.}$= 3.2 min	T_{motion}= 9.44 sec $T_{comp.}$= 2.4 min	T_{motion}= 9.44 sec $T_{comp.}$= 2.4 min (*Stage I: global*, ±180°) ↓ T_{motion}=8.47 sec $\Delta T_{comp.}$= +85 s (*Stage II: local*, ±30°) ↓ T_{motion}=7.63 sec $\Delta T_{comp.}$= +85 s (*Stage II+: local*, ±15°)	T_{motion}= 9.44 sec $T_{comp.}$= 2.4 min (*Stage I: global*, ±180°) ↓ T_{motion}=8.47 sec $\Delta T_{comp.}$= +85 s (*Stage II: local*, ±30°) ↓ T_{motion}=8.47 sec $\Delta T_{comp.}$= ~0 s (*Stage II+: smoothing*)
$\Delta q_P = 1.0°$ (3.6×10^4 vts.)	T_{motion}= 6.15 sec $T_{comp.}$= 20.9 h	T_{motion}= 8.20 sec $T_{comp.}$= 15.5 min		
$\Delta q_P = 0.5°$ (7.2×10^4 vts.)	T_{motion}= 6.00 sec* $T_{comp.}$ ~ 37 days†	T_{motion}= 7.51 sec $T_{comp.}$= 55.4 min		
Trajectory property	Time-optimal, non-smooth	Time-optimal, non-smooth in some cases	Time-optimal, not very smooth	Smooth, time-suboptimal

*Obtained using the developed one-stage DP algorithm without acceleration constraints

† Estimated using polynomial approximation of the algorithm complexity (leading term)†

In order to compare the developed algorithms, the positioner joint coordinate q_P was treated as the redundant variable and firstly was discretized in the interval [–180°, 180°] with the step $\Delta q_P = 3.0°$. Then, by sequentially applying the positioner direct kinematic and robot inverse kinematic transformations, a column of 121 candidate configuration states was obtained for each of 100 task points. To ensure that no collision and no singular configuration happen, each candidate configuration state was verified. For this operation, the robot links and the path segments were presented as short straight lines, and the intersections of the robot links and path segments were checked. If the intersection was detected, the current configuration state was marked as "inadmissible" one. Besides, similar mark was used for the vertices for which the robot inverse kinematics is insolvable or the manipulator configuration is too close to the singularity. After this procedure, the original sequence of the task points $\mathbf{p}_i$ was converted into the task graph composed of 12100 vertices containing the joint coordinates of the robot and positioner. The obtained graph was used for the stage I of the optimization algorithm. The computing time for the graph generation was 43 seconds. Similarly, there were generated the task graphs for the smaller discretization steps $\Delta q_P = 1.0°$ and $\Delta q_P = 0.5°$. In the first case, the task graph was composed of 36100 vertices and it took around 2 minutes to generate it. For the second case, the task was converted into the graph composed of 72100 vertices, it took about 5 minutes.

Using the obtained graphs, a conventional technique based on the *Dijkstra*'s shortest path was applied first to find the time-optimal trajectory. In the case of $\Delta q_P = 3.0°$, it took 3.2 minutes to go through the graph generation, data transformation and path planning in order to find a solution with the robot motion time T_{motion} = 7.63 sec. By reducing the discretization step down to $\Delta q_P = 1.0°$, a better solution with T_{motion} = 6.15 sec was obtained but it required more than 20 hours of computations. Further, for the discretization step of $\Delta q_P = 0.5°$, the conventional technique became extremely slow and did not allow us to complete the optimization in acceptable time. In fact, a rough estimation of $T_{comp.}$ based on polynomial approximation yields the computing time more than 30 days, which is not acceptable in practice. Nevertheless, relevant time-optimal solution with T_{motion} = 6.00 sec for this case was found using the developed one-stage DP based algorithm with released acceleration constraints, which obviously must give the same result as the *Dijkstra*'s shortest path. Hence, the conventional technique, which does not take into account the particularities of the task graph, cannot be applied to the considered engineering problem because of excessive computational expenses. Besides, the trajectory smoothness requirement is also not respected here.

It should be noted that the above estimation of the computing time is not in good agreement with the classical formula $O(|E|+|V|\cdot\log|V|)$ describing the complexity of the *Dijkstra*'s algorithm via the number of the graph edges $|E|$ and vertices $|V|$ (Fredman and Tarjan, 1987). This issue is caused by a number of additional computations included in the path planning algorithm that deal with the data transformation and computing the distances between the nodes, which are in fact the travelling times between the manipulator and the positioner configurations corresponding to the considered vertices.

In contrast to the conventional *Dijkstra*'s shortest path technique, the developed one-stage DP algorithm is essentially faster and allows user to take into account the acceleration constraints that ensure smoothness of the robot and positioner motion. In particular, the computing time for the discretization $\Delta q_P \in \{3.0°, 1.0°, 0.5°\}$ is equal to 2.4, 15.5 and 59.9 minutes respectively. It is understandable that the optimal motion time should be higher here compared to the previous column where the acceleration constraints are omitted. While reducing the discretization step, the optimal motion time gradually decreases from 9.44 to 8.20 and 7.51 seconds. It is worth mentioning that

further reduction of the discretization step is not reasonable because it yields almost the same optimal motion time of the robotic system (the minimum value of T_{motion} obtained in computational experiments for smaller Δq_P is equal to 7.49 sec). Besides, for $\Delta q_P = 0.5°$ this algorithm generates smooth motions that satisfies the engineering requirements. Nevertheless, even for this simple benchmark example, the computing time is still too high if the discretization step is small enough, which limits the practical applicability of the developed one-stage DP algorithm.

The first version of the enhanced two-stage DP algorithm which combines the initial search with the rough discretization and the local optimization with the fine discretization provided a slightly higher value of the optimal motion time which is equal to 7.63 sec. The latter is understandable because the local search space is limited here and it does not obligatory include the global optimal solution. Nevertheless, such small increase of the motion time is not very essential for practice while the profit from the reduction of computational expenses is very high. In particular, the total computing time is about 5.2 minutes only, for executing the first stage with $\Delta q_P = 3.0°$ and repetition of the second stage twice with $\Delta q_P = 1.0°$ and $\Delta q_P = 0.5°$. It should be also mentioned that the smoothness of the obtained trajectory satisfies the engineering requirements.

The second version of the enhanced two-stage DP algorithm combining the rough search, local search and the trajectory smoothing yielded higher value of the motion time 8.47 sec but it is faster than the previous one. In particular, for this technique the total computing time is about 3.8 min and the trajectory smoothness is very good. It is clear that in practice, the user can repeat the local optimization several times and then apply the smoothing, which allows to achieve the same motion time as above. Generally, the first and second versions of the enhanced two-stage DP algorithms are complimentary and may be applied in different variations depending on the particular engineering problem.

Advantages of the proposed motion planning techniques are illustrated by Figures 49–52 that present the trajectories generated using the enhanced versions of the developed motion planning algorithms. The first group of them, which includes Figures 49 and 50, is related to the version #1 and demonstrates benefit of the local optimization. It also shows improvement of the optimization results with reduction of the discretization step, which gradually decreases from 3.0° to 0.5°. As follows from Figure 49, the first stage that is running with relatively large discretization step $\Delta q_P = 3.0°$ yields quite smooth motions but the robotic system motion time is not achieved its minimal value. In contrast, Figure 50 showing results of the local optimization for $\Delta q_P = 0.5°$ with the range $\pm 30°$ confirms benefit of the proposed approach, which for this example allowed reducing the motion time by 20%. However, the trajectory smoothness drops down here. So further reduction of Δq_P is needed but it obviously requires additional computing time.

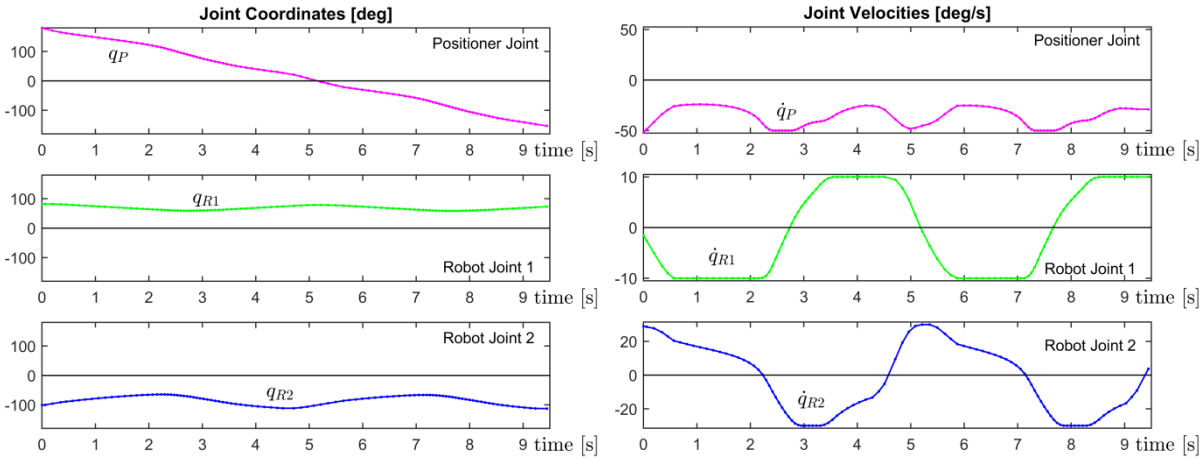


Figure 49 Enhanced algorithm #1: trajectories obtained at the Stage I
(Δq_P = 3°, T_{motion} = 9.44s)

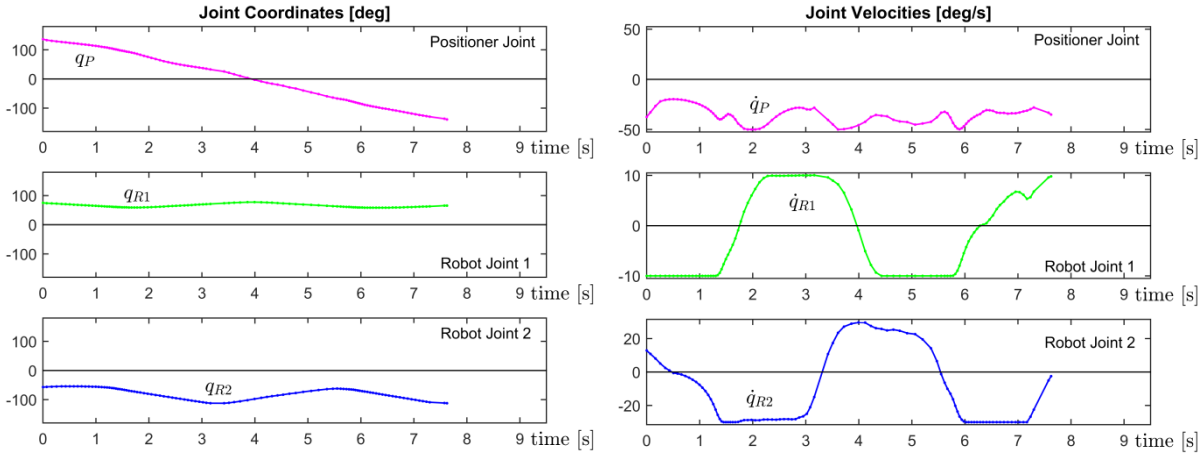


Figure 50 Enhanced algorithm #1: trajectories obtained at the Stage II
(Δq_P = 0.5°, T_{motion} = 7.63s)

The second group of illustrations, which includes Figures 51 and 52, is related to the version #2 and demonstrates the gain from the smoothing of the redundant variable profile. Here, the local optimization was stopped with the larger discretization step $\Delta q_P = 1.0°$ that does not provide the satisfactory smoothness, see Figure 51. Further, the function $q_P(t)$ describing the positioner motion is smoothed by applying the cubic spline approximation without changing the motion time. The robotic manipulator motions are also adjusted using the kinematic equations. It allows us to obtain very smooth trajectories, see Figure 52, which are obviously suboptimal with respect to the motion time. It is clear that the similar smoothing technique can be also integrated into the first version allowing achieving smaller motion time if it is necessary from the engineering point of view. It is also worth mentioning that in some cases the smoothing can cause some small violations of the velocity/acceleration constraints but it was not observed for the considered benchmark example.

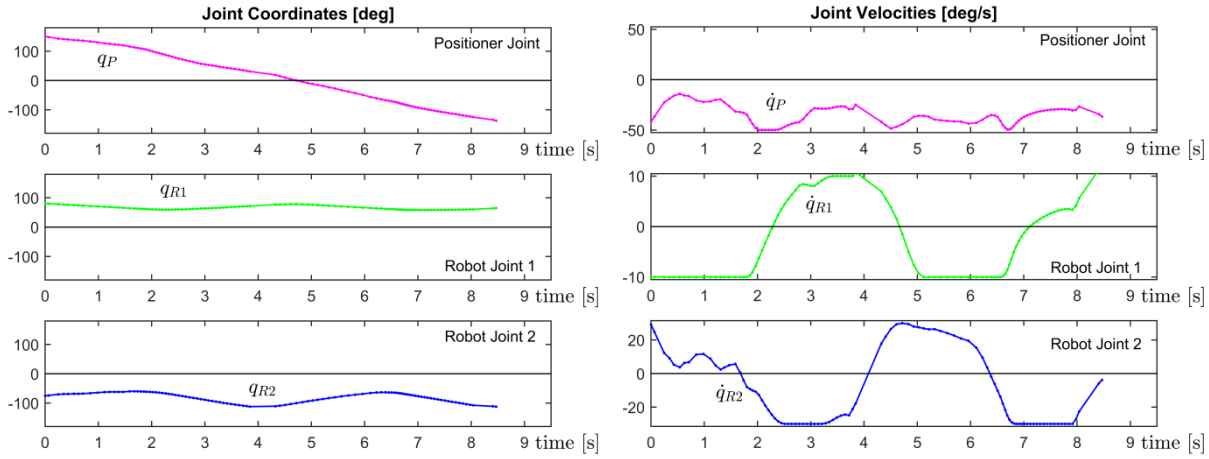


Figure 51 Enhanced algorithm #2: trajectories obtained at the Stage II
(Δq_P = 1°, T_{motion} = 8.47s)

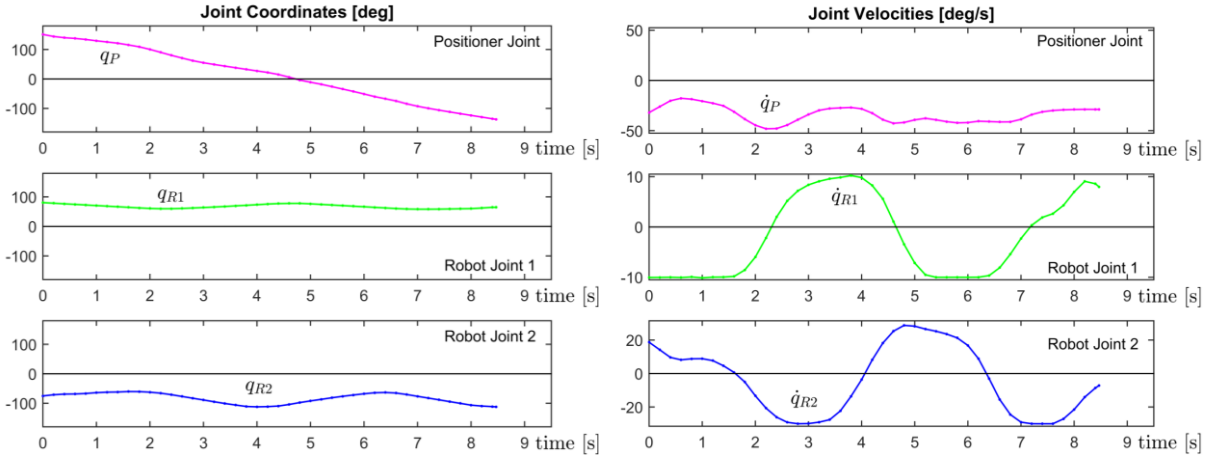


Figure 52 Enhanced algorithm #2: trajectories obtained after smoothing
(T_{motion} = 8.47s)

Therefore, the presented simulation results confirm the efficiency of the developed enhanced algorithms for the motion planning in the redundant robotic system. They can be applied in several ways, combining stages of global/local optimization and smoothing of the redundant variable profiles. While generating the robotic system motions, the user can find reasonable balance between the computing expenses and achieved level of the key design objectives, which are the total motion time and the trajectory smoothness. Another conclusion following from the simulation is related to the selection of the discretization step, which is an issue to be investigated in more details. As follows from the presented case study, a smaller discretization step may provide faster and smoother motions but relevant computation cost is extremely high. On the other side, excessive discretization does not yield significant improvement of the performance index (the robotic system motion time). For instance, for the considered example, reducing the discretization step from $\Delta q_P = 0.5°$ to $\Delta q_P = 0.3°$ gives almost negligible improvement of the motion time, by 0.4% only (tested using the one-stage DP algorithm). At the same time, it requires 3.9 hours of computation instead of 55.4 minutes for $\Delta q_P = 0.5°$. Hence, it is reasonable to provide the user with some simple rules for selecting reasonable discretization step in order to avoid too excessive computations.

3.4 PARAMETERS TUNING FOR MOTION PLANNNING ALGORITHM

As follows from the previous section, the developed algorithms have some essential advantages such as high computational efficiency and capability to take into account both of the velocity and acceleration constraints. But proper selection of discretization step for the redundant variable is not trivial. An inappropriate discretization step may lead either to a bad solution or to high computational effort. For this reason, some recommendations for selecting the discretization step are presented below to tune the optimization algorithm.

3.4.1 Influence of Discretization Step on the Generated Trajectories

As it was shown before, a smaller discretization step is better for the considered problem and it allows generating faster and smoother motions because of better approximation of the original continuous problem. On the other hand, too excessive discretization may not yield significant improvement of the primary objective, which is the robotic system motion time here, while taking too much computing time. To investigate the influence of discretization step on the generated trajectories, a slightly different benchmark example is considered here dealing with a straight line lay-up task presented in Figure 53. For this task, the desired linear path was uniformly discretized and replaced by 40 discrete points $\mathbf{p}_i = (x_i, y_i)^T$ where $i = 1, 2, \ldots 40$. These points should be visited sequentially by the robot end-effector in minimum time by using in the best way the motion capabilities of both the robot and the positioner.

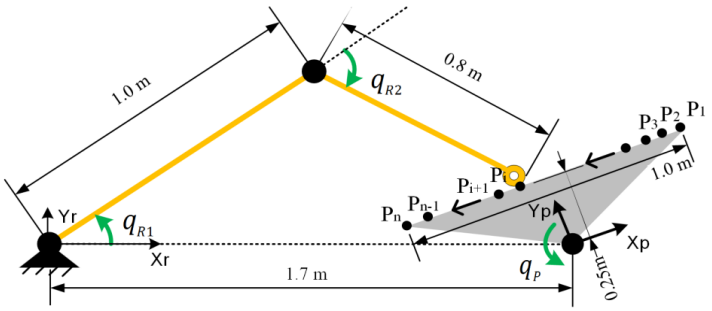


Figure 53 A benchmark task for investigating the influence of Δq_P

For the considered benchmark task, the motion planning problem was solved for different discretization steps $\Delta q_P \in \{2.00°, 1.00°, 0.75°, 0.50°, 0.25°, 0.10°\}$ using the above presented one-stage algorithm, which requires more computations compared to the two-stage ones but obviously allows us to obtain the global optimal solution. Relevant results are presented in Table 14 and in Figure 54, which show that decreasing of the discretization step Δq_P from 2.00° to 0.50° allows achieving the motion time reduction by about 30%. However, further decreasing of Δq_P down to 0.25° and 0.10° yields negligible influence on the motion time that is the primary optimization objective in the developed algorithms. Hence, for this case study, the optimal discretization step is about 0.5°.

There are also some interesting phenomena that were observed in the optimization results for relatively large discretization step Δq_P. In particular, in the case of $\Delta q_P = 2°$, the optimization algorithm generates a technically unreasonable solution that does not take advantage of the positioner motion capabilities. For this solution, the desired motion is executed by the robot only while the positioner stands still (locked), i.e. $q_P = const$, $\mathbf{q}_R = var$. The reason for this is that the discretization

step Δq_P here is so high that time required for the positioner rotation from q_P to $q_P \pm \Delta q_P$ is always larger than the robot alone moving time between the subsequent task points. Another specific phenomenon can be observed for slightly smaller discretization step $\Delta q_P = 1°$, where the algorithm produces a solution with non-smooth intermittent positioner rotation (start-stop motion), i.e. $q_P, q_P, \ldots (q_P + \Delta q_P), (q_P + \Delta q_P), \ldots (q_P + 2\Delta q_P), \ldots$. It is clear that both of these solutions are not acceptable in practice because they do not use all advantages of the robotic system redundancy. Hence, the discretization step for the redundant variable Δq_P should be small enough to avoid both the positioner locking and its start-stop motions.

Table 14 Simulation results for linear lay-up task for different discretization step Δq_P

	$\Delta q_P = 2.00°$	$\Delta q_P = 1.00°$	$\Delta q_P = 0.75°$	$\Delta q_P = 0.50°$	$\Delta q_P = 0.25°$	$\Delta q_P = 0.10°$
T_{motion} *without a-constr.*	1.897 s (~38s comp.)	1.836 s (~2min comp.)	1.540 s (~4min comp.)	1.298 s (~9min comp.)	1.290 s (~47min comp.)	1.290 s (~5h comp.)
T_{motion} *with a-constr.*	1.897 s (~67s comp.)	2.107 s (~4min comp.)	1.594 s (~8min comp.)	1.300 s (~17min comp.)	1.290 s (~1.2h comp.)	1.290 s (~9h comp.)

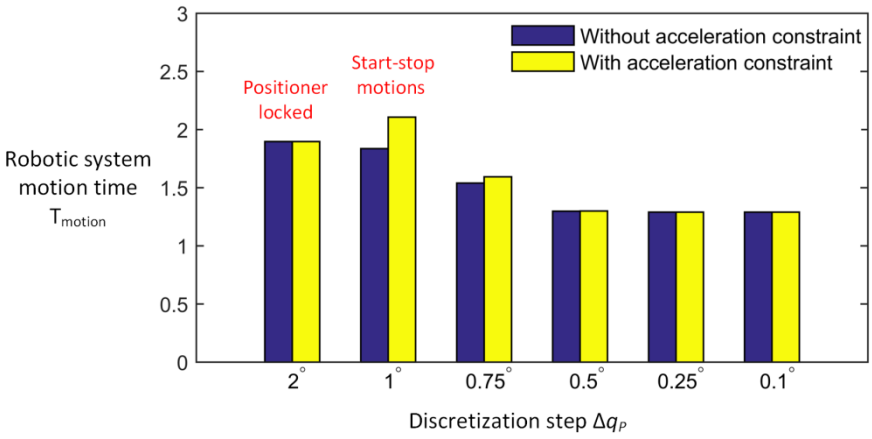


Figure 54 Evolution of robotic system motion time with reduction of different discretization step Δq_P

In addition, it was noted that the discretization step reduction does not always yield monotonic decreasing of the robotic system motion time, while it is intuitively follows from the physical nature of the considered motion planning problem. For example, the discretization step reduction from $\Delta q_P = 2°$ to $\Delta q_P = 1°$ leads to even worse solution, where the system motion time is about 10% higher (case with acceleration constraints). This non-monotonic phenomenon can be explained by heuristic integration of the acceleration constraints into the optimization algorithm, which may slightly violate the dynamic programming principle. In more details it is explained in Figure 55 where it is presented a portion of the task graph corresponding to 17^{th}-19^{th} task locations. For the convenience, the configuration states are denoted by capital letters, and the numbers in each cell indicates the length of the shortest path to the current one. For instance, in the cell F, 1.1334 is the shortest path from previous C to the current one, and the distance between C and F is 0.0667. In the search process, the acceleration constraint is applied on the candidate sub-path composed of three configuration states. For example, here there are three possible ways to J, which are B-E-J, C-F-J and D-G-J, and all of them are checked for the acceleration constraint. Then, the shortest path satisfying the acceleration constraint is recorded in J. In this example, the recorded path is D-G-J with the length 1.6737, since shorter options C-F-J and B-E-J with the lengths 1.2004 and 1.2111 violate the acceleration limits. However, we can notice that

another possible path B-F-J with the length 1.2020 was not detected by the algorithm because the recorded previous cell of F is C since C-F is obviously shorter than B-F. So, because of the heuristic integration of acceleration constraints in the DP-based algorithms, some better trajectories may be missed, which explains increasing of the objective function while reducing the discretization step Δq_P from 2° down to 1°. However, for smaller Δq_P this phenomenon is not observed.

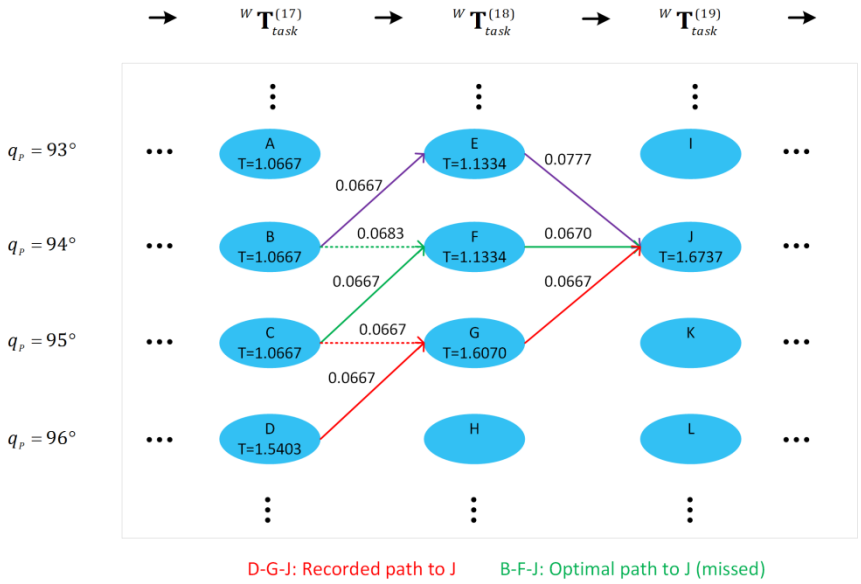


Figure 55 A portion of the task graph corresponding to the task locations 17, 18 and 19

Hence, to apply the developed technique in practice, users need some simple "rules of thumb" that allows setting an initial value of Δq_P. Then, the optimization algorithm can be applied several times (sequentially reducing Δq_P) until the objective function convergence.

3.4.2 Practical Recommendations for Selecting the Discretization Step

To find a reasonable initial value of the discretization step, let us investigate in details robot and positioner motions between two sequential task locations. It is clear that for smooth positioner motions, it is required that corresponding increments of the coordinate q_P should include at least several discretization steps. To estimate the maximum value of Δq_P, let us consider the movement between two adjacent task locations (P_i-P_{i+1}) and denote corresponding increment of the positioner coordinate as $\Delta\theta$ and the length of the path segment as Δs.

Since the workpiece is rotated along the positioner axis, as small segment of processing path can be approximated by a circle arc (see Figure 56), and the task point displacement due to positioner can be approximately expressed as $r_{\max} \cdot \Delta\theta$ where r_{max} is the furthest point distance to the positioner axis.

As follows from the physical sense, to avoid undesired intermittent positioner rotations, the following inequality $\Delta s > r_{\max} \cdot \Delta\theta$ should be satisfied, since in the Cartesian space the positioner velocity is usually smaller than the velocity of the robot. It can be also rewritten in the form $\Delta\theta < \Delta s / r_{\max}$. Moreover, to ensure optimal coordinated motions of the positioner and the robot, it should be satisfied the equality $\Delta s = r_{\max} \cdot \Delta\theta + v_R^{\max} \cdot \Delta\theta / \dot{q}_P^{\max}$ where $v_R^{\max}$ is the maximum Cartesian speed of the robot end-effector. The latter gives us a rough estimation of the positioner increment

$$\Delta\theta = \frac{\Delta s}{r_{\max} + v_R^{\max} / \dot{q}_P^{\max}} \tag{110}$$

which allows setting an initial value of the discretization step $\Delta q_P = \Delta\theta / k$ where k should be higher than one to avoid the positioner locking or the start-stop motions. For example, for the above benchmark example, this expression gives the increment value $\Delta\theta = 0.55°$, which explains some undesired phenomena observed for $\Delta q_P = 2°$ and $\Delta q_P = 1°$. Hence, such simple "rules of thumb" based on the equation (110) helps the user to select an initial value of Δq_P. Then, the optimization algorithm is applied iteratively with sequentially reduced discretization step until the robotic system motion time convergence.

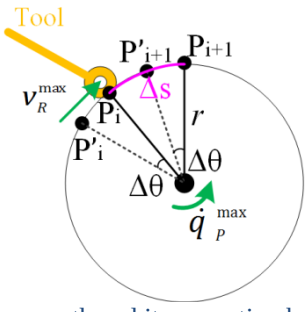


Figure 56 A sample of the lay-up path and its execution by the robot and the positioner

Another issue to be discussed here is related to the sampling of the lay-up path (i.e. the task discretization). Since the original task is defined as a continuous Cartesian curve, it seems reasonable to describe it using relatively high number of discrete points. However, as follows from our study, the excessive sampling of the path leads to essential computational efforts while providing just slight benefits for the fiber placement quality, since relevant technology is not extremely sensitive to the positional accuracy of the robot end-effector. In particular, some positional errors can be compensated by the end-effector that provides pressure for the fiber compacting. Besides, this technology requires some overlaying of the successful fiber coats, and the overlaying degree is not very critical here. From our experience, the lay-up path discretization with 100 to 200 nodes is quite sufficient for the considered examples. But this number should be certainly increased for complicated objects that are presented in the Chapter 4.

3.5 SUMMARY

This chapter is devoted to the development of a new motion planning method for the redundant robotic systems utilized in the automated composite lay-up processes. The main contributions are in the area of redundancy resolution via optimization techniques. The proposed method is based on the transforming the original problem into a combinatorial one and applying the dynamic programming principle. To reduce the computing time, the time-optimal motion planning is divided in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with large discretization step. Then, the same technique is applied in the local search space, which is created with smaller step in the neighborhood of the obtained trajectory. Alternatively, the second stage may implement a straightforward smoothing of the redundant variable profiles. Efficiency of them and advantages compared to the conventional techniques are confirmed by several case studies.

In more details, contributions of Chapter 3 can be summarized as follows

(i) *Transformation of the original continuous optimization problem into a combinatorial one* by discretizing the redundant variables (actuated coordinates of the positioner and workspace extension unit) and sequentially applying to all task locations the direct kinematics of the positioner/extensioner as well as inverse kinematics of the robot.

(ii) *Development of a new motion planning method* which allows minimizing the total motion time and ensuring smooth movements of all mechanical components. This method is based on dynamic programming and, in contrast to the conventional techniques, it allows taking into account the acceleration constraints related to the trajectory smoothness.

(iii) *Development of the enhancement strategies* for the developed motion planning technique. To reduce the computing time required for the motion planning, the time-optimal motion planning is divided in two stages combining global and local searches. As an alternative, the second stage may implement a straightforward smoothing of the redundant variable profiles based on the spline approximation. The efficiency of these strategies was confirmed by case studies.

(iv) *Development of the practical recommendation for the parameters tuning* in the proposed motion planning algorithm, such as selection of the discretization step for the redundant variables and size of the local optimization sub-space.

The motion planning method developed in this chapter allows generating smooth time-optimal trajectories required for the composite lay-up technology. Its advantages are confirmed by an industrial case study presented in the following chapter.

The main results of Chapter 3 have been presented in conferences EUCOMES'2016 and ICMIT'2017, they also have been published in the journal *Mechanism and Machine Theory* (2017) (Gao et al., 2017c, Gao et al., 2017a, Gao et al., 2017b).

CHAPTER 4 EXPERIMENTAL VALIDATION AND INDUSTRIAL IMPLEMENTATION

This chapter deals with industrial implementation of the developed motion planning method on the factory floor (for manufacturing of a high-pressure vessel). Using the proposed method, there were generated time-optimal smooth motions for the manipulator and positioner of the SPIDE-TP robotic platform allowing speeding up the thermoplastic tape lay-up process. Correctness of these motions was verified in 3D simulation environment of CATIA software package. Besides, it was created a program in KRL language implementing these motions and ensuring coordinated control of KUKA KR210 robot and AFPT workpiece positioner. There are presented results of industrial experiments carried out with this program, which confirmed smoothness of the manipulator/positioner movements and essential reduction of the time required for the thermoplastic tape lay-up process.

4.1 MANUFACTURING PROCESS PREPARATION FOR HIGH-PRESSURE VESSEL FABRICATION

High-pressure vessels are containers designed to hold gases or liquids under a pressure substantially higher than the ambient one. Common pressure vessels are made of steel. However, since rolled and forged parts were welded together during fabrication, some mechanical properties could be adversely affected, which leads to increase the thickness to resist the high pressure. So, such all-metal construction is usually very heavy, weighing approximately 1.4kg/L. To save the weight, there are some other manufacturing techniques based on composite materials. For example, the constructions of metal liner with full composite overwrap (generally aluminum with a carbon fiber composite) and all-composite structure comprising a polymer liner with carbon fiber composite allow to achieve the weight parameter 0.45 kg/L and 0.3 kg/L respectively (www.compositesworld.com). This section presents the manufacturing process and relevant robotic system used for fabrication of a high-pressure vessel covered by thermoplastic fiber at CETIM company (Nantes, France).

4.1.1 Manufacturing Task Description and its Regularization

The high-pressure vessel considered in this chapter is composed of a cylindrical part and two elliptical domes at both ends of the cylinder. The cylinder has the diameter of 168 mm and the length of 400 mm, the maximum pressure is 500 bar. The vessel is covered by several layers of the thermoplastic fiber as shown in Figure 57, which are winded by the robot and consolidated in situ following certain patterns.

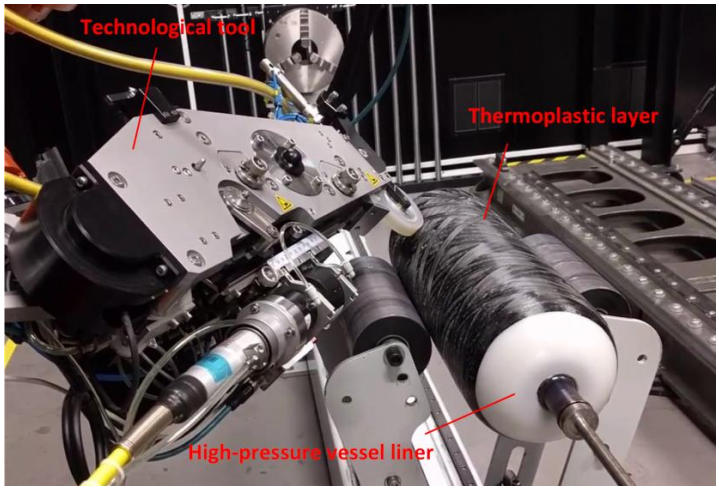


Figure 57 The high-pressure vessel with thermoplastic fiber covering, ©CETIM.

To generate the lay-up patterns, special software from Composicad was used that allows user to create the winding path for the variety of products such as pipes, tubes, tanks, vessels, etc. (www.composicad.com). This software includes a number of packages targeted at specific geometry of the workpiece. For the considered pressure vessel, the dedicated package Composicad-Silver was used, which is specially designed for shapes with axial symmetry. This package allows user to specify each layer with the information of lamina type (a helical winding, a circumferential winding, a connector or a transition winding), weight, thickness, etc. Then, the software calculates the minimum number of circuits required for complete coverage. For a number of layers, Composicad also

calculates the possible pattern repeats. Finally, the generated pattern is exported to the xls/.txt file that contains the Cartesian positions of the winding path as well as the normal direction vectors, and winding angles. It is also displayed as shown in Figure 58.

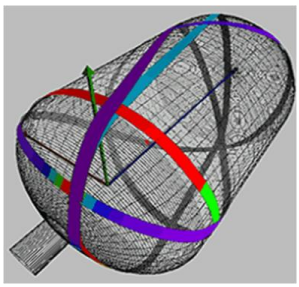


Figure 58 A sample of the laying task for a high-pressure vessel, ©CETIM.

It is worth mentioning that Composicad provides a functionality of time optimized motion generation while fixing the tool path with respect to the robot base frame (called as the "time optimal fixed trajectory path generation" in Composicad-Silver). Within this functionality, the user can specify the maximum Cartesian speed/acceleration of the technological tool as well as the maximum angular speed/acceleration of the workpiece rotation provided by the positioner. It is clear that, because of constraining the tool path in the global frame, such approach cannot be treated as strictly time-optimal compared to those developed in this work. Besides, the robot kinematics and its motion capabilities expressed by the maximum velocities/accelerations in the actuated coordinates are not taken into account in Composicad. Nevertheless, this quasi-optimal technique, which ignores redundancy of the robotic system, can be acceptable in some practical applications. On the other side, it gives user some preliminary estimation of the fiber-laying time that can be further reduced by applying the proposed motion planning algorithms for the redundant robotic systems.

For the high-pressure vessel considered here, a sample of the laying task ensuring a single circuit placement of helical lamina on the cylinder and two elliptical domes was created using the Composicad-Silver. Relevant data containing description of a discrete 3D-augmented curve were exported into an Excel file. They include Cartesian positions and normal directions of the laying path with respect to the workpiece frame as shown in Figure 59. To describe this path in the desired way, there were applied expressions (14)-(17) allowing presenting the laying task in the form of a sequence of 4×4 homogeneous transformation matrices. Besides, an equivalent representation was obtained using expressions (20)-(22) that yield relevant sequence of 6×1 vectors composed of Cartesian coordinates and Euler angles, which are commonly used in robot control and programming.

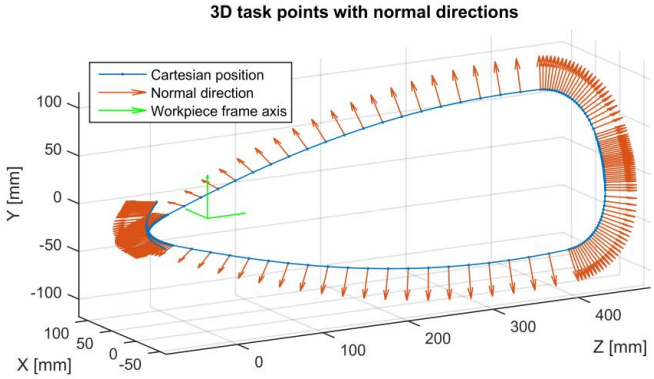


Figure 59 Discrete representation of the lay-up task in the form of the augmented line

By analyzing the task data file obtained from Composicad-Silver, it was noticed that the path sampling was not very regular. In particular, the sampling step varies from 1.5 to 21.0 mm and it is higher for cylinder surface and lower for the domes (see Figure 59). In practice, implementation of very short path segments creates certain difficulties for robot controllers that impose strict constraint on the minimum motion time between the trajectory nodes. The latter does not allow achieving the desired speed of the technological tool if the nodes are too close to each other. So, before applying the developed motion planning algorithms, the task data were modified and presented in the form with a regular sampling step. Besides, from analysis of the original data file generated by Composicad-Silver, it was observed that there are rather sharp variations on some Euler angles profiles, especially at the connection of the cylinder and the domes (see Figure 60a). It is clear that this phenomenon will leads to unsmooth robotic system motions if task path is not improved. In order to avoid these, the original task was slightly smoothed using cubic spline before implementing the motion planning algorithms. The final form of the laying task data is presented in Figure 60b.

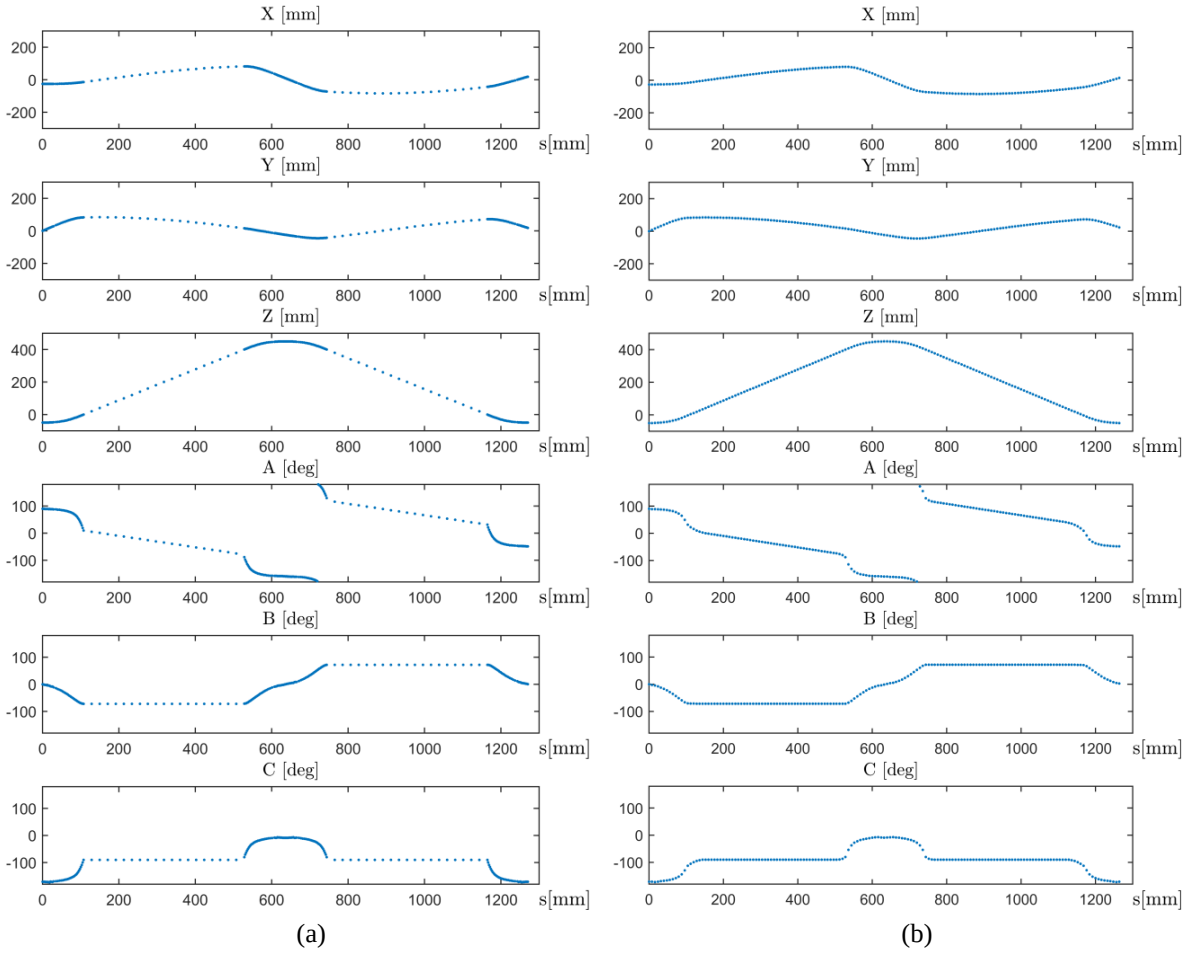


Figure 60 Regularization of the lay-up task discrete model
(smoothing and replacing an irregular step 1.5 to 21.0 mm by the regular step 8.0 mm)

After applying the above operations, a new Excel file of the laying task was generated that contains elements of 4×4 homogeneous transformation matrices and components of corresponding 6×1 location vectors, which describes a helical curve with 159 uniformly distributed nodes and the step 8.0 mm.

4.1.2 Selection of Robotic System Components and Workcell Lay-out Design

For manufacturing of the considered high-pressure vessel, the redundant robotic system called the SPIDE-TP platform (CETIM, Nantes) was used. This platform can be utilized for fabricating vessels of different sizes (diameters from 25 mm to 500 mm and lengths up to 3500 mm), including the vessel with the length 400 mm and diameter 168 mm presented in Figure 57. The SPIDE-TP system is composed of a 6-axis robot with 1-axis linear track from KUKA, a 1-axis workpiece positioner and a special technological tool from AFPT (see Figure 61).

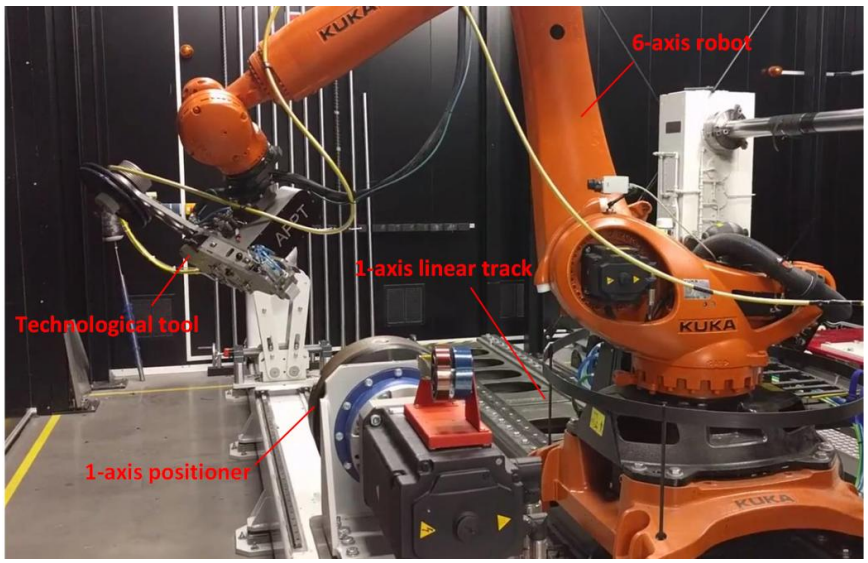


Figure 61 SPIDE-TP platform for manufacturing high-pressure vessels, ©CETIM.

The technological tool from AFPT used here is specially designed for producing rotational symmetric workpieces (www.afpt.de). It perfectly suits the shape of high-pressure vessel considered in this work. This tool is presented in Figure 62 and is composed of a compaction roller, a thermal camera, a heat source (diode-lasers), a material storage (fiber creel) and a tensioning system inside. The tool mechanics guides the thermoplastic fiber towards a mandrel or mold where the laser is used to heat the fiber to its melting temperature. Then the tape is consolidated under pressure by the compaction roller (adjusted by an air piston-cylinder) and cooled down. During the laying, the fiber temperature is maintained at the requested level using feedback control of the laser power (maximum 4 kW) based on the measurements from the thermal camera. This ensures optimal consolidation and prevents degradation of the thermoplastics.

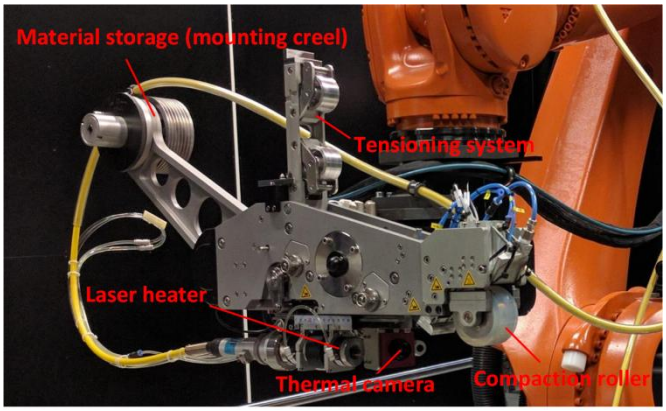


Figure 62 Technological tool used in SPIDE-TP platform, ©CETIM.

The workpiece positioner (see Figure 61) is also from AFPT company. It has one horizontal rotational axis and the following mandrel parameters: height of 1089.90 mm and length of 4000.00 mm. This positioner rotates the workpiece in the full range ±180° simultaneously with the robot actuators, with the maximum velocity 142°/s and the accelerating time 0.5 s. It is clear that such positioner size is excessive for the high-pressure vessel considered in this work (with length of 400 mm and diameter of 168 mm), but it is used here to increase the robotic platform versatility.

The 6-axis robot employed here is KUKA KR210 R3100 ultra, which has a rather high payload of 210 kg and a maximum reach of 3095 mm (www.kuka.com). More detailed information concerning the robot workspace is presented in Figure 63. The manipulator joint limits, velocities and accelerations are given in Table 11. This robot is equipped with the digital controller KR C4, similar to other types of KUKA robots.

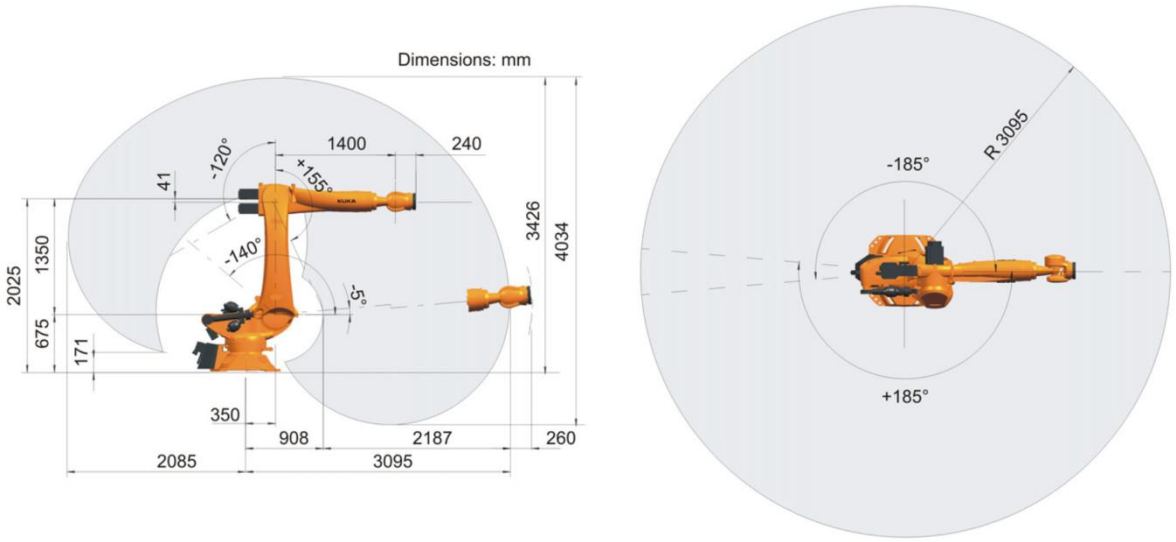


Figure 63 Work envelop of robot KUKA KR210 R3100 ultra, ©KUKA GmbH

The robot is installed on the linear track KUKA KL 2000, which allows extending the work envelop up to 4500 mm. This linear unit has maximum payload 2000 kg and provides a translational motion along a horizontal axis with the speed up to 1.96 m/s. The linear track is connected to the robot controller KR C4 coordinating actuation of six robot axes and two external axes (of the linear track and the rotational positioner).

Mutual location of the SPIDE-TP platform components, i.e. the robotic system layout, is predetermined by the platform manufacturer AFPT. Here, the liner track is paralleled the positioner mandrel with the distance 1396.08 mm between their centerlines. The workpiece is attached to the positioner mandrel at the distance of 1077.18 mm to the flange center, which in our geometric model is the distance between the origins of the workpiece frame and the positioner flange frame. Such design ensures the considered vessel to be completely covered by the robot work envelop.

It is clear that the presented robotic system is redundant with respect to the considered technological task, which requires six degrees of freedom only. In fact, in addition to the usual six degrees of freedom of the robotic manipulator, there are two extra degrees of freedom here (rotational and translational ones) that are provided by the positioner and the linear track. For this design, the workpiece rotation allows adjusting its orientation for better reachability of the technological tool

while the robot base translation increases the manipulator working range allowing processing large dimensional products.

For considered manufacturing task that deals with rather small workpiece, the robot can easily execute the composite tape laying process without changing its base location. Besides, the minimum travel distance of the linear track is comparable with the vessel length. So, for this product, there is no reason to activate the linear track during the technological process. Nevertheless, the problem of optimal robot placement still exists here while it is limited to the optimization of the linear track coordinate defining the robot base location relative to the workpiece. In this work, this problem was solved in straightforward way, by sampling the linear track coordinate and applying the algorithm developed in Chapter 3 to generate the time-optimal motions for each robot base location. Then, the optimal robot location was found by selecting the smallest motion time of the robotic system. To speed up this procedure, the motion planning Algorithm 2 was applied, and the discretization step for the redundant variable was relatively high (it was equal to 2°). Relevant computational results are presented in Table 15 and Figure 64. As follows from them, the optimal robot location corresponds to the linear track coordinate 3000 mm, which ensures the robotic system motion time 4.11 sec. It is worth mentioning that here the optimal robot placement problem does not have very clear minimum of the objective function, which is only 6.4% less than the biggest value corresponding to the track coordinate 2000 mm. So, in practice while optimizing the robot location, it is reasonable to pay primary attention to accessibility of the task points and collision avoidance.

Table 15 Robotic system motion time with respect to different robot base location on the linear track.

Robot location (mm)	2000	2200	2400	2600	2800	3000	3200	3400	3600	3800	4000
Motion time (sec)	4.55	4.36	4.27	4.32	4.19	4.11	4.19	4.31	4.41	4.58	4.86

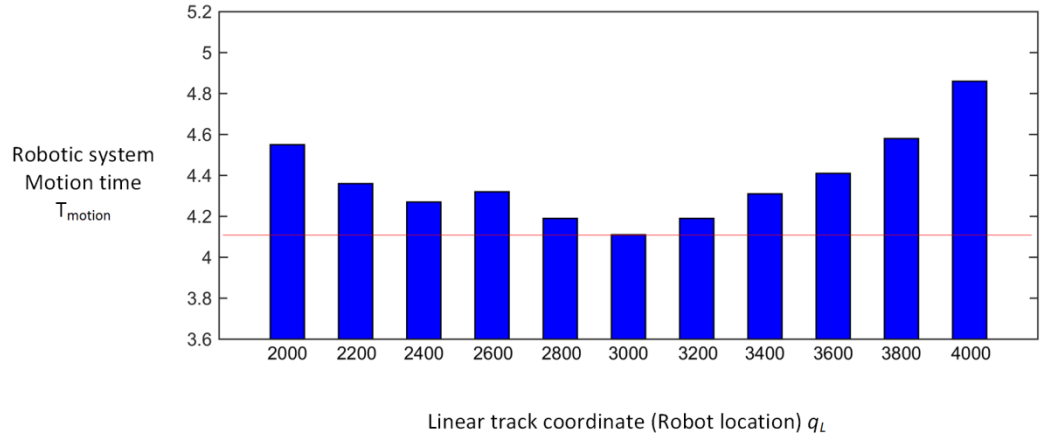


Figure 64 Selecting the optimal robot location on the linear track.

4.2 ROBOTIC SYSTEM MOTION PLANNING USING DEVELOPED ALGORITHMS

To generate the time-optimal motions of the considered robotic system, the redundant variable (positioner joint angle) was discretized and it was created the task graph containing all robot/positioner configurations corresponding to the given laying task. Then, the developed DP-based algorithms were applied to this graph to find the desired trajectory.

4.2.1 Generation of the Task Graph for the Thermoplastic Tape Lay-up

To apply the developed algorithms to the time-optimal motion planning, the considered technological task must be presented in the form of the directed graph. Relevant technique is described in Section 3.1, which includes discretization of the redundant variable and creation of the graph vertices that are also referred to as the configuration cells. To generate the desired graph, the redundant variable (the positioner joint angle q_P) was discretized with the step 1°, which produced the initial graph with 57399 vertices that are arranged in 361 rows and 159 columns. For each vertex, to verify the admissibility of the corresponding task point, it was solved the robot inverse kinematics and it was also checked the collisions and the distance to singularities. Application of these constraints allowed reducing the number of the graph vertices down to 24089. More detailed information concerning the task graph generation is presented below.

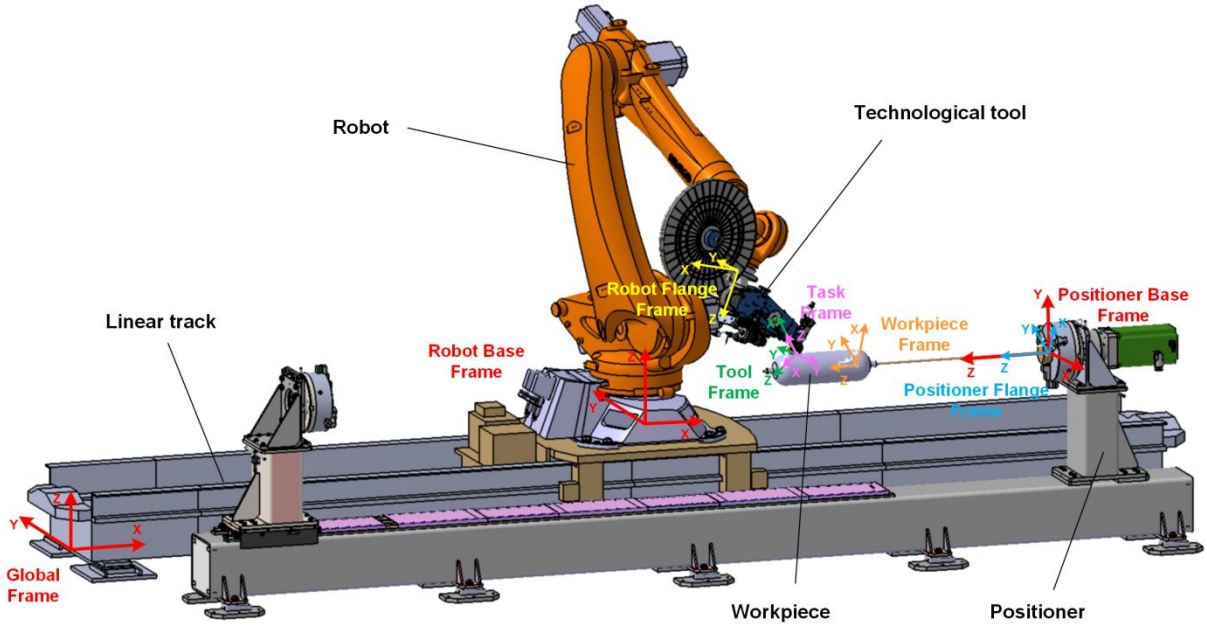


Figure 65 Robotic platform SPIDE-TP and arrangement of coordinate frames

The layout of the robotic platform used in this work is shown in Figure 65. Mutual location of the system components is defined by the 6×1 vectors presented in Table 16 where each line contains the Cartesian coordinates of the corresponding frame origins and Euler angles describing their positions and orientations with respect to the global coordinate system. For computational convenience, all these location vectors were also presented in the form of 4×4 homogeneous transformation matrices. It should be mentioned that in the table the robot base location is defined directly in the global frame (taking into account the linear track coordinate 3000 mm). Besides, mutual location of the task and

tool frames are defined in slightly different way compared to the Section 2.2, which is imposed by the industrial partner.

Table 16 Spatial parameters defining the emplacement of the robotic system components

Transformation type	Position coordinates			Orientation angles		
	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
Robot base / Global frame	3000.00	0.00	557.00	0.00	0.00	0.00
Positioner base / Global frame	4096.10	-1396.08	1089.90	-90.00	0.00	90.00
Robot tool / Robot flange	-327.78	-326.77	300.80	4.46	63.33	3.21
Workpiece / Positioner flange	0.00	0.00	1077.18	0.00	0.00	0.00
Task frame / Robot tool	0.00	0.00	0.00	0.00	-90.00	180.00

Table 17 Geometric parameters of transformation describing the robot and positioner kinematics

Transformation type	Position coordinates			Orientation angles		
	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
Robot frame 1 / Robot base	0.00	0.00	675.00	0.00	0.00	q_{R1}
Robot frame 2 / Robot frame 1	350.00	0.00	0.00	0.00	q_{R2}	0.00
Robot frame 3 / Robot frame 2	1350.00	0.00	0.00	0.00	q_{R3}	0.00
Robot frame 4 / Robot frame 3	1400.00	-41.00	0.00	q_{R4}	0.00	0.00
Robot frame 5 / Robot frame 4	0.00	0.00	0.00	0.00	q_{R5}	0.00
Robot frame 5 / Robot frame 6	0.00	0.00	0.00	q_{R6}	0.00	0.00
Robot flange / Robot frame 6	240.00	0.00	0.00	0.00	90.00	0.00
Positioner flange / Positioner base	0.00	0.00	0.00	0.00	0.00	q_P

The robot and positioner kinematic parameters are presented in Table 17 where $q_{R1} \dots q_{R6}$ and q_P denote corresponding joint coordinates. It should be mentioned that in contrast to other works where the kinematic models are based on the DH parameters; here another equivalent parameterization is used that is more convenient and computationally efficient. Definitions of the corresponding frames are presented in Section 2.2. Besides, it is worth mentioning that following the standard of the robot manufacturer, the manipulator configuration was defined using the indices S and T (KUKA, 2010) where S allows to avoid ambiguities in the first, third and fifth joint angles in a usual way while T

defines combinations of the angle signs allowing to move axes through angles greater than +180° or less than -180°. More details concerning the configuration indices are given in Table 18 and Table 19. For example, the entry S 'B110' defines the robot configuration with shoulder forward, elbow up and wrist down. Similarly, the entry T 'B001010' means that the angles q_{R2} and q_{R4} are negative while the others are all positive. In this work, the continuity of the composite laying process did not allow to change the shoulder and elbow postures during the robot motion while the wrist posture might be adjusted. For this reason, only two values of the S index and four values of T index were used: 'B010', 'B110' for the index S and 'B000010', 'B001010', 'B100010' and 'B101010' for the index T.

Table 18 Definition of the configuration index S for 6-axis KUKA robot (Status index)

Value	Bit 2	Bit 1	Bit 0
0	Wrist down	Elbow down	Shoulder forward
1	Wrist up	Elbow up	Shoulder backward

Table 19 Definition of the configuration index T for 6-axis KUKA robot (Turn index)

Value	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	$q_6 \geq 0°$	$q_5 \geq 0°$	$q_4 \geq 0°$	$q_3 \geq 0°$	$q_2 \geq 0°$	$q_1 \geq 0°$
1	$q_6 < 0°$	$q_5 < 0°$	$q_4 < 0°$	$q_3 < 0°$	$q_2 < 0°$	$q_1 < 0°$

When generating the task graph, the robot kinematic model was used first, which allowed to check accessibilities of the task points and to evaluate distances to manipulator singular configurations. At this stage, all 57399 vertices were verified and some of them were eliminated from the task graph, if one of the following conditions was satisfied: (i) the inverse kinematic solution did not exist, (ii) the joint coordinates were outside of the limits or (iii) the Jacobian condition number was more than 6. The task graph after applications of kinematic constraints is shown in Figure 66a, it includes 30229 admissible vertices. Then, the difficult configurations were eliminated which allowed reducing the number of the admissible vertices down to 28907, as shown in Figure 66b. Finally, the rough/fine collision constraints were applied using Matlab codes and commercial CAD system CATIA, and the final task graph was generated with 24089 admissible vertices (see Figure 66c).

In order to use collision detection capabilities of CATIA, a detailed 3D model of the SPIDE-TP platform was created and "DMU Kinematics" workbench was activated. Then, using the visual interface of the "Check Clash" module, two types of interferences between the system components were selected: 1) interference between the technological tool and robot/positioner; 2) interference between the robot links and other components of the workcell. Further, by activating the sensors on the "Kinematic Simulation" panel and executing this application, the collisions between the components were detected, if any. Corresponding areas are highlighted in yellow in the CAD graphical window and relevant sensor value (interference flag) becomes non-zero, as shown in Figures 67 and 68. Finally, the collision check data for all vertices were recorded to an Excel file describing the task graph that was used further to generate optimal motions of the robot and positioner.

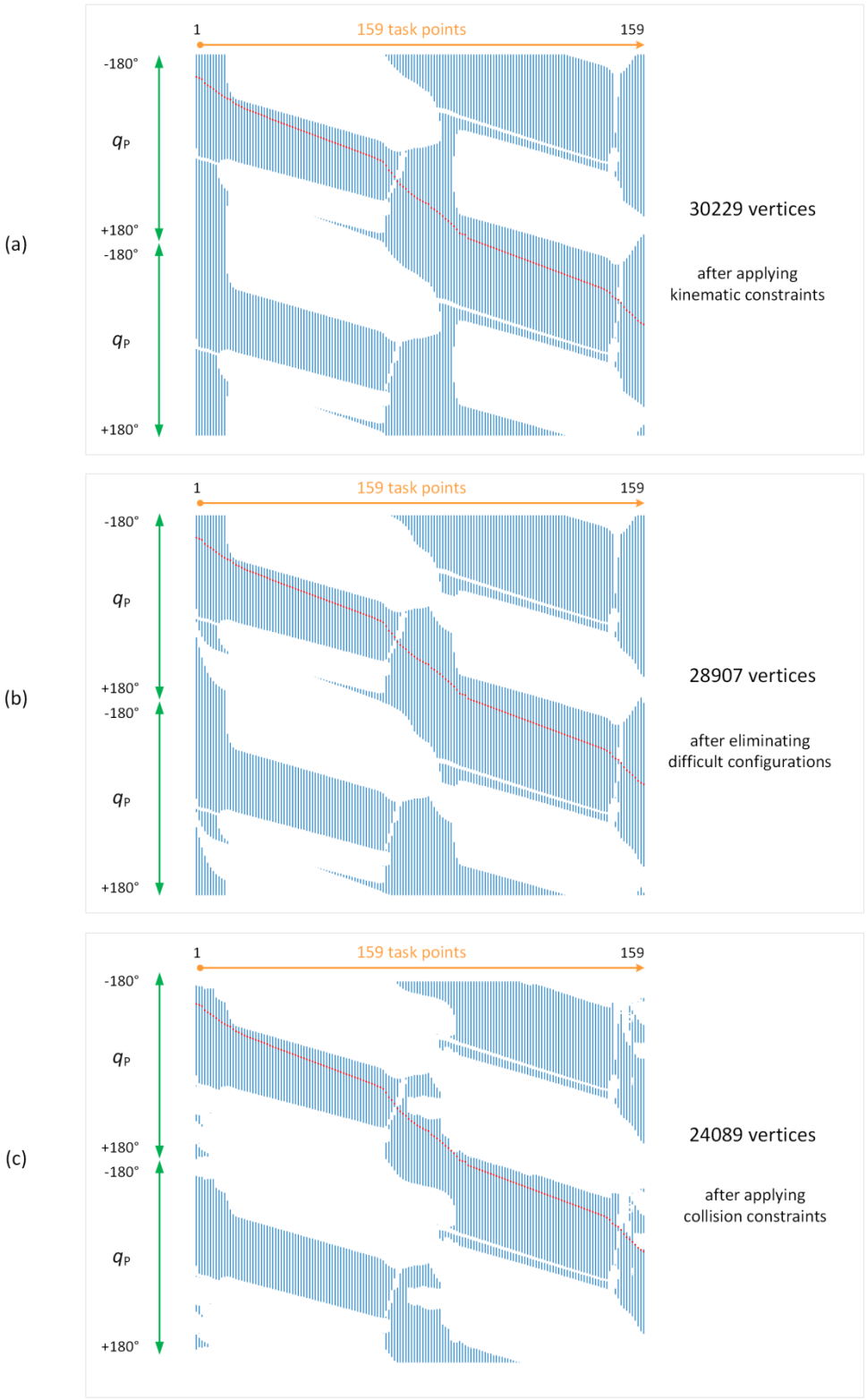


Figure 66 Task graph evolution after applications of kinematic and collision constraints ("blue"– admissible vertices; "white" – inadmissible vertices; "red" – admissible path).

It should be mentioned that generation of such graph required quite a lot of computing time. In particular, it took about one minute to generate the graph taking into account the kinematic constraints and to eliminate difficult configurations. However, the rough/fine collision test in CATIA required more than 10 hours.

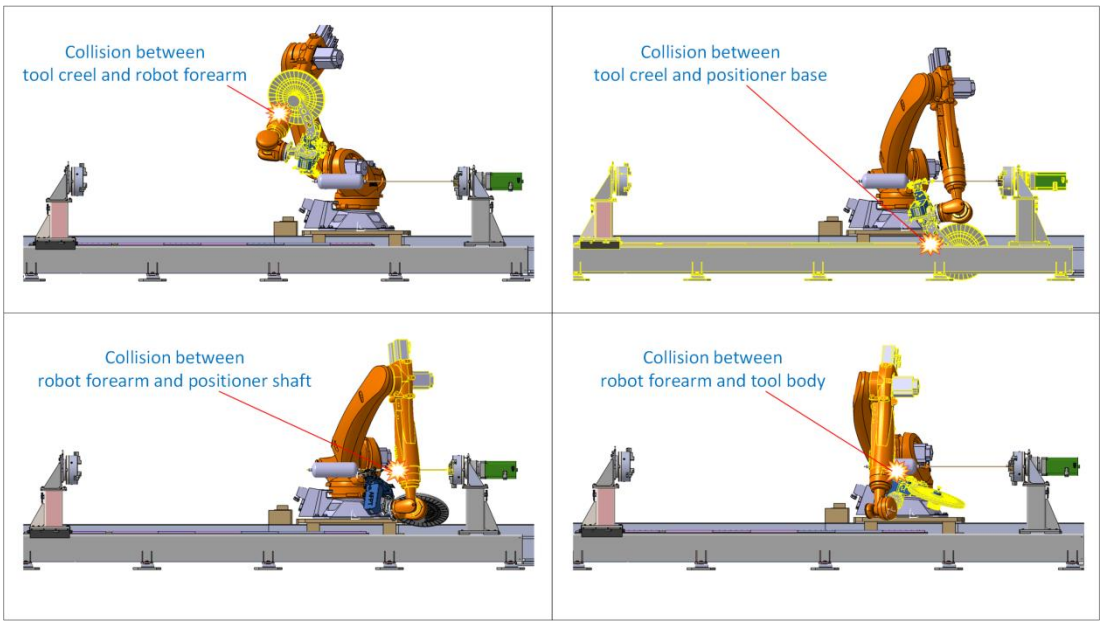


Figure 67 Collisions between different components of robotic platform SPIDE-TP

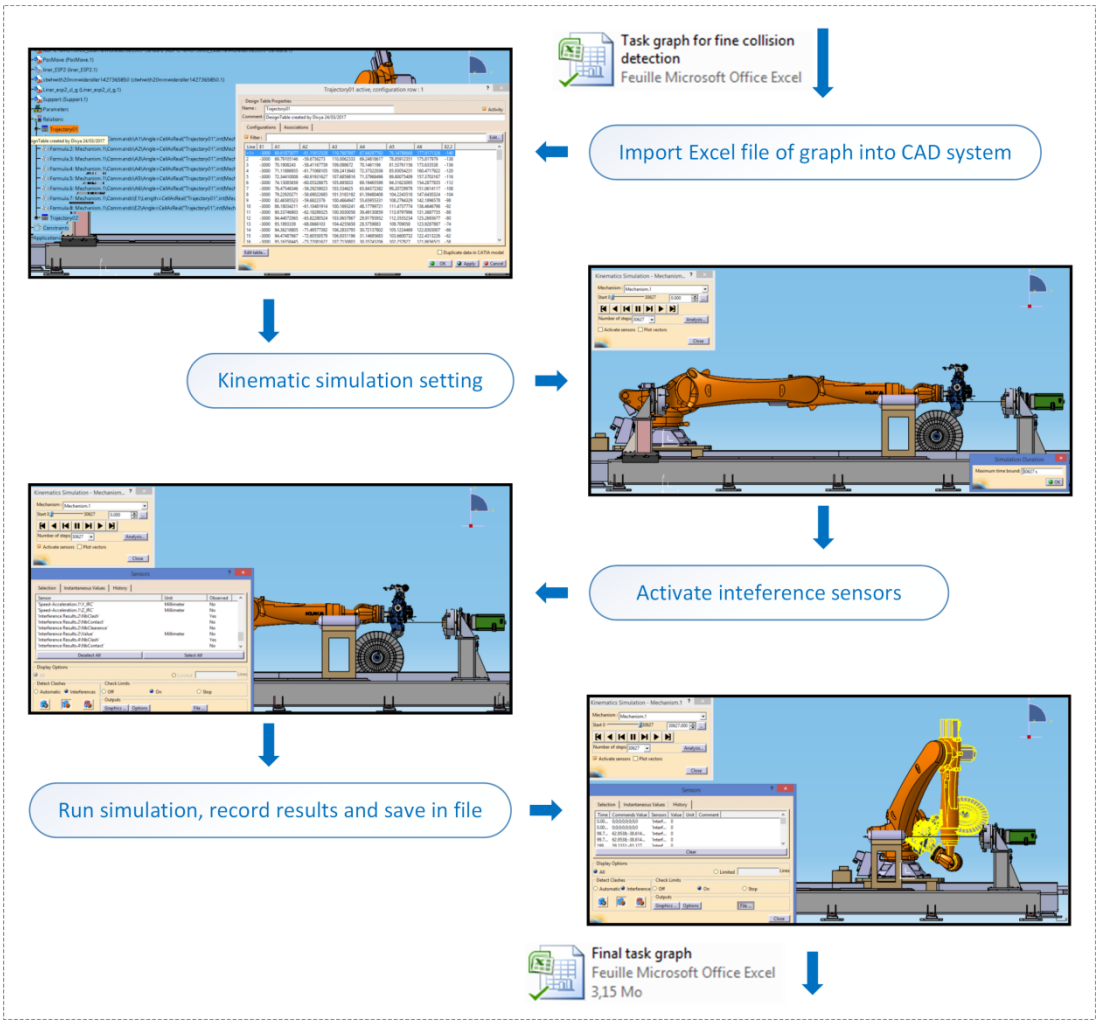


Figure 68 Procedure of fine collision detection in CATIA environment

4.2.2 Generation of Time-optimal Coordinated Motions of Robot and Positioner

The obtained task graph contains all possible postures of the robotic manipulator and positioner, which ensure following the laying path without collisions, without violation of the joint limits and keeping safe distance to the singularities. This graph includes 24089 vertices and is organized as a matrix with 361 rows and 159 columns where some of the cells are empty. Using the task graph and applying the developed algorithms presented in Chapter 3, it is possible to generate the desired time-optimal motions of the robotic system for implementing the given laying path.

While generating the time-optimal motions, the algorithms require some additional constraints defining the actuators capacities to move the manipulator and positioner. These capacities are expressed in the form of the maximum allowable velocity and acceleration for each actuated joint that are presented in Table 20. It is worth mentioning that in practice the acceleration constraints are defined in the robot controller indirectly, via the special parameter (acceleration time) that it was assumed here to be equal to 0.25 s but it can be modified by the user depending on the payload. For this reason, during the motion planning the acceleration constraints were applied in "soft" manner, with different acceleration times. Another reason justifying this operation is related to approximate evaluation of the acceleration in the optimization algorithm that is based on the finite difference technique that uses three time instances only, in accordance with expression (103). As it has been observed, strict application of the acceleration constraints may produce some undesired peaks on the motion profiles that disappear if the acceleration constraints were slightly weaken (relaxed). However, detailed analysis of the final trajectories generated in such way shows that the obtained motions are smooth enough and satisfy all considered constraints.

Table 20 Joint limits and maximum velocities/accelerations for the robot and positioner

AXIS	Joint limits	Maximum velocity	Maximum acceleration
Robot axis #1	[-185°; 185°]	105 deg/s	420 deg/s^2
Robot axis #2	[-140°; -5°]	101 deg/s	404 deg/s^2
Robot axis #3	[-120°; 155°]	107 deg/s	428 deg/ s^2
Robot axis #4	[-350°; 350°]	136 deg/s	544 deg/ s^2
Robot axis #5	[-122°; 122°]	129 deg/s	516 deg/ s^2
Robot axis #6	[-350°; 350°]	206 deg/s	824 deg/ s^2
Positioner axis	[-180°; 180°]	142 deg/s	284 deg/s^2

To find the desired time-optimal motions, the developed DP-based algorithm with the discretization $\Delta q_P = 1°$ was applied first and the trajectories were smoothed further using the cubic spline approximation (Algorithms 2 and 4, see Section 3.3). It took about one hour of computations for searching of the best path on the task graph and 0.1 sec for the smoothing. It is worth mentioning that further reduction of the discretization step is extremely time consuming, so the Algorithm 3 (with the local optimization) was not applied for this practical problem where the collision check requires more than 10 hours even for $\Delta q_P = 1°$. Nevertheless, the results were quite acceptable for practice. In particular, for the generated trajectory the robotic system motion time is about 4.0 sec, which is much better compared to 14.0 sec that our industrial partner got using the software package Composicad.

The generated optimal trajectories are presented in Figure 69, which contains the time profiles for all joint coordinates and corresponding velocities estimated using the moving window technique. As follows from detailed analysis, the obtained profiles are quite smooth and satisfy both the joint limits and the velocity/acceleration constraints. It is worth mentioning that at each time interval one of the constraints is active, i.e. either the velocity or acceleration reaches its maximum/minimum value for at least one of the joint coordinate. Besides, it was observed that after the trajectory smoothing, some segments of the velocity profiles were slightly over their limits (for the positioner and axis #4 of the robot). However, such minor violations of the constraints in Figure 69 are usually acceptable in practice and are compensated by the robotic system controller. On the other hand, these violations can be easily eliminated by simple modifications of the trajectories, by means of the motion time extension between the corresponding task points.

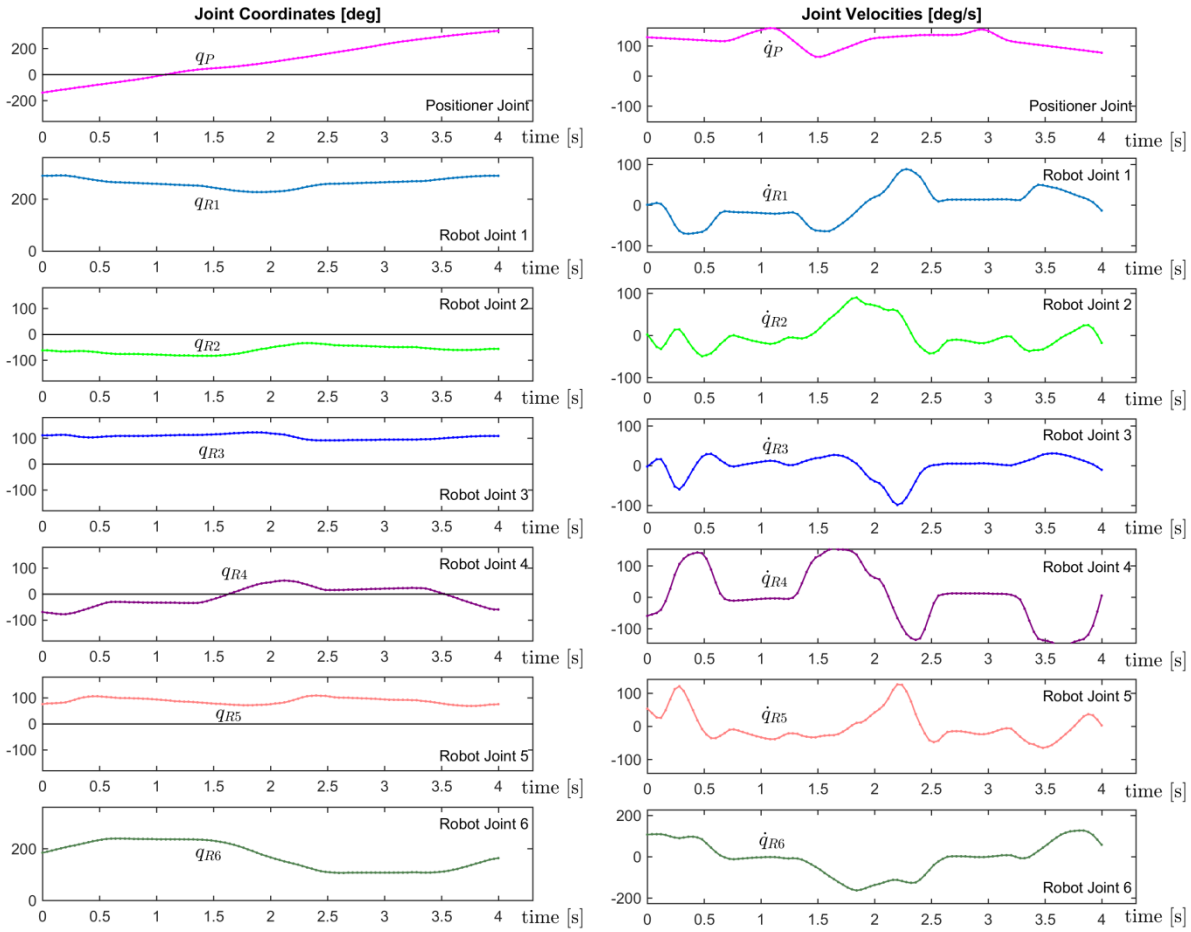


Figure 69 Profiles of time-optimal motion for thermoplastic fiber covering of the high-pressure vessel

It is worth mentioning that for the considered technological problem, the task graph has rather complicated topology that includes some bottleneck areas. In particular, it can be seen a very narrow gap between the cells corresponding to the task points #89 and #90 (Figure 70). So, there is very limited number of the graph vertices to be included in the optimal path for this area. From engineering point of view, this segment of the laying task is very difficult to be implemented without collisions. In fact, as follows from the simulation of the time-optimal motion in CATIA V5, the technological tool moves very close to the robot forearm during the transition between the task points #89 and #90 (see Figure 70, where the robot wrist was set to be invisible for better view).

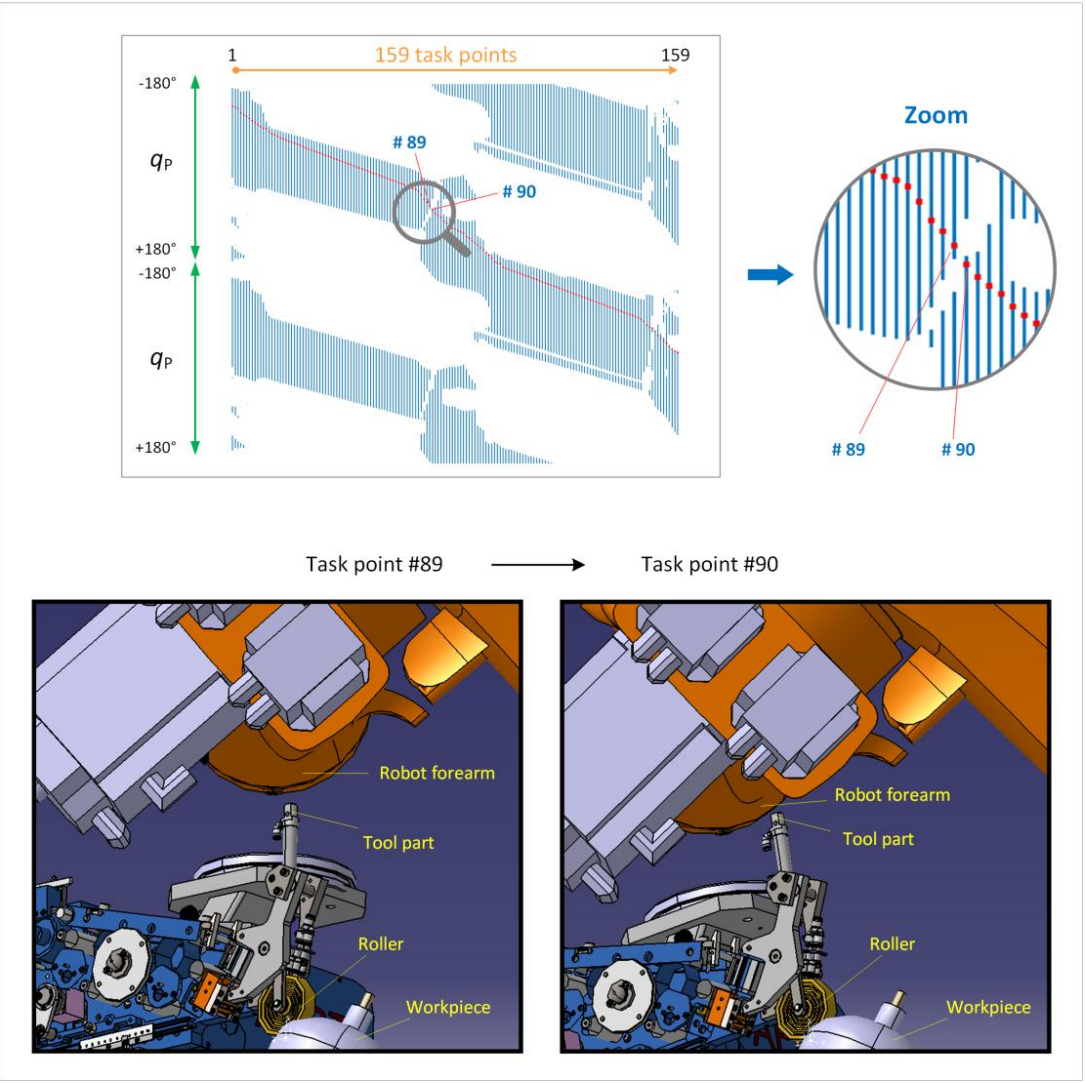


Figure 70 Bottleneck areas at the task graph (too close to collisions)

In addition, it is reasonable to compare the obtained results with ones generated using another technique that was developed for the laser cutting applications (Pashkevich et al., 2004) and assumes constant Cartesian speed of the robotic tool with respect to the workpiece. In the frame of this work, it corresponds to the constant laying speed of the thermoplastic fiber. Using the latter assumption and relevant objective function (maximum weighted joint coordinate increment between the task points), the motion planning problem was solved in different way. The obtained results are presented in Figure 71, which clearly shows advantages of the developed technique that allows reducing the motion time of the robotic system from 8.4 sec down to 4.0 sec.

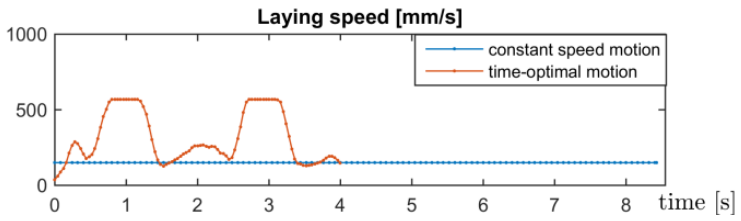


Figure 71 Laying speeds for the motions generated using the developed and known algorithms

It is also worth mentioning that the laying task considered here deals with just one circuit of the laminate. In practice, the laying process includes numerous circuits and robot/positioner motions should be repeated many times, with some small adjustment of the tool path. For this reason, to meet the demand of the industrial partners, it was slightly modified the final step of the Algorithm 2 that deals with selection of the final robot configuration from the candidates belonging to the last column of the task graph matrix. Initially, it was selected to minimize the motion time only. After the modification, the selection of the final configuration was based on the combined criteria, which takes into account both the motion time and the difference between the initial and final robot configurations (by applying the weighted sum method). This modification allowed us to achieve almost identical robot configurations at the circuit beginning/end while it yielded very small increasing of the system motion time, by 0.09 sec only. It should be noted that the time-optimal profiles presented in Figure 69 were generated using the modified version of the Algorithm 2. So for this motion, the initial and final values of the robot joint coordinates are very close to each other.

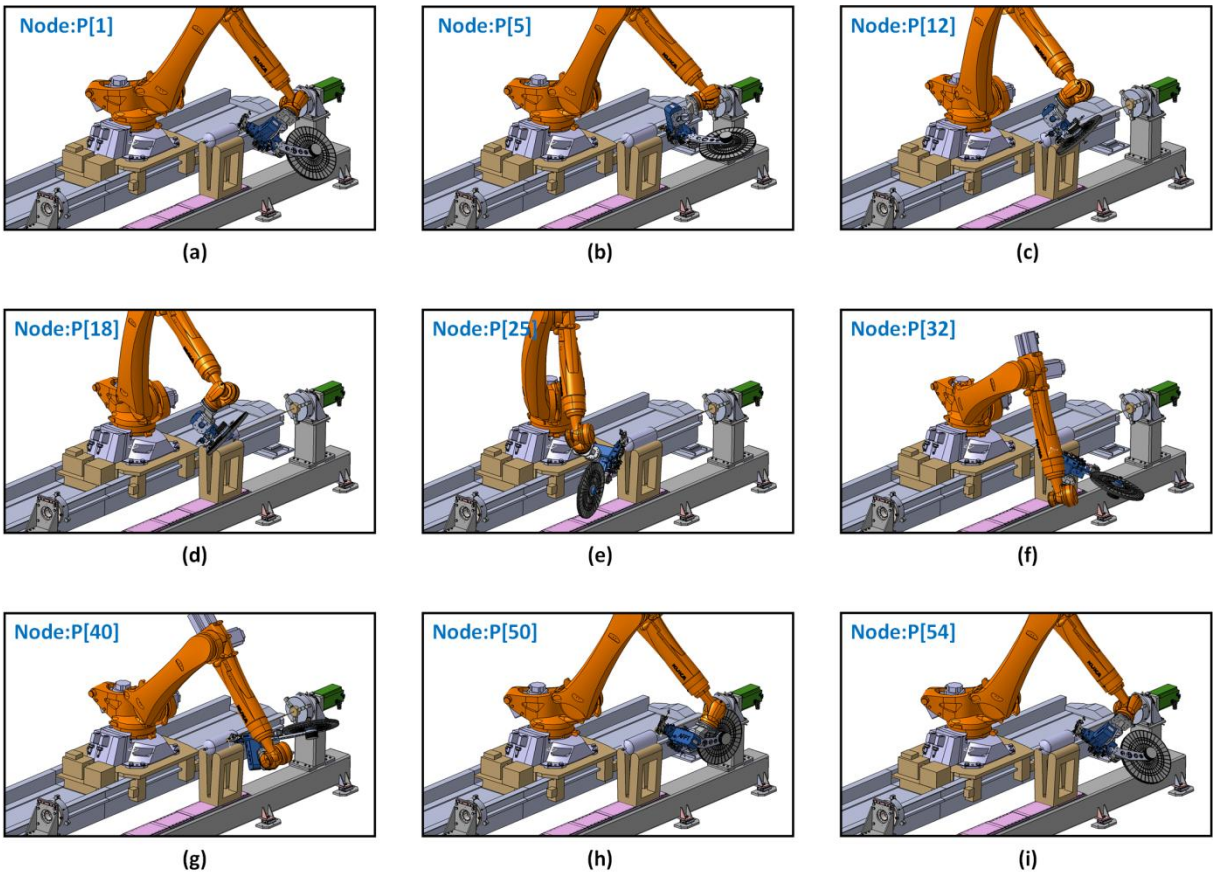


Figure 72 Configurations of the robotic system for the time-optimal trajectory simulated in CATIA

To verify correctness of the obtained trajectory, it was carried out a number of simulations in a 3D visual environment of CATIA. It was used the same 3D model of the SPIDE-TP platform as for the collision test at the task graph generation stage (see Figures 67 and 68). To simulate the obtained motion, the "DMU Kinematics" workbench was activated. Then, using the visual interface of the "Simulation with Laws" module, the robotic system model was actuated using the Excel file describing the sequential configurations of the manipulator and positioner. Several pictures captured from this simulation are presented in Figure 72, which shows the robotic system configurations corresponding to the trajectory nodes: #1, #5, #12, #18, #25, #32, #40, #50 and #54. Relevant video is

available at the URL https://www.youtube.com/watch?v=ioGFXuV2gDU. As follows from the relevant analysis, the simulated motions of the robot and the positioner are well coordinated and meet the technological requirements of the thermoplastic lay-up process. Besides, the initial and final locations of the technological tool are close to each other (see Figures 72a and 72i), which allows in practice easily repeating the same laying motion several times. However, it should be mentioned that this simulation allowed to verify correctness of the obtained laying path only, while evaluation of the laying speed was not possible in this way because 3D animation provided by CATIA does not take into account the time information concerning the examined trajectory.

Hence, the developed algorithm allowed us to generate the desired time-optimal trajectory that should be further implemented by the robotic system controller. Relevant data were presented in the form of the Excel file containing the robot and positioner joint coordinates and corresponding time instances. Using this data, there were also calculated the robot configuration indices and 6×1 location vectors describing the technological tool position and orientation in the global frame, which are required for the robot programming.

4.3 OPTIMAL MOTION IMPLEMENTATION IN ROBOTIC SYSTEM

To implement the obtained time-optimal motion to the SPIDE-TP platform, the trajectory was programmed using the KRL language for KUKA robotic system controllers. Then, an experimental evaluation on the factory floor was carried out to verify its applicability.

4.3.1 Motion Implementation using KRL Robot Programming Language

The time-optimal trajectory obtained in the previous section is presented as a sequence of the robot/ positioner joint coordinates and corresponding time instances defining the motion profiles. However, most of modern robot controllers neglect the time information and apply their own built-in algorithms to generate trajectory using only the sequence of the desired end-effector locations, which can be defined by either the joint coordinates or the end-effector position/orientation. Let us give some details concerning typical motion commands implemented in the robot controller and some corresponding analysis (investigation).

(a) *Description via the joint coordinates*

P = {A1 10, A2 -80.6, A3 -50, A4 0, A5 14.2, A6 0, E1 -3000, E2 200}

(b) *Description via the end-effector location*

P = {X 12.3, Y 100.0, Z 50, A 9.2, B 50, C 0, E1 -3000, E2 200, S ’B010’, T ’B1010’}

Figure 73 Description of the robotic platform configuration in KR C4 controller

In the control system KR C4 that is used in SPIDE-TP platform, the configurations of the manipulator and external mechanisms (positioner and liner track) are described by either 8 joint coordinates or 10 numbers defining the end-effector position/orientation, the positioner rotation angle,

the linear track displacement as well as the manipulator configuration indices. An example of such descriptions is presented in Figure 73. Here, the first line straightforwardly defines the manipulator joint angles denoted as {A1, A2… A6}, the linear track displacement {E1} and the positioner rotation angle {E2}. In contrast, the second line uses an alternative description of the manipulator posture via the Cartesian coordinates {X, Y, Z} and Euler angles {A, B, C} of the end-effector as well as via the manipulator configuration indices {S} and {T} presented in Tables 18 and 19. It is worth mentioning that in our experimental study the linear track coordinate E1 was constant and it was set to its optimal value equal to -3000 mm (as justified in Section 4.1).

To describe motions of the robotic manipulator and external mechanisms, the control system KR C4 uses the programming language KRL. In the frame of this language, there are two possible ways to implement the time-optimal trajectory obtained in Section 4.2. The first of them is based on the conventional robot motion commands PTP/LIN, while the second uses SPLINE blocks. The command PTP implements the fastest linear-path motion in the joint space taking into account the velocity/acceleration constraints for each actuated coordiante. It is clear that in the Cartesian space this command produces a curved path, so it is not reasonable to use it for implementation of the desired motion of the robotic system for the compsite tape laying. The command LIN implements the fastest linear-path motion in the Cartesian space taking into account the velocity/acceleration constraints for each actuated coordiante as well as the Cartesain velocity/acceleration constraints for the robot end-effector. This command produces a linear segment in the Cartesian space, so it can be used for implementation of the desired motion assuming that the piecewise linear interpolation of the composite lay-up path is acceptable from engineering point of view.

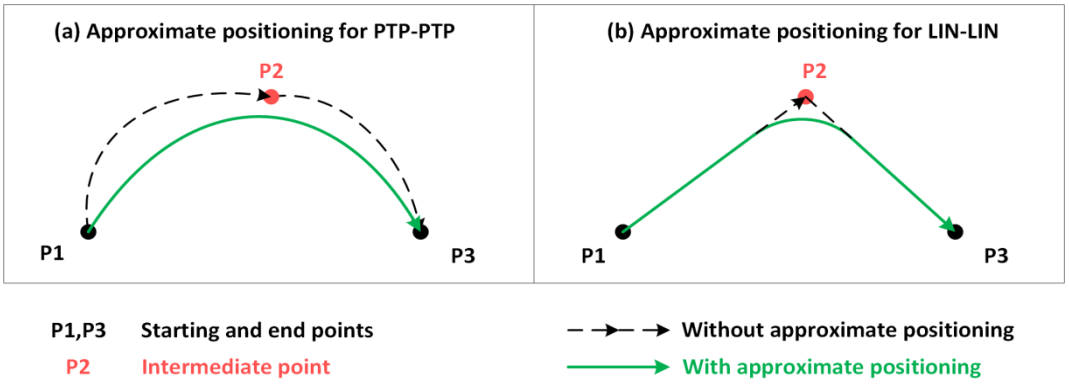


Figure 74 Approximate positioning for PTP-PTP and LIN-LIN motion sequences

It should be stressed that, the basic versions of the PTP and LIN commands implement “start-stop” motions that includes three sections (accelerating, constant speed and decelerating) with zero speeds at the beginning and the end. It is clear that such “start-stop” implementation is not accepetable for the considered composite lay-up technology where the end-effector Cartesian speed should be as high as possible and the motion stops at the nodes are not permited. To avoid this diffculty, the programming language KRL allows user to apply so-called approximate positioning at the trajectory end points, which is based on the superposition of deceleration and acceleration sections of the subsequent motion segements. This technique yields non-stop continuous motions for which the end-effector passes the nodes neighbourhood only instead of visiting the nodes exactly (see Figure 74), so it can be hardly accepted for the considered tape laying task. To activate the approximate positioning in KRL language, the robot motion command must include a special suffix, such as C_PTP for the PTP-PTP sequence or C_DIS, C_ORI or C_VEL for the LIN-LIN sequence. The latter define the type of switching criteria

that can be based on the translational/ rotational distance to the target node or the velocity level at the beginning of the approximation. The overlapping degree for the approximate positioning is defined via the special system variable $APO.CPTP or $APO.CVEL. For example, the setting $APO.CPTP=100 or $APO.CVEL=100 leads to the complete 100% overlapping of the previous deccelerating and the subsequent accelerating sections (KUKA, 2010).

Durations of the PTP and LIN motions is computed by the KR C4 controller online, using the velocity and acceleration constraints in the joint space (for all axes) and similar constriants in the Cartesian space for the end-effector motion. The default values of these constraints are defined in special tables, but they can be customized by the user by means of the dedicated commands of the KRL language. For instance, the commands $VEL_AXIS [1]=50 and $ACC_AXIS [1]=75 define the maximum velocity and acceleration for the axis#1 on the level of 50% and 75% of their default values correspondingly. Similarly, the commands {$VEL.CP=2, $VEL.ORI 1=400, $VEL.ORI 2=400} define the maximum end-effector translational speed as 2 m/s and the maximum rotational velocities for the orientation angles as 400°/s. Related commands {$ACC.CP, $ACC.ORI 1, $ACC.ORI 2} can be also used to set maximum translational/rotational accelerations in the Cartesian space.

Using the velocity and acceleration constraints, the KR C4 controller generates trapezoidal velocity profiles either in the joint or Cartesian space and applies the superpositioning to avoid the stops at the intermediate task points (see Figure 42). It is also worth mentionning a very important particularity of the motion generation in the robot controller, where a sequence of very short segments cannot be executed with maximum allowable speed. In this case, the velocity profiles have the triangular shapes whose duration is limited by the controller capacity (it is defined in the control software as a constant). So, the resulting average speed for such segments is essentially lower than the expected level, which is defined by the dedicated commands. Hence, it is necessary to avoid very short segments while implementing the time-optimal trajectory via the LIN-LIN sequences (another disadvantage of such implementation is related to the approximate positioning).

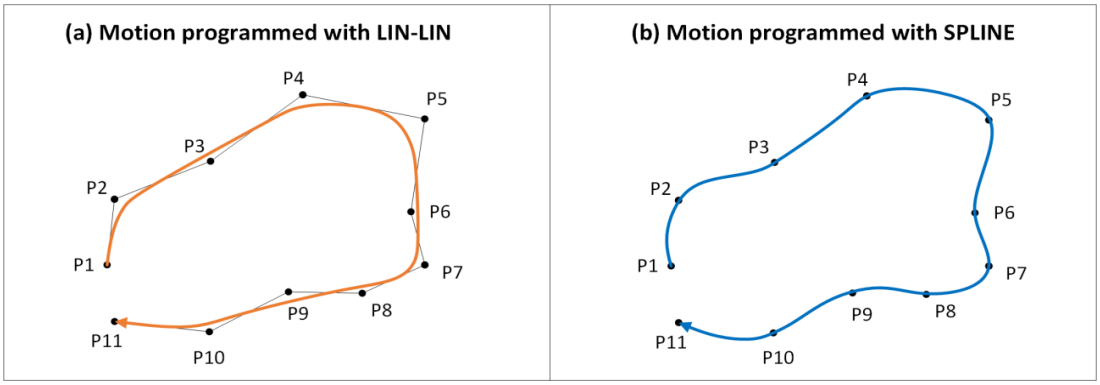


Figure 75 Curved path implementation using LIN-LIN sequence and SPLINE block

An alternative way to implenment the desired motion is based on the SPLINE block commands. Their execution employs the quintic polynomial interpolation between the trajectory nodes, instead of the first-order one that is used for LIN or PTP. Besides, compared to the approximated LIN-LIN sequences, here the end-effector passes the given nodes exactly and without stops (see Figure 75). For this reason, the spline based technique is very attractive for implementing of complex paths. In KRL programming language, a spline motion is defined as a set of several individual segments. The segments are grouped together to form the overall motion in a so-called spline block, which is

executed by the robot controller as a single motion. The SPLINE block is limited by the SPLINE and ENDSPLINE statements and contains a sequence of commands SPL describing the trajectory nodes. In addition, the desired the motion time for each segment can be defined indirectly by setting the system variables {$VEL.CP, $VEL.ORI 1, $VEL.ORI 2} or in the direct way, using special TIME_BLOCK containing the travelling times between the nodes. An example of such descriptions is presented in Figure 75.

Inside of the SPLINE block it is possible to use the SPL commands both outside and inside of the TIME_BLOCK. If the SPL command is located outside of it, the motion time between the current and target nodes is computed using the maximum velocity/acceleration settings provided by the relevant system varibables, similar to the LIN-LIN sequence. Otherwise, if the SPL command is inside of the TIME_BLOCK, the motion time between the nodes is defined via the statement TIME_BLOCK PART, as a percentage of the total spline motion time that is specified at the TIME_BLOCK END statement. For example, for the KRL program presented in Figure 76 the motion from the node P1 to P54 is executed in 4.7 sec, while for the segment [P1, P2] the motion time is set as 3.0% of the total one, which corresponds to 0.14 sec. It should be also noted that this program also includes the SPL commands before and after the TIME_BLOCK that allow achieving the desired velocity/acceleration values at the spline motion beginning and the end.

```
SPLINE
        SPL P0
        SPL P1
        TIME_BLOCK START
                SPL P2
                TIME_BLOCK PART = 3.0
                SPL P3
                TIME_BLOCK PART = 2.3
                …
                SPL P54
                TIME_BLOCK PART = 1.9
        TIME_BLOCK END = 4.7
        SPL P55
ENDSPLINE
```

Figure 76 Description of curved path using spline interpolation

Table 21 Comparison of motion commands of KR C4 controller for curved path implementation

Motion commands	**PTP-PTP**	**LIN-LIN**	**SPLINE** block
Interpolation space	Joint space	Cartesian space	Cartesian space
Positioning	Approximate	Approximate	Exact
Time assignment	Speed/Acceleration settings	Speed/Acceleration settings	Speed/Acceleration & Time settings

Hence, as follows from the comparison study presented above and summarized in Table 21, the spline interpolation is the most attractive for implementation of the time-optimal motions generated by the developed algorithms. Its main advantages are the capability to implement continuous motion without stops, exact positioning at each task location as well as possibility to assign the desired motion

time for each trajectory segment. Besides, short path segments are also acceptable for the SPLINE blocks available in the KR C4 controller since they do not cause the velocity reduction, in contrast to the LIN-LIN sequences.

4.3.2 Experimental Evaluation of the Implemented Time-Optimal Motions

To evaluate the implemented time-optimal trajectory generated by the developed algorithm, the obtained motion was programmed using the SPLINE block. Relevant data describing the robot and positioner joint coordinates (trajectory nodes) and corresponding time instances were imported from the .xls file into a text editor and presented in the form combining the end-effector location and the joint coordinates for the external axes using the format shown in Figure 73. The latter yielded 159 trajectory nodes denoted in the KRL program as P[1],…P[159], which were used as the arguments in the SPLINE block commands SPL P[1],…SPL P[159]. Besides, there were defined the HOME configuration and the starting point P0 preceding the spline motion. There were also set the system variables defining the maximum allowable speeds/accelerations and added some technological commands for the control of the thermoplastic tape feeding, tensioning and heating (see Figure 77). The obtained text file containing the program in the KRL language was saved with the .src extension that is compatible with KR C4 controller. Finally, the robotic system motion program was downloaded to the controller and executed. It is also worth mentioning that it was discovered after the first experiments that some of the nodes could be eliminated without any infulence on the quality of the thermoplastic tape laying. This allowed us to reduce the number of the trajectory nodes down to 54, which were used in further experimental study.

The experimental study included execution of the above presented KRL program and evaluation of the corresponding motions of the robotic manipulator and positioner. For safety reasons, the robotic system was run with slightly reduced speed, which was achieved by setting the maximum velocities of the actuated axes at the level of 75% of the maximum values. Besides, the workpiece was not mounted on the positioner flange to avoid unexpected collisions. The system motions were registered by a video camera, and the total motion time was estimated using a timer. The experiment shows that even for these settings it took only 7.2 sec to implement the desired laying path, while the Composicad software produced the trajectory with the execution time more than 14.0 sec. It is clear that the motion time for the studied trajectory can be easily reduced down to ~5.4 sec by simple increasing of the maximum velocity settings up to 100%. Nevertheless, it is still higher compared to the 4.0 sec corresponding to the theoretical time-optimal trajectory presented in Section 4.2 (this issue is discussed in details below).

Several photos from this experiment are presented in Figure 78, which shows the robotic system configurations corresponding to the trajectory nodes: #1, #5, #12, #18, #25, #32, #40, #50 and #54. Relevant video is available at the URL https://www.youtube.com/watch?v=9nvdlzpJHFE. As follows from the related analysis, the implemented motions of the robot and the positioner are well coordinated and rather fast. Besides, the initial and final locations of the technological tool (Figures 78a and 78i) are close to each other, which allows in practice easily repeating the same laying motion several times. However, there are several segments where the motion smoothness should be impoved if the allowable velocities are increased up to 100% of their maximum values.

```
; ---------------------------------------------- Initialization Section ----------------------------------------------
HOME = {A1 0.00, A2 -120.00, A3 135.00, A4 0.00, A5 75.00, A6 90.00, E1 -3000.00, E2 0.00}
PTP HOME
$VEL.CP=2              ; Setting maximum Cartesian velocity at 2m/s
$VEL.ORI1=400          ; Setting maximum swivel velocity at 400°/s
$VEL.ORI2=400          ; Setting maximum rotational velocity at 400°/s
$ACC.CP=5              ; Setting maximum Cartesian acceleration at 5m/s^2
$ACC.ORI1=300          ; Setting maximum swivel acceleration at 300°/s^2
$ACC.ORI2=300          ; Setting maximum rotational acceleration at 300°/s^2
$APO.CDIS=5            ; Approximate positioning setting to 5mm
; ---------------------------------------------- Trajectory Points Definition ----------------------------------------------
P[1] = {X 19.62, Y 17.12, Z 1027.28, A 41.11, B 80.86, C 90.13, E1 -3000.00, E2 -138.89, S 2, T 2}
P[2] = {X 33.61, Y 10.09, Z 1028.84, A 16.70, B 78.18, C 47.87,  E1 -3000.00, E2 -120.30, S 2, T 2}
…
P[54] = {X 13.42, Y 23.68, Z 1027.53, A 60.42, B 80.95, C 106.93, E1 -3000.00, E2 2.01, S 2 , T 2}
; ---------------------------------------------- Main Section ----------------------------------------------
rTapeFeed("on","fly")        ; Tape feeding system activated
rTapetension("on","fly")     ; Tape tensioning system activated
rlaserstandby("on","all")    ; Laser heating system ready
rlaserbeam("on","all")       ; Laser beaming
rtapeCut("off","all")        ; Cutter inactivated
; ……
P0 = {X 19.62, Y 50.00, Z 1027.28,  A 41.11, B 80.86, C 90.1., E1 -3000.00, E2 -138.90, S 2, T 2}
LIN P0 C_DIS
SPLINE
        SPL P[1] WITH $VEL.CP=0.08        ; Spline motion from P0 to P[1] with speed 0.08m/s
        SPL P[2] WITH $VEL.CP=0.18        ; Spline motion from P[1] to P[2] with speed 0.18m/s
        …
        SPL P[54] WITH $VEL.CP=0.15       ; Spline motion from P[53] to P[54] with speed 0.15m/s
ENDSPLINE
; ……
rlaserbeam("off","fly")      ; Laser beaming stopped
wait sec wait_stop
rTapeCut("on","all")         ; Cutter activated
rStop()
rLaserStandby("off","all")   ; Laser heating system closed
PTP HOME
```

Figure 77 The KRL program of the time-optimal motion with speed settings at path segments

Detailed analysis of the non-smooth trajectory segments shows that such imperfect behavior was caused by some drawbacks of the Composicad software, which provided the developed motion planning algorithms with initial data (the Cartesian coordinates and orientation angles of the laying path nodes). In fact, the non-smooth segments are located at the connection of the cylinder and the domes (see Figure 60a) where ComposicaD allowed rather sharp variations of the orientations.

Although the given initial laying path and the generated time-optimal trajectory were slightly smoothed (see Figures 60b and 69), the obtained motion is still difficult for implementation using the capabilities of the KR C4 controller.

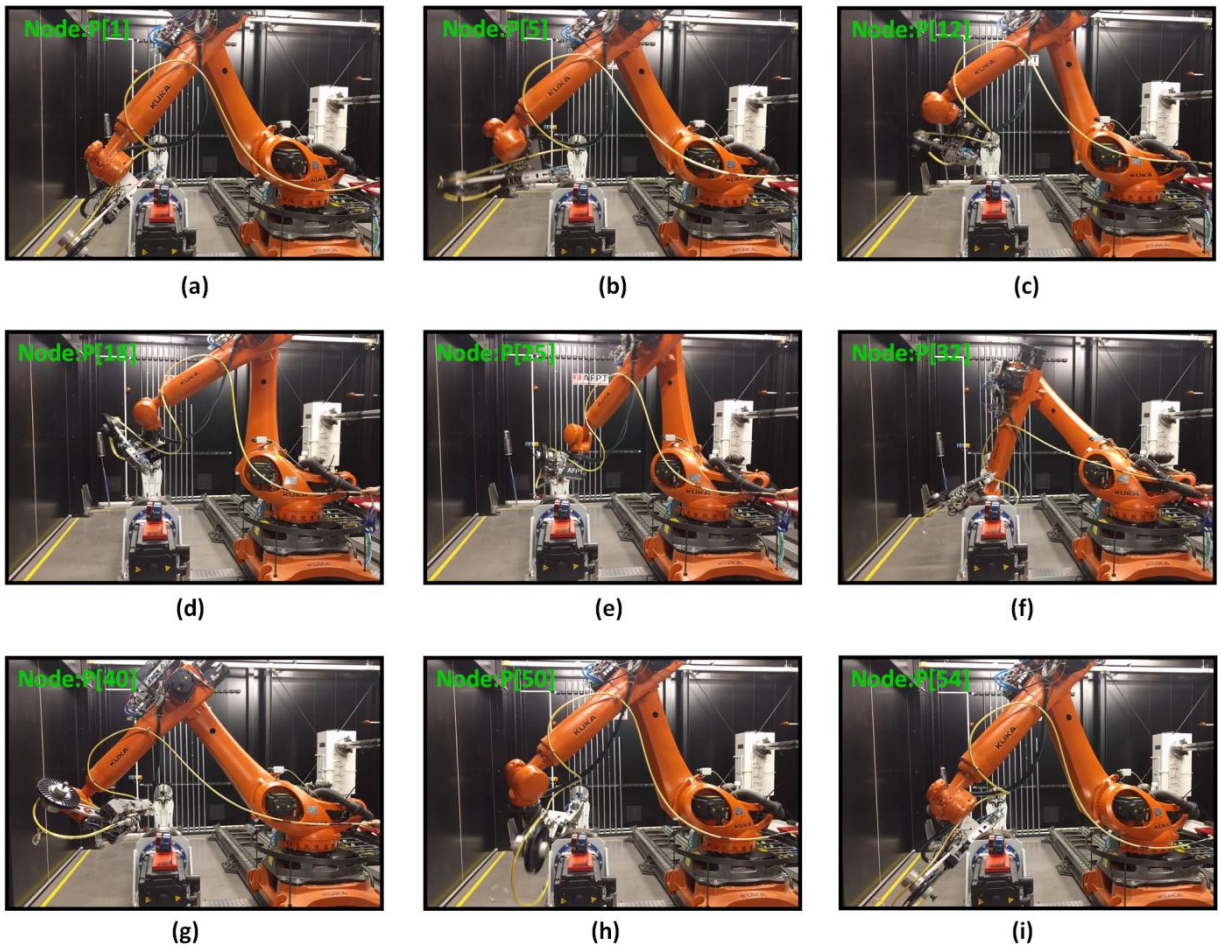


Figure 78 Configurations of the robotic system for the time-optimal trajectory implemented using SPLINE block with the speed settings

Another important issue to be discussed here is related to the robotic system motion time obtained in the experiments. The problem is that the duration of the implemented motion is essentially higher than the expected theoretical value. In fact, the developed motion planning algorithm yielded the trajectory with the motion time 4.0 sec, while the best value achieved in the experiments was 5.3 sec (using the 100% settings for the allowable speeds). The most probable reason for this motion slowdown is the difference between the nominal values of the maximum velocities/accelerations declared in the technical documents (used as the principle constraints for the motion planning) and their real values specified in the control software. In practice, ordinary users can manipulate with the percentage of the maximum velocity/acceleration for each actuated axis only, while they do not have access to setting their absolute values. Besides, some additional time is required for accelerating/deccelerating when entering/exiting the SPLINE block, within which the velocities/accelerations are maintained at the level corresponding to the time-optimal motion for a single laying circuit.

```
; ---------- Main Section ----------
rTapeFeed("on","fly")
rTapetension("on","fly")
rlaserstandby("on","all")
rlaserbeam("on","all")
rtapeCut("off","all")
; ……
P0 = {X 19.62, Y 50.00, Z 1027.28, A 41.11, B 80.86, C 90.10,   E1 -3000.00, E2 -138.90, S 2, T 2}
PE= {X 13.42, Y 50.00, Z 1027.53, A 60.42, B 80.95, C 106.93, E1 -3000.00, E2 2.01,      S 2 , T 2}
LIN P0 C_DIS
SPLINE
        SPL P[1]
        TIME_BLOCK START
                SPL P[2]
                TIME_BLOCK PART = 3.0
                SPL P[3]
                TIME_BLOCK PART = 2.3
                …
                SPL P[54]
                TIME_BLOCK PART = 1.9
        TIME_BLOCK END = 4.0
        SPL PE
ENDSPLINE
; ……
rlaserbeam("off","fly")
wait sec wait_stop
 rTapeCut("on","all")
 rStop()
rLaserStandby("off","all")
PTP HOME C_DIS
```

Figure 79 The KRL program of the time-optimal motion with time settings at path segments

To overcome the above mentioned difficulties, the KRL program of the time-optimal motion was modified using the TIME_BLOCK allowing to specify the motion time for each path segment. Besides, additional commands LIN P0 and SPL PE were included in the main section (before and after the TIME_BLOCK) in order to achieve the desired velocity/acceleration at the spline motion beginning and end (see Figure 79). This modified version of the KRL program will be tested soon by our industrial partner. There are a number of reasons to expect that this modification allows implementing the time-optimal trajectory with better precision and reaching the theoretical level of the motion time. On the other hand, detailed information concerning implementation of the SPLINE and TIME blocks in KR C4 controller is not available in technical literature, so some additional experiments will be conducted in future to reduce distance between the theory and practice.

4.4 SUMMARY

This chapter deals with industrial implementation of the developed motion planning method on the factory floor (for manufacturing of a high-pressure vessel). Using the proposed method, there were generated time-optimal smooth motions for the manipulator and positioner of the SPIDE-TP robotic platform allowing speeding up the thermoplastic tape lay-up process. Correctness of these motions was verified in 3D simulation environment of CATIA software package. Besides, it was created a program in KRL language implementing these motions and ensuring coordinated control of KUKA KR210 robot and AFPT workpiece positioner. There are presented results of industrial experiments carried out with this program, which confirmed smoothness of the manipulator/positioner movements and essential reduction of the time required for the thermoplastic tape lay-up process.

In more details, the contribution of Chapter 4 can be summarized as follows

(i) Manufacturing process preparation for the given technological task via defining and regularizing the task locations on the lay-up path provided by an industrial partner and optimization of the workcell layout by selecting the best robot base location.

(ii) Generation of the time-optimal coordinated motion for the robotic manipulator and the workpiece positioner that implement the given lay-up task. The obtained time-optimal trajectory allowed essentially reducing the total motion time of the robotic system required for one circuit of the composite laying process.

(iii) Implementation of the generated time-optimal motion. The obtained trajectories were used for creating a motion control program in KRL language, which was tested on the factory floor. The experiment showed that it took only 7.2 sec to implement the desired circuit of the laying path using the time-optimal trajectory, while the commercial Composicad software produced the trajectory with the execution time more than 14.0 sec.

Hence, the industrial experiments confirmed advantages of the developed optimal motion planning method and achieving the main objective of this work, which is targeted at the productivity improvement of the robotic systems for composite lay-up processes. Nevertheless, during these experiments, it was also detected some imperfect behavior of the robot and positioner that gives some perspectives for the future work.

The main results of Chapter 4 have been presented at the conference ICOME’2017.

CONCLUSIONS AND PERSPECTIVES

CONTRIBUTION OF THE THESIS

This thesis is devoted to the optimal motion planning in redundant robotic systems allowing improving the productivity of the automated composite lay-up workcell. Theoretical contributions of the thesis are in the area of redundancy resolution and time-optimal smooth motion generation for robotic systems composed of the robotic manipulator, workpiece positioner and workspace extension unit. The most essential results can be summarized as follows:

(i) *Formalization of the optimal motion planning problem for typical robotic lay-up platforms*. The proposed approach presents the considered problem as finding of the time-optimal trajectory in the joint space of the robotic system under specific constraints related to the technological task. The latter include both conventional kinematic constraints (joint limits, maximum joint velocities/accelerations), allowable distances to the singularities and obstacles as well as the Cartesian path constraints issued from the composite lay-up process.

(ii) *A new method of coordinated time-optimal motion planning for redundant robotic systems ensuring smooth movements of all mechanical components*. It is based on transformation of the original optimization problem into a combinatorial one and application of the dynamic programming principle. At the first stage, the redundant variables are discretized and the task graph is created by sequentially applying to all task locations the direct kinematics of the positioner (and workspace extension unit) as well as inverse kinematics of the robot. Then, the developed optimization algorithm based on dynamic programming generates the time-optimal motion taking into account all constraints related to the system kinematics and considered technology. Efficiency of the developed method and its advantages compared to the conventional techniques are confirmed by several case studies.

(iii) *Enhancement strategies for the developed motion planning method.* To reduce the computing time required for the motion planning, it was proposed to avoid combinatorial search in high-dimensional space. To achieve this target, the time-optimal motion planning is divided in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with relatively large discretization step. Then, the same technique is applied in the local search space, which is created with small discretization step in the neighborhood of the trajectory obtained at the previous stage. As an alternative, the second stage may implement a straightforward smoothing of the redundant variable profiles based on the spline approximation. As follows from relevant study, the proposed enhancement strategies allow essentially reducing the computing time, down to the level acceptable in engineering practice.

(iv) *Application of the developed optimal motion planning method to real industrial problems.* Using the thesis results, there were generated time-optimal smooth motions for the manipulator and positioner of the SPIDE-TP robotic platform allowing speeding up the thermoplastic tape lay-up process for manufacturing of the composite high-pressure vessel.

For these motions, it was created a program in KRL language ensuring coordinated control of KUKA KR210 robot and AFPT workpiece positioner. Experimental evaluation of the motions described by this program confirmed smoothness of the manipulator/positioner movements and essential reduction of the time required for the thermoplastic tape lay-up process.

The obtained results contribute to the area of redundancy resolution in robotic systems and give the user an efficient method for generation of time-optimal smooth motions whose implementation allow improving productivity of the automated composite lay-up workcell.

LIMITATIONS OF THE OBTAINED RESULTS

In spite of the essential advantages, there are still some limitations that are related both to the developed method of the time-optimal smooth motions planning and their implementation on the factory floor. The most significant of them are presented below.

Limitations of the developed method of optimal motion planning:

(i) **Computing time** for optimal motion planning **may be too high** if the redundancy degree is greater than one. The most important bottleneck is the fine collision test in the CATIA environment that is integrated in the preparation of the task graph. For example, for the industrial case study considered in Chapter 4, it took almost ten hours to test collisions for about 25000 robotic system configurations even setting triangulation accuracy down to 4.0 mm (0.2 mm by default). So, adding even a single additional redundant variable with 100 discrete values leads to increasing of the collision test time up to 1000 hours, which can be hardly accepted in practice. For this reason, the speeding up of the collision test deserves some special efforts.

(ii) In practice, **the principle constraints** on the velocities and accelerations in the actuated joints of the robot, positioner and workspace extensioner **are not known exactly**. It was discovered during the industrial experiments that the maximum velocities and accelerations declared in the manual are higher compared to the real values fixed in the KR C4 controller software. The latter leads to over-estimation of the actuator capacities and under-estimation of the motion time obtained using the developed method. For this reason, it is necessary to conduct dedicated experiments with the industrial robots allowing **evaluating correct values** of the maximum velocities/accelerations used in the motion planning algorithms implemented in the KRL programming language.

Limitations of the implementation on the factory floor:

(iii) **The lay-up task description** generated by the commercial software **may be imperfect**. For example, for the high-pressure vessel, a single circuit of the laying path generated by Composicad is not perfectly smooth with respect to the tool orientation angles. It includes discontinuities located at the connections of the cylinder and the domes (see Figure 60a). Although the given initial laying path and the generated time-optimal trajectory were slightly corrected (smoothed using the polynomial spline), the obtained motion is still

difficult to implement using the KR C4 controller capabilities. As follows from the industrial experiments, the robotic system movements are a little bit edgy in the neighborhood of the connection points. For this reason, the quality of the lay-up task description deserves particular attention.

(iv) The **geometric model** of the robotic system from the industrial partner **may be imperfect** and contain inaccurate parameters. In particular, the TOOL parameters (describing transformation between the robot flange and the tool center point) were not well calibrated. In our experiments, the latter caused some interference between the technological tool and the workpiece shaft, while no collisions were detected in 3D simulation in the CATIA environment. Besides, the compaction force applied to the thermoplastic by the technological tool was slightly over the desired level at some segments of the laying path. Hence, **careful calibration** of the geometric model is a very important issue in implementation of the time-optimal trajectories generated using the developed method.

Nevertheless, for the considered application area, the above mentioned limitations are not crucial and the developed method of the optimal motion planning provides essential improvement of the robotic system productivity. On the other hand, these limitations show some research directions for future work.

PERSPECTIVES AND FUTURE WORK

To generalize the obtained results, it is reasonable to continue research in the several directions and to concentrate on the following issues:

(i) *Enhancing the developed method* by combining the task graph generation and the optimal path searching on this graph. It will allow reducing the computing time by avoiding exhaustive collision check at the preliminary phase, which is applied to all possible configurations of the considered robotic system satisfying the task constraints. This modification of the proposed motion planning method requires integration of the collision check and verification of the acceleration constraints into dynamic programming based algorithm.

(ii) *Comparing the developed method* based on the combinatorial optimization with the continuous nonlinear time-optimal control in the space of the redundant variables and their derivatives or in the space of variables describing the end-effector motion. This approach looks promising but includes numerous difficulties related to the constraints transformation from the robotic system joint space to the considered state space.

(iii) *Speeding up the developed method* by preliminary segmentation of the given task into several sub-paths corresponding to a single circuit of the lay-up process. This may allow reducing the computing time by sequentially solving a number of low-dimensional combinatorial optimization problems and integrating further the obtained solutions into an aggregated trajectory ensuring implementation of the numerous lay-up circuits. It is worth mentioning that some preliminary work has been already done during industrial

experiments when the developed method has been slightly modified by adding an additional constraint on the similarity of the robot initial and final configurations.

(iv) *Evaluation of the developed motion planning method and its enhanced versions* by applying to wider set of application examples, which include both various helical circuits and circumferential ones. Also, it is reasonable to test the method for the large dimensional objects that require activating the workspace extension unit and evaluate its efficiency for the products, which do not possess axial symmetry**.**

PUBLICATIONS

Journal paper:

Jiuchun Gao, Anatol Pashkevich, Stéphane Caro. Optimization of the robot and positioner motion in a redundant fiber placement workcell, *Mechanism and Machine Theory*, Volume 114, 2017, Pages 170-189, ISSN 0094-114X.

International conference proceedings:

Jiuchun Gao, Anatol Pashkevich, Stéphane Caro. Manipulator Motion Planning in Redundant Robotic System for Fiber Placement Process. The 6th European Conference on Mechanism Science, Nantes, France, 2016 (EUCOMES'2016). In: Wenger P., Flores P. (eds) *New Trends in Mechanism and Machine Science*. Mechanisms and Machine Science, Volume 43, 2017, Pages 243-252. Springer, Cham.

Jiuchun Gao, Anatol Pashkevich, and Stéphane Caro. Optimal Trajectories Generation in Robotic Fiber Placement Systems. The 4th International Conference on Manufacturing and Industrial Technologies, Lisbon, Portugal, 2017 (ICMIT'2017). IOP Conference Series: *Materials Science and Engineering*. Volume. 212. No. 1. IOP Publishing, 2017.

Divya Shah, Jiuchun Gao, Anatol Pashkevich, Stéphane Caro and Benoît Courtemanche. Computer-Aided Design and Optimization of a Redundant Robotic System for Automated Fiber Placement Process. The 3rd International Conference on Mechanical Engineering, Surabaya, Indonesia, 2017 (ICOME'2017).

Jiuchun Gao, Anatol Pashkevich, Marco Cicellini and Stéphane Caro. Optimal Coordination of Robot Motions with Positioner and Linear Track in a Fiber Placement Workcell. The 15th International Conference on Informatics in Control, Automation and Robotics, Porto, Portugal, 2018 (ICINCO'2018). (*accepted*)

Jiuchun Gao, Anatol Pashkevich, Marco Cicellini and Stéphane Caro. Optimization of Robot and Positioner Motions in Manufacturing of High-pressure Composite Vessels. The 12th IFAC Symposium on Robot Control, Budapest, Hungary, 2018 (SYROCO'2018). (*accepted*)

REFERENCES

ABDALLA, F., MUTASHER, S., KHALID, Y., SAPUAN, S., HAMOUDA, A., SAHARI, B. & HAMDAN, M. 2007. Design and fabrication of low cost filament winding machine. *Materials & design,* 28**,** 234-239.

AHMAD, S. & LUO, S. 1989. Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in Cartesian space. *IEEE Transactions on Robotics and Automation,* 5**,** 409-417.

ALFORD, C. & BELYEU, S. Coordinated control of two robot arms. Robotics and Automation. Proceedings. 1984 IEEE International Conference on, 1984. IEEE, 468-473.

ALTING, L. E. O. & ZHANG, H. 1989. Computer Aided Process Planning: the state-of-the-art survey. *International Journal of Production Research,* 27**,** 553-585.

ANDRES, J., GRACIA, L. & TORNERO, J. 2012. Implementation and testing of a CAM postprocessor for an industrial redundant workcell with evaluation of several fuzzified Redundancy Resolution Schemes. *Robotics and Computer-Integrated Manufacturing,* 28**,** 265-274.

ASHBY, M. F. & CEBON, D. 1993. Materials selection in mechanical design. *Le Journal de Physique IV,* 3**,** C7-1-C7-9.

ATA, A. & MYO, T. 2006. Collision-free trajectory planning for manipulators using generalized pattern search. *International journal of simulation modelling,* 5**,** 145-154.

ATA, A. A. & MYO, T. R. 2005. Optimal point-to-point trajectory tracking of redundant manipulators using generalized pattern search. *International Journal of Advanced Robotic Systems,* 2**,** 24.

BAGGE, M. 2014. *Process planning for precision manufacturing: An approach based on methodological studies.* KTH Royal Institute of Technology.

BARRAL, D. 2003. Optimization tool for robot placement.: U.S. Patent 6,526,373.

BEAKOU, A., CANO, M., LE CAM, J.-B. & VERNEY, V. 2011. Modelling slit tape buckling during automated prepreg manufacturing: A local approach. *Composite structures,* 93**,** 2628-2635.

BELLMAN, R. 1954. The theory of dynamic programming. RAND CORP SANTA MONICA CA.

BELLMAN, R. 1958. On a routing problem. *Quarterly of applied mathematics,* 16**,** 87-90.

BELLMAN, R. 2013. *Dynamic programming*, Courier Corporation.

BEN-ISRAEL, A. & GREVILLE, T. N. 2003. *Generalized inverses: theory and applications*, Springer Science & Business Media.

BERTSEKAS, D. P., BERTSEKAS, D. P., BERTSEKAS, D. P. & BERTSEKAS, D. P. 1995. *Dynamic programming and optimal control*, Athena Scientific Belmont, MA.

BLOM, A. W., LOPES, C. S., KROMWIJK, P. J., GURDAL, Z. & CAMANHO, P. P. 2009. A theoretical model to study the influence of tow-drop areas on the stiffness and strength of variable-stiffness laminates. *Journal of composite materials*.

BRATUKHIN, A. & BOGOLYUBOV, V. 2012. *Composite manufacturing technology*, Springer Science & Business Media.

BROCK, O. 2000. *Generating robot motion: The integration of planning and execution*, Stanford University.

BUSS, S. R. 2004. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation,* 17**,** 16.

CAMPBELL, F. C. 2010. *Structural composite materials*, ASM international.

CHANG, C., CHUNG, M. J. & BIEN, Z. 1990. Collision-free motion planning for two articulated robot arms using minimum distance functions. *Robotica,* 8**,** 137-144.

CHAWLA, K. K. 2012. *Composite materials: science and engineering*, Springer Science & Business Media.

CHIAVERINI, S., ORIOLO, G. & MACIEJEWSKI, A. A. 2016. Redundant Robots. *In:* SICILIANO, B. & KHATIB, O. (eds.) *Springer Handbook of Robotics.* Cham: Springer International Publishing.

CHRISTENSEN, R. M. 2012. *Mechanics of composite materials*, Courier Corporation.

CHUNG, D. D. 2003. *COMPOSITE MATERIALS: Functional Materials for Modern Technologies (1)*, Springer.

CORMEN, T. H. 2009. *Introduction to algorithms*, MIT press.

CRAIG, J. J. 2005. *Introduction to robotics: mechanics and control*, Pearson Prentice Hall Upper Saddle River.

DA GRAÇA MARCOS, M., MACHADO, J. T. & AZEVEDO-PERDICOULIS, T.-P. 2009. Trajectory planning of redundant manipulators using genetic algorithms. *Communications in nonlinear science and numerical simulation,* 14**,** 2858-2869.

DEBORAH, D. C. 2010. Composite materials: Science and applications. *Engineering Materials and Processes*.

DEBOUT, P., CHANAL, H. & DUC, E. 2011. Tool path smoothing of a redundant machine: Application to Automated Fiber Placement. *Computer-Aided Design,* 43**,** 122-132.

DENKENA, B., SCHMIDT, C. & WEBER, P. 2016. Automated Fiber Placement Head for Manufacturing of Innovative Aerospace Stiffening Structures. *Procedia Manufacturing,* 6**,** 96-104.

DEVLIEG, R., JEFFRIES, K. & VOGELI, P. 2007. High-speed fiber placement on large complex structures. SAE Technical Paper.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische mathematik,* 1**,** 269-271.

DIRK, H.-J. L., WARD, C. & POTTER, K. D. 2012. The engineering aspects of automated prepreg layup: History, present and future. *Composites Part B: Engineering,* 43**,** 997-1009.

DOLGUI, A. & PASHKEVICH, A. 2006. Enhanced Optimisation Techniques for Off-line Programming of Laser Cutting Robots. Research Report G2I-EMSE 2007-500-015, November 2006, Ecole des Mines de Saint-Etienne.

DOLGUI, A. & PASHKEVICH, A. 2009. Manipulator motion planning for high-speed robotic laser cutting. *International Journal of Production Research,* 47**,** 5691-5715.

EBERLY, D. 2015. Robust computation of distance between line segments.

ELDÉN, L. 1982. A weighted pseudoinverse, generalized singular values, and constrained least squares problems. *BIT Numerical Mathematics,* 22**,** 487-502.

ELKINGTON, M., BLOOM, D., WARD, C., CHATZIMICHALI, A. & POTTER, K. 2015. Hand layup: understanding the manual process. *Advanced Manufacturing: Polymer & Composites Science,* 1**,** 138-151.

ELMARAGHY, H. & NASSEHI, A. 2014. Computer-Aided Process Planning. *In:* LAPERRIÈRE, L. & REINHART, G. (eds.) *CIRP Encyclopedia of Production Engineering.* Berlin, Heidelberg: Springer Berlin Heidelberg.

ENNEN, P., EWERT, D., SCHILBERG, D. & JESCHKE, S. 2016. Efficient collision avoidance for industrial manipulators with overlapping workspaces. *Automation, Communication and Cybernetics in Science and Engineering 2015/2016.* Springer.

ERICSON, C. 2004. *Real-time collision detection,* CRC Press.

FAHIMI, F. 2008. *Autonomous robots: modeling, path planning, and control,* Springer Science & Business Media.

FERNANDEZ, K. & COOK, G. E. 1988. A generalized method for automatic downhand and wirefeed control of a welding robot and positioner.

FLACCO, F. & DE LUCA, A. 2015. Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties. *Robotics and Autonomous Systems,* 70**,** 191-201.

FLEISCHER, J. & SCHAEDEL, J. 2013. Joining automotive space frame structures by filament winding. *CIRP Journal of Manufacturing Science and Technology,* 6**,** 98-101.

FLYNN, R., NIELSON, J. & RUDBERG, T. Production Implementation of Multiple Machine, High Speed Fiber Placement for Large Structures. Proceedings of the SAE 2010 Aerospace Manufacturing and Automated Fastening Conference & Exhibition, Wichita, KS, 2010.

FLYNN, R., RUDBERG, T. & STAMEN, J. 2011. Automated Fiber Placement Machine Developments: Modular Heads, Tool Point Programming and Volumetric Compensation Bring New Flexibility in a Scalable AFP Cell. *SME Technical Paper*.

FORD JR, L. R. 1956. Network flow theory. RAND CORP SANTA MONICA CA.

FREDMAN, M. L. & TARJAN, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM),* 34**,** 596-615.

FRKETIC, J., DICKENS, T. & RAMAKRISHNAN, S. 2017. Automated manufacturing and processing of fiber-reinforced polymer (FRP) composites: An additive review of contemporary and modern techniques for advanced materials manufacturing. *Additive Manufacturing*.

GALLET-HAMLYN, C. 2011. Multiple-use robots for composite part manufacturing. *JEC composites***,** 28-30.

GAN, Y., DAI, X. & LI, D. 2013. Off-line programming techniques for multirobot cooperation system. *International Journal of Advanced Robotic Systems,* 10**,** 282.

GAN, Y., DAI, X. & LI, J. 2012. Cooperative path planning and constraints analysis for master-slave industrial robots. *International Journal of Advanced Robotic Systems,* 9**,** 88.

GAO, J., PASHKEVICH, A. & CARO, S. 2017a. Manipulator Motion Planning in Redundant Robotic System for Fiber Placement Process. *New Trends in Mechanism and Machine Science.* Springer.

GAO, J., PASHKEVICH, A. & CARO, S. Optimal Trajectories Generation in Robotic Fiber Placement Systems. Materials Science and Engineering Conference Series, 2017b. 012009.

GAO, J., PASHKEVICH, A. & CARO, S. 2017c. Optimization of the robot and positioner motion in a redundant fiber placement workcell. *Mechanism and Machine Theory,* 114**,** 170-189.

GAROUSHI, S. 2018. Fiber-Reinforced Composites. *In:* MILETIC, V. (ed.) *Dental Composite Materials for Direct Restorations.* Cham: Springer International Publishing.

GAY, D. 2014. *Composite materials: design and applications*, CRC press.

GELFAND, I. M. & SILVERMAN, R. A. 2000. *Calculus of variations*, Courier Corporation.

GOUBALT, P. & MAYES, S. 1996. Comparative analysis of metal and composite materials for the primary structures of a patrol craft. *Naval engineers journal,* 108**,** 387-397.

GRIMSHAW, M. N., GRANT, C. G. & DIAZ, J. M. L. Advanced technology tape laying for affordable manufacturing of large composite structures. International sampe symposium and exhibition, 2001. Citeseer, 2484-2494.

GROOVER, M. P. 2007. *Automation, production systems, and computer-integrated manufacturing*, Prentice Hall Press.

GUETA, L. B., CHENG, J., CHIBA, R., ARAI, T., UEYAMA, T. & OTA, J. Multiple-goal task realization utilizing redundant degrees of freedom of task and tool attachment optimization. Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011a. IEEE, 1714-1719.

GUETA, L. B., CHIBA, R., ARAI, T., UEYAMA, T. & OTA, J. Design of the end-effector tool attachment for robot arm with multiple reconfigurable goals. Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on, 2008a. IEEE, 876-881.

GUETA, L. B., CHIBA, R., ARAI, T., UEYAMA, T. & OTA, J. Compact design of work cell with robot arm and positioning table under a task completion time constraint. Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, 2009a. IEEE, 807-813.

GUETA, L. B., CHIBA, R., ARAI, T., UEYAMA, T. & OTA, J. Hybrid design for multiple-goal task realization of robot arm with rotating table. Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, 2009b. IEEE, 1279-1284.

GUETA, L. B., CHIBA, R., ARAI, T., UEYAMA, T. & OTA, J. 2011b. Practical point-to-point multiple-goal task realization in a robot arm with a rotating table. *Advanced Robotics,* 25**,** 717-738.

GUETA, L. B., CHIBA, R., ARAI, T., UEYAMA, T., RUBRICO, J. I. U. & OTA, J. 2017. Compact design of a redundant manipulator system and application to multiple-goal tasks with temporal constraint. *Journal of Advanced Mechanical Design, Systems, and Manufacturing,* 11**,** JAMDSM0012-JAMDSM0012.

GUETA, L. B., CHIBA, R., OTA, J., UEYAMA, T. & ARAI, T. Coordinated motion control of a robot arm and a positioning table with arrangement of multiple goals. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, 2008b. IEEE, 2252-2258.

HALE, J. 2006. Boeing 787 from the ground up. *Aero,* 4**,** 7.

HARTENBERG, R. S. & DENAVIT, J. 1955. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied mechanics,* 77**,** 215-221.

HASENJAEGER, B. 2013. Programming and simulating automated fiber placement (AFP) CNC machines. *SAMPE JOURNAL,* 49**,** 7-13.

HERNANDEZ-MORENO, H., DOUCHIN, B., COLLOMBET, F., CHOQUEUSE, D. & DAVIES, P. 2008. Influence of winding pattern on the mechanical behavior of filament wound composite cylinders under external pressure. *Composites Science and Technology,* 68**,** 1015-1024.

HOLMES, J. G., MESSINA, E. R., RESNICK, B. J. & TEACH, C. C. 1986. Method and apparatus for controlling manipulator and workpiece positioner. Google Patents.

HUO, L. & BARON, L. 2008. The joint-limits and singularity avoidance in robotic welding. *Industrial Robot: An International Journal,* 35**,** 456-464.

HUSTY, M. L., PFURNER, M. & SCHRÖCKER, H.-P. 2007. A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator. *Mechanism and machine theory,* 42**,** 66-81.

IZCO, L., ISTURIZ, J. & MOTILVA, M. 2006. High speed tow placement system for complex surfaces with cut/clamp/& restart capabilities at 85 m/min (3350 IPM). SAE Technical Paper.

JONES, R. M. 1998. *Mechanics of composite materials*, CRC press.

JOUANEH, M. & DORNFELD, D. 1988. A kinematic approach for coordinated motion of a robot and a positioning table. *Journal of Manufacturing Systems,* 7**,** 307-314.

JOUANEH, M., DORNFELD, D. A. & TOMIZUKA, M. 1990a. Trajectory planning for coordinated motion of a robot and a positioning table. II. Optimal trajectory specification. *IEEE Transactions on Robotics and Automation,* 6**,** 746-759.

JOUANEH, M. K., WANG, Z. & DORNFELD, D. A. 1990b. Trajectory planning for coordinated motion of a robot and a positioning table. I. Path specification. *IEEE Transactions on Robotics and Automation,* 6**,** 735-745.

KAW, A. K. 2005. *Mechanics of composite materials,* CRC press.

KAZEROUNIAN, K. & NEDUNGADI, A. 1988. Redundancy resolution of serial manipulators based on robot dynamics. *Mechanism and machine theory,* 23**,** 295-303.

KLEIN, C. A. & HUANG, C.-H. 1983. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics***,** 245-250.

KROMBHOLZ, C., PERNER, M., BOCK, M. & RÖSTERMUNDT, D. Improving the production quality of the advanced automated fiber placement process by means of online path correction. 28th congress of the international council of the aeronautical sciences, 2012. 3922-3931.

KUKA 2010. Kuka System Software 5.5-Operating and Programming Instructions for System Integrators.

LAVAL, C. 2006. CADWIND 2006–20 years of filament winding experience. *Reinforced Plastics,* 50**,** 34-37.

LEONDES, C. T. 2000. *Computer-Aided Design, Engineering, and Manufacturing: Systems Techniques and Applications, Volume V, The Design of Manufacturing Systems*, CRC Press.

LINDBÄCK, J. E., BJÖRNSSON, A. & JOHANSEN, K. New Automated Composite Manufacturing Process:: Is it possible to find a cost effective manufacturing method with the use of robotic

equipment? The 5th International Swedish Production Symposium 6th-8th of November 2012 Linköping, Sweden, 2012. 523-531.

LIU, C. & TOMIZUKA, M. Algorithmic safety measures for intelligent industrial co-robots. Robotics and Automation (ICRA), 2016 IEEE International Conference on, 2016. IEEE, 3095-3102.

LUBIN, G. 2013. *Handbook of composites*, Springer Science & Business Media.

LUETH, T. C. Automated planning of robot workcell layouts. Proceedings 1992 IEEE International Conference on Robotics and Automation, 12-14 May 1992 1992. 1103-1108 vol.2.

MÄKINEN, K.-E., HELLBRATT, S.-E. & OLSSON, K.-A. 1998. The development of sandwich structures for naval vessels during 25 years. *Mechanics of Sandwich Structures.* Springer.

MALLICK, P. K. 2007. *Fiber-reinforced composites: materials, manufacturing, and design*, CRC press.

MANOCHA, D. & CANNY, J. F. 1994. Efficient inverse kinematics for general 6R manipulators. *IEEE transactions on robotics and automation,* 10**,** 648-657.

MARSH, G. 2007. Airbus takes on Boeing with reinforced plastic A350 XWB. *Reinforced plastics,* 51**,** 26-29.

MARSH, G. 2011. Automating aerospace composites production with fibre placement. *REINFORCED plastics,* 55**,** 32-37.

MENASRI, R., NAKIB, A., DAACHI, B., OULHADJ, H. & SIARRY, P. 2015. A trajectory planning of redundant manipulators based on bilevel optimization. *Applied Mathematics and Computation,* 250**,** 934-947.

MÖLLER, T. 1997. A Fast Triangle-Triangle Intersection Test. *Journal of Graphics Tools,* 2**,** 25-30.

MOURITZ, A., GELLERT, E., BURCHILL, P. & CHALLIS, K. 2001. Review of advanced composite structures for naval ships and submarines. *Composite structures,* 53**,** 21-42.

NEARCHOU, A. & ASPRAGATHOS, N. 1996. Application of genetic algorithms to point-to-point motion of redundant manipulators. *Mechanism and machine theory,* 31**,** 261-270.

NEARCHOU, A. C. 1998. Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism and machine theory,* 33**,** 273-292.

NEARCHOU, A. C. & ASPRAGATHOS, N. A. 1997. A genetic path planning algorithm for redundant articulated robots. *Robotica,* 15**,** 213-224.

NENCHEV, D. N. 1989. Redundancy resolution through local optimization: A review. *Journal of Field Robotics,* 6**,** 769-798.

NICOLAIS, L., MEO, M. & MILELLA, E. 2011. *Composite materials: a vision for the future*, Springer Science & Business Media.

NIKU, S. B. 2001. *Introduction to robotics: analysis, systems, applications*, Prentice Hall New Jersey.

NOF, S. Y. 1999. *Handbook of industrial robotics*, John Wiley & Sons.

OLSEN, H. B. & CRAIG, J. J. Automated composite tape lay-up using robotic devices. Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on, 1993. IEEE, 291-297.

PADULA, F. & PERDEREAU, V. 2011. A new pseudoinverse for manipulator collision avoidance. *IFAC Proceedings Volumes,* 44**,** 14687-14692.

PAN, Z., POLDEN, J., LARKIN, N., VAN DUIN, S. & NORRISH, J. 2012. Recent progress on programming methods for industrial robots. *Robotics and Computer-Integrated Manufacturing,* 28**,** 87-94.

PANK, D. R. & JACKSON, J. J. 1993. Metal- matrix composite processing technologies for aircraft engine applications. *Journal of Materials Engineering and Performance,* 2**,** 341-346.

PARK, S.-J. & SEO, M.-K. 2015. Manufacture of Carbon Fiber Composites. *Carbon Fibers*. Dordrecht: Springer Netherlands.

PARKER, J. K., KHOOGAR, A. R. & GOLDBERG, D. E. Inverse kinematics of redundant robots using genetic algorithms. Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, 1989. IEEE, 271-276.

PASHKEVICH, A. 1997. Real-time inverse kinematics for robots with offset and reduced wrist. *Control Engineering Practice,* 5**,** 1443-1450.

PASHKEVICH, A. P., DOLGUI, A. B. & CHUMAKOV, O. A. 2004. Multiobjective optimization of robot motion for laser cutting applications. *International Journal of Computer Integrated Manufacturing,* 17**,** 171-183.

PASHKEVICH, A. P., DOLGUI, A. B. & SEMKIN, K. I. 2003. Kinematic aspects of a robot-positioner system in an arc welding application. *Control Engineering Practice,* 11**,** 633-647.

PATEL, R. V. & SHADPEY, F. 2005. *Control of redundant robot manipulators: theory and experiments*, Springer Science & Business Media.

PEIPER, D. L. 1968. The kinematics of manipulators under computer control. DTIC Document.

PETERS, S. T. 2013. *Handbook of composites*, Springer Science & Business Media.

PHAM, D.-C., SRIDHAR, N., QIAN, X., SOBEY, A. J., ACHINTHA, M. & SHENOI, A. 2016. A review on design, manufacture and mechanics of composite risers. *Ocean Engineering,* 112**,** 82-96.

PILATO, L. A. & MICHNO, M. J. 1994. *Advanced composite materials*, Springer Science & Business Media.

PIRES, E. & MACHADO, J. Trajectory optimization for redundant robots using genetic algorithms. Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation, 2000a. Morgan Kaufmann Publishers Inc., 967-967.

PIRES, E. S., DE MOURA OLIVEIRA, P. B. & MACHADO, J. T. 2007. Manipulator trajectory planning using a MOEA. *Applied Soft Computing,* 7**,** 659-667.

PIRES, E. S. & MACHADO, J. T. A GA perspective of the energy requirements for manipulators maneuvering in a workspace with obstacles. Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, 2000b. IEEE, 1110-1116.

PONTRYAGIN, L. S. 1987. *Mathematical theory of optimal processes*, CRC Press.

RABENECK, D. 2010. Boosting composite production. *aerotec. The engineering and technology magazine for the aerospace industry***,** 28-30.

ROUSSEAU, J., PERREUX, D. & VERDIERE, N. 1999. The influence of winding patterns on the damage behaviour of filament-wound pipes. *Composites Science and Technology,* 59**,** 1439-1449.

ROWER, J. 2010. Robot Driven Automatic Tapehead for Complex Composite Lay-Ups. *SAE Technical Paper Offer 10AMAF-0066 (never published).*

RUDBERG, T., PURVIS, A. J., FAUBION, G. & NANCARROW, J. 2011. One Piece AFP Spar Manufacture. *SAE International Journal of Aerospace,* 4**,** 965-972.

SALAMA, M. M., STJERN, G., STORHAUG, T., SPENCER, B. & ECHTERMEYER, A. The first offshore field installation for a composite riser joint. Offshore technology conference, 2002. Offshore Technology Conference.

SALISBURY, J. K. & CRAIG, J. J. 1982. Articulated Hands:Force Control and Kinematic Issues. *The International Journal of Robotics Research,* 1**,** 4-17.

SASANE, A. 2016. *Optimization in Function Spaces*, Courier Dover Publications.

SHAMA RAO, N., SIMHA, T., RAO, K. & RAVI KUMAR, G. 2014. Carbon composites are becoming competitive and cost effective. *White paper from www. infosys. com.*

SHIRINZADEH, B., ALICI, G., FOONG, C. W. & CASSIDY, G. 2004. Fabrication process of open surfaces by robotic fibre placement. *Robotics and Computer-Integrated Manufacturing,* 20**,** 17-28.

SICILIANO, B. 1990. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems,* 3**,** 201-212.

SKINNER, M. L. 2006. Trends, advances and innovations in filament winding. *Reinforced Plastics,* 50**,** 28-33.

SLOAN, J. 2008. ATL & AFP: Defining the megatrends in composite aerostructures. *HIGH PERFORMANCE COMPOSITES,* 16**,** 68.

TABARAH, E., BENHABIB, B. & FENTON, R. G. 1994. Optimal motion coordination of two robots—a polynomial parameterization approach to trajectory resolution. *Journal of robotic systems,* 11**,** 615-629.

TAMARELLE, P. & SPARKS, C. High-performance composite tubes for offshore applications. Offshore technology conference, 1987. Offshore Technology Conference.

TUCKER, M. & PERREIRA, N. D. 1987. Generalized inverses for robotic manipulators. *Mechanism and machine theory,* 22**,** 507-514.

WANG, J., LI, Y. & ZHAO, X. 2010. Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm. *International Journal of Advanced Robotic Systems,* 7**,** 1-9.

WHITNEY, D. E. 1969. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems,* 10**,** 47-53.

WU, L., CUI, K. & CHEN, S.-B. 2000. Redundancy coordination of multiple robotic devices for welding through genetic algorithm. *Robotica,* 18**,** 669-676.

YOSHIKAWA, T. 1985. Manipulability of Robotic Mechanisms. *The International Journal of Robotics Research,* 4**,** 3-9.

ZA'ER, S., MIRZA, N. M., MIRZA, S. M. & ARIF, M. 2002. Cartesian path generation of robot manipulators using continuous genetic algorithms. *Robotics and autonomous systems,* 41**,** 179-223.

ZHANG, J. & FANG, X. 2017. Challenges and key technologies in robotic cell layout design and optimization. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science,* 231**,** 2912-2924.

Titre : Planification des mouvements optimaux dans les systèmes robotiques redondants pour un processus d'enroulement filamentaire composite automatisée

Mots clés : Système robotique redondant, Planification de mouvements, Trajectoire temps-optimale, Programmation dynamique, Enroulement filamentaire

Résumé : La thèse traite de la planification des mouvements optimaux dans les systèmes robotiques redondants pour l'automatisation des processus d'enroulement filamentaire. L'objectif principal est d'améliorer la productivité des cellules de travail en développant une nouvelle méthodologie d'optimisation des mouvements coordonnés du robot manipulateur, du positionneur de pièce et de l'unité d'extension de l'espace de travail. Contrairement aux travaux précédents, la méthodologie proposée offre une grande efficacité de calcul et tient compte à la fois des contraintes technologiques et des contraintes du système robotique, qui décrivent les capacités des actionneurs et s'expriment par les vitesses et accélérations maximales admissibles dans les articulations actionnées. La technique développée est basée sur la conversion du problème continu original en un problème combinatoire, où toutes les configurations possibles des composants mécaniquessont représentées sous la forme d'un graphe multicouche dirigé et le mouvement temporel optimal est généré en utilisant le principe de programmation dynamique. Ce mouvement optimal correspond au plus court chemin sur le graphique satisfaisant les contraintes de lissage. Les avantages de la méthodologie développée sont confirmés par une application industrielle d'enroulement filamentaire pour la fabrication de pièces thermoplastiques au CETIM.

Title : Optimal motion planning in redundant robotic systems for automated composite lay-up process

Keywords : Redundant robotic system, Motion planning, Time-optimal trajectory, Dynamic programming, Automated composite lay-up

Abstract : The thesis deals with the optimal motion planning in redundant robotic systems for automation of the composite lay-up processes. The primary goal is to improve the lay-up workcell productivity by developing a novel methodology of optimizing coordinated motions of the robotic manipulator, workpiece positioner and workspace extension unit, which ensure the shortest processing time and smooth movements of all mechanical components. In contrast to the previous works, the proposed methodology provides high computational efficiency and also takes into account both the technological constraints and the robotic system constraints, which describe capacities of the actuators and are expressed by the maximum allowable velocities and accelerations in the actuated joints. The developed technique is based on conversion of the original continuous problem into a combinatorial one, where all possible configurations of the mechanical components are represented as a directed multi-layer graph and the desired time-optimal motion is generated using dynamic programming principle for searching the shortest path on the graph satisfying the smoothness constraints. It is also proposed an enhancement of this technique by dividing the optimization procedure in two stages combining global and local searches. At the first stage, the developed algorithm is applied in the global search space generated with large discretization step. Then, the same technique is applied in the local search space, which is created with smaller step in the neighborhood of the obtained trajectory. The advantages of the developed methodology are confirmed by industrial implementation on the factory floor that deals with manufacturing of the high-pressure vessel.