TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABREVIATIONS

AFP	Automated Fiber Placement
BD-PSO	Blind Dynamic Particle Swarm Optimization
cnfg	Configuration
DHM	Denavit-Hartenberg Modified
DG	Dependency Group
DOF	Degree Of Freedom
DOP	Degree Of Placement
DOR	Degree Of Redundancy
EE	End-Effector
FKP	Forward Kinematic Problem
GA	Genetic Algorithm
HDPSO	Hybrid Discrete Particle Swarm Optimization
IKP	Inverse Kinematic Problem
NFC	Number of Function Call
NR	Newton-Raphson
PM	Parallel Mechanism
PSO	Particle Swarm Optimization
RGB	Red Green Blue
RP	Redundancy Provider

SM	Serial Manipulator
T&T	Tilt and Torsion
TSP	Traveling Salesman Problem
WJTT	Weighted Joint Travel Time
WRT	With Respect To

INTRODUCTION

In today's industrial robotic cells, efficiency is a key factor to mass production. Faster process cycle time leads to more production and more business value. Industrial robots are mainly used in repetitive applications, therefore, they successfully perform tasks programmed in teach mode and through off-line programming. Some examples of typical robot applications are pick and place operations, welding, painting, gluing, assembly, inspection, machining, drilling or composite fiber placement. One aspect that can be improved in order to minimize cycle time and thus increase productivity, is task placement. The goal of task placement optimization is to find the best layout of the workcell that minimizes the production time while avoiding singularities and obstacles.

In the simplest case, a layout consists of one robot and one path (task). In such a layout placement of the path with respect to the robot is to be optimized. Consider, for example, a laser cutting robot that needs to cut a metal plate that is held by a jig. Although we talk about path placement, in practical terms, we need to find the optimal placement of the jig. Such robotic cells can be more complex; for example multi robot collaboration with the presence of external axes, presence of obstacles in the environment, discretization of the operations, etc. These challenges add to the level of complexity and therefore need advanced optimization techniques.

This thesis is organized as following. First, a background review of the problems at hand is represented. Most recent and important literature are reviewed and the core value of solving the problems are described. Objectives and contribution of the thesis is expressed. There are three chapters dedicated to three main projects introduced in upcoming paragraphs. These projects are related to each other through redundancy resolution and path placement commonalities.

In Chapter 2, a generic method is proposed to cover the first three layouts. The method is structured around an industrial automated fiber placement project, in a collaboration between

École de technologie supérieure, Concordia University and École Polytechnique de Montréal. The goal is to simultaneously do path placement and trajectory planning. The contents of that chapter was published in (FarzanehKaloorazi *et al.* (2018b)).

In Chapter 3, a generic method based on a novel concept of swept volume is proposed to identify the number of independent parameters in the task placement and applied on ski goggles gluing task. Traditionally speaking, the identification has to be represented before the optimization, but here it is vice versa to emphasis the importance of identification. The contents of that chapter was published in (FarzanehKaloorazi *et al.* (2018a)).

In Chapter 4, the problem of the fourth layout, namely a task placement for discrete operation is addressed. A novel combined method is proposed to simultaneously optimize the sequences of images and camera placement for a turbine blade inspection workcell. Each chapter has a dedicated section for relevant introduction and literature review, as well as methodology explanation, results and conclusion sections.

CHAPTER 1

BACKGROUND AND PROBLEMATIC

1.1 Problem description

Optimal motion planning has been a relevant area for robotics researchers for many years. Several authors have worked on motion planning based on different optimization objectives (Ur-Rehman *et al.* (2010)). A review of trajectory planning techniques is given in (Ata (2007)). Various optimization objectives can be considered, such as energy consumption (Field & Stepanenko (1996); Hirakawa & Kawamura (1997)), travel or machining cycle time (Chan & Zalzala (1993); Pateloup *et al.* (2004)), minimum traveled distance (Tian & Collins (2003)), while satisfying several geometric, kinematic and dynamic constraints.

The trajectory planning deals with the determination of the path and velocity/acceleration profiles (or the time history of the robot's joints), the start and end points of the trajectory being predefined and fixed in the workspace. Another less explored aspect of trajectory planning is the placement of a given path within the workspace. It aims at determining the optimum location of a predefined path to be followed by the EE of a PMs or a robot, within its workspace with respect to one or many given objective(s) and constraint(s). This path can be the shape of a component to be machined, a welded profile or an artistic/decorative profile etc. In such situations, the trajectory planner cannot alter the shape of the path but he/she can only play with the location of that path within the workspace in order to optimize one or several criterion(a). Such an approach can be very interesting in many robotic applications. For example, in machining applications, the location of the workpiece within the workspace may affect the electric energy used by its actuators.

The path placement problem has not been extensively studied in the past. Nevertheless, some researchers proposed to solve it with respect to various optimization objectives. Several performance criteria for path location problems can be considered simultaneously (multi-objective) or individually, such as travel time, different kinetostatic performance indices (such as, ma-

nipulability or the conditioning number of the normalized kinematic Jacobian matrix), kinematic performance (velocity, acceleration), collisions, wear and vibration reduction, energy consumption etc.

Empirical performance criteria based on geometric and physical properties are commonly used for real-time decision-making (and design) of redundant anthropomorphic workcells performing high level tasks while avoiding obstacles, providing safety, and responding to human commands. Tisius *et al.* (2009) described 50 operational criteria and 50 potential criteria for serial manipulator operation. Machining or fiber placement quality, as well as robot energy consumption may also depend on the workpiece placement.

Optimal location of a task and optimal trajectory planning are an important research fields of robotics. In most cases, the presented works in the relevant literature introduce an index that allows the quantification of some aspect(s) of the manipulator's performance during task execution, a method, the mathematical formulation of the optimal task placement problem according to the considered constraints and the introduced index and a search method (Valsamos *et al.* (2018)). In most of these works, the results for the considered test cases that are examined to validate each of the proposed methods are mainly (with a very few exceptions such as (Ur-Rehman *et al.* (2010), Erdős *et al.* (2016)) in this case) numerical and simulation based.

1.2 Literature review

Researches related to our work can be categorized in three major sections, namely, workpiece (path) placement, redundancy resolution and Particle Swarm Optimization (PSO).

1.2.1 Workpiece (path) placement

Workpiece placement is always subject to redundancy, even in the presence of just one manipulator. The Degree Of Redundancy (DOR) in the workpiece placement is independent of the robot DOF and can be higher. (Caro *et al.* (2013)) determines the optimum placement of the workpiece to be machined, considering the elastostatic model of the robot and the cutting forces

applied on the tool. In (Robin *et al.* (2011)), a redundancy resolution for polishing operations is suggested. The kinematics capability is chosen to be the speed ratio of the End-Effector (EE), since they wanted to obtain voluntarily non-isotropic behavior. In (Ur-Rehman *et al.* (2010)), multi-objective path placement optimization for Parallel Mechanism (PM)s uses GA in order to minimize actuator torques, energy consumption, and shaking forces, for a 3 DOF PM. It is argued in (Vosniakos & Matsas (2010)) that it is more efficient to perform the milling operation in regions of the robot's workspace where manipulability is highest. Therefore the best initial pose of the robot is obtained to maximize manipulability. In (Hemmerle & Prinz (1991)), the authors numerically solved the problem of path placement for redundant manipulators. This included the problem of where to place the components (tables, other robots, or machining stations) relative to each other, as well as how to resolve the redundancies of the workcell. None of the aforementioned studies considers the path placement for a coordinated redundant robotic workcell that has a second manipulator detached from the main manipulator. They all place the path on a fixed platform rather than a moving platform. Furthermore, they neglect to perform the trajectory optimization simultaneously for both the main manipulator and the redundant manipulator.

In a more recent article, (Gao *et al.* (2017c,b,a)), the authors propose an approach to optimize the path planning of a robot and positioner in a redundant workcell for a fiber placement task. Time-optimal profiles for the joint variables are obtained by discretization of the problem, wherein all possible motions of the robot and positioner are represented as directed multi-layer graphs. This technique is based on the discrete dynamic programming principle, which allows finding the global optimum by sequentially solving all possible sets of the problems of lower dimensions. However, since the workspace of the positioner is discretized, for each motion of the positioner, the time-optimal solution is obtained by reducing to analysis of the preceding node at each step. Furthermore, the method does not take the placement of the positioner and the path on the positioner into account. The goal of (Gao *et al.* (2017c,b,a)) is one part of the objective of this chapter. This part will be referred to as the 1D optimization, i.e. optimization of the rotary table's motion.

The same objective of fiber placement redundancy resolution is investigated in (Hely *et al.* (2017); Hely (2016)). They obtain the optimal trajectory for rotary table by the means of gradient projection method. This method is a powerful tool when it comes to 1 DOF redundancy resolution, but for higher DOF workcells, it can only result into local optimums.

The optimal positioning of tasks in robot applications is an extremely important step in the design of robotic cells as it will allow the system to achieve the required high performance given the selected performance measure. In (Valsamos *et al.* (2018)) the optimal positioning of a robotic task is presented with the aim to minimize the required joint velocities during task execution, for a 6 DOF manipulator. The method is used to determine the optimal location for a path following task in the workspace of a UR5 manipulator. Results show that the optimal task placement allows for a significant reduction of joint velocities to maintain a given constant EE velocity during task execution.

Mlynek *et al.* (2018) discuss the quality of the manufacturing process technology of a shaped composite in 3D space. The mathematical model of the winding process and the matrix calculus are used to determine the optimized 3D trajectory of EE. The differential evolution algorithm is applied to finding the optimized 3D trajectory of the EE.

1.2.2 Redundancy resolution

In this work, we use the term “robot coordination” to mean a collaboration between two or more robots that leads to a high DOR in the workcell. In (Shimizu *et al.* (2008)), an analytic methodology for computing the inverse kinematic problem for 7 DOF redundant manipulators is proposed. The Jacobian matrix of redundant manipulators has a non-empty null space. Manipulability is enhanced by taking advantage of the null space in (Yoshikawa (1985)). Efficient use of the null space enables other types of subtask resolution to be done, such as torque optimization (Nokleby *et al.* (2005)), obstacle avoidance (Nearchou (1998); FarzanehKaloorazi *et al.* (2014)), and singularity avoidance (Nenchev *et al.* (2004); Kaloorazi *et al.* (2015)). These Jacobian-based redundancy resolutions are usable for tracking a trajectory of the ma-

nipulator tip, which may be generated dynamically. However, these methods are not suitable for the analysis of the global configuration space constrained by joint limits. To handle the global reachable region under joint limits, we have to solve the IKP in the position domain. In (Hollerbach & Suh (1987)), methods for resolving kinematic redundancies of manipulators by the effect on joint torque are examined. When the generalized inverse is formulated in terms of accelerations and incorporated into the dynamics, the effect of redundancy resolution on joint torque can be directly reflected. One method chooses the joint acceleration null-space vector to minimize joint torque in a least squares sense; when the least squares is weighted by allowable torque range, the joint torques tend to be kept within their limits. They present contrasting methods employing only the pseudo-inverse with and without weighting by the inertia matrix.

In (Wang *et al.* (2010)), the authors extend the closed-loop inverse kinematic algorithm to the acceleration level to meet some applications that require the joint accelerations. They have solved redundancy at the velocities and acceleration levels via pseudo-inverse method. In (Wang *et al.* (2010)), the objective function of joint limits avoidance is combined into the redundancy resolution as an optimization approach of the null space motion.

Jacobian-based redundancy resolution is applicable to dynamically tracking the trajectory of the robot. In (Hollerbach & Suh (1987)), the authors minimize the joint torques by choosing the joint acceleration null-space vector. In (Wu *et al.* (2009)), the authors derive the inverse dynamics for an actuated redundant 3-DOF PM by optimizing the driving force using the least squares approach. Various artificial intelligent methods are proposed by (Zhang & Lei (2011)) to solve the IKP of an actuation-redundant 3-DOF PM, such as multilayer perceptron neural networks, radial basis functions, and support vector machines. In (Cheng (1995)), the authors present a novel approach to on-line collision-free path planning of a two-arm manipulator system. They implemented a system that generates a collision-free path for one manipulator while the other is moving.

1.2.3 Particle swarm optimization

The optimization methods proposed in this study are based on a stochastic technique called PSO as the optimization core for components placement. PSO simulates the social movement behavior of a group (typically animals), such as a flock of birds or school of fish (Kennedy (2011); Kenndy & Eberhart (1995)). Like other meta-heuristic methods, PSO does not need any higher-order information, such as the gradient of the problem being optimized. Therefore, it does not require the problem to be differentiable. However, meta-heuristic methods do not guarantee the solution to be global optimal. Moreover, they usually involve a greater number of function calls than classic methods.

We chose PSO over other optimization methods because for the problem at hand, it is impossible to find an explicit answer to the first derivative of the cost function. Therefore, it is only possible to use a numerical optimization method. Out of all the numerical methods – including classical methods such as Newton-Raphson (NR) and more advanced methods based on the second derivative of the cost function – we chose an evolutionary algorithm because it provides us with the reliability and convergence time we need (Najjari & Guilbault (2015); Savsani *et al.* (2010); Kucuk (2013)). There are various evolutionary algorithms, some of them are well-known and have been used for a variety of problems such as Genetic Algorithm (GA) (Khorshidi *et al.* (2011)). In comparison to other evolutionary algorithms, PSO is easy to implement and obtains the optimum solution efficiently. It has a simple concept, it is robustness to control parameters. It is insensitive to scaling of design variables and can be easily parallelized for concurrent processing. It is derivative free and has very few algorithm parameters and eventually is a very efficient global search algorithm.

Among all the articles comparing PSO with GA, the reader is referred to Hassan *et al.* (2005) for more details. In this paper to carry out the t-tests, eight sample test problems were solved using both PSO and the GA over multiple runs. The test problems includes three well-known benchmark test problems; these are: the Banana (Rosenbrock function, the Eggcrate Function, and Golinski's Speed Reducer. Two space systems design problems were also investigated to

test the algorithms on real-life engineering problems. The eight test problems represent a wide range of complexity, nonlinearity, and constraint levels. Two metrics were identified for the two t-tests. The effectiveness test for both PSO and the GA uses a quality of solution metric that measures the normalized difference between the solutions obtained by the heuristic approaches and known solutions of the test problems. The efficiency test uses the number of function evaluations needed by the heuristic approaches to reach convergence. The same convergence criterion is enforced on PSO and the GA. The results of the t-tests support the hypothesis that while both PSO and the GA obtain high quality solutions, with quality indices of 99% or more with a 99% confidence level for most test problems, the computational effort required by PSO to arrive to such high quality solutions is less than the effort required to arrive at the same high quality solutions by the GA. The results further show the computational efficiency superiority of PSO over the GA is statically proven with a 99% confidence level in 7 out of the 8 test problems investigated. Further analysis shows that the difference in computational effort between PSO and the GA is problem dependent. It appears that PSO outperforms the GA with a larger differential in computational efficiency when used to solve unconstrained nonlinear problems with continuous design variables and less efficiency differential when applied to constrained nonlinear problems with continuous or discrete design variables.

An important distinction is that GA is initially a discrete technique that is also suitable for combinatorial problems, and PSO is a continuous technique that is very poorly suited to combinatorial problems. To equate their search space explorations, the particle movement in PSO can be seen as a form of path re-linking among personal best positions. In this sense, both PSO and GA can be seen as generating new solutions in the neighbourhood of two parents (via crossover in GA and via attractions to two personal best positions in PSO). This multi-parent effect should be a key advantage over single-point techniques such as simulated annealing, tabu search, and 1+lambda-ES.

To discuss an advantage of PSO over GA, PSO is really two populations, namely best memories and current positions. This allows greater diversity and exploration over a single population (which with elitism would only be a population of best memories). Also, the momentum

effects on particle movement can allow faster convergence (e.g. when a particle is moving in the direction of a gradient) and more variety/diversity in search trajectories. The details will be discussed in the upcoming sections.

1.3 Objectives and contributions of the thesis

In this study the goal is to propose and verify an efficient and simple solution to optimize the workcell layout. The core idea of this thesis is *how to find the best task placement in highly redundant cells*. As mentioned in the literature review, the task placement problem has only been addressed for simple layouts with zero or low DOR, without requiring any motion planning. On the other hand, trajectory and motion planning are solely explored assuming the placement of all components are already known and fixed.

The task placement problem has two aspects. Either the task is a set of points (frames) in space, for which all the points are given with respect to a reference frame and are locked one after another. Or the task is a set of points without any predetermined order and can be explored by any sequence. Each aspect needs to be approached in a particular way. We have addressed both aspects and described their methodologies through examples. For instance, fiber placement problem illustrates the first aspect and turbine blade inspection problem exhibits the second one. However, both solutions are represented generically and are applicable to any robotic placement problem within the scope of this thesis. Moreover, problems are simplified and reformed in order to use evolutionary techniques.

Our main contribution is proposing algorithms to solve task placement problem and motion planning, simultaneously. In proposed algorithms, best configuration selection is taken into account as well as robots' joint limits, singularity, workspace and collision avoidance. Furthermore, we propose an innovative algorithm for combined combinatorial and continuous optimization. This method can find application in a vast portion of engineering problems. Moreover, the number of independent placement parameters is often neglected, which is a basic requirement for task placement. We propose a systematic method to obtain the minimum

number of effective placement parameters, therefore task placements are optimized much more efficiently.

1.4 Overview of the results

The goal of the present study is to improve the efficiency of robotic workcells in various application fields through optimizing the placement of the task. The task placement problem is investigated for four layouts: (a) a single serial robot, (b) one serial robot and one parallel robot with a rotary table, (c) one serial robot and a redundant external module with trajectory planning and (d) one robot with a discrete task placement. The first layout is studied as a special case of the second layout. To have a rough estimation of the best possible placement, first a creative geometric-based approach is used to find a good-enough placement for the task for maximizing the reachable workspace. Then numerical approaches to achieve optimal task placement for multi-objective functions are introduced.

CHAPTER 2

SIMULTANEOUS PATH PLACEMENT AND TRAJECTORY PLANNING OPTIMIZATION FOR A REDUNDANT COORDINATED ROBOTIC WORKCELL

2.1 Introduction

Due to composites' light weight, high stiffness, and high strength, they have found applications in many areas, including aerospace, automobiles, wind turbines, and civil infrastructures. The increasing use of composites gives rise to the need to improve their method of manufacturing. Composite structures are typically manufactured using labor-intensive methods such as hand lay-up. However, this technique has many disadvantages, such as poor repeatability, a significant amount of wasted materials, and long process times. The advent of automated tape placement and Automated Fiber Placement (AFP) machines has significantly improved the manufacturing of composites in terms of speed of material deposition, repeatability, good compaction, reduction of waste, and seamless transfer from design to manufacturing.

2.1.1 Automated fiber placement

AFP machines have been used to make the fuselage of Boeing 787s and components of many aircraft structures (McCarville (2009); Chen & Chen (2015)). Current AFP machines are generally built for manufacturing aircraft components that are shaped like shallow plates or shells. Normally, they have a 6 Degree Of Freedom (DOF) Serial Manipulator (SM) in order to place the fiber on the workpiece and a rotary table to spin the workpiece. For certain applications where the shapes are more complex, such as tubes with double curvature, some modifications must be made to extend the machines' capabilities. One possible way to increase manufacturing flexibility is to add another 6 DOF manipulator to the workcell. Due to the heavy weight of the piece and other advantages of parallel mechanisms (Merlet (2006a); Nzue *et al.* (2013); Gao *et al.* (2002); Chiu & Perng (2004); Dasgupta & Mruthyunjaya (2000)), we have chosen to use a hexapod Parallel Manipulator (PM) in this work. The hexapod is a classic design for position and motion control, and is often used in flight simulation (Husty (1996); Yu (1987)).

Mounting the rotary table (1 DOF) on the 6 DOF hexapod permits maximum freedom in operation. The 6 DOF SM thus enables a 13 DOF coordinated robotic workcell to do the fiber placement task.

The quality of the end product – the composite – is largely dependent on how the original fiber path is generated (Shirinzadeh *et al.* (2007); Blom *et al.* (2009)). Before we can build a framework for collaboration between a 6-DOF hexapod and a SM, we must first analyze the robots' kinematics. It is essential to solve the Inverse Kinematic Problem (IKP) and Forward Kinematic Problem (FKP) of both robots in order to control them. The main concern of this chapter is to obtain the optimized motions for the aforementioned 13 DOF workcell.

In (Hassan *et al.* (2018)) a two-stage approach is proposed to perform fiber placement. The first stage considers multiple objectives to optimally allocate each autonomous industrial robots with surface areas, while the second stage aims to generate coordinated paths for the autonomous industrial robots.

In (Gao *et al.* (2018a), Gao *et al.* (2018b)) a methodology for optimal coordination of motions in robotic systems with robot, positioner and linear track is proposed in order to reduce the total processing time. The developed technique transforms the original continuous problem into a discrete one where the desired time-optimal motions are presented as a shortest path on the task graph satisfying the problem-specific acceleration and velocity constraints imposed on the joint coordinates. The desired time optimal motions are generated using enhanced dynamic programming algorithm that considers both of these constraints. Two case studies are presented to demonstrate efficiency of the approach and evaluate benefits of simultaneous actuation of all robotic system axes.

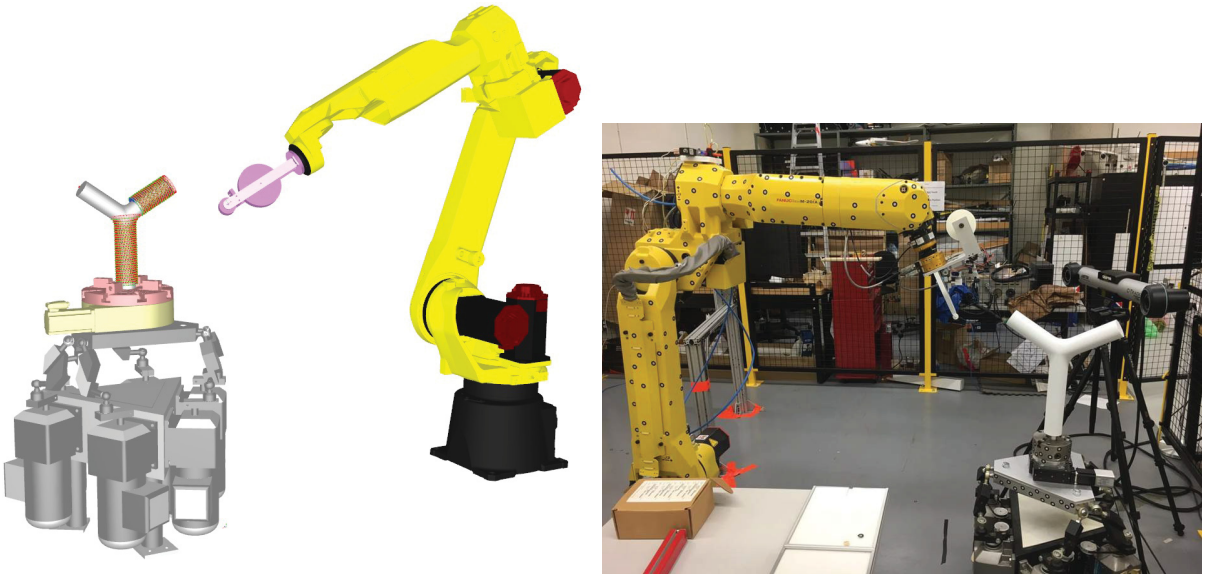
In this chapter, we intend to optimize the path placement of a redundant coordinated robotic workcell, Fig. 4.1. Assuming that the path is given using the method represented in (Hely *et al.* (2017); Hely (2016)), it is possible to place it anywhere within the workspace boundaries. In this chapter, a path is a set of time-independent frames in the Cartesian space given in the form of a text file (each line in the file has 16 entries representing 16 elements in the 4 by 4

homogeneous transformation matrix) and a trajectory is the motion in space (either Cartesian or joint space) relevant to each frame in the path. The optimization problem is to find the best path placement in order to satisfy the optimization criteria. As shown in Fig. 4.1, the coordinated workcell consists of an SM, a hexapod PM, and a rotary table mounted on the hexapod. The workpiece to be wrapped is placed on the rotary table, and the fiber placer head is a tool attached to the EE of the SM. In fiber placement applications, the tool must always be perpendicular to the surface of the workpiece. The given path is in a 6D space.

It is worth mentioning that since the hexapod is already integrated into the system, we will be using it as a tilt and torsion table (see section 2.2.2.1). Adding 1, 2 or 3 supplementary axes is indeed a classic approach, but our work generalizes this to a six-axis machine and could be adapted to these simpler cases. It is true that this operation can be done with fewer DOF, but it requires setting up a new workcell which is less justified practically. Since setting up a robotic workcell is a costly deed, one needs to design a multipurpose workcell. This workcell is set up in such a way that it is used for multiple purposes, one of such being the focus of this chapter, i.e., fiber placement. In this chapter, collision avoidance among the different parts of the workcell is not taken into account. However it is considered in the third chapter.

The proposed optimization method consists of a 1D optimization for the rotary table's rotation angle and a 6D optimization for hexapod's pose. The motion optimization of the rotary table is coupled with the hexapod EE pose. Therefore, they must be solved within one optimization routine. The path is fixed to the moving frame of the rotary table, the base frame of the rotary table is fixed to the EE of the hexapod, and the base frame of the hexapod is fixed with respect to the global base frame of the SM. The hexapod moves the rotary table and places it in the optimized pose regarding the singular poses of the SM. Once done, the rotary table moves the workpiece in order to place each point of the path in the best place.

The remainder of this chapter is organized as follows. First, the kinematic equations of the hexapod and SM are briefly addressed. The general idea of PSO is represented. Then, the methodology to optimize the path placement problem of the workcell is explained through a



a) b) Taken from (Zhang *et al.* (2014))

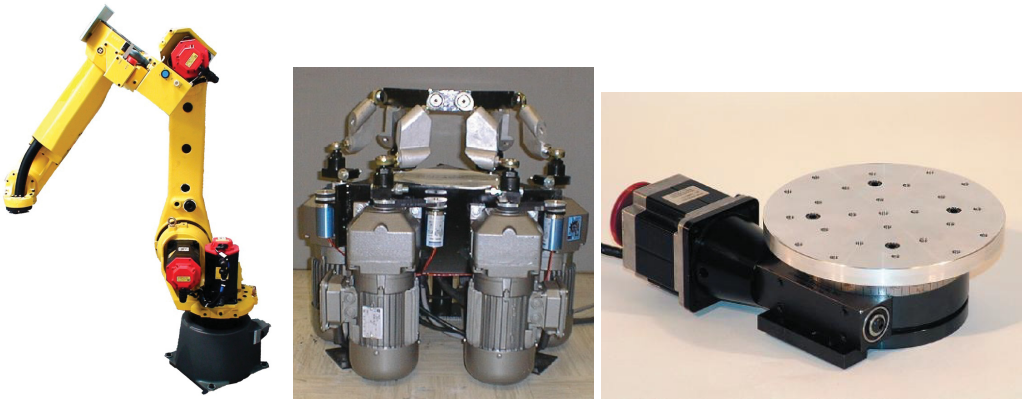
Figure 2.1 The redundant coordinated robotic workcell. The workpiece to be wrapped is placed on the rotary table, which is mounted on the hexapod. The SM has a fiber placement tool attached to its EE.

simple planar example. Finally, the proposed method is applied to a 6D case study and the obtained results are presented.

2.2 Kinematics of the hexapod and Serial Manipulator

In the 13 DOF workcell under study, three mechanisms are involved: (a) a 6 DOF SM; (b) a hexapod; and (c) a rotary table (as shown in Fig. 2.2). The SM is a Fanuc LR Mate 120i industrial robot. All its joints are revolute joint actuators and its last three joints intersect at the *wrist center*, as shown in Fig 2.2a. The hexapod is a 6-RSS PM, where R stands for actuated revolute joint and S stands for spherical passive joint (see Fig. 2.2b).

In this section, the FKP and IKP of the robots are explained. The FKP is to obtain the pose (position and orientation) of the EE (in Cartesian space), having the values of the actuators (in the joint space). The EE's pose is the displacement along the x, y and z axis and rotation about the x, y and z axis. The IKP is to find the values of all actuators corresponding to a given EE



a) 6 DOF SM b) 6 DOF hexapod c) Rotary table

Figure 2.2 Three mechanisms collaborating in the project.

pose. Both the IKP and FKP are represented in the base frame of each robot. In what follows, the IKP and FKP for the SM and hexapod are presented.

2.2.1 Kinematics of the SM

The kinematic model of the SM is derived based on the Denavit-Hartenberg Modified (DHM) notation (Denavit (2000)). Table 2.1 presents the DHM information for the Fanuc robot. The transformation matrix from frame i to $i-1$ can be shown as follows:

$$\mathbf{A}_i^{i-1} = \mathrm{Rot}(\mathrm{x},\alpha_i)\mathrm{Trans}(a_i,0,0)\mathrm{Rot}(\mathrm{z},\theta_i)\mathrm{Trans}(d_i,0,0), \tag{2.1}$$

$$\mathbf{A}_i^{i-1} = \begin{bmatrix} c(\theta_i) & -s(\theta_i) & 0 & a_i \\ s(\theta_i)c(\alpha_i) & c(\theta_i)c(\alpha_i) & -s(\alpha_i) & -d_i s(\alpha_i) \\ s(\theta_i)s(\alpha_i) & c(\theta_i)s(\theta_i) & c(\alpha_i) & d_i c(\alpha_i) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

The FKP of any SM can be obtained via chain multiplication of the transformation matrix of the robot's joints, as follows:

Table 2.1 DHM parameters of the SM in cm.

i	θ_i	α_i	a_i	d_i
1	θ_1	0	0	0
2	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	150	0
3	θ_3	0	790	0
4	θ_4	$-\frac{\pi}{2}$	250	835
5	θ_5	0	$\frac{\pi}{2}$	0
6	θ_6	0	$-\frac{\pi}{2}$	100

$$\mathbf{A}_6^0 = \mathbf{A}_1^0\mathbf{A}_1^2\mathbf{A}_2^3\mathbf{A}_4^3\mathbf{A}_5^4\mathbf{A}_6^5. \tag{2.3}$$

The FKP of SMs is easy to solve; the main challenge is to solve the IKP for SMs. The majority of 6 DOF industrial robots have a wrist arrangement for the last three joints that leads to an analytical solution of the IKP. Various methods have been introduced to solve the IKP of the wrist-partitioned SMs, such as geometrical methods (Husty *et al.* (2007)), and mathematical ones (Qiao *et al.* (2010)). This chapter uses the formulation suggested in Craig's book; for more details on the IKP, the reader is referred to (Craig).

Having in mind that the direct kinematics is as follows:

$$X = \mathbf{T}_6^0 = \mathbf{A}_1^0\mathbf{A}_2^1\mathbf{A}_3^2\mathbf{A}_4^3\mathbf{A}_5^4\mathbf{A}_6^5 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

we can find the position of the origin of the 4^{th} frame using one of the two next equations:

$$\mathbf{T}_4^0 = \mathbf{A}_1^0\mathbf{A}_2^1\mathbf{A}_3^2\mathbf{A}_4^3 = \mathbf{X}(\mathbf{A}_5^4\mathbf{A}_6^5)^{-1} = \begin{bmatrix} & \mathbf{R}_4^0 & & \mathbf{p}_{4,\text{origin}}^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.5}$$

$$\mathbf{p}^0_{4,origin} = \mathbf{A}^0_1\mathbf{A}^1_2\mathbf{A}^2_3\mathbf{A}^3_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{X} \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix}. \tag{2.6}$$

It is straight forward to find the first three joints based on:

$$\mathbf{p}^0_{4,\text{origin}} = \begin{bmatrix} p_{x4} \\ p_{y4} \\ p_{z4} \end{bmatrix} = \begin{bmatrix} c_1(a_2 + a_3 s_2 + d_4 c_{23} + a_4 s_{23}) \\ s_1(a_2 + a_3 s_2 + d_4 c_{23} + a_4 s_{23}) \\ d_1 + a_3 c_2 - d_4 s_{23} + a_4 c_{23} \end{bmatrix} = \begin{bmatrix} p_{x4} - d_6 a_x \\ p_{y4} - d_6 a_y \\ p_{z4} - d_6 a_z \end{bmatrix}. \tag{2.7}$$

Combining the two first equations from Eq. (2.7), solution for first joint q_1 is direct considering that the term $a_2 + a_3 s_2 + d_4 c_{32} + a_4 s_{32}$ is different from zero:

$$\frac{c_1}{p_x - d_6 a_x} = \frac{s_1}{p_y - d_6 a_y}. \tag{2.8}$$

There are two solutions for q_1:

$$q_{1\{1,2\}} = \text{atan2}(m_1(p_y - d_6 a_y), m_1(p_x - d_6 a_x)), \tag{2.9}$$

$m = [+1; -1]$ is an index, either +1 or -1, to indicate the working mode of the robot. In the case that $p_y = d_6 a_y$ and $p_x = d_6 a_x$ the so-called alignment singularity occurs. In this case we cannot apply Eq. (2.8). Having the solution for q_1, q_3 can be obtained using next two equations:

$$c_1 p_{x_4} + s_1 p_{y_4} - a_2 = a_3 s_2 + d_4 c_{23} + a_4 s_{23} \tag{2.10}$$

$$a_3 c_2 - d_4 s_{23} + a_4 c_{23} = p_{z_4} - d_1. \tag{2.11}$$

Simplifying the above equations one has:

$$c_3 = \frac{A + Bs_3}{C}, \tag{2.12}$$

in which:

$$A = (p_{z_4} - d_1)^2 + (c_1 p_{x_4} + s_1 p_{y_4} - a_2)^2 - a_3^2 - d_4^2 - a_4^2 \tag{2.13}$$

$$B = 2a_3 d_4 \tag{2.14}$$

$$C = 2a_4 a_3. \tag{2.15}$$

Two cases may occur; $C = 0$ or $C \neq 0$. If $C = 0$ it means that a_4 is zero (a_3 could also be zero but in this case it would not be useful to have the third axes).

$$s_3 = -\frac{A}{B}. \tag{2.16}$$

Therefore q_3:

$$q_{3\{1,2\}} = \text{atan2}(s_3, m_2\sqrt{1 - s_3^2}), \tag{2.17}$$

$m_2 = [+1; -1]$ is an index for working mode. In the case that $C \neq 0$:

$$s_{3\{1,2\}} = \frac{-\frac{AB}{C^2} \pm \sqrt{\frac{A^2B^2}{C^4} + (1 + \frac{B^2}{C^2})(1 - \frac{A^2}{C^2})}}{1 + \frac{B^2}{C^2}}. \tag{2.18}$$

In case that the absolute value for s_3 is bigger than 1, the given pose is not reachable by the robot and it is called elbow singularity. Having s_3, q_3 is:

$$q_{3\{1,2\}} = \text{atan2}(s_{3\{1,2\}}, \frac{A + Bs_{3\{1,2\}}}{C}). \tag{2.19}$$

In order to find q_2, from Eq. (2.11) one has:

$$a_3 - d_4 s_3 + a_4 c_3 = k_1 s_2 + k_2 c_2 \tag{2.20}$$

$$d_4 c_3 + a_4 s_3 = c_2 k_1 - s_2 k_2, \tag{2.21}$$

where:

$$k_1 = c_1 p_{x_4} + s_1 p_{y_4} - a_2 \tag{2.22}$$

$$k_2 = p_{x_4} - d_1, \tag{2.23}$$

As we have q_3, q_2 is defined based on the following:

$$\begin{bmatrix} k_1 & k_2 \\ -k_2 & k_1 \end{bmatrix} \begin{bmatrix} s_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_3 + d_4 s_3 + a_4 c_3 \\ d_4 c_3 + a_4 s_3 \end{bmatrix} \tag{2.24}$$

$$q_2 = \text{atan2}(s_2, c_2). \tag{2.25}$$

To find the last three joints we must use $\mathbf{R}_6^3$:

$$\mathbf{T}_6^3 = (\mathbf{A}_1^0 \mathbf{A}_2^1 \mathbf{A}_3^2)^{-1} \mathbf{X} = \mathbf{A}_4^3 \mathbf{A}_5^4 \mathbf{A}_6^5 = \begin{bmatrix} & \mathbf{R}_6^3 & & \mathbf{p}_{6,\text{origin}}^3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.26}$$

where

$$\mathbf{R}_6^3 = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} = \begin{bmatrix} * & * & -c_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \\ * & * & s_4 s_5 \end{bmatrix}, \tag{2.27}$$

we obtained the expressions marked with as they are not of interest. We obtained two solutions for q_5:

$$q_{5\{1,2\}} = \text{atan2}(m_3 \sqrt{1 - r_{2,3}^2}, r_{2,3}). \tag{2.28}$$

Once q_5 is found, q_4 and q_6 are completely defined if q_5:

$$q_4 = \text{atan2}(\frac{r_{3,3}}{s_5}, \frac{-r_{1,3}}{s_5}), \tag{2.29}$$

$$q_6 = \text{atan2}(\frac{r_{2,2}}{s_5}, \frac{-r_{2,2}}{s_5}). \tag{2.30}$$

If the robot works always in the range of $[-180, 180]$ degree, there are 8 possible solutions. In the case of $q_5 = 0$ which is also called as wrist singularity:

$$\mathbf{R}_6^3 = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} = \begin{bmatrix} -c_4c_6 + s_4s_6 & *0 & \\ 0 & 0 & 1 \\ c_4s_6 + s_4c_6 & * & 0 \end{bmatrix}. \tag{2.31}$$

One can completely find q_6 if q_4 is known (there is one DOF so it can be considered that $q_4 = 0$):

$$s_4r_{1,1} + c_4r_{3,1} = s_4^2s_6 + c_4^2s_6 = s_6 \tag{2.32}$$

$$s_4r_{3,1} - c_4r_{1,1} = c_6 \tag{2.33}$$

$$q_6 = \text{atan2}(s_6, c_6). \tag{2.34}$$

2.2.1.1 Jacobian matrix and singularities of the SM

Consider all kinematic equations of EE, $m = 6$. Each of them is a function of $n = 6$ DOF:

$$\begin{aligned} x &= x(\theta_1, \ldots, \theta_6) \\ y &= y(\theta_1, \ldots, \theta_6) \\ z &= z(\theta_1, \ldots, \theta_6) \\ \beta &= \beta(\theta_1, \ldots, \theta_6) \\ \alpha &= \alpha(\theta_1, \ldots, \theta_6) \\ \gamma &= \gamma(\theta_1, \ldots, \theta_6). \end{aligned} \tag{2.35}$$

The time derivative of the above equations is found as follows:

$$\begin{aligned} \frac{dx}{dt} &= \frac{\partial x}{\partial \theta_1}\frac{d\theta_1}{dt} + \cdots \frac{\partial x}{\partial \theta_6}\frac{d\theta_6}{dt} \\ &\vdots \\ \frac{d\gamma}{dt} &= \frac{\partial \gamma}{\partial \theta_1}\frac{d\theta_1}{dt} + \cdots \frac{\partial \gamma}{\partial \theta_6}\frac{d\theta_6}{dt} \end{aligned} \tag{2.36}$$

In Eq. (2.35), x, y, z are displacement and β, α, γ are orientations of the EE of the Fanuc robot.

To obtain the Jacobian matrix of the SMs using Eq. (2.3), one has:

$$\begin{aligned} x &= \mathbf{A}_6^0(1,4) \\ y &= \mathbf{A}_6^0(2,4) \\ z &= \mathbf{A}_6^0(3,4) \\ \beta &= \text{atan2}(\mathbf{A}_6^0(1,3), \sqrt{1 - \sin(\mathbf{A}_6^0(1,3))^2}) \\ \alpha &= \text{atan2}(-\mathbf{A}_6^0(2,3)/\cos(\beta), \mathbf{A}_6^0(3,3)/\cos(\beta)) \\ \gamma &= \text{atan2}(-\mathbf{A}_6^0(1,2)/\cos(\beta), \mathbf{A}_6^0(1,1)/\cos(\beta)). \end{aligned} \tag{2.37}$$

This can be written in vector form:

$$\dot{x} = \mathbf{J}\dot{\theta}, \tag{2.38}$$

where $\mathbf{J}$ is the Jacobian matrix – in which the elements are the partial derivatives of the kinematics equations, $\dot{x}$ is the vector of EE velocities, and $\dot{\theta}$ is the vector of joint velocities. The relationship between the EE velocity and the (known) joint velocities is thus fully described by the Jacobian. The EE velocity is a linear function of the joint velocities. Any configuration in the workspace of the robot for which $\det(\mathbf{J})$ vanishes is called a singularity. In singularity configurations, the robot may lose some DOF or reach some uncontrollable DOF. For more information on how to obtain the Jacobian matrix, please refer to (Nubiola (2011); Kohli & Hsu (1987)). For singularities in SMs, see ((Angeles & Angeles, 2002; Zlatanov *et al.*, 1998)).

2.2.2 Kinematics of the hexapod

In contrast to SMs, it is easy to solve the IKP for PMs and difficult (in some cases impossible) to find an analytical solution for the FKP. The IKP of the hexapod under study is obtained analytically. The FKP is solved using a numerical method.

The kinematic model of the hexapod is represented in Fig. 2.3. The rotation matrix representing the orientation of the mobile platform ($o'_{x'y'z'}$) relative to the base frame (o_{xyz}) is as follows:

$$\mathbf{R}_{EE} = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}. \tag{2.39}$$

The following relation holds for each link (see Fig. 2.3 and please refer to (Zhang *et al.* (2014)) for more details):

$$\overrightarrow{OO'} = \overrightarrow{OB_i} + \overrightarrow{B_iT_i} + \overrightarrow{T_iA_i} + \overrightarrow{A_iO'}, \tag{2.40}$$

where the angle of the vector $\overrightarrow{B_iT_i}$ is the joint angle. The initial position of an actuator is set by the user. In Eq. (2.40), $\overrightarrow{OB_i}$ is fixed, $\overrightarrow{A_iO'}$ is defined by the EE pose, and the length of the distal links T_iA_i and proximal link B_iT_i are known. Therefore, by solving a quadratic equation,

the IKP of the hexapod is solved. This quadratic equation corresponds to the intersection of the joint circle with the center of B_i, and the sphere with the center of A_i.

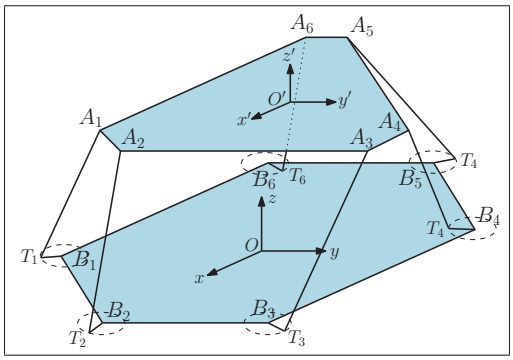


Figure 2.3 The kinematic model of the hexapod. Revolute actuators are located in $B_{1...6}$.

Equation (2.41) represents this system of equations (values are in mm).

$$\begin{cases} (x_{t_i} - x_{a_i})^2 + (y_{t_i} - y_{a_i})^2 + (z_{t_i} - z_{a_i})^2 = l_{AT}^2 \\ (x_{t_i} - x_{b_i})^2 + (y_{t_i} - y_{b_i})^2 = l_{AT}^2 \\ z_{t_i} = |t_i T_i| \\ x_{t_i}^2 + y_{t_i}^2 = |B_i T_i|^2 \end{cases} \tag{2.41}$$

The fixed values $|B_iT_i|$ and in Eq. (2.41) is known from the geometrical properties of hexapod.There can be up to $2^6 = 64$ solutions to the hexapod IKP. The hexapod FKP is solved numerically; see (Zhang *et al.* (2014)).

2.2.2.1 Workspace of the hexapod

The hexapod provides a 6 DOF workspace. Since the rotary table provides one orientational DOF and is mounted on the hexapod, one orientational DOF in the workspace of the hexapod is redundant. One can consider the orientation about the z-axis of hexapod's EE as the redundant DOF. Furthermore, the displacement of the EE is very limited, and compared to the workspace of the SM, it is very small: about 1%. Therefore, the translation along the x, y and z axes is negligible. Hence, the hexapod is essentially only useful for tilting about the horizontal axis.

One orientational workspace representation, which is very applicable here, is called *Tilt and Torsion* (T&T) workspace (Bonev & Gosselin (2006); Jiang & Gosselin (2009)). T&T has proven to be useful for the workspace representation of symmetrical parallel mechanisms. The next step is to use a simple numerical procedure to obtain the so-called constant-torsion workspace.

The T&T representation only uses two angles in such a way that the axis of one angle is variable and defined by the other angle. As shown in Fig. 2.4a, first the body frame is tilted about the horizontal axis a, at an angle θ. Then, as illustrated in Fig. 2.4b, the body frame is rotated about the updated z axis, called the z' axis, at an angle σ. The first step is the so-called *tilt*, and the second step is *torsion*.

T&T angles are defined in two steps, tilt and torsion. It does not mean that only two angles define T&T angles, but simply that the axis of tilt is variable and defined by another angle. In the first step, illustrated in Fig. 2.4a, the body frame is tilted about a horizontal axis a, at an angle θ, referred to as the *tilt*. Axis a is defined by an angle ϕ, called the *azimuth*, which is the angle between the projection of the body's z' axis onto the fixed xy plane and the fixed x axis. In the second step, Fig. 2.4b, the body frame is rotated about the body's z' axis at an angle σ, called the *torsion*.

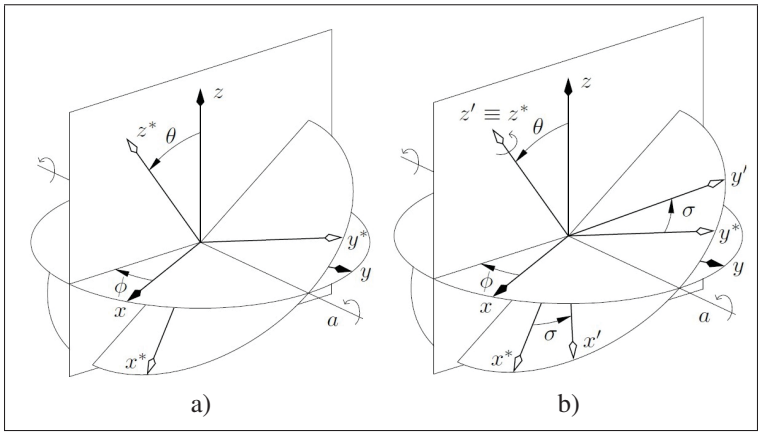


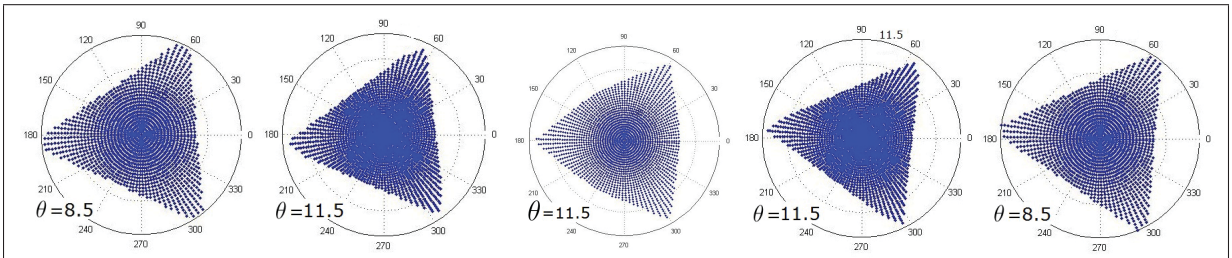
Figure 2.4 The T&T angles defined by successive rotations: (a) tilt; (b) torsion (taken from Bonev & Gosselin (2006)).

The rotation matrix **R** defines the orientation described by T&T angles (see (Bonev *et al.* (2002))), as follows

$$\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\sigma-\phi) \tag{2.42}$$

$$\mathbf{R} = \begin{bmatrix} c(\phi)c(\theta)c(\sigma-\theta)-s(\phi)s(\sigma-\phi) & -c(\phi)c(\theta)s(\sigma-\phi)-s(\phi)c(\sigma-\phi) & c(\phi)s(\theta) \\ s(\phi)c(\theta)c(\sigma-\phi)+c(\phi)s(\sigma-\phi) & -s(\phi)c(\theta)s(\sigma-\phi)+c(\phi)c(\sigma-\phi) & s(\phi)s(\theta) \\ -s(\theta)c(\sigma-\phi) & s(\theta)c(\sigma-\phi) & c(\theta) \end{bmatrix}. \tag{2.43}$$

The T&T workspace of the hexapod for a range of σ is represented in Fig. 2.5. In this figure, the T&T workspace is obtained by means of discretization, at a specific pose of the EE, i.e. $[x,y,z] = [0,0,0.12]$ cm, which is roughly the center of the workspace. In Fig. 2.6, the maximum projected orientation workspace of the mechanism is depicted for a range of other variables, i.e. $x = [-0.4,0.4], y = [-0.4,0.4]$, $z = [0.09,0.15], \sigma = [-0.2,0.2]$. As can be observed in Fig. 2.6, the maximum tilt provided by the hexapod is about 12 degrees.



a) $\sigma = -34°$ b) $\sigma = -17°$ c) $\sigma = 0°$ d) $\sigma = 17°$ e) $\sigma = 34°$

Figure 2.5 Projected orientation workspace $[x,y,z] = [0,0,0.12]$.

It is essential to obtain the workspace of the hexapod in order to limit the search space of the transition phase in the algorithm. While optimizing the placement of the main branch of the workpiece, the optimization is constrained by the joint limits, i.e. workspace, of the SM. Therefore, once the optimized place has been obtained for the main branch of the work-

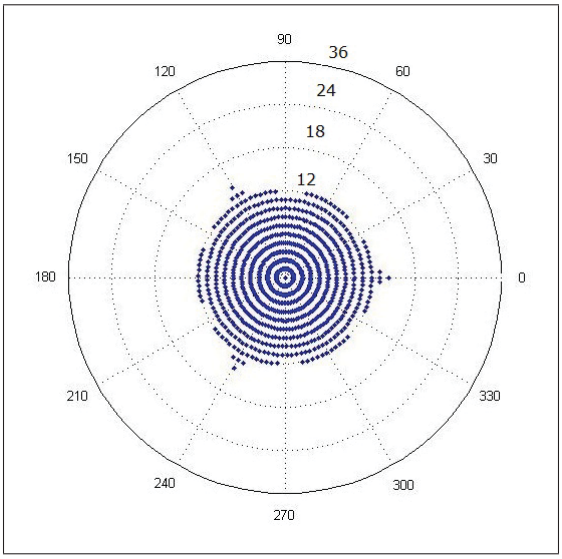


Figure 2.6 Maximal projected orientation workspace $x = [-0.4, 0.4]$, $y = [-0.4, 0.4]$, $z = [0.09, 0.15]$, $\sigma = [-11.5, 11.5]$.

piece, the hexapod has to be placed according to the obtained place. This can be anywhere in the workspace of the SM and is not related to the workspace of the hexapod. But once the placement of the hexapod is fixed for the main branch, the optimum placement for the second branch has to be within the workspace of the hexapod. This is because the transition phase is the movement of the EE of the hexapod in order to place the second branch in an optimum place. Therefore, the search space for the transition phase is restricted to the obtained T&T workspace. Essentially, due to the very limited translational workspace of the hexapod, it is only being used as a tilt and torsion table.

At a singular pose, the EE loses one or more degrees of twist freedom (instantaneously, the EE cannot move in these directions). Serial robots with less than six independent joints are always singular in the sense that they can never span a six-dimensional twist space. This is often called an architectural singularity. A singularity is usually not an isolated point in the workspace of the robot, but a sub-manifold. When robot's EE reaches a singular pose, it is impossible to control it. Therefore one has to avoid such poses. These poses are related to the Jacobian matrix of the robot. When Jacobian matrix vanishes, the relation between the actuated joints and EE space is destroyed. Most of the time we want a robot to follow a specific trajectory

in the workspace. Also, we want it to be as far from the singularity as possible. The word “Trajectory" refers to a set of points. The number of points needed to represent a trajectory depends on the accuracy needed for a specific task. The more accurate the trajectory, the more calculation time of inverse kinematic of the robot and singularity analysis. In more detail, a trajectory is a set of frames. It means for each point of the trajectory, we have 6 values, i.e., $x, y, z, \theta, \phi, \psi$; three Cartesian coordination and three rotations around Cartesian axes.

2.3 Relative positions of the hexapod and the Fanuc robot

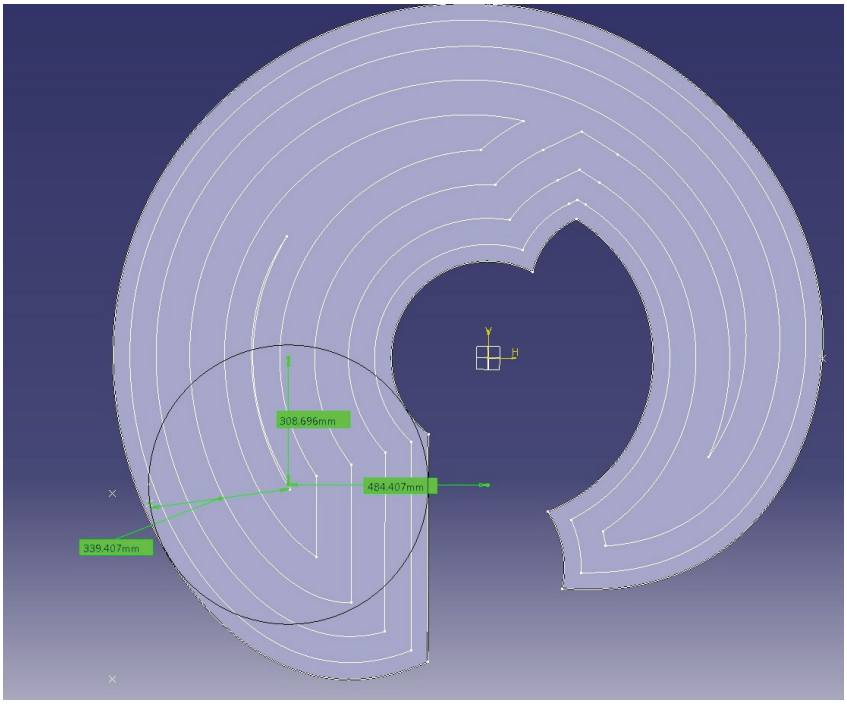


Figure 2.7 Applying the concept of closed-curve offset to obtain the center of the maximum embedded circle in the workspace of the Fanuc robot (Taken from Kaloorazi *et al.* (2015)).

Since there are three robots working together to do the fiber placement, the relative positions of them are important. For example, they must not be too far or too close to each other. Rotary table is fixed to the center of EE of hexapod, therefore only relative position of hexapod to Fanuc is of interest. In this section, having the maximum shared workspace between the Fanuc robot and the hexapod is important. If we put the center of the workspace of the hexapod in the

center of the workspace of the Fanuc robot, we have the maximum shared workspace. Based on (Kaloorazi *et al.* (2015)), a geometrical method in a CAD software is used to find the center of the workspace of the Fanuc robot. The workspace of the Fanuc robot and the obtained center are represented in Fig. 2.7. The obtained center point is also known as the center of the maximum embedded workspace circle. This task has been done in Catia, using an offset algorithm. In Fig. 2.8, the relative positions of the two robots is represented. Note that in this figure, the singularity of the Fanuc robot is not considered and as it can be observed, the Fanuc robot is in a singularity.

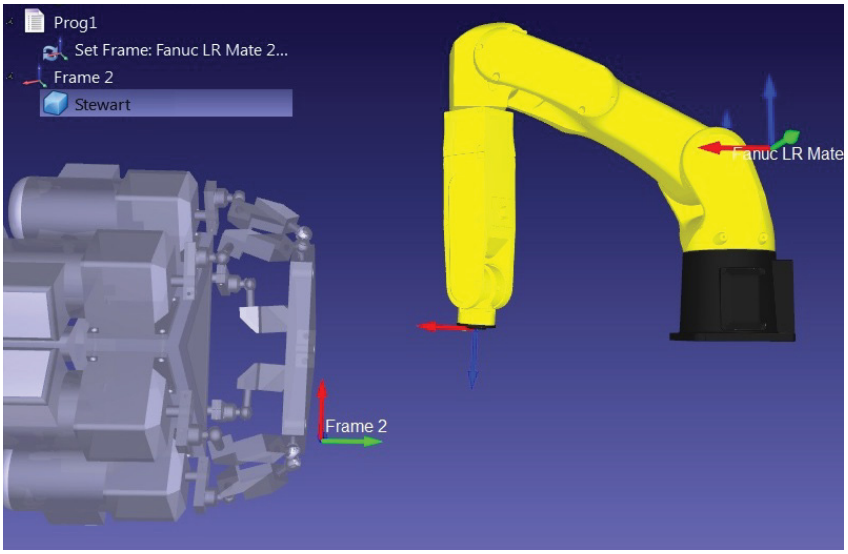


Figure 2.8 A first try to place hexapod in Fanuc's workspace. The singularity of Fanuc is not considered and also it is not applicable regarding the constraint of being on the same level as Fanuc.

We assume that the base of the Fanuc robot is fixed and we want to find a proper location to place the hexapod. In general, the hexapod can be placed in the 6D workspace of the Fanuc robot. However, we have a constraint of placing it on the floor (at same level as the Fanuc robot). Therefore, we do not have the freedom of displacement along the z axis and rotation about x and y. Furthermore, the first joint of the Fanuc robot has a wide range of rotation about the z axis, so it is unnecessary to have a displacement along the y axis. Also, the rotary table rotates about the z axis and there is no need to consider this freedom for the hexapod

placement. The only freedom we are left with is the displacement along the x axis. The center of the embedded circle in the workspace of the Fanuc robot is represented in Fig. 2.9. The result of the hexapod placement is represented in Fig. 2.10. As it can be observed, the base-to-base distance between the two robots is chosen in such a way that the workspace center of the hexapod be placed in the center of the offset curves of the Fanuc robot. However, almost half of the Fanuc robot's workspace is under the ground level and practically lost. It is suggested that the Fanuc robot be mounted on a riser to be able to take advantage of its full workspace.

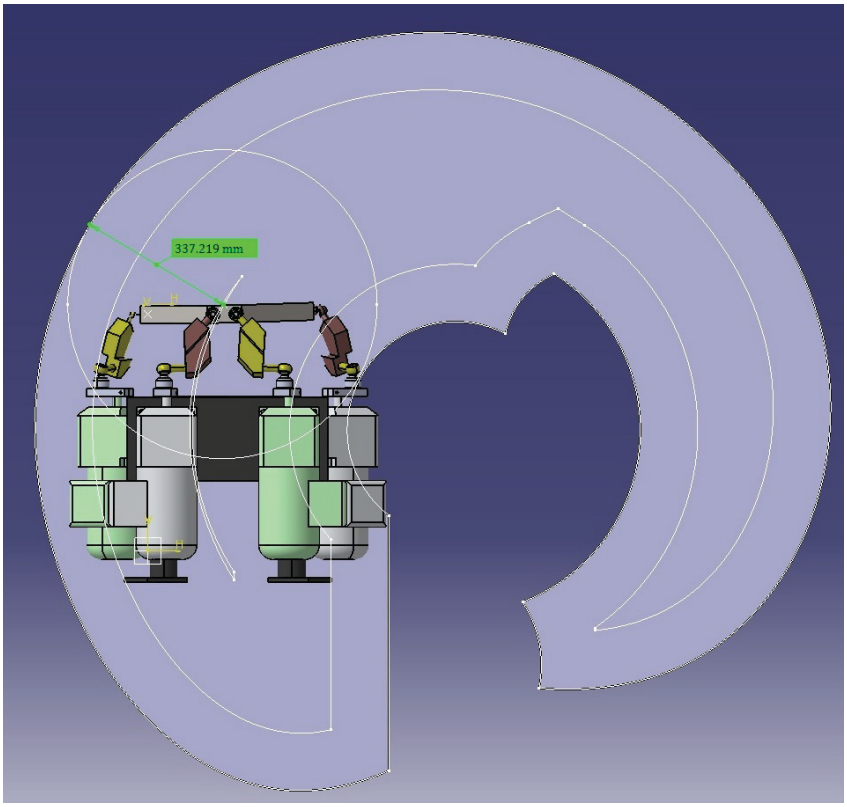


Figure 2.9 Maximum embedded circle in the workspace of Fanuc, regarding same level placement.

2.4 Particle swarm optimization

The reason of using PSO to find the optimum solution of the problem is the higher chance of finding a global optimum in the search space in comparison with other methods. This is the

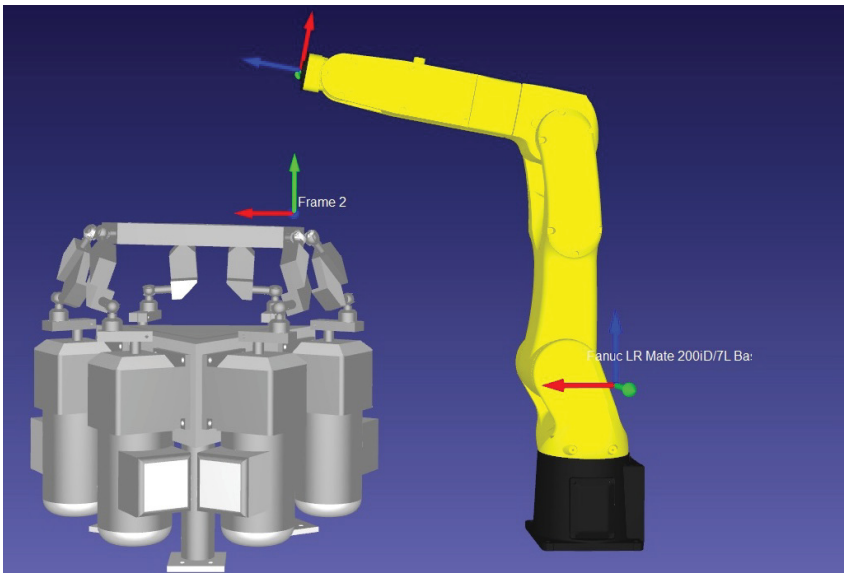


Figure 2.10 Final placement of hexapod relative to Fanuc.

case for evolutionary algorithms that they do not guarantee, but have a chance to find the global optimum of the problem.

In contrary to classic methods, such as Newton's method and SQP, evolutionary methods do not depend on the gradient and/or Hessian of the function. Therefore, they can be used in a vast field of applications, in which the gradient and Hessian is very expensive to obtain or even not available at all. In our case, the gradient and Hessian of the objective functions are very hard to obtain and also we want to find the global optimum of the problem. Among all the evolutionary algorithms, PSO has been chosen. Because it is easy to implement, the convergence speed is higher and its Number of Function Call (NFC) is less than GA (Kennedy (2011)). Unlike conventoinal evolutionary algorithms, such as GA, PSO does not use selection (Kennedy (2011)). PSO is based on social intelligence. In Fig. 2.11a, the main concept of PSO is to find the deepest part of the lake. Each member shares their observations with others, and for the next move, each individual decides on its direction based on the entire swarm's information. Furthermore, in this study, some features are added to the basic PSO in order to improve the algorithm's functionality. The PSO algorithm is briefly explained in what follows; for more information on the origin of PSO, see (Kennedy (2011)).

As a basic interpretation of PSO, a population of swarms includes the candidate solutions, i.e. particles, (Zhang *et al.* (2015)). The particles move around in the initiated search-space based on the simple PSO formulae. Each particle is driven by its personal best memory and the entire flock's best memory. In each iteration, improved positions are discovered, which will be used to guide the movements of the particles in the next iteration. A series of iterations will hopefully lead to the global optimum. However, as with all numeric optimizations, there is no guarantee of finding the absolute global optimum solution. The chances of finding the global optimum increase with more iterations, more particles, and greater fine-tuning of the PSO parameters.

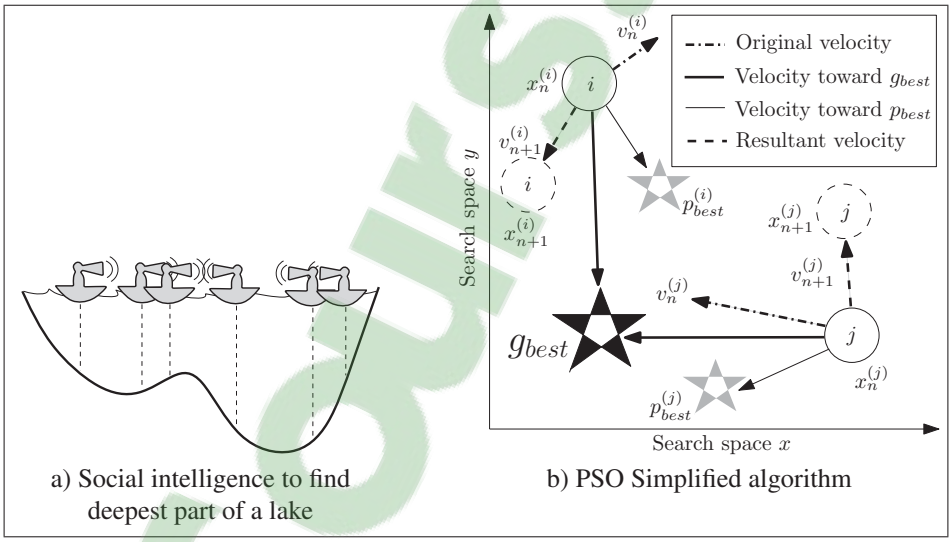


Figure 2.11 PSO (a) to find the deepest part of the lake and (b) a simplified representation of the involved parameters. Choosing the search direction of two particles in the swarm, i.e. particle *i* and *j*. g_{best} is the global best in the current iteration. p_{best} is the best memory of the particle itself. The resulting velocity is a combination of personal and global best memories, as well as the previous velocity of the particle and an element of randomness to shuffle the particle around.

A general overview of the velocity vector is presented in Fig. 2.11b. In this figure, the velocity of a particle for each iteration is being calculated by a combination of the previous velocity and best known personal and global positions. The PSO velocity update equation is as following:

$$\mathbf{v}_{i,d} \leftarrow \omega \mathbf{v}_{i,d} + \phi_p r_p(\mathbf{p}_{i,d} - \mathbf{x}_{i,d}) + \phi_g r_g(\mathbf{g}_{i,d} - \mathbf{x}_{i,d}), \tag{2.44}$$

where ω, ϕ_p, and ϕ_g are tuning parameters, which must be chosen carefully in order to achieve reliable results. Velocity $\mathbf{v}_{i,d}$ is the current velocity of the i^{th} particle for the d^{th} dimension; $\mathbf{x}_{i,d}$ is the current position of a particle; $\mathbf{p}_{i,d}$ is the best personal memory of the i^{th} particle; and $\mathbf{g}_{i,d}$ is the best global memory of the whole swarm.

During the procedure, it is probable that for a particular member of the path, while applying the velocity vector, that member locates outside of the optimization constraints – in other words, outside the robot's workspace. In this case, the reflection effect will be applied to the particle. The next section explains this feature in greater detail.

2.5 Methodology

The optimization objective of the proposed method is to find the best placement of a given path in the SM workspace. The optimization variables are placement components in the Cartesian space, i.e. the position and orientation of the path. Path $\mathbf{T}_{4\times4\times n}$ is the input of the optimization algorithm, which is a set of homogeneous 4×4 matrices representing n points in the path.

The objective is to be as far as possible from singular poses of the SM. There are several indexes we can use to measure the performance of a robot, such as condition number, manipulability, dexterity, etc. (Merlet (2006b)). However, none of these can be considered the best metric, due to scale and unit differences. In this chapter, distance-to-the-singularity index is simply defined as absolute value of Jacobian matrix determinant of SM. The bigger this value, the further from the singularity is the robot. Therefore, the obtained optimized placement is a solution with the biggest determinant value. The joints limits of the SM are considered as the constraints of the optimization. For the hexapod, the joint limits are only considered in the transition phase, which will be discussed in upcoming sections.

The optimization problem can be structured as follows:

$$\begin{aligned} \max f &= -\sum_{p=1}^{m} |\det(\mathbf{J}_p)| \\ \text{s.t.} \quad & \mathbf{q}_{i_{\min}} < \mathbf{q}_i < \mathbf{q}_{i_{\max}}, \end{aligned} \tag{2.45}$$

where $\mathbf{J}_p$ is the Jacobian matrix of SM for point p, f is absolute summation of determinant of $\mathbf{J}_p$, for all the points in the path ($p = 1 \ldots m$). The optimization constraint is the joint limits of the SM, which must be within a certain range. Optimization parameters are 12 placement parameters plus 1 DOF for rotary table actuation (13 in total). Once the placement of path on the rotary table and rotary table on the hexapod is known and joint value of rotary table is set for each point of the path, $\mathbf{J}_p$ is a unique value for each point and performance index consists of summation of all $\mathbf{J}_p$.

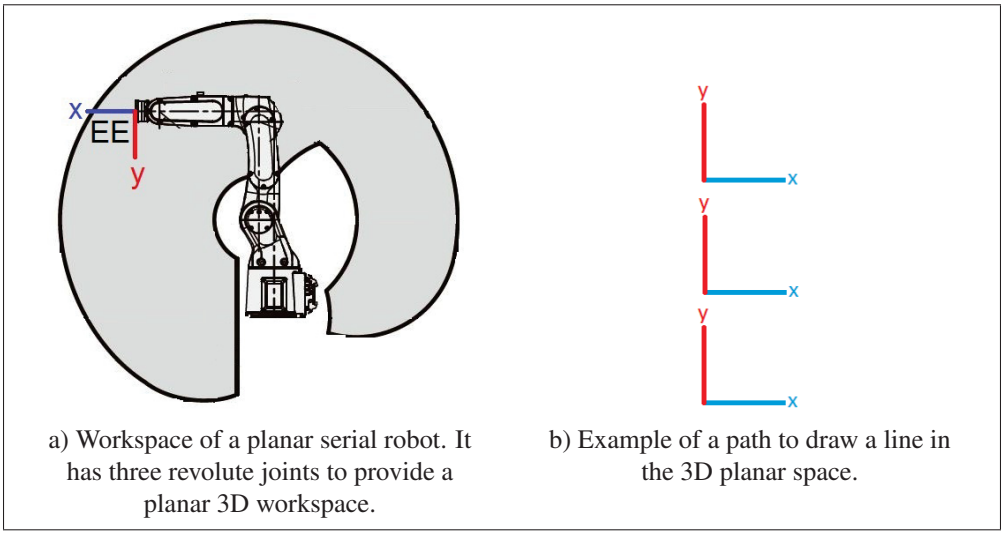
The problem of obtaining the best path placement mainly arises from the nature of the coordinated robotic workcells. The optimization is performed in a mixed space of Cartesian space (12 placement parameters) and joint space (1 rotary table actuation). Using PSO, each particle represents a specific place for the path and rotary table, and represents rotary table joint value. Just like in the example in Fig. 2.11a, the particles search throughout the workspace by testing the path in different places and sharing results in order to calculate individual velocities for the next iteration. For the first iteration, a population of the particles is randomly generated within the hexapod's workspace.

In general, there are six design parameters in a 6D search space, i.e. the pose of the reference frame of the path to the moving frame of the hexapod. The number of objective functions equals the number of points in the path. For example, in a three-point path, three independent values will be obtained for the cost function of a particle. This can be regarded as a multi-objective optimization problem. Various methods can be used to transform a multi-objective problem to a single-objective one; this study uses a simple summation, which means all the points in the path have the same effect on the cost function. The single cost value of a particle is obtained by the summation of the costs of all points divided by number of points. For future

works, we recommend using a weighted sum function, in which it is possible to have different levels of importance for different points in the path. For example, if the points around the junction of the workpiece are more important than the points around the main branch, one can add more weight to the former points, and therefore obtain those results that are more favorable to junction points.

At the first iteration, a hypercube space (depending on the number of optimization parameters) is assigned as the search space. It is possible for the particles to violate the initial search space during subsequent iterations. Therefore, another feature, called the reflection effect, can be added to the basic PSO. In the reflection effect, when a particle goes outside of the initial search space boundaries, a reflection function brings it back inside the search space. The formulation of this add-on is simple. By adding only the negative value of the velocity vector, the particle is brought back inside. However, sometimes it is better to not have this add-on in the algorithm, because in its presence, the final answer is forced to be found only inside the initial search space. Without this add-on, because of a randomness coefficient in the velocity equation, a particle can search beyond the initial search space boundaries. In this chapter, the reflection effect is not applied for the initial search space, but it is applied for the constraint, i.e. the mechanism workspace.

For all evolutionary algorithms, tuning of the code's parameters is an important task. A basic way to do so is simple trial-and-error. In some particular cases, partitioners may use a lighter optimization algorithm to tune the parameters of the main algorithm. This can be helpful in large-scale projects where an optimization routine will be repeated over and over. In this project, we used the simple trial-and-error method. In evolutionary algorithms, it is probable that running the algorithm multiple times will lead to different results. The degree of difference depends on population size and the number of iterations: the greater the population and number of iterations, the more similar the results.



a) Workspace of a planar serial robot. It has three revolute joints to provide a planar 3D workspace.

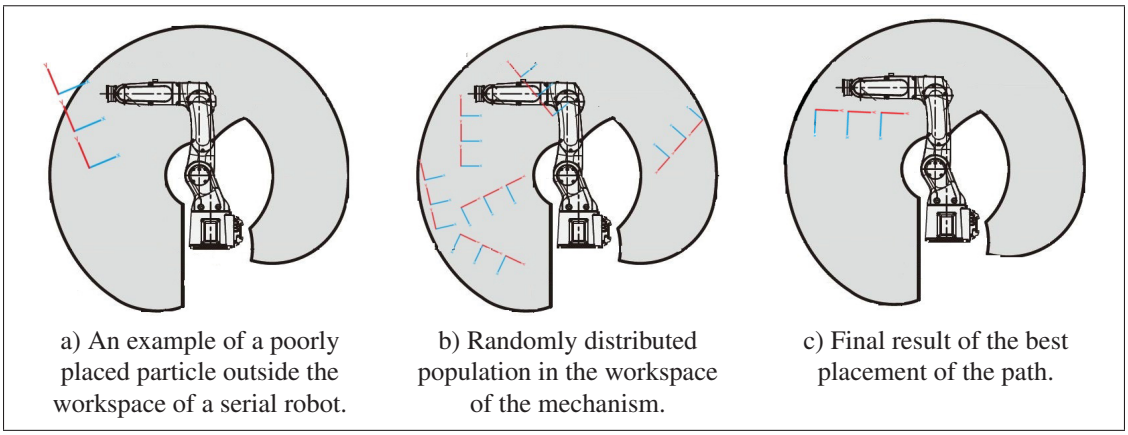
b) Example of a path to draw a line in the 3D planar space.

Figure 2.12 Items involved in the placement problem: (a) the robot; (b) the path.

2.5.1 Simplified illustration

In order to have a better understanding of the algorithm, a simple example of the process is presented as follows. In this example, a simple path of three points in a planar space (x, y, ϕ) is taken into account (as shown in Fig. 2.12b). In this case, there are three design variables (x, y and ϕ of the reference frame of the path, i.e. the first point), and three objective functions (three points). An example of a planar serial robot's workspace is presented in Fig. 2.12a. Summation of the three objective functions is considered as the final objective function to be optimized. The goal of the algorithm is to maximize the net distance of the path from the singular poses of the planar robot. This is computed using the determinant of the Jacobian matrix of the robot at each point of the trajectory. An example of a poorly located particle is shown in Fig. 2.13a. In this case, we bring the particle back into the workspace and apply a small random movement on it. Fig. 2.13b presents an example of a PSO problem's initial population, which is randomly distributed in the workspace of the robot. A pseudo-code of the algorithm is presented in Algorithm 2.1 The final result of the simple path placement example is represented in Fig. 2.13c. As it can be observed in the figure, the path is more inclined toward the middle of the workspace to be far from the borders (end of workspace singularity).

Furthermore, it is oriented in such a way that the average of third joint's angle is close to 90 degrees.



a) An example of a poorly placed particle outside the workspace of a serial robot.

b) Randomly distributed population in the workspace of the mechanism.

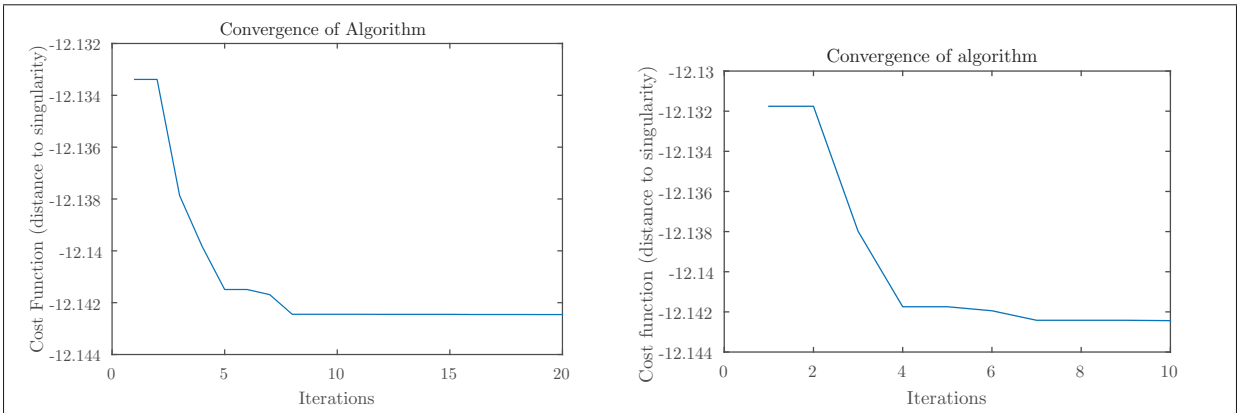
c) Final result of the best placement of the path.

Figure 2.13 Different aspects of the path placement and the final result.

In this optimization, the number of particles in the population, *nPop*, is 30, and the maximum number of iterations, *MaxIt*, is 20. "Maximum number of iterations" means that the process may end before this number of iterations. That is because there is another criterion applied here: *If changes in the cost are less than ε, then terminate the process.* When the latter criterion is inactive, the number of function call is roughly $NFC = 20 \times 30 = 600$. This process is done in 0.7067 seconds by a 3.5 GHz processor. The result of the optimization, not considering the stopping criteria of ε, is presented in Fig. 2.14a. When we add stopping criteria of ε in such a way that *if for the last 4 iterations, the changes in the cost are less than $\varepsilon = 0.002$, then terminate the process*, the result is as shown in Fig. 2.14b and the number of function calls is $NFC = 10 \times 30 = 300$, which produces a 0.3536 second process.

2.6 Redundancy optimization

The previous section introduced a method of finding the best placement for the path inside the workspace of the SM. In this section, we apply the method to the 13 DOF workcell to take advantage of the redundancy and use it to find the best kinematic solution for the robots at hand. The purpose of the optimization is to be as far as possible from the singular poses of the



a) Without any stopping criteria.

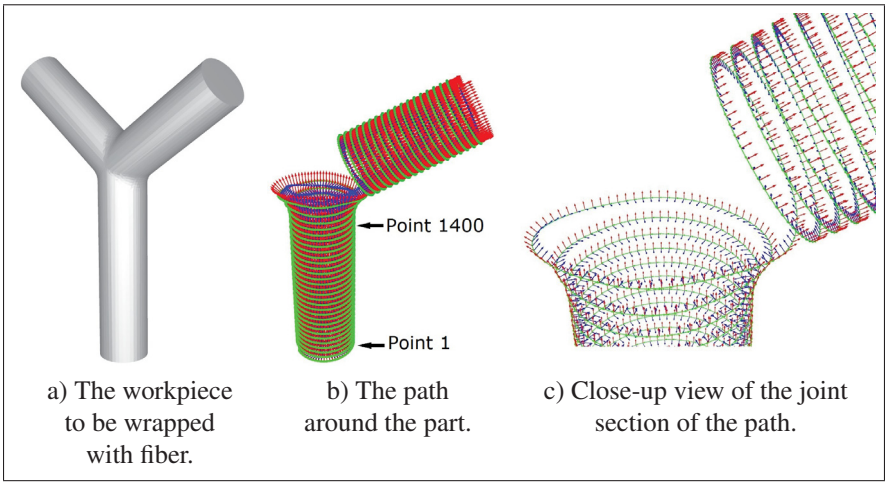
b) With a stopping criterion of $\varepsilon < 0.002$.

Figure 2.14 Convergence of the algorithm. The general stopping criteria is a limit of 20 iterations.

SM. Here we only consider the singularity of the SM based on the determinant of the Jacobian matrix. The singularity of the hexapod is not taken into consideration, because there is no singularity within the limited hexapod's workspace.

Recalling the general goal of the project – fiber placement – we consider a Y-shaped workpiece to be wrapped. The rotary table is mounted on the hexapod, and the central axis of the main branch of the Y-shaped workpiece is (ideally) coincident with the rotation axis of the rotary table. (Later we will see that the algorithm works even if the rotation and central axes are not coincident.)

The strategy to follow in this algorithm is to manipulate only the SM and the rotary table during the fiber placement of each branch, and keep the hexapod still. In other words, if we have three branches in a part – which is the case for the Y-shaped workpiece – the hexapod moves once before we wrap the first branch to place the entire branch in an optimized place; then the rotary table and the SM do their jobs to wrap the branch. For the second branch, the hexapod moves to place the second branch in the optimized pose. (Note that this pose is not necessarily a good pose for the first branch, but as it has already been wrapped, this is not a problem). Likewise, for the third branch, hexapod moves in order to place the branch in its optimized pose. While each branch is being wrapped, a system of 7 DOF is active (including the SM and rotary table).



a) The workpiece to be wrapped with fiber.

b) The path around the part.

c) Close-up view of the joint section of the path.

Figure 2.15 The initial path around the workpiece. Each point is represented as a frame consisting of the origin and RGB as *xyz* axes. The path has 2795 points starting from the bottom. Points 1 to 1399 are a simple helix around the main branch, and point 1400 is where the junction appears (Taken from Hely *et al.* (2017)).

There is a transition phase between each branch during which the rotary table and the SM are fixed and the hexapod moves the entire workpiece to place it in the optimized pose in the workspace of the SM. Note that during the transition phase, the SM is attached and fixed to the workpiece. Since the workpiece is being moved by the hexapod, the SM has to move to maintain its EE on the transition point of the part; however, the movement of the SM is not a part of the optimization, and it is merely guided by the hexapod. This is very important, since the fragile fibres are easily broken under shear stress.

The procedure of this algorithm includes a 6 DOF optimization to obtain the optimized pose of the hexapod (i.e., the placement of the rotary table). Each possible hexapod pose is a particle in the population. For each particle, a 1D optimization is done to obtain the rotation of the rotary table. The 1D optimization is simply performed using a method based on NR. Start of the NR is chosen to be the rotary angle of the previous point of the path. Since the points in the path are usually very close to each other, the 1D optimization converges within a few iterations (although this still depends on the desired precision). The 1D optimization is applied to all the points in the path.

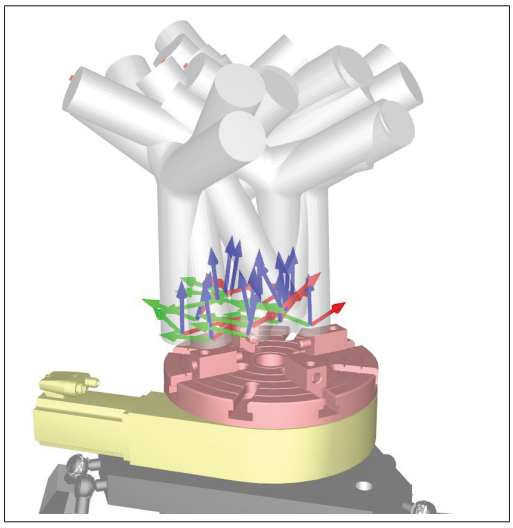


Figure 2.16 The initial population of the path's reference frame is generated randomly within the hexapod workspace.

2.6.1 The pseudo-code of the algorithm

Algorithm 2.1 represents the pseudo-code of the proposed method. The input of the algorithm is **path** as the path around the workpiece. The path is given as a set of frames perpendicular to the surface of the workpiece. In particular, an array of transformation matrices ($\mathbf{path}_{4\times4\times p}$). Geometrical parameters of the robots are presented in the DH table for the SM and link lengths of the hexapod. These parameters will be used to compute the IKP of the mechanisms as well as the Jacobian matrix and singularity index. $\mathbf{R}_{rot}(\theta)$ represents a rotation matrix of θ degrees around the rotary table's rotation axis. The path has m points and the population of the swarm is n. Lines 3 to 10 show the first iteration in which a random population is generated and for which the 1D optimization has been done. In line 4, a 4×4 transformation matrix, i.e. **Trans**, is randomly generated within the workspace of the hexapod, i.e. PM_w. In line 5, the random transformation matrix is multiplied to the whole path, resulting in the i^{th} *Particle*. Note that "{:}" refers to all the members of a set. In lines 6 to 8, the cost function is calculated for each point of the particle. In line 7, the 1D optimization is done by a simple NR method, i.e. `fmaxsearch`. Function `singularity_dist` computes the determinant of the Jacobian matrix of the SM; it returns the distance to the singularity. In the 1D optimization, θ as the

Algorithm 2.1 The pseudo-code of the optimization algorithm, based on PSO, to obtain placement of a given path in the 13 DOF workcell under study.

1 **Input**: **path** as trajectory on the part; Geometrical parameters of the robots; $\mathbf{R}_{\text{rot}}(\theta)$ as the rotation matrix about the rotary table's axis.
2 **Parameters**: n number of particles; p number of points in the path.
3 **for** i from 1 to n **do**
4 $\quad \mathbf{rnd}_{\mathbf{Trans}} = \texttt{rnd}(\mathbf{Trans}, PM_w)$;
5 $\quad Particle\{i\} = \mathbf{rnd}_{\mathbf{Trans}} * \mathbf{path}\{:\}$
6 $\quad$ **for** j from 1 to p **do**
7 $\quad\quad [dist\{j\}, \theta_{\text{best}}\{j\}] =$ $\texttt{fmaxsearch}(\texttt{singularity\_dist}(\mathbf{R}_{\text{rot}}(\theta) * Particle\{i\}\{j\}))$
8 $\quad$ **end**
9 $\quad ParticlePersonalBest\{i\} = \texttt{sum}(dist\{:\})$
10 **end**
11 $ParticleGlobalBest = \texttt{max}(ParticlePersonalBest\{:\})$
12 **while** desired precision **do**
13 $\quad$ **for** i from 1 to n **do**
14 $\quad\quad Particle\{i\}$ = Eq. 4.1
15 $\quad\quad$ **for** j from 1 to p **do**
16 $\quad\quad\quad [dist\{j\}, \theta_{\text{best}}\{j\}] =$ $\texttt{fmaxsearch}(\texttt{singularity\_dist}(\mathbf{R}_{\text{rot}}(\theta) * Particle\{i\}\{j\}))$
17 $\quad\quad$ **end**
18 $\quad\quad$ **if** $ParticlePersonalBest\{i\} < \texttt{sum}(dist\{:\})$ **then**
19 $\quad\quad\quad ParticlePersonalBest\{i\} = \texttt{sum}(dist\{:\})$
20 $\quad\quad$ **end**
21 $\quad$ **end**
22 $\quad$ **if** $ParticleGlobalBest < \texttt{max}(ParticlePersonalBest\{:\})$ **then**
23 $\quad\quad ParticleGlobalBest = \texttt{max}(ParticlePersonalBest\{:\})$
24 $\quad$ **end**
25 **end**
26 **Output**: *ParticleGlobalBest* as the hexapod's poses, $\theta_{\text{best}}\{:\}$ as the rotary table's path.

rotation angle of the rotary table is the variable to be optimized. The objective is to maximize the distance from the singular poses of the SM. $Particle\{i\}\{j\}$ refers to the j^{th} point in the i^{th} particle of the population. The output of line 7 is the max distance-to-singularity of the j^{th} point, i.e. $dist\{j\}$, and also the best θ. Line 9 shows that in the initialization iteration, the best personal memory of a particle *ParticlePersonalBest*, is the summation of all the distances

through out the path, i.e. $dist\{:\}$. In the next iterations, the *ParticlePersonalBest* may change if a better result is obtained.

In line 11, the particle having maximum *ParticlePersonalBest* among all, is the global best and is stored in *ParticleGlobalBest*. Lines 12 to 25 are the iterative part of the algorithm, and it continues until the desired accuracy of the result is achieved. It is basically the same routine as lines 3 to 10. The difference is that in line 14, instead of random generation, this time based on the PSO logic represented in Eq. (4.1), the position of the particle is determined. Another difference is in lines 18 and 22, in which the obtained results for the current iteration are compared to the previous best results. If the new result is better, it replaces the old result; if not, the old result is still the personal and global best.

It is probable that for one point in the path, the determinant of the Jacobian is a positive value and for the next one is negative value (instead of zero for both of them). In this case, the algorithm obtains a safe distance to singularity for both points, but in fact the SM has to cross a singularity, i.e. change the working mode in order to move on to the next point. Therefore, one must check if the sign of the determinant has changed. If it has, the particle is completely useless, and ∞ is returned as the cost function. Furthermore, in the application of the fiber placement, a sudden direction change for the rotary table is not allowed, because it may cause the fiber to break. Such a condition has been considered in the code in such a way that the speed and direction of the rotary table is limited.

This algorithm is extensible to any redundant robotic workcell, and it is possible to modify the optimization criteria and add more of them. As always, computational time depends on the desired accuracy and complexity of the system. In this example, for each 1D optimization the IKP of the SM is solved and the determinant of the Jacobian is computed. Based on the symmetry of the path, some sections have been simplified in order to decrease the number of points. In this study, there were 10 iterations for 20 particles, performed with a 2.3 GHz processor and 8 GB of ram, and it took about 21 minutes to converge on the results.

2.6.2 Results

An example of a helix-type path around the Y-shaped workpiece is presented in Fig. 2.15. In accordance with the concept of PSO, the first population is generated randomly. An example of the random generation within the workspace of the hexapod is represented in Fig. 2.16. The frames shown in the figure are the reference frame of the path and basically each frame represents a particle. At the end of the first iteration, we have a population of the random poses of the hexapod and for each random pose, the rotation angle of the rotary table is optimized. The hexapod pose is the same as the placement of the rotary table with respect to the SM, with a fixed offset from the hexapod's EE, In the next iteration, the velocity of the particles change with respect to Eq. (4.1). In this iteration, the hexapod pose has somewhat improved, and again a 1D optimization runs for each point of the path. Finally, after a number of iterations, the pose of the hexapod is optimized and the best rotation angle for the rotary table is obtained for each point of the path.

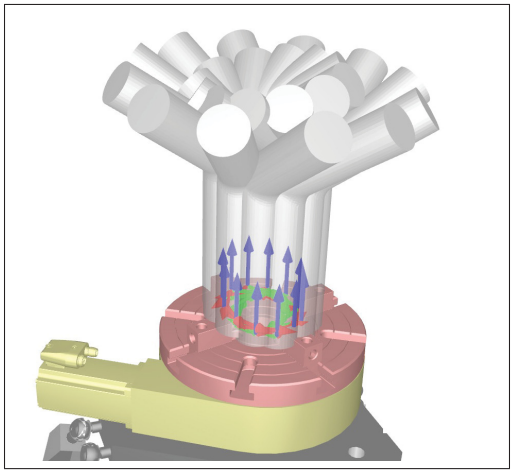


Figure 2.17 Some possible angles of one point of the trajectory are optimized by a 1D optimization. The result is an optimized angle of the rotary table for this point.

The concept of the 1D rotary table optimization is illustrated in Fig. 2.17. This figure only shows one point of the path, for which the rotary table is able to move the point in a circular manner. For each point of the path, the optimized rotation angle of the rotary table, with respect

to the objective function, is obtained. For each particle, a series of rotation angles associated to the points of the path is then obtained. The summation of the objective functions, (i.e., the determinant of the Jacobian matrix of the SM) at all the points of the path represents the total cost of the particle.

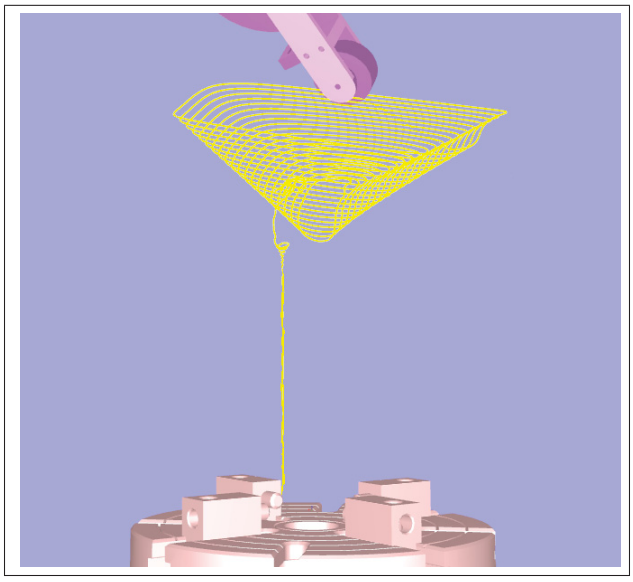


Figure 2.18 The footprint of fiber placement tool tip, at the end of the fiber placement process. The yellow curve is the final trajectory for the SM to follow. It is a linear trajectory for the main branch and an expanding helix for the second branch.

The tool tip footprint of final trajectory for the SM to follow is presented in Fig. 2.18. It is a semi-linear trajectory from point 1 to 1399 in the path. Since the axis of the main branch coincides with the rotation axis of the rotary table, and the path is a helix, the trajectory is a line for the main branch. For the rest of the path from point 1400 to the end, it is an expanding helix. The transition point is defined by the practitioner. In this study, point 1617 is the transition point. In the transition point, the rotary table holds its angle and the tip of the SM's tool is fixed to the transition point. Meanwhile, the hexapod adjusts itself to a new pose, which is the optimum for the second branch. Then after the transition is done, the hexapod will stay fixed and the SM and rotary table start working and continue to do their fiber placement. Also, the trajectory for the rotary table to follow during the fiber placement is constantly decreasing in angle, see Fig. 2.19. For points number 1 to 1399, it is a linear descent, and therefore it is

neglected for the sake of a better representation. Note that it is essential for the rotary table to be capable of infinite rotation. The joint trajectory for the SM to follow is presented in Fig. 2.20. From point 1 to around 1500, it has almost zero movements, since it is associated with the main branch. The transition phase happens at point 1617. After this point it has a periodic movement. One can observe that for the linear part, joint 5 (J_5) operates near 90 degrees, for which it is the furthest distance to the singularity (given the fact that $J_5 = 0, 180$ is a singular pose for the SM). Afterwards the average movement of J_5 is around 90 degrees. J_5 stays within the range $J_5 = [78, 128]$, which is a safe margin for the SM to operate.

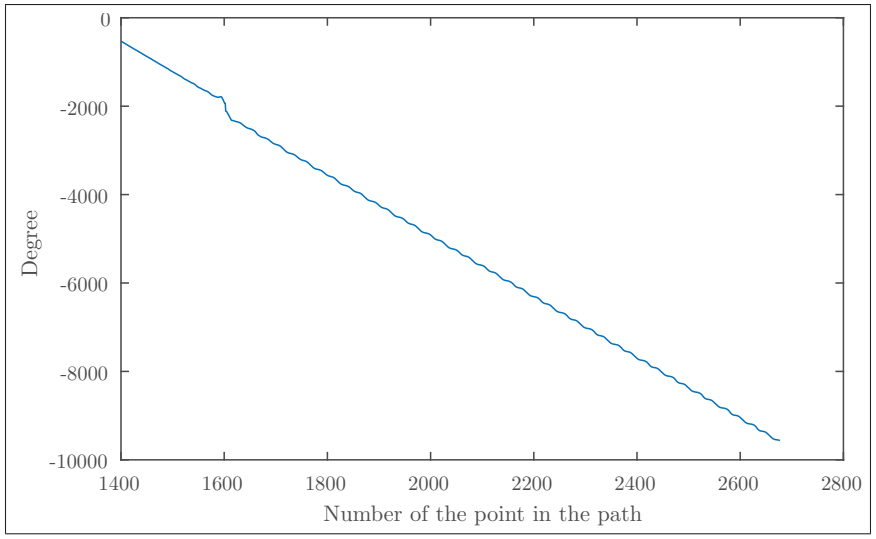


Figure 2.19 The trajectory for the rotary table to follow. It is linear for points number 1 to 1399, and therefore it is neglected for the sake of a better representation.

The final pose of the hexapod and the placement of the rotary table on the hexapod is presented in Fig. 2.21. As it can be observed, the final placement does not resemble a conventional workcell. In this case, either the hexapod on the floor, or the rotary table on the hexapod has to be placed in such a way that satisfies the obtained result. An easy way to do so is to use a spacer in order to provide the fixed transformation from the base of the rotary table to the hexapod EE. The hexapod is fixed to the floor at a distance of $x = 1549$ mm from the world frame (SM Base). The optimized placement of the rotary table is actually to be defined in the world frame (restricted inside the workspace of the hexapod), and its placement on the hexapod

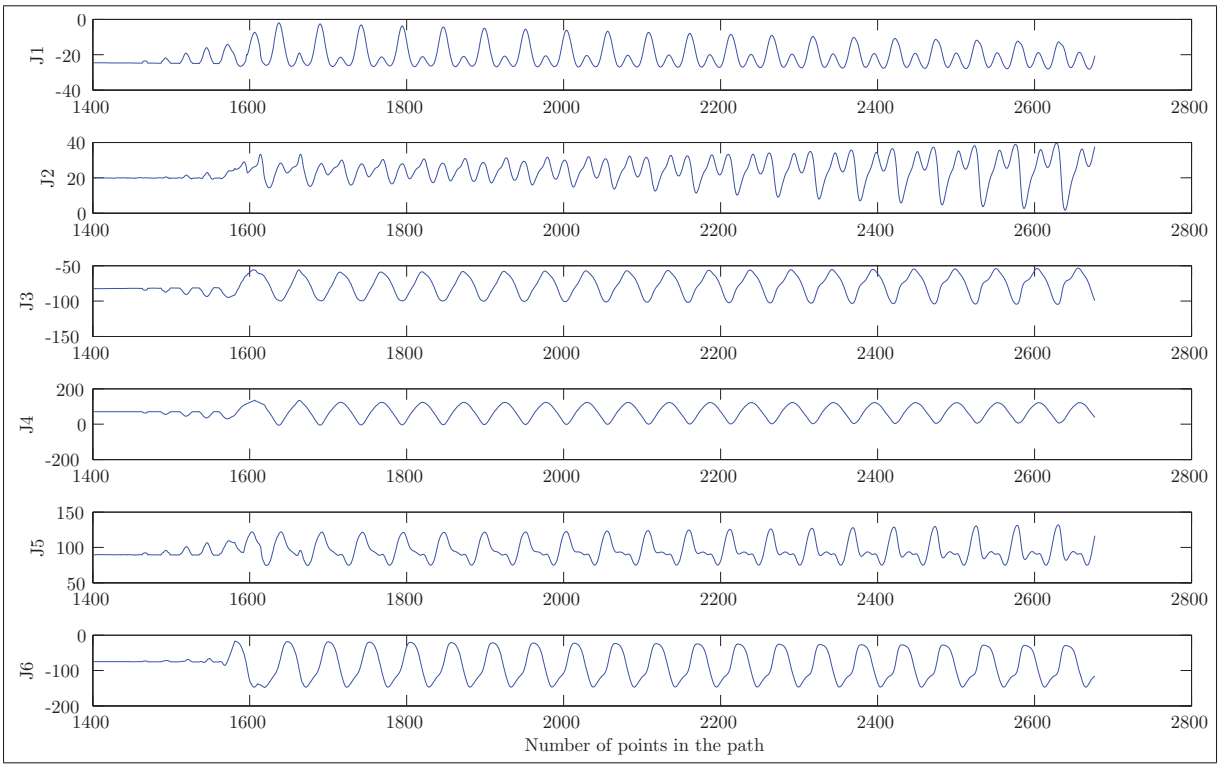


Figure 2.20 The trajectory for the SM to follow. It is linear for point numbers 1 to 1399, and therefore it is neglected for the sake of a better representation.

can actually be variable. The start pose of the hexapod depends on the size and shape of the spacer. However, in order to take advantage of the full hexapod workspace, the rotary table is fixed on the moving frame of the hexapod in such a way that the hexapod's starting point is aligned with the center of the T&T workspace. Therefore, the fixed frame of the rotary table is represented in the moving frame of the hexapod as following: $\{x = 56,\ y = 56,\ z = 144,\ \theta = 23,\ \phi = 75,\ \psi = 16$, in Euler *zyx* convention.

2.7 Conclusion

In this chapter, an optimization method based on PSO was introduced. A redundant coordinated robotic workcell of 13 DOF – including a 6 DOF SM, a 6 DOF hexapod parallel mechanism, and a 1 DOF rotary table – was mounted on the hexapod. The workcell's task was to wrap a fiber around a Y-shaped workpiece to create a composite material. The optimization had

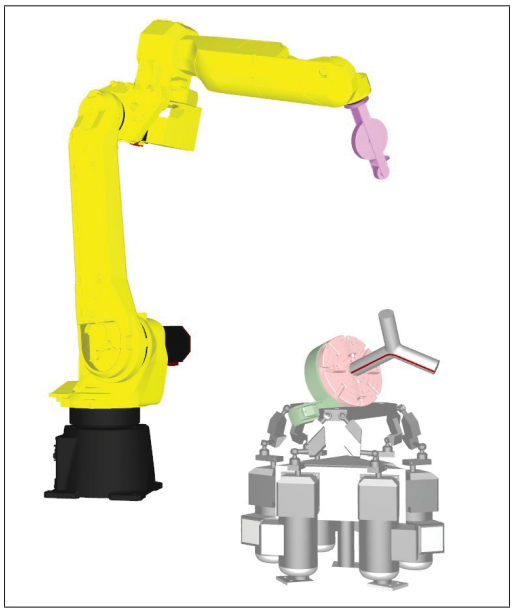


Figure 2.21 The final placement of the rotary table on the hexapod. It is not conventional to place the rotary table with an orientation to the surface of the EE. A spacer is needed in this situation in order to place and fix the rotary table on the hexapod.

two objectives: first, for a given path on the Y-shaped part, to obtain a solution regarding redundancy in the DOF; and second, regarding the singular poses of the SM, for the obtained solutions be far from singularities. The optimization method is basically a 1D optimization within a 6D optimization. The procedure was explained through an illustrative example, and the result for a sample path was presented.

CHAPTER 3

PARAMETERS IDENTIFICATION OF THE PATH PLACEMENT OPTIMIZATION PROBLEM FOR A REDUNDANT COORDINATED ROBOTIC WORKCELL

3.1 Introduction

Path placement is an inevitable part of any robotic task. Sometimes the given path is simple and it is not important to find the best placement of the path within the workspace of the robot. However, in many cases the given path is complex and numerous factors play a constraining role in the path placement task, such as the fiber placement task represented in the previous chapter (Hely *et al.* (2017)). This problem is common in practical applications, where it is usually solved by trial and error. In this chapter, we introduce a systematic methodology to find the number of independent placement parameters. A path is a set of poses with respect to a workpiece that need to be attained by the tool reference frame in order to do a specific task. For example, in a ski goggles gluing task, the path is the set of poses on the contact edge of the goggles' glass and frame. A trajectory is defined as a set of the joint movements of a robot in order to cover a path.

Path placement is an optimization problem. In order to find the optimum solution to the problem, there are two general approaches: analytical and numerical (Rao (2009)). In a numerical optimization, such as described in the previous chapter, typically many iterations are needed before an acceptable solution is obtained. Therefore, it is important to identify the necessary parameters and optimize only for them. Trying to find a solution among all the available parameters in an optimization problem may lead to a highly resource-consuming process. In most cases, there are some dependent parameters involved. It saves a great deal of time later in the optimization process if the user, before starting the optimization process and developing the corresponding code, analyzes all the parameters at hand and chooses the least number of independent parameters necessary for the optimization process. These are the parameters that will cover all aspects of the problem.

There are many practical applications for coordinated robotic workcells, such as gluing systems (MA & ZHANG (2007)), welding (Ahmad & Luo (1989)), cutting and painting. In coordinated robotic workcells, manipulator motion planning is an important issue, because of the redundancy of the system regarding the manufacturing task. In order to generate the proper motion, the decomposition of the given task into the manipulator motion and positioner motion is required. In the literature, several studies deal with this matter. A conventional method is based on redundancy resolution using the generalized inverse (pseudo inverse) of the kinematic Jacobian (Kazerounian & Nedungadi (1988); Flacco & De Luca (2015); Wu *et al.* (2000); Buss (2004)). The unique solution can be obtained by using the pseudo-inverse in the sense of least squares, which corresponds to the smallest Euclidean norm of the displacement vector in the joint space. However, this method is not capable of generating the time-optimal solution, regarding the velocity and acceleration limitations of the actuators, which are absolutely important in real-life industrial applications. The idea of "master-slave" is another approach to plan the motion for redundant robotic systems (Tabarah *et al.* (1994); Gan *et al.* (2013), in which the trajectories of the "master" manipulator is set at first, then the corresponding conjugate trajectories of the "slave" are determined. Even though this method is quite simple and computationally efficient, assigning the master trajectory totally ignores the redundancy benefits. Furthermore, in this case it is impossible to take into account the actuator limits.

In the literature, there have been several studies on workpiece placement. (Caro *et al.* (2013)) introduces a methodology that aims toward determining the best placement of the workpiece to be machined knowing the elastostatic model of the robot and the cutting forces exerted on the tool. In (Ur-Rehman *et al.* (2010)), the authors deal with the multi-objective path placement optimization for parallel kinematics machines based on energy consumption, actuator torques and shaking forces. In (Vosniakos & Matsas (2010)) the authors discuss that it is desirable to perform the milling operation in regions of the robot's workspace where the manipulability (both kinematic and dynamic) is highest, thereby exhausting the robot's potential to cope with the process. The authors did so by selecting the most suitable initial pose of the robot with respect to the workpiece. In (Hemmerle & Prinz (1991)), the authors numerically solved the

problem of path placement for redundant manipulators. This included the problem of where to place the components (such as other robots, tables, or machining stations) relative to each other, as well as how to resolve the redundancies of the workcell and process in an optimal fashion. None of the aforementioned studies considers the path placement for a coordinated redundant robotic workcell having a second manipulator detached from the main manipulator. They place a path on a fixed platform rather than on a moving platform. Furthermore, the trajectory optimization of the redundant workcell has not been done simultaneously for both the main manipulator and the redundant manipulator.

The path placement problem is, by its nature, an under-constrained problem. In general, there are six parameters needed to define the position and the orientation (pose) of a given path with respect to the robot, i.e., x, y, z, $R(x)$, $R(y)$ and $R(z)$. In this chapter, the goal is to find the number of parameters needed to optimize the path placement for a redundant coordinated robotic workcell, which consists of a 6-DOF SM and a Redundancy Provider (RP). The RP is essentially a moving platform on which the path is to be placed on. Throughout this chapter the basic RPs with one DOF are taken into the account. In the literature the RP is often referred to as a *positioner* (Gao *et al.* (2017c)). In this chapter, we will consider two types of RPs: (1) rotary table, and (2) linear guide. The rotary table (turn table) is an actuated revolute joint. By duality, a linear guide is an actuated prismatic joint. The path is going to be placed on the RP. This coordinated workcell allows us to have a bigger *virtual workspace*. This means that by using the same SM, it is possible to reach some poses on the path that were not reachable in the absence of the RP. The workspace is called virtual because the real workspace of the SM does not change, but by using the RP and moving the given path (either by rotation in the case of the rotary table, or by linear movement in the case of the linear guide) those unreachable poses on the path will be brought into the workspace of the SM.

In the presence of the RP, the number of decision parameters is even more than six, because there are six parameters to place the path on the RP and six parameters to place the RP with respect to the SM. Furthermore, the RP itself provides the user with one DOF which has to be considered in the placement process. Not all the parameters are independent. The goal of this

chapter is to identify how many parameters are independent for each case. Once the number of independent parameters is obtained, the optimization algorithm will be executed considering only those parameters as design variables.

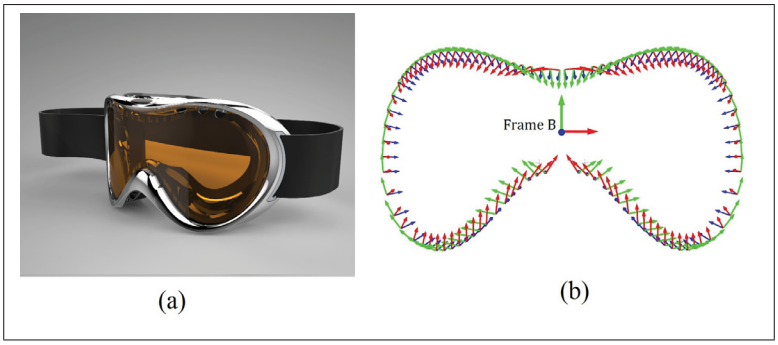


Figure 3.1 (a) Ski goggles; (b) ski goggles path used throughout this chapter as an example. Each point of the path is a frame (origin and orientation). Red, Green and Blue (RGB) are x, y and z axes, respectively.

The remainder of the chapter is organized as follows. First, the problem is introduced and the objective function to be optimized is represented. Then a classical approach to dependency identification is examined. The number of independent parameters for the two cases of the RP is investigated. Then, based on case studies, an innovative approach is proposed. The optimization algorithm is briefly described. Then the result of the optimization process when using only the independent parameters is compared to the optimization when using all the available parameters. The results for the rotary table and the linear guide are also compared. Finally, the conclusion of the chapter is presented.

3.2 Problem description

The workcell under study includes a 6 DOF generic SM and an RP, i.e., rotary table or linear guide. For the sake of better understanding, throughout this chapter a gluing application is considered and the methods are formed around this application. However the method can be generalized to any application. The application is to apply glue on ski goggles. The path on the ski goggles is depicted in Fig. 3.1. The path is relatively complex, in the sense that

the orientations vary. The workcell, including the SM and the rotary table, is represented in Fig. 3.2 and the workcell including the linear guide is represented in Fig. 3.3.

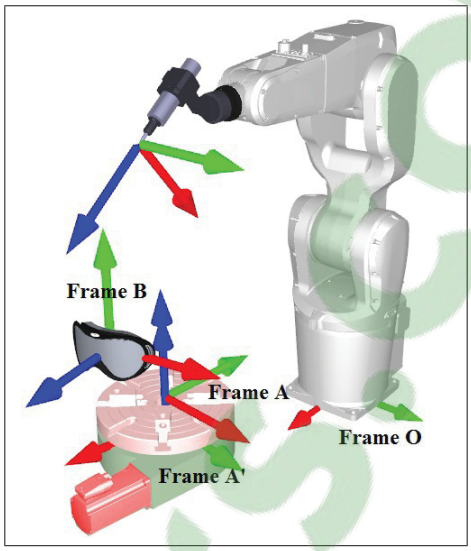


Figure 3.2 The redundant coordinated robotic workcell with a rotary table as the RP. Frame O_{xyz} is the base of the SM and also the world reference frame, frame A'_{xyz} is the rotary table's base frame, Frame A_{xyz} is the moving frame of the rotary table, and B_{xyz} is the reference frame of the path.

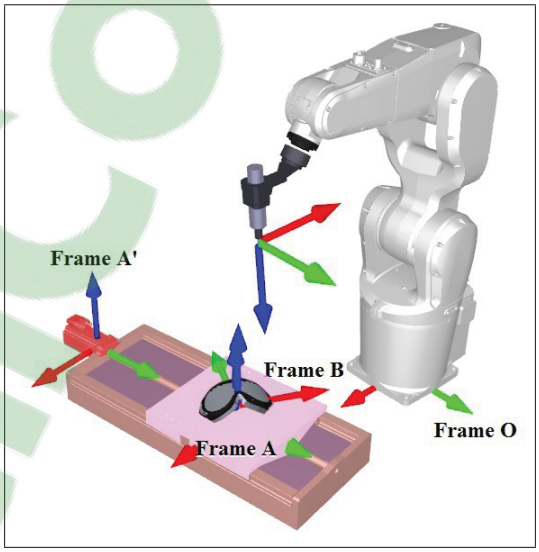


Figure 3.3 The redundant coordinated robotic workcell with a linear guide as the RP. The frames definition is the same as Fig. 3.2.

The problem is defined as the placement of the base frame of the RP (A'_{xyz}) with respect to the global base frame (O_{xyz}); and placement of the reference frame of the path (B_{xyz}) with respect to the moving frame of RP (A_{xyz}). Note that A_{xyz} and A'_{xyz} are related by the RP kinematic equations. Since RP is a simple 1 DOF mechanism, its IKP and the FKP are the same and straightforward. The relation between A_{xyz} and A'_{xyz} can be demonstrated by a 4×4 transformation matrix. In the case of the rotary table the transformation matrix is represented in Eq. (3.1):

$$\mathbf{T}^{A'}_{A\ \text{Rotary table}} = \begin{bmatrix} \cos q_r & -\sin q_r & 0 & 0 \\ \sin q_r & \cos q_r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where $\mathbf{T}^{A'}_{A\ \text{Rotary table}}$ is the transformation matrix corresponding to the moving frame of the rotary table, and q_r is the actuator variable of the rotary table. This is a rotation around the z axis of the rotary table.

Applying the same reasoning, for the linear guide the transformation matrix is represented in Eq. (3.2):

$$\mathbf{T}^{A'}_{A\ \text{Linear guide}} = \begin{bmatrix} 1 & 0 & 0 & q_l \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.2}$$

where $\mathbf{T}^{A'}_{A\ \text{Linear guide}}$ is the transformation matrix corresponding to the moving frame of the linear guide, and q_l is the actuator variable of the linear guide. This is a translation along the x axis of the linear guide.

The workcell at hand is a coordinated robotic workcell which has 7 DOF (6 provided by the SM and 1 by the RP). Both the robot and the RP work together simultaneously and the redundant

DOF provides the option to optimize the trajectory planning of both. In short, the overall optimization process is to find the best placement for the path on the RP and the RP in the workplace of the SM in such a way that optimizes the trajectory planning of the coordinated workcell, i.e., the trajectory planning of the SM. One can remodel the trajectory planning problem of the whole workcell only as the trajectory planning of RP, because once the trajectory of RP is fixed, the number of solutions for the kinematic problem of the SM is finite and depends on its working mode. The objective function of the optimization task is to minimize the cycle time and avoid the singularities. A more detailed explanation of the optimization method is given in the upcoming section.

The general homogeneous transformation matrix is as shown in Eq. (3.3):

$$\mathbf{T}=\begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma & x \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & y \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma & z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.3}$$

where notation c and s stand for sin and cos, respectively. x, y and z are the displacements of a frame with respect to its reference frame. Similarly α, β and γ are orientations of the frame with respect to its reference. For instance, $\mathbf{T}_B^A$ is transformation matrix of frame B with respect to its parent which is A. In the next section the reasoning to find the number of independent parameters is discussed for both cases.

3.2.1 Optimization objective

The optimization objective is to find the best placement of the RP inside the workspace of the SM, and the placement of the path on the RP, in such a way that the SM operates with the minimum cycle time, while maintaining a safe distance from singularities. The singularities are those poses of the end-effector of the SM where the robot loses some DOF and becomes

uncontrollable in some directions (Tsai (1999)). Moreover, the joint limits of the SM are taken into account.

In this chapter, the cycle time refers to the absolute movement of the joints. This can be also interpreted as the energy consumption. Assuming constant acceleration, one can minimize the energy consumption by minimizing the movement. The optimization problem can be structured as in Eq. (3.4).

$$\begin{aligned} \min f &= \sum_{p=2}^{n} |\mathrm{IKP}(\mathscr{X}_p) - \mathrm{IKP}(\mathscr{X}_{p-1})| \\ \mathscr{X} &= g(\mathbf{t}_{k\times 1}, q_{RP}) \\ \text{s.t.} \quad & \mathbf{q}_{i_{\min}} < \mathbf{q}_i < \mathbf{q}_{i_{\max}} \\ & |(\det(\mathbf{J}))| > \varepsilon \end{aligned} \tag{3.4}$$

In Eqs. (3.4), f is the cycle time, $\mathrm{IKP}(\mathscr{X}_p)$ is the inverse kinematic solution of the SM for the target $\mathscr{X}_p$, and n is the number of points in the path. Values of $\mathbf{t}_k$ are the optimization variables, i.e., the placement parameters. Target $\mathscr{X}$ is a pose in 6D Cartesian space and a function of the placement ($\mathbf{t}_k$) and the RP actuator (q_{RP}). A set of targets creates the trajectory of the SM. If $k = 12$, then the optimization has been done without considering the proposed method; $k = 8$ (for rotary table) and $k = 6$ (for linear guide) implies that the optimization has been done with respect to the identified parameters. The constraints of the optimization are the joint limits of SM ($\mathbf{q}_i$) which have to be within a certain range ($\mathbf{q}_{i_{\min}}$ and $\mathbf{q}_{i_{\max}}$). Moreover, the absolute value of the determinant of the kinematic Jacobian matrix ($\mathbf{J}$) is being limited to be bigger than a certain value ε. Because the smaller $|\det(\mathbf{J})|$ is, the closer to a singularity configuration is the robot. Moreover, the control system of industrial robots has a threshold for the proximity to the singularity, in which it turns off the motors while doing a Cartesian movement near a singularity.

3.3 Classical approach to dependency identification

The problem of identifying number of independent arguments involved in a given function can be tackled by differentiating the function with respect to all the arguments and obtain the Jacobian matrix.

Assume f from Eq. (3.4) to be the function for which we are interested in determining the number of independent arguments. After simplifying the expression and then differentiating it, one has:

$$f = f(\mathbf{q}, \mathbf{t}_{k\times 1}, q_{RP}), \tag{3.5}$$

$$df = [\frac{\partial f}{\partial t_1} \dots \frac{\partial f}{\partial t_k}] \cdot [dt_1 \dots dt_k]^T, \tag{3.6}$$

$$df = \mathbf{J} \cdot d\mathbf{t}. \tag{3.7}$$

Note that we only differentiate f with respect to $\mathbf{t_k}$ and not $\mathbf{q}$ nor q_{RP}, because only $\mathbf{t_k}$ is a design parameter of the placement, i.e., the decision variables of the optimization, and the actuation of the RP is not an argument for which we want to find out if it is dependent or not. In the case that the placement of the path in the workspace of the robot with no RP is taken into the consideration, there are 6 decision variables, namely the placement of the path in the workspace of the SM, but not all of them are effective on f.

In Eq. (3.5)-(3.7), only if f is differentiable then this problem is analytically solvable. In our case, the objective function is discrete due to the discrete nature of the path. Apart from that, the absolute value in the objective function makes it non-differentiable. However, it is possible to break down the absolute value to positive and negative domains of the function. Obtaining the derivative of such a complex function can be a very cumbersome task and in some cases, it is not possible to find an analytic solution. Assuming that $\mathbf{J}$ is available, one can evaluate the number of effective arguments by knowing the number of linearly independent columns.

This classic approach works well before we introduce the RP to the workcell. But with the RP, the number of design variables (the decision arguments in f) is doubled, but one can instantly see that the number of linearly independent columns in $\mathbf{J}$ will stay the same as the case before. This is because the transformation between A_{xyz} and A'_{xyz} ($\mathbf{T}_A^{A'}$) appears as a linear transformation among the rest of the transformations ($\mathbf{T}_{A'}^{O}$, $\mathbf{T}_B^A$). Therefore, the linear dependency of the columns stays the same. Even if we consider q_{RP} as a decision variable, we have to consider a different q_{RP} for each point in the path since it can change regardless of the differentiation (which is a small change in the variables resulting in a small change in the f).

As a simple analogy, assume $h = h(v + [d])$; h is a function of a real variable v and an interval variable $[d]$, see Fig. 3.4. This function is not differentiable since there is an interval variable involved. The presence of an RP have the same impact on the objective function. With the RP, the actuator of the RP (q_{RP}) behaves as an interval variable. It provides a range for the placement of the path. Therefore, it makes it impossible to find the number of independent variables by the classic approach. In the upcoming section, an innovative method is introduced to identify the number of independent parameters.

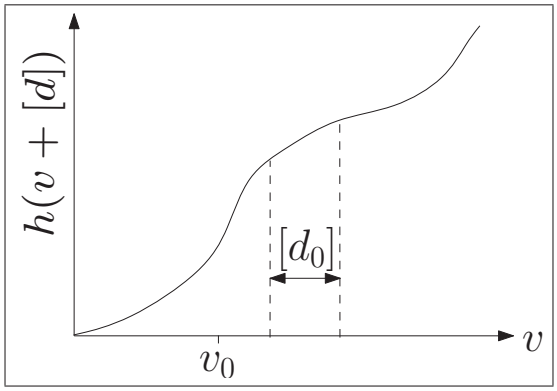


Figure 3.4 A function of a real variable and an interval. Because the output if the function is an interval, it is not differentiable.

3.4 The innovative parameter identification method

This section describes the innovative method of identifying the number of independent parameters involved in the placement optimization for both RP cases, i.e., the rotary table and linear guide. First, for each case the method is discussed, and then the general method is concluded from the case studies.

Two approaches can be applied to this problem; (1) considering all possibilities and eliminate those which are dependent and (2) starting from scratch and add up those parameters which are independent. The beginning of this section, i.e., the study case by case, is more focused on the 1st approach and the later part, general method, is more based on the 2nd approach.

Definition 1. *A parameter is considered to be "independent" if it makes a brand new trajectory from the SM's point of view, and that trajectory cannot be made by any other parameters or combinations of parameters.*

Since all the optimization goals are set to improve the SM's operations, only those parameters which make a difference for the SM are considered to be the independent parameters. As shown in Fig. 3.2 and 3.3, a generic industrial SM with 6 DOF, has 6 actuated joints (q_i, $i = 1,\ldots,6$). We assume that q_1 provides a uniform polar workspace for the SM, because q_1 has a range of almost 360 degrees. Therefore, two parameters of frame A'_{xyz} are dependent, namely x and y. Hence, one can define the distance between the origin of the base frame of the RP to the SM by only one parameter. This is a common factor between both cases. In the upcoming sections more factors which cause dependencies are studied. Note that in this chapter we do not consider the joint limit of q_1.

3.4.1 Swept volume

Before representing the methodology, it is necessary to introduce the concept of *Swept volume*. The Swept volume is a hyper-volume in the $(n+1)$D space generated by sweeping (translation and/or rotation) of an nD path. For example, a point is a 0D object (path). Its Swept volume is

a line, which is a 1D object. As another example, a disk is a 2D object. The Swept volume of a disk can be a cylinder, which is a 3D object. Therefore, the Swept volume of a 6D path is a 7D hyper-volume. The 7th dimension is provided by the RP.

In other words, the Swept volume is an extra placement option provided by the RP, but the difference between this and the fixed placement options (placement of frame A' in frame O and frame B in frame A) is that once the fixed placements are done and the coordinated system is ready to operate, non of the fixed placements can change but RP can move the path and make a new placement, simultaneously with the operation of the SM. This is the interval effect of the RP.

The shape of the Swept volume depends on the type of sweep and the original path. An example of the Swept volume of a 6D ski goggles path generated by the rotary table is shown in Fig. 3.5 and by the linear guide in Fig. 3.6. Note that each member of the ski goggles path has an orientation with respect to the path reference frame B_{xyz} as well as its position, but since it is not possible to depict a 7D path or even 6D path, we can only show a cross-sectional view of the Swept volume. In Fig. 3.5 and Fig. 3.6 the orientation of the path points is neglected. As it can be observed, the original path is a spacial curve in 3D space and it turns into a surface, which is 1 dimension higher than a curve.

3.4.2 Rotary table

Considering the coordinated robotic workcell with a rotary table, there are two cases possible: the rotary table with (1) unlimited rotation, and (2) limited rotation capability, e.g. swivel table.

3.4.2.1 Unlimited rotation

All 12 candidate parameters are represented in Table 3.1, where the Dependency Group (DG) shows which parameters are dependent to each other. Those parameters sharing the same DG are dependent and, as can be observed, there are 8 independent parameters.

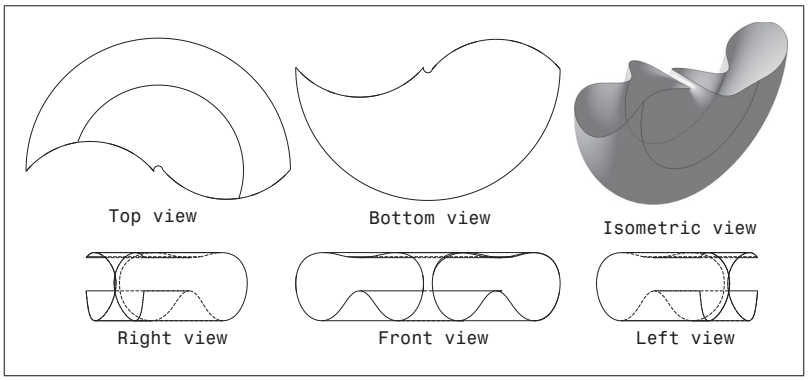


Figure 3.5 Wireframe draft of the Swept volume of a 6D ski goggles path generated by the rotary table. Note that it is not possible to depict a 7D path, so one can see only 3 dimensions of the result. This is a cross-section representation which neglects the orientation of each point of the path. Therefore, the path is degraded to a 1D curve embedded in 3D space and the resultant swept volume shows a 2nd dimension provided by the RP which results in a surface embedded in 3D space.

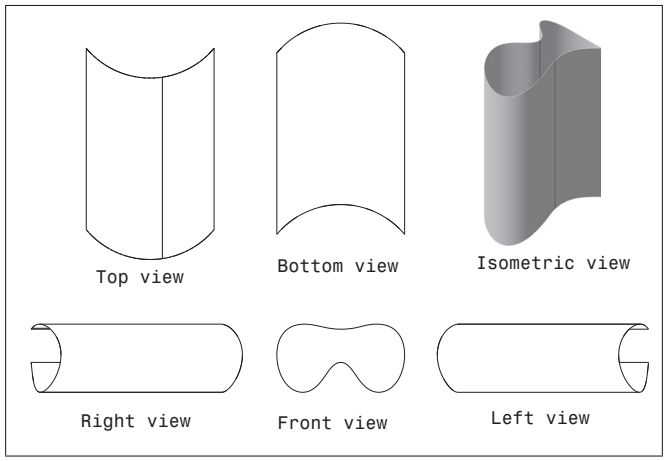


Figure 3.6 Wireframe draft of the Swept volume of a 6D ski goggles path generated by the linear guide. Note that it is not possible to depict a 7D path, therefore one can see only 3D of the result.

The reasoning for each group in Table 3.1 is as follows:

1. The first joint of the SM, q_1, provides a uniform polar workspace. Therefore, only the radial distance between the RP's base frame and O_{xyz} makes a difference for the SM. The

combination of $x_{A'_{xyz}}$ and $y_{A'_{xyz}}$ determines the radial distance of the RP to the SM, so they are dependent.

2. As it can be observed from Fig. 3.2, the direction of $z_{A'_{xyz}}$ is the same as $z_{A_{xyz}}$. Therefore, the combination of these two parameters is representing one independent parameter. Note that the pose of B_{xyz} is represented with respect to A_{xyz}, so $z_{B_{xyz}}$ is a value along $\overrightarrow{z}_{A_{xyz}}$. Hence, the orientation of B_{xyz} will not deny the aforementioned fact.

3. The moving frame of the rotary table rotates around its z axis and provides a uniform radial workspace, the same reasoning as DG 1 is applicable here: the placement of the reference frame of the path (B_{xyz}) with respect to the moving frame of RP (A_{xyz}) only depends on the radial distance of B_{xyz}, and therefore, $x_{B_{xyz}}$ and $y_{B_{xyz}}$ are dependent.

4. The rest of the DGs (4,. . . ,8) are all independent parameters. That is because each of them causes a new Swept volume and there is no symmetricity to tie two or more parameters together.

Table 3.1 Parameter identification for the rotary table. DG refers to dependency group. Those parameters sharing the same DG are dependent to each other.

RP's base (A')	DG	Path's reference (B)	DG
$x_{A'_{xyz}}$	1	$x_{B_{xyz}}$	3
$y_{A'_{xyz}}$	1	$y_{B_{xyz}}$	3
$z_{A'_{xyz}}$	2	$z_{B_{xyz}}$	2
$\alpha_{A'_{xyz}}$	4	$\alpha_{B_{xyz}}$	6
$\beta_{A'_{xyz}}$	5	$\beta_{B_{xyz}}$	7
$\gamma_{A'_{xyz}}$	0	$\gamma_{B_{xyz}}$	8

DG 0 means that the orientation around the z axis of A'_{xyz} does not count because it is the same as variation of q_r, i.e., rotation of the rotary table's actuator. Therefore, from the initial 12 candidates, 8 parameters are independent. It is up to the user to choose one of the candidates from each DG. For example, in DG 1, either $x_{A'_{xyz}}$ or $y_{A'_{xyz}}$ can be chosen to be an optimization variable.

3.4.2.2 Limited rotation

In a case where the rotation of the rotary table is limited (swivel table), $\gamma_{A'_{xyz}}$ should be considered as an optimization variable. Since q_r is limited, the orientation around z axis of A'_{xyz} changes the resultant path for the SM. So in this case 9 parameters are important out of 12. It is noteworthy that actually the 9th parameter can be chosen from group DG 1. It means that the rotary table can be placed behind the SM while maintaining the same orientation around the z axis. However, the result would be the same.

3.4.3 Linear guide

In the case of the coordinated robotic workcell with the linear guide as the RP, we consider two cases of the linear guide: with (1) unlimited, and (2) limited actuator length. It is not a realistic assumption to consider that the linear guide is capable of infinite actuation, but for the sake of respecting the duality between the rotary table and the linear guide, both unlimited and limited cases are studied.

3.4.3.1 Unlimited translation

All 12 candidate parameters are represented in Table 3.2. In Table 3.2, the Dependency Group (DG) shows which parameters are dependent to each other. Those parameters sharing the same DG are dependent and as can be observed, there are 6 independent parameters.

The reasoning for each group in Table 3.2 is as follows:

1. As in the case of the rotary table, the first joint of the SM, q_1, provides a uniform polar workspace. Therefore, only the radial distance between the RP's base frame and O_{xyz} makes a difference for the SM. But in this case, the combination of $x_{A'_{xyz}}$, $y_{A'_{xyz}}$, $\gamma_{A'_{xyz}}$ and $y_{B_{xyz}}$ all together determines this radial distance, so they are dependent.

   1^* indicates that $y_{A'_{xyz}}$ and $y_{B_{xyz}}$ are dependent. Because the actuation of the linear guide provides a pure translational movement, the combination of the aforementioned parameters

represents the distance from the origin of B_{xyz} to q_1 axis. Furthermore, since $y_{A'_{xyz}}$ is tied up together with $x_{A'_{xyz}}$ representing the radial distance to q_1, 1 and 1^* are dependent. In the case of infinite motion of the linear guide, the geometric problem can be modeled as a line to point distance on a plane. The rotation around the z axis does not change the distance, so $\gamma_{A'_{xyz}}$ also belongs to the group of DG 1.

2. As can be observed from Fig. 3.3, the direction of $z_{A'_{xyz}}$ is the same as z_{A_xyz}, so the combination of these two parameters represents one independent parameter. Note that the pose of B_{xyz} is represented with respect to A_{xyz}, and therefore $z_{B_{xyz}}$ is a value along $\overrightarrow{z}_{A_xyz}$. Hence, the orientation of B_{xyz} will not deny the aforementioned fact.

3. Variation in $\alpha_{B_{xyz}}$ does not change the shape of the Swept volume. But the orientation around $x_{A'_{xyz}}$ will change the pose of the Swept volume from the SM's point of view. The user can choose either of $\alpha_{A'_{xyz}}$ and $\alpha_{B_{xyz}}$ as an optimization variable.

4. The rest of the DGs (4,...,6) are all independent parameters because they make variations in the final path seen by SM and there is no symmetricity to tie two or more parameter together.

DG 0 means that the variation in $x_{B_{xyz}}$, represented in A_{xyz}, does not count because it is the same as variation of q_l. Therefore, out of the initial 12 candidates, 6 parameters are independent. The user can choose between the dependent parameters. For example, in DG 1, either $x_{A'_{xyz}}$, $y_{A'_{xyz}}$, $\gamma_{A'_{xyz}}$ or $y_{B_{xyz}}$ can be chosen to be an optimization variable.

3.4.3.2 Limited translation

In the event that the linear guide can only provide a limited range of motion, which is the realistic case, one more parameter will be added to the previous 6. It can be chosen from the group DG 1 or it can be the DG 0. Hence, 7 parameters are needed.

Table 3.2 Parameter identification for the linear guide.

RP's base	DG	Path's base	DG
$x_{A'_{xyz}}$	1*	$x_{B_{xyz}}$	1*
$y_{A'_{xyz}}$	1	$y_{B_{xyz}}$	0
$z_{A'_{xyz}}$	2	$z_{B_{xyz}}$	2
$\alpha_{A'_{xyz}}$	4	$\alpha_{B_{xyz}}$	5
$\beta_{A'_{xyz}}$	3	$\beta_{B_{xyz}}$	3
$\gamma_{A'_{xyz}}$	1	$\gamma_{B_{xyz}}$	6

3.4.4 General method

After performing the case studies and investigating the parameter identification of the placement optimization problem, a general approach can be deduced which may be extended to more complicated coordinated robotic workcells in future works. The formulation of the method is represented as follows:

$$n = d - s - r + w \tag{3.8}$$

$$n_{\text{rotary table}} : 8 = 6 - 1 - 1 + 4 \tag{3.9}$$

$$n_{\text{linear guide}} : 6 = 6 - 1 - 1 + 2 \tag{3.10}$$

- n: the number of independent parameters.
- d: dimensions of the workspace. A generic path in a 6D space can be placed by defining 6 parameters. For a planar coordinated robotic system $d = 3$.
- s: the number of parameters which are dependent because of the symmetricity of the SM's workspace. In the case of a generic SM, 1 degree of symmetricity is provided by q_1.
- r: the number of RP's DOF, i.e., the number of the RP actuators. In this chapter we investigated 1 DOF RPs, therefore $r = 1$.

- w: the number of parameters out of 6 (6 being the total number of parameters involved in the placement of the path on the RP), that their change will make a different shape of the Swept volume.

It is worth mentioning that, if for example the RP has 2 DOF, $r = 2$. But this does not mean that the higher the DOF, the lower the flexibility of the workcell, because the higher RP's DOF leads to fewer placement parameters and more freedom during the optimization of the trajectory planning of the coordinated workcell.

As shown in Eq. (3.9), applying the general approach to the rotary table gives us the already known 8 parameters, $8 = 6-1-1+4$. Here, 6 is for the original freedom of placement, -1 is because of q_1, and the second -1 is because of the rotation axis of RP. Finally, $+4$ adds those parameters related to the placement of the path on the rotary table which create new Swept volumes. One of them is for the distance to the center of the rotation of the rotary table and three of them for the three orientations of B_{xyz} with respect to A_{xyz}.

Similarly for the linear guide, $+6-1-1+2=6$. -1 is because of q_1, -1 because of the linear guide actuation, and $+2$ is for the placement of the path on the linear guide orientation in such a way that it creates new Swept volume shapes, which is the rotation of the path around $\overrightarrow{y_{A_{xyz}}}$ and $\overrightarrow{z_{A_{xyz}}}$. In this case, the displacement of the path on the linear guide (displacement along axes x, y and z) and rotation along the actuation axis of the linear guide ($\overrightarrow{x_{A_{xyz}}}$), will not change the Swept volume shape and it would be exactly as shown in Fig. 3.6.

The number of independent parameters is obtained, but which one of them is to be chosen depends on the specific application. For the most applications, it does not matter which parameter from a DG is considered for the optimization (the case of this chapter). In some cases there are some limitations along the way. For example painting with an open container in which the nature of the path dictates some restrictions due to the direction of gravity, or for example when the linear guide is bound to be horizontal due to design restrictions.

3.5 Result comparison

In this section, the proposed method is applied to optimize the objective function represented in Eq. (3.4) in order to test the effectiveness, before and after applying the identified number of parameters. Then the results of the optimization both with and without the effect of the identified parameters are compared. In the last subsection we investigate which RP performs better in the case of ski goggles gluing optimization.

3.5.1 PSO method

The main objective of this chapter is on the innovative method to find the number of independent parameters, and the optimization algorithm is out of the scope of this chapter. The optimization algorithm is only summarized in what follow, and a detailed explanation is represented in the previous chapter. The algorithm includes two parts. The first part is based on an evolutionary algorithm, namely PSO, to find the optimum placement for the RP and the path ($\mathbf{t}_k$). The second part is based on a traditional NR (Rao (2009)) to find the optimized solution for the RP kinematics. In other words, the PSO suggests a placement (some values for $\mathbf{t}_k$), and the NR finds the best q_{RP} for the suggested placement. Then for the next iteration, the PSO suggests another placement, which is deduced based on the current and past personal memory and group memory of all the particles in the PSO population. For any given iteration of the PSO, the NR finds the best q_{RP} for the obtained placement. The iterative process leads to finding the optimized placement and optimized q_{RP} (trajectory planning) at the same time. Finally, the optimized placement is fixed, the optimized trajectory planning for the RP is obtained, and the optimized trajectory planning for the SM is straightforward to calculate.

3.5.2 Results comparison between the identified parameters and all parameters

In this section, the results in terms of accuracy and speed of the convergence to the optimized placement is compared between two conditions: (a) $k = 12$, and (b) $k = 8$. The results are represented for the ski goggles gluing. As can be observed from Fig. 3.7, the placement op-

timization converges to the final solution in 26 iterations when using the identified number of parameters ($k = 8$), but it takes 40 iterations to obtain the same result in the case of using all the candidates ($k = 12$). Therefore, there is a 32% improvement in the optimization process by only choosing the necessary optimization variables. Note that the objective function (cycle time) is the summation of all the SM's joint movements to go through the path, point by point.

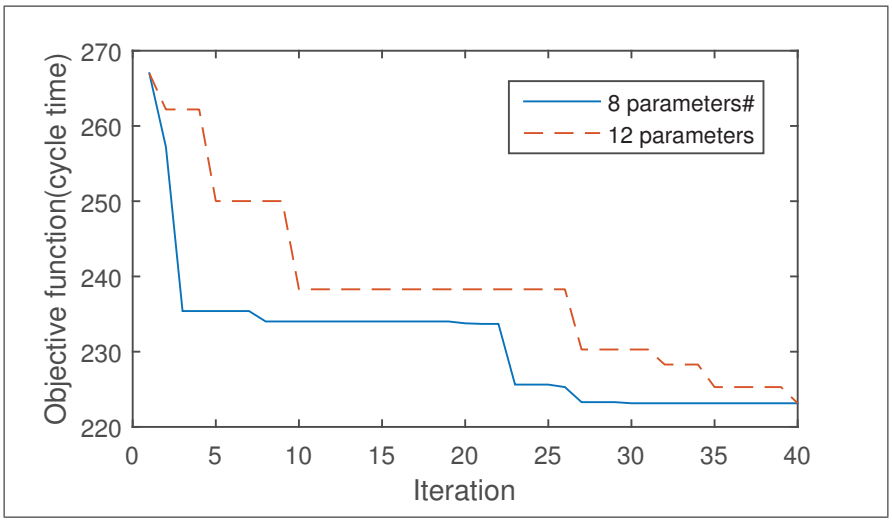


Figure 3.7 Comparison of the optimization results using the rotary table as the RP, when all 12 candidate parameters are taken into account vs using only 8 independent parameters. An improvement of 32% is achieved.

3.5.3 Results comparison between rotary table and linear guide

This section compares the results of the performance of the two RPs at gluing the ski goggles. As can be observed from Fig. 3.8, the rotary table performs about 70% better than the linear guide. The rotary table's superiority was expected because of the shape of the path. The ski goggles has a rounded path, and the rotary table enables the SM to reach over all the points in the path. If the given path was a lengthy object, such as a solar panel, the linear guide would have performed better.

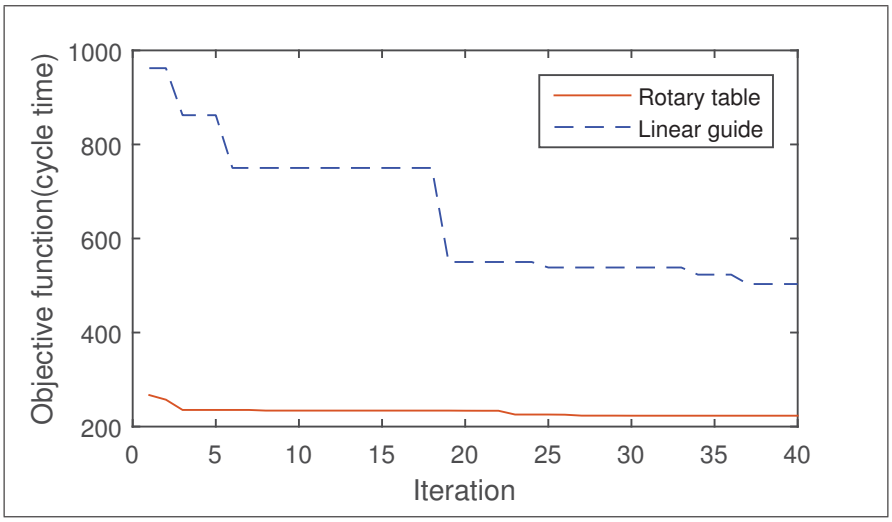


Figure 3.8 Comparison of the optimization results between the rotary table and the linear guide. The rotary table performs about 70% better than the linear guide.

3.5.4 Discussion

After comparing the results between different RPs and different number of parameters, it is worth providing the results for the case with no RP, i.e., the robotic system only consists of a SM. The results represented in this section helps the partitioner to make a decision based on value per cost of adding an RP. However, the problem of the optimized placement still exists, i.e., the path placement. Therefore, an optimization algorithm has been applied to the problem and the result is represented. For this optimization, the reference frame of the path can be placed by defining 6 parameters, namely the 6 DOF in the Cartesian space. The algorithm is the same as the main algorithm, with less complexity. Even though the number of involved parameters is 6 instead of 12, still the parameters identification is needed. In this optimization, in order to avoid complexities, no parameter identification has been done and all the 6 parameters have been optimized. As represented in Fig. 3.9 when no RP is used, after 40 iterations, the optimum result is obtained. As it can be observed, the obtained cycle time is close to what has been already achieved using a linear guide as the RP. The final result of the objective function with no RP converges to 590, versus 510 for the case with a linear guide. It is up to the practi-

tioner to decide weather to use an RP or not. In our case, using a rotary table does make sense regarding the cost of adding one, however, it is not financially justified to use a linear guide.

The best obtained placement configuration of the coordinated robotic system, which corresponds to the rotary table, is as shown in FIg. 3.10 and the corresponding transformation matrix is represented in Eq. (3.11).

$$A'_{\text{Rotary table}} = \begin{bmatrix} -0.028970 & 0.842236 & 0.538330 & 979.940000 \\ 0.998889 & 0.044417 & -0.015737 & 241.860000 \\ -0.037165 & 0.537276 & -0.842587 & 346.990000 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 \end{bmatrix}. \tag{3.11}$$

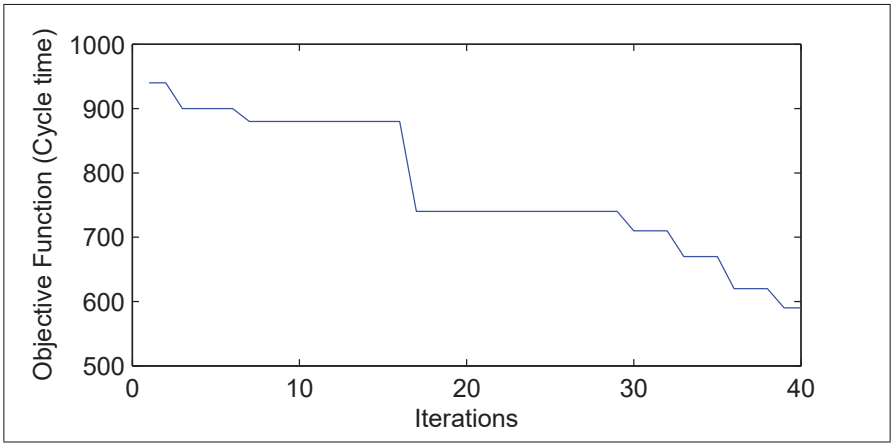


Figure 3.9 Convergence of the ski goggles gluing, when no RP is used.

As it can be observed in Fig. 3.10, the RP is placed in a relatively far distance from the SM. This decreases the cycle time. Since the lever arm is greater, the displacement at the point of application is as small as possible. However, it demands more torque from the proximal joints.

3.6 Conclusion

In this chapter, an innovative method of identifying the number of independent parameters was proposed in order to optimize the placement of a given path in a coordinated redundant

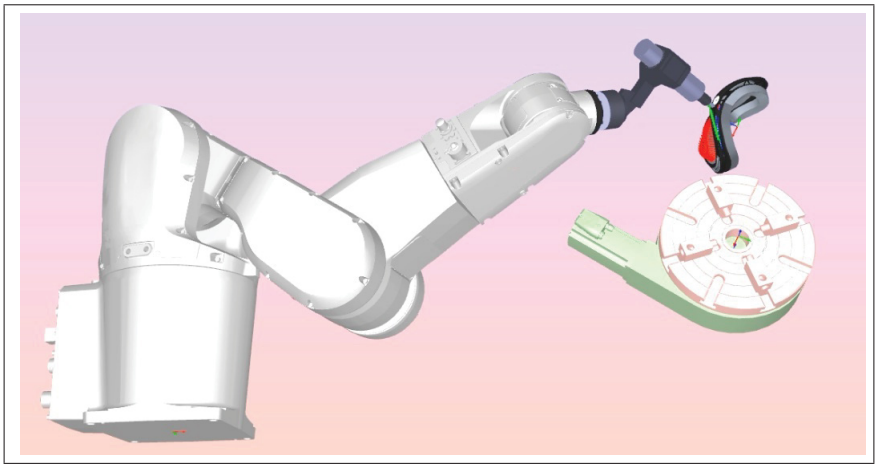


Figure 3.10 The final configuration of the coordinated robotic system to glue the ski goggles, using a rotary table as the RP.

robotic workcell. Results showed that there are 6 and 8 parameters needed, respectively, to place a generic path on a rotary table and linear guide. In the case where one takes into the account the joint limit of the aforementioned redundancy providers, 1 more parameter will affect the placement optimization. Moreover, the results indicate that because of the nature of the problem of ski goggles gluing, a rotary table performs better than a linear guide as a redundancy provider in the coordinated workcell. In future works, expansion to higher DOF redundancy providers is recommended.

CHAPTER 4

SIMULTANEOUS TASK PLACEMENT AND SEQUENCE OPTIMIZATION IN AN INSPECTION ROBOTIC CELL

4.1 Introduction

In today's industrial robotic cells, efficiency is a key factor to mass production. Less process cycle time leads to more production and more business value. Two common automation problems that need optimization are task sequencing and task placement.

Even though many researchers have been interested in optimizing the work flow of a multi-task workcell Alatartsev *et al.* (2015); Kovács (2016); Kolakowska *et al.* (2014); Vicencio *et al.* (2014), there is still no method that is able to efficiently optimize task sequence and consider all degrees of freedom (DOF) and constraints, e.g., workcell layout, collision-free trajectory planning, multiple inverse kinematics solutions (robot configurations).

The goal of this study is to minimize the process time of a turbine blade inspection workcell. The process includes the following tasks: a polished turbine blade is gripped by a serial robot and shown to a stationary camera with different poses. This chapter proposes a novel algorithm to find the optimized placement of the camera and the best sequence of the images, simultaneously. The objective is to minimize the inspection time, while avoiding collisions Kaloorazi *et al.* (2013, 2015). This method will combine the continuous camera placement and combinatorial image sequencing problems and solve them together.

Camera placement in a workcell is a redundancy resolution problem FarzanehKaloorazi *et al.* (2018a). Similar to part or path placement, in camera placement, six Degrees Of Redundancy (DOR) are introduced to the problem. Infinite solutions can be found to a redundancy resolution problem and various approaches are introduced in the literature.

For example, in Caro *et al.* (2013), optimum placement of the workpiece to be machined is investigated, considering the forces applied to the robot. In Ur-Rehman *et al.* (2010), the

actuator torques, energy consumption, and shaking force are minimized for a multi-objective path placement optimization involving parallel mechanisms. Finally, simultaneous placement optimization and trajectory planning was performed in FarzanehKaloorazi *et al.* (2018b). In that publication, a serial robot and a rotary stage (1 DOF redundancy) work together in an automated fiber placement cell. A meta-heuristic method was proposed based on PSO (particle swarm optimization) to find the best placement of the rotary stage, while optimizing the angle value of the rotary stage.

Sequence optimization is also a common problem in robotics and has been addressed in the literature. In Zhang & Li (2009); Zacharia & Aspragathos (2004), using GA (genetic algorithm), the sequence of visiting tasks and robot configurations are optimized. However, the task placement was not considered. In Baizid *et al.* (2010), a GA-based method is developed to optimize the task point visit order, considering the robot configuration and the workcell layout. However in Baizid *et al.* (2010), the robot placement is limited to position and to only four points. In Baizid *et al.* (2014), a similar problem is address but this time by adding discrete limited orientation placement options for the robot. The issue with GA is that it is naturally limited to discrete search spaces. In these studies a high number of population size and iterations were required. For example in Baizid *et al.* (2010), the generation number is 500,000 and the iteration number is 222,857. Special modifications are needed in order to optimize a continues problem by GA. In Tubaileh *et al.* (2007), the task sequence and feasible robot configurations are obtained by using sequential quadratic programming, but not the task placement.

In Zacharia & Aspragathos (2005), the traveling salesman problem (TSP) is adapted to robotics. Travel time between any two poses is considered to be affected by the selection of the robot configuration. Again, the method is based on GA and an encoding technique is introduced to take into account all solutions of the inverse kinematic problem (IKP). In Aneja & Kamoun (1999), two decisions that need to be made simultaneously are investigated: finding the robot move cycle, and finding the input part sequence. The objective is to maximize the throughput rate of the cell. The problem is formulated as a special kind of TSP. An O(n4) time algorithm which solves this problem optimally has been provided in Hall *et al.* (1997) and the authors ex-

tend that work and provide an algorithm of complexity O(n log n). Finally, in Kovács (2016), the goal is to find the appropriate order of welding tasks where the robot path is considered during sequencing. For modeling the problem, an extension of the TSP with neighborhoods, denoted as TSP-ND, is introduced. A GRASP meta-heuristic algorithm is proposed for solving it.

In this chapter, a novel method based on PSO is proposed to simultaneously optimize the continuous problem of camera placement and the combinatorial problem of image sequence optimization. PSO is a meta-heuristic optimization method, proven to obtain the optimal result in various engineering problems, without requiring the function to be derivable Kennedy (2011). In PSO, a population of particles search for the optimum value of the function and inform each other about the best results they have obtained. For an individual particle, personal and global (swarm) best memory, as well as the inertia from the previous movement, defines the current velocity of the particle.

In the proposed method, each particle conveys six real values (camera placement in 6-DOF space) and a set of integers (sequence of the images). Therefore, the problem in hand is solved for the location of the camera while the best image sequence is obtained. Even though PSO is widely used by researchers to solve continuous problems, it is less utilized for combinatorial cases. The proposed method is specifically new in term of combinatorial engineering problems.

The remainder of this chapter is as follows. Firstly, the problem is explained in detail and the challenges are investigated. Then, the proposed methodology is discussed. Pseudo-code for the proposed algorithm is represented and explained. After that, two case studies are presented, which examine the efficiency of the algorithm. One is a pure combinatorial problem and the other is on the turbine blade inspection problem at hand. At the end, the results are discussed.

4.2 Problem description

In an automated robotic workcell and specifically in an inspection application, the placement of the elements involved is highly important. Imagine a turbine blade finishing setup in which the

objective of the robot is to grab a machined blade, surface finish it on a grinding belt and then inspect it (Fig. 4.1). Each working cycle of the robot consists of two phases. In the first phase, the robot takes the blade from a conveyor and brings it to the grinding belt. In the second phase, the polished blade is to be inspected by a stationary camera. The inspection process consists of taking multiple images of the polished blade. In this section, the problem is described and the proposed solutions are postponed to the next section.

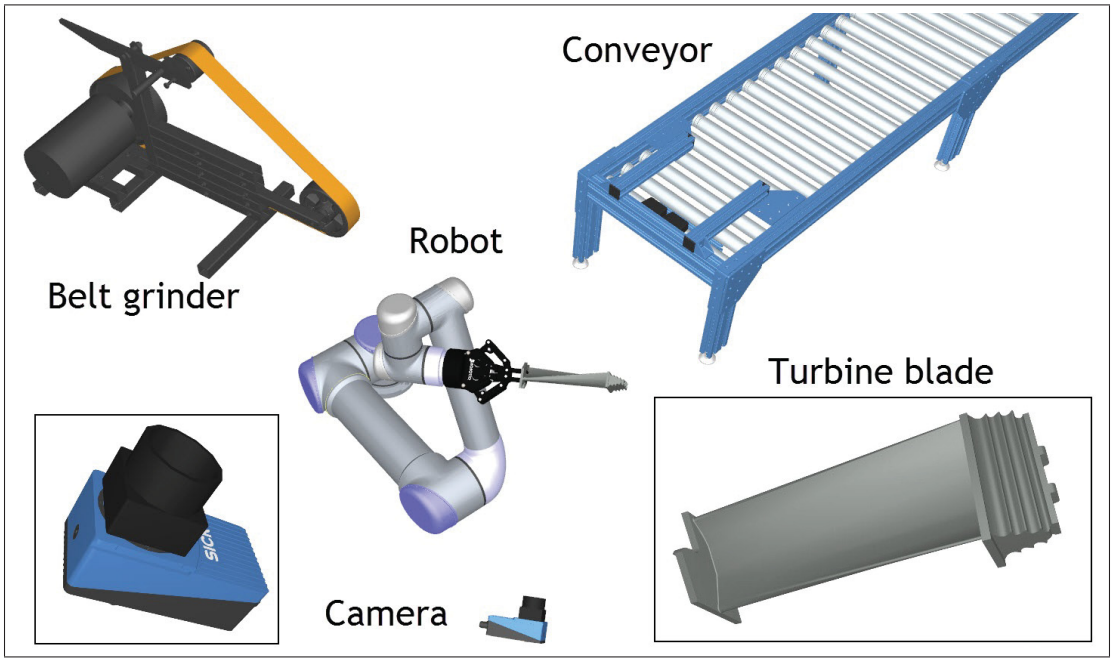


Figure 4.1 The robotic workcell to polish and inspect turbine blades. The robot grabs the blade from the conveyor, brings it to belt grinder, then brings it to the camera.

The elements of the workcell, i.e., the robot, the conveyor, the polishing machine and the camera, are to be placed with respect to the world frame. Since the polishing machine, as well as the robot, are typically bulky and fixed to the workcell, they cannot be easily moved. The only device that is relatively easy to displace is the camera. Therefore, a proper placement of the camera is needed to be able to take all the required images of the blade in a time efficient manner. These images actually correspond to a set of predefined poses of the blade with respect to the camera. All image poses are fixed with respect to the camera frame.

Definition 1. *A feasible placement of the camera is a placement in which all the required images can be taken.*

If for one or more required images, the inverse kinematics of the robot has no solution, the corresponding placement of the camera is deemed unfeasible.

4.2.1 Optimization criteria

Regarding the feasibility of the camera placement, there may exist more than one solution. However, it is typically very hard to find any feasible solution for this camera placement problem, as solutions can be found only in a small region of the robot workspace. In this chapter, we try to find the best solution among all the feasible camera placements. To find the best placement, various optimization objectives can be considered, such as energy consumption Field & Stepanenko (1996); Hirakawa & Kawamura (1997), cycle time Chan & Zalzala (1993); Pateloup *et al.* (2004), minimum traveled distance Tian & Collins (2003), etc. Here, the following optimization objectives are to be achieved by the proposed algorithm: find the best placement of the camera in the workcell for which (1) the cycle time is minimized, (2) there exist no collisions, and (3) the placement is feasible for all images. In what follows, these criteria are explored.

4.2.1.1 Collision avoidance

The collision-free placement of the camera is essential. Assuming that the placement of the robot and the belt grinder is already free of collisions, the camera needs to be located in such a way that it does not collide with neither the robot, nor the conveyor, nor the belt grinder. Furthermore, the camera has to be out of the robot path followed in the first phase (i.e., from conveyor to grinder).

Collision detection methods have been investigated for decades and several efficient methods are being used extensively FarzanehKaloorazi *et al.* (2017, 2014). Among them, 3D collision detection is very popular in robotics applications. They can be categorized into four groups:

space–time volume intersection, swept volume interference, multiple interference detection and trajectory parameterization Jiménez *et al.* (2001). In this chapter, an open source collision detection for both still and moving CAD models is used, called Bullet. The results then are simulated in RoboDK and collision detection is doubled checked.

4.2.1.2 Cycle time

Since the placement of the camera does not affect the first phase of the automated process, only the time for inspecting the blade is considered and is simply referred to as the cycle time throughout this chapter.

The exact cycle time of the robot depends on numerous factors. The actual cycle time of the robot can be quite difficult to obtain as it depends on various parameters set by the user (e.g., maximum acceleration) and on the robot controller. In terms of an iterative optimization, it is not feasible to run the real robot to get the actual cycle time (or even to run the robot manufacturer's simulation software). Therefore, we need to find the best approximation for calculating the cycle time.

4.2.2 Sequence of images

For any camera placement, the order of the images to be taken has a direct impact on cycle time. To find the optimal sequence, the well-known Traveling Salesman Problem (TSP) has to be solved Lawler (1985). In TSP, the objective is to find the shortest path to visit each of several cities. In our inspection problem, the robot needs to move between each two inspection poses.

In a classical TSP the distance between each pair of given cities is known in advance. Usually a $C \times C$ chart with $(C^2 - C)/2$ unique entries for the distance between each pair is given, where C is the number of cities. Moreover, the distance between two cities is constant and is not subject to change. There are three major differences between the classical TSP and the problem at hand. This fact demands certain strategy changes.

Firstly, it is not guaranteed that, if the best sequence is obtained for a certain camera placement, this same sequence will serve the best result for another placement. Secondly, unlike the classical TSP, the distance chart is not given in advance. This chart needs to be calculated for each given placement of the camera. It would be extremely computationally expensive to obtain all the distances and then solve the TSP for each camera placement in an iterative solution. Finally, for each inspection pose, there are up to eight possible robot configurations to be considered (i.e., the salesman needs to visit only one of several cities in each area).

4.2.3 Degrees of Placement

The number of independent parameters describing the placement of the camera are known to be the degree of placement (DOP), also known as degree of redundancy. In general, the camera placement has 6 DOP, i.e., 3 translational and 3 rotational DOP along and around x-, y- and z-axis, respectively. Depending on the conditions of the workcell and the vision system, the DOP can be reduced. For instance, rotation around the z-axis of the camera is equivalent to rotating a taken image in the post-processing, if the lighting conditions are uniform. Moreover, in some cases due to the symmetry of the robot's workspace, some of the DOP fall into a dependency group. For example, since the first joint of the robot is generally revolute and its axis parallel to the robot's base z-axis, the following three parameters are dependent: displacement along x- and y- axes and rotation around the z-axis. Only two of them can independently affect the cycle time. For a detailed study on parameter dependency, refer to FarzanehKaloorazi *et al.* (2018a).

4.3 Methodology

The problem of inspection workcell optimization can be divided into two subproblems: (1) finding the best sequence of the images and (2) finding the best placement for the camera. On one hand, the first subproblem is a combinatorial problem Lawler (1985), in which the objective is to find the optimal ordered set of combinations, among a finite number of combinations, which minimizes the cycle time. On the other hand, the second subproblem is a continuous optimization problem Storn & Price (1997), in which the objective is to find the best location of

the camera in a 6D continuous space, among infinite possibilities. Even though the objectives of both subproblems are the same, they cannot be easily combined, because of their different search spaces. This chapter is proposing a novel approach based on a modified PSO to solve the TSP, thus solving the two subproblems simultaneously.

In what follows, first the generic cost function, constraints and parameters discussed in Sec. 4.2.1 are brought back to perspective, specifically structured, and represented for the problem at hand. Then, the proposed modified methodology using PSO is described to combine the two subproblems. Finally, the pseudo-code of the proposed algorithm is represented.

4.3.1 Practical problem structure

In this section, the optimization objective (cycle time) and the optimization constraint (collision avoidance) are structured for the camera placement subproblem. Moreover, the optimization parameters for the combinatorial subproblem (sequence of images) and parameters for the continuous subproblem (camera placement) are investigated.

4.3.1.1 Objective function (cycle time)

As mentioned earlier, for an efficient iterative optimization one needs to calculate an estimate for the cycle time. In this chapter, the Weighted Joint Travel Time (WJTT) estimation will be used Caro *et al.* (2013). The weight factor for each joint depends mostly on its maximum velocity, especially when relatively large joint motions are involved and maximum joint accelerations are used. Ideally, however, the weight factor must be adjusted experimentally.

In today's commercially-available industrial robots, the (generally) fastest and simplest way to move the robot end-effector from one pose to another, is to use a joint-mode motion command. In joint mode, all joints move simultaneously, following linear trajectories in the joint space. Thus, in general, only one of the joints will be moving at maximal speed and its travel time will determine the cycle time for the pose-to-pose motion. This joint is usually the slowest of

them.Thus, for a total of C images to be taken, the total cycle time that must be optimized is the sum of the $C-1$ pose-to-pose motions.

There are $C!$ sequences of images. In practice, a turbine blade inspection may require dozens of images, which means that there are millions of different sequences. Furthermore, for each image (inspection pose), there are up to eight different joint sets (robot configurations). This means that there are $8^C C!$ different sequences to verify.

4.3.1.2 DOP

As mentioned in Sec. 4.2.3, although theoretically, the DOP in a camera placement problem can be less than 6, due to practical limitations (e.g., unsymmetrical workcell, uneven lighting conditions), all the 6 DOP are taken into account.

4.3.2 Combining combinatorial and continuous optimization

In this section, the PSO is first briefly addressed and then the novelty of this work to combine the combinatorial and the continuous optimization for an inspection workcell is represented.

4.3.2.1 PSO

PSO is a population-based stochastic method, first introduced in Kennedy (1997). The classical PSO algorithm is highly capable of solving continuous problems. It over-performs the other numerical optimizers in many engineering problems Hassan *et al.* (2005). In PSO, a population of particles evaluate the search space and share their results among each other. In each iteration, considering the best personal memory and the best group memory, each particle decides on a new direction to search. An optimized result is obtained after a stopping criteria is reached. Similar to other stochastic optimizers, a global optimum is not guaranteed, but is very likely to reach. In what follows, the formulation of PSO is summarized. For more details refer to FarzanehKaloorazi *et al.* (2018b).

The PSO procedure starts with an initial population (swarm) of particles, typically randomly distributed in the search space. Each particle consists of a value set called location, i.e., optimization parameters, having s dimensions. Moreover, each particle stores a value for personal and a value for global best memory, along with the location of these memories. Each particle evaluates the fitness function with respect to its own location and saves it as its best personal memory. After memorizing its best personal location, each particle checks whether its own personal best is better than the current global best, and if so, global memory is updated and shared with other members of the population. For the next iteration, a velocity vector is calculated for each particle. The particle's location is changed due to the velocity and, if necessary, its personal or global best memory is updated. The above can be summarized as:

$$\mathbf{v}_{i,s}^{\text{new}} \leftarrow \phi_\omega \mathbf{v}_{i,s}^{\text{old}} + \phi_p r_p(\mathbf{p}_{i,s} - \mathbf{x}_{i,s}) + \phi_g r_g(\mathbf{g}_s - \mathbf{x}_{i,s}), \tag{4.1}$$

where velocity $\mathbf{v}_{i,s}^{\text{new}}$ is the location update for the next iteration, velocity $\mathbf{v}_{i,s}^{\text{old}}$ is the current velocity of the i^{th} particle for the s^{th} dimension, working as a movement inertia ($\mathbf{v}_{i,s}$ is zero for the very first iteration), $\mathbf{x}_{i,s}$ is the current location of i^{th} particle, $\mathbf{p}_{i,s}$ is the best location personal memory of the i^{th} particle, and $\mathbf{g}_s$ is the best location global memory of the whole swarm. Tunable parameters ϕ_ω, ϕ_p, and ϕ_g are the learning coefficients from previous velocity, personal memory and global memory, respectively. The tunable parameters have a value between $\{0,1\}$ and must be chosen carefully in order to achieve reliable results. Furthermore, uniform random mutation factor r_p for the personal and r_g for the global learning ($r_p, r_g = \{0,1\}$) improves the search ability of the swarm.

4.3.2.2 Modifications to fit TSP

As it has been discussed, the continuous optimization, i.e., the placement of the camera, and the combinatorial optimization, i.e., the sequence of images, need to be simultaneously resolved. Therefore, it is necessary to modify the classical PSO in order to first, be able to solve a combinatorial problem, i.e., TSP, and then to do it along side with the continuous optimization.

In this chapter a basic combinatorial PSO is used, Wang *et al.* (2003), and some features are added to improve the performance of the algorithm. The proposed combinatorial algorithm is then tested on a few TSP benchmarks to verify its ability.

The algorithm is described by the means of TSP. Assume C cities to be traveled by a salesman. The latter has to pass each city once and only once. The route is an open chain, not a loop. The start and end city do not matter. The goal is to obtain a combination of the cities which minimizes the traveled distance. Using PSO, each particle is a combination. It keeps track of its own best and global memory. The procedure starts with a randomly generated population of particles, i.e., random combinations. Each particle saves its current combination as the best personal memory, and also replaces the global best memory, if needed. In the next iteration, a series of *swaps* alters the current combination of each particle. A swap is done only between two cities in the sequence.

To choose which cities to swap, similar to Eq. (4.1), a velocity update type routine, based on learning coefficients, is executed. Learning coefficients ψ_p, ψ_g and ψ_m (personal, global and mutation respectively) are real numbers between $\{0, 1\}$. Five factors are related to the new velocity: personal and global best combination memories, personal and global previous velocity inertia and mutation. After a few iterations, the optimized combination is obtained. The Number of Function Calls (NFC) depends on the population size and the number of iterations. More iterations and particles are needed for larger problems. The pseudo-code of combinatorial PSO combined with continuous is represented in Alg. 4.1.

The proposed combinatorial PSO is tested on few benchmarks and the results for ATT48 problem is represented. ATT48 is a set of 48 points in 2D space, corresponding to the capitals of the states in the USA. The optimum answer to this problem has been represented in the literature long ago. Yet an efficient algorithm to find the best sequence is still a challenge today. The proven shortest path is 10,628 and is nearly achieved by Hybrid Discrete PSO (HDPSO) Wang *et al.* (2006). However, this result has been achieved by an excessive NFC. Typically, a

value around 34,000 can be obtained by a regular cost efficient method Odili & Mohmad Kahar (2016). Note that the lengths are normalized values.

In the literature to solve the TSP, the population is initialized starting from one point and connecting each city to its nearest neighbor Huilian (2010). Furthermore, during the optimization, swaps are done between the cities based on fixed and given distances Wang *et al.* (2003). This approach requires a chart of distances, known and invariable form the beginning. The problem at hand is different than the classical TSP. For the camera placement problem, the cycle time from one image to another is subject to change and depends on the camera's pose, i.e., the distances between the cities can change. Therefore, those methods that require a chart of distances given at the beginning, do not work for this problem. Moreover, in our problem, the start and end city are not fixed and the salesman does not travel in a loop, therefore we have $C!$ possibilities. Due to the aforementioned conditions, this combinatorial image sequence optimization needs more NFC compared to the classical TSP.

The proposed algorithm is called Blind Dynamic map TSP PSO (BD-PSO). *Blind* because it is independent of cities map and calculates the distances as it goes city by city. *Dynamic map* because it can handle changeable distances, i.e., the map dynamically changes. The algorithm is able to find the best camera placement and the smallest cycle time, simultaneously. Put in TSP notion, imagine that there is a book of different maps for a territory. BD-PSO finds which map yields the shortest tour and provides the tour itself. Details about embedded BD-PSO into continuous PSO is represented in the following section.

4.4 Algorithm

In this section, the proposed algorithm is represented in a pseudo-code format. In Alg. 4.1, the main routine of the algorithm is shown. First, the inputs and outputs of the algorithm are declared. The algorithm consists of two sections: initialization of the population and iterations.

Algorithm 4.1 The pseudo-code of the optimization algorithm, based on BD-PSO, to optimize the cost function, fnc.

Input: fnc, S, C, $\mathbf{V}_{\text{lower,upper}}$, $it_{\max}$, n_{pop}, ϕ_ω, ϕ_p, ϕ_g, ψ_ω, ψ_p, ψ_g, ψ_m.
Output: $\mathbf{p}_{\text{cam}}^{\text{best}}$, $\texttt{seq}^{\text{best}}$.
Initialization
for i from 1 to n_{pop} **do**
  $\mathfrak{P}_i^{\mathbf{P}} = \text{UniDist}(S, \mathbf{V}_{\text{lower,upper}})$
  $\mathfrak{V}_i^{\mathbf{P}} = \mathbf{0}_S$
  $\mathfrak{P}_i^{\texttt{seq}} = \text{RndInt}(C)$
  $\mathfrak{V}_i^{\texttt{seq}} = \mathfrak{P}_i^{\texttt{seq}}$
  $\mathfrak{P}_i best = \text{fnc}(\mathfrak{P}_i^{\mathbf{P}}, \mathfrak{P}_i^{\texttt{seq}})$
end
$\mathfrak{P}_g best = \min(\mathfrak{P}_{i \to n} best)$
Iterations
while it_{max} **do**
  for i from 1 to n_{pop} **do**
    for s from 1 to S **do**
      $\mathfrak{P}_i^{\mathbf{P}} =$ Eq. (4.1)
    end
    $\mathfrak{P}_i^{\texttt{seq}} \Leftarrow$ Alg. 4.2
    if $fnc(\mathfrak{P}_i^{\mathbf{P}}, \mathfrak{P}_i^{\texttt{seq}}) < \mathfrak{P}_i best$ **then**
      $\mathfrak{P}_i best = \text{fnc}(\mathfrak{P}_i^{\mathbf{P}}, \mathfrak{P}_i^{\texttt{seq}})$
      if $\mathfrak{P}_i best < \mathfrak{P}_g best$ **then**
        $\mathfrak{P}_g best = \mathfrak{P}_i best$
      end
    end
  end
end
$\mathbf{p}_{\text{cam}}^{\text{best}} \leftarrow \mathfrak{P}_g^{\mathbf{p}} best$
$\texttt{seq}^{\text{best}} \leftarrow \mathfrak{P}_g^{\texttt{seq}} best$

4.4.1 Main routine

As it can be observed in the pseudo-code, the required inputs of the optimization procedure are as follows. The cost function fnc, i.e., the fitness function, determines the fitness value of each particle $\mathfrak{P}$. Each particle is twofold. Continuous parameters, i.e., the camera placement variables shown as $\mathfrak{P}^{\mathbf{p}}$, and combinatorial parameters, i.e., the sequence of the images shown as $\mathfrak{P}^{\texttt{seq}}$. The number of continuous optimization parameters S (for space) is the dimensions

of the search space. The number of combinatorial optimization parameters C (for city) is the number of images. Tensor $\mathbf{V}_{\text{lower,upper}}$ is of size $2 \times S$ and consists of two vectors: lower and upper bound of the initial search field in the continuous space. Maximum number of iterations $it_{\max}$ and population size n_{pop} are standard PSO parameters.

Note that, in this chapter the stopping criteria is set to be the maximum iterations, therefore, the total NFC is $n_{\text{pop}} \times (it_{\max} + 1)$. The rest are the tuning parameters of the optimization algorithm. Coefficients ϕ and ψ are associated with continuous and combinatorial, respectively. Coefficients ϕ_ω, ϕ_p and ϕ_g are the inertia, personal and global learning coefficients of camera placement, respectively. Coefficients ψ_ω, ψ_p, ψ_g and ψ_m are the inertia, personal and global learning coefficients and mutation factor of the image sequence optimization, respectively.

The outputs of the algorithm are the best pose of the camera $\mathbf{p}_{\text{cam}}^{\text{best}}$ and best sequence of images $\mathtt{seq}^{\text{best}}$.

4.4.1.1 Initialization

The initialization section consists of a loop assigning the initial value to all particles. Index i shows the individual particle. In line 4, a vector of size S with uniform distribution within the lower and upper boundary limits is assigned to continuous particles, $\mathfrak{P}^{\mathbf{p}}$. In our inspection workcell, $S = 6$ and each continuous particle consists of 6 values, namely the displacement along and orientation around x-, y- and z-axis. The initial continuous velocity of each particle, $\mathfrak{V}^{\mathbf{p}}$, is a zero vector of size S.

The combinatorial part of the particle is then initialized. A string of numbers from 1 to C is randomly shuffled and assigned to $\mathfrak{P}^{\mathtt{seq}}$. Then, the velocity of combinatorial part $\mathfrak{V}^{\mathtt{seq}}$ is initialized to be the same as the particle itself, because this implies zero velocity. Next, once the pose and combination of the i^{th} particle is set, they are passed into the cost function fnc and the fitness of the particle is returned.

The fnc is represented in Alg. 4.3 and will be described in the upcoming paragraphs. The output of fnc is the cycle time of the particle. For the initialization, the current pose and combination is stored as the particle's personal best memory $\mathfrak{P}_i best$. Note that $\mathfrak{P}_i best$ contains both continuous and combinatorial parameters.

Finally, after all particles are assigned with the initial values and their best memory is captured, the best global memory is extracted from comparing all the members of population and the one with the minimum cycle time is stored in $\mathfrak{P}_g best$.

4.4.1.2 Iterations

In the iterative section, each particle decides about its search direction based on personal and global memories and explores new possibilities. From lines 11 to 26, the algorithm starts the iterations and stops when $it_{\max}$ number of iterations are done. In lines 12 to 25, for each particle i, the pose and combination is updated and the fitness value is obtained. In lines 13 to 15, the pose of the camera for each dimension S is updated based on Eq. (4.1). In line 16, the combination of the sequential part of particle is updated based on Alg. 4.2. In lines 17 to 22, the fitness is evaluated with updated particles values. In lines 17 and 19, if for a particle the new pose and combination yield to a better cycle time, the personal best memory is updated with the new values. In lines 19 and 20, if the current particle's personal updated memory is better that of its global, the global best memory is replaced with the current particle's attributes. After $it_{\max}$ number of iterations, the existing global best pose and combination $(\mathfrak{P}_g^{\mathbf{p}} best, \mathfrak{P}_g^{\mathtt{seq}} best)$ are represented as the optimized camera placement and image sequence.

4.4.2 BD-PSO

In Alg. 4.2, the pseudo-code of the BD-PSO is represented. In line 1, three counters are created. Counters l_p, l_g and l_v keep track of how many times a particle has learned from its personal best combination, swarm global best combination and previous velocity combination, respectively.

Algorithm 4.2 The pseudo-code of the combinatorial BD-PSO to update the particles combination. rnd$(0,1)$ is a random real number generated independently each time.

$l_{\mathrm{p}} = 0$, $l_{\mathrm{g}} = 0$, $l_{\mathrm{v}} = 0$
——————————— Inertia from previous iteration ———————————-
for c from 1 to C **do**
  if $\mathfrak{P}_{i,c}^{\mathtt{seq}} \neq \mathfrak{V}_{i,c}^{\mathtt{seq}}$ ***and*** $rnd(0,1) < \psi_{\omega}$ **then**
    in $\mathfrak{P}_{i,c}^{\mathtt{seq}}$ swap($\mathfrak{P}_{i,c}^{\mathtt{seq}}$, $\mathfrak{V}_{i,c}^{\mathtt{seq}}$)
    $l_{\mathrm{V}}++$
  end
end
——————————————— Personal learn ———————————————
for c from 1 to C **do**
  if $\mathfrak{P}_{i,c}^{\mathtt{seq}} \neq \mathfrak{P}_{p,c}^{\mathtt{seq}} best$ ***and*** $rnd(0,1) < \psi_{p}$ **then**
    in $\mathfrak{P}_{i,c}^{\mathtt{seq}}$ swap($\mathfrak{P}_{i,c}^{\mathtt{seq}}, \mathfrak{P}_{p,c}^{\mathtt{seq}} best$)
    $l_{\mathrm{P}}++$
  —— Prepare velocity for next iteration
  else if $\mathfrak{P}_{i,c}^{\mathtt{seq}} \neq \mathfrak{P}_{p,c}^{\mathtt{seq}} best$ **then**
    in $\mathfrak{V}_{i,c}^{\mathtt{seq}}$ swap($\mathfrak{V}_{i,c}^{\mathtt{seq}}, \mathfrak{P}_{p,c}^{\mathtt{seq}} best$)
  end
end
——————————————— Global learn ———————————————-
for c from 1 to C **do**
  if $\mathfrak{P}_{i,c}^{\mathtt{seq}} \neq \mathfrak{P}_{g,c}^{\mathtt{seq}} best$ ***and*** $rnd(0,1) < \psi_{g}$ **then**
    in $\mathfrak{P}_{i,c}^{\mathtt{seq}}$ swap($\mathfrak{P}_{i,c}^{\mathtt{seq}}, \mathfrak{P}_{g,c}^{\mathtt{seq}} best$)
    $l_{\mathrm{g}}++$
  —— Prepare velocity for next iteration
  else if $\mathfrak{P}_{i,c}^{\mathtt{seq}} \neq \mathfrak{P}_{g,c}^{\mathtt{seq}} best$ **then**
    in $\mathfrak{V}_{i,c}^{\mathtt{seq}}$ swap($\mathfrak{V}_{i,c}^{\mathtt{seq}}, \mathfrak{P}_{g,c}^{\mathtt{seq}} best$)
  end
end
——————————————— Mutation ———————————————
if $l_v + l_p + l_g = 0$ **then**
  for μ from 1 to ceil$(C/5)$ **do**
    in $\mathfrak{P}_{i,c}^{\mathtt{seq}}$ swap two random cities
  end
end

All these learning counters are zero at the beginning. The BD-PSO consists of four major sections.

4.4.2.1 Inertia

In lines 2 to 7, the first section which is the effect of inertia is represented. In these lines, it is checked that if a city c in the particle sequence $\mathfrak{P}_i^{\mathtt{seq}}$ is different than the city in the velocity sequence $\mathfrak{V}_i^{\mathtt{seq}}$, possibly city c in the particle will be swapped with the one in the velocity. The velocity sequence is generated from last iterations and contains those possible swaps that were not done in the previous sequence.

In line 3, if the aforementioned condition is met, there is a chance that the swap is done, because it is done only if a randomly generated real number between 1 and 0 ($\mathrm{rnd}(0,1)$) is smaller than the inertia coefficient. This random chance swap prevents the algorithm from premature convergence. This will be observed in other learning sections as well. Without this likelihood implementation, after the first iteration all the particles would be the same as global best. This is not desirable and we want the particles to explore new possibilities.

In line 4, the swap operation is applied. See Eq. (4.2) as an example for a swap in particle's combination. Assume that in this example $c = 2$ (i.e., the swap is being checked for the second city) and $\mathrm{rnd}(0,1) < \psi_\omega$ is true.

$$\begin{aligned} \text{Velocity combination:} \quad & \mathfrak{V}_i^{\mathtt{seq}} = \{5,\ 3,\ 6,\ 4,\ 1,\ 2\} \\ \text{Particle combination before swap:} \quad & \mathfrak{P}_i^{\mathtt{seq}} = \{5,\ 2,\ 4,\ 3,\ 6,\ 1\} \\ \text{Particle combination after swap:} \quad & \mathfrak{P}_i^{\mathtt{seq}} = \{5,\ 3,\ 4,\ 2,\ 6,\ 1\}. \end{aligned} \tag{4.2}$$

As it can be observed, second city in velocity is 3 and second city in particle is 2. Therefore, the second city in particle is swapped to match the velocity's second city (2 and 3 are swapped in particle).

This concept is applied to every city in particle and if they are lucky enough ($\mathrm{rnd}(0,1) < \psi_\omega$), they will be swapped. Note that the random number is independently generated each time it is called. In line 5, velocity learning counter is increased by 1. It will be useful later.

4.4.2.2 Personal learning

In the second section from line 8 to 15, the particle learns from its own personal best. In lines 9 to 11, similar to previous section, the particle updates its sequence if it differs from its personal best memory, and if the randomly generated number is smaller than the personal learning coefficient. In line 11 the personal learning counter increases. In lines 12 to 14, if city c in the particle is different than its counterpart in the particle's personal best, but did not have the chance to swap (the random number was larger than l_p), the swap is stored in the velocity. This velocity will be used in the next iteration. In this vein, in the next iteration the particle still has a chance to take advantage of the unused swap. Therefore, in line 13, the swap is done for the velocity sequence and c^{th} city in the velocity is updated with the according city of particle's personal best.

4.4.2.3 Global learning

In the third section of the algorithm from line 16 to 23, the same concept from the previous section is applied for global learning . The particle has a chance to learn from the best memory of the whole swarm. In line 17, the learning coefficient l_g determines the chance and in line 18 swap is done and global learning counter increases in line 19. In line 21, the possible useful swaps are stored in velocity for next iteration.

4.4.2.4 Mutation

In the final section of this algorithm, from line 24 to 28, a mutation is applied to the particle. Mutation is applied when a particle has learned nothing from the previous sections, meaning that the particle's sequence is exactly the same as its previous iteration with no change. In this case, random swaps in the particle happens. It improves the efficiency of the algorithm, because if a particle is the same as its previous iteration, a call to the fitness function is made with no new results. Therefore, in line 24, the algorithm checks if all the learning counters are still zero, i.e., no change from the beginning, some of the cities are chosen randomly and

swapped. In line 25, depending on the total number of cities, some of the cities are chosen randomly and swapped. Approximately 20% of the cities are randomly swapped.

Algorithm 4.3 The pseudo-code of the Cost function, fnc, to calculate the cycle time of the inspection phase.

1 **Input**: $\mathfrak{P}_i^{\mathbf{p}}$ as the pose of the camera, $\mathfrak{P}_i^{\mathtt{seq}}$ as the sequence of the images
2 **Output**: Cycle time
——————————— Set camera and images pose ———————————
3 $\mathbf{H}_{\text{base}}^{\text{cam}} = \text{pose2tran}(\mathfrak{P}_i^{\mathbf{p}})$
4 **for** *c from 1 to C* **do**
5 $\quad {}_c\mathbf{H}_{\text{base}}^{\text{flange}} = \mathbf{H}_{\text{blade}}^{\text{flange}} \mathbf{H}_{\mathtt{img}_c}^{\text{blade}} \mathbf{H}_{\text{cam}}^{\mathtt{img}_c} \mathbf{H}_{\text{base}}^{\text{cam}}$
6 **end**
——————————————— Calculations ———————————————
7 **try:**
8 $\quad \text{IKP}({}_1\mathbf{H}_{\text{base}}^{\text{flange}})$
9 $\quad$ **for** *c in* $\mathfrak{P}_i^{\mathtt{seq}}$ **do**
10 $\quad\quad \mathbf{ct}_c = \text{CycleTime}({}_c\mathbf{H}_{\text{base}}^{\text{flange}}, {}_{(c+1)}\mathbf{H}_{\text{base}}^{\text{flange}})$
11 $\quad\quad \text{CollisionCheck}({}_c\mathbf{H}_{\text{base}}^{\text{flange}}, {}_{(c+1)}\mathbf{H}_{\text{base}}^{\text{flange}})$
12 $\quad$ **end**
13 **catch** *fail*:
14 $\quad$ Collision or out of reach
15 $\quad$ **return** inf
16 $\quad$ **break**
17 **end**
18 **return** $\sum \mathbf{ct}$

4.4.3 Cost function (fnc)

The fitness function fnc of the optimization algorithm is represented in Alg. 4.3. The inputs of fnc, as shown in line 1, are both the continuous and combinatorial part of a particle, i.e., the pose of the camera and the sequence of the images, respectively. The output of fnc, i.e., the fitness value, is a single number as the cycle time of the blade inspection operation. The function fnc is twofold: (1) set camera's pose and consequently the inspection poses and (2) the cycle time calculation and collision check.

4.4.3.1 Set camera and image pose

In the first section, in line 3, using the pose2tran() function, the camera's pose is converted from the 6×1 pose vector of $\mathfrak{P}^{\mathbf{p}}$, i.e., translation and orientation, into a homogeneous 4×4 transformation matrix represented with respect to the global base frame. In lines 4 to 6, each image pose, which is given in the camera's frame, is represented as a transformation matrix with respect to the base. In line 5, the transformation matrix from robot's flange to the base frame is obtained. In line 5, the transformation matrices are as follows: camera's location with respect to the base ($\mathbf{H}^{\text{cam}}_{\text{base}}$), c^{th} image with respect to camera ($\mathbf{H}^{\texttt{img}_c}_{\text{cam}}$), reference frame of the blade with respect to c^{th} image ($\mathbf{H}^{\text{blade}}_{\texttt{img}_c}$) and robot's flange with respect to blade's reference frame ($\mathbf{H}^{\text{flange}}_{\text{blade}}$). Note that $\mathbf{H}^{\texttt{img}_c}_{\text{cam}}$, $\mathbf{H}^{\text{blade}}_{\texttt{img}_c}$ and $\mathbf{H}^{\text{flange}}_{\text{blade}}$ are invariable and only $\mathbf{H}^{\text{cam}}_{\text{base}}$ can change. Consequently, the pose of the flange with respect to the base (${}_c\mathbf{H}^{\text{flange}}_{\text{base}}$) is obtained for c^{th} image.

4.4.3.2 Calculations

At this point all we need is to compute the cycle time and check for any collisions. In lines 7 to 12, the algorithm *tries* to calculate the cycle time and checks for collisions. In line 8, the IKP is solved for the first image in the sequence. Since this is being done in a *try and catch* block, if any of the functions in the *try* block fails, it is *caught* in line 13 and the fitness value of the particle is returned as infinite. In lines 9 to 12, for each image c, the cycle time is calculated (line 10) and stored in $\mathbf{ct_c}$. Furthermore in line 11, all possible collisions are checked.

The try block may fail in two general cases: (1) if there is a collision or (2) if the IKP has no solution. The IKP may fail in two cases: when the Cartesian target $\mathbf{H}^{\text{flange}}_{\text{base}}$ (1) is out of reach for the robot or (2) is at a singularity. In line 17, in case of no failure, the sum of the cycle times for all the images is returned as the fitness value of the particle. Otherwise, inf is returned and basically the current state of the particle is useless and will be ignored by the swarm.

4.4.4 Cycle time and collision check

In Alg. 4.4, two subroutines which were used in the previous section are explained, namely cycle time and collision check.

4.4.4.1 Cycle time

J1	J2	J3	J4	J5	J6
0.35	0.25	0.2	0.1	0.05	0.05

Table 4.1 Weights used in WJTT

In order to calculate the cycle time, two sets of joint values are needed: start and end one. In line 1, the resultant matrix transformation to get the pose of c^{th} image, ${}_{c}\mathbf{H}^{\text{flange}}_{\text{base}}$ and the image after, ${}_{(c+1)}\mathbf{H}^{\text{flange}}_{\text{base}}$ are given as inputs. In line 2, the output of this subroutine is the cycle time, ct. In line 3, the IKP is solved for the image pose corresponding to the beginning of movement and robot's joint values are stored in $\mathbf{q}_{\text{st}}$. In line 4 to 7, the joint value difference between start and end joint set is obtained for all eight robot configurations. In line 5, the IKP is solved for the end joint set at the m^{th} configuration of the robot and the solution is stored in an array called $\mathbf{q}_{\text{end}}$. In line 6 the difference in the joint values are calculated and stored in an array called $\mathbf{D}$. Note that the difference is fist multiplied to the weight vector ω (represented in Table. 4.1) to apply the joint response time. In line 7, the set of smallest joint value difference among all the configurations is stored in $\mathbf{d}$. In case the IKP fails, the loop breaks and failure is returned to superior calculation routine. Otherwise, in line 9, the maximum cycle time value among the six joints is returned as the cycle time.

The working mode of the robot may change from one image to another. The method behind choosing the right working mode (also referred to as configuration) is illustrated in Fig. 4.2. In this figure, from left to right, the *approach point* is a fixed point in space, also referred to as *home position*. After the robot has polished the blade, it re-positions at the approach point. From the approach point, the IKP is solved for all 8 possible configurations of the robot in order

to reach the pose for image 1. The configuration that requires the smallest cycle time is chosen. In Fig. 4.2, the smallest cycle time is shown by a dotted circle around the approach point. In the illustrated example, the 3rd working mode result in smallest cycle time. Then, from the 3rd working mode to reach image 2, the 4th working mode results in the smallest cycle time. For some working modes, it might happen that no solution can be found for the robot. The working mode selection process continues for all images. Note that this is a local optimum solution and each working mode is only guaranteed to be the best choice regarding the previous solution.

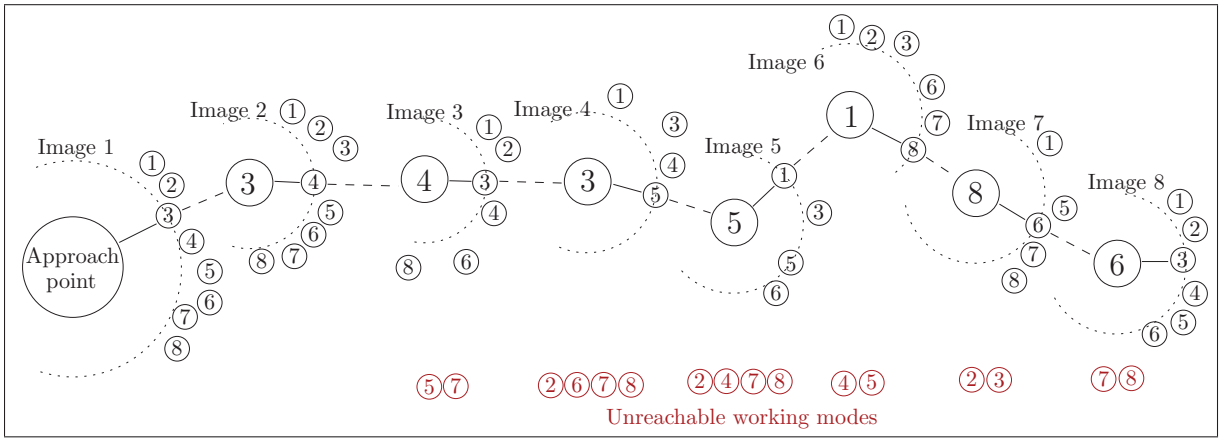


Figure 4.2 Working mode selection for each image.

4.4.4.2 Collision check

In the second subroutine, all possible collisions in the workcell are checked. Inputs are the CAD models of all the parts in the workcell. The output is only a flag indicating if the collision check is passed. If only one of them fails, the whole cycle fails. This task is performed by an external CAD engine collision check program in line 11. However, it is important to check for the collisions not only at the beginning and ending point, but also along the joint mode pose-to-pose movement. Therefore, in line 10, joint values between start $\mathbf{q_{st}}$ and end $\mathbf{q_{end}}$ joint sets are divided into approximately 10 steps. Of course, if the resolution between $\mathbf{q_{st}}$ and $\mathbf{q_{end}}$ is higher, the results are more reliable, but the calculation time will increase as well.

Algorithm 4.4 The pseudo-code of two routines: CycleTime() and CollisionCheck().

```
——————————————— CycleTime ———————————————
Input: ${}_{c}\mathbf{H}^{\text{flange}}_{\text{base}}$ and ${}_{(c+1)}\mathbf{H}^{\text{flange}}_{\text{base}}$
Output: ct
$\mathbf{q}_{\text{st}}$ = IKP(${}_{c}\mathbf{H}^{\text{flange}}_{\text{base}}$)
for m from 1 to 8 do
    $\mathbf{q}_{\text{end}} = \text{IKP}\Big|_{m}\left({}_{(c+1)}\mathbf{H}^{\text{flange}}_{\text{base}}\right)$
    $\mathbf{D}_m = \omega(\mathbf{q}_{\text{st}} - \mathbf{q}_{\text{end}})$
end
$\mathbf{d} = \max_{\text{Column}} \mathbf{D}$
return fail if out of reach
return min d as ct
——————————————— CollisionCheck ———————————————
Input: Cad model of all parts in workcell
Output: fail or pass
for step from $\mathbf{q}_{\text{st}}$ to $\mathbf{q}_{\text{end}}$ each $(\max(\mathbf{q}_{\text{end}} - \mathbf{q}_{\text{st}})/10)$ degree do
    Check for geometrical collision
    return fail if collision
end
```

4.5 Case study and discussion

In this section the proposed algorithm is tested to see whether it is reliable to solve combined (continuous and combinatorial) problems. First, the continuous factor is eliminated in order to verify the algorithm's power only in combinatorial problems, where a map of 13 cities is considered for the TSP and the performance of the algorithm is analyzed. Then, the case study of this chapter, i.e., turbine blade inspection cell, is resolved using the proposed algorithm.

4.5.1 Combinatorial test case of 13 cities

The city map of this combinatorial problem test case is shown at the top of Fig. 4.3. These points (cities) are chosen in a fashion that it is easy to predict the optimal path, which is a circular route. Furthermore, the points are chosen so that they are not symmetrically or homogeneously distributed. It is worth recalling the fact that BD-PSO algorithm is blind in terms of

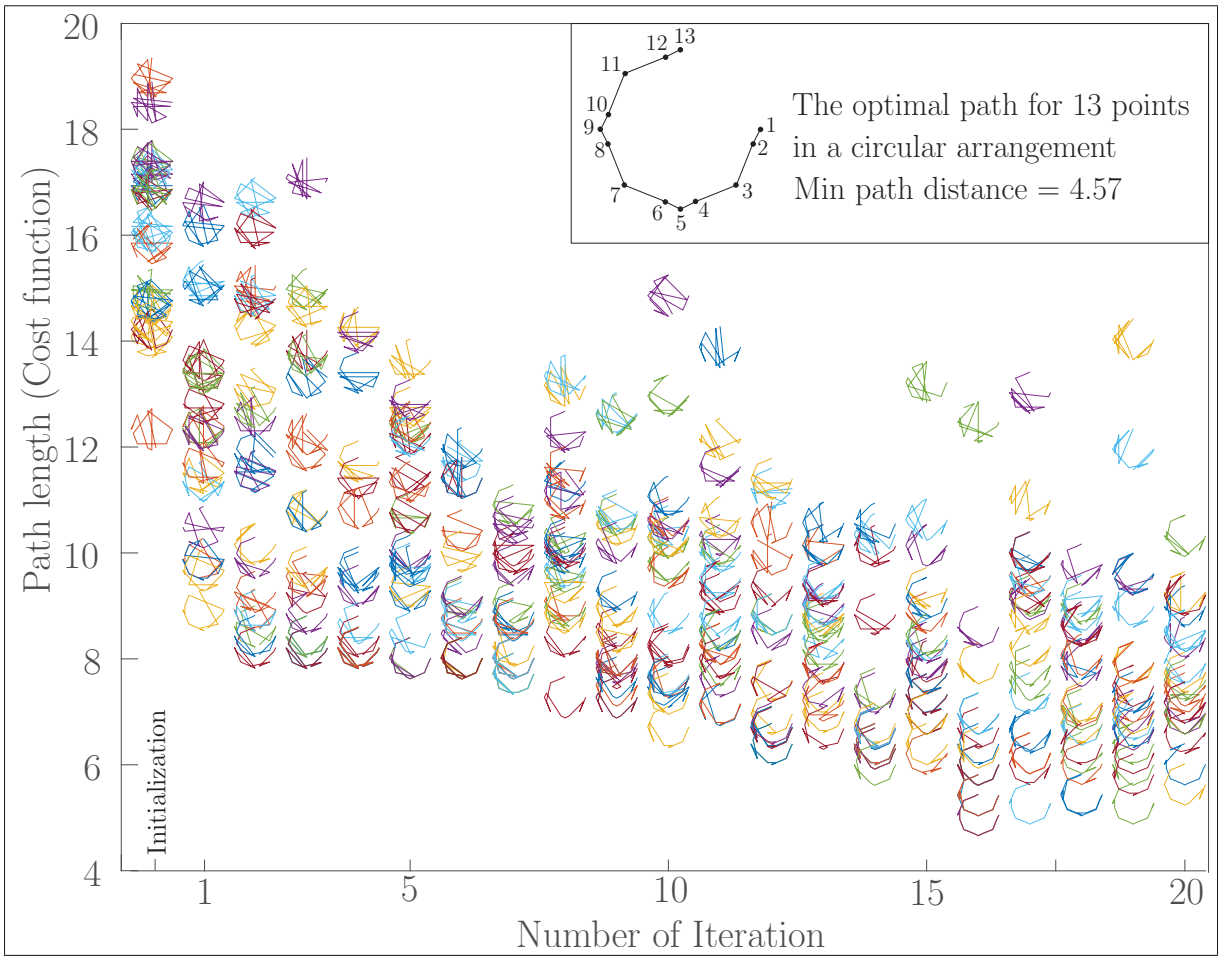


Figure 4.3 A test case to measure reliability of the BD-PSO algorithm.

knowing the distances between the cities in advance. Therefore, no decision is made based on proximity of two points or predicting what would be the consequence of certain swaps. The solution purely evolves by manipulating a sequence of numbers, i.e., $\texttt{seq} = [1, 2, \ldots, 13]$.

In Fig. 4.3, the evolution of particles is depicted throughout the iterations. For this optimization, 20 particles are evaluated for 20 iterations. In total, $400 = 20 \times 20$ NFC. Each column in the figure indicates an iteration, therefore, there are 20 particles represented in each column. For each particle, a broken line is drawn with respect to the sequence represented by the particle. For example, on the top of the figure, the drawing of the minimal route is shown. In the first column located on NFC 0, the initial random generation of 20 particles in the swarm is

illustrated. The cost function, i.e., path length, obtained for the initial generation is between 12.2 to 19, see Fig. 4.4. In the second column, a sensible drop in route length is observed between 9 and 17. From the first column to the eighth, all the particles are converging to a solution, because there is always either a global, personal or velocity learned swap done ($l_g + l_p + l_v \neq 0$). From the ninth iteration, a diverging behavior is observed, see Fig. 4.3. That is where the mutation has started. This happens when some of the particles are brought forward to the next iteration without a single change. In order to avoid calling the cost function for a previously calculated sequence, at least one random swap is done in the particle. For this reason in the later iterations, more outliers are emerging to explore new opportunities.

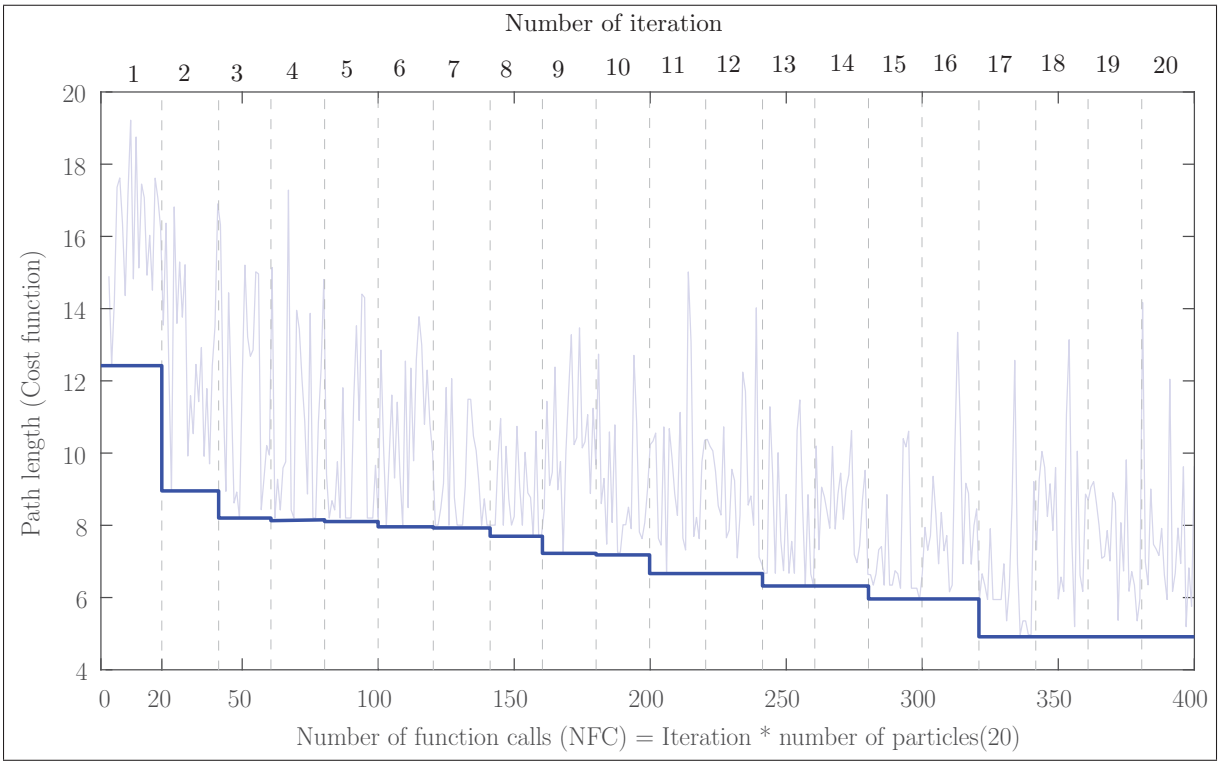


Figure 4.4 Convergence plot of 13-point test case using BD-PSO. From NFC 0 to 20 is the initial random generated population. Spikes of longer path length in the plot shows the mutation-based attempt to find new solutions.

For this optimization the stopping criteria is set to the maximum number of iterations. As it can be observed, after 400 NFC the result of path length is 4.87. Even though the optimal solution is known to be 4.57, the obtained result is quite satisfactory, because without any optimization,

$13! = 6,227,020,800$ NFC is needed to find the best solution. Yet after only 400 NFC, a near optimal solution is found. Moreover, this is done in a blind fashion, without knowing the distances between each two points.

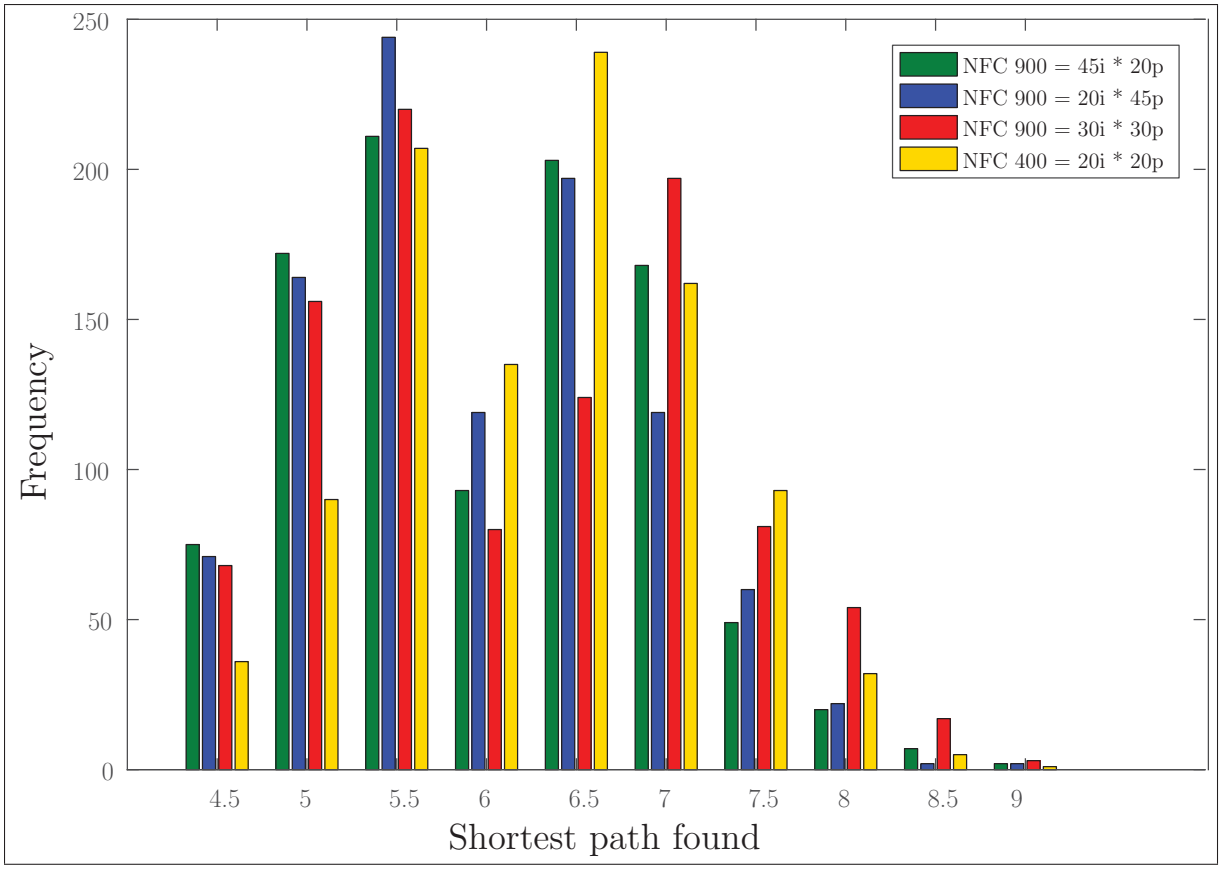


Figure 4.5 Histogram of comparison between four combinations of population size and max number of iterations. For each combination, the algorithm is tested 1000 times and distribution of the result shows the repeatability of the algorithm.

Finally, a comparison is done between various combinations of population size and maximum number of iterations in Fig. 4.5. Furthermore, this study compares 1000 runs of the algorithm to obtain its repeatability. In Fig. 4.5, four different combinations are compared, where i stands for the maximum number of iterations and p for the population of swarm. For example, NFC $900 = 45i \times 20p$ means 45 iterations are done for a population of 20 particles. Evidently, better results are obtained with more calls to the cost function, i.e., NFC 900 gives a better overall result than NFC 400. However at a fixed NFC, it would be interesting to examine whether a

larger p is more important than a larger i. As a future work, a full analysis of the method, e.g., fine tuning the learning coefficients and NFC combinations, using more TSP benchmarks, is suggested.

4.5.2 Turbine blade inspection test case

In this section, the proposed algorithm is applied on a simulated turbine blade polish and inspection workcell. As it can be observed in Fig. 4.1, a 6-DOF serial robot (UR5, from Universal Robots) is located in the middle of the workcell. It grabs a turbine blade from the conveyor and takes it to belt grinder machine (phase 1). After the polishing operation is done, the robot takes the blade in front of the camera to take the images (phase 2). The location of the camera and the sequence of the images are optimized with the proposed algorithm.

In this example, eight images from different angles are needed, see Fig. 4.6. Therefore, there are eight inspection poses of the end-effector with respect to the camera. These poses are illustrated in Fig. 4.7. Eight reference frames correspond to the poses.

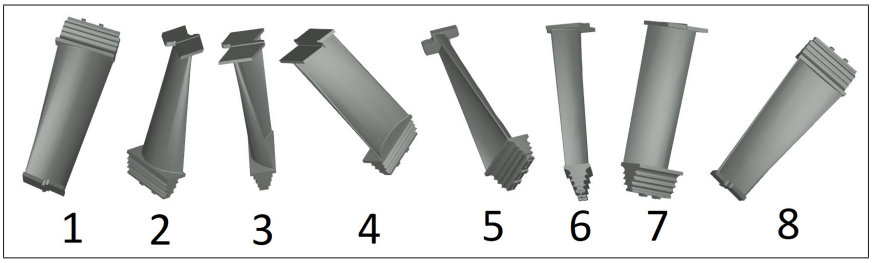


Figure 4.6 Eight images needed to be taken from different angles of the turbine blade for the sake of inspection.

In Fig. 4.8, a comprehensive evolution of particles during the optimization process is depicted. On the top the figure, the shortest path of an 8-point map is shown. This is a representation of the sequence of eight images.

In Fig. 4.8, the evolution of the particles to optimize the inspection workcell using BD-PSO is depicted. Each octagonal shape drawing represents the sequence of the particle. For example,

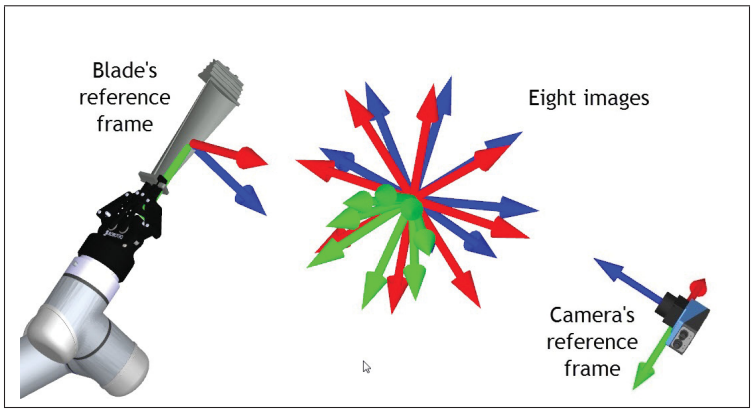


Figure 4.7 Eight reference frames corresponding to the eight images.

the upper particle in the first column (with highest cycle time value of around 565) represents the following image sequence: $\{2,7,5,8,3,6,1,4\}$. One can match the particle to the large octagonal (top right of Fig. 4.8) to obtain this sequence. Each column represents one iteration. The first column is the initial randomly generated population. The cost function is the cycle time of taking all eight images. The population size is 30 particles. As it can be noticed from the figure, the first column has less than 30 particles. This is due to the fact that for some particles (pose of camera and combination), there is no solution found, either because of collisions or because of workspace limits. From the first column (for which the cost is between 300 and 570) to the second column (cost between 210 and 425) a drastic drop in cycle time is observed. This drop continues up to the fourth iteration, where the mutation feature starts to act on particles. After that, there is always mutations in the particle to avoid repeated calculations.

It is worth comparing the impact of image sequence and camera pose on the cycle time to see which one is more effective to reduce the cycle time. For instance, as it can be seen in Fig. 4.8 between 25 and 35 NFC, for the same sequence at the bottom, there exist a slight difference in cycle time among the particles. These differences are caused by the camera pose adjustment and ranges in about 207 to 216 (9 unit difference in cycle time). Comparing the difference in the sequence, which ranges between 160 to 187 (about 17 units in cycle time), it is safe to say that the image sequence optimization is much more important than the camera location adjustment. However, finding a camera location that results in a solution is still very

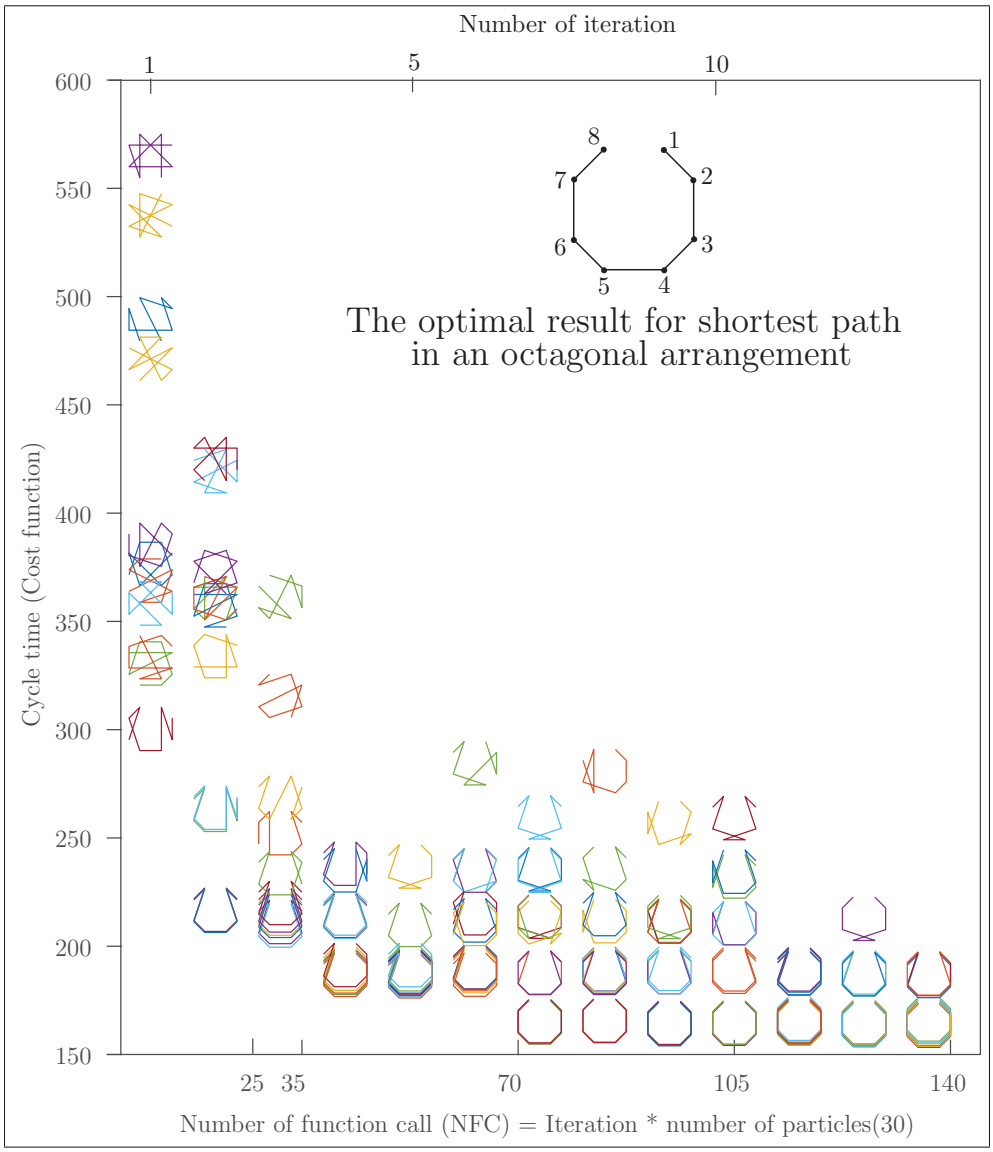


Figure 4.8 Evolution of the particles to optimize the inspection workcell using BD-PSO. Each octagonal shape drawing represents the sequence of the particle.

important. Moreover, the camera placement may be of more importance given a different set of images. Eventually, a near optimal solution is obtained in about 70 NFC (0.17% of 8! total possibilities), see Fig. 4.10. The final placement of the camera is represented in Fig. 4.9 and the obtained sequence is $\{1,2,3,4,5,6,7,8\}$.

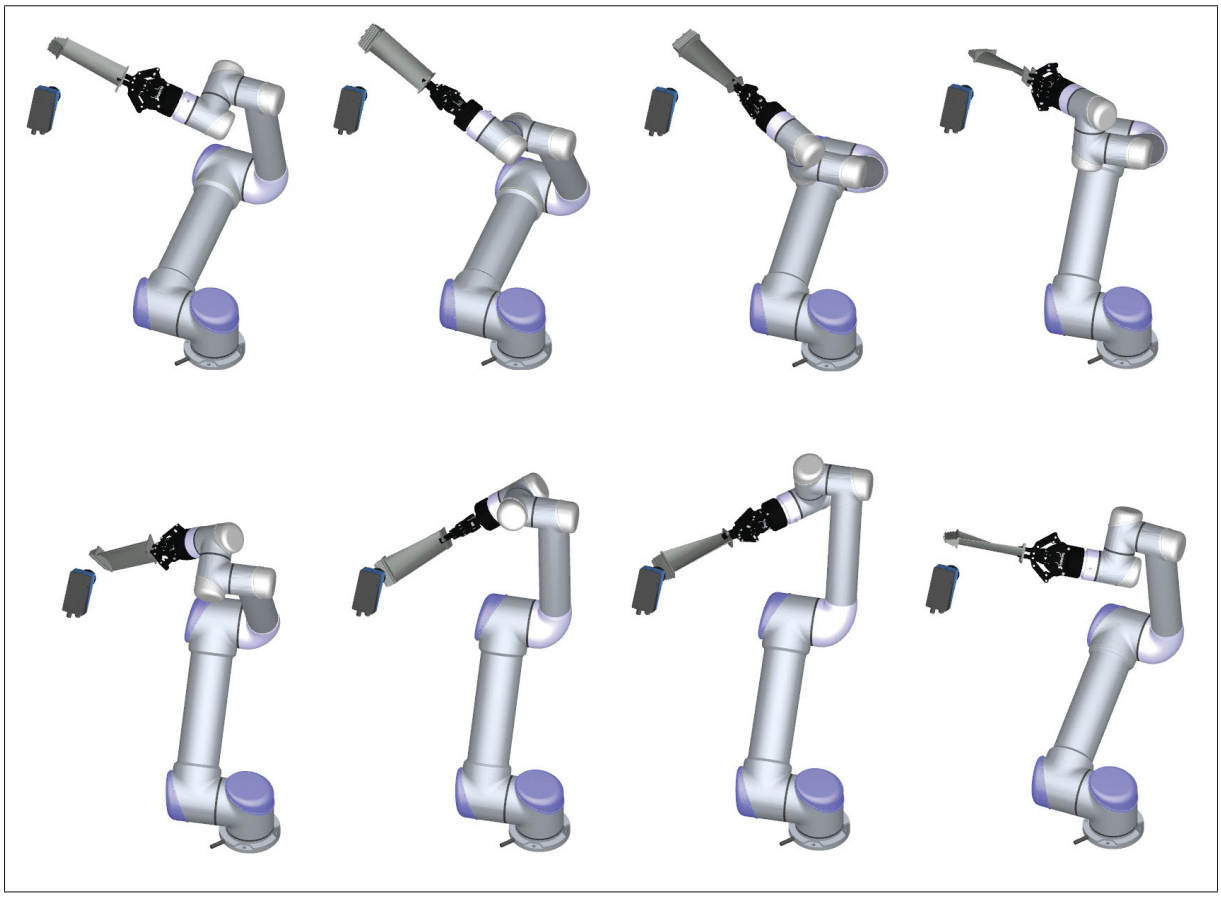


Figure 4.9 8 robot poses to perform image inspection for 8 complicated images.

4.6 Conclusion

In this chapter, an evolutionary based optimization method, called BD-PSO was proposed to simultaneously solve the continuous problem of camera placement optimization and the combinatorial problem of image sequence optimization. The continuous search space of camera placement includes 6 degrees of freedom. The combinatorial solution was the shortest traveling path in a TSP for a blind and dynamic map space, meaning that the distances between the cities are not known (blind) and can change as a function of the camera placement (dynamic). Details of the algorithm were represented where the learning coefficients for personal, global, previous velocity and mutation are defined for both aspects of the problem. The proposed algorithm was verified by comparing the result to a TSP benchmark. A near optimal result of

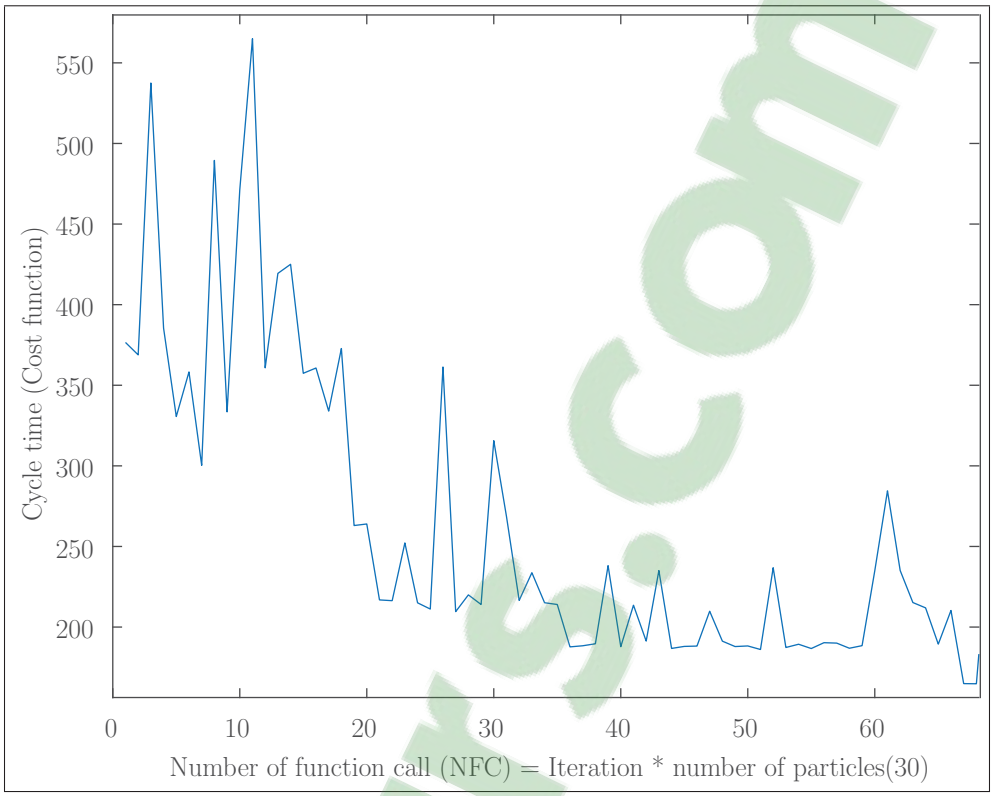


Figure 4.10 Convergence of cycle time in the optimization of inspection workcell.

(4.87, compared to 4.57) was obtained after 6.42×10^{-8} NFC fraction of total possibilities. A case study was introduced for a turbine blade polishing and inspection workcell.

Optimal design of workcell layout was obtained. In this optimization the objective was to minimize the total cycle time of the process and collision avoidance was respected as a limit of the solutions. In future works, fine TSP solution is suggested to be performed at the end when the camera location is fixed, for high number of images.

CHAPTER 5

CONCLUSIONS

Path placement optimization is a key step for an efficient robotic setup. Without a proper placement, occasionally, it can be wrongly interpreted that a given robotic task is impossible to perform, yet it is only a matter of properly choosing the right layout. Among all the benefits of foreseeing components arrangement within the layout, singularity avoidance and cycle time improvement are of prominent value. There are a variety of parameters involved in path and component placement. Often times the number of such parameters is especially high and not all are independently effective on our decision makings. Path placement is usually combined with path planning and should be solved simultaneously.

The present study touched on path placement and path planning optimization in industrial robotic workcells. As a whole, this thesis views the placement problem from different angles. It consisted of three main topics. First, in Chapter 2, path placement and component placement in a highly redundant collaborative redundant setup was addressed. The cell layout was obtained, based on optimized path placement in a fiber placement cell. Singularity avoidance and redundancy resolution was achieved in a 13-DOF cell. Usually a highly experienced human practitioner is required to investigate such cells, with lots of trial and error, to solve path planning and component placement. But with the proposed algorithm represented in Chapter 2, all aforementioned tasks are efficiently and automatically performed within a few iterations.

Second, in Chapter 3, taking a step backward to see the big picture, the number of independent parameters effective in path placement was investigated. Without having this number carefully studied, a big portion of the previous algorithm was doomed to run for inefficient loops. This is specially the case for higher DOF workcells and redundancy resolution. Results showed about 30% improvement in calculation time for a ski goggles gluing task. Furthermore, useful suggestions were given on what redundancy provider to choose.

Finally in Chapter 4, a complex combinatorial component placement and path planning task for turbine blade inspection cell was investigated. A novel algorithm was introduced to simultaneously obtain the best placement of an inspector camera and sequence of the images to be taken. The proposed method is especially usefully in highly dynamic and unpredictable environments where most other classic algorithms are unable to provide a reliable result. Other non-robotic applications, with a dynamic and unknown map of TSP problem can also benefit from this algorithm.

In the future works, regarding the methodology to obtain the minimal number of parameters required for task placement, only seven DOF systems (one DOR) are considered. This study can be extended to higher DORs. For example using the same approach, obtain the minimum number of effective parameters for dual rotaries, dual rails, etc. It would be interesting and quite challenging to combine rotary and linear RPs in an eight DOF redundant cell as well. Moreover for optimizing trajectory, only the distance to singularity is considered as the optimization criterion and collision avoidance as a constraint. But other measures, such as condition number, manipulability or dexterity, or energy consumption can be explored. Furthermore, the combinatorial optimization method to solve TSP is applicable to other disciplines and can be improved by combining with other approaches.

Related future works that are not in scope of this thesis can be task distribution in multi-robot cells which is still quite challenging to this date. Also, online programming for such cells brings many new challenges to the table. A major challenge in future works is how to implement some of proposed methods so that they can be used in an intuitive manner by users who are not experts in robotics or optimization. This is a real-world issue right now for the author as he is working in industrial robotic simulation section.

In conclusion, this thesis investigated different aspects of path and component placement. Various applications and numerous situations were investigated that may arise in a robotic cell layout design and placement, as well as path planning. The authors hope that the present work

can be of value to the robotics community and help future works to rely on an efficient way of doing path placement, in a variety of robotics problems.

BIBLIOGRAPHY

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(4), 409–417.

Alatartsev, S., Stellmacher, S. & Ortmeier, F. (2015). Robotic task sequencing problem: A survey. *Journal of Intelligent & Robotic Systems*, 80(2), 279–298.

Aneja, Y. P. & Kamoun, H. (1999). Scheduling of parts and robot activities in a two machine robotic cell. *Computers & Operations Research*, 26(4), 297–312.

Angeles, J. & Angeles, J. (2002). *Fundamentals of robotic mechanical systems*. Springer.

Ata, A. A. (2007). Optimal trajectory planning of manipulators: a review. *Journal of Engineering Science and Technology*, 2(1), 32–54.

Baizid, K., Chellali, R., Yousnadj, A., Meddahi, A. & Bentaleb, T. (2010). Genetic algorithms based method for time optimization in robotized site. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1359–1364.

Baizid, K., Meddahi, A., Yousnadj, A., Chellali, R., Khan, H. & Iqbal, J. (2014). Robotized task time scheduling and optimization based on Genetic Algorithms for non redundant industrial manipulators. *Robotic and Sensors Environments (ROSE), 2014 IEEE International Symposium on*, pp. 112–117.

Blom, A. W., Tatting, B. F., Hol, J. M. & Gürdal, Z. (2009). Fiber path definitions for elastically tailored conical shells. *Composites part B: engineering*, 40(1), 77–84.

Bonev, I., Zlatanov, D. & Gosselin, C. (2002). Advantages of the modified Euler angles in the design and control of PKMs. *2002 Parallel Kinematic Machines International Conference*, pp. 171–188.

Bonev, I. A. & Gosselin, C. M. (2006). Analytical determination of the workspace of symmetrical spherical parallel mechanisms. *IEEE Transactions on Robotics*, 22(5), 1011–1017.

Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19), 16.

Caro, S., Dumas, C., Garnier, S. & Furet, B. (2013). Workpiece placement optimization for machining operations with a KUKA KR270-2 robot. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 2921–2926.

Chan, K. & Zalzala, A. (1993). Genetic-based minimum-time trajectory planning of articulated manipulators with torque constraints. *Genetic Algorithms for Control Systems Engineering, IEE Colloquium on*, pp. 4–1.

Chen, P.-W. & Chen, K.-J. (2015). A CRASHWORTHINESS SIMULATION FOR A LIGHT AIRCRAFT CONSTRUCTED OF COMPOSITE MATERIALS. *Transactions of the Canadian Society for Mechanical Engineering*, 39(4), 829–843.

Cheng, X. (1995). On-line collision-free path planning for service and assembly tasks by a two-arm robot. *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, 2, 1523–1528.

Chiu, Y.-J. & Perng, M.-H. (2004). Self-calibration of a general hexapod manipulator with enhanced precision in 5-DOF motions. *Mechanism and Machine Theory*, 39(1), 1–23.

Craig, J. J. Introduction to robotics. 1986. *Reading, MA: Addison*.

Dasgupta, B. & Mruthyunjaya, T. (2000). The Stewart platform manipulator: a review. *Mechanism and machine theory*, 35(1), 15–40.

Denavit, J. (2000). Richard S Hartenberg and the symbolic notation (a personal reflection). *Mechanism and Machine Theory*, 35(6), 757–760.

Erdős, G., Kardos, C., Kemény, Z., Kovács, A. & Váncza, J. (2016). Process planning and offline programming for robotic remote laser welding systems. *International Journal of Computer Integrated Manufacturing*, 29(12), 1287–1306.

FarzanehKaloorazi, M. H., Bonev, I. A. & Birglen, L. (2018a). Parameters identification of the path placement optimization problem for a redundant coordinated robotic workcell. *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. V05BT07A087–V05BT07A087.

FarzanehKaloorazi, M., Masouleh, M. T. & Caro, S. (2014). Collision-free workspace of 3-RPR planar parallel mechanism via interval analysis. In *Advances in Robot Kinematics* (pp. 327–334). Springer.

FarzanehKaloorazi, M., Masouleh, M. T. & Caro, S. (2017). Collision-free workspace of parallel mechanisms based on an interval analysis approach. *Robotica*, 35(8), 1747–1760.

FarzanehKaloorazi, M., Bonev, I. A. & Birglen, L. (2018b). Simultaneous path placement and trajectory planning optimization for a redundant coordinated robotic workcell. *Mechanism and Machine Theory*, 130, 346–362.

Field, G. & Stepanenko, Y. (1996). Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators. *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, 3, 2755–2760.

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.

Gan, Y., Dai, X. & Li, D. (2013). Off-line programming techniques for multirobot cooperation system. *International Journal of Advanced Robotic Systems*, 10(7), 282.

Gao, F., Li, W., Zhao, X., Jin, Z. & Zhao, H. (2002). New kinematic structures for 2-, 3-, 4-, and 5-DOF parallel manipulator designs. *Mechanism and machine theory*, 37(11), 1395–1411.

Gao, J., Pashkevich, A. & Caro, S. (2017a). Manipulator Motion Planning in Redundant Robotic System for Fiber Placement Process. In *New Trends in Mechanism and Machine Science* (pp. 243–252). Springer.

Gao, J., Pashkevich, A. & Caro, S. (2017b). Optimal Trajectories Generation in Robotic Fiber Placement Systems. *Materials Science and Engineering Conference Series*, 212(1), 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.

Gao, J., Pashkevich, A., Cicellini, M. & Caro, S. (2018a). Optimal Coordination of Robot Motions with Positioner and Linear Track in a Fiber Placement Workcell. *The 15th International Conference of Informatics in Control, Automation and Robotics (ICINCO 2018)*.

Gao, J., Pashkevich, A., Cicellini, M. & Caro, S. (2018b). Optimization of Robot and Positioner Motions in Manufacturing of High-pressure Composite Vessels. *IFAC-PapersOnLine*, 51(22), 44–49.

Hall, N. G., Kamoun, H. & Sriskandarajah, C. (1997). Scheduling in robotic cells: classification, two and three machine cells. *Operations Research*, 45(3), 421–439.

Hassan, M., Liu, D. & Xu, D. (2018). A Two-Stage Approach to Collaborative Fiber Placement through Coordination of Multiple Autonomous Industrial Robots. *Journal of Intelligent & Robotic Systems*, 1–19.

Hassan, R., Cohanim, B., De Weck, O. & Venter, G. (2005). A comparison of particle swarm optimization and the genetic algorithm. *46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, pp. 1897.

Hely, C. (2016). *Planification de trajectoires pour placement automatisé de fibres sur surfaces de géométries complexes*. (Ph.D. thesis, École Polytechnique de Montréal).

Hely, C., Birglen, L. & Xie, W.-F. (2017). Feasibility Study of Robotic Fibre Placement on Intersecting Multi-Axial Revolution Surfaces. *Robotics and Computer Integrated Manufacturing, in press, DOI*, 1016(10).

Hemmerle, J. S. & Prinz, F. B. (1991). Optimal path placement for kinematically redundant manipulators. *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 1234–1244.

Hirakawa, A. R. & Kawamura, A. (1997). Trajectory planning of redundant manipulators for minimum energy consumption without matrix inversion. *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, 3, 2415–2420.

Hollerbach, J. M. & Suh, K. C. (1987). Redundancy resolution of manipulators through torque optimization. *Robotics and Automation, IEEE Journal of*, 3(4), 308–316.

Huilian, F. (2010). Discrete particle swarm optimization for TSP based on neighborhood. *Journal of Computational Information Systems*, 6(10), 3407–3414.

Husty, M. L. (1996). An algorithm for solving the direct kinematics of general Stewart-Gough platforms. *Mechanism and Machine Theory*, 31(4), 365–379.

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(1), 66–81.

Jiang, Q. & Gosselin, C. M. (2009). Determination of the maximal singularity-free orientation workspace for the Gough–Stewart platform. *Mechanism and Machine Theory*, 44(6), 1281–1293.

Jiménez, P., Thomas, F. & Torras, C. (2001). 3D collision detection: a survey. *Computers & Graphics*, 25(2), 269–285.

Kaloorazi, M., Masouleh, M. T. & Caro, S. (2013). Interval-analysis-based determination of the singularity-free workspace of Gough-Stewart parallel robots. *Electrical Engineering (ICEE), 2013 21st Iranian Conference on*, pp. 1–6.

Kaloorazi, M.-H. F., Masouleh, M. T. & Caro, S. (2015). Determination of the maximal singularity-free workspace of 3-DOF parallel mechanisms with a constructive geometric approach. *Mechanism and Machine Theory*, 84, 25–36.

Kazerounian, K. & Nedungadi, A. (1988). Redundancy resolution of serial manipulators based on robot dynamics. *Mechanism and machine theory*, 23(4), 295–303.

Kenndy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 4, 1942–1948.

Kennedy, J. (1997). The particle swarm: social adaptation of knowledge. *Evolutionary Computation, 1997., IEEE International Conference on*, pp. 303–308.

Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning* (pp. 760–766). Springer.

Khorshidi, M., Soheilypour, M., Peyro, M., Atai, A. & Panahi, M. S. (2011). Optimal design of four-bar mechanisms using a hybrid multi-objective GA with adaptive local search. *Mechanism and Machine Theory*, 46(10), 1453–1465.

Kohli, D. & Hsu, M.-s. (1987). The Jacobian analysis of workspaces of mechanical manipulators. *Mechanism and Machine Theory*, 22(3), 265–275.

Kolakowska, E., Smith, S. F. & Kristiansen, M. (2014). Constraint optimization model of a scheduling problem for a robotic arm in automatic systems. *Robotics and Autonomous Systems*, 62(2), 267–280.

Kovács, A. (2016). Integrated task sequencing and path planning for robotic remote laser welding. *International Journal of Production Research*, 54(4), 1210–1224.

Kucuk, S. (2013). Energy minimization for 3-RRR fully planar parallel manipulator using particle swarm optimization. *Mechanism and Machine Theory*, 62, 129–149.

Lawler, E. L. (1985). The traveling salesman problem: a guided tour of combinatorial optimization. *Wiley-Interscience Series in Discrete Mathematics*.

MA, K. & ZHANG, B. (2007). The application of robot in the automated gluing system [J]. *Journal of North China Institute of Science and Technology*, 1, 47–52.

McCarville, D. A. (2009). Evolution of and projections for automated composite material placement equipment in the aerospace industry.

Merlet, J. P. (2006a). *Parallel Robots*. Springer.

Merlet, J.-P. (2006b). Jacobian, manipulability, condition number, and accuracy of parallel robots. *Journal of Mechanical Design*, 128(1), 199–206.

Mlynek, J., Petru, M. & Martinec, T. (2018). Optimization of Industrial Robot Trajectory in Composite Production. *2018 18th International Conference on Mechatronics-Mechatronika (ME)*, pp. 1–6.

Najjari, M. & Guilbault, R. (2015). Formula derived from particle swarm optimization (PSO) for optimum design of cylindrical roller profile under EHL regime. *Mechanism and Machine Theory*, 90, 162–174.

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(3), 273–292.

Nenchev, D. N., Tsumaki, Y. & Takahashi, M. (2004). Singularity-consistent kinematic redundancy resolution for the SRS manipulator. *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 4, 3607–3612.

Nokleby, S., Fisher, R., Podhorodeski, R. & Firmani, F. (2005). Force capabilities of redundantly-actuated parallel manipulators. *Mechanism and machine theory*, 40(5), 578–599.

Nubiola, A. (2011). *Calibration of a serial robot using a laser tracker*. (Ph.D. thesis, École de technologie supérieure).

Nzue, R.-M. A., Brethé, J.-F., Vasselin, E. & Lefebvre, D. (2013). Comparison of serial and parallel robot repeatability based on different performance criteria. *Mechanism and Machine Theory*, 61, 136–155.

Odili, J. B. & Mohmad Kahar, M. N. (2016). Solving the traveling Salesman's problem using the African Buffalo optimization. *Computational intelligence and neuroscience*, 2016, 3.

Pateloup, V., Duc, E. & Ray, P. (2004). Corner optimization for pocket machining. *International Journal of Machine Tools and Manufacture*, 44(12-13), 1343–1353.

Qiao, S., Liao, Q., Wei, S. & Su, H.-J. (2010). Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions. *Mechanism and Machine Theory*, 45(2), 193–199.

Rao, S. S. (2009). *Engineering optimization: theory and practice*. John Wiley & Sons.

Robin, V., Sabourin, L. & Gogu, G. (2011). Optimization of a robotized cell with redundant architecture. *Robotics and Computer-Integrated Manufacturing*, 27(1), 13–21.

Savsani, V., Rao, R. & Vakharia, D. (2010). Optimal weight design of a gear train using particle swarm optimization and simulated annealing algorithms. *Mechanism and machine theory*, 45(3), 531–541.

Shimizu, M., Kakuya, H., Yoon, W.-K., Kitagaki, K. & Kosuge, K. (2008). Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution. *Robotics, IEEE Transactions on*, 24(5), 1131–1142.

Shirinzadeh, B., Cassidy, G., Oetomo, D., Alici, G. & Ang Jr, M. H. (2007). Trajectory generation for open-contoured structures in robotic fibre placement. *Robotics and Computer-Integrated Manufacturing*, 23(4), 380–394.

Storn, R. & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4), 341–359.

Tabarah, E., Benhabib, B. & Fenton, R. G. (1994). Optimal motion coordination of two robots—a polynomial parameterization approach to trajectory resolution. *Journal of Field Robotics*, 11(7), 615–629.

Tian, L. & Collins, C. (2003). Motion planning for redundant manipulators using a floating point genetic algorithm. *Journal of Intelligent and Robotic Systems*, 38(3-4), 297–312.

Tisius, M., Pryor, M., Kapoor, C. & Tesar, D. (2009). An empirical approach to performance criteria for manipulation. *Journal of Mechanisms and Robotics*, 1(3), 031002.

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

Tubaileh, A., Hammad, I. & Kafafi, L. (2007). Robot cell planning. *ENG TECH*, 26, 250–253.

Ur-Rehman, R., Caro, S., Chablat, D. & Wenger, P. (2010). Multi-objective path placement optimization of parallel kinematics machines based on energy consumption, shaking forces and maximum actuator torques: Application to the Orthoglide. *Mechanism and Machine Theory*, 45(8), 1125–1141.

Valsamos, C., Wolniakowski, A., Miatliuk, K. & Moulianitis, V. (2018). Minimization of Joint Velocities During the Execution of a Robotic Task by a 6 DoF Articulated Manipulator. *International Conference on Robotics in Alpe-Adria Danube Region*, pp. 368–375.

Vicencio, K., Davis, B. & Gentilini, I. (2014). Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods. *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2985–2990.

Vosniakos, G.-C. & Matsas, E. (2010). Improving feasibility of robotic milling through robot placement optimisation. *Robotics and Computer-Integrated Manufacturing*, 26(5), 517–525.

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(4), 1–9.

Wang, K.-P., Huang, L., Zhou, C.-G. & Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *Machine Learning and Cybernetics, 2003 International Conference on*, 3, 1583–1585.

Wang, T.-D., Li, X. & Wang, X. (2006). *Simulated Evolution and Learning: 6th International Conference, SEAL 2006, Hefei, China, October 15-18, 2006, Proceedings*. Springer.

Wu, J., Wang, J., Wang, L. & Li, T. (2009). Dynamics and control of a planar 3-DOF parallel manipulator with actuation redundancy. *Mechanism and Machine Theory*, 44(4), 835–849.

Wu, L., Cui, K. & Chen, S.-B. (2000). Redundancy coordination of multiple robotic devices for welding through genetic algorithm. *Robotica*, 18(6), 669–676.

Yoshikawa, T. (1985). Manipulability and redundancy control of robotic mechanisms. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, 2, 1004–1009.

Yu, Y.-Q. (1987). Optimum shaking force and shaking moment balancing of the RSS'R spatial linkage. *Mechanism and Machine Theory*, 22(1), 39–45.

Zacharia, P. T. & Aspragathos, N. (2005). Optimal robot task scheduling based on genetic algorithms. *Robotics and Computer-Integrated Manufacturing*, 21(1), 67–79.

Zacharia, P. T. & Aspragathos, N. A. (2004). Optimization of industrial manipulators cycle time based on genetic algorithms. *Industrial Informatics, 2004. INDIN'04. 2004 2nd IEEE International Conference on*, pp. 517–522.

Zhang, D. & Lei, J. (2011). Kinematic analysis of a novel 3-DOF actuation redundant parallel manipulator using artificial intelligence approach. *Robotics and Computer-Integrated Manufacturing*, 27(1), 157–163.

Zhang, J. & Li, A.-P. (2009). Genetic algorithm for robot workcell layout problem. *Software Engineering, 2009. WCSE'09. WRI World Congress on*, 4, 460–464.

Zhang, X., Xie, W.-F. & Hoa, S. V. (2014). Modeling and Workspace Analysis of Collaborative Advanced Fiber Placement Machine. *Proceedings of the ASME 2014 International Mechanical Engineering Congress and Exposition*.

Zhang, Y., Wang, S. & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015.

Zlatanov, D., Fenton, R. & Benhabib, B. (1998). Identification and classification of the singular configurations of mechanisms. *Mechanism and Machine Theory*, 33(6), 743–760.