

UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dipartimento di Ingegneria e Architettura

Dottorato di ricerca in Ingegneria Industriale

Ciclo XXXVI

Motion planning and control of an automated sorting system for laser machine

Theory and practice

Coordinatore:
Chiar.mo Prof. Gianni Royer Carfagni

Dottorando:
Marco Riboli

Tutore:
Chiar.mo Prof. Marco Silvestri

Thesis submitted in 2024

Table of contents

Preface **v**

1 Introduction **1**
1.1 Collision-free motion planning techniques 1
1.2 Sorting system for laser machines: description of robot kinematic 3
1.3 Project motivation 6

2 Univariate B-splines: a general overview **11**
2.1 Knot insertion and removal 12
2.2 Degree elevation and reduction 13
2.3 Junction and Splitting 14
2.4 Differentiation and integration 16
2.5 Sum of two 1-D splines 17
2.6 Splines curve fitting 17
2.7 Channel problem definition 19
2.7.1 Tight linear envelopes for splines 19
2.7.2 Smooth paths in a polygonal channel 20
2.7.3 Initial knots sequence and iterative refinement 27

3 Geometrical path modelling **31**
3.1 Single end-effector path 32
3.2 Dual-arm gantry system path 38
3.2.1 7-d.o.f. dual-arm Cartesian robot path planning 38
3.2.2 Collision detection solution 43
3.2.3 Frame sampling strategy 46
3.2.4 Polygonal channel definition 49
3.2.5 Collision avoidance strategy 52
3.3 Dual-gantry system path 55
3.3.1 14-d.o.f. dual-gantry system path planning 55
3.3.2 Roto-traslation solution 60

4 Motion profile calculation **63**
4.1 Iso-Parametric Trajectory Planning 63
4.2 Solution adopted 63

5 Implementation **75**
5.1 Software architecture 75
5.2 Control design 76
5.2.1 Real-time set-points generator 76
5.2.2 Adaptive band-wise linear least squares friction model 78

6 Result and discussion **83**
6.1 Sorting task with freed heads 83
6.2 A general sorting task 91

7 Conclusion **99**

A Channel problem numerical benchmark **101**

B Job macro list example **105**

Bibliography **107**

Preface

This thesis delves into a novel and scalable approach for motion planning in dual-gantry kinematics. Each gantry system has two arms, and this method effectively addresses the self-collision issue while ensuring G^2-continuity and minimizing curvatures. Starting from collision-free start and target positions, the proposed technique defines the geometric master path using classical five-segment planar trajectory, described by fifth degree spline. Using a path scaling technique, the multidimensional spline defining the geometric path of the four end-effectors is then generated from the master path. The rototranslation of the loaded parts is evaluated in frames to establish sampled overlapping curves. These vectors are then used to compute splines that modify previously defined geometric paths to avoid collisions. In this process, a quadratic programming minimization is employed to reduce the curvature of these splines while preserving the kinematic properties of the original path. Isoparametric trajectory planning ensures compliance with kinematic axis constraints, resulting in an optimized motion profile to reduce task execution time. Finally, this approach is applied to a sorting system for a laser machine (LST) with a maximum of fourteen degrees of freedom. Two case studies for a single gantry configuration are presented and discussed to illustrate and clarify the methodology in typical sorting operations. The choice of detailing these case studies is justified as that configuration is the most widely used in this industrial field. The first of these applications involves a sequence of movements in which the anti-collision procedure is not used and corresponds to at least half of the machine's movements. In fact, these movements correspond to all those performed with unloaded end-effector tools and those in which the loaded part is in the shadow of the arm. Afterwards, some results regarding this case study are proposed. They are obtained by comparing the new method with the existing one in terms of overall performance. A more complex case study, in which the anti-collision algorithm is used, is then examined in depth. It's worth noting that the approach outlined in this work can be extended to other robotic systems with similar kinematic structures, providing a valuable tool for motion planning in pick-and-place applications.

The mathematical notations and abbreviations used in the remainder of this thesis are summarized in the following page.

Notation

$=$ equal in the equations and equality operator in the algorithmic sections.
$:=$ equal by definition.
$\leftarrow$ assignment in algorithmic sections.
$\wedge$ logical and.
$\vee$ logical or.
$\boldsymbol{\tau}$, $(\tau_i)_{i=0}^{n-1}$,$\{\tau_0, \ldots, \tau_{n-1}\}$ various ways of describing the same n-vector.
$\#\boldsymbol{\tau}$ number of elements of the $\boldsymbol{\tau}$ vector.
$|x|$ absolute value of x.
$\|\boldsymbol{P}\|$ magnitude of vector $\boldsymbol{P}$.
$\mathbf{0}$ null vector.
$\mathcal{L}(x \mid f(x_1), f(x_2))$, $\mathcal{L}(x \mid \{x_1, y_1\}, \{x_2, y_2\})$ linear interpolation of the function $f(x) = y$ between points $\{x_1, f(x_1)\}$ and $\{x_2, f(x_2)\}$.
$\odot$ element-wise product.
$(f \circ g)(x)$ functions composition, defined by $f(g(x))$.
$\text{sgn}(x)$ sign function, defined by $x/|x|$.
$\mathcal{F}(\{x, y\} \mid \{x_1, y_1\}, \{x_2, y_2\})$ direction of the cross product between vectors $\{x_2 - x_1, y_2 - y_1\}$ and $\{x - x_1, y - y_1\}$, defined by $\text{sgn}((x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1))$.
x mod y returns the remainder of the integer division x/y.
$\text{proj}(f(x), v)$ projection of the function $f(x)$ on the v-axis.

Acronyms

CAD := Computer Aided Design.
CAM := Computer Aided Machining.
p-form := Polynomial form.
pp-form := Piecewise polynomial form.
B-form := B-spline form.
BB-form := Bernstein-Bézier form.
LST := Sorting system for laser machine.
CSC := Compressed Sparse Column.
IPTP := Iso-parametric trajectory planning.
VC := Velocity constraint.
AC := Acceleration constraint.
RT := Ratio transmission.
RI := Rotor inertia.
LIMR := Load inertia moment ratio.
MC := Machine constant.
I_N := Nominal current.

Chapter 1

Introduction

1.1 Collision-free motion planning techniques

One of the most significant problems in industrial robotic applications is motion planning. It basically refers to the entire procedure that identifies the tasks that a robot must perform in order to move from its present state to a desired target state. Finding a collision-free path is often the first step in this procedure, which should guarantee a good compromise between the tool path's arc length and its smoothing features to produce effective movements. In addition to potential environmental obstacles, which are generally classified as static or dynamic depending on whether they are all fixed or not, there are an increasing number of industrial scenarios in which multiple manipulators (or multiple end-effectors of the same robot) move simultaneously in the working area.

This study focuses on the motion planning of a sorting system for laser machines (LST), where a maximum of four gripping tools can collaborate in the same workspace. In this situation, it is not possible to plan the movement taking into account each end-effector separately, but it is necessary to consider each end-effector simultaneously in order to avoid self-collision and maximise the performance of the entire system. As will be explained in more detail in the next section, the mechanical system adopted is designed to be easily scalable over the number of end-effectors (from one to four), and, in its most complete configuration, adopts two Cartesian dual-arm robots cooperating with each other. For this reason, a brief overview of the state of the art in dual-arm robotic systems and, more generally, of existing methods for preventing self-collision of multiple manipulators sharing the same workspace will be presented.

In this context, Lee et al. [1] present one of the first examples of collision-free motion planning for a dual-arm workcell. In that work, an event-based approach for online and offline collision-free trajectory planning for dual-arm assembly systems is proposed. This method is based on a collision-free trajectory planner on top of the collision detection algorithm introduced by the same authors in [2], and proposes an application to a workcell

consisting of two SCARA robots. A few years later, a new approach based on configuration space (C-Space) was successfully used in [3] for collision-free motion planning of a 2-D horizontal articulated double-arm robot (SCARATES robot). In general, the problem of collision-free motion planning for this type of system presents several solutions in the literature, which may be broadly categorised into three categories:

- sampling-based;
- optimisation-based;
- trajectory planning approaches.

Methods based on the Rapidly-exploring Random Trees (RRTs) and its variations are common sampling-based techniques in high-dimensional configuration spaces [4], where each dimension corresponds to a robot's degree of freedom (d.o.f.). The fundamental concept involves the stochastic sampling of the configuration space, culminating in the creation of a tree-like architectural framework interlinking these samples. Commencing its operation, the algorithm initializes with a sole sample within the joint space, representing the initial state and forming the root of this arborescent structure. Sequentially, new samples are incorporated by iteratively choosing random states within the configuration space. Subsequently, the nearest pre-existing sample is identified and, when feasible, a novel branch within the tree is established. This process persists until a trajectory connecting the initial and target positions is discovered. An example of using the RRT algorithm for solving the pick-and-place motion planning problem of a dual-arm robot is proposed by Kurosu et al. [5]. Subsequently, Shi et al. [6], address the same problem with a variant of this algorithm called GA_RTT. This new algorithm merges the probabilistic bias approach and the A* algorithm (a search algorithm obtained by adding a heuristic function to Dijkstra's algorithm [7]) with RRT, surpassing the performance of the initial algorithm. Even though sampling-based methodologies discover numerous applications within the field of motion planning, they might demand a growing multitude of samples as the dimensionality of the configuration space expands. This requisite is essential to ensure a precise representation of the configuration space dynamics, as emphasized in earlier studies [8]. The concept behind this phenomenon, known as the curse of dimensionality (COD), was first introduced by Richard Bellman in the area of approximation theory [9]. It follows that for high number d.o.f. systems, sampling-based methods can lead to high calculation times, and in the worst cases are incompatible with run-time applications. For the specific machine investigated in this study, the high number of d.o.f. (14 in the dual-gantry configuration) prompted us to omit the utilization of these techniques. Instead, we opted for a hybrid method using the techniques introduced below.

Optimization-based approaches present an alternative solution to address collision free motion planning. These approaches calculate a collision-free path by minimising a cost function, which could be actuation time, energy consumed or geometric curvature. The interesting aspect of these methods lies in their computational efficiency, since the problem is structured by a convex cost function and linear or quadratic constraints. However,

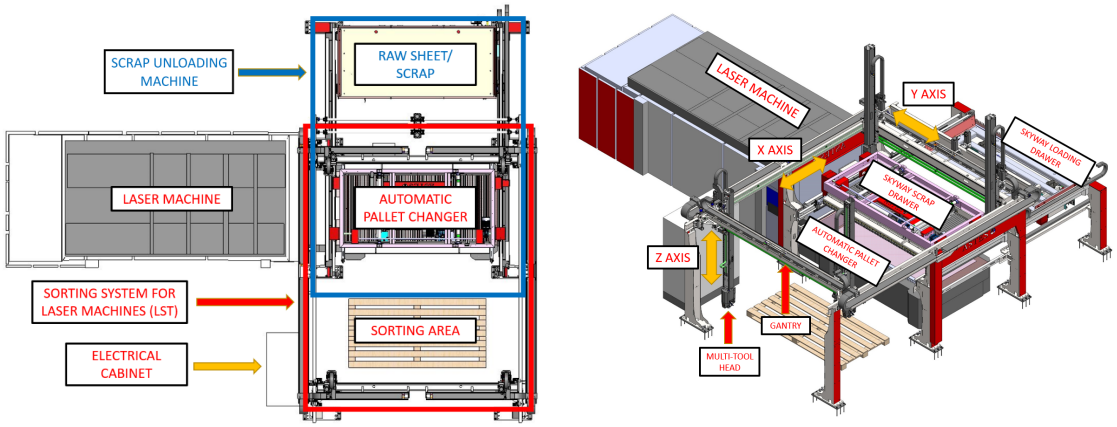
the aforementioned factors do not always readily translate into practical applications. Additionally, iterative methods for nonlinear constrained optimization, such as sequential quadratic programming (SQP), typically entail computation times that are incompatible with real-time robotic applications [10]. An application of an optimisation-based algorithm used for collision-free motion planning of a dual-arm robot is recently proposed by Völz and Graichen [11]. In this application, an object must be moved using both arms of the robot, thereby defining a closed kinematic chain. It is brought to the reader's attention that solving such constrained problems using sampling-based planners would be a daunting task, mainly due to the extremely low probability of a randomly generated sample satisfying the closure constraint.

Trajectory planning methods that utilize splines have also been proposed for achieving smooth motion planning in robotics. These approaches traditionally employ splines to represent the geometric path, optimizing its shape through computer-aided design (CAD) techniques to identify a collision-free trajectory. For instance, Chen and Li [12] introduced a method for generating collision-free paths for two industrial robots operating within a shared workspace. They applied B-spline knot refinement [13, 14] and a local modification scheme. This method leverages the local properties of B-splines, making adjustments to the trajectory only in areas where collisions are detected, while keeping the remaining segments of the curve in their original positions without modification. Another noteworthy application of B-spline theory in motion planning for a dual-arm robot is presented by Choi et al. [15]. In this study, a post-processing technique is proposed for enhancing the quality of a previously calculated geometric path (e.g., obtained through a sampling-based approach). This technique focuses on smoothing the trajectory and preventing collisions.

This thesis deals with the collision-free motion planning of a double gantry system with four arms working together in the same working area (14-d.o.f.), and represents an extension of the research presented in [16]. The contribution of this work is to provide a general approach for smooth and collision-free motion planning valid for this type of robot, which can be categorised as a hybrid technique based on optimisation and spline modelling. In the next section, the kinematics of the robot are explained and some information about the specific application are given.

1.2 Sorting system for laser machines: description of robot kinematic

Fig. 1.1 shows a classic layout of the sorting system for laser machine (LST) subject of this study. The sorting operation consists of taking the parts processed by the laser machine, nested in the sheet metal in the loading zone, and rearranging them into stacks within the unloading zone. Referring to Fig. 1.1a, the laser machine ejects the cut sheet onto the automatic pallet changer to allow the machined parts to be sorted onto pallets in the sorting



(a) (b)

Figure 1.1: Schematic of the laser machine sorting system (LST). (a) 2-D layout of work areas. (b) Perspective view of the machine.

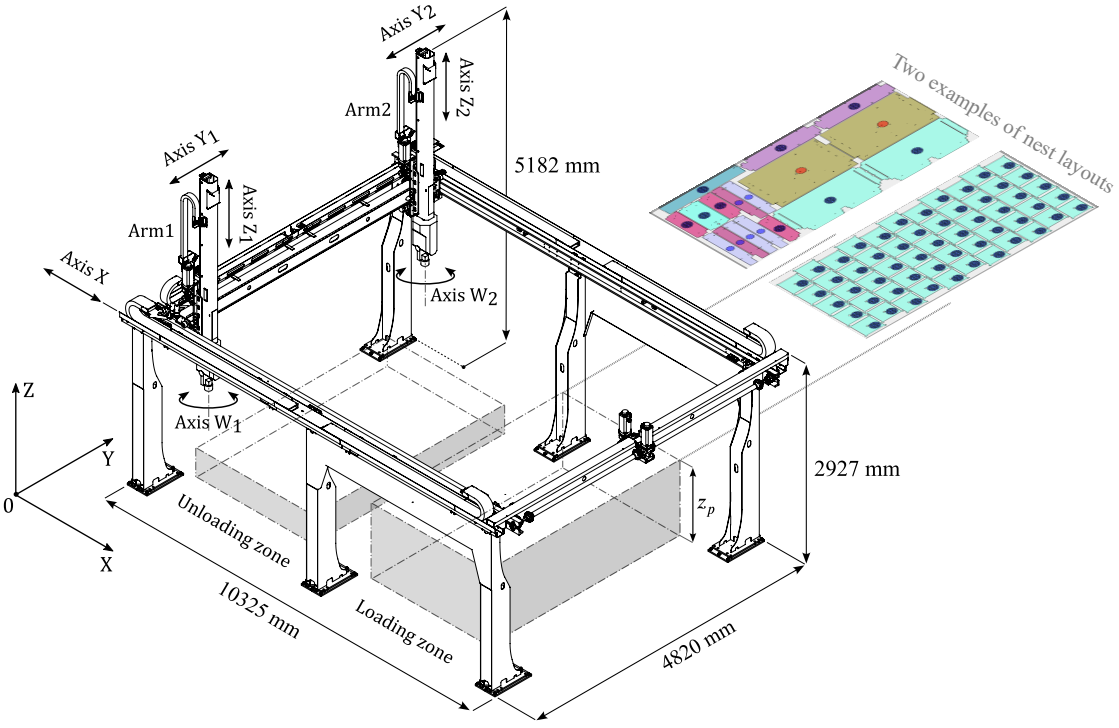


Figure 1.2: Sorting system for laser machine (LST) with dual-arm Cartesian kinematics. The size parameters are related to the commercially available robot SortPro. Further technical data on this system can be found in [17].

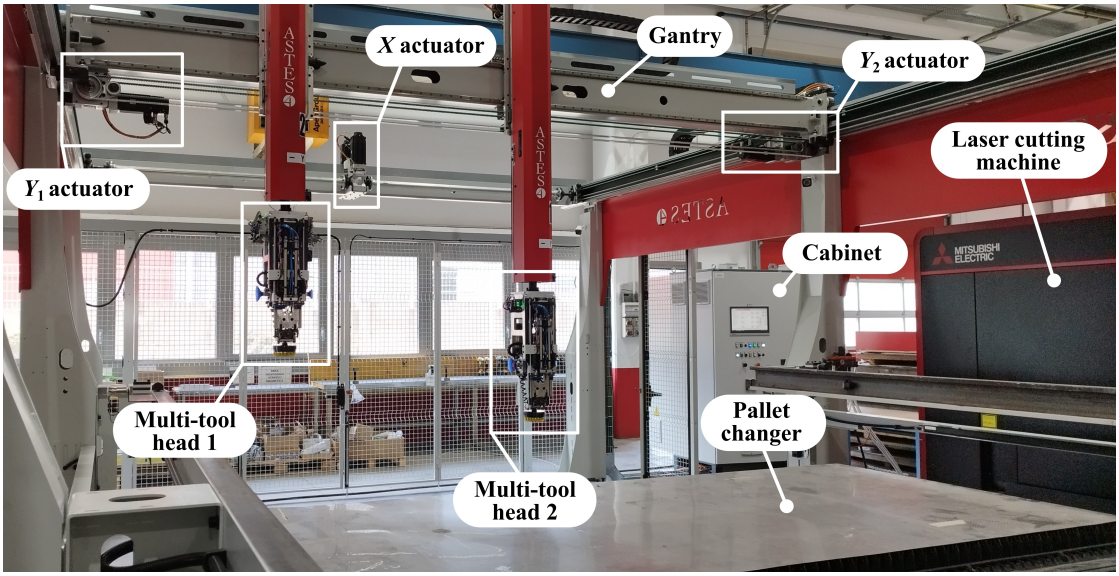


Figure 1.3: Photo of the LST system with dual-arm Cartesian kinematics.

area. Pallets are usually placed on automated drawers, which can enter and leave the sorting area (not represented in the picture). At the end of the process the scrap is unloaded by an automated machine, and the pallet changer is prepared to accept a new machined sheet metal. The kinematics adopted, shown in Fig. 1.1b, is a Cartesian dual gantry system. Each gantry mounts two arms, for a total of four gripping tools that can cooperate with each other. Gripping tools are allowed to rotate around the vertical axis for angular positioning of the sorted parts. Depending on the material and shape of the machined parts, the gripping tool can be a magnet, a suction cup or a multiple suction cup. In this regard, the machine studied in this work adopts multi-tool heads [18] able to adapt the picking to each situation by changing the tool on the fly and allowing a choice between several types of tools (or a combination of them), which differ in technology or size. Excluding automatic drawers, scrap unloading machine and multi-tool head motors, the system has a total of fourteen d.o.f., i.e. seven for each gantry system.

The system kinematics for a single gantry configuration are better detailed in Fig. 1.2. The solution for robot kinematics is a dual-arm Cartesian kinematics, where two arms are rigidly linked on the same gantry along the X-axis of the machine. Each end-effector mounted on the two arms has a three more d.o.f. defined by Y_i, Z_i, and W_i-axis, with $i = 1, 2$. This type of solution, although it may seem arborescent at first glance, offers several advantages in this specific application. The first is greater efficiency due to the great variability in the size and weight of the parts, which also depends on the material and thickness of the sheet metal. Small and light parts can be loaded by a single end-effector and sorted in parallel, while heavier and larger parts can be transported by several end-effectors simultaneously with multiple loading. In this latter case, moreover, the inflection

of the part is classically reduced compared to the loading of the same part that would occur with a single gripping tool, which improves gripping stability. From this point of view, a Cartesian system can offer advantages in terms of work distribution between the arms, when compared, for example, with the use of two articulated robots (as already proven in [19]), resulting in shorter sorting times.

For the sake of completeness, a photo of the LST system with Cartesian dual-arm kinematics is shown in Fig. 1.3.

1.3 Project motivation

The introduction of an automatic LST system in a workshop is a complex process that requires consideration of the entire workflow. The cutting time of the laser machine varies considerably depending on the characteristics of the processed parts. For example, a nest containing numerous parts that have a large number of holes takes a relatively long time to process, and the automation does not lead to cycle time concerns. In this case, the sorting operation should be slowed down in order to promote energy savings in the plant. In other cases, however, the cutting time can be significantly reduced and the sorting time soon becomes the bottleneck of the production process. A decrease in sorting time is therefore followed by a throughput maximization of the entire production process, hence the interest of LST system manufacturers in optimizing the performaces. From this point of view, an appropriate motion planning strategy allows to decrease the robot's point-to-point positioning time, helping to improve process efficiency without burdening the production costs of these machines.

The layouts of the parts in the nest can be extremely different from each other. Three examples of layouts are reported in Fig. 1.5. In series production there are usually many parts of the same type in a layout (e.g. Fig. 1.5a), while in workshops with low batch sizes, layouts present many different types of parts [20], like the examples shown in Fig. 1.5b and Fig. 1.5c. Nested parts can have different sizes, from small parts that reside in the shadow of the end-effector, to parts with a shape that exceeds the section of the arm on which the gripping tool is installed. The presence of holes within the part shape may also require off-centre loading, in particular when these are performed using suction cups. It is important to emphasize that during the sorting operation, these parts may have to be re-oriented in the unloading zone, as the nesting algorithms that configure the sheet metal layout may allow free rotation of the parts [21], or even just because a different storage orientation is desired. In order to prevent collisions during movement, it is therefore necessary to consider the entire area covered by the end-effector and loaded part in their respective angular orientation. These observations lead to the conclusion that a strategy based on the partitioning of work zones, of the type used in [19], is not sufficient to ensure a collision-free path. Self-collision is therefore a crucial issue in these systems, and an appropriate motion planning strategy must be adopted. However, despite the geometrical constraints dictated by col-

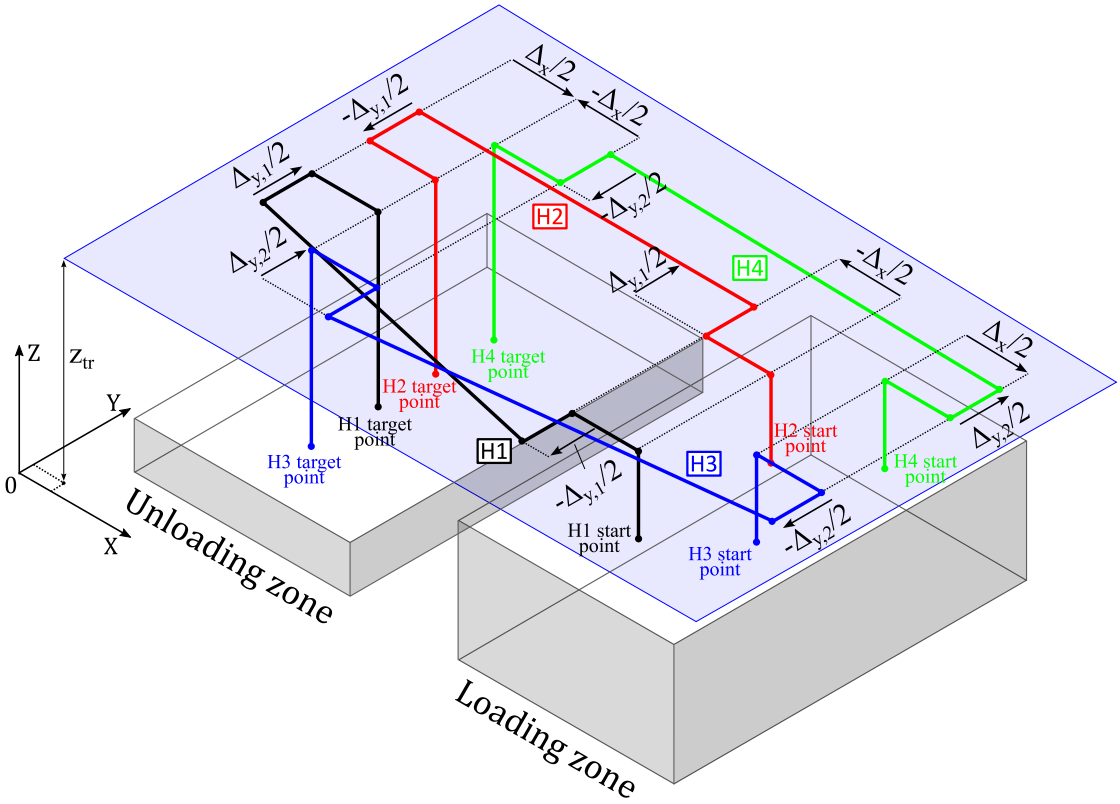
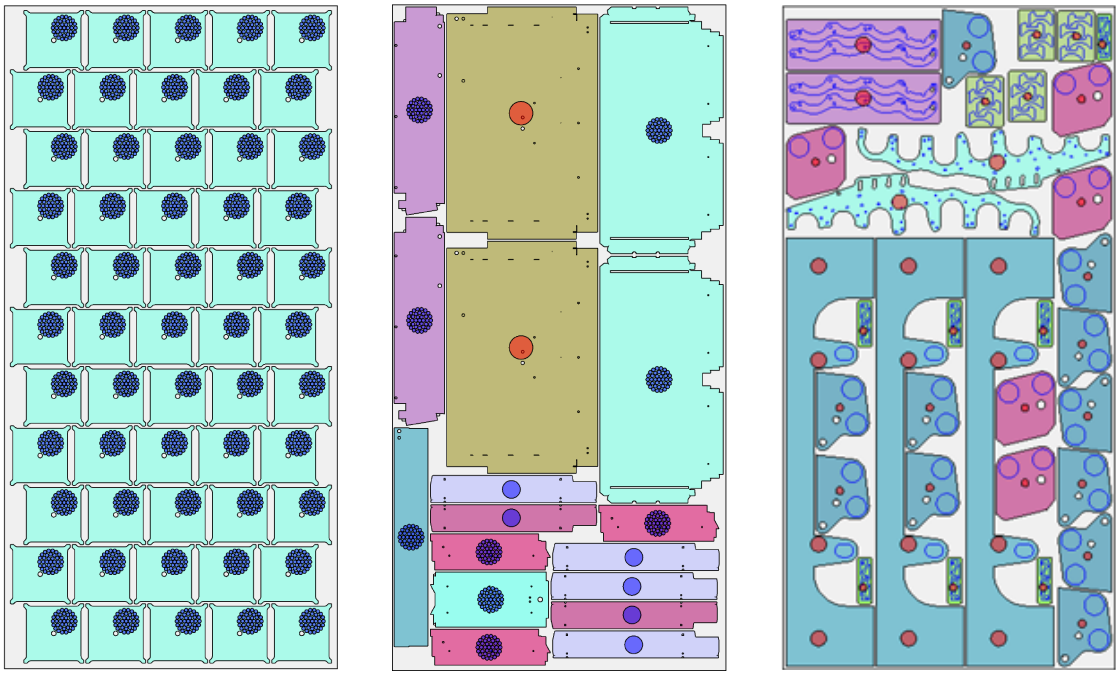


Figure 1.4: Collision avoidance strategy previously used for dual gantry system configuration.

lision avoidance, the smoothness of the trajectory path cannot be ignored in the motion planning strategy to ensure good quality of the motion.

In industry, the self-collision problem of Cartesian dual-arm kinematics, and more generally of dual-gantry systems, is classically handled with simple strategies that limit the robot's movements to classical rectangular trajectories. In this regard, Fig. 1.4 shows an example of a solution currently used to avoid collisions in this type of system. The figure corresponds to the case of a parallel in-line unload of four parts, each loaded by a different end-effector. With reference only to the plane parallel to the XY-plane and placed at a travel height z_{tr}, this strategy can, in a simplified way, be seen in the following consecutive operations:

1. adopting an appropriate geometric criterion, calculate the minimum distance Δ_x of the two gantries in order to avoid collisions along the X-direction;
2. for both sets of start and target points, calculate intermediate positions by translation along the X-direction, with appropriate sign, of the semi-distance $\Delta_x/2$;
3. adopting an appropriate geometric criterion, calculate, for each gantry, the minimum distances $\Delta_{y,1}$ and $\Delta_{y,2}$ of the two arms to avoid collisions along the Y-direction;



(a) (b) (c)

Figure 1.5: Three example of nesting layouts.

4. similarly to step 2, calculate new intermediate points obtained from the previous ones by translation along the Y-direction, with appropriate sign, of the semi-distances $\Delta_{y,1}/2$ and $\Delta_{y,2}/2$;
5. the collision-free geometric path is provided for each end-effector by the five-segment polyline connecting the start point with the calculated intermediate positions and the target.

In addition to this, some considerations must be added regarding the constraints imposed by the limits of the working area, which will result in asymmetries in these opening movements.

This approach results in a maximum of six stopping points in the pick-and-place trajectory (depending on the specific configuration of start and target points), resulting in suboptimal machine performance in terms of cycle time. Each of these intermediate points also corresponds to a deceleration phase and a subsequent acceleration phase of the motion profile that follows the geometric path. Consequently, these phases result in dynamic stresses that impose loads on the tool-part contact, thereby elevating the risk of operational failure due to detachment of the loaded part. In this work, the problem of motion optimisation is addressed in order to optimise the performance of the system in terms of cycle time and trajectory smoothness.

Before proceeding to the other sections, please note that this work has the following

limitations:

- Dynamic constraints introduced by gripping tools, as well as the deflection of loaded parts, will not be dealt with in this work. The high complexity of the problem due to the high variability of boundary conditions, such as tool type and size, the number of end-effectors involved in gripping, and the thickness of the sheet, needs studies tailored to the specific application case and will be the subject of future work.
- This work is limited to solve the problem of collision-free motion planning, given start and target points. These positions should be determined to ensure the feasibility of point-to-point movement from the end-effectors, and to minimize the overall displacements required to sort all the nest. This process classically requires several iterations, which involve the execution of the point-to-point motion planning algorithm. It follows that the computational efficiency of motion planning is a priority to ensure adequate computational performance for the determination of the load-unload order of the nest parts, and the calculation of start and target positions. However, this topic is to be considered beyond the scope of this work, and no further details will be provided.
- This research had, as a fundamental premise, that of define a motion planning approach of the machine by intervening exclusively on the software program, without introducing any modification either to the mechanical components or to the control hardware.

Chapter 2

Univariate B-splines: a general overview

Since the proposed approach is based on splines in their B-form (B-spline form), a brief description of these parametric curves and the basic algorithms used to model the shape of the geometric path and motion profile is provided.

A spline of degree d is a piecewise polynomial vector function $\boldsymbol{r}(\tau) : \mathbb{R} \to \mathbb{R}^k$ in the parametric variable τ, defined, in its B-form, by the linear combination:

$$\boldsymbol{r}(\tau) = \sum_{j=0}^{m} B_{j,d,\boldsymbol{u}}(\tau)\boldsymbol{b}_j \ , \quad u_s \leq \tau \leq u_e \tag{2.1}$$

with $\boldsymbol{b}_j = (b_{j,i})_{i=0}^{k}$ called control points. In the case of $k = 1$, i.e. a 1-D spline, the control points are often simply called coefficients. In this case, in the remainder of this paper, we will refer to their scalar form b_j. The basis function $B_{j,d,\boldsymbol{u}}(\tau)$ are defined by Cox-de Boor recursion formula [22]:

$$\begin{cases} B_{j,0,\boldsymbol{u}}\left(\tau\right) = \begin{cases} 1, & \text{if } u_j \leq \tau < u_{j+1} \\ 0, & \text{otherwise} \end{cases} \\ B_{j,d,\boldsymbol{u}}\left(\tau\right) = \dfrac{\tau - u_j}{u_{j+d} - u_j} B_{j,d-1,\boldsymbol{u}} + \dfrac{u_{j+d+1} - \tau}{u_{j+d+1} - u_{j+1}} B_{j+1,d-1,\boldsymbol{u}} \end{cases} \tag{2.2}$$

based on a non-decreasing knot sequence $\boldsymbol{u} = (u_i)_{i=0}^{m+d+1}$. In this work the knots will be always non-periodic and non-uniform, so in the format:

$$\boldsymbol{u} = \{\underbrace{u_s, \ldots, u_s}_{d+1}, u_{d+1}, \ldots, u_m, \underbrace{u_e, \ldots, u_e}_{d+1}\} \tag{2.3}$$

In the special case where $n = d$, and for convenience setting $u_s = 0$ and $u_e = 1$, the knot vector becomes:

$$\boldsymbol{u} = \{\underbrace{0, \ldots, 0}_{d+1}, \underbrace{1, \ldots, 1}_{d+1}\} \tag{2.4}$$

and the curve degenerates into the BB-form (Bernstein-Bézier form):

$$\boldsymbol{r}'(\tau) = \sum_{j=0}^{d} B'_{j,d}(\tau)\boldsymbol{b}_j\ , \quad 0 \leq \tau \leq 1 \tag{2.5}$$

where the basis functions $B'_{i,d}$ are the classical d^{th}-degree Bernstein polynomial:

$$B'_{j,d}(\tau) \coloneqq \binom{d}{j}\tau^j\,(1-\tau)^{d-j} = \frac{d!}{j!\,(d-j)!}\tau^j\,(1-\tau)^{d-j} \tag{2.6}$$

2.1 Knot insertion and removal

The knot insertion procedure starts from a known spline defined by (2.1) to calculate the control points of a new spline with the same shape and knots $\bar{\boldsymbol{u}}$, obtained from $\boldsymbol{u}$ by sorted insertion (increasing) of the knot $\bar{\tau}$ with multiplicity r. Let $\bar{\tau} \in [u_k, u_{k+1})$ be a knot of initial multiplicity s in $\boldsymbol{u}$, and suppose it is to be inserted r times with $r + s \leq d$, the new control points $\bar{\boldsymbol{b}}$ are [14]:

$$\bar{\boldsymbol{b}}_i^r = \alpha_i^r \boldsymbol{b}_i^{r-1} + (1-\alpha_i^r)\boldsymbol{b}_{i-1}^{r-1} \tag{2.7}$$

with

$$\alpha_i^r = \begin{cases} 1, & i \leq k-d+r+1 \\ \dfrac{\bar{\tau}-u_i}{u_{i+d-r+1}-u_i}, & k-d+r \leq i \leq k-s \\ 0, & i \geq k-s+1 \end{cases}$$

When there's a need to insert numerous knots simultaneously, invoking the knot insertion algorithm multiple times proves inefficient. Instead, we employ a more efficient approach known as knot refinement. This algorithm enhances performance without necessitating further mathematical additions to (2.7). As it is only a software problem, it is not presented in this work and is referred to [13].

Reverse operation of knot insertion is called knot removal. Let u_k be an interior knot of multiplicity s in $\boldsymbol{u}$. The general equations for computing the new control points $\bar{\boldsymbol{b}}$ for r removal of u_k are ([14, 23]):

$$\begin{aligned} \bar{\boldsymbol{b}}_i^r &= \frac{\boldsymbol{b}_i^{r-1} - (1-\alpha_i)\boldsymbol{b}_{i-1}^r}{\alpha_i}, \quad k-d-r+1 \leq i \leq \tfrac{1}{2}(2k-d-s-r) \\ \bar{\boldsymbol{b}}_j^r &= \frac{\boldsymbol{b}_j^{r-1} - \alpha_j \cdot \boldsymbol{b}_{j+1}^r}{(1-\alpha_j)}, \quad \tfrac{1}{2}(2k-d-s+r+1) \leq j \leq k-s+r-1 \end{aligned} \tag{2.8}$$

with

$$\alpha_i = \frac{u_k - u_i}{u_{i+p+r} - u_i}, \quad \alpha_j = \frac{u_k - u_{j-r+1}}{u_{j+p+1} - u_{j-r+1}}$$

However, it is not generally known in advance if a knot is removable without changing the shape of the curve and if it is how many times. For this reason, a tolerance value is generally used. For further details see [23].

In the case it is necessary to remove as many knots as possible, the knot removal algorithm can be called for each internal knot, but this is inefficient. In order to solve the problem, Tiller [23] suggests an optimized routine to minimizes the number of shift of coefficients and knots, which is used in our implementation.

2.2 Degree elevation and reduction

Let $\boldsymbol{r}(\tau)$ defined by (2.1) with knots:

$$\boldsymbol{u} = \{\underbrace{u_s, \ldots, u_s}_{d+1}, \underbrace{u_1, \ldots, u_1}_{m_1}, \ldots, \underbrace{u_{\tilde{s}}, \ldots, u_{\tilde{s}}}_{m_{\tilde{s}}}, \underbrace{u_e, \ldots, u_e}_{d+1}\} \tag{2.9}$$

where $\tilde{s}$ is the number of distint internal knots $\tilde{\boldsymbol{u}}$ and the coefficients $m_1, \ldots, m_{\tilde{s}}$ are their multiplicities. The term degree elevation indicates the operation that defines knots and control points of a new spline ${}^1\boldsymbol{r}(\tau)$ of degree $d+1$ with the same shape of $\boldsymbol{r}(\tau)$. In the syntax used in this work ${}^k\boldsymbol{r}(\tau)$ is the spline $\boldsymbol{r}(\tau)$ raised by degree k times. The knots ${}^1\boldsymbol{u}$ of ${}^1\boldsymbol{r}(\tau)$ can be written as:

$${}^1\boldsymbol{u} = \{\underbrace{u_s, \ldots, u_s}_{d+2}, \underbrace{u_1, \ldots, u_1}_{m_1+1}, \ldots, \underbrace{u_{\tilde{s}}, \ldots, u_{\tilde{s}}}_{m_{\tilde{s}}+1}, \underbrace{u_e, \ldots, u_e}_{d+2}\} \tag{2.10}$$

with maximum knot index $\bar{m} = m + \tilde{s} + 1$. Moreover, (2.10) allows to keep the same continuity as $\boldsymbol{r}(\tau)$. In [24] an efficient algorithm to get the $\bar{m}+1$ coefficients is reported. This approach performs the following operations on-the-fly:

- extracts the i^{th} Bézier segment (see [14] for curve decomposition algorithm),
- increases its degree,
- eliminates unnecessary knots.

The calculation of the degree-elevated Bézier coefficients after r degree elevations can be performed with:

$${}^r\boldsymbol{b}_i = \sum_{j=\max(0,i-r)}^{\min(d,i)} \frac{\binom{d}{j}\binom{r}{i-j}}{\binom{d+r}{i}} \boldsymbol{b}_j \,, \quad i = 0, \ldots, d+r \tag{2.11}$$

While it is always possible to elevate the spline degree, a curve may not be reducible. Let ${}^1\boldsymbol{r}(\tau)$ be a d^{th}-degree spline with knots defined by (2.9). ${}^1\boldsymbol{r}(\tau)$ is degree reducible if it has an accurate representation [14]:

$${}^1\boldsymbol{r}(\tau) \equiv \boldsymbol{r}(\tau) = \sum_{i=0}^{m-\tilde{s}-1} B_{i,d-1,\boldsymbol{u}}(\tau)\boldsymbol{b}_i \tag{2.12}$$

with knots

$$\boldsymbol{u} = \{\underbrace{u_s, \ldots, u_s}_{d}, \underbrace{u_1, \ldots, u_1}_{m_1-1}, \ldots, \underbrace{u_{\tilde{s}}, \ldots, u_{\tilde{s}}}_{m_{\tilde{s}}-1}, \underbrace{u_e, \ldots, u_e}_{d}\} \tag{2.13}$$

The algorithm used for the degree reduction of B-splines follows the same three-step strategy used for degree elevation, where the degree elevation of the i^{th} Bézier curve is replaced with a degree reduction. The literature provides several degree reduction algorithms for Bézier curves [25, 26]. We adopt the algorithm proposed in [27]. Let

$$\begin{aligned} r &= (d-1) \operatorname{div} 2 \\ \alpha_i &= i/d \\ {}^1\boldsymbol{r}'(\tau) &= \sum_{i=0}^{d} B'_{i,d}(\tau)\, {}^1\boldsymbol{b}_i \\ \boldsymbol{r}'(\tau) &= \sum_{i=0}^{d-1} B'_{i,d-1}(\tau)\, \boldsymbol{b}_i \end{aligned}$$

then, if $p \bmod 2 = 0$ (even):

$$\begin{aligned} \boldsymbol{b}_0 &= {}^1\boldsymbol{b}_0, \quad \boldsymbol{b}_{d-1} = {}^1\boldsymbol{b}_d \\ \boldsymbol{b}_i &= \frac{{}^1\boldsymbol{b}_i - \alpha_i \boldsymbol{b}_{i-1}}{1-\alpha_i}, \quad \text{for } i = 1, \ldots, r \\ \boldsymbol{b}_i &= \frac{{}^1\boldsymbol{b}_{i+1} - (1-\alpha_{i+1})\boldsymbol{b}_{i+1}}{\alpha_{i+1}}, \quad \text{for } i = d-2, \ldots, r+1 \end{aligned} \tag{2.14}$$

else, if $p \bmod 2 = 1$ (odd):

$$\begin{aligned} \boldsymbol{b}_0 &= {}^1\boldsymbol{b}_0, \quad \boldsymbol{b}_{d-1} = {}^1\boldsymbol{b}_d \\ \boldsymbol{b}_i &= \frac{{}^1\boldsymbol{b}_i - \alpha_i \boldsymbol{b}_{i-1}}{1-\alpha_i}, \quad \text{for } i = 1, \ldots, r-1 \\ \boldsymbol{b}_r &= \frac{1}{2}\left(\frac{{}^1\boldsymbol{b}_r - \alpha_r \boldsymbol{b}_{r-1}}{1-\alpha_r} + \frac{{}^1\boldsymbol{b}_{r+1} - (1-\alpha_{r+1})\boldsymbol{b}_{r+1}}{\alpha_{r+1}}\right) \\ \boldsymbol{b}_i &= \frac{{}^1\boldsymbol{b}_{i+1} - (1-\alpha_{i+1})\boldsymbol{b}_{i+1}}{\alpha_{i+1}}, \quad \text{for } i = d-2, \ldots, r+1 \end{aligned} \tag{2.15}$$

2.3 Junction and Splitting

In this study, a spline modeling-based methodology for geometric path and timing law planning is adopted. Spline junction and slicing play a crucial role in the employed modeling procedures, and the specific calculation methods of these operation are detailed in this section.

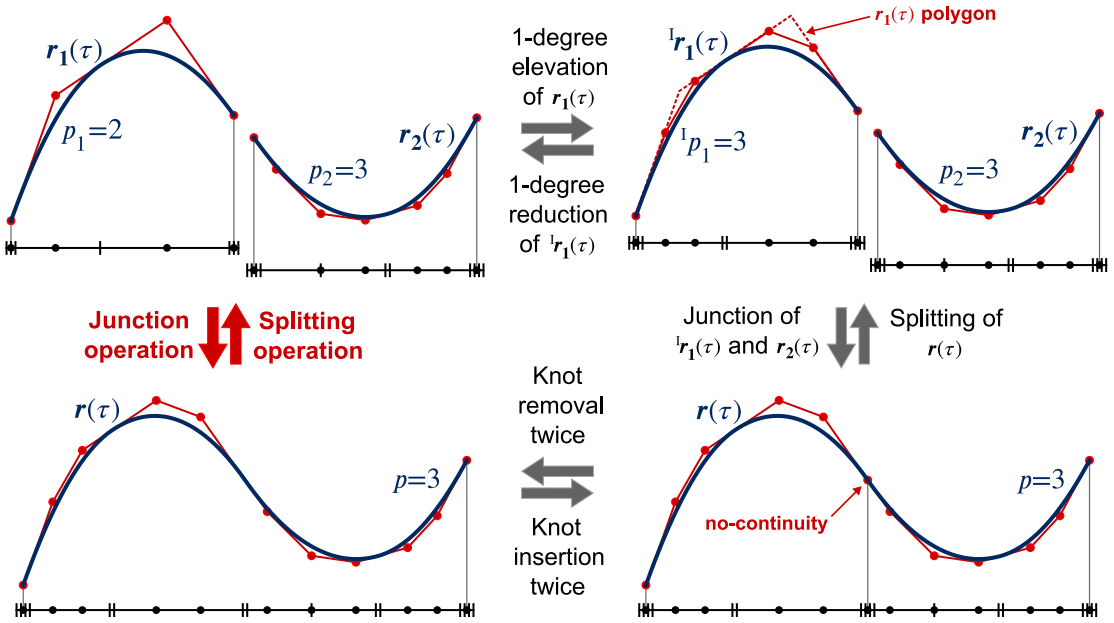


Figure 2.1: Generic pattern for 1-D spline junction and splitting (taken from [28]).

A general pattern for performing these operations, limited for simplicity to 1-D splines, is shown in Fig. 2.1. This pattern can be easily extended to generic multi-dimensional splines. Initially, the splines may not have the same degree; in this case, the curves must be made compatible. Since a spline (in its B-form) is not always reducible, in the more general case the lower degree curve is raised with (2.11) until it reaches the same degree of the higher degree spline. With reference to Fig. 2.1, $\boldsymbol{r_1}$ is elevated by one degree, obtaining the spline ${}^1\boldsymbol{r}_1$ of control points ${}^1\boldsymbol{b}^1 = \{{}^1b^1_0, \ldots, {}^1b^1_{m_1}\}$ and knots:

$$ {}^1\boldsymbol{u}_1 = \{\underbrace{{}^1u_{1,s}, \ldots, {}^1u_{1,s}}_{d+1}, {}^1u_{1,d+1}, \ldots, {}^1u_{1,m_1}, \underbrace{{}^1u_{1,e}, \ldots, {}^1u_{1,e}}_{d+1}\} $$

Let $\boldsymbol{r_2}$ be defined by control points $\boldsymbol{b^2} = \{b^2_0, \ldots, b^2_{m_2}\}$ and knots:

$$ \boldsymbol{u_2} = \{\underbrace{u_{2,s}, \ldots, u_{2,s}}_{d+1}, u_{2,\,d+1}, \ldots, u_{2,\,m_2}, \underbrace{u_{2,e}, \ldots, u_{2,e}}_{d+1}\} $$

By defining $\Delta u = {}^1u_{1,e} - u_{2,s}$, the spline $\boldsymbol{r}(\tau)$ given by the concatenation of ${}^1\boldsymbol{r}_1$ and $\boldsymbol{r_2}$ is defined by the knots:

$$ \boldsymbol{u} = \{\underbrace{{}^1u_{1,s}, \ldots, {}^1u_{1,s}}_{d+1}, {}^1u_{1,d+1}, \ldots, {}^1u_{1,m_1}, \underbrace{{}^1u_{1,e}, \ldots, {}^1u_{1,e}}_{d+1}, \ldots, $$
$$ (u_{2,d+1}+\Delta u), \ldots, (u_{2,m_2}+\Delta u), \underbrace{(u_{2,e}+\Delta u), \ldots, (u_{2,e}+\Delta u)}_{d+1}\} $$

and control points $\boldsymbol{b} = \{{}^1b_0^1, \ldots, {}^1b_{m_1}^1, b_0^2, \ldots, b_{m_2}^2\}$, where in ${}^1u_{1,e}$ the spline is no-continuous. If ${}^1b_{m_1}^1 \equiv b_0^2$, then ${}^1u_{1,e}$ is removable at least once. In general, (2.8) are then used to remove the knot as many times as possible, obtaining the right continuity in the junction point.

Fig. 2.1 also shows how this pattern is invertible using (2.7) and degree reduction algorithms [27]. In general, to split a spline $\boldsymbol{r}(\tau)$ into knot $\bar{\tau} \in (u_s, u_e)$ with initial multiplicity s, $\bar{\tau}$ must be inserted $d + 1 - s$ times.

2.4 Differentiation and integration

It is worth mentioning the expressions for the derivative and antiderivative of a spline in its B-form with respect to the parametric variable τ, as these formulations are used extensively in the subsequent parts of this work. The r^{th} derivative of $\boldsymbol{r}(\tau)$ is computed by [22]:

$$D^{(r)}\boldsymbol{r}(\tau) = \sum_{i=0}^{m-r} B_{i,d-r,\boldsymbol{u}^{(r)}}(\tau)\boldsymbol{b}_i^{(r)} \tag{2.16}$$

with knots:

$$\boldsymbol{u}^{(r)} = \{\underbrace{u_s, \ldots, u_s}_{d-r+1}, u_{d+1}, \ldots, u_m, \underbrace{u_e, \ldots, u_e}_{d-r+1}\}$$

where the control points are defined from:

$$\boldsymbol{b}_i^{(r)} = \frac{d-r+1}{u_{i+d+r} - u_{i+r}} \left(\boldsymbol{b}_{i+1}^{(r-1)} - \boldsymbol{b}_i^{(r-1)}\right), \quad \text{with } r > 0$$

The antiderivative of $\boldsymbol{r}(\tau)$ is instead computed by [22]:

$$D^{(-1)}\boldsymbol{r}(\tau) = \sum_{i=0}^{m+1} B_{i,d+1,\boldsymbol{u}^{(-1)}}(\tau)\boldsymbol{b}_i^{(-1)} \tag{2.17}$$

with knots:

$$\boldsymbol{u}^{(-1)} = \{\underbrace{u_s, \ldots, u_s}_{d+2}, u_{d+1}, \ldots, u_m, \underbrace{u_e, \ldots, u_e}_{d+2}\}$$

and control points:

$$\boldsymbol{b}_i^{(-1)} = \sum_{j=0}^{i} \boldsymbol{b}_j \frac{u_{j+d+1} - u_j}{d+1}, \quad \text{for } i = 0, \ldots, m+1$$

Then, (2.17) is iterated r times to obtain the r^{th} antiderivative.

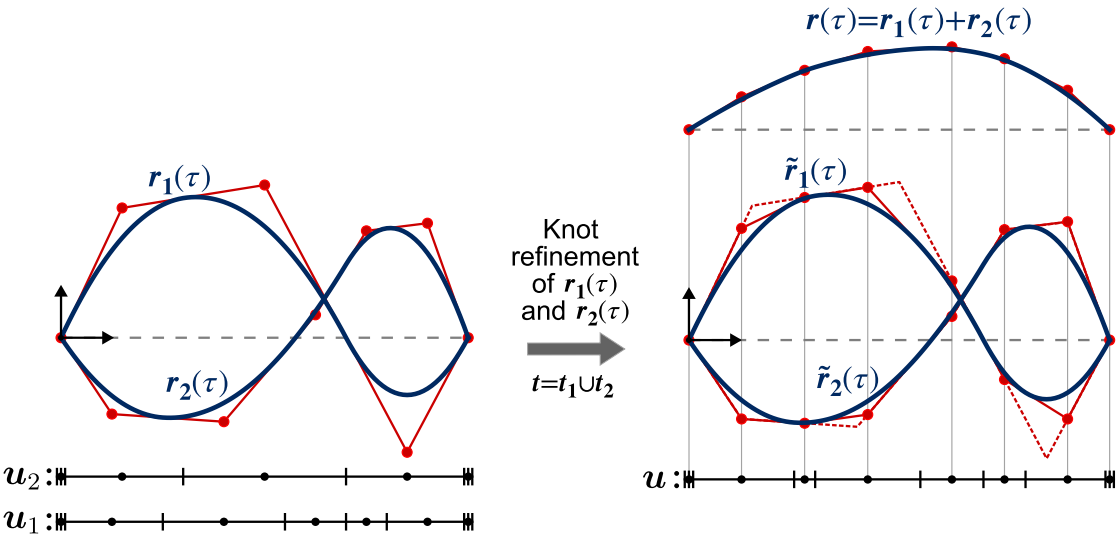


Figure 2.2: Sum operator between two 1-D splines. (taken from [28]).

2.5 Sum of two 1-D splines

Let $\boldsymbol{r_1}(\tau)$ and $\boldsymbol{r_2}(\tau)$ be two 1-D splines of degree $d_1 \equiv d_2 \equiv d$ and domain $\boldsymbol{u}_1 \equiv \boldsymbol{u}_2 \equiv \boldsymbol{u}$. From (2.1) the simple formula for adding the two curves is [29]:

$$\begin{aligned} \boldsymbol{r}(\tau) = \boldsymbol{r_1}(\tau) + \boldsymbol{r_2}(\tau) = \sum_{i=0}^{n} B_{i,d,\boldsymbol{u}}(\tau) b_i^1 + \sum_{i=0}^{n} B_{i,d,\boldsymbol{u}}(\tau) b_i^2 = \\ = \sum_{i=0}^{n} B_{i,d,\boldsymbol{u}}(\tau) \left(b_i^1 + b_i^2\right) \end{aligned} \tag{2.18}$$

The unknown coefficients $\boldsymbol{b}$ are defined by $b_i = b_i^1 + b_i^2$, and the problem is reduced to make the curves compatible. With reference to Fig. 2.2, assuming that $u_{s,1} \equiv u_{s,2}$ and $u_{e,1} \equiv u_{e,2}$, knot refinement and degree elevation algorithms are performed to make the curves compatible [30]. If $u_{s,1} \neq u_{s,2}$ or $u_{e,1} \neq u_{e,2}$, it is sufficient that $\boldsymbol{u}_1 \cap \boldsymbol{u}_2 \neq 0$ and it is always possible to go back to the previous case using splitting operations. When the operation is completed, the previously splitted curve pieces can be joined, and the problem is solved.

2.6 Splines curve fitting

Let $\boldsymbol{\tau} = (\tau_k)_{k=0}^n$ be a non-decrasing interpolation abscissas, with $\tau_i \leq \tau_{i+1}$, and $\boldsymbol{q} = (\boldsymbol{q}_k)_{k=0}^n$ be the interpolating multi-dimensional ordinates associated with $\boldsymbol{\tau}$. The coefficients $\boldsymbol{b}$ of the spline $\boldsymbol{r}(t)$ which interpolates the points $\{\tau_k, \boldsymbol{q}_k\}$ are computed by solving

the system:

$$\boldsymbol{q}_k = \boldsymbol{r}(\tau_k) = \sum_{i=0}^{n} B_{i,d,\boldsymbol{u}}(\tau_k)\boldsymbol{b}_i \tag{2.19}$$

The knots $\boldsymbol{u}$ must be chosen in order to fullfill the Schoenberg-Whitney condition [22]:

$$u_i < \tau_i < u_{i+d+1}, \quad i = 1, \ldots, m+d$$

To this end, the knots can be select with the formula [14]:

$$\begin{aligned} u_0 = \cdots = u_d = u_s\,, \quad u_m = \cdots = u_{m+d+1} = u_e \\ u_{j+d} = \frac{1}{d}\sum_{i=j}^{j+d-1} \tau_i\,, \quad j = 1, \ldots, n-d \end{aligned}$$

An extremely general algorithm is reported in [31], which adds to system (2.19) the $\iota + \kappa$ conditions:

$$\boldsymbol{d}_\kappa^{(j)} = D^{(j)}\boldsymbol{r}(\bar{\tau}) = \sum_{i=0}^{m} D^{(j)} B_{i,d,\boldsymbol{u}}(\bar{\tau})\boldsymbol{b}_i \tag{2.20}$$

with

$$\begin{cases} j = 0, \ldots, \kappa & \text{if } \bar{\tau} = u_s \\ j = 0, \ldots, \iota & \text{if } \bar{\tau} = u_e \end{cases}$$

in order to constrain up to the κ^{th} and ι^{th} derivatives at the starting and ending points of the curve, respectively. In matrix form (substituting for simplicity $D^j B_{i,d,\boldsymbol{u}}$ to B_i^j):

$$\begin{bmatrix} 1 & & & & & & & & & \\ B_0^1(u_s) & B_1^1(u_s) & & & & & & & & \\ \vdots & \ddots & & & & & & & & \\ B_0^\kappa(u_s) & \cdots & B_k^\kappa(u_s) & & & & & & & \\ & & B_0(u_1) & \cdots & B_d(u_1) & & & & & \\ & & \ddots & & \ddots & & & & & \\ & & B_{m-d}(u_{n-1}) & \cdots & B_m(u_{n-1}) & & & & & \\ & & & & B_{m-l}^\iota(u_e) & \cdots & B_m^\iota(u_e) & & & \\ & & & & \ddots & & \vdots & & & \\ & & & & & & B_{m-1}^1(u_e) & B_m^1(u_e) & & \\ & & & & & & & 1 & & \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_{m-1} \\ \boldsymbol{b}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_0 \\ \boldsymbol{d}_s^1 \\ \vdots \\ \boldsymbol{d}_s^\kappa \\ \boldsymbol{q}_1 \\ \vdots \\ \boldsymbol{q}_{n-1} \\ \boldsymbol{d}_e^\iota \\ \vdots \\ \boldsymbol{d}_e^1 \\ \boldsymbol{q}_n \end{bmatrix} \tag{2.21}$$

with $m = n + \kappa + \iota$. The extra knots must be inserted in order to obtain a system (2.21) banded, and diagonally dominant in the part where the basic non-vanishing functions are placed. To this end, the extra knots can be insert in the first and last spans. A way to place extra knots that ensures numerical stability is presented in [31].

2.7 Channel problem definition

This section explains the problem of finding a smooth 1-D spline path between a channel defined by two polylines, $\underline{\boldsymbol{c}}(\tau)$ and $\bar{\boldsymbol{c}}(\tau)$. The presented approach is based on the method proposed by [32, 33], which defines a thin envelope of the spline constrained in the channel. The spline is uniquely determined by an optimisation-based method, that minimises an objective function related to the curvature.

2.7.1 Tight linear envelopes for splines

The control polygon $\boldsymbol{\ell}(\tau) : \mathbb{R} \to \mathbb{R}$ of the 1-D spline $\boldsymbol{r}(\tau)$ is the polyline interpolating the points $(u_j^*, b_j)_{j=0}^m$, with u_j^* the j^{th} Greville abscissa, i.e. $\boldsymbol{\ell}(u_j^*) = b_j$. The weighted second differences $\Delta_2 b_j$ of the control points are defined as:

$$\Delta_2 b_j = \frac{b_{j+1} - b_j}{u_{j+1}^* - u_j^*} - \frac{b_j - b_{j-1}}{u_j^* - u_{j-1}^*} \tag{2.22}$$

Let $u_k^* \in [u_l, u_{l+1}]$, the contribution of the i^{th} basis function to the distance between the spline and the control point b_j is captured by the function [33]:

$$\gamma_{ki}(u_k^*) = \begin{cases} \displaystyle\sum_{j=l-d}^{i} (u_i^* - u_j^*) B_{j,d,\boldsymbol{u}}(u_k^*), & i \leq k \\ \displaystyle\sum_{j=i}^{l} (u_j^* - u_i^*) B_{j,d,\boldsymbol{u}}(u_k^*), & i \geq k+1 \end{cases} \tag{2.23}$$

With these definitions, the upper and lower envelope $\underline{\boldsymbol{e}}(\tau) \leq \boldsymbol{r}(\tau) \leq \overline{\boldsymbol{e}}(\tau)$ can be defined with the polylines $\underline{\boldsymbol{e}}(\tau)$ and $\overline{\boldsymbol{e}}(\tau)$, respectively interpolating the points $(u_j^*, \underline{e}_j)_{j=0}^m$ and $(u_j^*, \overline{e}_j)_{j=0}^m$, with $\underline{\boldsymbol{e}}_j$ and $\overline{\boldsymbol{e}}_j$ defined as follow:

$$\underline{e}_k = b_k + \sum_{i=l-d}^{l} \Delta_i^- \gamma_{ki}(u_k^*) \tag{2.24}$$

$$\overline{e}_k = b_k + \sum_{i=l-d}^{l} \Delta_i^+ \gamma_{ki}(u_k^*) \tag{2.25}$$

with

$$\Delta_i^- = \min(0, \Delta_2 b_j) \tag{2.26}$$

$$\Delta_i^+ = \max(0, \Delta_2 b_j) \tag{2.27}$$

An example of a thin envelope calculated with (2.24) and (2.25) is shown in Fig. 2.3.

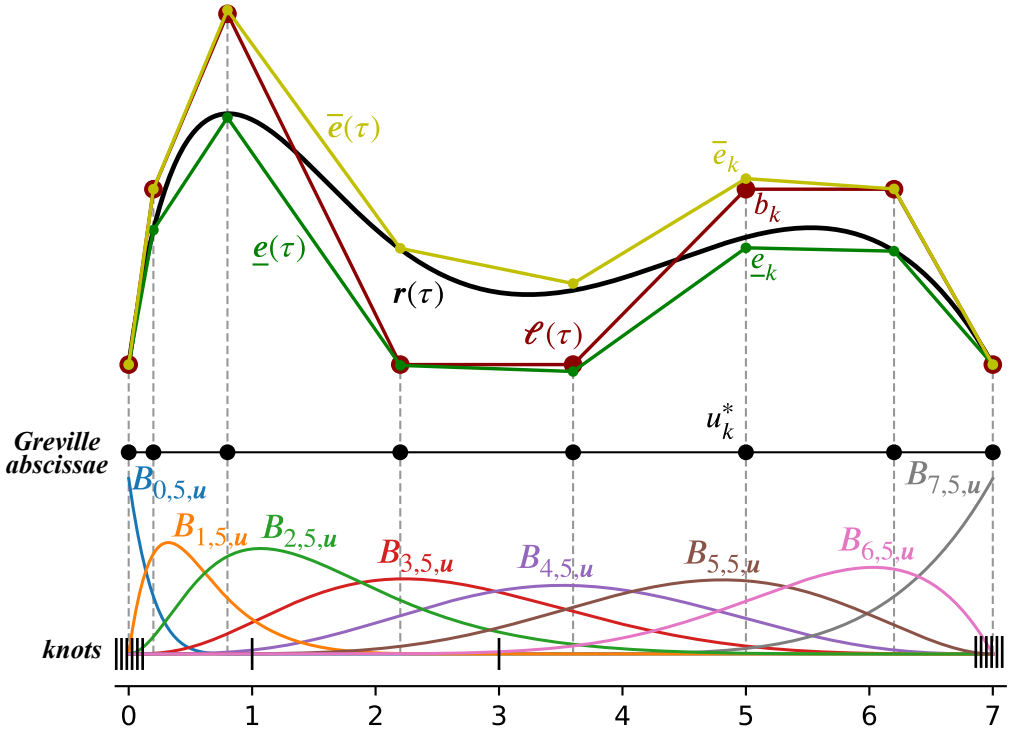


Figure 2.3: A quintic spline and its envelope according to [33]. The knots are (0,0,0,0,0,0,1,3,7,7,7,7,7,7) and the control points are (0,1,2,0,0,1,1,0). This example is taken from [33].

2.7.2 Smooth paths in a polygonal channel

With the linear envelope described in Section 2.7.1, the channel problem can be defined as a quadratic programming (QP) optimization subject to linear constraints. The curvature is minimized considering the objective function:

$$\text{minimize } \rightarrow f(\boldsymbol{b}) = \sum_{i=1}^{m-1} (\Delta_2 b_i)^2 \quad \text{with: } \boldsymbol{b} = (b_i)_{i=1}^{m-1} \tag{2.28}$$

From (2.22):

$$\begin{aligned}\Delta_2 b_i &= \frac{b_{i+1} - b_i}{u_{i+1}^* - u_i^*} - \frac{b_i - b_{i-1}}{u_i^* - u_{i-1}^*} = \frac{1}{\Delta u_{i+1}^* \Delta u_i^*} \left[(b_{i+1} - b_i)\Delta u_i^* - (b_i - b_{i-1})\Delta u_{i+1}^* \right] \\ &= \tfrac{1}{K_i} \left[b_{i+1}\Delta u_i^* - b_i(\Delta u_i^* + \Delta u_{i+1}^*) + b_{i-1}\Delta u_{i+1}^* \right]\end{aligned}$$

with:

$$\Delta u_i^* = u_i^* - u_{i-1}^* > 0$$

$$K_i = \Delta u^*_{i+1}\Delta u^*_i > 0$$

It follow that:

$$(\Delta_2 b_i)^2 = \frac{1}{K_i^2}\left[b_{i+1}^2\Delta u_i^{*2} + b_i^2(\Delta u_i^* + \Delta u_{i+1}^*)^2 + b_{i-1}^2\Delta u_{i+1}^{*2} - 2b_{i+1}b_i\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*) - 2b_ib_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^* + 2b_{i+1}b_{i-1}\Delta u_i^*\Delta u_{i+1}^*\right] = g(b_{i-1}, b_i, b_{i+1})$$

$\forall b_i$ with $i = 2, \ldots, m-2$:

$$\frac{\delta f(\boldsymbol{b})}{\delta b_i} = \frac{\delta(\Delta_2 b_1)^2}{\delta b_i} + \cdots + \underbrace{\frac{\delta(\Delta_2 b_{i-1})^2}{\delta b_i}}_{(1^{st}\text{ comp.})} + \underbrace{\frac{\delta(\Delta_2 b_i)^2}{\delta b_i}}_{(2^{nd}\text{ comp.})} + \underbrace{\frac{\delta(\Delta_2 b_{i+1})^2}{\delta b_i}}_{(3^{rd}\text{ comp.})} + \cdots + \frac{\delta(\Delta_2 b_{m-1})^2}{\delta b_i} \tag{2.29}$$

Only these component are not null

1st component:

$$\begin{aligned}\frac{\delta(\Delta_2 b_{i-1})^2}{\delta b_i} =& \frac{\delta}{\delta b_i}\Bigg[\frac{1}{K_{i-1}^2}\left[b_i^2\Delta u_{i-1}^{*2} + b_{i-1}^2(\Delta u_{i-1}^* + \Delta u_i^*)^2\right.\\ &\left. + b_{i-2}^2\Delta u_i^{*2} - 2b_ib_{i-1}\Delta u_{i-1}^*(\Delta u_{i-1}^* + \Delta u_i^*) - \right.\\ &\left. 2b_{i-1}b_{i-2}(\Delta u_{i-1}^* + \Delta u_i^*)\Delta u_i^* + 2b_ib_{i-2}\Delta u_{i-1}^*\Delta u_i^*\right]\Bigg]\\ =& \frac{2}{K_{i-1}^2}\left[b_i\Delta u_{i-1}^{*2} - b_{i-1}\Delta u_{i-1}^*(\Delta u_{i-1}^* + \Delta u_i^*) + b_{i-2}\Delta u_{i-1}^*\Delta u_i^*\right]\end{aligned}$$

2nd component:

$$\begin{aligned}\frac{\delta(\Delta_2 b_i)^2}{\delta b_i} =& \frac{\delta}{\delta b_i}\Bigg[\frac{1}{K_i^2}\left[b_{i+1}^2\Delta u_i^{*2} + b_i^2(\Delta u_i^* + \Delta u_{i+1}^*)^2 + \right.\\ &\left. b_{i-1}^2\Delta u_{i+1}^{*2} - 2b_{i+1}b_i\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*) - \right.\\ &\left. 2b_ib_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^* + 2b_{i+1}b_{i-1}\Delta u_i^*\Delta u_{i+1}^*\right]\Bigg]\\ =& \frac{2}{K_i^2}\left[b_i(\Delta u_i^* + \Delta u_{i+1}^*)^2 - b_{i+1}\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*)\right.\\ &\left. - b_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^*\right]\end{aligned}$$

3rd component:

$$\frac{\delta(\Delta_2 b_{i+1})^2}{\delta b_i} = \frac{\delta}{\delta b_i}\left[\frac{1}{K_{i+1}^2}\left[b_{i+2}^2\Delta u_{i+1}^{*2} + b_{i+1}^2(\Delta u_{i+1}^* + \Delta u_{i+2}^*)^2\right.\right.$$

$$
\begin{aligned}
&+ b_i^2 \Delta u_{i+2}^{*2} - 2b_{i+2}b_{i+1}\Delta u_{i+1}^{*}(\Delta u_{i+1}^{*} + \Delta u_{i+2}^{*}) - \\
&2b_{i+1}b_i(\Delta u_{i+1}^{*} + \Delta u_{i+2}^{*})\Delta u_{i+2}^{*} + 2b_{i+2}b_i\Delta u_{i+1}^{*}\Delta u_{i+2}^{*}\big]\Big] \\
=&\frac{2}{K_{i+1}^2}\left[b_i\Delta u_{i+2}^{*2} - b_{i+1}(\Delta u_{i+1}^{*} + \Delta u_{i+2}^{*})\Delta u_{i+2}^{*} + b_{i+2}\Delta u_{i+1}^{*}\Delta u_{i+2}^{*}\right]
\end{aligned}
$$

Summing the three components:

$$
\begin{aligned}
\frac{\delta f(\boldsymbol{b})}{\delta b_i} =& \frac{2b_i\Delta u_{i-1}^{*2}}{K_{i-1}^2} - \frac{2b_{i-1}\Delta u_{i-1}^{*}(\Delta u_{i-1}^{*} + \Delta u_i^{*})}{K_{i-1}^2} + \frac{2b_{i-2}\Delta u_{i-1}^{*}\Delta u_i^{*}}{K_{i-1}^2} \\
&+ \frac{2b_i(\Delta u_i^{*} + \Delta u_{i+1}^{*})^2}{K_i^2} - \frac{2b_{i+1}\Delta u_i^{*}(\Delta u_i^{*} + \Delta u_{i+1}^{*})}{K_i^2} \\
&- \frac{2b_{i-1}(\Delta u_i^{*} + \Delta u_{i+1}^{*})\Delta u_{i+1}^{*}}{K_i^2} + \frac{2b_i\Delta u_{i+2}^{*2}}{K_{i+1}^2} \\
&- \frac{2b_{i+1}(\Delta u_{i+1}^{*} + \Delta u_{i+2}^{*})\Delta u_{i+2}^{*}}{K_{i+1}^2} + \frac{2b_{i+2}\Delta u_{i+1}^{*}\Delta u_{i+2}^{*}}{K_{i+1}^2} \\
=& 2b_{i-2}A_i - 2b_{i-1}B_i + 2b_iC_i - 2b_{i+1}D_i + 2b_{i+2}E_i
\end{aligned} \tag{2.30}
$$

with:

$$
\begin{aligned}
A_i &= \frac{\Delta u_{i-1}^{*}\Delta u_i^{*}}{K_{i-1}^2} > 0 \\
B_i &= \frac{\Delta u_{i-1}^{*}(\Delta u_{i-1}^{*} + \Delta u_i^{*})}{K_{i-1}^2} + \frac{(\Delta u_i^{*} + \Delta u_{i+1}^{*})\Delta u_{i+1}^{*}}{K_i^2} > 0 \\
C_i &= \frac{\Delta u_{i-1}^{*2}}{K_{i-1}^2} + \frac{(\Delta u_i^{*} + \Delta u_{i+1}^{*})^2}{K_i^2} + \frac{\Delta u_{i+2}^{*2}}{K_{i+1}^2} > 0 \\
D_i &= \frac{\Delta u_i^{*}(\Delta u_i^{*} + \Delta u_{i+1}^{*})}{K_i^2} + \frac{(\Delta u_{i+1}^{*} + \Delta u_{i+2}^{*})\Delta u_{i+2}^{*}}{K_{i+1}^2} > 0 \\
E_i &= \frac{\Delta u_{i+1}^{*}\Delta u_{i+2}^{*}}{K_{i+1}^2} > 0
\end{aligned}
$$

from (2.30):

$$
\begin{aligned}
&\left.\frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_j}\right|_{j \notin [i-2, i-1, i, i+1, i+2]} = 0 \\
&\frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i-2}} = 2A_i, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i-1}} = -2B_i, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_i} = 2C_i, \\
&\frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i+1}} = -2D_i, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i+2}} = 2E_i
\end{aligned}
$$

The special cases $\delta f(\boldsymbol{b})/\delta b_1$ and $\delta f(\boldsymbol{b})/\delta b_{m-1}$ remain to be defined. From (2.29) it follows that

$$\frac{\delta f(\boldsymbol{b})}{\delta b_1} = -2b_0 B_1 + 2b_1 C_1 - 2b_2 D_1 + 2b_3 E_1 \tag{2.31}$$

with

$$\begin{aligned}
B_1 &= \frac{(\Delta u_1^* + \Delta u_2^*)\Delta u_2^*}{K_1^2} > 0 \\
C_1 &= \frac{(\Delta u_1^* + \Delta u_2^*)^2}{K_1^2} + \frac{\Delta u_3^{*2}}{K_2^2} > 0 \\
D_1 &= \frac{\Delta u_1^*(\Delta u_1^* + \Delta u_2^*)}{K_1^2} + \frac{(\Delta u_2^* + \Delta u_3^*)\Delta u_3^*}{K_2^2} > 0 \\
E_1 &= \frac{\Delta u_2^* \Delta u_3^*}{K_2^2} > 0
\end{aligned}$$

and

$$\frac{\delta f(\boldsymbol{b})}{\delta b_{m-1}} = 2b_{m-3} A_{m-1} - 2b_{m-2} B_{m-1} + 2b_{m-1} C_{m-1} - 2b_m D_{m-1} \tag{2.32}$$

with

$$\begin{aligned}
A_{m-1} &= \frac{\Delta u_{m-2}^* \Delta u_{m-1}^*}{K_{m-2}^2} > 0 \\
B_{m-1} &= \frac{\Delta u_{m-2}^*(\Delta u_{m-2}^* + \Delta u_{m-1}^*)}{K_{m-2}^2} + \frac{(\Delta u_{m-1}^* + \Delta u_m^*)\Delta u_m^*}{K_{m-1}^2} > 0 \\
C_{m-1} &= \frac{\Delta u_{m-2}^{*2}}{K_{m-2}^2} + \frac{(\Delta u_{m-1}^* + \Delta u_m^*)^2}{K_{m-1}^2} > 0 \\
D_{m-1} &= \frac{\Delta u_{m-1}^*(\Delta u_{m-1}^* + \Delta u_m^*)}{K_{m-1}^2} > 0
\end{aligned}$$

In our case, $b_0 = b_m = 0$, so the first component of (2.31) and the last component of (2.32) are null. From (2.31) and (2.32):

$$\begin{aligned}
&\left.\frac{\delta^2 f(\boldsymbol{b})}{\delta b_1 \delta b_j}\right|_{j>3} = 0, \quad \left.\frac{\delta^2 f(\boldsymbol{b})}{\delta b_{m-1} \delta b_j}\right|_{j<m-3} = 0 \\
&\frac{\delta^2 f(\boldsymbol{b})}{\delta^2 b_1} = 2C_1, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_1 \delta b_2} = -2D_1, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_3} = 2E_1 \\
&\frac{\delta^2 f(\boldsymbol{b})}{\delta^2 b_{m-1}} = 2C_{m-1}, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_{m-1} \delta b_{m-2}} = -2B_{m-1},
\end{aligned}$$

$$\frac{\delta^2 f(\boldsymbol{b})}{\delta b_{m-1}\delta b_{m-3}} = 2A_{m-1}$$

The Hessian matrix, defined from $[\nabla^2 f(\boldsymbol{b})]_{ij} \equiv \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_j}$ with $i, j = 1, \ldots, m-1$, thus results in the following band matrix:

$$\nabla^2 f(\boldsymbol{b}) = 2 \begin{bmatrix} C_1 & -D_1 & E_1 & & & & & & 0 \\ -B_2 & C_2 & -D_2 & E_2 & & & & & \\ A_3 & -B_3 & C_3 & -D_3 & E_3 & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & A_{m-3} & -B_{m-3} & C_{m-3} & -D_{m-3} & E_{m-3} & \\ & & & & A_{m-2} & -B_{m-2} & C_{m-2} & -D_{m-2} & \\ 0 & & & & & A_{m-1} & -B_{m-1} & C_{m-1} & \end{bmatrix} \tag{2.33}$$

It follows that the matrix P of equation (2.35) is defined by

$$P = \frac{1}{2}\nabla^2 f(\boldsymbol{b}) \tag{2.34}$$

It follow that (2.28) can be seen in standard form

$$\boldsymbol{f}(\boldsymbol{b}) = \boldsymbol{b}^T P \boldsymbol{b} \qquad \text{with: } P \in \mathbb{R}^{(m-1)\times(m-1)} \tag{2.35}$$

Furthermore, since P is derived from the cost function (2.28) expressed as a sum of squares, it is easy to verify that it is positive semi-definite.

The optimisation problem of the objective function (2.28) is constrained by the inequality relations that ensure $\underline{\boldsymbol{c}}(\tau) \leq \underline{\boldsymbol{e}}(\tau) \leq \boldsymbol{r}(\tau) \leq \overline{\boldsymbol{e}}(\tau) \leq \overline{\boldsymbol{c}}(\tau)$:

$$\underline{\boldsymbol{c}}(u_k^*) \leq \underline{\boldsymbol{e}}(u_k^*) = b_k + \sum_{i=l-d}^{l} \Delta_i^- \gamma_{ki}(u_k^*) \quad k = 1, \ldots, m-1 \tag{2.36}$$

$$\overline{\boldsymbol{c}}(u_k^*) \geq \overline{\boldsymbol{e}}(u_k^*) = b_k + \sum_{i=l-d}^{l} \Delta_i^+ \gamma_{ki}(u_k^*) \quad k = 1, \ldots, m-1 \tag{2.37}$$

$$\begin{aligned} \underline{\boldsymbol{c}}(x_j) \leq \mathcal{L}\big(x_j \mid \underline{\boldsymbol{e}}(u_k^*), \underline{\boldsymbol{e}}_{k+1}(u_{k+1}^*)\big) &= w_1 b_k + w_2 b_{k+1} + \\ w_1 \sum_{i=l_k-d}^{l_k} \Delta_i^- \gamma_{ki}(u_k^*) + w_2 \sum_{i=l_{k+1}-d}^{l_{k+1}} \Delta_i^- \gamma_{k+1,i}(u_{k+1}^*) & \qquad \forall j \in \mathcal{S}^- \end{aligned} \tag{2.38}$$

$$\begin{aligned} \overline{\boldsymbol{c}}(x_j) \geq \mathcal{L}\big(x_j \mid \overline{\boldsymbol{e}}(u_k^*), \overline{\boldsymbol{e}}_{k+1}(u_{k+1}^*)\big) &= w_1 b_k + w_2 b_{k+1} + \\ w_1 \sum_{i=l_k-d}^{l_k} \Delta_i^+ \gamma_{ki}(u_k^*) + w_2 \sum_{i=l_{k+1}-d}^{l_{k+1}} \Delta_i^+ \gamma_{k+1,i}(u_{k+1}^*) & \qquad \forall j \in \mathcal{S}^+ \end{aligned} \tag{2.39}$$

with $\mathcal{S}^-$ the set of indices j corresponding to a concave corners of $\underline{\boldsymbol{c}}(x_j)$, $\mathcal{S}^+$ the set of indices j corresponding to a convex corners of $\overline{\boldsymbol{c}}(x_j)$, w_1 and w_2 defined as follows:

$$w_1 = \frac{u^*_{k+1} - x_j}{u^*_{k+1} - u^*_k}, \qquad w_2 = \frac{x_j - u^*_k}{u^*_{k+1} - u^*_k} \tag{2.40}$$

To eliminate the non-linearity resulting from relations (2.26) and (2.27), the $2m-2$ slack variables $\Delta_i^-,\ \Delta_i^+$ with $i = 1, \ldots, m-1$ are included, in addition to the inequality constraints:

$$\Delta_i^- - \Delta_2 b_i \leq 0 \qquad i = 1, \ldots, m-1 \tag{2.41}$$
$$\Delta_i^+ - \Delta_2 b_i \geq 0 \qquad i = 1, \ldots, m-1 \tag{2.42}$$

and boundaries:

$$\Delta_i^- \leq 0 \qquad i = 1, \ldots, m-1 \tag{2.43}$$
$$\Delta_i^+ \geq 0 \qquad i = 1, \ldots, m-1 \tag{2.44}$$

With this approach, expressions (2.36) and (2.37) define linear constraints. In order to ensure the G^j-continuity of the geometric path, the extreme first and second derivatives must be set. For this purpose, the following equality constraints are imposed [31]:

$$D_0^{(k)} = \sum_{i=1}^{k} B^{(k)}_{i,d,\boldsymbol{u}}(u_i) b_i \qquad k = 1, \ldots, j \tag{2.45}$$

$$D_m^{(k)} = \sum_{i=1}^{k} B^{(k)}_{m-i,d,\boldsymbol{u}}(u_{m-i}) b_{m-i} \qquad k = 1, \ldots, j \tag{2.46}$$

where:

$$B^{(k)}_{i,d,\boldsymbol{u}}(\tau) = d\left(\frac{B^{(k-1)}_{i,d-1,\boldsymbol{t}}(\tau)}{u_{i+d} - u_i} - \frac{B^{(k-1)}_{i+1,d-1,\boldsymbol{t}}(\tau)}{u_{i+d+1} - u_{i+1}}\right) \tag{2.47}$$

Note that equations (2.45) and (2.46) impose conditions on the channel composed of the $\underline{\boldsymbol{c}}(\tau)$ and $\overline{\boldsymbol{c}}(\tau)$ polylines, which must allow the control points of $\boldsymbol{r}(\tau)$ to be positioned correctly.

Finally, the optimisation problem can be described in standard quadratic form as follows:

$$\text{minimize} \qquad \boldsymbol{x}^T \overline{P} \boldsymbol{x} \tag{2.48}$$
$$\text{subject to} \qquad \boldsymbol{l} \leq A\boldsymbol{x} \leq \boldsymbol{u} \tag{2.49}$$

where

$$\boldsymbol{x} = [b_1, \ldots, b_{m-1}, \Delta_1^-, \ldots, \Delta_{m-1}^-, \Delta_1^+, \ldots, \Delta_{m-1}^+]^T \tag{2.50}$$

$$\overline{P} = \begin{array}{c} \left[\begin{array}{c|c} P & 0 \\ \hline 0 & 0 \end{array}\right] \begin{array}{l} m-1 \\ 2m-2 \end{array} \\ \begin{array}{cc} m-1 & 2m-2 \end{array} \end{array} \tag{2.51}$$

$$A = \begin{array}{c} \left[\begin{array}{c|c|c} I & \Gamma & 0 \\ \hline I & 0 & \Gamma \\ \hline W^- & S^- & 0 \\ \hline W^+ & 0 & S^+ \\ \hline -D & I & 0 \\ \hline -D & 0 & I \\ \hline 0 & I & 0 \\ \hline 0 & 0 & I \\ \hline B & 0 & 0 \end{array}\right] \begin{array}{ll} m-1 & \triangleright \text{ eq. (2.36)} \\ m-1 & \triangleright \text{ eq. (2.37)} \\ \#\mathcal{S}^- & \triangleright \text{ eq. (2.38)} \\ \#\mathcal{S}^+ & \triangleright \text{ eq. (2.39)} \\ m-1 & \triangleright \text{ eq. (2.41)} \\ m-1 & \triangleright \text{ eq. (2.42)} \\ m-1 & \triangleright \text{ eq. (2.43)} \\ m-1 & \triangleright \text{ eq. (2.44)} \\ 2j & \triangleright \text{ eq. (2.45), (2.46)} \end{array} \\ \begin{array}{ccc} m-1 & m-1 & m-1 \end{array} \end{array} \tag{2.52}$$

$$\begin{aligned} \boldsymbol{l} = [&\underline{c}(u_1^*), \ldots, \underline{c}(u_{m-1}^*), \underbrace{-\infty, \ldots, -\infty}_{m-1}, \underline{c}(x_j)\ \forall j \in \mathcal{S}^-, \\ &\underbrace{-\infty, \ldots, -\infty}_{\#\mathcal{S}^+}, \underbrace{-\infty, \ldots, -\infty}_{m-1}, \underbrace{0, \ldots, 0}_{m-1}, \\ &\underbrace{-\infty, \ldots, -\infty}_{m-1}, \underbrace{0, \ldots, 0}_{m-1}, \underbrace{D_0^{(1)}, \ldots, D_0^{(j)}}_{j}, \underbrace{D_m^{(1)}, \ldots, D_m^{(j)}}_{j}]^T \end{aligned} \tag{2.53}$$

$$\begin{aligned} \boldsymbol{u} = [&\underbrace{\infty, \ldots, \infty}_{m-1}, \overline{c}(u_1^*), \ldots, \overline{c}(u_{m-1}^*), \underbrace{\infty, \ldots, \infty}_{\#\mathcal{S}^-}, \overline{c}(x_j)\ \forall j \in \mathcal{S}^+, \\ &\underbrace{0, \ldots, 0}_{m-1}, \underbrace{\infty, \ldots, \infty}_{m-1}, \underbrace{0, \ldots, 0}_{m-1}, \\ &\underbrace{\infty, \ldots, \infty}_{m-1}, \underbrace{D_0^{(1)}, \ldots, D_0^{(j)}}_{j}, \underbrace{D_m^{(1)}, \ldots, D_m^{(j)}}_{j}]^T \end{aligned} \tag{2.54}$$

with I the identity matrix, Γ a sparse matrix with entries $[\Gamma]_{ij} = \gamma_{ij}(u_i^*)$, with $i = 1, \ldots, m-1$, $j = l-d, \ldots, l$, l the knot span of u_i^*, B is defined by the constraints (2.45) and (2.46):

$$B = \begin{bmatrix} B_1^{(1)}(u_1) & & & & & 0 \\ \vdots & \ddots & & & & \\ B_1^{(k)}(u_1) & \ldots & B_k^{(k)}(u_k) & & & \\ & & & B_{m-k}^{(k)}(u_{m-k}) & \ldots & B_{m-1}^{(k)}(u_{m-1}) \\ & & & & \ddots & \vdots \\ 0 & & & & & B_{m-1}^{(1)}(u_{m-1}) \end{bmatrix} \tag{2.55}$$

and D the following symmetric tridiagonal matrix:

$$D = \begin{bmatrix} -\frac{(\Delta u_1^* + \Delta u_2^*)}{K_1} & \frac{\Delta u_1^*}{K_1} & & & & 0 \\ \frac{\Delta u_3^*}{K_2} & -\frac{(\Delta u_2^* + \Delta u_3^*)}{K_2} & \frac{\Delta u_2^*}{K_2} & & & \\ & & & \ddots & & \\ & & & \frac{\Delta u_{m-1}^*}{K_{m-2}} & -\frac{(\Delta u_{m-2}^* + \Delta u_{m-1}^*)}{K_{m-2}} & \frac{\Delta u_{m-2}^*}{K_{m-2}} \\ 0 & & & & \frac{\Delta u_m^*}{K_{m-1}} & -\frac{(\Delta u_{m-1}^* + \Delta u_m^*)}{K_{m-1}} \end{bmatrix} \tag{2.56}$$

Finally, W and S are sparse matrices dependent on the specific channel, and related to equations (2.38) and (2.39). Note that since P is positive semi-definite, $\overline{P}$ is positive semi-definite, and it follows that the optimisation problem is convex.

2.7.3 Initial knots sequence and iterative refinement

The problem defined by equations (2.48) and (2.49) is determined once the knots of the unknown spline is defined. Note that the problem defined by the specific channel and spline degree d, may have no solution for the adopted sequence of knots. However, the solution to the problem can be ensured by iteratively knots inserting into the spline. This routine ensures that a solution is found, as the knots refinement leads the envelope to converge with the spline. [32] propose an initial knot sequence such that the corresponding sequence of Greville abscissae includes the breakpoints x_j of the input polylines $\underline{\boldsymbol{c}}(\tau)$ and $\overline{\boldsymbol{c}}(\tau)$. Fig. 2.4 reports an example of the use of this algorithm on a benchmark channel in [32]. One of the advantages of adopting the initial knots sequence suggested by Lutterkort is to introduce additional knots (and Greville abscissae) when the input sequence x_j contains large gaps compared to the surrounding intervals, which usually requires few (if any) additional knot insertions to obtain a solution to the QP minimization problem. Moreover, equations (2.38)

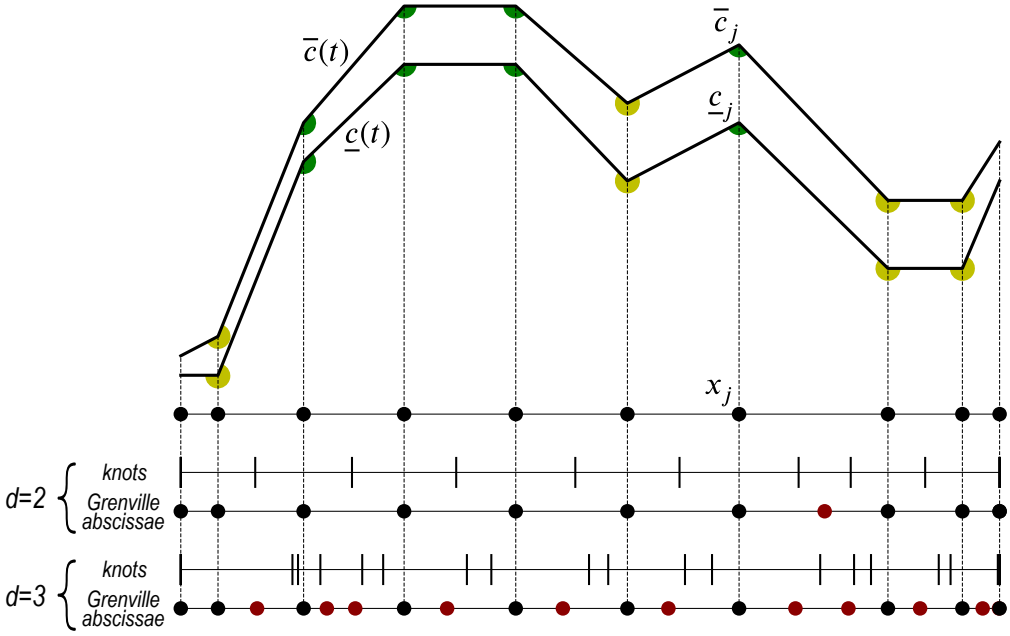


Figure 2.4: Benchmark of the algorithm reported in [32] to define the initial knot sequence. Input polygons defined by $\underline{c}$ = (1, 1, 12, 17, 17, 11, 14, 6.5, 6.5, 11), $\overline{c}$ = (2, 3, 14, 20, 20, 15, 18, 10, 10, 13), x = (0, 0.5, 1.65, 3, 4.5, 6, 7.5, 9.5, 10.5, 11).

and (2.39) are thus included into (2.36) and (2.37) and the constraint matrix A is easier to define. On the other hand, the knots generated by this algorithm define Greville abscissae dependent on the channel breakpoints, which often results in a high gradient in the spacing of the Greville abscissae. This gradient leads to spikes in the second derivative of the resulting spline, which, in this application case, lead to spikes in the torque set-points. To overcome this limitation and have Greville abscissae spaced more evenly, the authors adopt an initial knots sequence $\boldsymbol{u} = (u_i)_{i=0}^{m+d+1}$ obtained by averaging formula from an equispaced mesh in the parametric domain, in accordance with the following:

$$\begin{aligned} &u_0 = \cdots = u_d = 0\,, \qquad u_{m+1} = \cdots = u_{m+d+1} = \overline{u}_m, \\ &u_{j+d} = \frac{\overline{u}_m}{\max(d,1)m} \sum_{i=j}^{j+d-1} i, \qquad j = 1\ldots, m-d. \end{aligned} \tag{2.57}$$

Then, the number of knots $m + d + 1$ is iteratively increased until the method converges. Note that, because of constraints (2.38) and (2.39), the minimum number of knots used to start the iteration must satisfy the following formulations:

$$u_1^* \leq \min\left(\{x_j\}\right) \;\wedge\; u_{m-1}^* \geq \max\left(\{x_j\}\right) \quad \forall j \in (\mathcal{S}^- \cup \mathcal{S}^+) \tag{2.58}$$

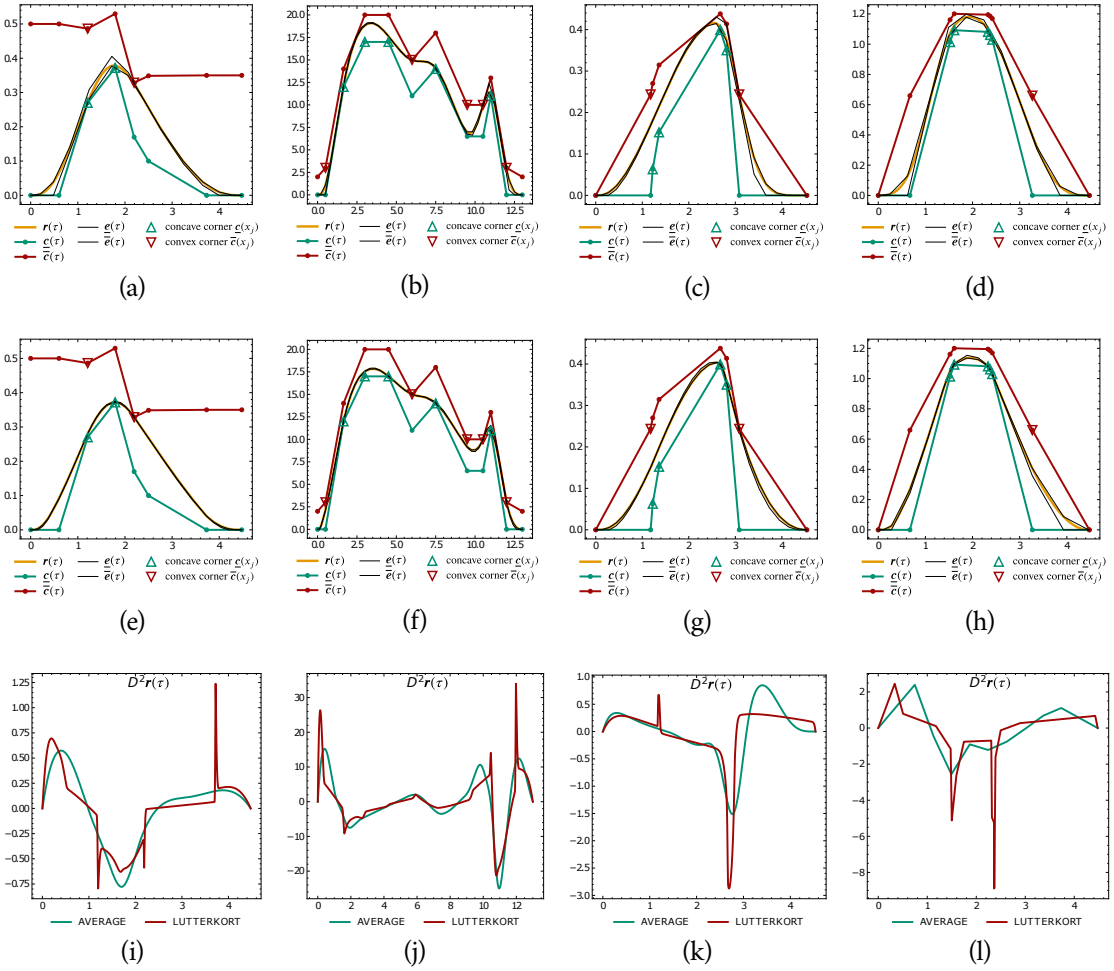


Figure 2.5: Channel problem solutions for four input polygons computed with OSQP solver [34]. (a) to (d) solutions with initial knot sequence computed with (2.57). (e) to (h) solutions with initial knot sequence computed with Lutterkort algorithm [32]. (i) to (l) Comparison of the second derivative of the QP minimization problem spline solution with the two initial knot sequences.

Table 2.1: Benchmark data from 2.5. Performance evaluated on Intel® Core™ i7-10750H CPU, 2.60 GHz.

Channel num.	d	Num. of iterations	Num. of variables[a]	Num. of constraints[a]	CPU time[a]
1 (2.5a)	5	1	36	80	2.59×10^{-4} s
2 (2.5b)	5	21	108	230	8.84×10^{-3} s
3 (2.5c)	8	6	60	130	1.36×10^{-3} s
4 (2.5d)	3	1	33	76	1.77×10^{-4} s

[a] Relative to the last iteration.

Fig. 2.5a to Fig. 2.5d show the solutions to the channel problem for four input polyline pairs, starting with an initial number of knots such that the number of Greville abscissae is equal to the number of breakpoints x_j. In all four cases the end derivatives are clamped with the equality constraints:

$$D_0^{(1)} = D_0^{(2)} = 0, \quad D_m^{(1)} = D_m^{(2)} = 0 \tag{2.59}$$

Solutions are computed by an operator splitting solver for convex quadratic programs (OSQP, [34]), and differ in the channel shape and the degree d of the unknown spline. OSQP solver requires as input the $\overline{P}$ and A matrices in compressed sparse column (CSC) format. A description of the storage CSC format can be found in [35]. In particular, for the matrix $\overline{P}$, only its upper triangular part is required. A numerical benchmark for the channel problem, in compliance with the OSQP solver interface, presented in [36], and corresponding to the first channel shown in Fig. 2.5a, is provided in Appendix A.

Tab. 2.1 shows, for each input channel, the spline degree d, the number of knot refinements required to obtain a solution, and some parameters of the last iteration, including calculation time. Fig. 2.5e to Fig. 2.5h show the solution of the same channel problems solved with a knot sequence calculated using the method suggested by [32]. In this case, the sequence of initial knots already ensures the convergence of the method for all four channels analysed, at the expense of a higher number of knots used. With particular reference to the second channel (Fig. 2.5b and Fig. 2.5f), the method proposed by the authors requires a total of 21 iterations, entailing a higher computational cost than the method detailed in [32]. Generally, this scenario arises for narrow channels, where, starting with a limited number of knots, several insertions are required to solve the channel problem. In the application case object of this work, however, the channels are mostly similar to channel 1 (Fig. 2.5a and Fig. 2.5e), and few or no iterations are usually required to ensure that the problem is solved. Finally, Fig. 2.5i to 2.5l show the comparison of the second derivatives of the resulting spline for the two knots definition methods. In all four of the channels analysed, the use of (2.57) to define the input knots of the QP minimization problem eliminates the spikes of the second derivative of the spline $\boldsymbol{r}(\tau)$, classically resulting in a lower maximum value of $D^{(2)}\boldsymbol{r}(\tau)$ than the comparison method used.

Chapter 3

Geometrical path modelling

In this section, the methodology for planning the geometric path performed by the robot is explained. The proposed method employs a hybrid technique based on optimization and trajectory planning to plan a collision-free path with minimized curvature based on the machine layout, bounding boxes of the parts involved in the point-to-point motion, as well as the initial and final state of the system. The selection of both start and target positions is crucial to ensure the feasibility of the movement from the end-effectors while minimizing the overall displacements necessary for sorting all the nest parts. Traditionally, this process entails multiple iterations, which involve the execution of the motion planning algorithm. Consequently, ensuring optimal computational efficiency in motion planning is of crucial importance. This efficiency is essential to effectively determine the load-unload order of the nest parts and to compute the requisite start and target positions. In the machine under investigation, this operation is carried out by proprietary software. This software, based on a partitioning of work zones among the various end-effectors, similar to the approach introduced in [19], employs a suitable computation strategy to determine the minimum number of movements required to sort all the pieces within the nest.

Depending on the specific plant in which the LST system is installed, different machine configurations must be allowed in order to achieve the best compromise between system efficiency and production/maintenance costs. To this end, the proposed approach is designed to allow a simple scaling of the system to the number of end-effectors for the following four configurations:

- One-arm gantry system (one end-effector).
- Dual-arm gantry system (two end-effectors).
- One-arm dual-gantry system (two end-effectors).
- Dual-arm dual-gantry system (four end-effectors)

In the remainder of this section, the problem of motion planning is therefore investigated from the single end-effector configuration, in other words the case of a simple Cartesian robot. In this scenario, the issue of self-collision is not addressed, and geometric path plan-

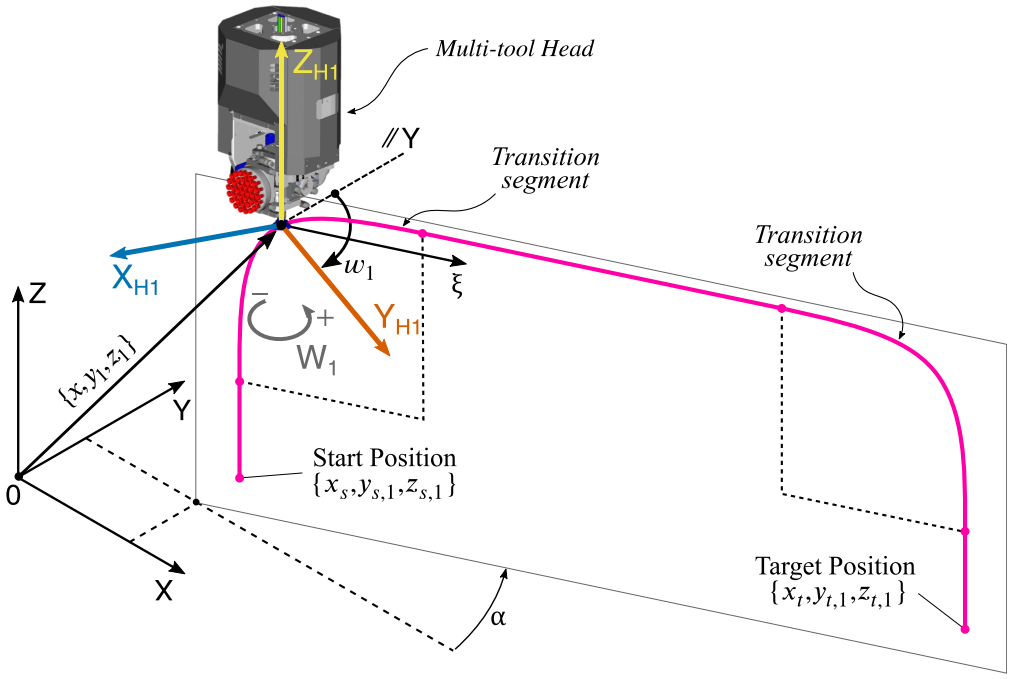
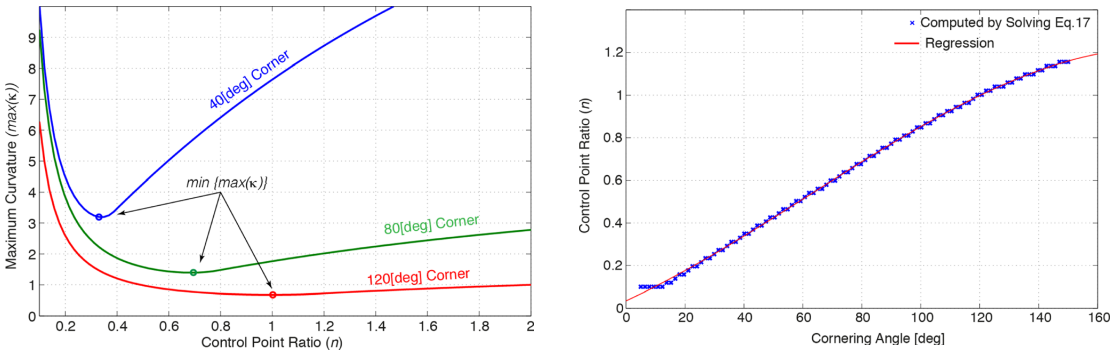


Figure 3.1: 2-D path shape on the motion plane Z_{H1}-ξ.

ning is performed using spline-based trajectory planning techniques. Subsequently, the method is extended for a dual-arm gantry system using an optimisation-based approach. This is also the configuration that finds most application in the specific industrial sector. Finally, the method is scaled to two gantry systems cooperating in the same workspace, for a total of four end-effectors. The solution adopted to allow the rototranslation of parts loaded by two gripping tools is also detailed.

3.1 Single end-effector path

In this section the path planning of a single end-effector is presented. Consider the classic rectangular pick-end-place path shape, consisting of two vertical segments (rise and fall) and one horizontal segment. Corner smoothing algorithms are generally used to obtain a G^j-continuity path, where j depends on the transition segment, and defining a five-segment geometric path. With reference to Fig. 3.1, we refer only to arm 1 of the dual-arm robot represented in Fig. 1.2. The $X_{H1}Y_{H1}Z_{H1}$-frame shown in the figure defines the coordinate system fixed to the head installed on arm 1 of the robot. The resulting geometric path defines a 2-D path on the plane of motion Z_{H1}-ξ, passing through the Z_{H1}-axis and inclined by α with respect to the ZX-plane. In this respect, many examples can be found in the field of numerical control machines, where the geometric path is generally described



(a) (b)

Figure 3.2: Images taken from [37]. (a) Corner curvature for various cornering angles and control point ratios. (b) Optimal n w.r.t. cornering angle.

in the form of G01 polylines. Some recent work on this subject can be found in [38, 39, 40]. For our application, of particular interest is the study conducted by [37], where a corner smoothing algorithm using quintic Bézier curves is proposed to ensure both G^2-continuity of the tool-path and minimised curvature. This parameterization is reported in Fig. 3.3a. Let $\boldsymbol{P} = (\boldsymbol{P}_i)_{i=0}^5$, with $\boldsymbol{P}_i = \{P_{x,i}, P_{y,i}, P_{z,i}\}$, the control points of the transition segment, and define the following 3-D Euclidean distances:

$$\|\boldsymbol{P}_j - \boldsymbol{P}_{j-1}\| = c\,, \quad \text{with: } j = 1, 2, 4, 5$$
$$\|\boldsymbol{P}_{trans} - \boldsymbol{P}_2\| = \|\boldsymbol{P}_3 - \boldsymbol{P}_{trans}\| = d$$

with $\boldsymbol{P}_{trans}$ the corner of the corresponding rectangular pick-and-place trajectory. Since the Bezier blend exhibits symmetry with respect to the angular bisector of adjacent segments, this results in either the maximum curvature being at the midpoint of the Bezier curve or the presence of two identical curvature extremes. This observation vaguely suggests that the problem is convex, which means that there is a global optimal n value that minimizes the maximum curvature value. In [37], this observation is followed by the definition of an optimized formulation of the ratio $n = c/d$ as a function of the transition angle θ to obtain the minimum transition curvature:

$$n(\theta) = \frac{1}{2.0769}\theta^{0.9927} \tag{3.1}$$

Condition (3.1) maximizes the total cornering velocity while respecting the acceleration limits of the actuators, and is defined by a simple power regression on data obtained from calculating the minimum curvature for a wide range of transition angles. For completeness, two images from the study proposed in [37] are shown in Fig 3.2. In Fig. 3.2a the maximum curvature for three values of the transition angle θ as the control point ratio n varies is

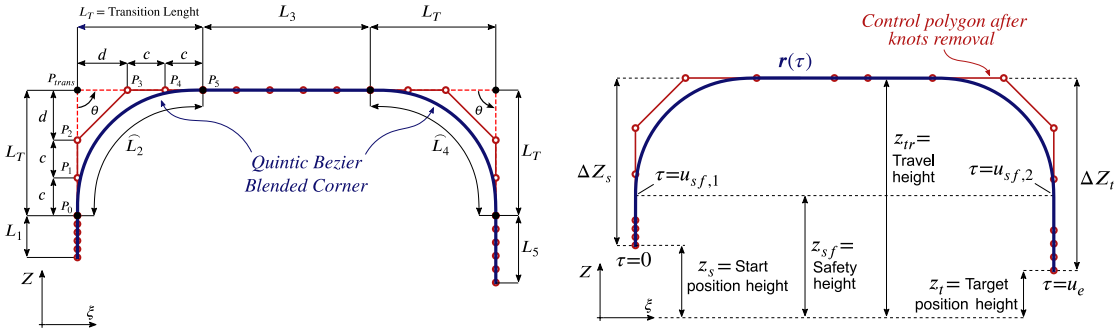


Figure 3.3: Five-segment 2-D geometric path. (a) Geometric path parameterization. (b) Reference heights. $u_{sf,1}$ and $u_{sf,2}$, if they exist, are the values of the parameter τ corresponding to the intersections of the geometric path with the safety height z_{sf}.

reported. Fig. 3.2b shows, on the other hand, the value of n for a wide curve angle $\theta = 10 \ldots 150°$, which minimises the maximum curvature.

In our case it will always be $\theta = \pi/2$, it follows

$$n(\pi/2) = \frac{1}{2.0769}(\pi/2)^{0.9927} \simeq 0.7538 \tag{3.2}$$

It should be noted that although (3.2) will always be adopted in this application, the method to follow is not costrained by this assumption. If a transition angle value $\theta \neq \pi/2$ is desired, it would then be possible to use (3.1) to calculate the appropriate control point ratio n.

Once the ratio n is set, the transition segments depend only on the transition length $L_T = 2 \cdot c + d$. In this application, with reference to Fig. 1.2 and Fig. 3.3b, the transition length L_T assumes a maximum value defined by the difference between the travel height z_{tr} and the loading zone height z_p:

$$L_{T,max} = z_{tr} - z_p \tag{3.3}$$

Given the start and target positions of the end-effector:

$$\boldsymbol{p}_s = \{x_s, y_s, z_s, w_s\}$$
$$\boldsymbol{p}_t = \{x_t, y_t, z_t, w_t\}$$

we can define:

$$\Delta X = x_t - x_s$$
$$\Delta Y = y_t - y_s$$
$$\Delta Z_s = z_{tr} - z_s$$

$$\Delta Z_t = z_t - z_{tr}$$
$$\Delta W = w_t - w_s$$

The transition length L_T is calculated as:

$$L_T = \min\left(L_{T,max}, \frac{\sqrt{\Delta X^2 + \Delta Y^2}}{2}\right) \tag{3.4}$$

The proposed approach involves two-step motion planning; first a 3-D path is planned condidering only X,Y, and Z axes to obtain specific smoothness characteristics, and only in a second step is added the W dimension. The 3-D path followed by the end-effector can be described by a spline $\boldsymbol{r}(\tau) : \mathbb{R} \to \mathbb{R}^3$, defined by (2.1), with $\boldsymbol{b}_i = \{x_i, y_i, z_i\}$. The construction of $\boldsymbol{r}(\tau)$ can be performed by joining the five segments $(\boldsymbol{r}_i)_{i=0}^4$ defined by 3-D Bézier curves in their B-form (thus without internal knots). In order to preserve the G^2-continuity and obtain a single spline of degree 5, the Bézier curves must be joined together having the same geometric velocity and acceleration at the welding points. For example, for the first transition segment must be:

$$D\boldsymbol{r}_1(u_{1,e}) = D\boldsymbol{r}_2(u_{2,s}), \quad D^{(2)}\boldsymbol{r}_1(u_{1,e}) = D^{(2)}\boldsymbol{r}_2(u_{2,s})$$
$$D\boldsymbol{r}_2(u_{2,e}) = D\boldsymbol{r}_3(u_{3,s}), \quad D^{(2)}\boldsymbol{r}_2(u_{2,e}) = D^{(2)}\boldsymbol{r}_3(u_{3,s})$$

with $\boldsymbol{r}_i$ the i^{th} segment having knots:

$$\boldsymbol{u}_i = \{\underbrace{u_{i,s}, \ldots, u_{i,s}}_{p+1}, \underbrace{u_{i,e}, \ldots, u_{i,e}}_{p+1}\}$$

By setting $u_{2,s} = 0$ and $u_{2,e} = 1$, [37]:

$$D\boldsymbol{r}_2(0) = 5(\boldsymbol{P}_1 - \boldsymbol{P}_0), \quad D^{(2)}\boldsymbol{r}_2(0) = 20(\boldsymbol{P}_2 - 2\boldsymbol{P}_1 + \boldsymbol{P}_0)$$
$$D\boldsymbol{r}_2(1) = 5(\boldsymbol{P}_5 - \boldsymbol{P}_4), \quad D^{(2)}\boldsymbol{r}_2(1) = 20(\boldsymbol{P}_5 - 2\boldsymbol{P}_4 + \boldsymbol{P}_3)$$

Passing to scalar notation:

$$\begin{aligned} \|D\boldsymbol{r}_2(0)\| &= \|D\boldsymbol{r}_2(1)\| = 5c \\ \|D^{(2)}\boldsymbol{r}_2(0)\| &= \|D^{(2)}\boldsymbol{r}_2(1)\| = 0 \end{aligned} \tag{3.5}$$

Assuming a second transition segment symmetrical to the first, (3.5) can also be used for segment 4. The G^2-continuity of the spline obtained by joining the five segments can thus be ensured by setting:

$$\begin{aligned} u_{1,s} &= 0, & u_{1,e} &= L_1/(5c) \\ u_{3,s} &= 0, & u_{3,e} &= L_3/(5c) \\ u_{5,s} &= 0, & u_{5,e} &= L_5/(5c) \end{aligned}$$

The spline resulting from joining segments $\boldsymbol{r}_i$, can be easily solved by following the operations:

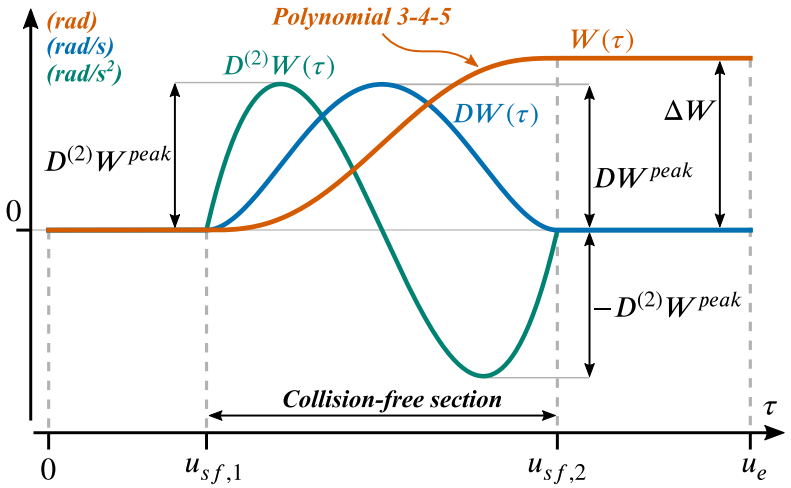


Figure 3.4: $W(\tau)$ motion profile. In the collision-free section the motion is described from polynomial 3-4-5 formulation, according to (3.6).

1. Elevate the degree of segments 1,3, and 5 to reach the same degree as the transition segments with the algorithm explained in Section 2.2.
2. Concatenate the Bézier curves by repeating for the number of segments the joining operation explained in Section 2.3.
3. Remove as many knots as possible while preserving the geometric path shape within a tolerance. To this end, we suggest an optimized routine proposed by [23] that minimize the number of displacements of coefficients and knots in the case of multiple knot removal.

The procedure just described can be used for a generic path, even with a number of segments other than five. As will become clearer in the next sections, in this application the five-segment geometric path of Fig. 3.3 is the most general case, but some movements reach intermediate points of the pick-and-place task using fewer than five segments.

When the 3-D path is defined, it's necessary to add the W information to manage the angular orientation of the end-effector. In general, the loaded part can only rotate in a collision-free section of the trajectory, defined by a safety height z_{sf}. With reference to Fig. 3.3b, $u_{sf,1}$ and $u_{sf,2}$ are the values of the parametric variable τ bounding the portion of $\boldsymbol{r}(\tau)$ corresponding to a Z value greater than the safaty height. In other words, the interval $[u_{sf,1}, u_{sf,2}]$ is defined such that:

$$\text{proj}(\boldsymbol{r}(\tau)|_{[u_{sf,1},u_{sf,2}]}, Z) \geq z_{sf}$$

This interval can easily be estimated by numerical techniques, if it is assumed that during the rise section the Z-component of $\boldsymbol{r}(\tau)$ is strictly increasing, and that during the fall section it is strictly decreasing. Note that, for the condition $\bar{\boldsymbol{r}}(u_s) \geq z_{sf} \wedge \bar{\boldsymbol{r}}(u_e) \geq z_{sf}$ results $[u_{sf,1}, u_{sf,2}] \equiv [0, u_e]$. As reported in 3.4, it follows that $W(\tau)$ is at most divided

into three sections, of which only one contributes to the movement. In order to preserve the G^2-continuity of the geometric path, a good choice is to describe $W(\tau)$ in the collision-free section using a 5^{th}-degree polynomial:

$$W(\tau)|_{[u_{sf,1},u_{sf,2}]} = \Delta W \left[10 \left(\frac{\tau}{\Delta u} \right)^3 - 15 \left(\frac{\tau}{\Delta u} \right)^4 + 6 \left(\frac{\tau}{\Delta u} \right)^5 \right] \tag{3.6}$$

with $\Delta u = u_{sf,2} - u_{sf,1}$. Expression (3.6) defines a polynomial in its p-form:

$$\boldsymbol{p}(x) = \sum_{i=0}^{d} x^i a_i = a_0 + x a_1 + x^2 a_2 + \ldots + x^d a_d \tag{3.7}$$

that can always be seen in its B-form, with knots:

$$\boldsymbol{u} = \{ \underbrace{u_{sf,1}, \ldots, u_{sf,1}}_{d+1}, \underbrace{u_{sf,2}, \ldots, u_{sf,2}}_{d+1} \}$$

and control points $\boldsymbol{b}$ defined by solving the system:

$$M\boldsymbol{b} = \boldsymbol{a}$$

where the lower triangular matrix M is defined by the entries:

$$[M]_{i,j} = \binom{d}{i} \binom{i}{j} (-1)^{i-j} \quad , \quad \text{with} \begin{cases} i = 0, \ldots, d \\ j = 0, \ldots, i \end{cases}$$

The motion profile of the end-effector orientation, defined by the 1-D spline $W(\tau)$ with scalar control points w_i, is then obtained concatenating the sections of which it is composed with similar methodology as explained above. Finally, $W(\tau)$ can be added as a component of $\boldsymbol{r}(\tau)$ by performing knot refinement algorithms to make the curves compatible, and defining a new knot vector $\bar{\boldsymbol{u}}$ as the union of the knots of the two curves [30]. The resulting spline $\bar{\boldsymbol{r}}(\tau)$, with control points $\bar{\boldsymbol{b}}_i = \{\bar{x}_i, \bar{y}_i, \bar{z}_i, \bar{w}_i\}$, defines the desired geometric path.

To finalize the path definition, it remains to define the parameterization of the function describing the geometric path. A common solution is to adopt the arc length parameterisation, then parameterise $\bar{\boldsymbol{r}}(\tau)$ on the curvilinear (or profile) abscissa of the end-effector's 3-D path. However, this solution is not particularly convenient in this application because it limits the movements along the W-axis to tasks in which the machine also performs a movement along at least one of the X, Y or Z axes. In fact, by adopting this parameterisation, requiring motion only along the W-axis leads to the condition $\bar{u}_s = \bar{u}_e = 0$, forcing the adoption of a customized motion planning strategy for this specific case. To prevent

this, the authors propose a parameterisation of $\bar{\boldsymbol{r}}(\tau)$ based on the definition of the 4-D curvilinear abscissa $s \in [0, s_{max}]$ defined in differential form as follow:

$$ds = \|\boldsymbol{K}_r \odot D\bar{\boldsymbol{r}}(\tau)\|d\tau$$

with $\boldsymbol{K}_r = (K_{r,i})_{i=0}^{3} = \{K_{r,x}, K_{r,y}, K_{r,z}, K_{r,w}\}$ a transmission ratios' vector, and where, depending if the axis is linear or rotational, the unit of measure of its components are rad/m or rad/rad. In this case, $K_{r,x}$, $K_{r,y}$, and $K_{r,z}$ are expressed in rad/m, while $K_{r,w}$ is expressed in rad/rad. The modified chord length s_{max} can be estimate by the equation:

$$s_{max} = \sum_{i=0}^{m-1} \sqrt{K_{r,x}\Delta x_i^2 + K_{r,y}\Delta y_i^2 + K_{r,z}\Delta z_i^2 + K_{r,w}\Delta w_i^2} \tag{3.8}$$

with

$$\begin{aligned}
\Delta x_i &= x_{i+1} - x_i \\
\Delta y_i &= y_{i+1} - y_i \\
\Delta z_i &= z_{i+1} - z_i \\
\Delta w_i &= w_{i+1} - w_i
\end{aligned}$$

The accuracy of the s_{max} estimated with (3.8) can be ensured by adopting an appropriate knot refinement strategy, as this operation leads the control polygon to converge with its spline. In this work, knots are inserted iteratively with bisection method until the calculated value does not deviate from that computed in the previous step less than a tolerance adopted. Once estimated s_{max} from (3.8), $\bar{\boldsymbol{r}}(\tau)$ can be parameterised on the curvilinear coordinate s by multiplying all knots $\bar{u}_i$ by $s_{max}/\bar{u}_e$.

3.2 Dual-arm gantry system path

3.2.1 7-d.o.f. dual-arm Cartesian robot path planning

The path planning procedure illustrated in the previous section can be used for a Cartesian robot with only one arm. In applications where two arms are rigidly linked along an axis, this solution cannot be used because it is not possible to plan the geometric path of each end-effector independently. In this section, in order to describe the constraint physically defined by the gantry linkage, the authors propose an extension of the previously presented solution for the case of a dual-arm Cartesian robot, in which the two arms are rigidly linked along the X-axis.

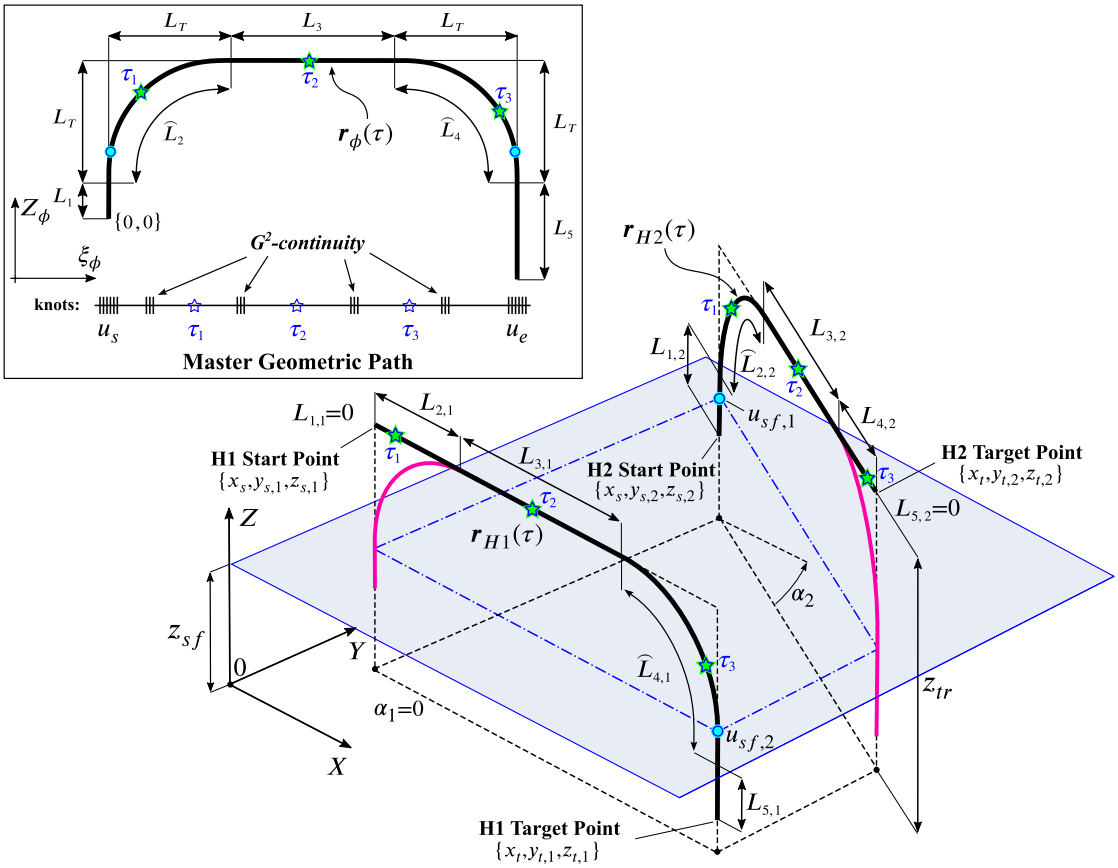


Figure 3.5: Geometric path representation of the dual-arm Cartesian robot with 5^{th}-degree spline. The circular markers correspond to the intersections of the path with the safety height calculated in accordance with (3.12). The star markers are examples of points calculated respectively on the master path $\boldsymbol{r}_\phi(\tau)$ and on $\boldsymbol{r}(\tau)$ for three values of the parametric variable τ.

Given the start and target points of the two end-effectors:

$$\begin{aligned}
\boldsymbol{p}_{s,1} &= \{x_s, y_{s,1}, z_{s,1}, w_{s,1}\} \\
\boldsymbol{p}_{s,2} &= \{x_s, y_{s,2}, z_{s,2}, w_{s,2}\} \\
\boldsymbol{p}_{t,1} &= \{x_t, y_{t,1}, z_{t,1}, w_{t,1}\} \\
\boldsymbol{p}_{t,2} &= \{x_t, y_{t,2}, z_{t,2}, w_{t,2}\}
\end{aligned}$$

may be defined:

$$\begin{aligned}
\Delta X &= x_t - x_s \\
\Delta Y_1 &= y_{t,1} - y_{s,1} \\
\Delta Y_2 &= y_{t,2} - y_{s,2}
\end{aligned}$$

$$\begin{aligned}
\Delta Z_{s,1} &= z_{tr} - z_{s,1} \\
\Delta Z_{t,1} &= z_{t,1} - z_{tr} \\
\Delta Z_{s,2} &= z_{tr} - z_{s,2} \\
\Delta Z_{t,2} &= z_{t,2} - z_{tr} \\
\Delta W_1 &= w_{t,1} - w_{s,1} \\
\Delta W_2 &= w_{t,2} - w_{s,2}
\end{aligned}$$

The main idea for extending the procedure of Section 3.1 is to define a master path, described from the 5[th]-degree spline $\boldsymbol{r}_\phi(\tau) : \mathbb{R} \rightarrow \mathbb{R}^3$ with $\boldsymbol{b}_{i,\phi} = \{x_{i,\phi}, y_{i,\phi}, z_{i,\phi}\}$ and $i = 0, \ldots, m$. Let $\boldsymbol{u} = (u_i)_{i=0}^{m+6}$ be the knots of $\boldsymbol{r}_\phi(\tau)$. Assume:

$$x_{0,\phi} = y_{0,\phi} = z_{0,\phi} = 0$$

which corresponds to define the spline relatively to the first control point. $\boldsymbol{r}_\phi(\tau)$ can be calculated as described in Section 3.1 using the geometric lengths L_i, with $i \in \{1, 3, 5\}$, obtained by solving the following system:

$$\begin{aligned}
&L_1 + L_T = \max(\Delta Z_{s,1}, \Delta Z_{s,2}) \\
&L_3 + 2 \cdot L_T = \max\left(\sqrt{\Delta X^2 + \Delta Y_1^2}, \sqrt{\Delta X^2 + \Delta Y_2^2}\right) \\
&L_T + L_5 = \max(|\Delta Z_{t,1}|, |\Delta Z_{t,2}|)
\end{aligned}$$

with L_T defined by (3.4).

The geometric path of the two end-effectors can be described starting from $\boldsymbol{r}_\phi(\tau)$ with a 5-D spline $\boldsymbol{r}(\tau)$ having the same knots $\boldsymbol{u}$ and control points $\boldsymbol{b}_i = \{x_i, y_{i,1}, z_{i,1}, y_{i,2}, z_{i,2}\}$. Assuming that $\boldsymbol{u}$ is properly calculated to describe the G^2-continuity at the transition segments (as shown in 3.5), with the correct knots multiplicity for a spline of degree d the i^{th} control point are uniquely determined through path scaling techniques from the relations:

$$x_i = x_{i,\phi} + x_s \tag{3.9}$$

$$y_{i,j} = y_{i,\phi} \cdot \frac{\Delta Y_j}{y_{m,\phi}} + y_{s,j} \tag{3.10}$$

$$z_{i,j} = \begin{cases} z_{i,\phi} \dfrac{\Delta Z_{s,j}}{(L_1 + L_T)} + z_{s,j}, & \text{if i<l} \\ z_{tr} + \big(z_{i,\phi} - (L_1 + L_T)\big) \dfrac{\Delta Z_{t,j}}{(L_T + L_5)}, & \text{if i>r} \\ z_{tr}, & \text{otherwise} \end{cases} \tag{3.11}$$

with $l = 2 \cdot d - 1$, $r = 4 \cdot d - 6$, $d = 5$ and $j = 1, 2$. With reference to 3.5, to describe the angular orientation of the end-effectors, it is useful to define the splines $\boldsymbol{r}_{H1}$ and $\boldsymbol{r}_{H2}$,

parametrized on the same parametric variable τ, with knots $\boldsymbol{u}$ and i^{th} control point defined by $\boldsymbol{b}_{i,1} = \{x_i, y_{i,1}, z_{i,1}\}$ and $\boldsymbol{b}_{i,2} = \{x_i, y_{i,2}, z_{i,2}\}$, respectively. Note that these splines describe the 3-D geometric paths of the two heads, which share the X-axis. Let the interval $[u_{sf,1}, u_{sf,2}]$ be such that:

$$\begin{cases} \text{proj}(\boldsymbol{r}_{H1}(\tau)|_{[u_{sf,1},u_{sf,2}]},\ Z) \geq z_{sf} \\ \text{proj}(\boldsymbol{r}_{H2}(\tau)|_{[u_{sf,1},u_{sf,2}]},\ Z) \geq z_{sf} \end{cases} \tag{3.12}$$

The geometric path components $W_1(\tau)$ and $W_2(\tau)$ of the two end-effectors, with i^{th} control point $w_{i,1}$ and $w_{i,2}$, can be calculated with the same routine detailed in the previous section applied to the splines $\boldsymbol{r}_{H1}(\tau)$ and $\boldsymbol{r}_{H2}(\tau)$. It is easy to prove that the W-components are the scaling of each other. In other words, it is sufficient to solve the system (3.1) for only one of them, and obtain the other by multiplying the computed control points by a scaling factor. In this work, it is assumed to calculate the W-component with the largest amplitude:

$$\Delta W_{max} = \max(|\Delta W_1|, |\Delta W_2|)$$

and the scaling ratio to be used to calculate the control points of the second spline is defined by:

$$W_r = \min(|\Delta W_1|, |\Delta W_2|)/\Delta W_{max}$$

For example, if $\Delta W_1 > \Delta W_2$, $W_1(\tau)$ is computed by the routine of section 3.1, while $W_2(\tau)$ is defined by the same degree and knots of $W_1(\tau)$, and the i^{th} control point

$$w_{i,2} = \text{sgn}(\Delta W_2) w_{i,1} W_r$$

The geometric path components $W_1(\tau)$ and $W_2(\tau)$ can then be added to the geometric path using the same method detailed in the previous section, defining the 7-D spline $\bar{\boldsymbol{r}}(\tau)$ with knots $\bar{\boldsymbol{u}}$ and control points: $\bar{\boldsymbol{b}}_i = \{\bar{x}_i, \bar{y}_{i,1}, \bar{z}_{i,1}, \bar{w}_{i,1}, \bar{y}_{i,2}, \bar{z}_{i,2}, \bar{w}_{i,2}\}$.

It remains to define the parameterisation to be adopted for $\bar{\boldsymbol{r}}(\tau)$. To this end, the authors propose a parametric variable $s \in [0, s_{max}]$ based on the master path $\boldsymbol{r}_\phi(\tau)$ and the W-component with the largest amplitude. Assuming the same trasmission ratios for the Y, Z, and W of the two arms:

$$\begin{aligned} \Delta s_{i,\phi}^2 &= K_{r,x}\Delta x_{i,\phi}^2 + K_{r,y}\Delta y_{i,\phi}^2 + K_{r,z}\Delta z_{i,\phi}^2 + K_{r,w}\max(\Delta w_{i,1}^2, \Delta w_{i,2}^2) \\ s_{max} &= \sum_{i=0}^{m-1} \sqrt{\Delta s_{i,\phi}^2} \end{aligned} \tag{3.13}$$

with

$$\begin{aligned} \Delta x_{i,\phi} &= x_{i+1,\phi} - x_{i,\phi} \\ \Delta y_{i,\phi} &= y_{i+1,\phi} - y_{i,\phi} \end{aligned}$$

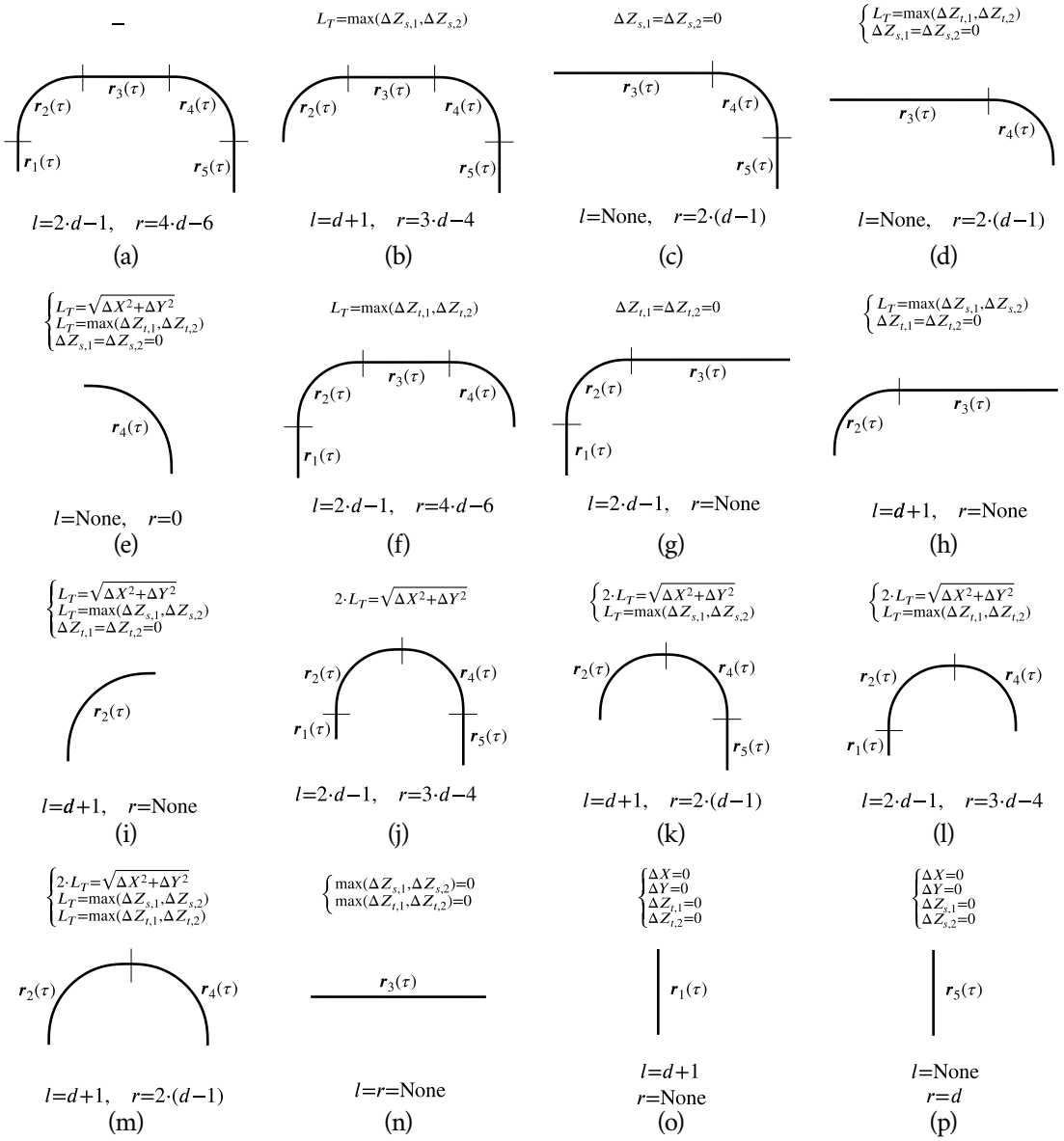


Figure 3.6: Sub-cases of the five-segment path of 3.3 used to define the master path $\boldsymbol{r}_\phi(\tau)$. For each block; at the top the conditions defining the sub-case, in the middle the segments $\boldsymbol{r}_i(\tau)$ used to define the path, at the bottom the indices l and r, to be used in (3.11). If one of the indices l or r is set to None, the relative equation is not used in that subcase.

$$\Delta z_{i,\phi} = z_{i+1,\phi} - z_{i,\phi}$$
$$\Delta w_{i,j} = w_{i+1,j} - w_{i,j} \quad \text{with: } j = 1, 2$$

As in the previous case, the accuracy of the s_{max} estimation can be ensured by adopting an appropriate knot refinement strategy. $\bar{\boldsymbol{r}}(\tau)$ can then be parameterised on the parametric variable s by multiplying all knots $\bar{u}_i$ by $s_{max}/\bar{u}_e$.

This section terminates with some clarifications regarding the five-segment trajectory reported in 3.3a, used here for the master path definition. With reference to the specific application, the geometric path analysed is only the most general case, and not all master paths require all five segments $\boldsymbol{r}_i(\tau)$. In these cases, the missing sections must be considered to adapt the formulations used to scale the path, in particular the knot indices l and r, referring to (3.11). For example, if the Z-component of the start positions $z_{s,1}$ and $z_{s,2}$ are equal to the travel height z_{tr}, then $\Delta Z_{s,1} = \Delta Z_{s,2} = 0$, the optimal choice in terms of path planning is to adopt a three-segment trajectory, consisting of the concatenation of $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$. Similarly, if the Z-components of the target positions are equal to the travel height, the geometric path is obtained from the concatenation of $\boldsymbol{r}_3$, $\boldsymbol{r}_4$ and $\boldsymbol{r}_5$. For the sake of clarity, 3.6 summarises all possible cases with their conditions and the necessary knot indices modifications.

As will highlighted in the following sections, for the proposed motion planning strategy, the maximum curvature of the geometric path has a great impact on machine performances. In this regard, if the transition length L_T calculated with (3.4) reaches extremely small values, the use of the five-segment trajectory for the master path description could produce slow movements. In this work, the problem is solved using a threshold value $L_{T,min}$. If L_T is less than this value, the geometric path is calculated by dividing the movements along the Z-axis resulting in a concatenation of 2 or 3 linear sub-paths.

3.2.2 Collision detection solution

Before detailing the solution adopted by the collision avoidance algorithm, it is necessary to illustrate the solution used to detect the collisions in the XY-plane of the machine and calculate the Y-intersections. The j^{th} arm of the robot ($j = 1, 2$) is modelled by a regular octagon circumscribing its projection on the XY-plane, with edges lenght l_e and centre coincident with the centre head $\{x_{c,j}, y_{c,j}\}$ defined by the intersection of the Z_{Hj}-axis with the XY-plane (see Fig. 3.1). Each part of the nest is modelled with a rectangular bounding box, that circumscribes its external profile. The k^{th} rectangle is fixed to a $X_{p,k}Y_{p,k}$ relative coordinate system, by convention centred at the vertex with absolute position vector of lower magnitude and axis direction consistent with the edges. With reference to Fig. 3.7, the k^{th} picking point is defined by the vector $\{c_{x,k}, c_{y,k}\}$ in the $X_{p,k}Y_{p,k}$ coordinate system. Assuming the k^{th} part loaded by the j^{th} arm, the relative position between the two polygons is uniquely determined by the coincidence of the picking point with the centre head and the

relative angular position defined by the particular head-part configuration in the loading task.

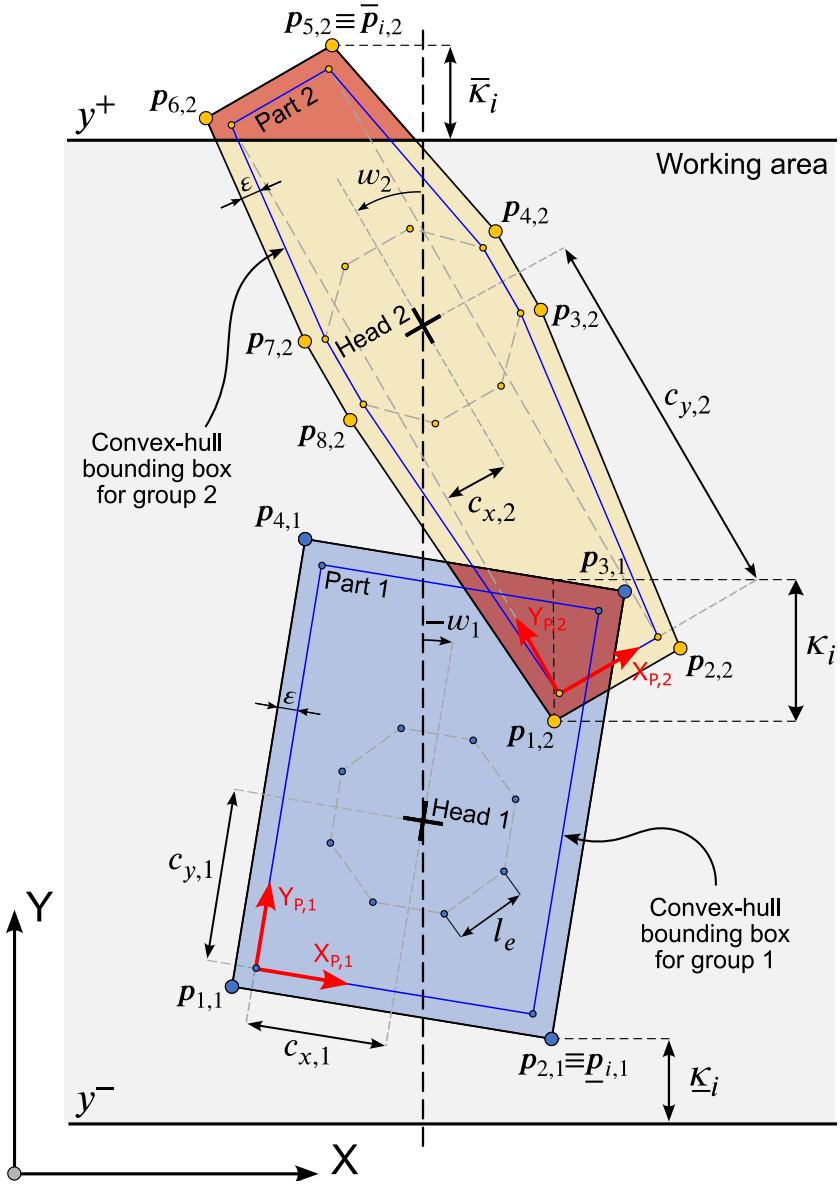


Figure 3.7: Calculation of Y-intersections and bounding box grouping strategy.

For each point-to-point movement of the robot, each end-effector may or may not have a part loaded. As a result, there are a minimum of two and a maximum of four polygons in the working area that define the bounding boxes of the dynamic elements. Each polygon is actually costrained to move fixed to one arm or the other, resulting in two groups of polygons to which the same geometric transformations are applied. To simplify and reduce

subsequent calculations, the polygons of each group are then merged into a single convex shape using the Jarvis' algorithm [41], resulting in only two polygons in the environment. An outer offset ε is then applied to the resulting polygons in order to ensure a safe distance between the polygon groups during the movement. To this end, we use a simple algorithm that calculates the offset vertices of a convex polygon by iteratively computing the intersection points between pairs of adjacent edges that have been displaced along the normal direction by a fixed distance of ε.

A schematic example of a generic frame where both end-effectors have a loaded part, is shown in Fig. 3.7. Let the k^{th} vertex of the j^{th} group ($j = 1, 2$) be defined by the position vector $\boldsymbol{p}_{k,j} = \{p_{k,j,x}, p_{k,j,y}\}$, with respect to the XY absolute coordinate system. The vertices of the polygons are numbered in counter-clockwise direction starting from the vertex with minimum value of $\|\boldsymbol{p}_{k,j}\|$. To calculate the Y-intersection between the polygons of the two groups for the i^{th} frame, the authors propose a method reported in Algorithm 1. For the sake of clarity, 3.8 reports some examples of Y-intersection calculations using Algorithm 1, which are mentioned in its explanation. The algorithm first selects the vertices with maximum and minimum ordinate for group 1 and group 2, respectively (refer to 3.8a), and defines these ordinates as $\bar{p}_{1,y}$ and $\underline{p}_{2,y}$. Based on these values, the vertex subsets $V_{1,2}$ and $V_{2,1}$ of the two polygons are selected. $V_{1,2}$ contains the vertices of the polygon of group 1 with abscissa greater than or equal to $\underline{p}_{2,y}$, while $V_{2,1}$ contains the vertices of the polygon of group 2 with abscissa less than or equal to $\bar{p}_{1,y}$. If these subsets are empty, e.g. as in Fig. 3.8e, the polygons are separated along Y-axis, and the algorithm returns a null value. If these subsets are not empty, for each vertex contained in $V_{1,2}$ and $V_{2,1}$, the respective edges of the other polygon whose domain contains its abscissa value are selected (refer to Fig. 3.8c). Since by assumption the polygon is convex, these edges are either two for each vertex or none. If no edge is selected, e.g. as in 3.8d, the polygons are separated along the X-axis, and the algorithm returns a null value. On the other hand, if some edges are selected in the previous step, the signed distance in the Y-direction to the respective vertex is calculated for each of them. Several selected vertex may not have the distance calculated with the right sign. This is the case represented in Fig. 3.8f, where the calculated distance between the vertex $\boldsymbol{p}_{7,1}$ and the selected edges is negative. The distances calculated on the edges with the right sign is collected, while the remainder is discarded. The desired overlap value κ_i is then calculated as the maximum between the distances collected. Note that the algorithm presented is generally valid for any pair of convex polygons in the XY-plane. In our specific case, the two polygons under consideration always have at least one point that has the same abscissa, due to the constraint introduced by the gantry kinematics. It follows that the two polygons are never separated along the X-axis.

In this application case, with reference to Fig. 3.7, the end-effectors and loaded parts are constrained to move within a working area, geometrically defined by the maximum and minimum admissible values in the Y direction y^+ and y^-. In the proposed approach, the y^+ constraint is assigned to group 2 only, while the y^- constraint is assigned to group 1.

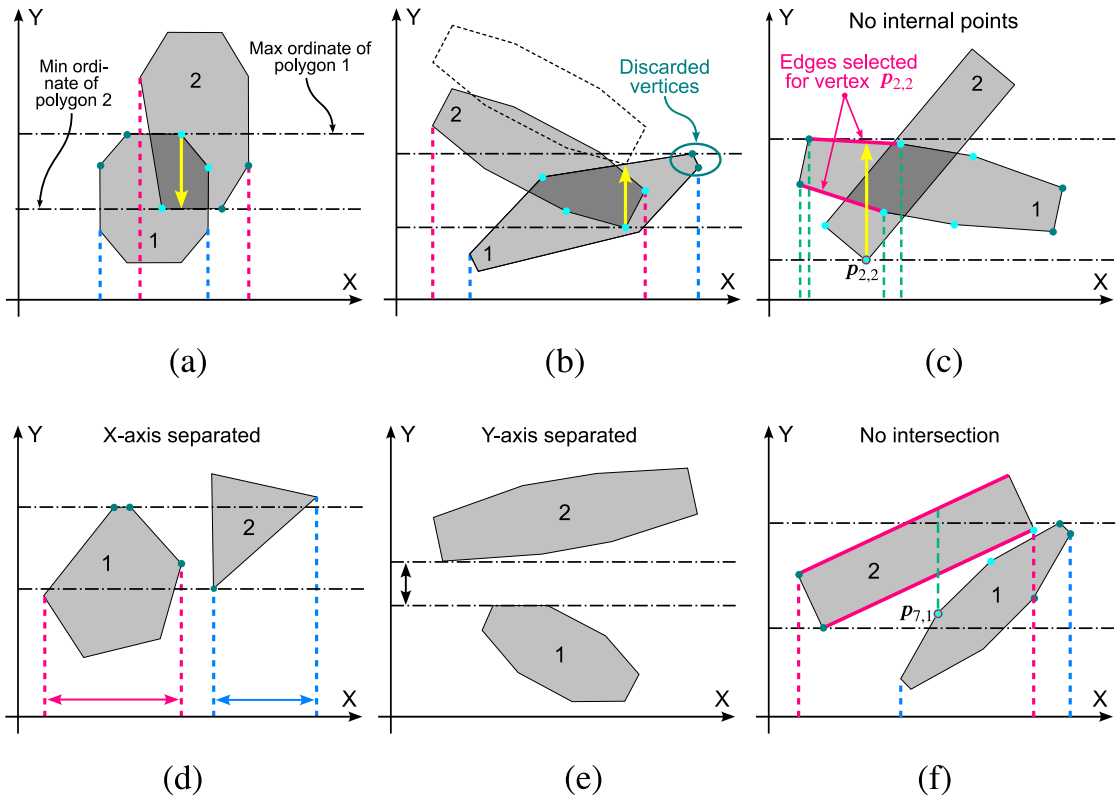


Figure 3.8: Some examples of intersections of convex polygons.

This is justified because if the polygon of group 2 does not satisfy the y^- constraint, it necessarily intersects the polygon of group 1. In other words, the constraint set by the other group is more restrictive than the y^- constraint. Similar logic can be applied to the y^+ constraint with the polygon of group 1. In order to satisfy the constraints introduced by the working area, the Y-intersections $\overline{\kappa}_i$ and $\underline{\kappa}_i$ for the i^{th} frame are computed with:

$$\overline{\kappa}_i = \overline{p}_{i,2,y} - y^+ \tag{3.14}$$

$$\underline{\kappa}_i = y^- - \underline{p}_{i,1,y} \tag{3.15}$$

where $\overline{\boldsymbol{p}}_{i,2} = \{\overline{p}_{i,2,x}, \overline{p}_{i,2,y}\}$ and $\underline{\boldsymbol{p}}_{i,1} = \{\underline{p}_{i,1,x}, \underline{p}_{i,1,y}\}$ are the vertices of groups 1 and 2 having the maximum and minimum ordinate of the polygon.

3.2.3 Frame sampling strategy

The previous section detailed the calculation of Y-intersections κ_i, $\overline{\kappa}_i$ and $\underline{\kappa}_i$, relative to a frame i corresponding to the i^{th} element of a strictly increasing sequence $\boldsymbol{\tau} = (\tau_i)_{i=0}^{\text{M}}$. This sequence represents a 1-D mesh on the parametric domain of the spline $\bar{\boldsymbol{r}}(\tau)$, and is defined by considering the arc lenght of the path in XY-plane and the outer offset used to

Algorithm 1 Convex Polygons Y-Intersection

Input:

1. $V_1 = \{\boldsymbol{p}_{1,1}, \ldots, \boldsymbol{p}_{\mathrm{N}_1,1}\}$, N_1 number of the polygon 1 vertices.
2. $V_2 = \{\boldsymbol{p}_{1,2}, \ldots, \boldsymbol{p}_{\mathrm{N}_2,2}\}$, N_2 number of the polygon 2 vertices.

Output:

1. Y-intersection for i^{th} frame κ_i.

Algorithm:

Select $\max_{r=1,\cdots,N_1}(p_{r,1,y})$ and define it as $\overline{p}_{1,y}$.
Select $\min_{r=1,\cdots,N_2}(p_{r,2,y})$ and define it as $\underline{p}_{2,y}$.
Define the subsets:

$$V_{12} = \{\boldsymbol{p}_{k,1} \,|\, p_{k,1,y} \geq \underline{p}_{2,y}\} \subseteq V_1, \quad V_{21} = \{\boldsymbol{p}_{k,2} \,|\, p_{k,2,y} \leq \overline{p}_{1,y}\} \subseteq V_2$$

if $V_{12} = V_{21} = \emptyset$ **then**
    return $\kappa_i = 0$
end if
Define the sets of indices:
$K_{r,1} = \{k \,|\, \boldsymbol{p}_{k,1} \in V_{12} \wedge \min(p_{k,2,x}, p_{k+1,2,x}) \leq p_{r,1,x} \leq \max(p_{k,2,x}, p_{k+1,2,x})\}, \quad r = 1, \ldots, \#V_{12}$
Define the sets of indices:
$K_{r,2} = \{k \,|\, \boldsymbol{p}_{k,2} \in V_{21} \wedge \min(p_{k,1,x}, p_{k+1,1,x}) \leq p_{r,2,x} \leq \max(p_{k,1,x}, p_{k+1,1,x})\}, \quad r = 1, \ldots, \#V_{21}$
Define the sets:

$$\overset{\circ}{K}_1 = \{\overset{\circ}{K}_{r,1}\} = \{K_{r,1} \,|\, K_{r,1} \neq \emptyset,\ r \in \{1, \cdots, \#V_{12}\}\}$$
$$\overset{\circ}{K}_2 = \{\overset{\circ}{K}_{r,2}\} = \{K_{r,2} \,|\, K_{r,2} \neq \emptyset,\ r \in \{1, \cdots, \#V_{21}\}\}$$

if $\overset{\circ}{K}_1 = \overset{\circ}{K}_2 = \emptyset$ **then**
    return $\kappa_i = 0$
end if
Define the sets:

$$V_1^* := \{\boldsymbol{p}_{r,1}^*\} = \{\boldsymbol{p}_{r,1} \in V_{12} \,|\, K_{r,1} \in \overset{\circ}{K}_1\}$$
$$V_2^* := \{\boldsymbol{p}_{r,2}^*\} = \{\boldsymbol{p}_{r,2} \in V_{21} \,|\, K_{r,2} \in \overset{\circ}{K}_2\}$$

if $\overset{\circ}{K}_1 \neq \emptyset$ **then**
    Define the matrix:

$$D^{(1)} = \begin{bmatrix} \delta_{1,1}^{(1)} & \delta_{1,2}^{(1)} \\ \delta_{2,1}^{(1)} & \delta_{2,2}^{(1)} \\ \vdots & \vdots \\ \delta_{\#\overset{\circ}{K}_1,1}^{(1)} & \delta_{\#\overset{\circ}{K}_1,2}^{(1)} \end{bmatrix}$$

    with:

$$\delta_{j,k}^{(1)} = p_{j,1,y}^* - \mathcal{L}\Big(p_{j,1,x}^* \Big| \boldsymbol{p}_{\overset{\circ}{K}_{k,j,1}+1,2}, \boldsymbol{p}_{\overset{\circ}{K}_{k,j,1},2}\Big)$$

    Set Δ_1 with

$$\Delta_1 = \max(D^{(1)}) \tag{3.16}$$

else
    $\Delta_1 = 0$
end if
Compute Δ_2 in the same way as Δ_1 by replacing $\max(D^{(1)})$ in (3.16) with $\min(D^{(2)})$.
return $\kappa_i = \max(\Delta_1, \Delta_2)$

expand the polygons. This section details the methodology used in this work to define the $\boldsymbol{\tau}$ vector.

With reference to Fig. 3.9, the following definitions are given for frame i and group j:

- $[\tau_i, \tau_{i+1}) := i^{\text{th}}$ span.
- $\boldsymbol{v}_{k,j}^{(i)} = \{v_{k,j,x}^{(i)}, v_{k,j,y}^{(i)}\} := k^{\text{th}}$ vertex of the convex polygon obtained by Jarvis' algorithm.
- $\boldsymbol{p}_{k,j}^{(i)} = \{p_{k,j,x}^{(i)}, p_{k,j,y}^{(i)}\} := k^{\text{th}}$ vertex of the polygon, expanded by an outer offset ε.
- $\boldsymbol{c}_j^{(i)} = \{x_{c,j}^{(i)}, y_{c,j}^{(i)}\} :=$ centre head 2-D point.
- $\Delta l_j^{(i)} = \{\Delta x_j^{(i)}, \Delta y_j^{(i)}\} :=$ traslation vector in the span i.
- $\Delta w_j^{(i)} :=$ rotation of the polygon around the centre head in the span i.

The vertices $\boldsymbol{v}_{k,j}^{(i+1)}$ relative to the frame $i+1$ are defined by:

$$\begin{bmatrix} v_{k,j,x}^{(i+1)} \\ v_{k,j,y}^{(i+1)} \\ 1 \end{bmatrix} = Q_j^{(i)} \begin{bmatrix} v_{k,j,x}^{(i)} \\ v_{k,j,y}^{(i)} \\ 1 \end{bmatrix} \quad \text{with: } k = 1, \ldots, \mathrm{N}_j \tag{3.17}$$

with $Q_j^{(i)}$ the rototranslation matrix of polygon j for the frame i, which describes the rigid transformation performed by the polygon in the span i, and is defined as:

$$Q_j^{(i)} = \begin{bmatrix} C_{\Delta w_j^{(i)}} & -S_{\Delta w_j^{(i)}} & x_{c,j}^{(i)}(1 - C_{\Delta w_j^{(i)}}) + y_{c,j}^{(i)} S_{\Delta w_j^{(i)}} + \Delta x_j^{(i)} \\ S_{\Delta w_j^{(i)}} & C_{\Delta w_j^{(i)}} & -x_{c,j}^{(i)} S_{\Delta w_j^{(i)}} + y_{c,j}^{(i)}(1 - C_{\Delta w_j^{(i)}}) + \Delta y_j^{(i)} \\ 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

where C_θ and S_θ correspond to $\cos(\theta)$ and $\sin(\theta)$.

The proposed solution for the definition of the $\boldsymbol{\tau}$ vector can be summarised as follows. Define $\boldsymbol{\tau}_s$ the initial frames sequence. For each i^{th} span of the vector $\boldsymbol{\tau}_s$ the following condition is checked:

$$\begin{gathered} \forall j \in \{1, 2\}, \forall k \in \{1, .., \mathrm{N}_j\}, \forall r \in \{1, .., \mathrm{N}_j\}, \\ \mathcal{F}\Big(\boldsymbol{v}_{k,j}^{(i+1)} \mid \boldsymbol{p}_{r,j}^{(i)}, \boldsymbol{p}_{(r+1)\bmod(\mathrm{N}_j+1),j}^{(i)}\Big) \geq 0 \end{gathered} \tag{3.19}$$

Condition (3.19) checks, for each group, that the vertices of the polygon of the frame $i+1$ are all contained within the expanded polygon of the frame i. This validation is performed by verifying that each of these vertices is to the left of all the edges of the expanded polygon. If condition (3.19) is not true, an additional frame is inserted in the middle of the span considered in the previous iteration. This procedure is then iterated until all considered spans of the resulting vector $\boldsymbol{\tau}$ are compliant with (3.19).

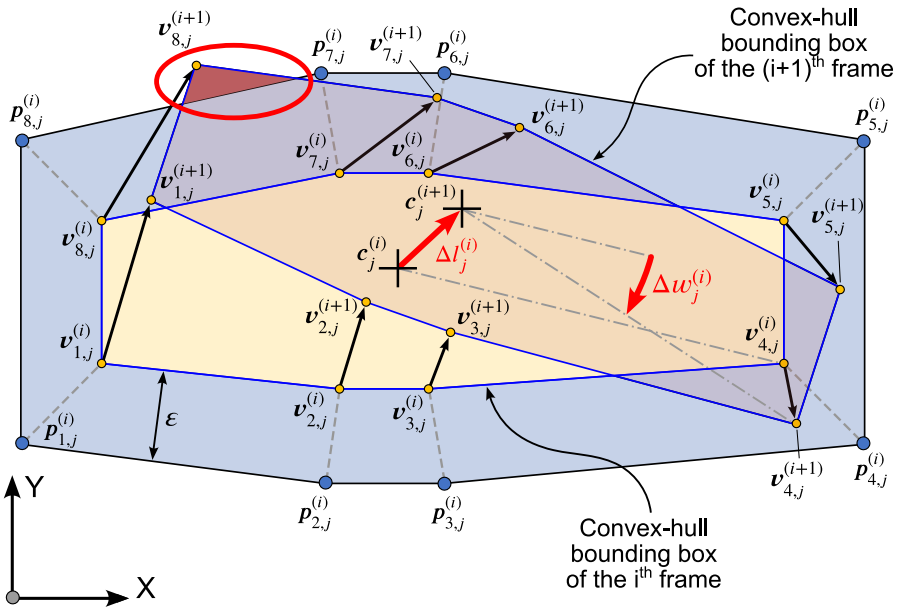


Figure 3.9: Convex polygon obtained by merging the polygons of group j in the frames i and $i+1$. The expanded polygon in the i^{th} frame is also shown in the figure. In this case, an intermediate frame corresponding to parameter $(\tau_i + \tau_{i+1})/2$ must be inserted since the polygon at frame $i+1$ is not contained in the expanded polygon of the previous frame.

3.2.4 Polygonal channel definition

Let $\bar{\boldsymbol{r}}(\tau) : \mathbb{R} \to \mathbb{R}^7$ be a 5^{th}-degree spline function, with $\bar{\boldsymbol{u}} = (\bar{u}_j)_{j=0}^{m+6}$ knots, defining a geometric path computed with the approach of Section 3.2.1. Knowing the robot's operating conditions, the Y-intersections κ_i, $\overline{\kappa}_i$ and $\underline{\kappa}_i$ can be computed on a strictly increasing frames sequence $\boldsymbol{\tau} = (\tau_i)_{i=0}^{\text{M}}$ computed as detailed in the previous section. This results in the definition of vectors:

- Overlap vector: $\boldsymbol{\kappa} = (\kappa_i)_{i=0}^{M}$ (3.20)
- Top Overrun vector: $\overline{\boldsymbol{\kappa}} = \{\max(\overline{\kappa}_i, 0),\ i \in [0, M]\}$ (3.21)
- Bottom Overrun vector: $\underline{\boldsymbol{\kappa}} = \{\max(\underline{\kappa}_i, 0),\ i \in [0, M]\}$ (3.22)

This section proposes a post-processing method for vectors defined by (3.20), (3.21) or (3.22) using the algorithm illustrated in section 2.7. Since this method works indiscriminately on $\boldsymbol{\kappa}$, $\overline{\boldsymbol{\kappa}}$ or $\underline{\boldsymbol{\kappa}}$, in the remainder of the section only $\boldsymbol{\kappa}$ is used, implying that it could refer to any one of them.

The problem described in Section 2.7 requires the definition of a channel in the form of a lower polyline $\underline{\boldsymbol{c}}(\tau)$ and an upper polyline $\overline{\boldsymbol{c}}(\tau)$. The proposed approach firstly calculates $\underline{\boldsymbol{c}}(\tau)$ starting from $\boldsymbol{\kappa}$, after which $\bar{\boldsymbol{c}}(\tau)$ is defined based on the lower polyline. In this application, the vector $\boldsymbol{\kappa}$ has always an initial and final sequence of null values, due to a

Algorithm 2 Upper Envelope Evaluation

Input:

1. Overlap vector $\boldsymbol{\kappa} = \{\kappa_i\}$ with $i = 0, \ldots, \mathrm{M}$.
2. Strictly increasing sequence $\boldsymbol{\tau} = \{\tau_i\}$ with $i = 0, \ldots, \mathrm{M}$.
3. Absolute tolerance Tol.

Output:

1. Indices sequence $\mathbf{I}$.

Algorithm:

1: Define a function $\boldsymbol{\ell}(\tau)$ according to:

$$\boldsymbol{\ell}(\tau) := \mathcal{L}\big(\tau \mid \{\tau_0, \kappa_0\}, \{\tau_{\mathrm{M}}, \kappa_{\mathrm{M}}\}\big) \tag{3.23}$$

2: $\boldsymbol{\Delta} = \{\Delta_i\}$ with $\Delta_i = \kappa_i - \boldsymbol{\ell}(\tau_i)$
3: $i^* = \{i \mid i \in [0, \mathrm{M}] \ \wedge \ \Delta_i \geq \Delta_r, \ \forall \Delta_r \in \boldsymbol{\Delta}\}$
4: **if** $\Delta_{i^*} <$ Tol **then**
5: **return** $\mathbf{I} = \{0, \mathrm{M}\}$
6: **end if**
7: Define a sequence $\mathbf{I}_l = \{0, \ldots, i^*\}$ using Algorithm 2 with input $\boldsymbol{\Delta}_l = \{\Delta_0, \ldots, \Delta_{i^*}\}$, $\boldsymbol{\tau}_l = \{\tau_0, \ldots, \tau_{i^*}\}$ and Tol.
8: Define a sequence $\mathbf{I}_r = \{i^*, \ldots, \tau_{\mathrm{M}}\}$ using Algorithm 2 with input $\boldsymbol{\Delta}_r = \{\Delta_{i^*}, \ldots, \Delta_{\mathrm{M}}\}$, $\boldsymbol{\tau}_r = \{\tau_{i^*}, \ldots, \tau_{\mathrm{M}}\}$ and Tol.
9: **return** $\mathbf{I} = \{0, \ldots, i^*, \ldots, \mathrm{M}\} = \mathbf{I}_l \cup \mathbf{I}_r$

sequence of initial and final frames with no interpenetration between polygons. Therefore, the following subsets are defined:

$$\boldsymbol{\kappa}' = \{\kappa_l, \ldots, \kappa_r\} \subseteq \boldsymbol{\kappa}$$
$$\boldsymbol{\tau}' = \{\tau_l, \ldots, \tau_r\} \subseteq \boldsymbol{\tau}$$

with

$$l = \min(i \mid \kappa_i \neq 0, \kappa_i \in \boldsymbol{\kappa}) - 1$$
$$r = \max(i \mid \kappa_i \neq 0, \kappa_i \in \boldsymbol{\kappa}) + 1$$

Then, the upper envelope of $\boldsymbol{\kappa}'$ is computed with Algorithm 2.

The solution adopted results in a unimodal lower polyline, that reduces the number of channel break points compared to how many would be required to follow the profile more accurately. The main advantage is that a lower number of constraints is needed to set the problem detailed in Section 2.7, which increases the computational performances. Furthermore, from the point of view of this application, it is convenient to limit the possible fluctuations of the resulting spline by defining a channel that guides towards this solution. Algorithm 2 takes as input the overlap vector $\boldsymbol{\kappa}$, the sequence $\boldsymbol{\tau}$ and a tolerance parameter Tol, returning a sequence of indices $\mathbf{I}$ corresponding to the selected points $\{\kappa_i, \tau_i\}$, $\forall i \in$

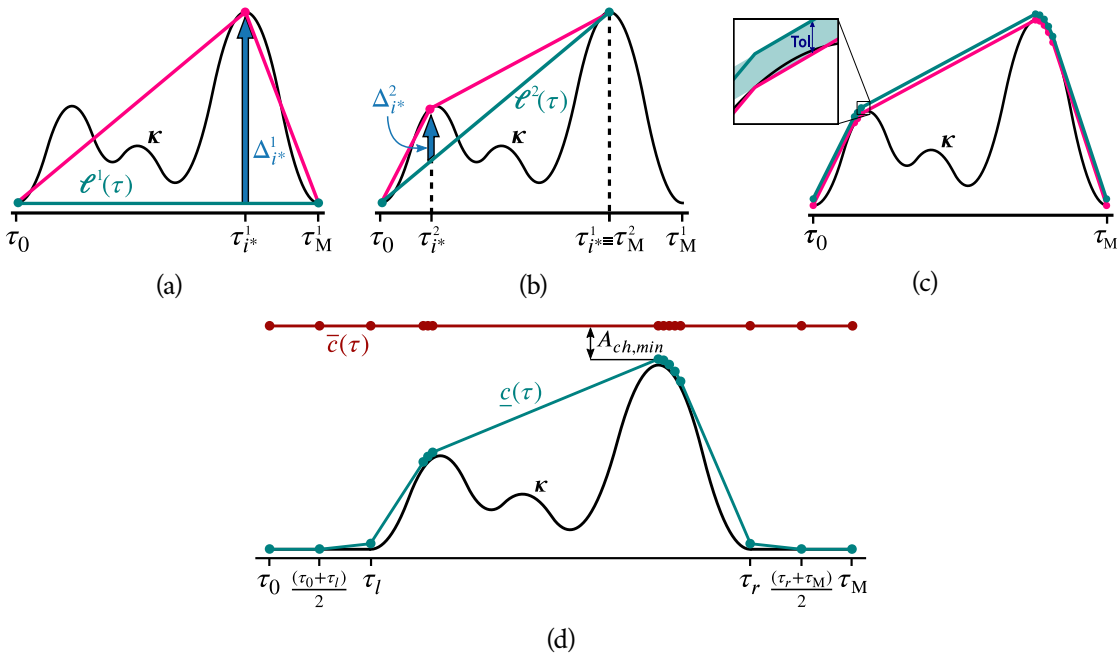


Figure 3.10: Upper envelope evaluation algorithm. (a) 1st step of Algorithm 2. (b) 2nd step of Algorithm 2. (c) Tolerance parameter translation. (d) Channel definition from overlap vector $\boldsymbol{\kappa}$, according to equations (3.24)

I. With reference to Fig. 3.10a and Fig. 3.10b, these points are selected recursively by searching for the maximum value of the sequence $\boldsymbol{\Delta}$, which is obtained by subtracting the values of the linear function (3.23), calculated in $\boldsymbol{\tau}$, from the $\boldsymbol{\kappa}$ vector. Recursion terminates on condition $\Delta_i \leq \text{Tol}, \forall \Delta_i \in \boldsymbol{\Delta}$.

Given the indices sequence $\mathbf{I} = (\mathrm{I}_j)_{j=0}^{n}$, the lower polyline $\underline{\boldsymbol{c}}(\tau)$ is defined as follows (refer to Fig. 3.10c):

$$
\begin{aligned}
&\underline{c}_0 = \underline{c}_1 = \underline{c}_{n+3} = \underline{c}_{n+4} = 0 \\
&\underline{c}_{j+2} = \kappa'_{I_j} + \text{Tol}, \quad j = 0, \ldots, n \\
&x_0 = \bar{u}_0, \quad x_{n+4} = \bar{u}_{m+6} \\
&x_1 = (\bar{u}_0 + \tau_l)/2, \quad x_{n+3} = (\bar{u}_{m+6} + \tau_r)/2 \\
&x_{j+2} = \tau'_{I_j}, \quad j = 0, \ldots, n \\
&\underline{c}(\tau) = \mathcal{L}\big(\tau | \{x_i, \underline{c}_i\}, \{x_{i+1}, \underline{c}_{i+1}\}\big) \text{ with } x_i \leq \tau \leq x_{i+1}, \quad \forall i \in [0, n+4]
\end{aligned}
\tag{3.24}
$$

Let $A_{ch,min}$ be the desired minimum channel amplitude the upper polyline $\bar{c}(\tau)$ adopted

in this work is defined by the coefficients:

$$\overline{c}_j = \max(\underline{c}_i) + A_{ch,min}, \quad i = 0, \ldots, n+4, \quad j = 0, \ldots, n+4 \tag{3.25}$$

Note that for a constant upper polyline $\overline{c}(\tau)$, such as that defined by (3.25), the inequality constraints (2.37) and (2.39) reported in Section 2.7 could be greatly simplified. However, it is preferred to report the general discussion in case a different formulation from (3.25) would be adopted.

3.2.5 Collision avoidance strategy

To avoid the collisions that can occur during the movement, this section proposes an approach that modifies the control points of the spline $\bar{\boldsymbol{r}}(\tau)$, defined in the Section 3.2.1, which describes the geometric path of the two end-effectors. This technique is based on the calculation of deviation splines using the algorithm described in Section 2.7, with the channel defined according to 3.2.4. Depending on whether the vector of Y-intersections is defined by (3.20), (3.21) or (3.22), the 5$^{\text{th}}$-degree 1-D spline, defined by (2.1), is named as follows:

- Overlap spline: $\boldsymbol{\sigma}(\tau)$ (3.26)
- Top Overrun spline: $\overline{\boldsymbol{\sigma}}(\tau)$ (3.27)
- Bottom Overrun spline: $\underline{\boldsymbol{\sigma}}(\tau)$ (3.28)

With control points

$$\boldsymbol{h} = (h_i)_{i=0}^{q}, \quad \underline{\boldsymbol{h}} = (\underline{h}_i)_{i=0}^{\underline{q}}, \quad \overline{\boldsymbol{h}} = (\overline{h}_i)_{i=0}^{\overline{q}} \tag{3.29}$$

and knots

$$\boldsymbol{k} = (k_i)_{i=0}^{q+6}, \quad \underline{\boldsymbol{k}} = (\underline{k}_i)_{i=0}^{\underline{q}+6}, \quad \overline{\boldsymbol{k}} = (\overline{k}_i)_{i=0}^{\overline{q}+6} \tag{3.30}$$

defined by the procedure reported in Section 2.7.3.

Referring to $\boldsymbol{\sigma}(\tau)$, the spline can be summed to the two components along the Y-axis projections of $\bar{\boldsymbol{r}}(\tau)$, first by making the functions compatible by knot refinement, then by summing the relative control points using the weights w_1 and w_2 [30]:

$$\begin{aligned} \bar{y}_{i,1} &\leftarrow \bar{y}_{i,1} - w_1 h_i \\ \bar{y}_{i,2} &\leftarrow \bar{y}_{i,2} + w_2 h_i \end{aligned} \tag{3.31}$$

Referring to equations (3.14) and (3.15), be:

$$\Delta y^{+} = \left\{ |\overline{\kappa}_i| \;\middle|\; \overline{\kappa}_i \leq 0 \;\wedge\; \overline{\kappa}_i \geq \overline{\kappa}_j,\ \forall i, j \in [0, N] \right\}$$
$$\Delta y^{-} = \left\{ |\underline{\kappa}_i| \;\middle|\; \underline{\kappa}_i \leq 0 \;\wedge\; \underline{\kappa}_i \geq \underline{\kappa}_j,\ \forall i, j \in [0, N] \right\}$$

$$\sigma_{max} = \max(\{\boldsymbol{\sigma}(\bar{u}_i) \mid \forall \bar{u}_i \in \bar{\boldsymbol{u}}\})$$
$$w_1^* = \Delta y^- / \sigma_{max}$$
$$w_2^* = \Delta y^+ / \sigma_{max}$$

The weights w_1 and w_2 of equations (3.31) can be calculated through:

$$\begin{cases} w_1 = w_2 = 0.5, & \text{if } w_1^* \geq 0.5 \ \wedge \ w_2^* \geq 0.5 \\ \begin{cases} w_1 = w_1^* \\ w_2 = (1 - w_1) \end{cases} & \text{if } (w_1^* < 0.5 \ \vee \ w_2^* < 0.5) \wedge w_1^* < w_2^* \\ \begin{cases} w_2 = w_2^* \\ w_1 = (1 - w_2) \end{cases} & \text{otherwise} \end{cases}$$

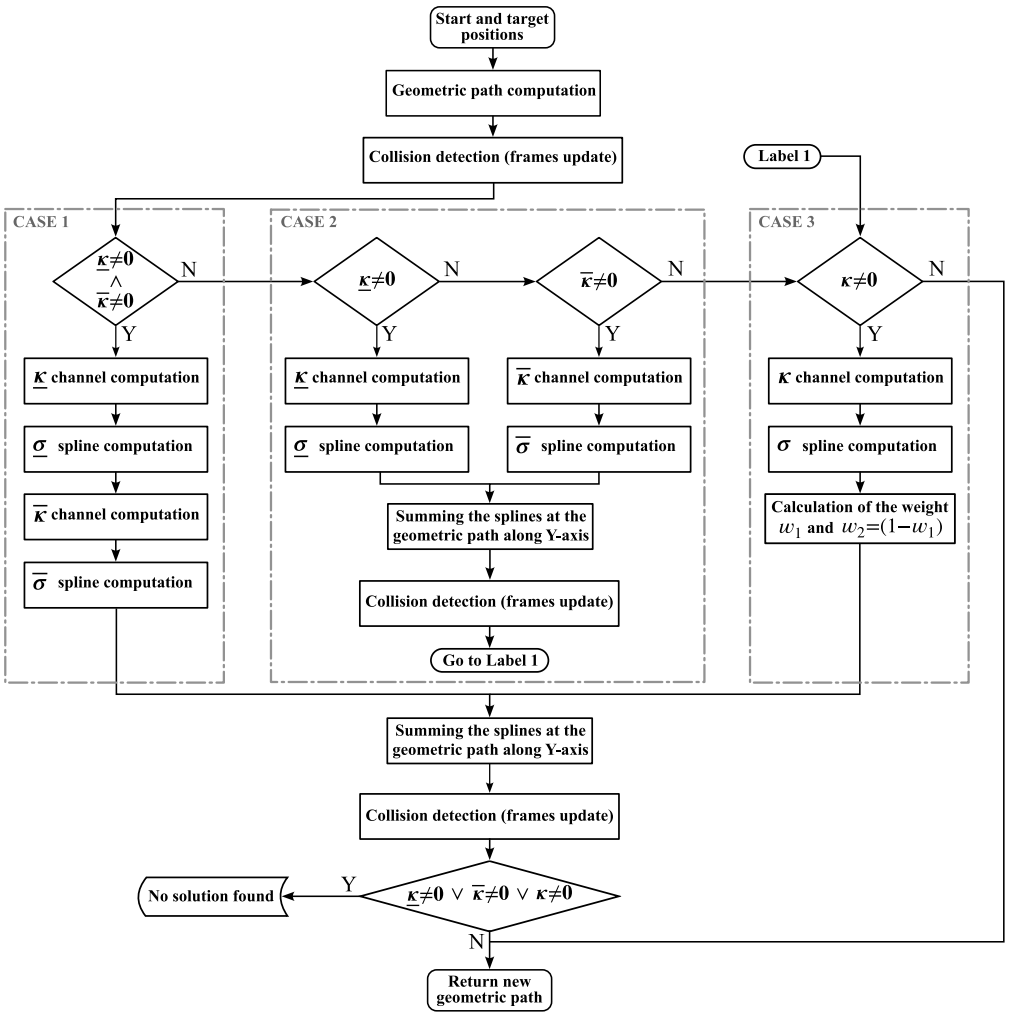


Figure 3.11: Pipeline of the collision-free path planning strategy.

More generally, depending on whether the deviation spline is $\boldsymbol{\sigma}$, $\overline{\boldsymbol{\sigma}}$ or $\underline{\boldsymbol{\sigma}}$, the weights w_1 and w_2 are calculated as follows:

- Overlap spline $\boldsymbol{\sigma}$: Eqn. (3.2.5)
- Top overrun spline $\overline{\boldsymbol{\sigma}}$: $w_1 = 0, \quad w_2 = 1$
- Bottom overrun spline $\underline{\boldsymbol{\sigma}}$: $w_1 = 1, \quad w_2 = 0$

Fig. 3.11 reports the pipeline of the proposed collision-free path planning strategy. First the geometric path is calculated using the strategy of Section 3.2.1 from the start and target positions, assumed to be free of self-collisions. These positions are calculated using a work zone partitioning approach similar to that used in [19], and is not dealt with in this work. Subsequently, the calculated path must be post-processed in order to ensure collision-free movement in tasks involving the rotation of loaded parts. To this end, a frame analysis is performed using the method proposed in Section 3.2.2 by determining the $\boldsymbol{k}$, $\underline{\boldsymbol{k}}$, and $\overline{\boldsymbol{k}}$ vectors. The pipeline is then divided into three cases according to the vectors obtained:

- Case 1: $\overline{\boldsymbol{\kappa}} \neq \mathbf{0} \wedge \underline{\boldsymbol{\kappa}} \neq \mathbf{0}$. Both overrun vectors have values greater than zero. In this case, both groups do not meet the constraints of the work zone. $\overline{\boldsymbol{\sigma}}$ and $\underline{\boldsymbol{\sigma}}$ are then calculated according to the approach described in Section 3.2.4 and added to the respective components of $\overline{\boldsymbol{r}}(\tau)$ as described in this section.

- Case 2: $\overline{\boldsymbol{\kappa}} \neq \mathbf{0} \vee \underline{\boldsymbol{\kappa}} \neq \mathbf{0}$. An overrun vector has values greater than zero. Depending on which group does not meet the work zone constraint, the appropriate overrun spline is calculated and added to the relevant component of $\overline{\boldsymbol{r}}(\tau)$. After that, the calculation of the $\boldsymbol{\kappa}$ vector is updated, and the process is iterated again.

- Case 3: $\overline{\boldsymbol{\kappa}} = \mathbf{0} \wedge \underline{\boldsymbol{\kappa}} = \mathbf{0} \wedge \boldsymbol{\kappa} \neq \mathbf{0}$. Only the overlap curve has values greater than zero. $\boldsymbol{\sigma}$ is then calculated according to the approach described in Section 3.2.4 and summed by means of the weights w_1 and w_2 to the $\bar{\boldsymbol{r}}(\tau)$ Y-axes components.

If none of the above cases apply, the spline $\bar{\boldsymbol{r}}(\tau)$ is returned without any variation. In this specific application case, this occurs about 50% of the time, i.e. during the loading phase. Otherwise, if one of the above cases occurs, an additional collision check must be performed. Should this check detect any violation of the constraints imposed by the work zone, or detect any collision, a solution cannot be found for this specific configuration of start and target points. If this case accours, another configuration of start and target points must be set; on the contrary, if the check is passed, the new collision-free spline $\bar{\boldsymbol{r}}(\tau)$ is returned.

In the case where the pipeline in Fig. 3.11 returned a modified geometric path, in line with the proposed approach its parameterisation must be corrected. The detailed collision avoidance strategy allows at most two deviation curves to be determined, to be applied to the Y-components of the geometric paths of the two end-effectors respectively. Equation (3.13) can still be used, provided that the contribution of the calculated deviation splines is

considered. In this work, if only one deviation curve is returned by the procedure of Fig. 3.11, it is added to the Y-component of the master path before calculating s_{max}. Considering e.g. the overlap spline $\boldsymbol{\sigma}(\tau)$, after making the $\boldsymbol{r}_\phi(\tau)$ and $\boldsymbol{\sigma}(\tau)$ compatible with knot refinement, the contribution is added summing the control points with the relation:

$$\bar{y}_{i,\phi} \leftarrow \bar{y}_{i,\phi} + h_i, \quad i = 0, \ldots, m$$

with m the maximum index of control points after knot refinement operation. In the case where the collision avoidance routine returns two deviation curves, the spline that makes a greater contribution to the calculation of s_{max} is selected.

3.3 Dual-gantry system path

This section elaborates on the solution employed for a dual-gantry machine configuration. Initially, it explores the approach used to ensure collision-free motion in the X-direction, aiming to prevent collisions between the two gantry systems. Following this, the integration of this methodology into the motion calculation logic is explained. Finally, a detailed account of the approach adopted to roto-translate parts loaded by two end-effectors is presented.

3.3.1 14-d.o.f. dual-gantry system path planning

In the machine configuration with four end-effectors, implementing motion control is essential to prevent collisions between the two gantry systems. The approach used to guarantee collision-free motion in the Y-direction for each gantry system can be readily modified to adjust the geometric path of both systems in the X-direction. The variations in this routine compared to the one described in Section 3.2.5 are purely implementational, requiring no additional theoretical development.

The flow-chart of the motion management is reported in Fig. 3.12, which shows that the path planning is processed in three sequential steps as follows:

- *Free path planning generation*. During this process, the path is generated for all heads considering only the starting and target positions of each. Here the collisions and the heads constraints are not considered, and each head is free to move along a XY straight line.

- *Roto-translation*. Before the avoidance computation, the roto-translation is applied to those heads that need it. In this case the position of the two picking heads is computed through the interpolation procedure detailed in Section 2.6. The solution adopted for roto-traslation task is detailed in Section 3.3.2.

- *Avoidance*. The avoidance is processed according with heads constraints. In general, the avoidance is initially computed in Y-direction and then in X-direction. If a solution is found the path is considered as valid and it is possible to procced to the motion profile computation.

It is essential to note that the constraints imposed by the workspace are no longer dictated solely by the maximum and minimum values of Y that can be adopted by the end-effector. In addition to these, constraints defined by the maximum and minimum values of X will also be considered. The outcome is to achieve a workspace accessible by the four end effectors, defined by a rectangle.

Another aspect that complicates the logic of the program is that the openings in the Y-direction are not independent of each other. The Y-avoidance is the part of the software that modifies the previously computed free paths on a single gantry to avoid any Y-collision between parts, heads, and tools. This phase of the avoidance must deal with two gantries and the various constraints between the relative head groups. For this reason, some analysis must be sequenced and depending on the results, the method changes. For this purpose, the Y-avoidance manager has been implemented to make decisions about how to execute the avoidance.

Fig. 3.13 resumes the management of the Y-avoidance computation and it is composed by the following main steps:

- *Slave and master heads definition and constraint analysis.* The first step consists in defining the master and slave heads. For this purpose, initially the gantry on the left is considered the master and the gantry on the right the slave. With reference with the numeration reported in Fig. 3.14, the top heads are the master and the bottom heads the slaves.

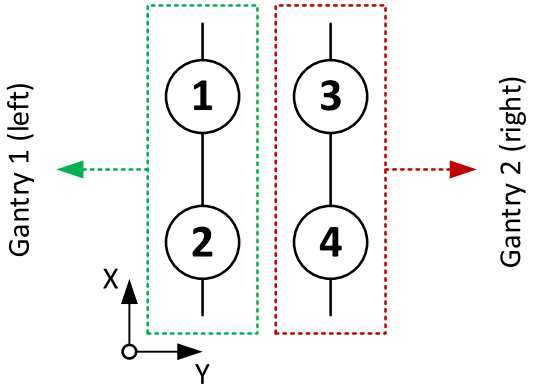


Figure 3.14: Head numeration.

Once that the master and slave heads are defined, the constraints are analyzed to define which heads are free to move, and which heads are locked by their respective

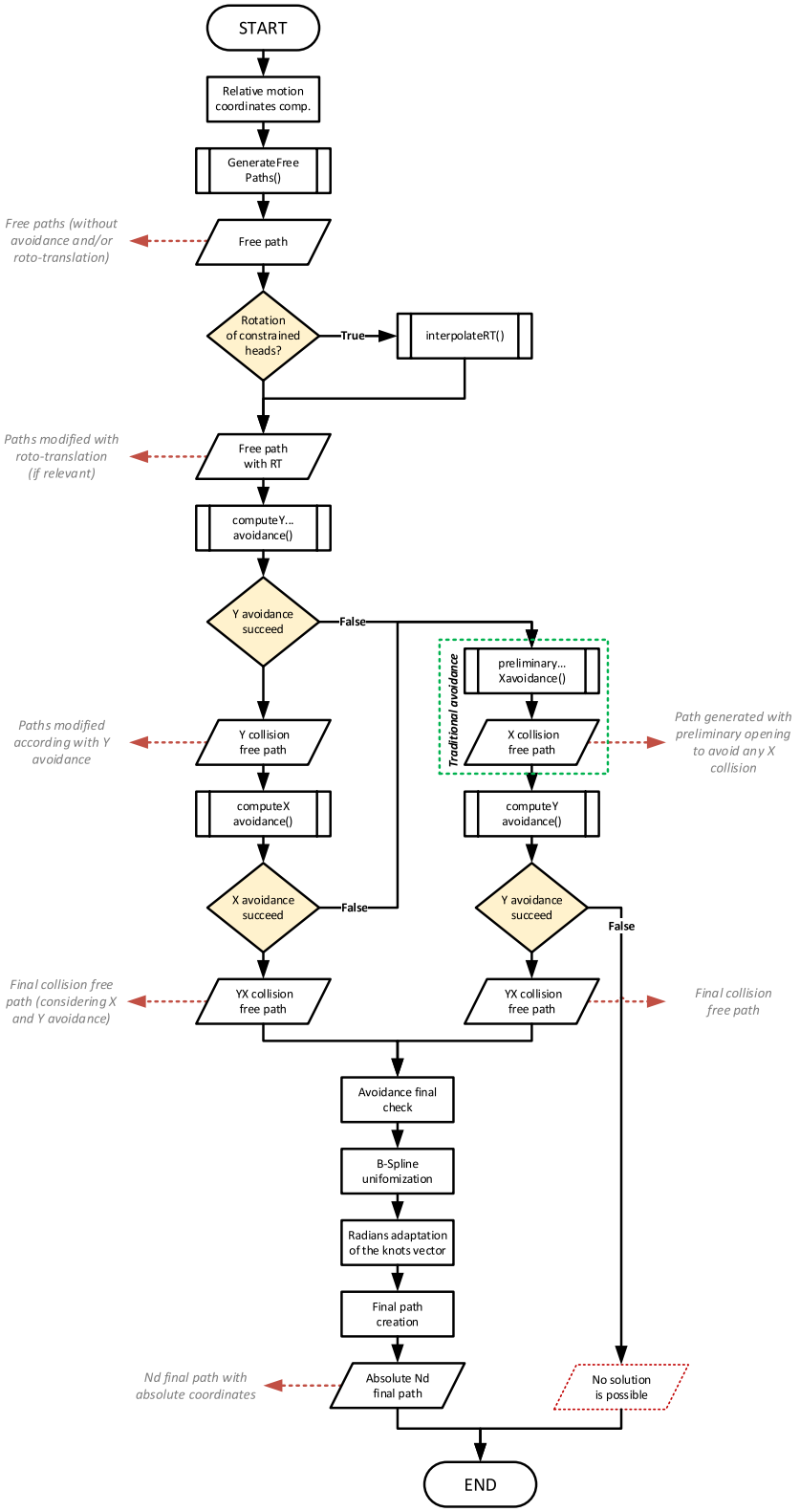


Figure 3.12: Flow-chart of the motion management for 14-d.o.f. machine configuration.

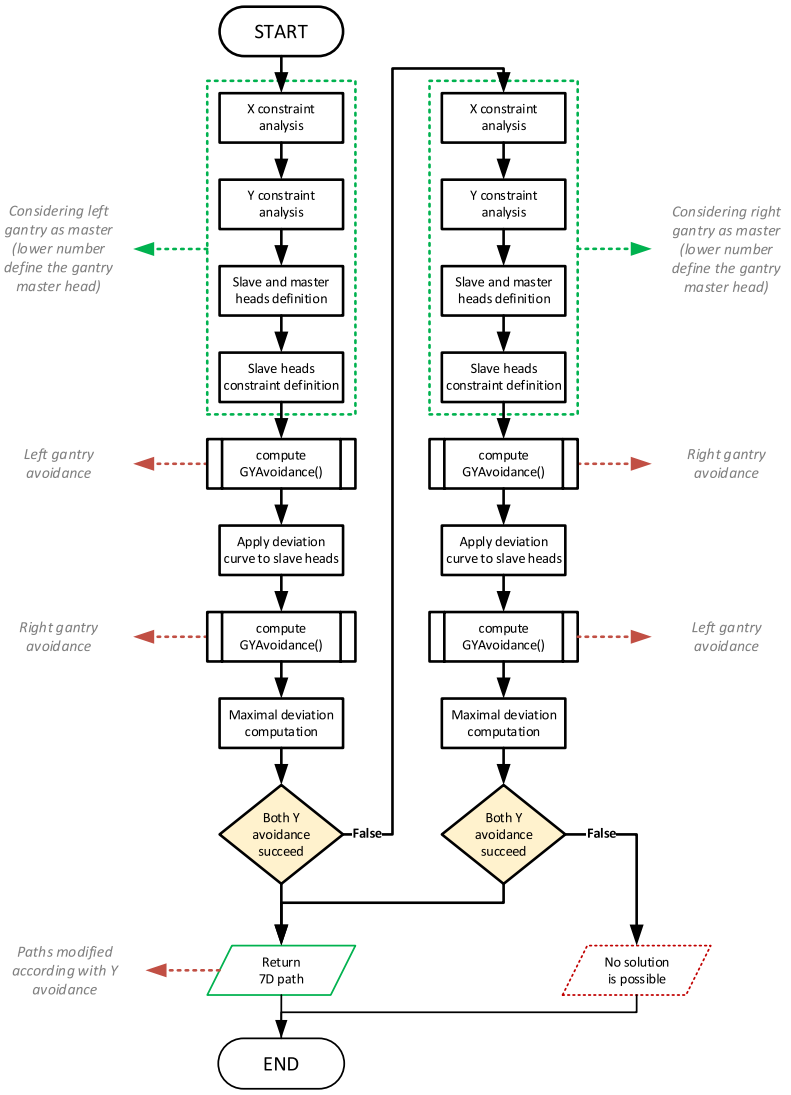


Figure 3.13: Flow-chart of the Y-avoidance management.

master heads.

- *Master gantry avoidance computation.* This step consists in computing the avoidance on the master gantry and apply the resulting deviation to the slave groups heads to transfer the movement to the heads that are rigidly constrained by a common loaded part.

- *Slave gantry avoidance computation.* This step consists in computing the avoidance on the slave gantry. In this case, slave heads constrained with the master gantry are locked and the only d.o.f. consists in moving those heads that are free to move.

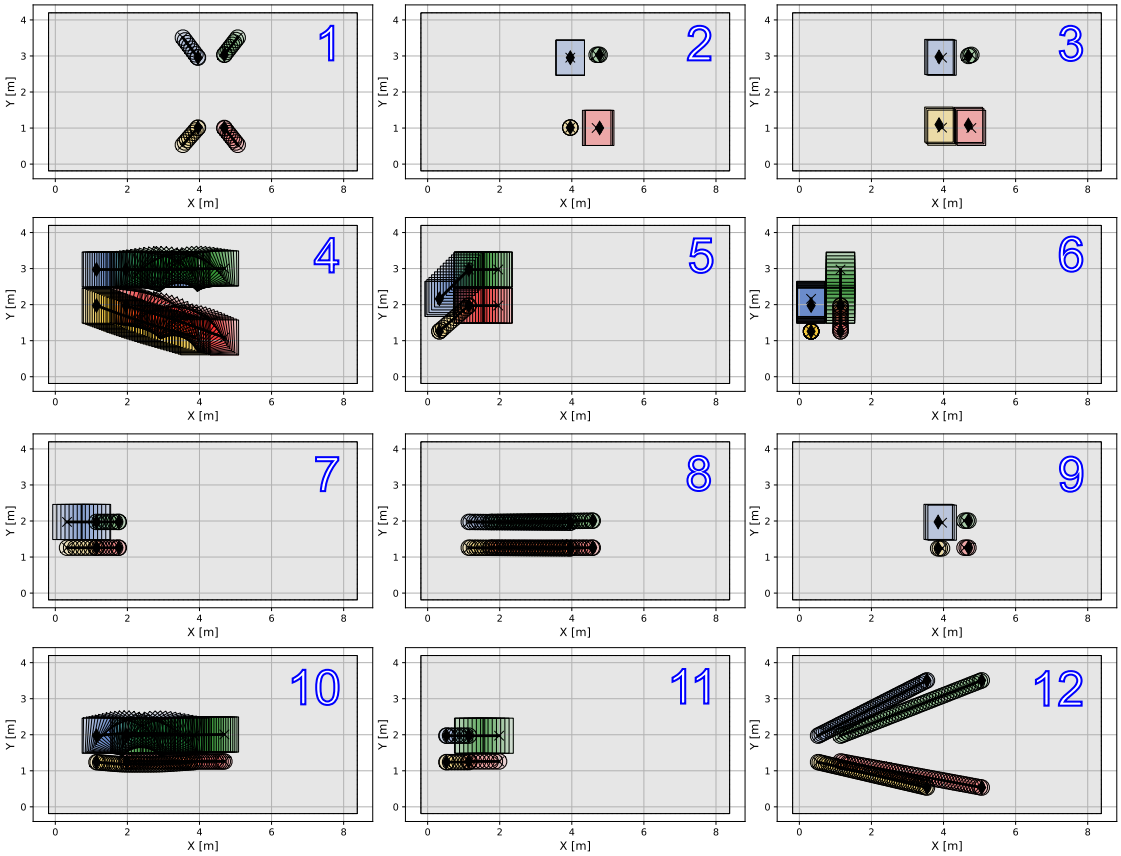
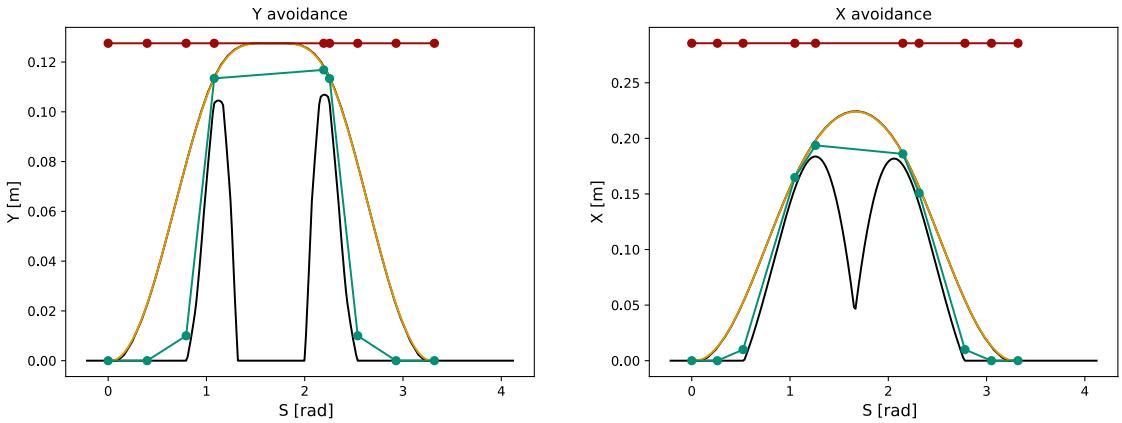


Figure 3.15: Example of job macro list solved by the presented method.

- *Final check of collision free trajectory.* Usually, the collision free trajectory should be obtained considering the left gantry as master. Nevertheless, in rare conditions, a solution cannot be found. In these cases, the same procedure is executed but considering the right gantry as master. On the analyzed job-lists, this was necessary rarely and allowed to manage all the submitted case.

In order to provide a case study of the proposed path planning approach in the case of the dual-gantry machine configuration, bounding box frames and related 2-D projections of the end-effectors' trajectories for an entire job macro list are shown in Fig. 3.15. The related job macro list can be found in Appendix B. Fig. 3.16 also shows the overlap curves calculated by the algorithm detailed in Section 3.2.5 for movement 10. Fig. 3.16a reports the Y-direction overlap spline for the first gantry system. Fig. 3.16a instead reports the X-direction overlap spline. The reported cases are demonstrative of how Algorithm 2 defines a channel consisting of a unimodal lower polyline, guiding the minimisation algorithm to define a fluctuation-free overlap curve.



(a) (b)

Figure 3.16: Overlap splines for movement 10 of Fig. 3.15. (a) Y-direction overlap spline for the 1st gantry system. (b) X-direction overlap spline.

3.3.2 Roto-traslation solution

One of the advantages of the dual-gatry machine configuration is the ability to roto-translate parts loaded with two end-effectors simultaneously. Two gripping points reduce the deflection of the part and improve its stability during movement, in addition to the consideration that a part might exceed the maximum weight that can be lifted by a single arm. This section details the numerical solution adopted for this task.

The roto-translation movement results in the addition of a rigid connection between two end-effectors installed on different gantries. Referring only to the XY plane of the machine, the geometric paths of the two end-effectors are defined as streams of 2-D points using an appropriate discretisation method. These points are then interpolated using the fitting method detailed in Section 2.6. The discretisation method adopted must allow the approximation error of the interpolation technique to be less than the positioning tolerance of the machine. Once the two interpolating 2-D splines have been defined, they can be substituted for the corresponding dimensions of the 14-D geometric path defined in the previous section, following a knot refinement operation [13].

With reference to Fig. 3.17a, the simplest way to solve the problem is to constraint one of the end-effectors to follow a linear geometric path. However, this approach is not optimal in most sorting tasks due to the occurrence of cusps in the unconstrained head path. This approach would then result in the presence of curvature spikes and a consequent deterioration in machine performance. A more effective method, and the one used in this work, is to constrain the midpoint of the segment connecting the two gripping positions to follow a linear path, as shown in Fig. 3.17b.

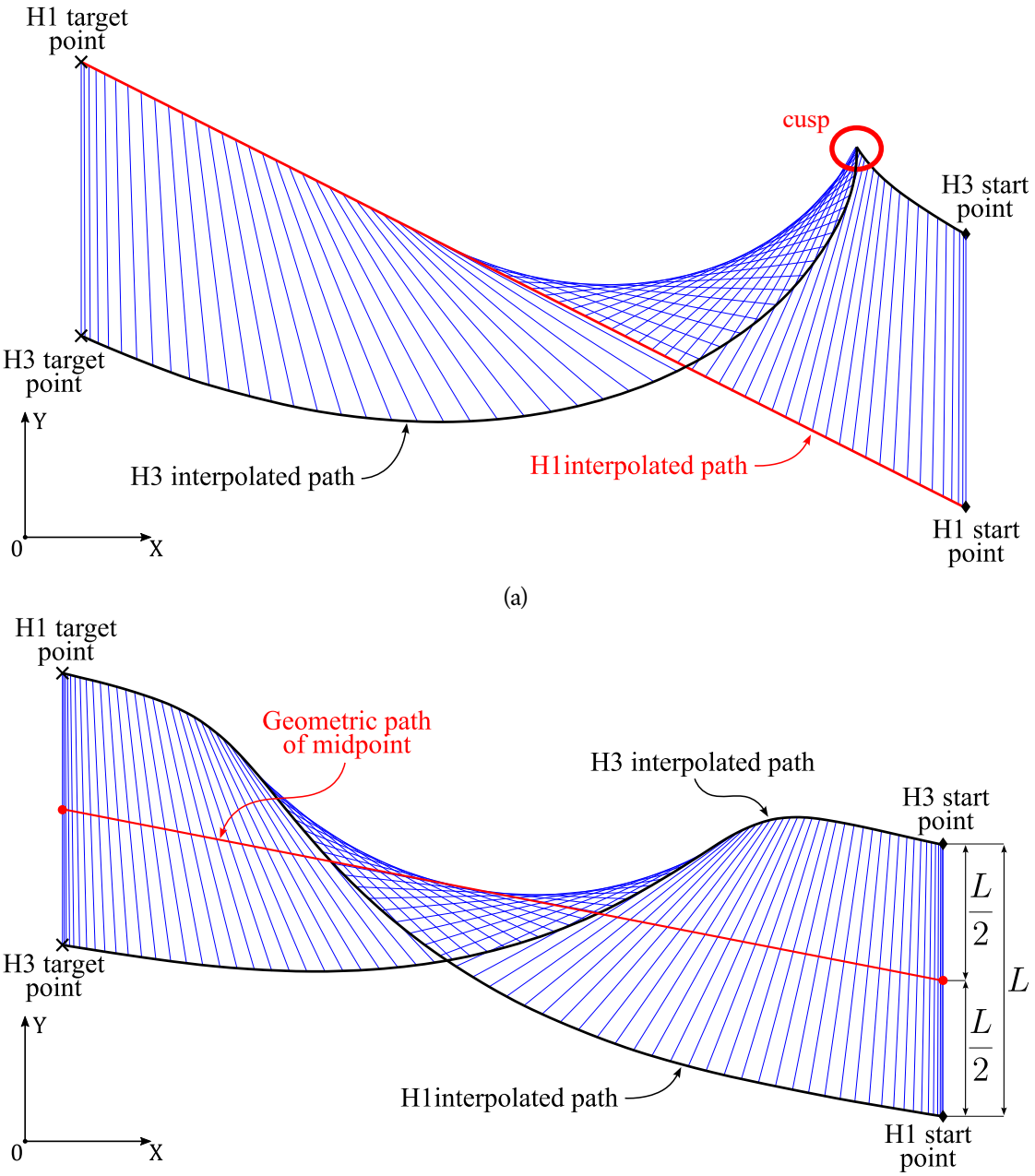
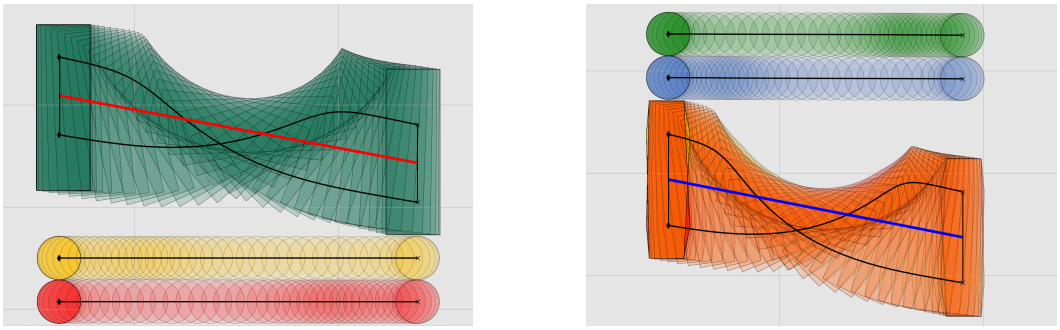


Figure 3.17: (a) Roto-traslation with $180°$ rotation obtained by constraining head 1 to a linear geometric path. (b) Roto-traslation with $180°$ rotation obtained by constraining the midpoint of the segment connecting the two gripping points to a linear geometric path.



(a) (b)

Figure 3.18: Two examples of roto-translation performed with two end-effectors of different gantry systems. In both cases the rotation is $180°$, while the other heads are free and modelled with a circle.

In this way, even in the worst cases, i.e. those corresponding to $180°$ rotations, the curvature of the geometric path will fall within an acceptable range.

The anti-collision strategy in the roto-traslation tasks does not differ much from that detailed in the previous sections. Specifically, the polygons involved in the collision detection algorithm (Section 3.2.2) are three instead of four. Two of these are calculated in the same way as in the previous sections, while the third is calculated in a similar way but considering the bounding boxes of the part and both heads being taken during the convex hull algorithm. The calculated deviation splines are then added to the heads not involved in the multiple loading, keeping the splines obtained by interpolation unchanged. Two examples of roto-traslation tasks in a real application are shown in Fig. 3.18. A sequence of frames from a real roto-translation task is shown in Fig. 3.19.

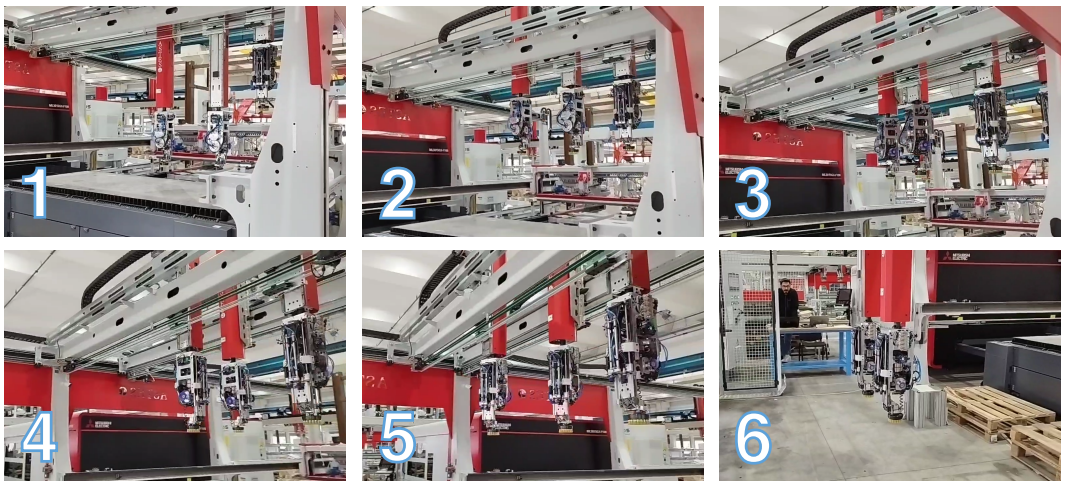


Figure 3.19: Frame sequence of a real roto-traslation task performed by the test machine.

Chapter 4

Motion profile calculation

4.1 Iso-Parametric Trajectory Planning

IPTP is a trajectory planning paradigm introduced in a previous author work [28], by which reusable I/O blocks, called macro-instructions, are defined to set kino-dynamic constraints. Subsequently, these macro-instructions can be combined with each other to obtain the final trajectory. In [28], a framework based on this paradigm is described, where each piece of trajectory is defined by a 1-D time-parametrized B-spline, and modified with object-oriented approach using computer-aided design techniques, such as scaling, translation, splitting and concatenation. The main objects class used in the framework is Trajectory1D, whose instances define pieces of trajectory by storing the spline parameters, which can be obtained through the public attributes $t[0..(m+d+1)]$, $c[0..m]$ and d, for knots, control points and degree, respectively. Another class, useful to store kinematic constraints, is Point1D, which allows the storage of a value associated with a time instant for a specific derivation order. Even for instances of this class, it is possible to obtain these parameters via public attributes, namely val, t and ord. This framework allows the trajectory designer to easly test different solutions before determining the most effective approach for the specific application, and is suitable both a joint space (or coordinate-based) and Cartesian space (or path-based) motion planning approach. In this section, to add time information to the previously defined geometric path, a motion profile planner using this framework is proposed.

In the remainder of this section, some notations from the previous work are adopted; in this regard, please refer to [28].

4.2 Solution adopted

Let $\boldsymbol{s}(t)$ be a 1-D time-parametrized spline of degree d, defined by (2.1), with unknown control points $\boldsymbol{k} = (k_i)_{i=0}^{m}$ and knots $\boldsymbol{t} = (t_i)_{i=0}^{m+d+1}$. Furthermore, $\boldsymbol{s}(t)$ is strictly increasing

in the domain range $[t_0, t_{m+d+1}]$, then:

$$\boldsymbol{s}(t_b) > \boldsymbol{s}(t_a) \quad \forall\, t_b > t_a\,, \quad \text{with: } t_b, t_a \in [t_0, t_{m+d+1}]$$

Let position, velocity, acceleration and jerk of the axes be defined by a 14-D vector function $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{a}, \boldsymbol{j} : \mathbb{R} \to \mathbb{R}^7$ in the time variable t. Adopting $\boldsymbol{s}(t)$ as the motion profile along the geometric path described in Section 3.2.1, it follows:

$$\boldsymbol{p}(t) = \{\boldsymbol{p}_i(t)\} = (\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t) \tag{4.1}$$
$$\boldsymbol{v}(t) = \{\boldsymbol{v}_i(t)\} = \dot{\boldsymbol{s}}(t)(D\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t) \tag{4.2}$$
$$\boldsymbol{a}(t) = \{\boldsymbol{a}_i(t)\} = \ddot{\boldsymbol{s}}(t)(D\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t) + \dot{\boldsymbol{s}}^2(t)(D^{(2)}\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t) \tag{4.3}$$
$$\begin{aligned}\boldsymbol{j}(t) = \{\boldsymbol{j}_i(t)\} = {} & \dddot{\boldsymbol{s}}(t)(D\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t) + 3\dot{\boldsymbol{s}}(t)\ddot{\boldsymbol{s}}(t)(D^{(2)}\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t) + \\ & \dot{\boldsymbol{s}}^3(t)(D^{(3)}\bar{\boldsymbol{r}} \circ \boldsymbol{s})(t)\end{aligned} \tag{4.4}$$

Defining VC_i and AC_i, $i = 0, \ldots, 6$, as the maximum velocity and acceleration limits of the i^{th}-axis, the problem of planning the time-optimal motion profile can be defined in the unknown vectors $\boldsymbol{t}$ and $\boldsymbol{k}$ as follows:

Constraints:

$$\begin{aligned}&\boldsymbol{s}(t_0) = 0, \quad \boldsymbol{s}(t_{m+d+1}) = s_{max} \\ &\dot{\boldsymbol{s}}(t_0) = 0, \quad \dot{\boldsymbol{s}}(t_{m+d+1}) = 0 \\ &\ddot{\boldsymbol{s}}(t_0) = 0, \quad \ddot{\boldsymbol{s}}(t_{m+d+1}) = 0 \\ &\max_{t\in[t_0,t_{m+d+1}]} \left|\boldsymbol{v}_i(t)\right| \leq VC_i, \quad \text{for } i = 0, \ldots, 6 \\ &\max_{t\in[t_0,t_{m+d+1}]} \left|\boldsymbol{a}_i(t)\right| \leq AC_i, \quad \text{for } i = 0, \ldots, 6\end{aligned} \tag{4.5}$$

Cost function:

$$\text{minimize } T = \sum_{i=0}^{m+d}(t_{i+1} - t_i) = \sum_{i=0}^{m+d}\Delta t_i \tag{4.6}$$

Fig. 4.1 reports the pipeline of the proposed solution for the motion profile calculation. The strategy adopted first defines a symmetrical motion profile startig from the geometric path $\bar{\boldsymbol{r}}(\tau)$, within the kinematic limits (4.5), using Algorithm 3. With reference to Fig. 4.2, the algorithm is parameterised on a Trajectory1D instance J_{in} that defines a positive acceleration impulse in the jerk order. In this application, J_{in} is defined by a cubic spline with the following knots and control points:

$$\begin{aligned}&\text{knots:} \quad \{0,\ 0.25A_{min},\ 0.5A_{min},\ 0.75A_{min},\ A_{min}\} \\ &\text{control points:} \quad \{0, J^{peak}, J^{peak}, J^{peak}, -J^{peak}, -J^{peak}, -J^{peak}, 0\}\end{aligned}$$

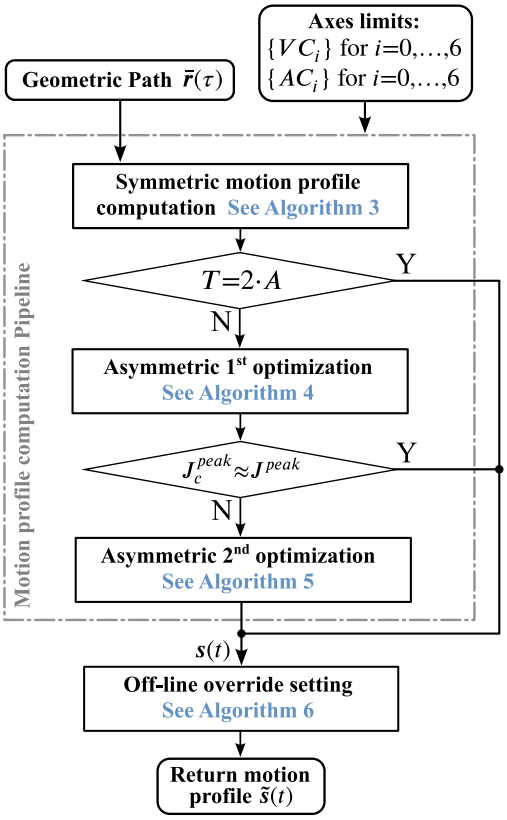


Figure 4.1: Pipeline of the motion profile computation.

with A_{min} the initial period of the acceleration section, and J^{peak} the upper limit imposed in absolute value on $\dddot{\boldsymbol{s}}(t)$. Note that the latter condition leads to limit the codomain of the elements of (4.4), in other words:

$$\exists k > 0 : |\boldsymbol{j}_i(t)| \leq k \quad \forall t \in [t_0, t_{n+d+1}], \quad \text{for } i = 0, \ldots, 6 \tag{4.7}$$

In Algorithm 3, the motion profile is defined with three sections, the first of acceleration, the second of constant velocity and the last of deceleration. In the jerk order these sections are referred to as J_l, J_c and J_r, respectively. J_l is obtained in the first iteration as a simple copy of J_{in}. The main mechanism of Algorithm 3 consists in iteratively enlarging J_l, of period A, by a parameter ε, defining the motion profile for each iteration concatenating J_l with a null Trajectory1D instance J_c and a third Trajectory1D instance J_r, obtained by mirroring J_l. The period of J_c, called A_c, is calculated for each iteration to set the condition s_{max}. The overall period of the motion profile is therefore equal to $T = 2A + A_c$, and this becomes shorter during iterations of ΔT. The algorithm terminates when one or more of the limits VC_i and AC_i are reached by the respective axes within a tolerance Tol. The function checkLimitsOnAxes is used to detect when the limits are exceed. Firstly it uses a brute-force method that finds the time gridpoint at which the greatest absolute value of the functions (4.2) and (4.3) occurs. In this application, an equispaced time grid of N samples is

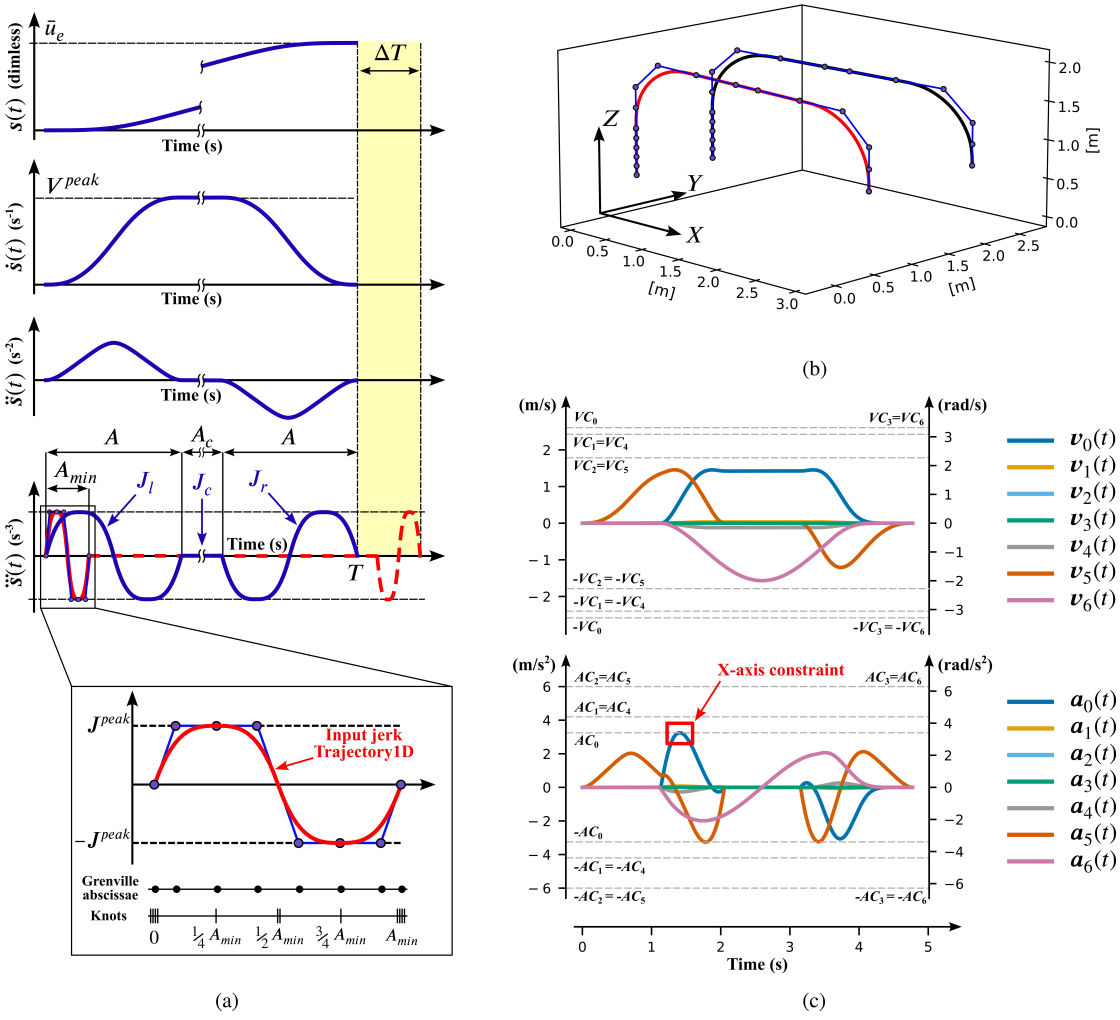


Figure 4.2: Schematic representation of the symmetric motion profile generation algorithm reported in Algorithm 3. (a) Motion profile and input jerk Trajectory1D. (b) Geometric path $\bar{\boldsymbol{r}}(\tau)$ of reference. (c) Joint trajectories obtained with (4.2) and (4.3). The algorithm is terminated for reaching the acceleration limit AC_0 on X-axis.

Algorithm 3 Symmetric motion profile

Input:

1. Geometric path described from 7-D spline $\bar{\boldsymbol{r}}(\tau)$, defined in parametric domain $[0, \bar{u}_e]$.
2. Kinematic limits VC_i and AC_i, with $i = 0, \ldots, 6$.
3. Input jerk Trajectory1D object J_{in}.
4. Number of samples N.
5. Tolerance Tol.
6. Initial incremental step ε and step ratio ϕ.

Output:

1. Trajectory1D object S.

Algorithm:

$J_l \leftarrow$Copy of J_{in}
$A \leftarrow A_{min} := |J_l.\boldsymbol{t}[0] - J_l.\boldsymbol{t}[-1]|$
Mirror J_l and set the new instance J_r.
Initialize a null Trajectory1D instance J_c.
repeat
    repeat
        $A \leftarrow A + \varepsilon$
        Adapt J_l in the new domain $[0..A]$.
        Compute $D^{(-2)}J_l$ and evaluate it in A by setting V^{peak}.
        Compute $D^{(-3)}J_l$ and evaluate it in A by setting $\bar{u}_e^*$.
        $A_c \leftarrow (\bar{u}_e - 2 \cdot \bar{u}_e^*)/V^{peak}$.
        if $A_c < 0$ **then**
            Inizialize an empty Point1D object P.
            $P.val \leftarrow \bar{u}_e/2; \quad P.ord \leftarrow 0; \quad P.t \leftarrow A$
            $J_l \leftarrow$setValueEn(J_l, P).
            Mirror J_l and save the new instance J_r.
            Concatenate J_l, J_r, and set J.
            Compute $D^{(-3)}J$ and set S.
            if checkLimitsOnAxes($\bar{\boldsymbol{r}}(\tau)$,$S$,$\{VC_i\}$,$\{AC_i\}$,N) **then**
                break
            end if
            return $S \leftarrow D^{(-3)}J$
        end if
        Adapt J_r in the new domain $[0..A]$.
        Adapt J_c in the new domain $[0..A_c]$.
        Concatenate J_l, J_c, J_r, and set J.
        Compute $D^{(-3)}J$ and set S
    until checkLimitsOnAxes($\bar{\boldsymbol{r}}(\tau)$, S, $\{VC_i\}$, $\{AC_i\}$, N)
    $A \leftarrow A - \varepsilon; \quad \varepsilon \leftarrow \varepsilon \cdot \phi$
until checkLimitsOnAxesWithTol($\bar{\boldsymbol{r}}(\tau)$, S, $\{VC_i\}$, $\{AC_i\}$, N, Tol)
return S

used. A local search method, based on the Golden section method [42], is then applied in the range of the selected grid point defined by the previous and the next point. Note that the value of N must be high enough to ensure that in the range where the local search method is applied the function is unimodal, otherwise the method may return a relative maximum instead of a global maximum. A variant of this function, called checkLimitsOnAxesWithTol, detects instead if these limits are set to less than a tolerance Tol. This approach allows a variable iteration step ε to be used, which leads to acceptable computational performance. When one or more of the limits VC_i and AC_i are exceed, if the most restrictive one falls outside the set tolerance, the algorithm returns to the previous step and restarts the iteration by scaling the incremental step ε of the step ratio ϕ. If none of the kinematic limits are reached during the iterations, the period A_c collapses to zero and the constraint s_{max} is set with the macro-instruction setValueEn (please refer to [28]) on J_l. If the motion profile resulting from the concatenation of J_l with J_r exceeds the kinematic limits of the axes, the iteration is repeated for an iteration step value ε scaled by ϕ. If $A_c = 0$, no further optimisation is possible unless the input parameters of the algorithm are changed, so the calculated motion profile enters in the speed override setting block. On the other hand, if the algorithm is terminated because one of the kinematic limits has been reached, some optimisations could be applied to further reduce the overall period T. Finally, it is pointed out that a real implementation of Algorithm 3 should initially check that the input parameters are such as to ensure that the kinematic limits to the joint trajectories do not exceed before starting the iteration.

As mentioned in the previous sections, with reference to 3.3a, the master path $\boldsymbol{r_m}(\tau)$ may not have all five segments. Actually, it is quite common for the geometric path to miss the rise or fall, defining a three-segment path (for an example see Fig. 4.3b). In these cases, Algorithm 3 classically terminates due to a kinematic limit in the smoothed corner section, while a margin remains in the linear section to optimise the motion. An asymmetrical motion profile can therefore provide significant advantages in terms of the objective function (4.6), which is why the pipeline in Fig. 4.1 inserts two further optimization steps. With reference to Fig. 4.3, the first asymmetric optimization consists of further enlarging the acceleration or deceleration section on the side where the kinematic limits allow it. Fig. 4.3a shows an example where the trajectory piece J_l, of period A_l, is enlarged until the X-axis reaches the limit AC_0, while J_r remains unchanged. This defines a two-level velocity profile, and an appropriate joint to bridge the gap ΔV must replace the null trajectory piece J_c to preserve the smoothness. With reference to the Algorithm 4, this joint can easily be defined from the same input jerk Trajectory1D J_{in} used by the Algorithm 3. Let A_l and A_r be the periods of acceleration and deceleration of S in a generic iteration. Using a slicing operation, select the two sections of the motion profile S corresponding to the domain $[0..A_l]$ and $[A_r - T..T]$, and call the resulting instances S_l and S_r. In order to define the velocity junction describing the central section, J_{in} is first renamed to J_c, then corrected in sign according to the sign of ΔV. The gap constraint ΔV is set in the jerk

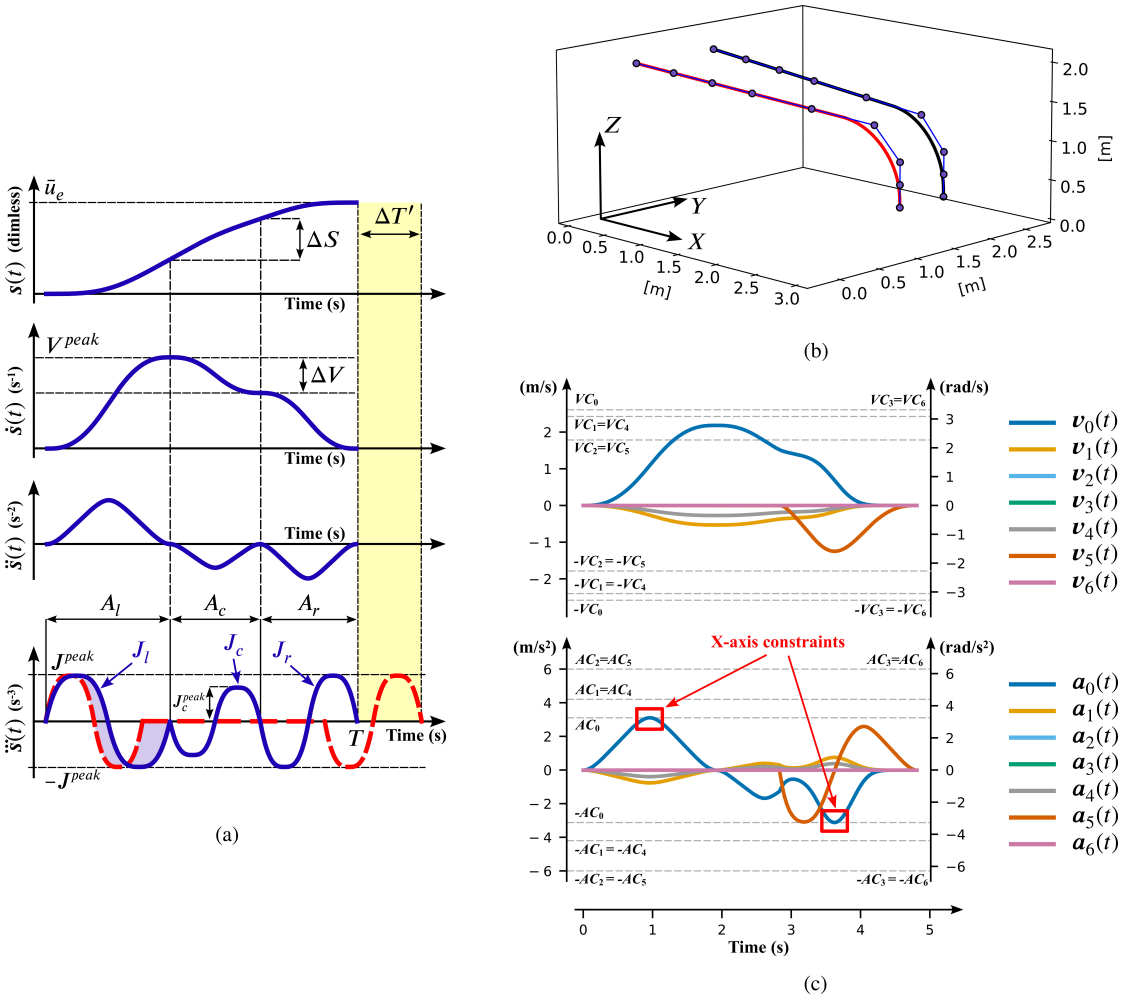


Figure 4.3: Schematic representation of the first optimisation of the motion profile. (a) Asymmetrical motion profile compared with the symmetrical solution. (b) Geometric path $\bar{\boldsymbol{r}}(\tau)$ of reference. (c) Joint trajectories obtained with (4.2) and (4.3).

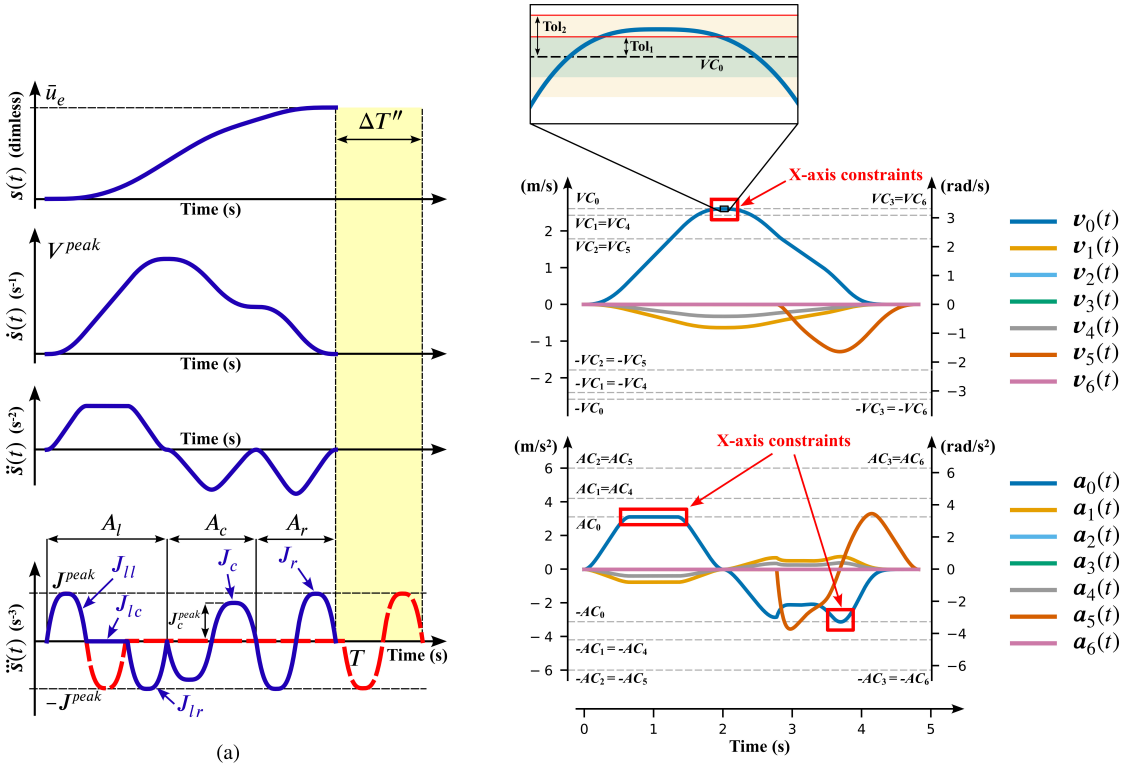


Figure 4.4: Schematic representation of the second optimisation of the motion profile. (a) Asymmetrical motion profile compared with the symmetrical solution. (b) Joint trajectories obtained with (4.2) and (4.3). In detail the strategy adopted for tolerances.

Algorithm 4 Asymmetric motion profile 1st optimization

Input:

1. Geometric path described from 7-D spline $\bar{\boldsymbol{r}}(\tau)$, defined in parametric domain $[0, \bar{u}_e]$.
2. Kinematic limits VC_i and AC_i, with $i = 0, \ldots, 6$, and J^{peak}.
3. Trajectory1D instance S from Algorithm 3.
4. Input jerk Trajectory1D object J_{in}.
5. Period A determined by Algorithm 3.
6. Number of samples N.
7. Tolerance Tol.
8. Initial incremental step ε and step ratio ϕ.

Output:

1. Trajectory1D object S.

Algorithm:

$A_l \leftarrow A_r \leftarrow A$
Split S in A and $S.t[-1] - A$, and set S_l, S_c, S_r.
Compute $D^{(1)}S_r$ and set V_r
Compute $D^{(3)}S_l$, $D^{(3)}S_r$ and set J_l, J_r.
Inizialize the empty Point1D objects P_1, P_2, P_3.
repeat
    repeat
        $A_l \leftarrow A_l + \varepsilon$
        Adapt J_l in the new domain $[0..A_l]$.
        Compute $D^{(-2)}J_l$, $D^{(-3)}J_l$ and set V_l, S_l.
        $\Delta S \leftarrow S_r.\text{cpts}[0] - S_l.\text{cpts}[-1]$
        $\Delta V \leftarrow V_r.\text{cpts}[0] - V_l.\text{cpts}[-1]$
        $J_c \leftarrow$ Copy of J_{in}
        Multiply J_c by the sign of ΔV.
        $P_1.val \leftarrow \Delta V; \quad P_1.ord \leftarrow 1; \quad P_1.t \leftarrow J_c.t[-1]$
        $J_c \leftarrow$setValueEn(J_c,P_1)
        Compute $D^{(-2)}J_c$ and save the result in V_c.
        $P_2.val \leftarrow V_l.\text{cpts}[-1]; \quad P_2.ord \leftarrow 1; \quad P_2.t \leftarrow V_c.t[0]$
        $V_c \leftarrow$setValueTr(V_c,P_2)
        $P_3.val \leftarrow \Delta S; \quad P_3.ord \leftarrow 0; \quad P_3.t \leftarrow V_c.t[-1]$
        $V_c \leftarrow$setValueEn(V_c,P_3)
        Compute $D^{(2)}V_c$ and set J_c.
        $A_c \leftarrow J_c.t[-1] - J_c.t[0]$
        Evaluate the absolute value of J_c in $A_c/4$ and set J_c^{peak}.
        if $J_c^{peak} > J^{peak}$ **then**
            break
        end if
        Concatenate V_l, V_c and set V_{lc}.
        Compute $D^{(-1)}V_{lc}$ and set S_{lc}.
    until checkLimitsOnAxes($\bar{\boldsymbol{r}}(\tau)$, S_{lc}, $\{VC_i\}$, $\{AC_i\}$, N)
    if $|J_c^{peak} - J^{peak}| <$ Tol **then**
        break
    end if
    $A_l \leftarrow A_l - \varepsilon; \quad \varepsilon \leftarrow \varepsilon \cdot \phi$
until checkLimitsOnAxesWithTol($\bar{\boldsymbol{r}}(\tau)$,$S_{lc}$,$\{VC_i\}$,$\{AC_i\}$,N,Tol)
Concatenate V_{lc}, V_r and set V.
Compute $D^{(-1)}V$ and set S
Repeat on the right side in the same way, trying to enlarge J_r.
return S

Algorithm 5 Asymmetric motion profile 2nd optimization

Input:

1. Geometric path described from 7-D spline $\bar{\boldsymbol{r}}(\tau)$, defined in parametric domain $[0, \bar{u}_e]$.
2. Kinematic limits VC_i and AC_i, with $i = 0, \ldots, 6$, and J^{peak}.
3. Trajectory1D instance S from Algorithm 3.
4. Input jerk Trajectory1D object J_{in}.
5. Periods A_l, A_c, A_r determined by Algorithm 4.
6. Number of samples N.
7. Tolerances Tol_1 and Tol_2.
8. Initial incremental step ε and step ratio ϕ.

Output:

1. Trajectory1D object S.

Algorithm:

$A_{ll} \leftarrow A_{lr} \leftarrow A_l/2$; $\quad A_{lc} \leftarrow 0$
Split S in A_l, $A_l + A_c$, and set S_l, S_c, S_r.
Split S_l in A_{ll} and set S_{ll}, S_{lr}.
Compute $D^{(1)}S_r$ and set V_r.
Compute $D^{(3)}S_l$, $D^{(3)}S_r$, $D^{(3)}S_{ll}$, $D^{(3)}S_{lr}$, and set J_l, J_r, J_{ll}, J_{lr}.
Inizialize the empty Point1D objects P_1, P_2, P_3.
Initialize a null Trajectory1D instance J_{lc}.
repeat
    repeat
        $A_{lc} \leftarrow A_{lc} + \varepsilon$
        Adapt J_{lc} in the domain $[0..A_{lc}]$.
        Concatenate J_{ll}, J_{lc}, J_{lr} and set J_l.
        Compute $D^{(-2)}J_l$, $D^{(-3)}J_l$ and set V_l, S_l.
        $\Delta S \leftarrow S_r.\text{cpts}[0] - S_l.\text{cpts}[-1]$
        $\Delta V \leftarrow V_r.\text{cpts}[0] - V_l.\text{cpts}[-1]$
        $J_c \leftarrow$ Copy of J_{in}
        Multiply J_c by the sign of ΔV.
        $P_1.val \leftarrow \Delta V$; $\quad P_1.ord \leftarrow 1$; $\quad P_1.t \leftarrow J_c.t[-1]$
        $J_c \leftarrow$setValueEn(J_c,P_1)
        Compute $D^{(-2)}J_c$ and set V_c.
        $P_2.val \leftarrow V_l.\text{cpts}[-1]$; $\quad P_2.ord \leftarrow 1$; $\quad P_2.t \leftarrow V_c.t[0]$
        $V_c \leftarrow$setValueTr(V_c,P_2)
        $P_3.val \leftarrow \Delta S$; $\quad P_3.ord \leftarrow 0$; $\quad P_3.t \leftarrow V_c.t[-1]$
        $V_c \leftarrow$setValueEn(V_c,P_3)
        Compute $D^{(2)}V_c$ and set J_c.
        $A_c \leftarrow J_c.t[-1] - J_c.t[0]$
        Evaluate the absolute value of J_c in $A_c/4$ and set J_c^{peak}
        if $J_c^{peak} > J^{peak}$ **then**
            break
        end if
        Concatenate V_l, V_c, V_r and set V.
        Compute $D^{(-1)}V$ and set S.
    until checkLimitsOnAxes($\bar{\boldsymbol{r}}(\tau)$, S, $\{VC_i + \text{Tol}_1\}$, $\{AC_i + \text{Tol}_1\}$,N)
    if $|J_c^{peak} - J^{peak}| < \text{Tol}$ **then**
        break
    end if
    $A_{lc} \leftarrow A_{lc} - \varepsilon$; $\quad \varepsilon \leftarrow \varepsilon \cdot \phi$
until checkLimitsOnAxesWithTol($\bar{\boldsymbol{r}}(\tau)$, S, $\{VC_i\}$, $\{AC_i\}$, N, Tol_2)
Repeat on the right side in the same way, trying to enlarge the trajectory piece J_r.
return S

order through setValueEn macro-instruction. Following, J_c is integrated twice obtaining V_c, which is translated vertically with setValueTr macro-instruction (please refer to [28]) to ensure continuity in the future concatenation with the acceleration and deceleration sections. Finally V_c is enlarged to set the constraint of ΔS, again using setValueEn. Referring to the specific case of Fig. 4.3a, the trajectory piece J_l is enlarged by ε at each step, and the procedure described above allows concatenate the result with J_r maintaining the smoothness of the motion profile. The iteration terminates when one of the limits AC_i or VC_i is reached within the tolerance Tol, or if the maximum amplitude of the centre section J_c^{peak} reaches the limit J^{peak} within the same tolerance. In the latter case, the pipeline in Fig. 4.1 ends by applying the appropriate scaling to the motion profile to set the desired velocity override. Instead, if Algorithm 4 terminates due to a kinematic limitation in joint space, a final optimisation attempts to add a constant acceleration section where the kinematic limits permit. In this application, acceleration limits are generally reached before velocity limits, and this second asymmetrical optimisation allows an increase in velocity while maintaining acceleration limited. However, note from (4.2) and (4.3), an increase in the acceleration period always leads to an increase in the velocity values of at least one axis in the interested section, but could also lead to an increase in the required torque if the curvature of the geometric path is non-zero. In this case, it is usually not possible to insert a constant-acceleration section without exceeding the limit reached by the Algorithm 4. Specifically, in the tasks in which the routine of Section 3.2.5 is involved, the 3-D path curvature of the two end-effectors is generally non-zero. As already mentioned, this routine applies a deviation to the geometric paths to avoid self-collision in the case there are loaded parts with dimensions that exceed those of the end-effector; it follows that the movements performed with unloaded arms are all planar (like those of Fig. 4.3b). The cases where a constant acceleration section can provide advantages on task time reduction are therefore relevant to the specific application.

With reference to Fig. 4.4, as in the previous case the resulting motion profile defines two velocity levels, and the required joint in the central section can be computed by the same routine of Algorithm 4. A null instance Trajectory1D J_{lc} is inserted with an iteratively increasing period between the two jerk impulses J_{ll} and J_{lr}. The iteration terminates when a new kinematic limit is reached, or if J_c^{peak} reaches the value of J^{peak}. In this case, however, it is not easy to select the motion profile sections to be checked during iterations, and a different strategy must be adopted to prevent the algorithm from terminating due to previously imposed limits. In the proposed solution, detailed in Fig. 4.4, a double tolerance is used to separate the limits already set at the start of the algorithm and those actually reached in the running routine. A first tolerance Tol_1 must be set to the minimum equal to Tol ($\text{Tol}_1 \geq$Tol). With reference to Fig. 4.4b, by setting the new constraint VC_0 to $VC_0 + \text{Tol}_1$, all the limits already reached at the beginning of the algorithm are less than or equal to the new limits. In general, the checkContraintsOnAxes function checks the axes limits $\{VC_i + \text{Tol}_1\}$ and $\{AC_i + \text{Tol}_1\}$, while the checkContraintsOnAxesWithTol function still

uses the limits $\{VC_i\}$ and $\{AC_i\}$, but checking with a tolerance $\text{Tol}_2 > \text{Tol}_1$. As a result of this operation, a new velocity limit is reached by the i^{th} axis when the maximum value of the function $\boldsymbol{v}_i$, calculated by (4.2), falls within the codomain band $\left]VC_i + \text{Tol}_1, VC_i + \text{Tol}_2\right]$. Similarly, a new acceleration limit is reached by the i^{th} axis when the maximum value of the function $\boldsymbol{a}_i$, calculated by (4.3), falls within the codomain band $\left]AC_i + \text{Tol}_1, AC_i + \text{Tol}_2\right]$. For the sake of completeness, the pseudo-code of the second optimisation routine is reported in Algorithm 5.

At the end of the pipeline in Fig. 4.1, a percentage velocity override is applied to the calculated motion profile $\boldsymbol{s}(t)$. In industrial applications, it is generally required that a percentage override can be set to scale the original trajectory velocity. For example, this feature is useful when testing planned motion, or to adjust the trajectory to a specific operating condition. The solution proposed in this section allows a simple off-line implementation of velocity override, which is reported in Algorithm 6.

Algorithm 6 Off-line velocity override setting

Input:

1. Motion profile $\boldsymbol{s}(t)$.
2. override%.

Output:

1. Motion profile $\tilde{\boldsymbol{s}}(t)$ of degree d, with knots $\tilde{\boldsymbol{t}} = (\tilde{t}_i)_{i=0}^{n+d+1}$ and control points $\tilde{\boldsymbol{k}} = (\tilde{k}_i)_{i=0}^{n}$.

Algorithm:

1: Starting from $\tilde{\boldsymbol{t}} = \boldsymbol{t}$ and $\tilde{\boldsymbol{k}} = \boldsymbol{k}$.
2: $\tilde{k}_i = k_i \cdot (\text{override\%}/100)$ **for** $i = 0, \ldots, n$
3: $s_f = |k_n / \tilde{k}_n|$ ▷ Scaling factor
4: $\dot{\tilde{\boldsymbol{s}}}(t) = D^{(1)}\tilde{\boldsymbol{s}}(t)$ with knots $\tilde{\boldsymbol{t}}^{(1)} = \left(\tilde{t}_i^{(1)}\right)_{i=0}^{n+d-1}$ ▷ Using (2.16)
5: $\tilde{t}_i^{(1)} = \tilde{t}_i^{(1)} \cdot s_f$ **for** $i = 0, \ldots, n+d-1$
6: $\tilde{\boldsymbol{s}}(t) = \int_0^{\tilde{t}_{n+d-1}^{(1)}} \dot{\tilde{\boldsymbol{s}}}(t)dt$ ▷ Using (2.17)
7: **return** $\tilde{\boldsymbol{s}}(t)$

Chapter 5

Implementation

5.1 Software architecture

This section summarises the workflow of the sorting process from a generic nest layout to the control design of the actuators.

Fig. 5.1 illustrates the motion planning workflow proposed in this thesis in a simplified diagram. The process begins with CAD files derived from the nest layout, obtained through CAM laser machine software. In more detail, the geometry of the nest is described with .dxf files as a set of simple curves such as lines and arcs of circumference, derived from a G-code. From these geometries, the bounding boxes of the parts are calculated as rectangles with sides consistent with the edges of the sheet metal. In accordance with the procedures described in the previous sections, the part boxing algorithm could be improved to define any convex polygon circumscribing the part. Using the machine layout information, a desktop application facilitates the generation of a text file containing basic instructions, called macro-instructions. These macro-instructions represent the tasks needed to execute the entire sorting operation. Each macro-instruction contains all the essential information for

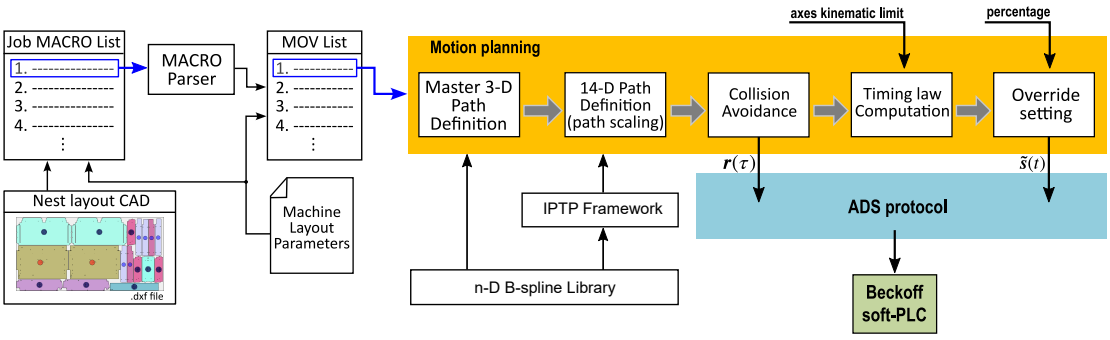


Figure 5.1: Motion planning workflow.

the machine to carry out the task, including part loading and unloading positions, bounding boxes of the parts, positions of free heads, and tool selection for gripping, among other details. Subsequently each macro-instruction is parsed to break it down into a sequence of rest-to-rest movements, which may include just one or none. Each movement is characterized by its starting and target positions, with the end-effectors positioned at zero velocity.

Each rest-to-rest movement is processed with the off-line motion planning procedure detailed in the previous sections, obtaining a geometric path defined by a 14-D spline and a motion profile defined by a 1-D time-parametrized spline (in whose nodes the information of the override applied to the movement is already contained). Clearly, the simple flowchart in Fig. 5.1 is not exhaustive of the entire motion logic of the machine, which is performed by a complex state-machine. The latter ensures the synchronisation of movements with a whole series of operations, such as tool changes or pauses required to check the success of the loading by laser scanner.

Each movement, represented by a geometric path $\boldsymbol{r}(\tau)$ and a motion profile $\tilde{s}(\tau)$, is sent as knots and control points vectors to the soft-PLC via ADS protocol [43]. From these data, a real-time logic implementing a B-spline interpolator allows the generation of position set-points and torque offsets for each motion-task time (in the machine used in this work lasting 2 ms). The next section describes in detail the real-time control procedure used in terms of axis control.

5.2 Control design

This section details the real-time control strategy adopted. In this regard, Fig. reports the control diagram proposed in this work. The control-chain is based on the cyclic synchronous position (CSP) mode described in CiA® 402 (IEC 61800-7), implementing the classic position, velocity and torque controllers cascaded for each axis. To feed the torque controller with a more accurate set-point, the implemented application incorporates a least-squares linear friction (BLLS) model, adapting the calculation of torque set-points to actual system changes during operation. From the operation point of view, the position controller must be maintained enabled to adapt the actuation and safely control the position in case of small trajectory corrections or failures that lead to an incorrect estimation of the system behaviour by the model. In the following sections, the implemented software applications of the real-time set-point generator and the adaptive BLLS model are detailed, respectively.

5.2.1 Real-time set-points generator

The real-time set-point generator must compute and send to motor drives the position, velocity and torque set-points for all axes and for each PLC motion task. It must also initiate the safety-stop procedure if an emergency occurs, whether this is generated externally by the operator panel or for exceeding the permissible tracking error. It follows that the

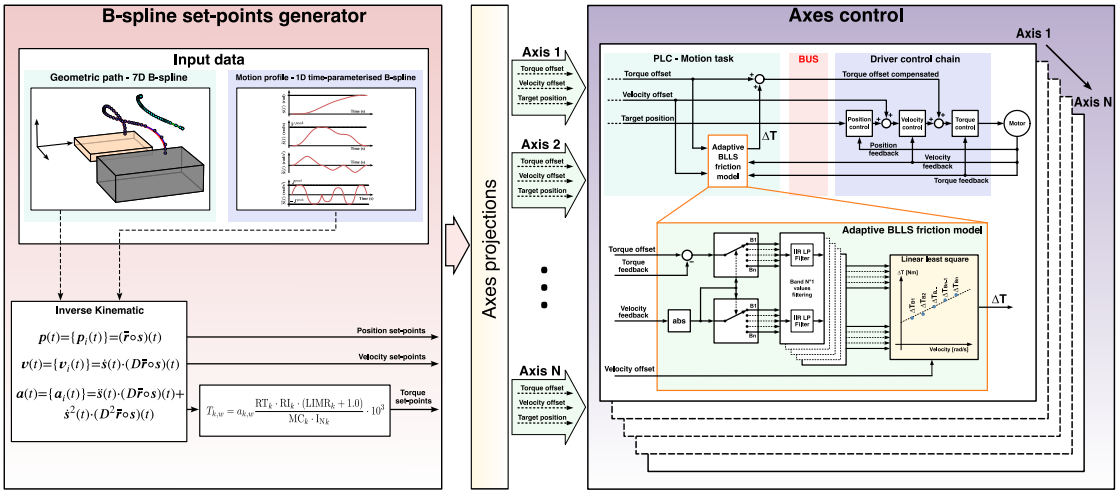


Figure 5.2: Real-time control diagram.

set-points computation, whether in standard or emergency operation, must have sufficient computational performance to ensure that the motion task time is not exceeded.

Algorithm 7 reports the pseudo-code of the function block that computes the set-points. In terms of the state machine that manages the PLC workflow, the calculation of set-points for a generic movement from a start point to a target point begins once the bEnable flag is raised. With reference to Fig. 5.1, geometric path and timing law are sent in binary format to the PLC, which stores the information in two instances of the custom data structure shown below:

```
type dutBSpline:
struct
p: int                                  // curve degree
n: int                                  // max index of the control points
array
r: int                                  // dimension of the curve
cpts: array[0..r,0..n] of double
knots: array[0..(n+d+1)] of double
end_struct
end_type
```

In standard machine operation, position, velocity, and acceleration set-points are computed using projection equations (4.1), (4.2) and (4.3). The evaluation of the two splines and their first and second derivatives is performed according to the algorithms proposed by [14] from the function with signature:

CurveDerivsEval(bspl: dutBSpline := , x: double := , d: int :=) → C: array[0..bspl.r,0..d] of double

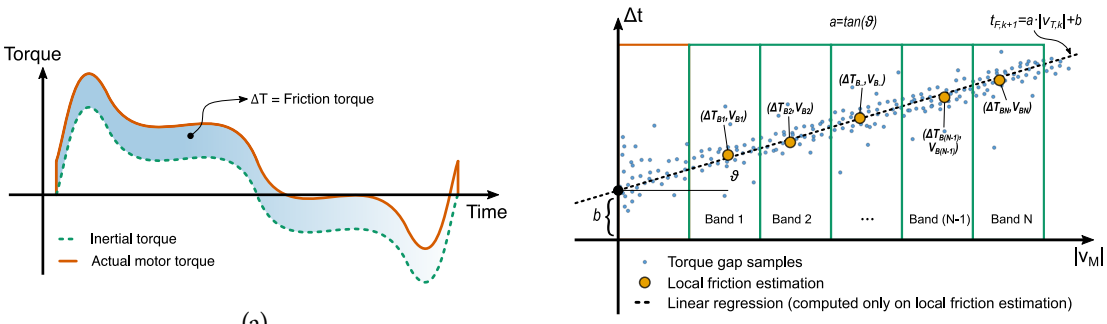


Figure 5.3: Friction compensation with BLLS adaptive model. (a) Friction torque. (b) Band linear regression.

with C[j,i] the i^{th} derivative evaluated in x along the j^{th} dimension of the spline. Finally, the torque set-point $T_{k,w}$, at the w^{th} step for the k^{th} axis, is then computed from the acceleration set-point $a_{k,w}$ via the equation:

$$T_{k,w} = a_{k,w} \frac{\text{RT}_k \cdot \text{RI}_k \cdot (\text{LIMR}_k + 1.0)}{\text{MC}_k \cdot \text{I}_{\text{N}k}} \cdot 10^3 \tag{5.1}$$

For the meaning of the acronyms in (5.1), please refer to the summary table in the preface.

5.2.2 Adaptive band-wise linear least squares friction model

In addition to the more easily estimated inertial component, the torque set-point calculation must also include non conservative dissipative effects. The latter are much more complex to estimate through a model-based strategy because characterized by many different causes in the real system, such as rolling friction (rails and bearings), viscous phenomena and the deformation of cables and pneumatic hoses. These phenomena, difficult to describe correctly and reliably in a well-known situation, are even more unaffordable when the machine operates in different condition over years. In addition, phenomena such as wear of mechanical components can significantly change the machine's behavior.

The existing controller also compensates all missing dissipative effects, but with some drawback:

- The cascade velocity controller compensates steady-state friction through the integral component, introducing a delay when the axis reverses its direction.
- Position control without feedforward significantly modifies the desired torque profile by introducing harmonics that can potentially excite the system causing vibrations.

In order to overcome these limitations, an adaptive friction estimator was included in the control diagram, as illustrated in Fig. 5.2. With reference to Fig. 5.3a, the BLLS estima-

Algorithm 7 On-line set-point generator

Input Variables:

1. mbspl: dutBSpline ▷ Geometric path in a custom struct
2. ubspl: dutBSpline ▷ Timing law in a custom struct
3. bAlarm: bool ▷ If true, alarms the machine
4. bEnable: bool ▷ If true, enable the function block

Output Variables:

1. fTargetPos: array[0..6] of double ▷ Position set-points
2. fTargetVel: array[0..6] of double ▷ Velocity set-points
3. fTargetTrq: array[0..6] of double ▷ Torque set-points

Local Variables:

1. fTimeRel: double ▷ Relative time in the task
2. fStartTargetPos: array[0..6] of double
3. C: array[0,0..2] of double
4. CK: array[0..6,0..2] of double

Global Variables:

1. fTol: double ▷ Absolute tolerance for double comparison

Algorithm:

if bEnable **and not** bAlarm **then**
    Set 'bAlarm' to **True** if the lag error exceeds the maximum allowed.
end if
if bAlarm **then**
    bEnable ← **False**
end if
if A rising edge of bEnable is detected **then**
    Reset local variables.
end if
if not bEnable **then**
    if the tangential velocities and angular velocities targets is $\leq$ 'fTol' **then**
        Set 'fTargetPos' maintaining the position on the axes.
        Set 'fTargetVel' and 'fTargetTrq' to null vectors.
    else
        Performing the safety-stop with controlled deceleration.
    end if
    return
end if
Update the relative time 'fTimeRel'
if ubspl.knots[-1] - fTimeRel $<$ fTol **then**
    for $i \leftarrow 0$ to 6 **do**
        fTargetPos[i] ← fStartTargetPos[i] + mbspl.cpts[i,-1]
        fTargetVel[i] ← fTargetTrq[i] ← 0.0
    end for
return
end if
C ← **CurveDerivsEval(**ubspl, fTimeRel**)** ▷ [14]
CK ← **CurveDerivsEval(**mbspl, C[0,0]**)** ▷ [14]
for $i \leftarrow 0$ to 6 **do**
    CK[i, 2] ← C[0,2]·CK[i, 1] + C[0,1]^2·CK[i, 2] ▷ equation (4.3)
    CK[i, 1] ← C[0,1]·CK[i, 1] ▷ equation (4.2)
    fTargetPos[i] ← fStartTargetPos[i] + CK[i, 0]
    fTargetVel[i] ← CK[i,1]
    Set fTargetTrq[i] with (5.1) using the acceleration value CK[i,2].
end for
return

tor computes the torque compensation $t_{F,k+1}$ for the $(k+1)^{\text{th}}$ motion-task step, using the parameters:

$$\Delta t_k = |t_{J,k} - t_{M,k}| \tag{5.2}$$
$$v_{m,k} = |v_{M,k}| \tag{5.3}$$

The calculation of a linear regression on a large number of samples is not a viable option due to the high memory and computing capacity required. With reference to Fig. 5.3b, the solution adopted is summarised in the following steps:

1. Δt_k is computed with (5.2) and is classified in the i^{th} band according to the corresponding velocity $v_{m,k}$;
2. Low-pass filtering of Δt_k and $v_{m,k}$. This operation returns the new $\Delta T_{B,i,k}$ and $V_{B,i,k}$ values, corresponding to the i^{th} velocity band at the k^{th} motion-task step.
3. Update of the friction model by linear regression of the $(\Delta T_{B,j,k}, V_{B,j,k})$ values, with $j = 1, \ldots, N$ and N numbers of bands.

It is brought to the reader's attention how the first velocity band is excluded from the model estimation. At these velocities, the static friction at start and the inertial energy at stop heavily influence the actual torque required by the motor, making it difficult to estimate. For this reason, only dynamic friction can be estimated by this approach. The low-pass IIR filter used at each step is defined by the following equations:

$$\begin{aligned} \Delta T_{B,i,k} &= \Delta T_{B,i,k-1}(1 - K_w) + \Delta t_k K_w \\ V_{B,i,k} &= V_{B,i,k-1}(1 - K_w) + v_{m,k} K_w \end{aligned} \tag{5.4}$$

with $0 \leq K_w \leq 1$. In (5.4), K_w set the cut frequency of the filter and can be interpreted as a kind of forgetting factor that can be set to make the filter more or less sensible to variations.

The linear friction model can be updated by recalculating the angular coefficient a and intercept b with the known regression formulations:

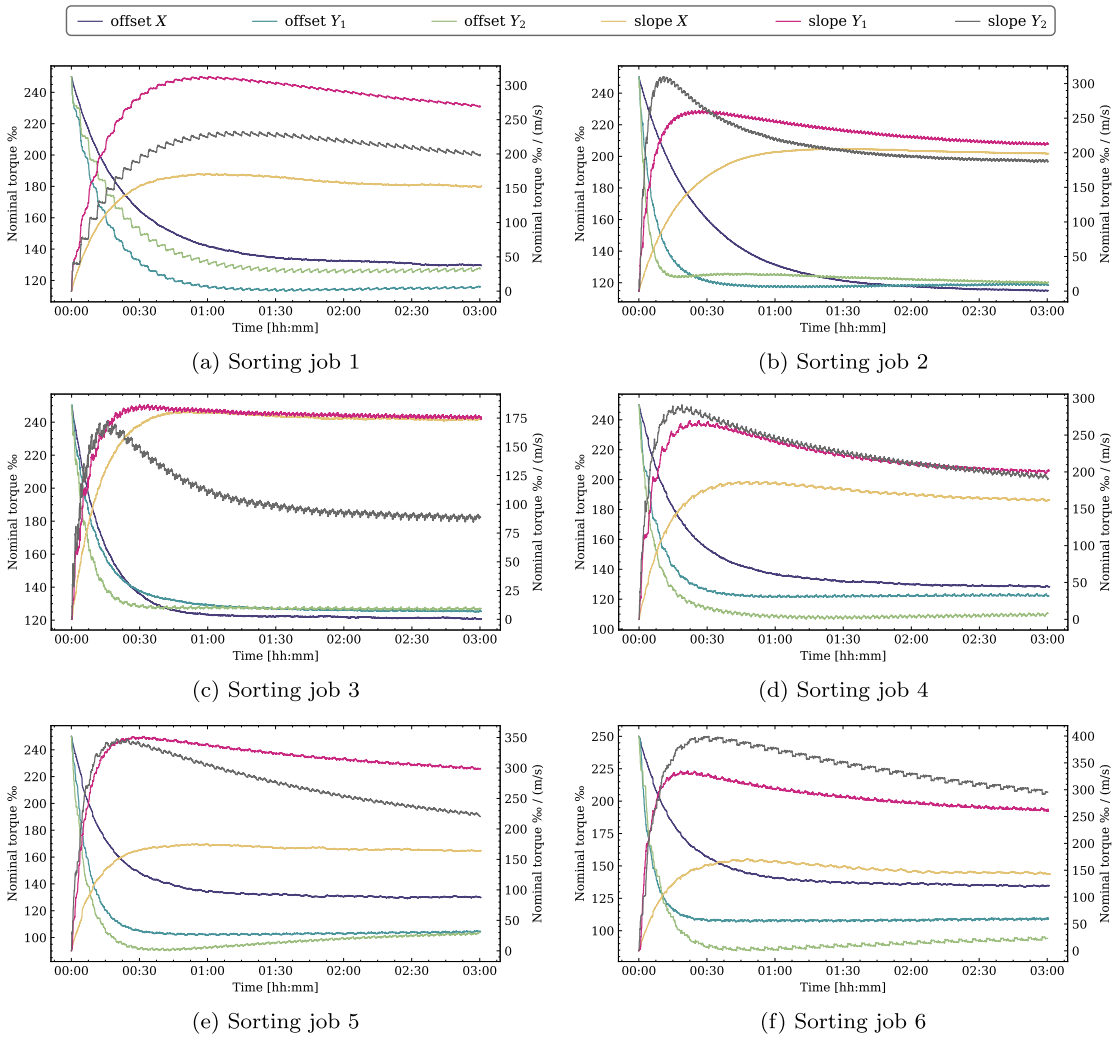
$$\begin{aligned} a &= \frac{\left(\sum \Delta T_{B,i}\right)\left(\sum V_{B,i}^2\right) - \left(\sum V_{B,i}\right)\left(\sum V_{B,i}\Delta T_{B,i}\right)}{N\left(\sum V_{B,i}^2\right) - \left(\sum V_{B,i}\right)^2} \\ b &= \frac{N\left(\sum V_{B,i}\right)\left(\sum V_{B,i}\Delta T_{B,i}\right) - \left(\sum V_{B,i}\right)\left(\sum \Delta T_{B,i}\right)}{N\left(\sum V_{B,i}^2\right) - \left(\sum V_{B,i}\right)^2} \end{aligned} \tag{5.5}$$

where for all sums $i = 1, ..., N$. These formulations highlight the matter of reducing as much as possible the number of input samples N to obtain an efficient computation. The torque set-point at the $(k+1)^{\text{th}}$ step is then calculated with:

$$t_{F,k+1} = -\text{sign}(v_{T,k})(a v_{T,k} + b)$$

To investigate the stability of the proposed approach, the parameters a and b, repre-

senting the angular coefficient and offset of the real-time calculated friction model, were tracked over a test duration of six hours for six different job macro lists. The result of these test is reported in Fig. 5.4. Starting from initial values of $0‰/(\mathrm{m/s})$ for the slope and $250‰$ for the offset, the procedure is stable in all examined cases, reaching a plateau of convergence within the time limit. The friction models obtained for the six tests are shown in Fig. 5.5. Observably, the models exhibit notable variations from one test to another, despite using the same test machine. This variance can be attributed to the machine primarily operating in different zones based on the specific sort macro list. Hence, this aspect substantiates the necessity for an adaptive friction model solution.



(a) Sorting job 1

(b) Sorting job 2

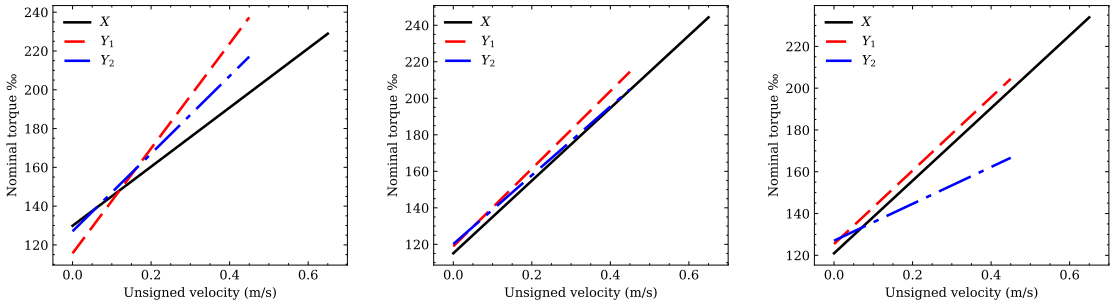
(c) Sorting job 3

(d) Sorting job 4

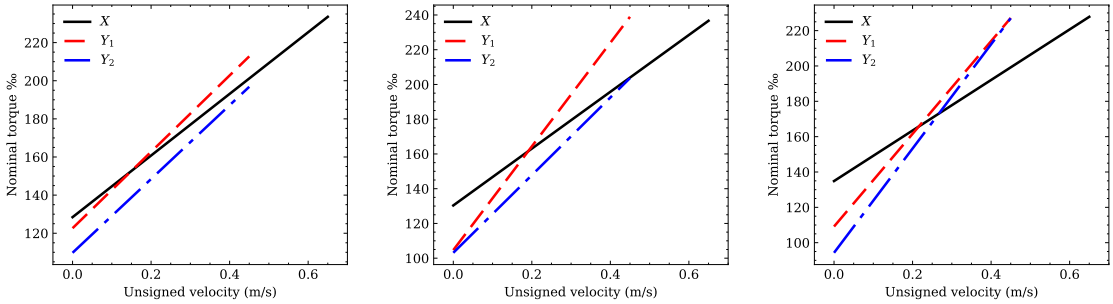
(e) Sorting job 5

(f) Sorting job 6

Figure 5.4: Convergence analysis of the BLLS friction model for 6 different sorting jobs.



(a) Sorting job 1 (b) Sorting job 2 (c) Sorting job 3



(d) Sorting job 4 (e) Sorting job 5 (f) Sorting job 6

Figure 5.5: Regression lines for the 6 sorting jobs of the convergence analyses of Fig. 5.4.

Chapter 6

Result and discussion

This section shows some results of the proposed motion planning structure in the most commonly used machine configuration, which is the 7-d.o.f. single gantry kinematics (two end-effectors). In particular, the following case studies are investigated:

- Section 6.1: This section examines a case study in which no collision avoidance method is required in addition to the determination of start and target points. This scenario occurs in practical situations when the machine is without loaded parts or when the parts are in the shadow of the arm. Limited to this case study, the parameterisation adopted for the geometric path is in accordance with the following:

$$s_{max} = \sum_{i=0}^{m-1} \sqrt{\Delta x_{i,\phi}^2 + \Delta y_{i,\phi}^2 + \Delta z_{i,\phi}^2}$$

- Section 6.2: This case study analyses a generic case of a nest sorting operation where it is necessary to apply the anti-collision techniques detailed in Section 3.2.5.

For both case studies, the transmission ratios and axes limits are reported in Tab. 6.1.

Table 6.1: Trasmission rations and axes limits.

	Axes			
	$X\ (i=0)$	$Y\ (i=1,4)$	$Z\ (i=2,5)$	$W\ (i=3,6)$
$K_{r,i}$	112.20 (rad/m)	124.94 (rad/m)	161.70 (rad/m)	104.35 (rad/rad)
AC_i	3.25 (m/s^2)	4.2 (m/s^2)	6 (m/s^2)	6.28 (rad/s^2)
VC_i	2.75 (m/s)	2.6 (m/s)	2.4 (m/s)	3.50 (rad/s)

6.1 Sorting task with freed heads

This section presents a use case of the provided motion planning solution for a traditional sheet metal sorting task, where there is no necessity for employing extra anti-collision

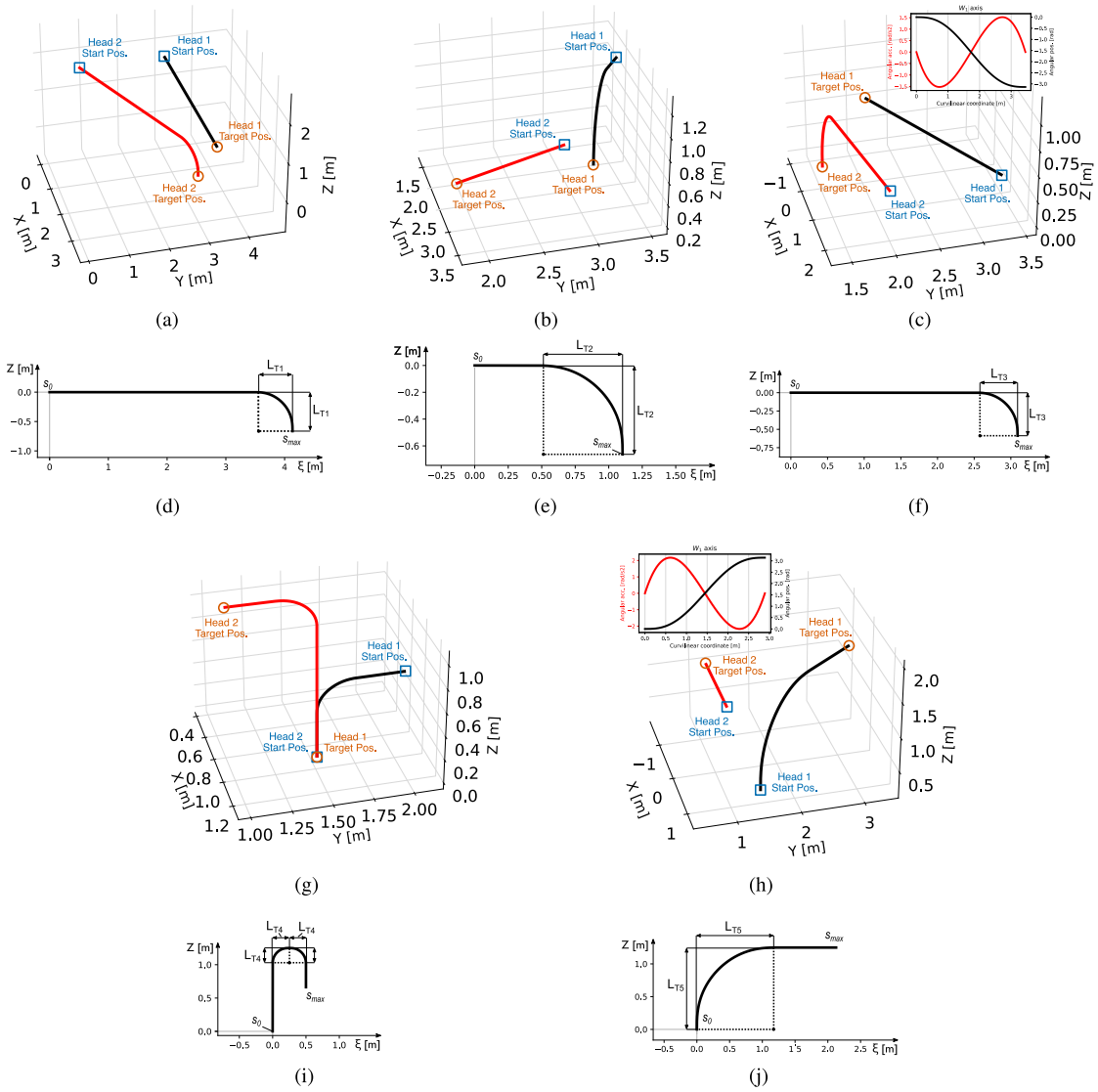
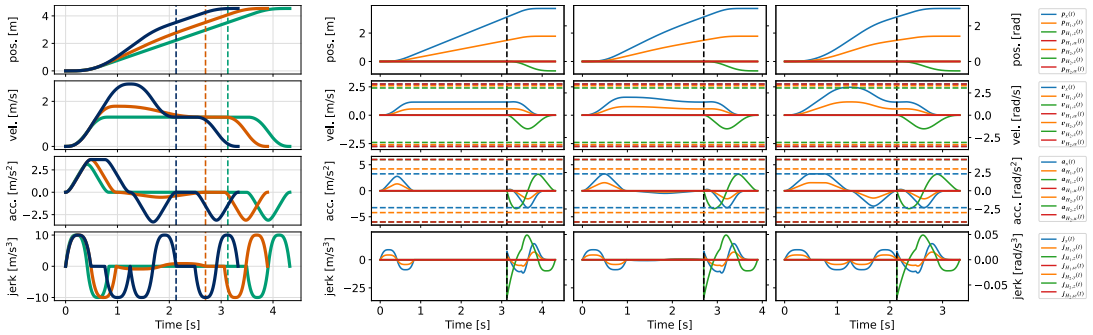
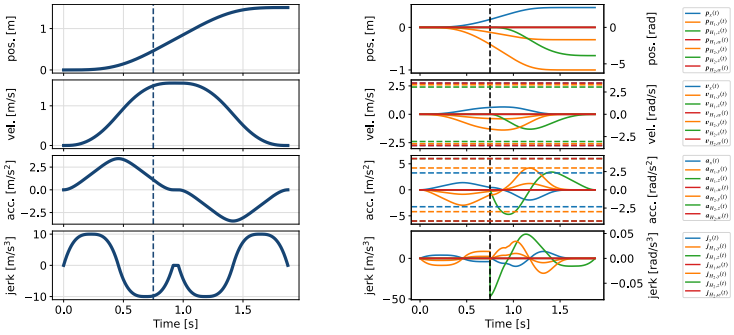


Figure 6.1: Graphical representation of the five test movements. (a),(b),(c),(g),(h) 3-D plot of the five movements (black line for head 1, red line for head 2). (d),(e),(f),(i),(j) 3-D master paths, respectively.



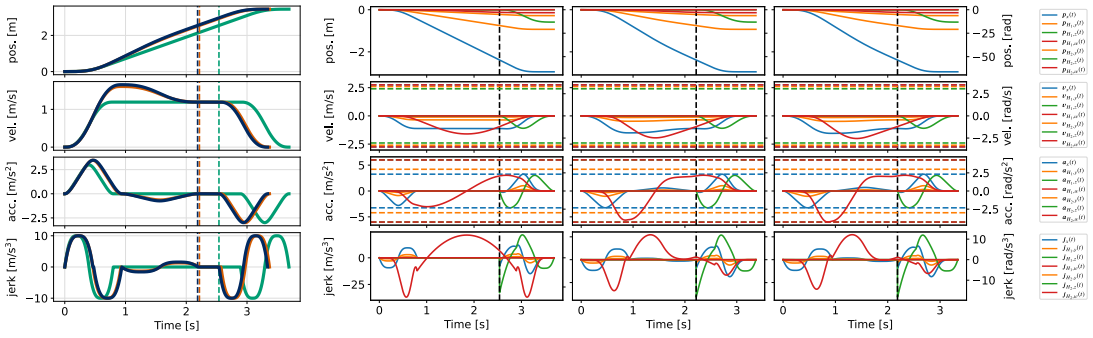
(a) (b) (c) (d)

Figure 6.2: Timing law computation of the 1st movement for $J^{peak} = 10\,\mathrm{m/s^3}$. (a) Timing law at the three optimisation levels. (b),(c),(d) Axis projections calculated via (4.1) to (4.3) for the three optimisation levels, respectively.



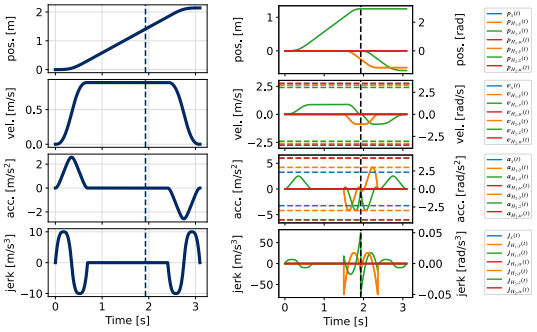
(a) (b)

Figure 6.3: Timing law computation of the 2nd movement for $J^{peak} = 10\,\mathrm{m/s^3}$. (a) Computed timing law. (b) Axis projections calculated via (4.1) to (4.3).



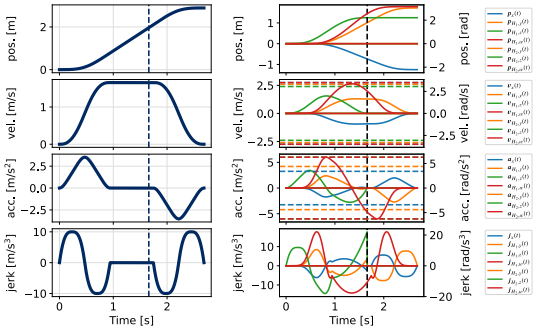
(a) (b) (c) (d)

Figure 6.4: Timing law computation of the 3rd movement for $J^{peak} = 10\,\mathrm{m/s^3}$. (a) Timing law at the three optimisation levels. (b),(c),(d) Axis projections calculated via (4.1) to (4.3) for the three optimisation levels, respectively.



(a) (b)

(a) Timing law computation of the 4th movement for $J^{peak} = 10\,\mathrm{m/s^3}$. The vertical lines separate one transition segment from the next. (a) Computed timing law. (b) Axis projections calculated via (4.1) to (4.3).



(a) (b)

(b) Timing law computation of the 5th movement for $J^{peak} = 10\,\mathrm{m/s^3}$. The vertical lines indicate the end of the transition segment. (a) Computed timing law. (b) Axis projections calculated via (4.1) to (4.3).

Figure 6.5

strategies beyond those already applied to determine loading and unloading positions. The test consists of a concatenation of five movements, corresponding to the loading of two parts from the nesting (one per head) and their respective unloading onto pallets in a single stack. Fig. 6.1 shows the master curves and heads geometric paths for the five movements. Start and target positions are reported in Tab. 6.2. Other parameters defining motion are reported in Tab. 6.3, such as the transition lengths L_{Ti} and the travel height $z_{tr,i}$ of the i^{th} movement.

In the first movement, shown in Fig. 6.1a, the head 2 move from the starting position to the gripping position of the first part on the automatic pallet changer. In the second movement (see Fig. 6.1b), head 2 moves into a break free position to allow head 1 to load the second part. The third movement is illustrated in Fig. 6.1c, and leads head 2 to unload the part onto the automatic drawer. The fourth movement (see Fig. 6.1g) sets the final height at head 2, allowing head 1 to unload the part above the previously unloaded part. Finally, the fifth movement (see Fig. 6.1h) leads the heads back to the initial position, allowing the test procedure to be repeated in order to draw some considerations on the solution adopted for the control.

Motion profiles and respective axis trajectories are shown in Fig. 6.2 to Fig. 6.5b. In Fig. 6.2a is reported the motion profile computed on the geometric path of Fig. 6.1a for the three optimisation levels returned by the workflow of Fig. 4.1. In this case the algorithm terminates in Algorithm 5 because the condition $J_c^{amp} \geq J^{peak} - \mathrm{Tol}$ becomes true. Although each resulting motion profile is valid and respects the kinematic constraints of the axes, in the result of Algorithm 5 the objective function (2.28) has a value about 28% lower than that returned by Algorithm 3. This rate is mainly due to a relatively small transition length

Table 6.2: Absolute coordinate.

Pos.	Head 1				Head 2			
	X mm	Y_1 mm	Z_1 mm	W_1 rad	X mm	Y_2 mm	Z_2 mm	W_2 rad
1	250.0	3050.0	1850.0	0.0	250.0	830.0	1850.0	0.0
2	3982.5	3050.0	1850.0	0.0	3982.5	2608.5	1187.0	0.0
3	4448.5	2760.5	1187.0	0.0	4448.5	1608.5	1187.0	0.0
4	1500.0	1830.0	1187.0	-π	1500.0	1330.0	600.0	0.0
5	1500.0	1330.0	600.0	-π	1500.0	830.0	1850.0	0.0
6	250.0	3050.0	1850.0	0.0	250.0	830.0	1850.0	0.0

L_{T1}, which results in a high curvature in the fall section. Referring to Fig. 6.2b, this curvature leads to a stop of Algorithm 3, which sets the maximum acceleration limit AC_X. The following Algorithm 4 allows the maximum acceleration value AC_X to be set even in the first section of the geometric path (see Fig. 6.2c), before the start of the transition segment, while Algorithm 5 maintains this acceleration to increase the maximum velocity as much as possible (see Fig. 6.2d). Both Algorithm 4 and Algorithm 5 therefore perform localised motion profile optimisations while preserving the kinematic limits previously imposed by Algorithm 4.

Fig. 6.3a shows the resulting motion profile for the geometric path of Fig. 6.1b. Different from the previous case, L_{T2} is relatively large, leading to a stop of the motion profile computation already at Algorithm 3. What stops the algorithm is the maximum acceleration condition AC_Y set on the head 2. Note the acceleration phase ends within the transition segment, which precludes further optimisation steps.

Fig. 6.4a reports the resulting motion profile for the geometric path of Fig. 6.1c. This case is similar to the 1$^{\text{st}}$ movement, except for the limitation imposed by the W-axis of head 1 which terminates Algorithm 5 when it sets the acceleration limit AC_W.

The motion profile shown in Fig. 6.5a, corresponding to the master path reported in Fig. 3.6j, appears to be the most limited in performance for $J^{peak} = 10\,\text{rad/s}^3$. The master path has the lowest transition length value L_{T4} among the tested movements, limited to half of the ξ displacement. The motion profile workflow terminates due to the limit AC_Y on both the head 1 and head 2.

Finally, Fig. 6.5b reports the resulting motion profile for the geometric path of Fig. 6.1h. In this case, the motion profile computation algorithm stops in Algorithm 3 by setting the limit AC_W to the head 1.

In order to provide a performance evaluation of the proposed approach, the task described by the start and target positions in Tab. 6.2 are also solved with double-S motion profiles [44] following a classical rectangular pick-and-place path. The methods are compared using the same velocity and acceleration limits to constrain the motion. The following upper limits of the jerk are adopted in double-S trajectories: $JC_X = 5\,\text{m/s}^3$,

Table 6.3: Characteristic parameters for test movements.

	movements				
	1st (pos 1 to 2)	2nd (pos 2 to 3)	3rd (pos 3 to 4)	4th (pos 4 to 5)	5th (pos 5 to 6)
L_T (mm)	663.0	663.0	587.0	250.0	1250.0
$z_{tr,1}$ (mm)	1850.0	1850.0	1187.0	1187.0	1850.0
$z_{tr,2}$ (mm)	1850.0	1187.0	1187.0	1850.0	1850.0
s_{max} (mm)	4539.73	1508.42	3450.57	2142.56	2890.14
α (deg)	25.48	65.01	162.48	90.00	126.01
Time (s)	3.326	1.875	3.339	3.095	2.685

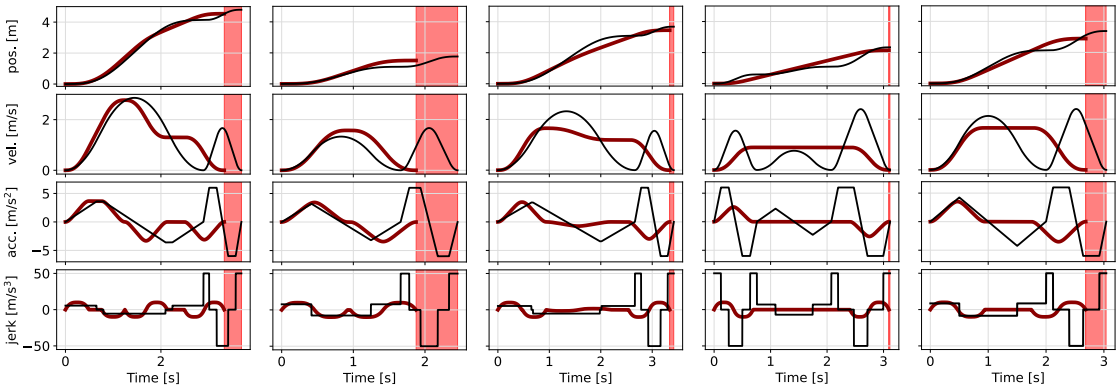


Figure 6.6: Time gain with respect to the classic rectangular pick-and-place path for $J^{peak} = 10\,\mathrm{m/s^3}$. (a) to (e) indicate the movements 1st to 5th, respectively.

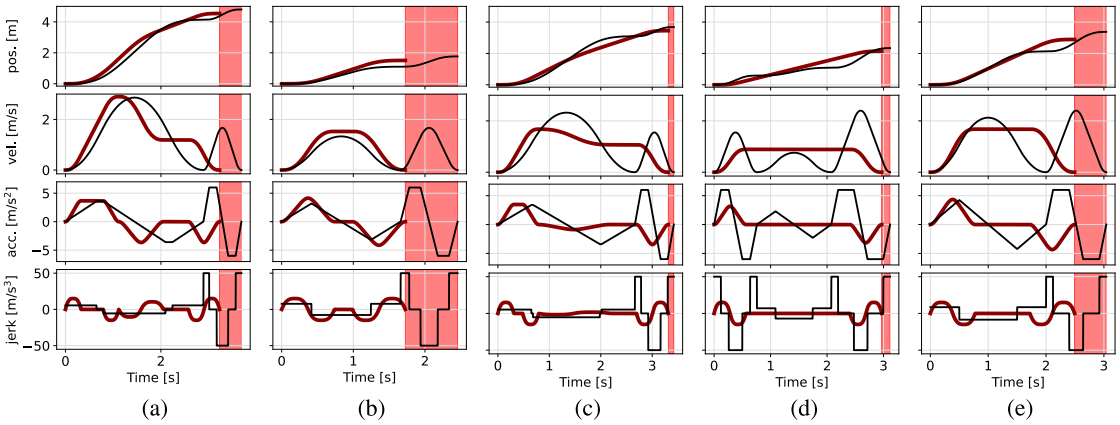
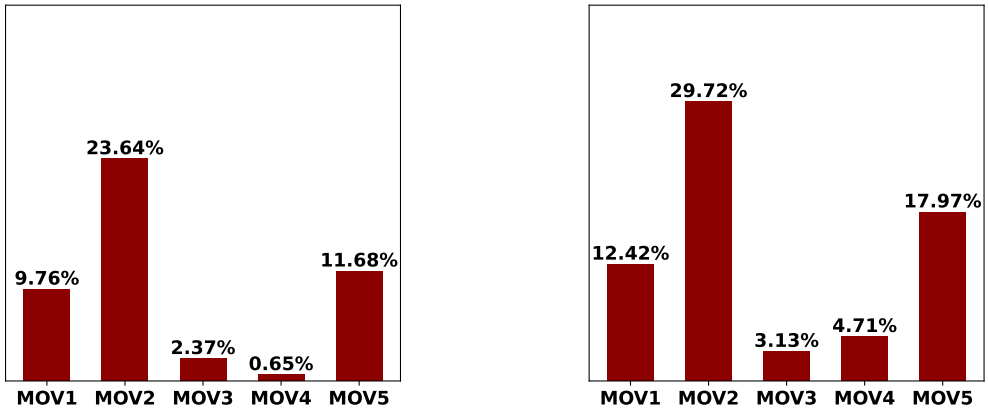


Figure 6.7: Time gain with respect to the classic rectangular pick-and-place path for $J^{peak} = 15\,\mathrm{m/s^3}$. (a) to (e) indicate the movements 1st to 5th, respectively.



(a) (b)

Figure 6.8: Time gain with respect to the classic rectangular pick-and-place path for different values of J^{peak}. The timing law followed by the axes is calculated with a double-S trajectory with cubic blend [44]. The maximum jerk limits on the axes used are: $JC_X = 5\,\mathrm{m/s^3}$, $JC_Y = 7\,\mathrm{m/s^3}$, $JC_Z = 50\,\mathrm{m/s^3}$, $JC_W = 16\,\mathrm{rad/s^3}$. (a) $J^{peak} = 10\,\mathrm{m/s^3}$. (b) $J^{peak} = 15\,\mathrm{m/s^3}$.

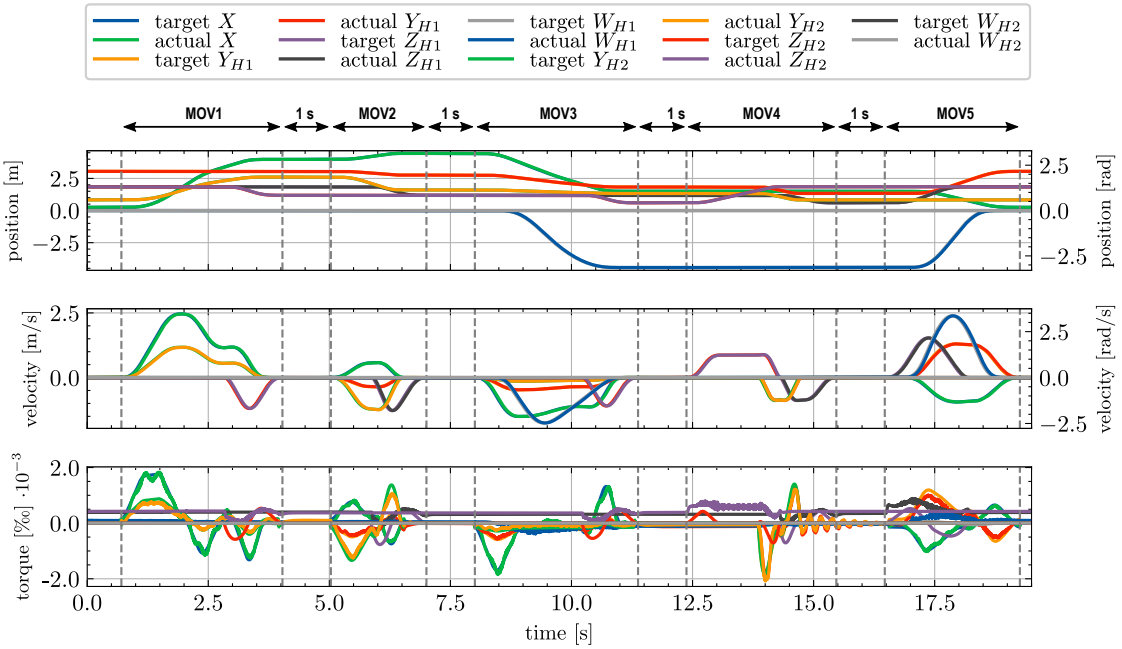


Figure 6.9: Position, velocity and torque data recorded via TwinCAT 3 Scope. The movements are spaced by pauses of 1 s.

$JC_Y = 7\,\mathrm{m/s^3}$, $JC_Z = 50\,\mathrm{m/s^3}$, $JC_W = 16\,\mathrm{rad/s^3}$. These maximum jerk values are set to limit residual vibration once the target position has been reached, which could otherwise impair the machine's operational functionality. In the proposed comparison, the double-S motion profiles are concatenated to each other considering zero rest. In practice, despite the jerk limitations, a stop time between one movement and the next is always necessary to allow the system to dampen oscillatory phenomena. Fig. 6.6 and Fig. 6.7 show the comparison between the two approaches in terms of the curvilinear abscissa on the relative geometric path, with a variation of J^{peak} from $10\,\mathrm{rad/s^3}$ to $15\,\mathrm{rad/s^3}$, for the five movements of the test. The time gained by our approach compared to the comparison approach is reported in percentages in Fig. 6.8. In general, the proposed method offers a shorter trajectory execution time, greater smoothing of the timing law, and reduced dynamic parameters at the end-effector module. This last characteristic holds particular significance in this specific application, as it helps restrict actions at the part-tool interface, which, if excessive, could result in slippage or the loss of the loaded part. As expected, the greatest performance advantages are obtained for high values of the transition length L_T, as in the case of the 2$^{\text{nd}}$ and 5$^{\text{th}}$ movements. The reason is that a high value of L_T ensures a limited curvature of the geometric path, which results in lower accelerations and velocities in the axes involved.

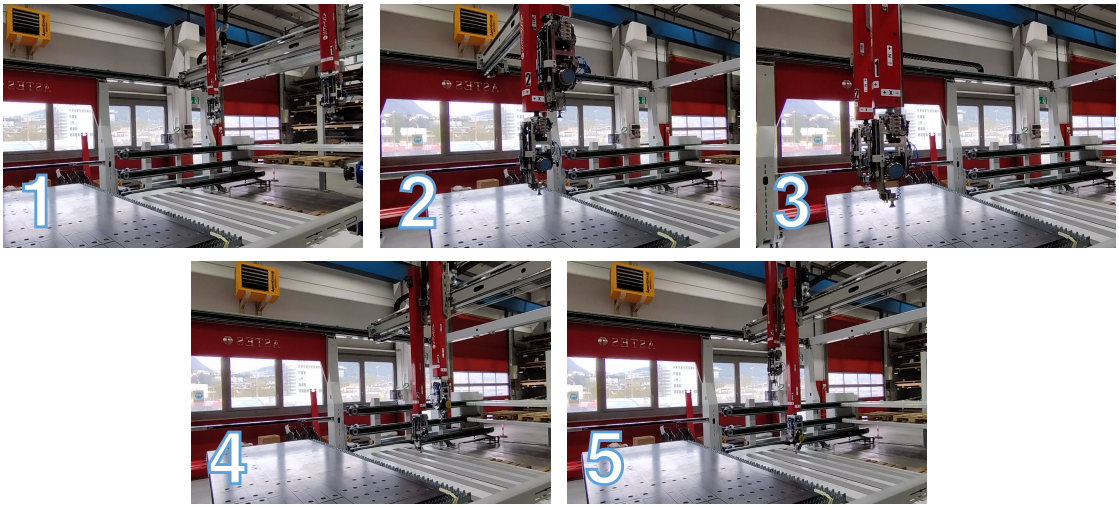


Figure 6.10: The process of sorting of the nest under study. Sub-pictures 1 to 5 represent the starting position of the point-to-point movements, according to Tab. 6.2.

Fig. 6.9 reports experimental position, velocity and torque data collected via TwinCAT® 3 Scope during the execution of the test. A stop time of 1 s is inserted between each movement in order to monitor any fluctuations around the target. The position and torque targets are usually well followed, except for the 4$^{\text{th}}$ movement where a resonance frequency along the Y-direction is emphasised. In the specific application, the synchronous belt drive

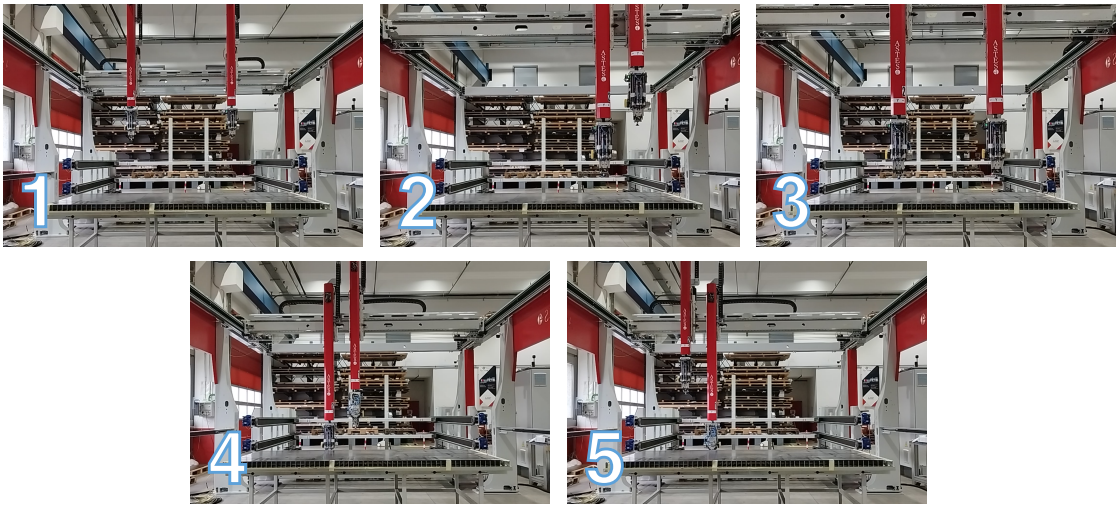


Figure 6.11: The process of sorting of the nest under study. Sub-pictures 1 to 5 represent the starting position of the point-to-point movements, according to Tab. 6.2.

adopted for axis movement introduces potential low-frequency resonances that must be considered during motion planning to ensure smoothness. The study and suppression of these phenomena, common for this type of machines, are not covered in this thesis, but will be the subject of a future work in which the topic will be addressed in more detail.

Finally, Fig. 6.10 and Fig. 6.11 show photos of the test machine in the start position for each movement of the sequence from two different angles.

6.2 A general sorting task

This section presents a use case of the proposed motion planning solution for a classic nest sorting task. Fig. 6.12 shows the 2-D layout of the machine utilised in the test, delineating loading and unloading areas, working area, and picking positions in the XY-plane, with reference to the absolute coordinate system. Fig. 6.12 also illustrates the rectangular bounding boxes of the parts in the nest, the stacks' position in the unloading area, and the sequence in which the parts are loaded and unloaded by the end-effectors. If the same loading or sorting numbers are assigned to multiple parts or stacks, this identifies parallel loading or unloading using both arms. The process of sorting the nest is resolved in eight point-to-point movements; four loading tasks and four unloading tasks. In conclusion, an additional task returns the system to its initial configuration, amounting to a total of nine movements. The starting positions, coinciding with the target points of the previous movement, are reported in Tab. 6.4. Regarding the dimensions of the parts involved in the sorting operation, Tab. 6.5 details the rectangular bounding boxes in the form of XY vertex coordinates with respect to the picking point. The polygons used in the frame sampling

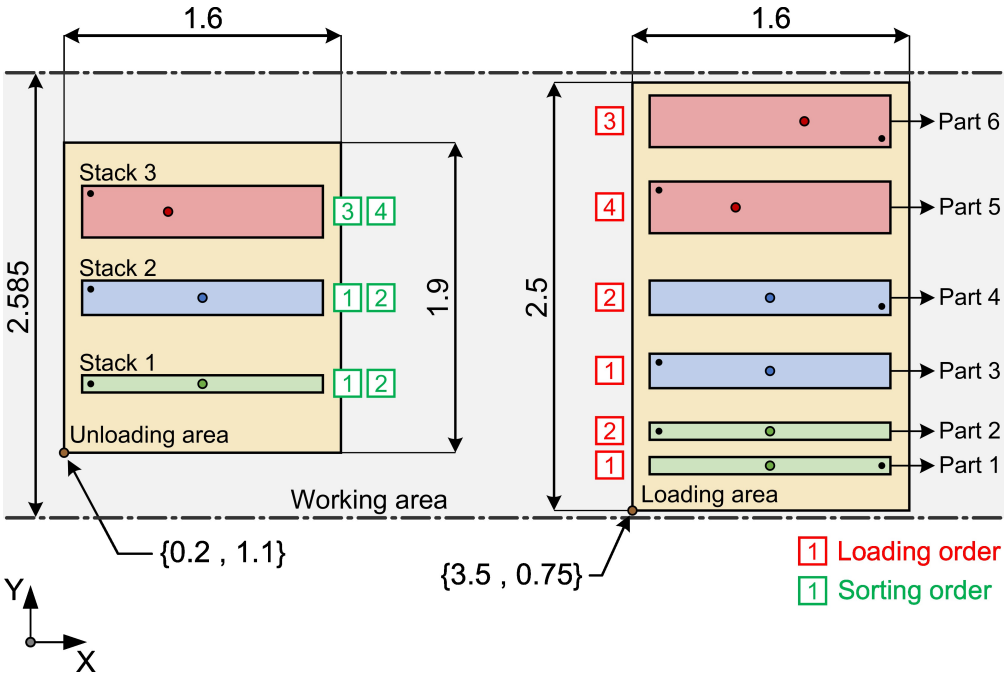
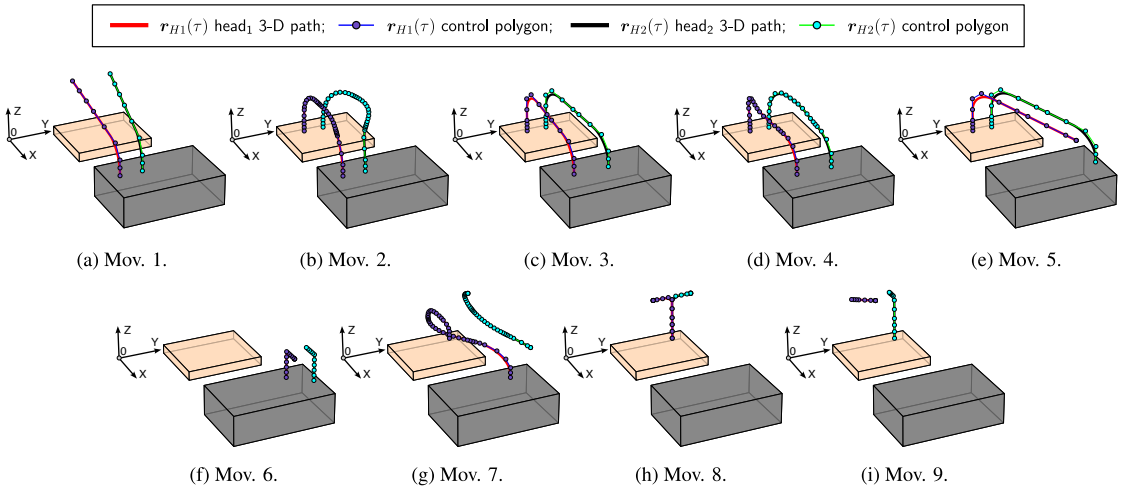


Figure 6.12: Working area 2-D layout referred to the case study. All dimensions are expressed in metres.

analysis are subsequently defined as reported in Section 3.2.2 by adopting an edge lenght of the head octagon of $l_e = 0.0746$ m and an outer offset of $\varepsilon = 0.025$ m. According with the proposed approach, the parameters that still must be defined to uniquely determine the geometric paths of the end-effectors are: $z_{sf} = 1.13$ m, $z_{tr} = 2$ m, $L_{T,min} = 0.225$ m, $A_{ch,min} = 0.05$ m.

The geometric paths computed using the approach detailed in Section 3.2 are shown in Fig. 6.13. In the first movement the start heights of the two heads are equal to the travel height z_{tr}, so the master path $\boldsymbol{r}_\phi(\tau)$ is missing the rise section by defining a three-segment trajectory (case 4 in Fig. 3.6). A similar case is the movement 9, where the target height of the two heads coincides with z_{tr}, and the master path is missing the fall section (case 8 in Fig. 3.6). In movements 1, 3, 5, and 6, end-effectors execute the parts loading, and as previously discussed, no deviation from the geometric paths calculated in accordance with Section 3.2.1 is required. In particular, movement 6 performs the loading of part 5 with head 1, and is executed as a rectangular trajectory due to a transition length L_T, calculated with (3.4), lower than the threshold $L_{T,min}$. The master path is therefore separated into three sub-paths, corresponding to cases 15, 14, and 16 in Fig. 3.6. Different is movement 8, where L_T is slightly greater than $L_{T,min}$, and the master path is defined according to case 10 in Fig. 3.6. From the path planning point of view, the most interesting movements are 2, 4 and 7, which perform the sorting of loaded parts. As described in Tab. 6.4, each of these



(a) Mov. 1. (b) Mov. 2. (c) Mov. 3. (d) Mov. 4. (e) Mov. 5.

(f) Mov. 6. (g) Mov. 7. (h) Mov. 8. (i) Mov. 9.

Figure 6.13: End-effector geometric paths for the nine movement of the case study.

Table 6.4: Start positions of the nine movements of the sequence.

Mov.	Axes						
	x_s(m)	$y_{s,1}$(m)	$z_{s,1}$(m)	$w_{s,1}$(rad)	$y_{s,2}$(m)	$z_{s,2}$(m)	$w_{s,2}$(rad)
1	0.5	1.500	2.00	0	2.500	2.00	0
2	4.3	1.025	1.13	0	1.575	1.13	0
3	1.0	1.500	0.51	π	2.000	0.51	0
4	4.3	1.225	1.13	0	2.075	1.13	0
5	1.0	1.500	0.51	0	2.000	0.51	π
6	4.5	2.525	2.00	0	3.025	1.13	0
7	4.1	2.525	1.13	0	3.025	2.00	0
8	0.8	2.500	0.51	0	3.000	2.00	π
9	0.8	2.00	2.00	0	2.500	0.51	π

Table 6.5: Vertices coordinates of the parts bounding boxes relative to the picking position. All dimensions are expressed in metres.

Part n.	Vertex coordinates			
	Vertex 1	Vertex 2	Vertex 3	Vertex 4
1	$\{-0.7, -0.05\}$	$\{0.7, -0.05\}$	$\{0.7, 0.05\}$	$\{-0.7, 0.05\}$
2	$\{-0.7, -0.05\}$	$\{0.7, -0.05\}$	$\{0.7, 0.05\}$	$\{-0.7, 0.05\}$
3	$\{-0.7, -0.1\}$	$\{0.7, -0.1\}$	$\{0.7, 0.1\}$	$\{-0.7, 0.1\}$
4	$\{-0.7, -0.1\}$	$\{0.7, -0.1\}$	$\{0.7, 0.1\}$	$\{-0.7, 0.1\}$
5	$\{-0.5, -0.15\}$	$\{0.9, -0.15\}$	$\{0.9, 0.15\}$	$\{-0.5, 0.15\}$
6	$\{-0.9, -0.15\}$	$\{0.5, -0.15\}$	$\{0.5, 0.15\}$	$\{-0.9, 0.15\}$

Table 6.6: Parameters of the nine computed movements. Movements 2, 4, and 7 reports the number of frames for all step of the anti-collision sequences. Movements 6 reports the number of knots for all sub-paths.

Mov.	Num. of frames	L_T(m)	s_{max}(rad)	Num. of knots	T(s)
1	205	0.87	537.3	15	4.17
2	252/263/263	0.87	806.9	42	4.95
3	163	0.87	663.5	21	4.86
4	252/263	0.87	785.1	30	4.94
5	170	0.87	704.2	21	5.00
6	28	0.2	326.0	3×12	6.62
7	238/236/242	0.87	831.5	46	4.97
8	42	0.25	516.9	21	5.08
9	39	0.292	296.1	18	3.95

movements performs a rotation of one of the parts involved in the task. Consequently, it may be necessary to compute and apply deviation curves to the Y-components of both end-effectors' paths in order to avoid self-collision between the robotic arms. The frame sampling analysis and the anti-collision sequences, followed by the overlap and overrun spline computation, are presented in Tab. 6.7. For each movement, an outer offset of $\varepsilon = 0.025$ m is used to expand the bounding boxes of group 1 and 2. The number of frames and other computed parameters for each movement are shown in Tab. 6.6.

In movement 2, the bounding box of the group 1 exceeds the working area during the rototranslation. According to the pipeline of Fig. 3.11, this movement falls into case 2 because the frame sampling analysis returns $\underline{\boldsymbol{\kappa}} \neq 0$. The bottom overrun spline $\underline{\boldsymbol{\sigma}}(\tau)$ is computed with the method detailed in section 4, and summed to the Y-axis of $\boldsymbol{r}_{H1}(\tau)$. After that, the frame sampling is updated, and the overlap spline $\boldsymbol{\sigma}(\tau)$ is computed and summed to the Y-axis of $\boldsymbol{r}_{H2}(\tau)$. In movement 4, the rotation of head 1 leads to $\boldsymbol{\kappa} \neq 0$, which corresponds to case 3 in the pipeline reported in Fig. 3.11. The overlap spline $\boldsymbol{\sigma}(\tau)$ is then computed and summed to the Y-axes of $\bar{\boldsymbol{r}}(\tau)$ using weights $w_1 = w_2 = 0.5$. Similarly to the 2$^{\text{nd}}$ movement, movement 7 also falls in case 2 of the pipeline of Fig. 3.11, with the difference that it is the bounding box of group 2 that exceeds the working area, resulting in $\overline{\boldsymbol{\kappa}} \neq 0$. The top overrun spline $\overline{\boldsymbol{\sigma}}(\tau)$ is computed and summed to the Y-axis of $\boldsymbol{r}_{H2}(\tau)$. Hereafter, the frame sampling is updated, and the overlap spline $\boldsymbol{\sigma}(\tau)$ is computed and summed to the Y-axis of $\boldsymbol{r}_{H1}(\tau)$.

The motion profile computed for the complete paths sequence using the pipeline reported in Section 4 is shown in Fig. 6.14. A pause of 1 s is inserted between successive movements. The following parameters are used for both Algorithm 3 and 4: $J^{peak} = 400$ rad/s^3, number of samples N $= 150$, Tol$= 10^{-9}$, initial incremental step $\varepsilon = 0.1$, step ratio $\phi = 0.1$. Tab. 6.6 details the periods T for each movement, corresponding to the cost function (4.6). In Fig. 6.15 the joint trajectories computed with (4.1), (4.2) and (4.3) are reported, highlighting the axis limits that terminate the algorithms of the pipeline of Fig. 4.1.

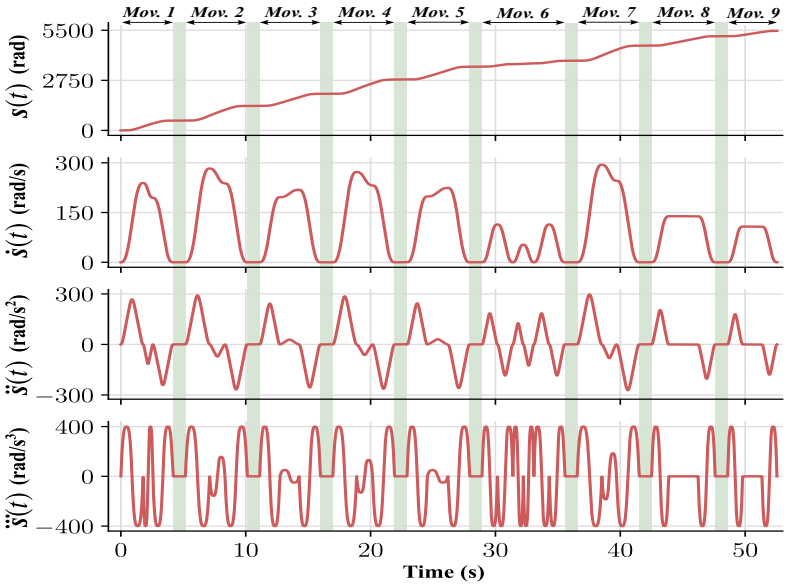
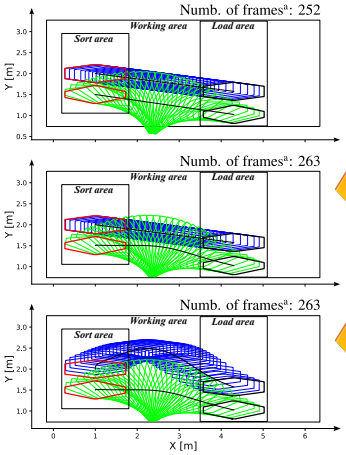
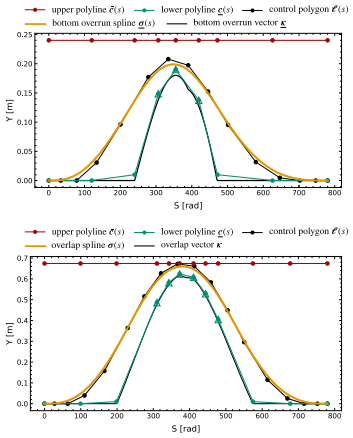
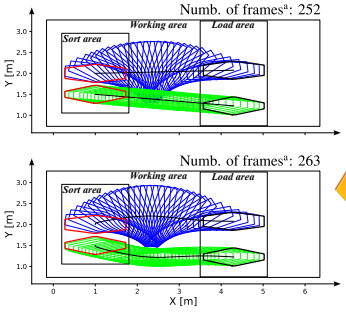
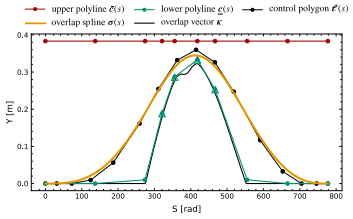
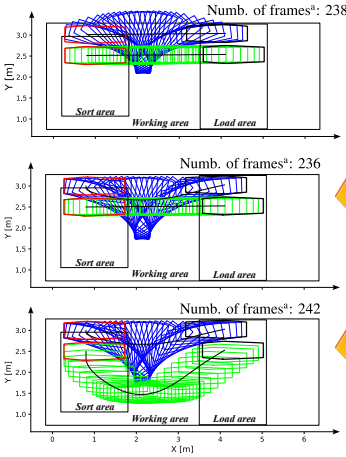
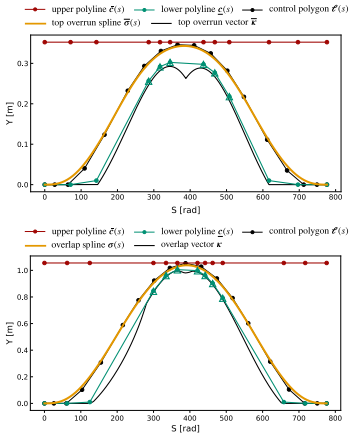


Figure 6.14: Motion profile of the sorting sequence under study.

Starting to analyse the results obtained from movement 1, the motion profile computation terminates due to the condition $J_c^{peak} = J^{peak}$ in Algorithm 4, while Algorithm 3 has previously set the acceleration limit on the X-axis at the transition segment. In Movements 2, 3, 4, 5, and 7, the acceleration limit on the X-axis is reached in both transition segments. The motion profile is asymmetrical due to the different height of the loading and unloading positions. In particular, Algorithm 3 terminates at the transition segment corresponding to the loading zone, while Algorithm 4 allows a higher velocity on the unloading side. In the three geometric paths that constitute movement 6, the calculation of the motion profile terminates after execution of Algorithm 3 for condition $T = 2A$, resulting in symmetrical profiles where a constant velocity section is missing. In this case, it is generally possible to increase the parameter J^{peak} in Z-axis direction, without compromising the machine's operation. The motion profile computation of the movement 8 returns a symmetrical profile where the acceleration limit along the Y-axis is set twice in the middle section of the path, consisting of the two transition segments joined together. In the end, movement 9 also results in a symmetrical motion profile, limited by the velocity limit along the W-axis.

In order to verify the effectiveness of the proposed approach, the movements sequence is tested first in a simulated environment and then using a test machine in the real world. Fig. 6.16 shows the geometric paths trails in the simulated environment, with the end-effectors in target position for each movement. Fig. 6.18 reports the resulting trails of the vertex positions of the parts bounding boxes, corresponding to the movements 2, 4, and 7. Finally, Fig. 6.17 shows the photos of the test machine in the start positions for each movement of the sequence.

Table 6.7: Results of the three movements in which the collision avoidance algorithm detailed in Section 3.2.5 is performed. In frame sampling analysis, green for the bounding box of group 1, blue for group 2. For each movement, an outer offset of $\varepsilon = 0.025$ m is used to expand the bounding boxes.

	Frame sampling	Overlap/Overrun spline	OSQP [28] solver data
Movement 2			Numb. of iteration : 25 Optimal objective: 1.01×10^{-6}m^2/rad^2 CPU Time[b]: 1.95×10^{-4}s Numb. of initial knots: 15 Numb. of iterative refinements: 1 Numb. of iterations: 25 Optimal objective: 7.42×10^{-6}m^2/rad^2 CPU Time[b]: 2.11×10^{-4}s Numb. of initial knots: 18 Numb. of iterative refinements: 1
Movement 4			Numb. of iterations: 25 Optimal objective: 2.67×10^{-6}m^2/rad^2 CPU Time[b]: 1.96×10^{-4}s Numb. of initial knots: 16 Numb. of iterative refinements: 1
Movement 7			Numb. of iterations: 25 Optimal objective: 1.87×10^{-6}m^2/rad^2 CPU Time[b]: 2.67×10^{-4}s Numb. of initial knots: 18 Numb. of iterative refinements: 1 Numb. of iterations: 50 Optimal objective: 1.46×10^{-5}m^2/rad^2 CPU Time[b]: 2.9×10^{-4}s Numb. of initial knots: 19 Numb. of iterative refinements: 1

[a] For illustrative purposes, only one out of five of the calculated frames is shown in the figure.
[b] Performance evaluated on Intel® Core™ i7-11800H CPU, 2.30 GHz..

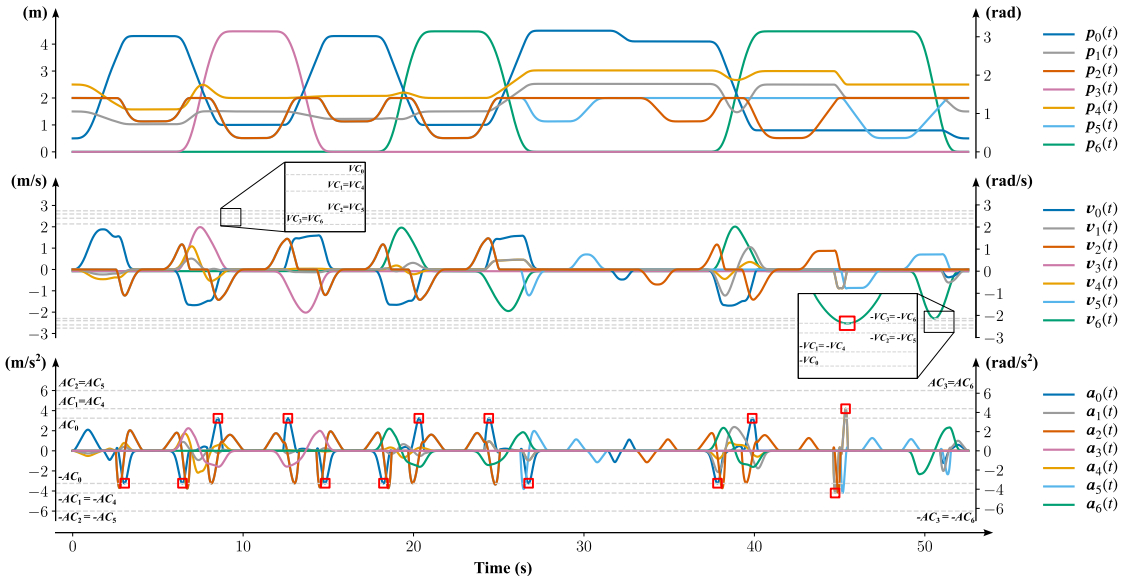


Figure 6.15: Joint trajectory of the movement sequence of the case study. The red square markers are the axis limits that terminate the algorithms of the pipeline in Fig. 4.1. Between each movement, a pause of 1 s is inserted. A 100% velocity override is used.

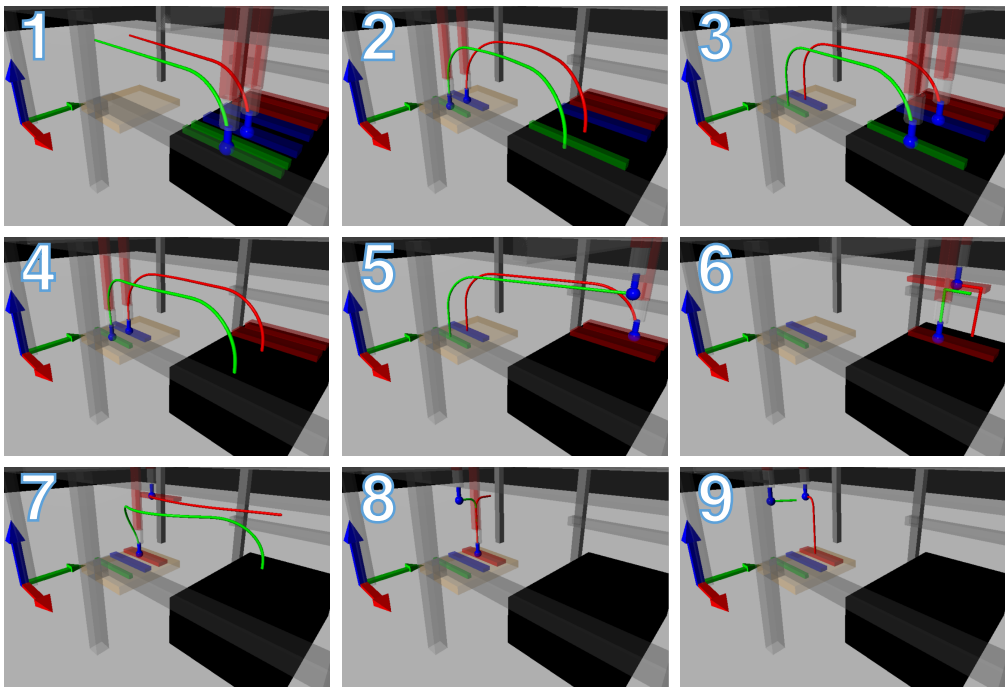


Figure 6.16: The process of sorting of the nest under study in the simulated environment. Sub-pictures 1 to 9 represent the target position and the end-effectors trails.

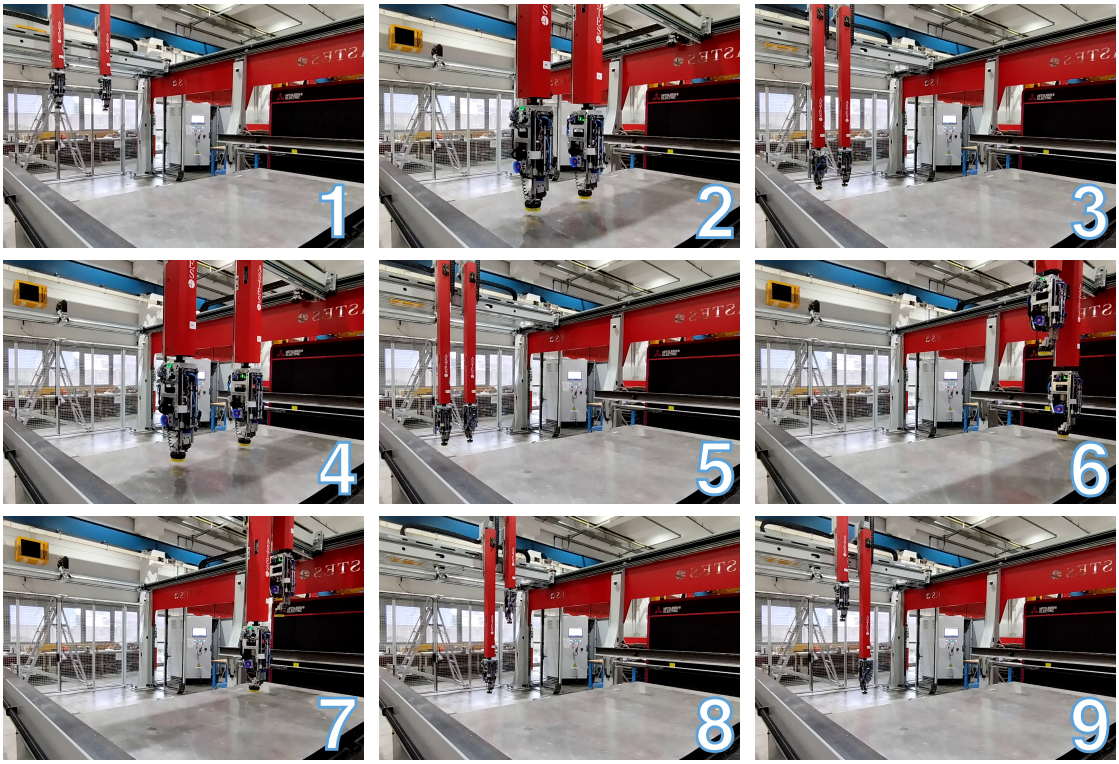


Figure 6.17: The process of sorting of the nest under study in the real world. Sub-pictures 1 to 9 represent the starting position of the point-to-point movements.

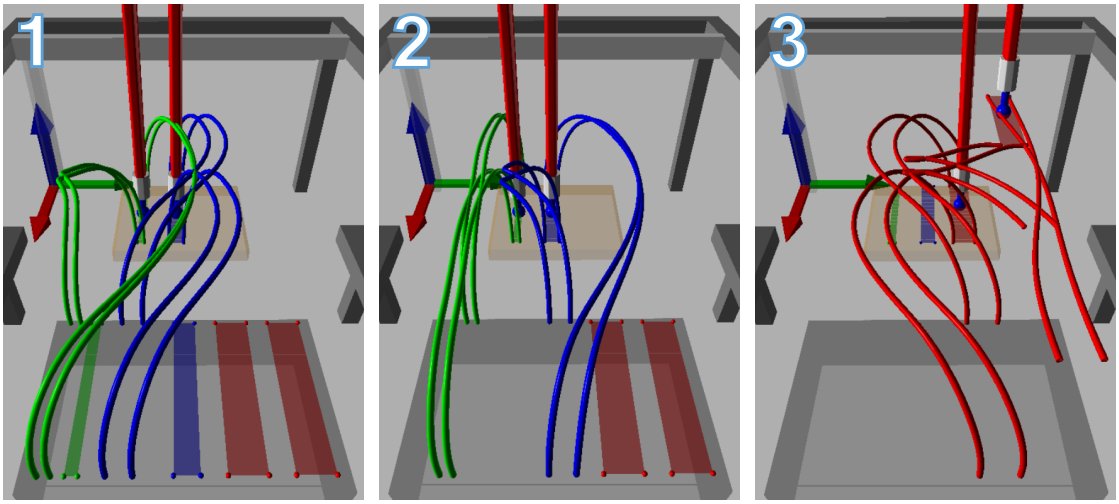


Figure 6.18: Vertex position trails of part bounding boxes. Sub-pictures 1 to 3 correspond to movements 2, 4, and 7.

Chapter 7

Conclusion

This thesis proposes a new path-based motion planning approach for 14-d.o.f., dual-gantry Cartesian robots, which ensures the C^2-continuity of axis trajectories within the velocity and acceleration constraints. The research specifically aims to enhance the motion of a sorting system for laser machine (LST), characterising this work in the field of pick-and-place application. The geometric path is described by a multi-dimensional spline in its B-form; first by defining a master path through a classical five-segment planar pick-and-place trajectory, where adjacent straight lines are blended by quintic Bézier curves with minimized curvature, and then the geometric paths of the end-effectors are defined using a path scaling technique. The problem of self-collision involving the loaded parts during rotation is addressed through bounding box sampling and the subsequent resolution of a quadratic programming (QP) problem. This process yields deviation splines that, when summed to the initially computed path, ensure a collision-free path with minimal curvature. The motion profile is defined using an IPTP-based, time-optimal algorithm, represented by a 1-D time-parametrised spline. The problem of maximum geometric path curvature is partially addressed through asymmetrical motion profile planning, which defines two velocity levels. It should be noted that the solution proposed in this work with regard to the calculation of the motion profile is only a starting point for the definition of a more complex algorithm. Using the IPTP framework, it will in fact be possible to extend this methodology in order to obtain an even more optimal solution, in addition to the possibility of adding dynamic constraints for the stability of the tool-part contact.

The efficacy of the proposed method is validated through a simulation and experiments. The proposed approach significantly reduces the point-to-point motion time compared to solutions traditionally adopted in the specific industrial field. In more detail, the performance gain is highly variable depending on the specific nest layout and machine configuration. In movements where the anti-collision algorithm is not used, the time gain is relatively limited to an approximate average of 10-15%. In more complex movements, where the anti-collision algorithm has a preponderant contribution, the gain achieved can be up to 40% higher than the solution briefly outlined in Section 1.3. Based on the experience of

numerous executed job macro lists, a rough estimate of a time average gain is worth about 20-25%. It should also be noted that the results obtained are strongly influenced by the start and target positions, in this work still calculated through a feasibility study based on the previous anti-collision algorithm. However, the proposed solution allows for more possible solutions, so the possibility of further performance improvement by calculating start and target points with the proposed algorithm is not excluded.

The proposed solution is also easily scalable to different machine configurations with one, two or four end-effectors respectively, with the consequent advantages in terms of customizing the product ad hoc for the customer. In the particular industrial field, this last solution is intriguing since it permits the rototranslation of parts loaded simultaneously by two end-effectors (due to high weight or deflection of the parts) and the translation of parts by four arms. This task is compatible with the presented software structure, and the solution adopted has been detailed in the text. The authors presume that an accurate study of the kino-dynamic limits that considers the constraints introduced by the gripping tool can provide the basis for further performance improvements. Future research could extend this work to other applications, exploring the planning of collision-free paths in three-dimensional space and evaluating the benefits of adopting a corner smoothing algorithm that allows for the C^3-continuity of axis trajectories.

Appendix A

Channel problem numerical benchmark

Table A.1: Numerical benchmark of the channel problem. This benchmark corresponds to the channel of Fig. 2.5a. The matrices are given in CSC format, in accordance with the OSQP solver interface [36]. In particular, for the matrix $\overline{P}$, only its upper triangular part is reported.

Channel data	
c_x =	(0.0, 0.6, 1.20900256, 1.7911149, 2.19411575, 2.5, 3.73145022, 4.47778724)
c_under =	(0.0, 0.0, 0.27, 0.37186333, 0.16986417, 0.1, 0.0, 0.0)
c_over =	(0.5, 0.5, 0.4862461, 0.52955628, 0.3285567, 0.34862461, 0.35, 0.35)
Input data	
n =	36
m =	80
P_nnz =	33
P_x =	(84.872169920163, -54.435253810863, 64.268719015406, 10.535855576296, -34.4171282159, 42.61168255302, 7.023903717531, -23.747483997366, 30.341352793415, 5.01707408395, -19.638261414321, 37.499045153184, 6.020488900741, -31.306542283851, 50.572106766221, 8.428684461037, -33.714737844147, 50.572106766221, 8.428684461037, -31.306542283851, 37.499045153184, 6.020488900741, -19.638261414321, 30.341352793415, 5.01707408395, -23.747483997366, 42.61168255302, 7.023903717531, -34.4171282159, 64.268719015406, 10.535855576296, -54.435253810863, 84.872169920163)
P_i =	(0, 0, 1, 0, 1, 2, 1, 2, 3, 2, 3, 4, 3, 4, 5, 4, 5, 6, 5, 6, 7, 6, 7, 8, 7, 8, 9, 8, 9, 10, 9, 10, 11)
P_p =	(0, 1, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33)
q =	(0.0, 0.0, 0.0,

A_x = (1.0, 1.0, 8.467724041901, -3.629024589386, 8.467724041901, -3.629024589386, 4.838699452515, -32.778217348477, 1.0, 1.0, -3.629024589386, 6.532244260895, -2.903219671509, -3.629024589386, 6.532244260895, -2.903219671509, 14.047807435061, 1.0, 1.0, 0.074999987419, 0.074999987419, -2.903219671509, 5.322569397767, -2.419349726258, -2.903219671509, 5.322569397767, -2.419349726258, 1.0, 1.0, 0.925000012581, 0.925000012581, -2.419349726258, 4.49307806305, -2.073728336792, -2.419349726258, 4.49307806305, -2.073728336792, 1.0, 1.0, 0.799999988387, -2.073728336792, 4.976948008301, -2.903219671509, -2.073728336792, 4.976948008301, -2.903219671509, 1.0, 1.0, 0.200000011613, 0.629999993032, -2.903219671509, 5.806439343018, -2.903219671509, -2.903219671509, 5.806439343018, -2.903219671509, 1.0, 1.0, 0.370000006968, -2.903219671509, 5.806439343018, -2.903219671509, -2.903219671509, 5.806439343018, -2.903219671509, 1.0, 1.0, -2.903219671509, 4.976948008301, -2.073728336792, -2.903219671509, 4.976948008301, -2.073728336792, 1.0, 1.0, -2.073728336792, 4.49307806305, -2.419349726257, -2.073728336792, 4.49307806305, -2.419349726257, 1.0, 1.0, -2.419349726257, 5.322569397767, -2.903219671509, -2.419349726257, 5.322569397767, -2.903219671509, 1.0, 1.0, -2.903219671509, 6.532244260895, -3.629024589386, -2.903219671509, 6.532244260895, -3.629024589386, -32.778217348477, 1.0, 1.0, -3.629024589386, 8.467724041901, -3.629024589386, 8.467724041901, 18.730409913415, 4.838699452515, 0.067720676437, 0.00891794916, 0.000066133473, 0.00000496001, 1.0, 1.0, 0.01034472186, 0.10113623388, 0.011088378987, 0.000011022246, 0.000841823862, 1.0, 1.0, 0.000674065424, 0.017216569912, 0.118849787147, 0.007365946232, 0.015727232898, 1.0, 1.0, 0.000014880031, 0.001029171558, 0.015237152198, 0.115409523877, 0.001913584291, 0.107896597262, 0.00153086741, 1.0, 1.0, 0.000223200472, 0.080370540205, 0.002870376436, 0.000206460439, 0.064870506551, 1.0, 1.0, 0.002870376436, 0.080370540205, 0.002870376436, 0.01837041009, 1.0, 1.0, 0.0, 0.002870376436, 0.080370540205, 0.002870376436, 0.000574075321, 1.0, 1.0, 0.0, 0.002870376436, 0.080370540205, 0.000223200472, 0.0, 1.0, 1.0, 0.0, 0.001913584291, 0.115409523877, 0.015237152198, 0.001029171558, 0.000014880031, 1.0, 1.0, 0.0, 0.007365946232, 0.118849787147, 0.017216569912, 0.000674065424, 1.0, 1.0, 0.000011022246, 0.011088378987, 0.10113623388, 0.01034472186, 1.0, 1.0, 0.000066133473, 0.00891794916, 0.067720676437, 1.0, 1.0, 0.067720676437, 0.00891794916, 0.000066133473, 0.00000496001, 1.0, 1.0, 0.01034472186, 0.10113623388, 0.011088378987, 0.000011022246, 0.000841823862, 1.0, 1.0, 0.000674065424, 0.017216569912, 0.118849787147, 0.007365946232, 0.015727232898, 1.0, 1.0, 0.000014880031, 0.001029171558, 0.015237152198, 0.115409523877, 0.001913584291, 0.107896597262, 1.0, 1.0, 0.000223200472, 0.080370540205, 0.002870376436, 0.000206460439, 0.001808337135, 1.0, 1.0, 0.002870376436, 0.080370540205, 0.002870376436, 0.051695479071, 1.0, 1.0, 0.0, 0.002870376436, 0.080370540205, 0.002870376436, 0.031545437571, 1.0, 1.0, 0.0, 0.002870376436, 0.080370540205, 0.000223200472, 0.001062039301, 1.0, 1.0, 0.0, 0.001913584291, 0.115409523877, 0.015237152198, 0.001029171558, 0.000014880031, 0.0, 1.0, 1.0, 0.0, 0.007365946232, 0.118849787147, 0.017216569912, 0.000674065424, 1.0, 1.0, 0.000011022246, 0.011088378987, 0.10113623388, 0.01034472186, 1.0, 1.0, 0.000066133473, 0.00891794916, 0.067720676437, 1.0, 1.0)

A_i = (0, 12, 28, 29, 40, 41, 76, 77, 1, 13, 28, 29, 30, 40, 41, 42, 77, 2, 14, 24, 26, 29, 30, 31, 41, 42, 43, 3, 15, 24, 26, 30, 31, 32, 42, 43, 44, 4, 16, 25, 31, 32, 33, 43, 44, 45, 5, 17, 25, 27, 32, 33, 34, 44, 45, 46, 6, 18, 27, 33, 34, 35, 45, 46, 47, 7, 19, 34, 35, 36, 46, 47, 48, 8, 20, 35, 36, 37, 47, 48, 49, 9, 21, 36, 37, 38, 48, 49, 50, 10, 22, 37, 38, 39, 49, 50, 51, 78, 11, 23, 38, 39, 50, 51, 78, 79, 0, 1, 2, 24, 28, 52, 0, 1, 2, 3, 24, 29, 53, 0, 1, 2, 3, 24, 30, 54, 0, 1, 2, 3, 4, 24, 25, 31, 55, 3, 4, 5, 24, 25, 32, 56, 4, 5, 6, 25, 33, 57, 4, 5, 6, 7, 25, 34, 58, 5, 6, 7, 8, 25, 35, 59, 6, 7, 8, 9, 10, 11, 36, 60, 7, 8, 9, 10, 11, 37, 61, 8, 9, 10, 11, 38, 62, 9, 10, 11, 39, 63, 12, 13, 14, 26, 40, 64, 12, 13, 14, 15, 26, 41, 65, 12, 13, 14, 15, 26, 42, 66, 12, 13, 14, 15, 16, 26, 43, 67, 15, 16, 17, 26, 27, 44, 68, 16, 17, 18, 27, 45, 69, 16, 17, 18, 19, 27, 46, 70, 17, 18, 19, 20, 27, 47, 71, 18, 19, 20, 21, 22, 23, 27, 48, 72, 19, 20, 21, 22, 23, 49, 73, 20, 21, 22, 23, 50, 74, 21, 22, 23, 51, 75)

A_p = (0, 8, 17, 27, 37, 46, 56, 65, 73, 81, 89, 98, 106, 112, 119, 126, 135, 142, 148, 155, 162, 170, 177, 183, 188, 194, 201, 208, 216, 223, 229, 236, 243, 252, 259, 265, 270)

l = (0.0, 0.0, 0.10049296287914557, 0.2754246734799375, 0.3598084973225302, 0.233744247695904, 0.12030111875166072, 0.0792471206459699, 0.04008814910177148, 0.006523316349601328, 0.0, 0.0, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, 0.27, 0.37186333, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, -inf, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)

u = (inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, 0.5, 0.5, 0.4948808512513204, 0.48855255890780636, 0.5244308146030696, 0.39212067119296967, 0.3427932800921568, 0.3489100430273474, 0.3494486316060691, 0.34991027895925925, 0.35, 0.35, inf, inf, 0.4862461, 0.3285567, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf, 0.0, 0.0, 0.0, 0.0)

Results

x = (0.0000003056369794922329, 0.00000147145512427 8828, 0.12942623198264538, 0.3080532574176377, 0.40543028498807143, 0.3591552095543335, 0.27622992955913106, 0.19300593651884318, 0.09264664556164519, 0.028908605230654496, 0.0000000799921857229 0267, 0.0000000238988829114 4244, 0.000019435213933097616, 0.000894796102157 1873, 0.00024852615371760765, -0.22955240029325558, -0.3368440640288313, -0.10657316009491917, -0.0010328603338968876, -0.2592274242658301, -0.09256152764549273, 0.00006502472821103738, 0.00036143153416504255, 0.0000162308712401745, 0.011041469693972166, 0.6367984860764005, 0.24900338097780952, 0.5432813770059411, 0.000026046375640042737, 0.0002648583889165337, -0.00035948286995235603, 0.03352371731281793, 0.0668154571829971, 0.07812696564842433, 0.08531515992715061, 0.000036057448090183906)

Appendix B

Job macro list example

```
0001 NULL_MCR SortCAM Version: 1.0.0.0 - Job list name: OUTMOVES-COMPARE-29
0002 ZSD,3,6:2::,C:C::,IN:IN::,,,,,,,,,,,,,,,,,
0003 EJL,3,,,,,,,,,,,,101,,,,,,,
0004 PT4,3,1:2:3:4,3:3:3:3,1103:1103:1103:1103,H:H:H:H,120:120:120:120,0:0:0:0,,0
    ,,,,,,,100,,,,,0
0005 PP4,3,1:::4,H:::H,1112.19:381.55,2428.47:487.65:2506.35:477.55,19:::19,0.00:
    0.00:0.00:0.00,595:::595,0,,372;-464;372;461;-369;461;-369;-464:::369;-461;369;
    464;-372;464;-372;-461,1:::1,73:::73,101:::101,71:::71,100,1:::1,0:::0,,,0
0006 PP4,3,:2::,:H::,1112.19:298.37,2428.47:487.65:2506.35:477.55,:19::,0.00:0.00:
    0.00:0.00,:595::,0,:::,:372;-464;372;461;-369;461;-369;-464::,:1::,:73::,:101:
    :,:71::,100,:1::,:0::,,:::,0
0007 PP4,3,::3:,::H:,1195.37:381.55,2449.37:570.83:2476.05:560.73,::19:,0.00:0.00:
    0.00:0.00,::595:,0,:::,::369;-461;369;464;-372;464;-372;-461:,::1:,::73:,::101:
    ,::71:,100,::1:,::0:,,:::,0
0008 ZSD,3,6:2::,M:C::,IN:IN::,,,,,,,,,,,,,,,,,
0009 DP4,3,:2::,:H::,1141.55:1141.55:1957.52:1957.52,2974.37:1974.23:2974.37:1974.23,
    0,0.00:0.00:180.00:180.00,:6::,:1::,:1::,:P::,:570::,:1::,:0::,,100,:0:,,,,
0010 ZSD,3,6:2::,M:C::,IN:IN::,,,,,,,,,,,,,,,,,
0011 DP4,3,:::4,:::H,325.59:325.59:1141.55:1141.55,2160.55:1259.00:2974.37:1974.23,
    0,0.00:0.00:0.00:0.00,:::6,:::1,:::1,:::P,:::570,:::1,:::0,,100,:::0,,,,
0012 ZSD,3,6:2::,M:C::,IN:IN::,,,,,,,,,,,,,,,,,
0013 DP4,3,::3:,::H:,325.59:325.59:1141.55:1141.55,1974.23:1259.00:1974.23:1259.00,
    0,0.00:0.00:0.00:0.00,::6:,::1:,::1:,::P:,::570:,::1:,::0:,,100,::0,,,,
0014 ZSD,3,6:2::,M:C::,IN:IN::,,,,,,,,,,,,,,,,,
0015 DP4,3,1:::,H:::,1141.56:1141.56:1761.96:1761.96,1974.23:1259.00:1974.23:1259.00,
    0,0.00:0.00:0.00:0.00,6:::,1:::,1:::,P:::,570:::,1:::,0:::,,100,0::,,,,
0016 ZSD,3,6:2::,C:C::,IN:IN::,,,,,,,,,,,,,,,,,
0017 DT4,3,1:2:3:4,3:3:3:3,1103:1103:1103:1103,H:H:H:H,,,,0,,,,,,,100,,,,,0
0018 PT4,3,1::3:,3::3:,1103::1103:,H::H:,120::120:,0::0:,,0,,,,,,,100,,,,,0
0019 PP4,3,1:::,H:::,1111.10:488.55,1427.55:715.63:1483.10:734.00,19:::,0.00:0.00:
    0.00:-180.00,595:::,0,:::,369;-461;369;464;-372;464;-372;-461:::,1:::,73:::,101
    :::,71:::,100,1:::,4:::,,:::,0
0020 PP4,3,::3:,::H:,1220.24:379.41,1449.23:715.63:1483.10:734.00,::19:,0.00:0.00:
    0.00:-180.00,::595:,0,:::,::372;-464;372;461;-369;461;-369;-464:,::1:,::73:,::
```

```
    101:,::71:,100,::1:,::4:,,:::,0
0021 ZSD,3,6:2::,M:C::,IN:IN::,,,,,,,,,,,,,,,,
0022 DP4,3,1:::,H:::,1141.55:1141.55:1982.39:1982.39,1974.23:1240.63:1974.23:
    1259.00,0,180.00:0.00:0.00:0.00,6:::,1:::,1:::,P:::,570:::,1:::,0:::,,100,
    0::,,,,
0023 ZSD,3,6:2::,M:C::,IN:IN::,,,,,,,,,,,,,,,,
0024 DP4,3,::3:,::H:,519.00:519.00:1141.55:1141.55,1974.23:1240.63:1974.23:
    1259.00,0,0.00:0.00:0.00:0.00,::6:,::1:,::1:,::P:,::570:,::1:,::0:,,100,
    ::0,,,,
0025 PNC,3,,1,6000001,,7.06,6,S,1,6,42,6,6,0,,,,,,0
0026 ZSD,3,2:6::,C:C::,IN:OUT::,,,,,,,,,,,,,,,,
0027 DT4,3,1::3:,3::3:,1103::1103:,H::H:,,,,0,,,,,,100,,,,,0
0028 ZSD,3,2:::,M:::,IN:::,,,,,,,,,,,,,,,,
0029 SCS,3,101,1,3000,3000,0,,0,,,,,201,2,97,100,,,,,0
0030 ZSD,3,2:::,C:::,OUT:::,,,,,,,,,,,,,,,,
0031 PAL,3,,1,1,3025,3025,0,,,,,,101,,100,,,,,0
0032 END,3,,,,,,,,,,,,,,,,,,
```

Bibliography

[1] S. Lee, H. Moradi, and Chunsik Yi. "A real-time dual-arm collision avoidance algorithm for assembly". In: *Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning (ISATP'97) - Towards Flexible and Agile Assembly and Manufacturing -*. 1997, pp. 7–12. DOI: 10.1109/ISATP.1997.615376.

[2] Sukhan Lee et al. "Geometric collision detection and potential field based time delay planning for dual arm systems". In: *Proceedings of International Conference on Robotics and Automation*. Vol. 1. 1997, 882–887 vol.1. DOI: 10.1109/ROBOT.1997.620145.

[3] "Collision-free motion planning of dual-arm reconfigurable robots". In: *Robotics and Computer-Integrated Manufacturing* 20.4 (2004), pp. 351–357. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2004.01.002.

[4] "Rapidly-exploring random trees : a new tool for path planning". In: *The annual research report* (1998).

[5] Jun Kurosu, Ayanori Yorozu, and Masaki Takahashi. "Simultaneous Dual-Arm Motion Planning for Minimizing Operation Time". In: *Applied Sciences* 7.12 (2017). DOI: 10.3390/app7121210.

[6] Wubin Shi et al. "Obstacle Avoidance Path Planning for the Dual-Arm Robot Based on an Improved RRT Algorithm". In: *Applied Sciences* 12.8 (2022). DOI: 10.3390/app12084087.

[7] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numer. Math.* 1.1 (1959), pp. 269–271. ISSN: 0029-599X. DOI: 10.1007/BF01386390.

[8] Sertac Karaman and Emilio Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning". In: *The International Journal of Robotics Research* 30 (3 2011). DOI: 10.1177/0278364911406761.

[9] R. Bellman. *Dynamic Programming*. Dover Books on Computer Science. Dover Publications, 2013. ISBN: 9780486317199.

[10] Tomas Berglund et al. "Planning Smooth and Obstacle-Avoiding B-Spline Paths for Autonomous Mining Vehicles". In: *IEEE Transactions on Automation Science and Engineering* 7.1 (2010), pp. 167–172. DOI: 10.1109/TASE.2009.2015886.

[11] Andreas Völz and Knut Graichen. "An Optimization-Based Approach to Dual-Arm Motion Planning with Closed Kinematics". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 8346–8351. DOI: 10.1109/IROS.2018.8593927.

[12] Youdong Chen and Ling Li. "Collision-free trajectory planning for dual-robot systems using B-splines". In: *International Journal of Advanced Robotic Systems* 14 (July 2017), p. 172988141772802. DOI: 10.1177/1729881417728021.

[13] Wolfgang Boehm and Hartmut Prautzsch. "The insertion algorithm". In: *Computer-Aided Design* 17.2 (1985), pp. 58–59. ISSN: 00104485. DOI: 10.1016/0010-4485(85)90246-5.

[14] L. A. Piegl and W. Tiller. *The NURBS Book*. 2nd. New York, NY, USA: Springer-Verlag, 1996. ISBN: 3540615458.

[15] Younsung Choi et al. "Dual-arm robot motion planning for collision avoidance using B-spline curve". In: *International Journal of Precision Engineering and Manufacturing* 18 (June 2017), pp. 835–843. DOI: 10.1007/s12541-017-0099-z.

[16] Marco Riboli et al. *Collision-Free and Smooth Motion Planning of Dual-Arm Cartesian Robot Based on B-Spline Representation*. Tech. rep. 2023. DOI: 10.2139/ssrn.4432038.

[17] ASTES4 SA. *SORTING - Different needs, one solution: Astes4 sorting system.* Accessed: 2023-08-03. URL: https://www.astes4.ch/products-sorting/.

[18] Fabio Pagani, Samuele Buschini, and Roberto Zaffaroni. *Multi-tool gripper head of a sorting apparatus and operating method thereof, Patent WO2021209932A1, oct 21, 2021.*

[19] Josh Barnett et al. "Work distribution of multiple Cartesian robot arms for kiwifruit harvesting". In: *Computers and Electronics in Agriculture* 169 (2020), p. 105202. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2019.105202.

[20] Frederick Struckmeier and Fernando León. "Nesting in the sheet metal industry: dealing with constraints of flatbed laser-cutting machines". In: *Procedia Manufacturing* 29 (Jan. 2019), pp. 575–582. DOI: 10.1016/j.promfg.2019.02.115.

[21] G. Scheithauer. *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods*. International Series in Operations Research & Management Science. Springer International Publishing, 2017. ISBN: 9783319644035.

[22] C. de Boor. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York, 2001. ISBN: 0387953663.

[23] W. Tiller. "Knot-removal algorithms for NURBS curves and surfaces". In: *Computer-Aided Design* 24.8 (1992), pp. 445–453. ISSN: 00104485. DOI: 10.1016/0010-4485(92)90012-Y.

[24] Les Piegl and Wayne Tiller. "Software-engineering approach to degree elevation of B-spline curves". In: *Computer-Aided Design* 26.1 (1994), pp. 17–28. ISSN: 00104485. DOI: 10.1016/0010-4485(94)90004-3.

[25] M.A. Watkins and A.J. Worsey. "Degree reduction of Bézier curves". In: *Computer-Aided Design* 20.7 (1988), pp. 398–405. ISSN: 00104485. DOI: 10.1016/0010-4485(88)90216-3.

[26] Matthias Eck. "Degree reduction of Bézier curves". In: *Computer Aided Geometric Design* 10.3-4 (1993), pp. 237–251. ISSN: 01678396. DOI: 10.1016/0167-8396(93)90039-6.

[27] Les Piegl and Wayne Tiller. "Algorithm for degree reduction of B-spline curves". In: *Computer-Aided Design* 27.2 (1995), pp. 101–110. ISSN: 00104485. DOI: 10.1016/0010-4485(95)92150-Q.

[28] Marco Riboli et al. "A New Framework for Joint Trajectory Planning Based on Time-Parameterized B-Splines". In: *Computer-Aided Design* 154 (2023), p. 103421. ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2022.103421.

[29] Gershon Elber. "Free Form Surface Analysis Using a Hybrid of Symbolic and numeric computation". PhD thesis. Utah, 1992, p. 121.

[30] Les Piegl and Wayne Tiller. "Symbolic operators for NURBS". In: *Computer-Aided Design* 29.5 (1997), pp. 361–368. ISSN: 00104485. DOI: 10.1016/S0010-4485(96)00074-7.

[31] L. A. Piegl and W. Tiller. "Curve Interpolation with Arbitrary End Derivatives". In: *Engineering with Computers* 16.1 (2000), pp. 73–79. ISSN: 0177-0667. DOI: 10.1007/s003660050038.

[32] David Lutterkort and Jörg Peters. "Smooth Paths in a Polygonal Channel." In: Jan. 1999, pp. 316–321. DOI: 10.1145/304893.304985.

[33] David Lutterkort and Jörg Peters. "Tight Linear Envelopes for Splines". In: *Numerische Mathematik* 89 (Oct. 2001), pp. 735–748. DOI: 10.1007/s002110100181.

[34] B. Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2.

[35] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003.

[36] Bartolomeo Stellato. *OSQP*. 2023. URL: https://osqp.org/.

[37] Burak Sencer and Eiji Shamoto. "A curvature optimal sharp corner smoothing algorithm for high-speed feed motion generation of NC systems along linear tool paths". In: *International Journal of Advanced Manufacturing Technology* 76 (Oct. 2014), pp. 1977–1992. DOI: 10.1007/s00170-014-6386-2.

[38] Jixiang Yang et al. "An analytical C3 continuous tool path corner smoothing algorithm for 6R robot manipulator". In: *Robotics and Computer-Integrated Manufacturing* 64 (2020), p. 101947. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.101947.

[39] Zhang Yongbin et al. "An analytical G3 continuous corner smoothing method with adaptive constraints adjustments for five-axis machine tool". In: *The International Journal of Advanced Manufacturing Technology* 109 (July 2020). DOI: 10.1007/s00170-020-05402-x.

[40] Jingfu Peng et al. "An Analytical Method for Decoupled Local Smoothing of Linear Paths in Industrial Robots". In: *Robotics and Computer-Integrated Manufacturing* 72 (2021), p. 102193. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102193.

[41] T.H. Cormen et al. *Introduction to Algorithms, fourth edition*. MIT Press, 2022. ISBN: 9780262046305.

[42] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. USA: Cambridge University Press, 2007. ISBN: 0521880688.

[43] Beckhoff-New Automation Technology. *Beckhoff Information System. ADS Communication.* Accessed: 2023-10-15. URL: https://infosys.beckhoff.com/english.php?content=../content/1033/cx8090_hw/1610551947.html&id=.

[44] L. Biagiotti and C. Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. Springer Berlin Heidelberg, 2008, p. 514. ISBN: 978-3-540-85628-3. DOI: 10.1007/978-3-540-85629-0.

Acknowledgements

I would like to express my gratitude to all the employees of ASTES4 SA, with a special acknowledgment to Cesare Malara, for their ongoing support of the OUTMOVES project. Their contributions were instrumental in ensuring the successful completion of the project by providing everything necessary for the work to be done effectively.

I extend my thanks to my colleague and friend, Matthieu Jaccard, for contributing to my development as both a researcher and an individual.

I would like to express my gratitude to my mentors, Prof. Marco Silvestri and Prof. Alessandra Aimi, for the unwavering support, both professionally and morally, that he has provided throughout my Ph.D. activity.

I thank my parents and my life partner, Lourdes, for their continuous support and trust they have always shown me throughout these years. I owe the results I have achieved to them.