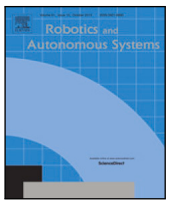
Robotics and Autonomous Systems 170 (2023) 104534

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Collision-free and smooth motion planning of dual-arm Cartesian robot based on B-spline representation

Marco Riboli [a,*], Matthieu Jaccard [b], Marco Silvestri [a,b], Alessandra Aimi [c], Cesare Malara [d]

[a] Department of Engineering and Architecture, University of Parma, Italy
[b] Department of Innovative Technologies, University of Applied Sciences of Southern Switzerland, SUPSI, Lugano, Switzerland
[c] Department of Mathematical, Physical and Computer Sciences, University of Parma, Italy
[d] ASTES4 SA, Balerna, Switzerland

ARTICLE INFO

Keywords:
Collision-free path planning
Dual-arm cartesian robot
Pick-and-place application
B-spline curves
Quadratic programming
Minimizing curvature
Iso-parametric trajectory planning

ABSTRACT

This paper discusses a new approach to motion planning of dual-arm gantry kinematic that solves the self-collision problem while ensuring G^2-continuity and low curvatures. Starting from collision-free start and target points, the proposed method defines the geometric path of the end-effectors with classical five-segment planar trajectories, described by fifth-degree B-splines. The rototranslation of the loaded parts is evaluated in frames to define sampled overlapping curves, used to compute splines that act as deviation curves being added on the previously defined geometric paths in order to prevent collisions. Here, a quadratic programming minimization is employed to reduce the curvature of these spline and preserve the kinematics properties of the initial path. The kinematic constraints of the axes are ensured by iso-parametric trajectory planning, which results in an optimized motion profile to reduce the task execution time. Finally, the proposed approach is applied to a sorting system for laser machine (LST) having a dual-arm 7-d.o.f. gantry kinematic. The results of a case study of a typical sorting operation are presented and discussed to illustrate and clarify the method. The approach presented in this work can be applied to other robotic systems with similar kinematic structures, providing a useful tool for motion planning in pick-and-place applications.

1. Introduction

Motion planning is one of the most important problem in industrial robotic applications. It generally refers to the full process that determines the tasks sequence that a robot must perform in order to move from its current state to a desired target state. This process generally involves finding a collision-free path, which should ensure a good compromise between the arc length of the tool path and the smoothing properties to result in efficient movements. In addition to possible obstacles in the environment, which is generally defined as static or dynamic depending on whether these obstacles are all fixed or not, there are increasingly frequent industrial cases where several manipulators (or several end-effectors of the same robot) move simultaneously in the working area. In this case, the motion cannot be planned considering the end-effectors individually, but must simultaneously consider all end-effectors involved in the motion in order to avoid self-collision and optimize the overall system. This work focuses on the motion planning of a Cartesian dual-arm robotic system, where two gripping tools collaborate in the same environment. In this regard, one of the first examples of collision-free motion planning of a dual-arm robot is provided by Lee et al. [1], who propose an approach based on a virtual road map (VRM) and apply it to the case of two SCARA robots sharing a common workspace. Subsequently, Fei et al. [2] suggest a method based on configuration space (C-Space), where the concepts reachable manifold and contact manifold are successfully applied to the collision-free motion planning of dual-arm SCARATES robot. More generally, the problem of collision-free motion planning finds different solutions in the robotics field, and can be broadly classified into sampling-based, optimization-based and trajectory planning methods.

Sampling-based methods, such as the rapidly-exploring random tree (RRT) algorithm [3] and its variants, are popular for motion planning in high-dimensional configuration spaces, where each dimension corresponds to a degree of freedom (d.o.f.) of the robot. The basic idea is to randomly sample the configuration space and construct a tree-like structure that connects the samples. The algorithm starts with a single sample in the joint space, corresponding to the initial state that defines the root of the tree. A new sample is repeatedly added by selecting a random state in the configuration space. Thereafter, the closest existing sample is selected and, if possible, a new branch of the tree is created. The algorithm continues to add samples until a path connecting the

* Corresponding author.
E-mail address: marco.riboli@unipr.it (M. Riboli).

https://doi.org/10.1016/j.robot.2023.104534
Received 2 April 2023; Received in revised form 8 August 2023; Accepted 21 September 2023
Available online 4 October 2023

start and target positions is found. Kurosu et al. [4] use the RRT algorithm to enable a dual-arm robot to perform pick-and-place tasks without collisions between the arms. More recently, Shi et al. [5] address the same problem by proposing a variant of this algorithm, called GA_RTT, that combines the probabilistic bias method and A* algorithm with RRT, where A* algorithm is a search algorithm obtained by adding a heuristic function to Dijkstra's algorithm [6]. Although sampling-based methods find many applications in the motion planning field, they may require an increasingly large number of samples as the number of dimensions of the configuration space increases, in order to accurately represent the configuration space [7]. This phenomenon is known in the literature as curse of dimensionality (COD), first described by Richard Bellman in the context of approximation theory [8], and leads (in the worst case) to an increase in the execution time of these algorithms.

An alternative solution to sampling-based methods to tackle the problem of collision-free motion planning is offered by optimization-based methods. These methods aim to find a collision-free path that minimizes a cost function, such as the time, curvature or energy required to follow the trajectory. A recent example is provided by Völz and Graichen [9], who offer a solution for the collision-free motion planning of a dual-arm robot that must move an object with both arms, whereby the two arms and the loaded object form a closed kinematic chain. One of the main advantages of these methods is to find high-quality solutions in a relatively short time. However, they can be sensitive to initial conditions and can be difficult to implement in real applications. Many optimization-based methods are indeed designed to work with convex optimization problems, which have a single global minimum. In real applications, such as in motion planning, it is often difficult to formalize the problem to obtain a convex cost function that can be defined in linear or quadratic standard form. On the other hand, iterative methods for non-linear constrained optimization, such as sequential quadratic programming (SQP), usually involve running times that are not compatible with run-time robotic applications [10].

Trajectory planning methods using splines have also been proposed for smooth motion planning in robotics. These methods classically use splines to represent the geometric path by optimizing the shape using computer-aided design (CAD) techniques to find a collision-free path. For example, Chen and Li [11] propose an approach to generate collision-free path of two industrial robots working in a shared workspace by applying the B-spline knot refinement [12,13] and the local modification scheme. The method uses the local properties of B-splines, modifying the trajectory only around the areas where a collision exists, while the remaining curve segments of the trajectory remain in their original position without modification. Another interesting application of B-spline theory for motion planning of a dual-arm robot is provided from Choi et al. [14]. In this work, a technique for post-processing of a previously calculated geometric path (e.g. using a sampling-based approach) is proposed, to ensure the quality of the trajectory in terms of smoothing and preventing collisions.

This paper deals with the collision-free motion planning of dual-arm 7-d.o.f. Cartesian robot, in which the two arms are rigidly linked along the X-axis of the machine. Although the problem of constrained kinematic motion planning of a Cartesian robot is widely studied in the literature [15,16], not many works investigate dual-arm Cartesian kinematics, and classically these refer to dual-gantry systems, where each end-effector has X_i, Y_i and Z_i-axis with $i = 1, 2$, resulting independent of each other. For example, Lin and Chen [17] use a dual-arm Cartesian robot to automate the packaging process in a bottling factory. In this case study, the self-collision of the two arms is prevented at run-time by strategically installed photoelectric sensors. More recently, the same kinematics is used by Barnett et al. [18] in the agricultural field to study the mechanization of kiwi harvesting. The problem of self-collision between the arms (theoretically scalable to more than two) is solved through a special configuration design that provides a partitioning of the work zones assigned to each end-effector and a sorting of the fruits in accordance with the X-axis of the robot.

In industry, and more specifically in pick-and-place applications, the self-collision problem of Cartesian dual-arm kinematics is classically handled with simple strategies that limit the robot's movements to classical rectangular trajectories. 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 categorized as a hybrid technique based on optimization and spline modelling. If no collisions are detected in the task, the two arms follow classical five-segment pick-and-place trajectories [19,20], where quintic Bézier curves blend the adjacent straight lines ensuring G^2-continuity of the geometric path. When a collision is detected, the previously calculated path is appropriately modified by acting on the control points of the spline. During this operation, an optimization-based method that ensures minimum curvature preserves the quality of the initial geometric path. The physical limits of velocity and torque on the axes are respected using an iso-parametric trajectory planning approach (IPTP) and developing an appropriate algorithm to generate the motion profile that uses a previously developed open-source software [21].

The remainder of this paper is organized as follows. Section 2 describes the kinematics and the machine used to test the method. Section 3 illustrates the mathematical theory of the path planning approach, starting with the 4-d.o.f. single-arm kinematics and then extending the solution to the dual-arm system. Section 4 details the motion profile algorithm that follows the geometric path within the kinematic constraints. An application case of the proposed method is presented and discussed in Section 5. Finally, a summary concludes this work in Section 6.

The mathematical notations and abbreviations used in the remainder of the paper, not specified in the text, are summarized below.

Notation
= equal in the equations and equality operator in the algorithmic sections.
:= equal by definition.
← assignment in algorithmic sections.
∧ logical and.
∨ logical or.
$\boldsymbol{\tau}$, $(\tau_i)_{i=0}^{n-1}$, $\{\tau_0, \dots, \tau_{n-1}\}$ various ways of describing the same n-vector.
$\#\boldsymbol{\tau}$ number of elements of the $\boldsymbol{\tau}$ vector.
$\lvert x\rvert$ absolute value of x.
$\lVert \boldsymbol{P}\rVert$ magnitude of vector $\boldsymbol{P}$.
$\mathcal{L}(x \mid f(x_1), f(x_2))$, $\mathcal{L}(x \mid \{x_1, y_1\}, \{x_2, y_2\})$ linear interpolation of the function $f(x) = y$ between points $\{x_1, f(x_1)\}$ and $\{x_2, f(x_2)\}$.
⊙ element-wise product.
$(f \circ g)(x)$ functions composition, defined by $f(g(x))$.
$\mathrm{sgn}(x)$ sign function, defined by $x/\lvert x\rvert$.
x mod y returns the remainder of the integer division x/y.
$\mathrm{proj}(f(x), v)$ projection of the function $f(x)$ on the v-axis.
Acronyms
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.
IPTP := Iso-parametric trajectory planning.
VC := Velocity constraint.
AC := Acceleration constraint.

2. Description of robot kinematic and the test machine

Fig. 1 shows a schematic representation of the sorting system for laser machine (LST) with dual-arm Cartesian kinematics. The robot kinematics solution involves two Cartesian arms 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 $i = 1, 2$.

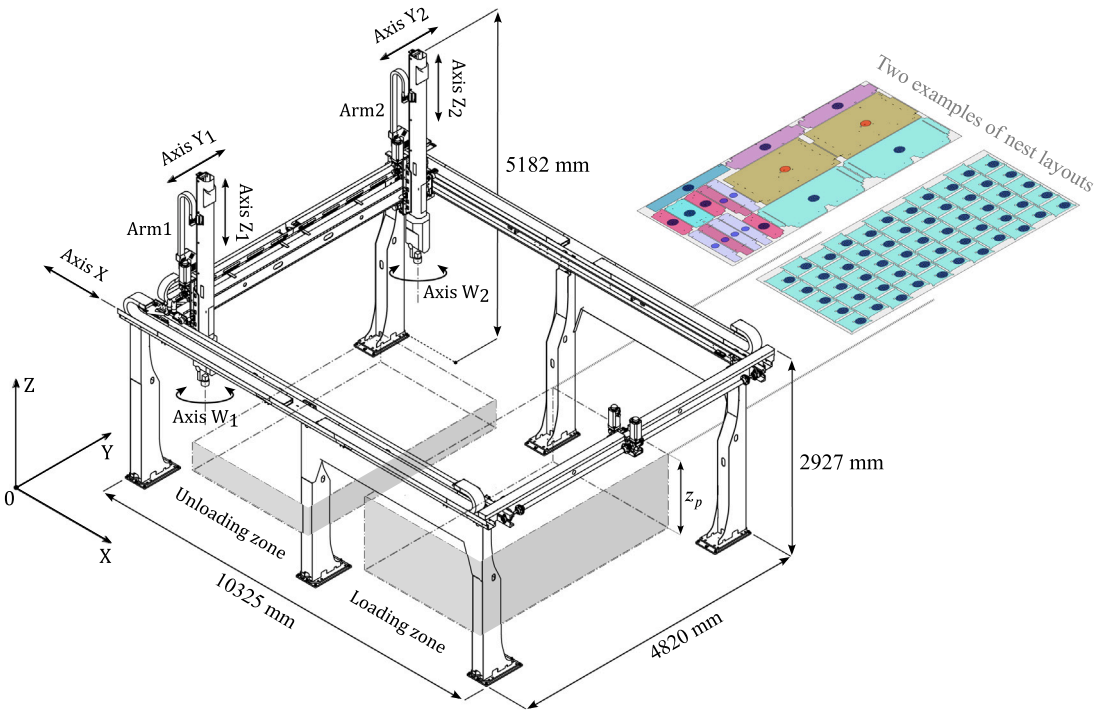


Fig. 1. Sorting system for laser machine (LST) with dual-arm Cartesian kinematics.

Multiple arm systems in nested part sorting tasks offer several advantages. 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 the case of multiple loadings, 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 [18]), resulting in shorter sorting times. In the test machine used in this study, the end-effector consists of a multi-tool head 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. For further details on this technology, please refer to [22].

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.

3. Path planning approach

3.1. Splines in B-form

Since the trajectory planning method draws on splines in their B-form, basic concepts about these functions are introduced here.

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:

$$\boldsymbol{r}(\tau) = \sum_{i=0}^{m} B_{i,d,\boldsymbol{u}}(\tau)\boldsymbol{b}_i \ , \quad u_s \le \tau \le u_e \tag{1}$$

with $\boldsymbol{b}_j = (b_{j,i})_{i=0}^{k}$ called control points. The basis function $B_{i,d,\boldsymbol{u}}(\tau)$ are defined by Cox–de Boor recursion formula [23]:

$$\begin{cases} B_{i,0,\boldsymbol{u}}(\tau) = \begin{cases} 1, & \text{if } u_i \le \tau < u_{i+1} \\ 0, & \text{otherwise} \end{cases} \\ B_{i,d,\boldsymbol{u}}(\tau) = \dfrac{\tau - u_i}{u_{i+d} - u_i} B_{i,d-1,\boldsymbol{u}} + \dfrac{u_{i+d+1} - \tau}{u_{i+d+1} - u_{i+1}} B_{i+1,d-1,\boldsymbol{u}} \end{cases} \tag{2}$$

based on a non-decreasing knot sequence $\boldsymbol{u} = (u_j)_{j=0}^{m+d+1}$.

In the proposed method the considered knot vectors are always non-periodic and non-uniform:

$$\boldsymbol{u} = \{\underbrace{u_s, \dots, u_s}_{d+1}, u_{d+1}, \dots, u_m, \underbrace{u_e, \dots, u_e}_{d+1}\} \tag{3}$$

It is useful to remind the expression of the derivative and the antiderivative of a spline given in B-form with respect to the parametric variable τ since they are frequently used in the remainder of the paper. The rth derivative of $\boldsymbol{r}(\tau)$ is computed by [23]:

$$D^{(r)}\boldsymbol{r}(\tau) = \sum_{i=0}^{m-r} B_{i,d-r,\boldsymbol{u}^{(r)}}(\tau)\boldsymbol{b}_i^{(r)} \tag{4}$$

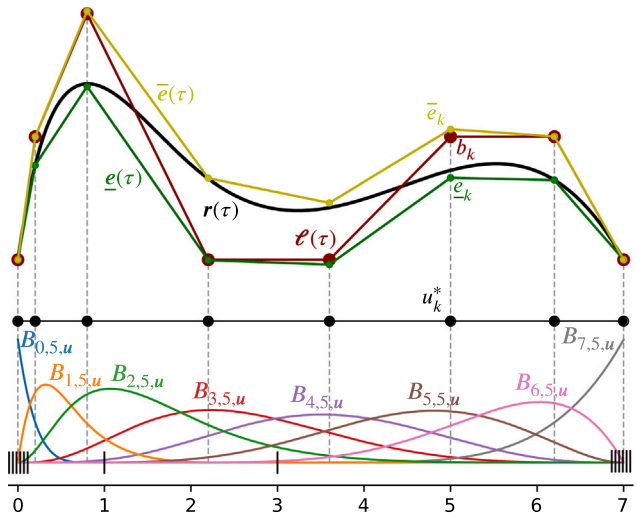


Fig. 2. A quintic spline and its envelope according to [25]. The knots are (0,0,0,0,0,0,1,3,7,7,7,7,7,7) and the control points are (0,1,2,0,0,1,1,0). This example is taken from [25].

with knots:

$$\boldsymbol{u}^{(r)} = \{\underbrace{u_s, \dots, u_s}_{d-r+1}, u_{d+1}, \dots, u_m, \underbrace{u_e, \dots, u_e}_{d-r+1}\} \tag{5}$$

where the control points are defined from:

$$\boldsymbol{b}_i^{(r)} = \frac{d-r+1}{u_{i+d+r} - u_{i+r}} \left(\boldsymbol{b}_{i+1}^{(r-1)} - \boldsymbol{b}_i^{(r-1)}\right), \quad \text{with } r > 0 \tag{6}$$

The antiderivative of $\boldsymbol{r}(\tau)$ is instead computed by [23]:

$$D^{(-1)}\boldsymbol{r}(\tau) = \sum_{i=0}^{m+1} B_{i,d+1,\boldsymbol{u}^{(-1)}}(\tau)\boldsymbol{b}_i^{(-1)} \tag{7}$$

with knots:

$$\boldsymbol{u}^{(-1)} = \{\underbrace{u_s, \dots, u_s}_{d+2}, u_{d+1}, \dots, u_m, \underbrace{u_e, \dots, u_e}_{d+2}\} \tag{8}$$

and control points:

$$\boldsymbol{b}_i^{(-1)} = \sum_{j=0}^{i} \boldsymbol{b}_j \frac{u_{j+d+1} - u_j}{d+1}, \quad \text{for } i = 0, \dots, m+1 \tag{9}$$

Then, (7) is iterated r times to obtain the rth antiderivative.

3.2. Channel problem definition

In this section, the problem of finding a smooth 1-D spline path between two polylines, $\underline{c}(\tau)$ and $\bar{c}(\tau)$, forming a channel is explained. For simplicity of notation, only within this section, the control points $\boldsymbol{b}_j$ in (1) are reduced to their scalar notation b_j. The method illustrated is based on that proposed by Lutterkort and Peters [24,25], which defines a thin envelope of the spline constrained within the channel. Then, the spline is uniquely determined by minimizing a cost function.

3.2.1. Tight linear envelopes for splines

The control polygon $\mathscr{C}(\tau) : \mathbb{R} \to \mathbb{R}$ 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 jth Greville abscissa, i.e. $\mathscr{C}(u_j^*) = b_j$. The weighted second differences $\Delta_2 b_j$ of the control points are defined as:

$$\Delta_2 b_j = \frac{b_{j+1} - b_j}{u_{j+1}^* - u_j^*} - \frac{b_j - b_{j-1}}{u_j^* - u_{j-1}^*} \tag{10}$$

Let $u_k^* \in [u_l, u_{l+1}]$, the contribution of the ith basis function to the distance between the spline and the control point b_j is captured by the function [25]:

$$\gamma_{ki}(u_k^*) = \begin{cases} \sum_{j=l-d}^{i} (u_i^* - u_j^*) B_{j,d,\boldsymbol{u}}(u_k^*), & i \le k \\ \sum_{j=i}^{l} (u_j^* - u_i^*) B_{j,d,\boldsymbol{u}}(u_k^*), & i \ge k+1 \end{cases} \tag{11}$$

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

$$\underline{e}_k = b_k + \sum_{i=l-d}^{l} \Delta_i^- \gamma_{ki}(u_k^*) \tag{12}$$

$$\bar{e}_k = b_k + \sum_{i=l-d}^{l} \Delta_i^+ \gamma_{ki}(u_k^*) \tag{13}$$

with

$$\Delta_i^- = \min(0, \Delta_2 b_j) \tag{14}$$

$$\Delta_i^+ = \max(0, \Delta_2 b_j) \tag{15}$$

An example of a thin envelope calculated with (12) and (13) is shown in Fig. 2.

3.2.2. Smooth paths in a polygonal channel

With the linear envelope described in Section 3.2.1, the channel problem can be defined as a quadratic programming (QP) optimization subject to linear constraints. The curvature is minimized considering the cost function:

$$\text{minimize} \ \to f(\boldsymbol{b}) = \sum_{i=1}^{m-1} (\Delta_2 b_i)^2 \quad \text{with: } \boldsymbol{b} = (b_i)_{i=1}^{m-1} \tag{16}$$

which can be seen in standard form

$$\boldsymbol{f}(\boldsymbol{b}) = \boldsymbol{b}^T P \boldsymbol{b} \qquad \text{with: } P \in \mathbb{R}^{(m-1)\times(m-1)} \tag{17}$$

The cost function matrix P is symmetrical and penta-diagonal in form (for proof see Appendix A):

$$P = \begin{bmatrix} C_1 & -D_1 & E_1 & & & & & 0 \\ -B_2 & C_2 & -D_2 & E_2 & & & & \\ A_3 & -B_3 & C_3 & -D_3 & E_3 & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & A_{m-3} & -B_{m-3} & C_{m-3} & -D_{m-3} & E_{m-3} \\ & & & & A_{m-2} & -B_{m-2} & C_{m-2} & -D_{m-2} \\ 0 & & & & & A_{m-1} & -B_{m-1} & C_{m-1} \end{bmatrix} \tag{18}$$

with ith coefficients, $i = 2, \dots, m-2$, defined by equations:

$$A_i = \frac{\Delta u_{i-1}^* \Delta u_i^*}{K_{i-1}^2} > 0 \tag{19}$$

$$B_i = \frac{\Delta u_{i-1}^* (\Delta u_{i-1}^* + \Delta u_i^*)}{K_{i-1}^2} + \frac{(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^*}{K_i^2} > 0 \tag{20}$$

$$C_i = \frac{\Delta u_{i-1}^{*2}}{K_{i-1}^2} + \frac{(\Delta u_i^* + \Delta u_{i+1}^*)^2}{K_i^2} + \frac{\Delta u_{i+2}^{*2}}{K_{i+1}^2} > 0 \tag{21}$$

$$D_i = \frac{\Delta u_i^* (\Delta u_i^* + \Delta u_{i+1}^*)}{K_i^2} + \frac{(\Delta u_{i+1}^* + \Delta u_{i+2}^*)\Delta u_{i+2}^*}{K_{i+1}^2} > 0 \tag{22}$$

$$E_i = \frac{\Delta u_{i+1}^* \Delta u_{i+2}^*}{K_{i+1}^2} > 0 \tag{23}$$

where:

$$\Delta u_i^* = u_i^* - u_{i-1}^* > 0 \tag{24}$$

$$K_i = \Delta u_{i+1}^* \cdot \Delta u_i^* > 0 \tag{25}$$

For $i = 1, m-1$ instead the coefficients are:

$$C_1 = \frac{(\Delta u_1^* + \Delta u_2^*)^2}{K_1^2} + \frac{\Delta u_3^{*2}}{K_2^2} > 0 \tag{26}$$

$$D_1 = \frac{\Delta u_1^* (\Delta u_1^* + \Delta u_2^*)}{K_1^2} + \frac{(\Delta u_2^* + \Delta u_3^*)\Delta u_3^*}{K_2^2} > 0 \tag{27}$$

$$E_1 = \frac{\Delta u_2^* \Delta u_3^*}{K_2^2} > 0 \tag{28}$$

$$A_{m-1} = \frac{\Delta u^*_{m-2}\Delta u^*_{m-1}}{K^2_{m-2}} > 0 \tag{29}$$

$$B_{m-1} = \frac{\Delta u^*_{m-2}(\Delta u^*_{m-2}+\Delta u^*_{m-1})}{K^2_{m-2}} + \frac{(\Delta u^*_{m-1}+\Delta u^*_m)\Delta u^*_m}{K^2_{m-1}} > 0 \tag{30}$$

$$C_{m-1} = \frac{\Delta u^{*2}_{m-2}}{K^2_{m-2}} + \frac{(\Delta u^*_{m-1}+\Delta u^*_m)^2}{K^2_{m-1}} > 0 \tag{31}$$

Furthermore, since P is derived from the cost function (16) expressed as a sum of squares, it is easy to verify that it is positive semi-definite.

The optimization problem of the objective function (16) is constrained by the inequality relations that ensure $\underline{c}(\tau) \le \underline{e}(\tau) \le \boldsymbol{r}(\tau) \le \overline{e}(\tau) \le \overline{c}(\tau)$:

$$\underline{c}(u^*_k) \le \underline{e}(u^*_k) = b_k + \textstyle\sum_{i=l-d}^{l} \Delta_i^- \gamma_{ki}(u^*_k) \quad k = 1,\dots,m-1 \tag{32}$$

$$\overline{c}(u^*_k) \ge \overline{e}(u^*_k) = b_k + \textstyle\sum_{i=l-d}^{l} \Delta_i^+ \gamma_{ki}(u^*_k) \quad k = 1,\dots,m-1 \tag{33}$$

$$\begin{aligned}\underline{c}(x_j) \le \mathcal{L}\big(x_j \mid \underline{e}(u^*_k), \underline{e}_{k+1}(u^*_{k+1})\big) = \\ w_1 b_k + w_2 b_{k+1} + w_1 \textstyle\sum_{i=l_k-d}^{l_k} \Delta_i^- \gamma_{ki}(u^*_k) + \\ w_2 \textstyle\sum_{i=l_{k+1}-d}^{l_{k+1}} \Delta_i^- \gamma_{k+1,i}(u^*_{k+1}) \qquad \forall j \in S^-\end{aligned} \tag{34}$$

$$\begin{aligned}\overline{c}(x_j) \ge \mathcal{L}\big(x_j \mid \overline{e}(u^*_k), \overline{e}_{k+1}(u^*_{k+1})\big) = \\ w_1 b_k + w_2 b_{k+1} + w_1 \textstyle\sum_{i=l_k-d}^{l_k} \Delta_i^+ \gamma_{ki}(u^*_k) + \\ w_2 \textstyle\sum_{i=l_{k+1}-d}^{l_{k+1}} \Delta_i^+ \gamma_{k+1,i}(u^*_{k+1}) \qquad \forall j \in S^+\end{aligned} \tag{35}$$

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

$$w_1 = \frac{u^*_{k+1}-x_j}{u^*_{k+1}-u^*_k}, \qquad w_2 = \frac{x_j-u^*_k}{u^*_{k+1}-u^*_k} \tag{36}$$

To eliminate the non-linearity resulting from relations (14) and (15), the $2m-2$ slack variables Δ_i^-, Δ_i^+ with $i = 1,\dots,m-1$ are included, in addition to the inequality constraints:

$$\Delta_i^- - \Delta_2 b_i \le 0 \qquad i = 1,\dots,m-1 \tag{37}$$

$$\Delta_i^+ - \Delta_2 b_i \ge 0 \qquad i = 1,\dots,m-1 \tag{38}$$

and boundaries:

$$\Delta_i^- \le 0 \qquad i = 1,\dots,m-1 \tag{39}$$

$$\Delta_i^+ \ge 0 \qquad i = 1,\dots,m-1 \tag{40}$$

With this approach, expressions (32) and (33) 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 [26]:

$$D_0^{(k)} = \textstyle\sum_{i=1}^{k} B^{(k)}_{i,d,\boldsymbol{u}}(u_i) b_i \qquad k = 1,\dots,j \tag{41}$$

$$D_m^{(k)} = \textstyle\sum_{i=1}^{k} B^{(k)}_{m-i,d,\boldsymbol{u}}(u_{m-i}) b_{m-i} \qquad k = 1,\dots,j \tag{42}$$

where:

$$B^{(k)}_{i,d,\boldsymbol{u}}(\tau) = d\left(\frac{B^{(k-1)}_{i,d-1,t}(\tau)}{u_{i+d}-u_i} - \frac{B^{(k-1)}_{i+1,d-1,t}(\tau)}{u_{i+d+1}-u_{i+1}}\right) \tag{43}$$

Note that (41) and (42) impose conditions on the channel composed of the $\underline{c}(\tau)$ and $\overline{c}(\tau)$ polylines, which must allow the control points of $\boldsymbol{r}(\tau)$ to be positioned correctly. An example clarifying this statement is reported in Section 3.3.4.

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

$$\text{minimize} \qquad \boldsymbol{x}^T \overline{P} \boldsymbol{x} \tag{44}$$

$$\text{subject to} \qquad \boldsymbol{l} \le A\boldsymbol{x} \le \boldsymbol{u} \tag{45}$$

where

$$\boldsymbol{x} = [b_1,\dots,b_{m-1},\Delta_1^-,\dots,\Delta_{m-1}^-,\Delta_1^+,\dots,\Delta_{m-1}^+]^T \tag{46}$$

$$\overline{P} = \begin{bmatrix} P & 0 \\ 0 & 0 \end{bmatrix} \begin{matrix} m-1 \\ 2m-2 \end{matrix} \qquad \begin{matrix} m-1 & 2m-2 \end{matrix} \tag{47}$$

$$A = \begin{bmatrix} I & \Gamma & 0 \\ I & 0 & \Gamma \\ W^- & S^- & 0 \\ W^+ & 0 & S^+ \\ -D & I & 0 \\ -D & 0 & I \\ 0 & I & 0 \\ 0 & 0 & I \\ B & 0 & 0 \end{bmatrix} \begin{matrix} m-1 & \triangleright \text{ Eq. (32)} \\ m-1 & \triangleright \text{ Eq. (33)} \\ \#S^- & \triangleright \text{ Eq. (34)} \\ \#S^+ & \triangleright \text{ Eq. (35)} \\ m-1 & \triangleright \text{ Eq. (37)} \\ m-1 & \triangleright \text{ Eq. (38)} \\ m-1 & \triangleright \text{ Eq. (39)} \\ m-1 & \triangleright \text{ Eq. (40)} \\ 2j & \triangleright \text{ Eqs. (41), (42)} \end{matrix} \qquad \begin{matrix} m-1 & m-1 & m-1 \end{matrix} \tag{48}$$

$$\begin{aligned}\boldsymbol{l} = [&\underline{c}(u^*_1),\dots,\underline{c}(u^*_{m-1}), \underbrace{-\infty,\dots,-\infty}_{m-1}, \underline{c}(x_j)\ \forall j \in S^-, \\ &\underbrace{-\infty,\dots,-\infty}_{\#S^+}, \underbrace{-\infty,\dots,-\infty}_{m-1}, \underbrace{0,\dots,0}_{m-1}, \\ &\underbrace{-\infty,\dots,-\infty}_{m-1}, \underbrace{0,\dots,0}_{m-1}, \underbrace{D_0^{(1)},\dots,D_0^{(j)}}_{j}, \underbrace{D_m^{(j)},\dots,D_m^{(1)}}_{j}]^T\end{aligned} \tag{49}$$

$$\begin{aligned}\boldsymbol{u} = [&\underbrace{\infty,\dots,\infty}_{m-1}, \overline{c}(u^*_1),\dots,\overline{c}(u^*_{m-1}), \underbrace{\infty,\dots,\infty}_{\#S^-}, \overline{c}(x_j)\ \forall j \in S^+, \\ &\underbrace{0,\dots,0}_{m-1}, \underbrace{\infty,\dots,\infty}_{m-1}, \underbrace{0,\dots,0}_{m-1}, \\ &\underbrace{\infty,\dots,\infty}_{m-1}, \underbrace{D_0^{(1)},\dots,D_0^{(j)}}_{j}, \underbrace{D_m^{(j)},\dots,D_m^{(1)}}_{j}]^T\end{aligned} \tag{50}$$

with I the identity matrix, Γ a sparse matrix with entries $[\Gamma]_{ij} = \gamma_{ij}(u^*_i)$, with $i = 1,\dots,m-1$, $j = l-d,\dots,l$, l the knot span of u^*_i, B is defined by the constraints (41) and (42):

$$B = \begin{bmatrix} B_1^{(1)}(u_1) & & & & & 0 \\ \vdots & \ddots & & & & \\ B_1^{(k)}(u_1) & \dots & B_k^{(k)}(u_k) & & & \\ & & & B^{(k)}_{m-k}(u_{m-k}) & \dots & B^{(k)}_{m-1}(u_{m-1}) \\ & & & & \ddots & \vdots \\ 0 & & & & & B^{(1)}_{m-1}(u_{m-1}) \end{bmatrix} \tag{51}$$

and D the following symmetric tridiagonal matrix:

$$D = \begin{bmatrix} -\frac{(\Delta u^*_1+\Delta u^*_2)}{K_1} & \frac{\Delta u^*_1}{K_1} & & & 0 \\ \frac{\Delta u^*_2}{K_2} & -\frac{(\Delta u^*_2+\Delta u^*_3)}{K_2} & \frac{\Delta u^*_2}{K_2} & & \\ & & \ddots & & \\ & & \frac{\Delta u^*_{m-1}}{K_{m-2}} & -\frac{(\Delta u^*_{m-2}+\Delta u^*_{m-1})}{K_{m-2}} & \frac{\Delta u^*_{m-2}}{K_{m-2}} \\ 0 & & & \frac{\Delta u^*_m}{K_{m-1}} & -\frac{(\Delta u^*_{m-1}+\Delta u^*_m)}{K_{m-1}} \end{bmatrix} \tag{52}$$

Finally, W and S are sparse matrices dependent on the specific channel, and related to Eqs. (34) and (35).

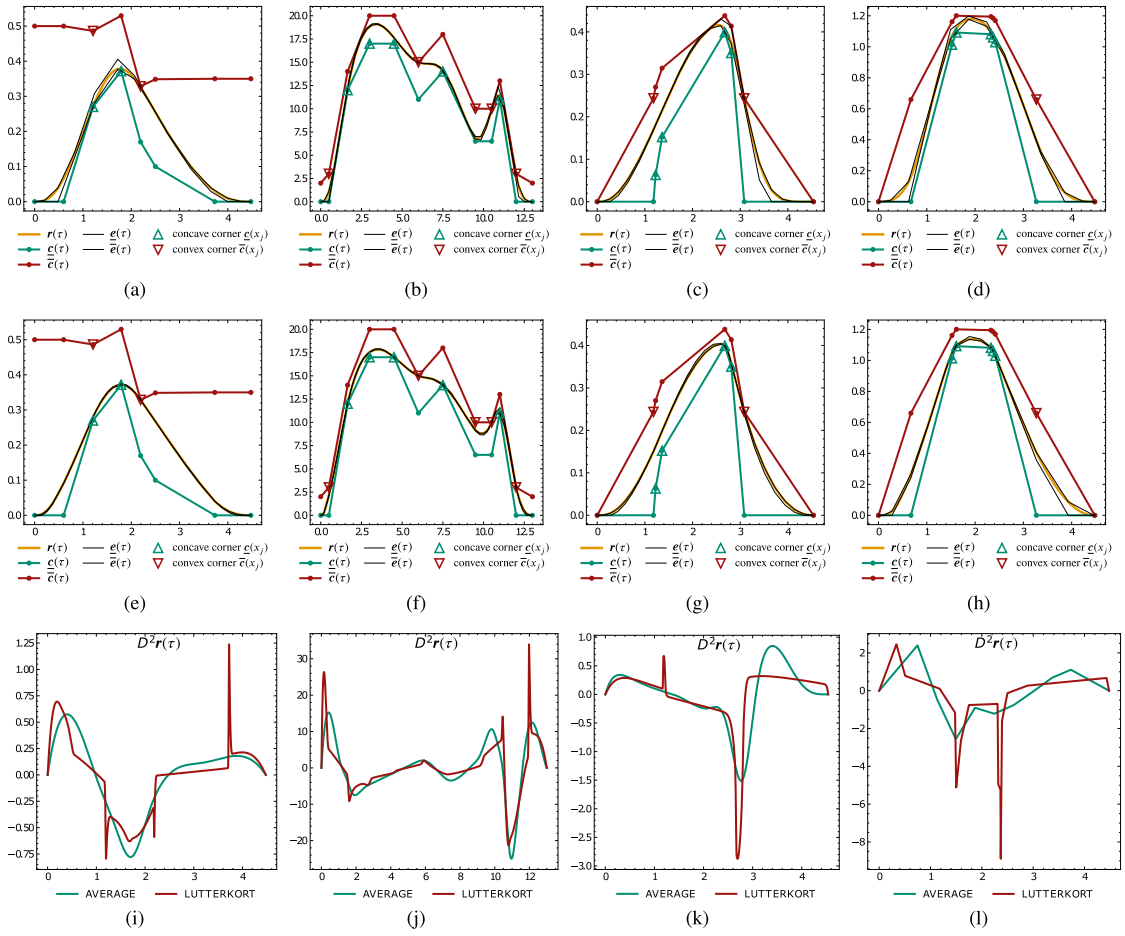


Fig. 3. Channel problem solutions for four input polygons computed with OSQP solver [27]. (a) to (d) solutions with initial knot sequence computed with (53). (e) to (h) solutions with initial knot sequence computed with Lutterkort algorithm [24]. (i) to (l) Comparison of the second derivative of the QP minimization problem spline solution with the two initial knot sequences.

Table 1
Benchmark data from 3. Performance evaluated on Intel® Core™ i7-10750H CPU, 2.60 GHz.

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

[a] Relative to the last iteration.

Note that since P is positive semi-definite, $\overline{P}$ is positive semi-definite, and it follows that the optimization problem is convex.

3.2.3. Initial knots sequence and iterative refinement

The problem defined by Eqs. (44) and (45) 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. Lutterkort and Peters [24] propose an initial knot sequence such that the corresponding sequence of Greville abscissae includes the breakpoints x_j of the input polylines $\underline{c}(\tau)$ and $\overline{c}(\tau)$. 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, Eqs. (34) and (35) are thus included into (32) and (33) 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:

$$\begin{aligned} &u_0 = \cdots = u_d = 0\,, \qquad u_{m+1} = \cdots = u_{m+d+1} = \overline{u}_m, \\ &t_{j+d} = \frac{1}{\max(d,1)} \sum_{i=j}^{j+d-1} \left[\frac{i}{m} \overline{u}_m \right], \qquad j = 1 \ldots, m-d. \end{aligned} \tag{53}$$

Then, the number of knots $m + d + 1$ is iteratively increased until the method converges. Note that, because of constraints (34) and (35), the

minimum number of knots used to start the iteration must satisfy the following formulations:

$$u_1^* \leq \min(\{x_j\}) \quad \wedge \quad u_{m-1}^* \geq \max(\{x_j\}) \quad \forall j \in (S^- \cup S^+) \tag{54}$$

Figs. 3(a) to 3(d) 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:

$$\begin{aligned} D_0^{(1)} &= D_0^{(2)} = 0 \\ D_m^{(1)} &= D_m^{(2)} = 0 \end{aligned} \tag{55}$$

Solutions are computed by an operator splitting solver for convex quadratic programs (OSQP, [27]), 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 [28]. 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 [29], and corresponding to the first channel shown in Fig. 3(a), is provided in the supplementary material attached to the paper.

Table 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. Figs. 3(e) to 3(h) show the solution of the same channel problems solved with a knot sequence calculated using the method suggested by Lutterkort and Peters [24]. 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 (Figs. 3(b) and 3(f)), the method proposed by the authors requires a total of 21 iterations, entailing a higher computational cost than the method detailed in [24]. 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 (Figs. 3(a) and 3(e)), and few or no iterations are usually required to ensure that the problem is solved. Finally, Figs. 3(i) to 3(l) 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 (53) 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.

3.3. Collision-free path planning strategy

This section explains the path planning strategy adopted. First, the geometric smoothed path planning problem of a single end-effector is addressed. The methodology is then extended to the second arm of the robot while maintaining the rigid connection constraint along the X-axis. Finally, the approach to prevent self-collisions of the robot is detailed.

3.3.1. 4-D.o.f. Cartesian robot path planning

In this section the path planning of a single end-effector not constrained on any axis with other heads is presented. This case does not correspond to the studied application but outlines part of the motion plan method that will be extended later for multiple constrained heads.

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. 4, we refer only to arm 1 of the dual-arm robot represented in Fig. 1. The $X_{H1}Y_{H1}Z_{H1}$-frame shown in the figure defines the coordinate

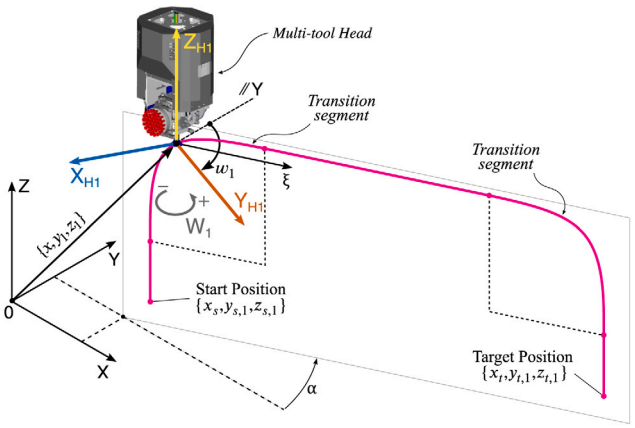


Fig. 4. 2-D path shape on the motion plane $Z_{H1} - \xi$.

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 $Z_{H1} - \xi$, 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 in the form of G01 polylines. Some recent work on this subject can be found in [30–32]. For our application, of particular interest is the study conducted by Sencer and Shamoto [33], where a corner smoothing algorithm using quintic Bézier curves is proposed to ensure both G^2-continuity of the tool-path and minimized curvature. This parametrization is reported in Fig. 5(a). 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:

$$\|\boldsymbol{P}_j - \boldsymbol{P}_{j-1}\| = c \ , \quad \text{with: } j = 1, 2, 4, 5 \tag{56}$$

$$\|\boldsymbol{P}_{trans} - \boldsymbol{P}_2\| = \|\boldsymbol{P}_3 - \boldsymbol{P}_{trans}\| = d \tag{57}$$

with $\boldsymbol{P}_{trans}$ the corner of the corresponding rectangular pick-and-place trajectory. Sencer and Shamoto [33] provide an optimized formulation of the ratio $n = c/d$ as a function of the transition angle θ (in this case always equal to $\pi/2$) to obtain the minimum transition curvature:

$$n(\pi/2) = (\pi/2)^{0.9927}/2.0769 \simeq 0.7538 \tag{58}$$

Condition (58) maximizes the total cornering velocity while respecting the acceleration limits of the actuators. 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 Figs. 1 and 5(b), 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:

$$L_{T,max} = z_{tr} - z_p \tag{59}$$

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

$$\boldsymbol{p}_s = \{x_s, y_s, z_s, w_s\} \tag{60}$$

$$\boldsymbol{p}_t = \{x_t, y_t, z_t, w_t\} \tag{61}$$

we can define:

$$\Delta X = x_t - x_s \tag{62}$$

$$\Delta Y = y_t - y_s \tag{63}$$

$$\Delta Z_s = z_{tr} - z_s \tag{64}$$

$$\Delta Z_t = z_t - z_{tr} \tag{65}$$

$$\Delta W = w_t - w_s \tag{66}$$

The transition length L_T is calculated as:

$$L_T = \min\left(L_{T,max}, \frac{\sqrt{\Delta X^2 + \Delta Y^2}}{2}\right) \tag{67}$$

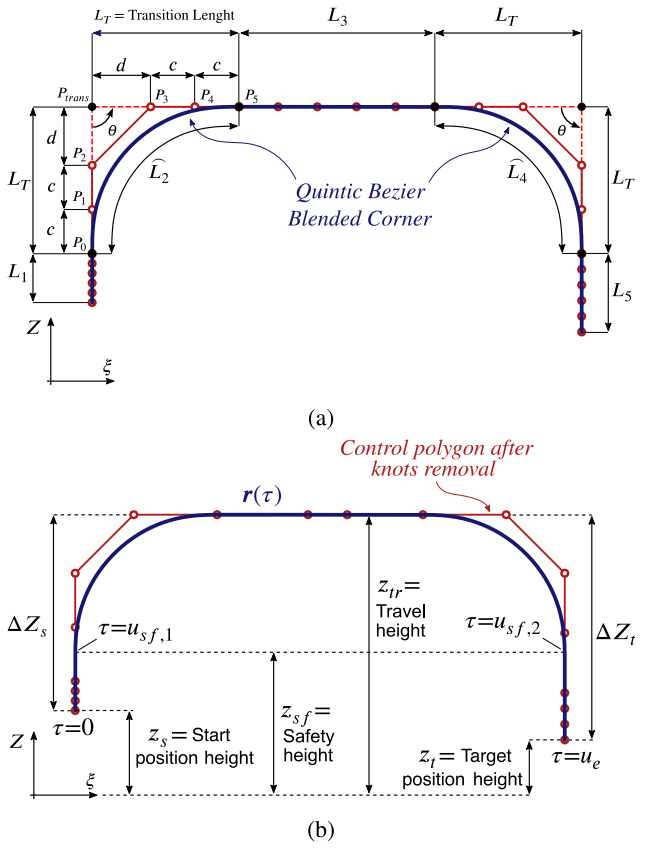


Fig. 5. Five-segment 2-D geometric path. (a) Geometric path parametrization. (b) Reference heights. $u_{sf,1}$ and $u_{sf,2}$, if they exist, are the values of the parameter τ corresponding to the intersections of the geometric path with the safety height z_{sf}.

The proposed approach involves two-step motion planning; first a 3-D path is planned considering 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 (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:

$$D\boldsymbol{r}_1(u_{1,e}) = D\boldsymbol{r}_2(u_{2,s}) \tag{68}$$

$$D^{(2)}\boldsymbol{r}_1(u_{1,e}) = D^{(2)}\boldsymbol{r}_2(u_{2,s}) \tag{69}$$

$$D\boldsymbol{r}_2(u_{2,e}) = D\boldsymbol{r}_3(u_{3,s}) \tag{70}$$

$$D^{(2)}\boldsymbol{r}_2(u_{2,e}) = D^{(2)}\boldsymbol{r}_3(u_{3,s}) \tag{71}$$

with $\boldsymbol{r}_i$ the ith segment having knots:

$$\boldsymbol{u}_i = \{\underbrace{u_{i,s}, \dots, u_{i,s}}_{p+1}, \underbrace{u_{i,e}, \dots, u_{i,e}}_{p+1}\} \tag{72}$$

By setting $u_{2,s} = 0$ and $u_{2,e} = 1$, [33]:

$$D\boldsymbol{r}_2(0) = 5(\boldsymbol{P}_1 - \boldsymbol{P}_0) \tag{73}$$

$$D\boldsymbol{r}_2(1) = 5(\boldsymbol{P}_5 - \boldsymbol{P}_4) \tag{74}$$

$$D^{(2)}\boldsymbol{r}_2(0) = 20(\boldsymbol{P}_2 - 2\boldsymbol{P}_1 + \boldsymbol{P}_0) \tag{75}$$

$$D^{(2)}\boldsymbol{r}_2(1) = 20(\boldsymbol{P}_5 - 2\boldsymbol{P}_4 + \boldsymbol{P}_3) \tag{76}$$

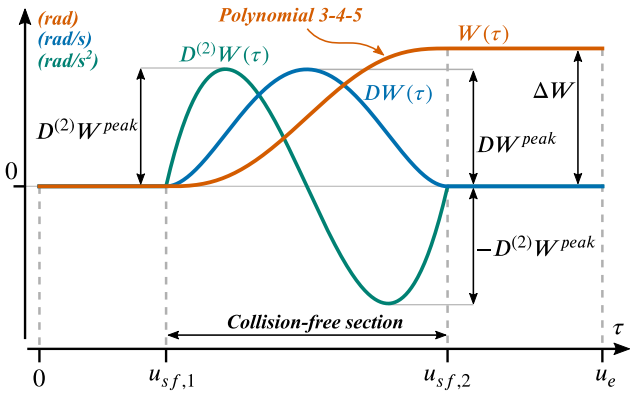


Fig. 6. $W(\tau)$ motion profile. In the collision-free section the motion is described from polynomial 3-4-5 formulation, according to (85).

Passing to scalar notation:

$$\|D\boldsymbol{r}_2(0)\| = \|D\boldsymbol{r}_2(1)\| = 5c \tag{77}$$

$$\|D^{(2)}\boldsymbol{r}_2(0)\| = \|D^{(2)}\boldsymbol{r}_2(1)\| = 0 \tag{78}$$

Assuming a second transition segment symmetrical to the first, (77) and (78) 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:

$$u_{1,s} = 0, \qquad u_{1,e} = L_1/(5c) \tag{79}$$

$$u_{3,s} = 0, \qquad u_{3,e} = L_3/(5c) \tag{80}$$

$$u_{5,s} = 0, \qquad u_{5,e} = L_5/(5c) \tag{81}$$

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

1. Elevate the degree of segments 1,3, and 5 to reach the same degree as the transition segments [13,34].
2. Concatenate the Bézier curves by repeating for the number of segments the joining operation explained below. Let $\boldsymbol{r}_l$ and $\boldsymbol{r}_r$ be the segment to the left of the welding point of the curves (i.e. the result of the previous concatenations) and the segment to its right, respectively. For example, at the first step, $\boldsymbol{r}_l$ is $\boldsymbol{r}_1$, while at the second step it concurs with the result of the concatenation between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, and so on. The spline obtained from the concatenation of $\boldsymbol{r}_l$ and $\boldsymbol{r}_r$ of the same degree d is defined by the knots:

$$\begin{aligned}\boldsymbol{u} = \{&\underbrace{u_{l,s}, \dots, u_{l,s}}_{d+1}, u_{l,d+1}, \dots, u_{1,m_l}, \underbrace{u_{l,e}, \dots, u_{l,e}}_{d+1}, \dots,\\ &(u_{r,d+1} + \Delta u), \dots, (u_{r,m_2} + \Delta u),\\ &\underbrace{(u_{r,e} + \Delta u), \dots, (u_{r,e} + \Delta u)}_{d+1}\}\end{aligned} \tag{82}$$

   with $\Delta u = u_{l,e} - u_{r,s}$, and control points:

$$\boldsymbol{b} = \{\boldsymbol{b}_{l,0}, \dots, \boldsymbol{b}_{l,m_l}, \boldsymbol{b}_{r,0}, \dots, \boldsymbol{b}_{r,m_r}\} \tag{83}$$

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 Tiller [35] 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. 5 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 is 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. 5(b), $u_{sf,1}$ 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 safety height. In other words, the interval $[u_{sf,1}, u_{sf,2}]$ is defined such that:

$$\text{proj}(\boldsymbol{r}(\tau)|_{[u_{sf,1},u_{sf,2}]}, Z) \geq z_{sf} \tag{84}$$

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 Fig. 6, 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 5th-degree polynomial:

$$W(\tau)|_{[u_{sf,1},u_{sf,2}]} = \Delta W \left[10\left(\frac{\tau}{\Delta u}\right)^3 - 15\left(\frac{\tau}{\Delta u}\right)^4 + 6\left(\frac{\tau}{\Delta u}\right)^5\right] \tag{85}$$

with $\Delta u = u_{sf,2} - u_{sf,1}$. Expression (85) defines a polynomial in its p-form:

$$\boldsymbol{p}(x) = \sum_{i=0}^{d} x^i a_i = a_0 + x a_1 + x^2 a_2 + \cdots + x^d a_d \tag{86}$$

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

$$\boldsymbol{u} = \{\underbrace{u_{sf,1}, \ldots, u_{sf,1}}_{d+1}, \underbrace{u_{sf,2}, \ldots, u_{sf,2}}_{d+1}\} \tag{87}$$

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

$$M\boldsymbol{b} = \boldsymbol{a} \tag{88}$$

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

$$[M]_{i,j} = \binom{d}{i}\binom{i}{j}(-1)^{i-j} \quad , \quad \text{with} \begin{cases} i = 0, \ldots, d \\ j = 0, \ldots, i \end{cases} \tag{89}$$

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 [36]. 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 parametrization of the function describing the geometric path. A common solution is to adopt the arc length parametrization, then parametrize $\bar{\boldsymbol{r}}(\tau)$ 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 parametrization, 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 parametrization 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:

$$ds = \|\boldsymbol{K}_r \odot D\bar{\boldsymbol{r}}(\tau)\| d\tau \tag{90}$$

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:

$$s_{max} = \sum_{i=0}^{m-1} \sqrt{K_{r,x}\Delta x_i^2 + K_{r,y}\Delta y_i^2 + K_{r,z}\Delta z_i^2 + K_{r,w}\Delta w_i^2} \tag{91}$$

with

$$\Delta x_i = x_{i+1} - x_i \tag{92}$$

$$\Delta y_i = y_{i+1} - y_i \tag{93}$$

$$\Delta z_i = z_{i+1} - z_i \tag{94}$$

$$\Delta w_i = w_{i+1} - w_i \tag{95}$$

The accuracy of the s_{max} estimated with (91) 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 (91), $\bar{\boldsymbol{r}}(\tau)$ can be parametrized on the curvilinear coordinate s by multiplying all knots $\bar{u}_i$ by $s_{max}/\bar{u}_e$.

3.3.2. 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.

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

$$\boldsymbol{p}_{s,1} = \{x_s, y_{s,1}, z_{s,1}, w_{s,1}\} \tag{96}$$

$$\boldsymbol{p}_{s,2} = \{x_s, y_{s,2}, z_{s,2}, w_{s,2}\} \tag{97}$$

$$\boldsymbol{p}_{t,1} = \{x_t, y_{t,1}, z_{t,1}, w_{t,1}\} \tag{98}$$

$$\boldsymbol{p}_{t,2} = \{x_t, y_{t,2}, z_{t,2}, w_{t,2}\} \tag{99}$$

may be defined:

$$\Delta X = x_t - x_s \tag{100}$$

$$\Delta Y_1 = y_{t,1} - y_{s,1} \tag{101}$$

$$\Delta Y_2 = y_{t,2} - y_{s,2} \tag{102}$$

$$\Delta Z_{s,1} = z_{tr} - z_{s,1} \tag{103}$$

$$\Delta Z_{t,1} = z_{t,1} - z_{tr} \tag{104}$$

$$\Delta Z_{s,2} = z_{tr} - z_{s,2} \tag{105}$$

$$\Delta Z_{t,2} = z_{t,2} - z_{tr} \tag{106}$$

$$\Delta W_1 = w_{t,1} - w_{s,1} \tag{107}$$

$$\Delta W_2 = w_{t,2} - w_{s,2} \tag{108}$$

The main idea for extending the procedure of Section 3.3.1 is to define a master path, described from the 5th-degree spline $\boldsymbol{r}_\phi(\tau) : \mathbb{R} \to \mathbb{R}^3$ 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:

$$x_{0,\phi} = y_{0,\phi} = z_{0,\phi} = 0 \tag{109}$$

which corresponds to define the spline relatively to the first control point. $\boldsymbol{r}_\phi(\tau)$ can be calculated as described in Section 3.3.1 using the geometric lengths L_i, with $i \in \{1, 3, 5\}$, obtained by solving the following system:

$$\begin{cases} L_1 + L_T = \max(\Delta Z_{s,1}, \Delta Z_{s,2}) & \text{(a)} \\ L_3 + 2 \cdot L_T = \max\left(\sqrt{\Delta X^2 + \Delta Y_1^2}, \sqrt{\Delta X^2 + \Delta Y_2^2}\right) & \text{(b)} \\ L_T + L_5 = \max(|\Delta Z_{t,1}|, |\Delta Z_{t,2}|) & \text{(c)} \end{cases} \tag{110}$$

with L_T defined by (67).

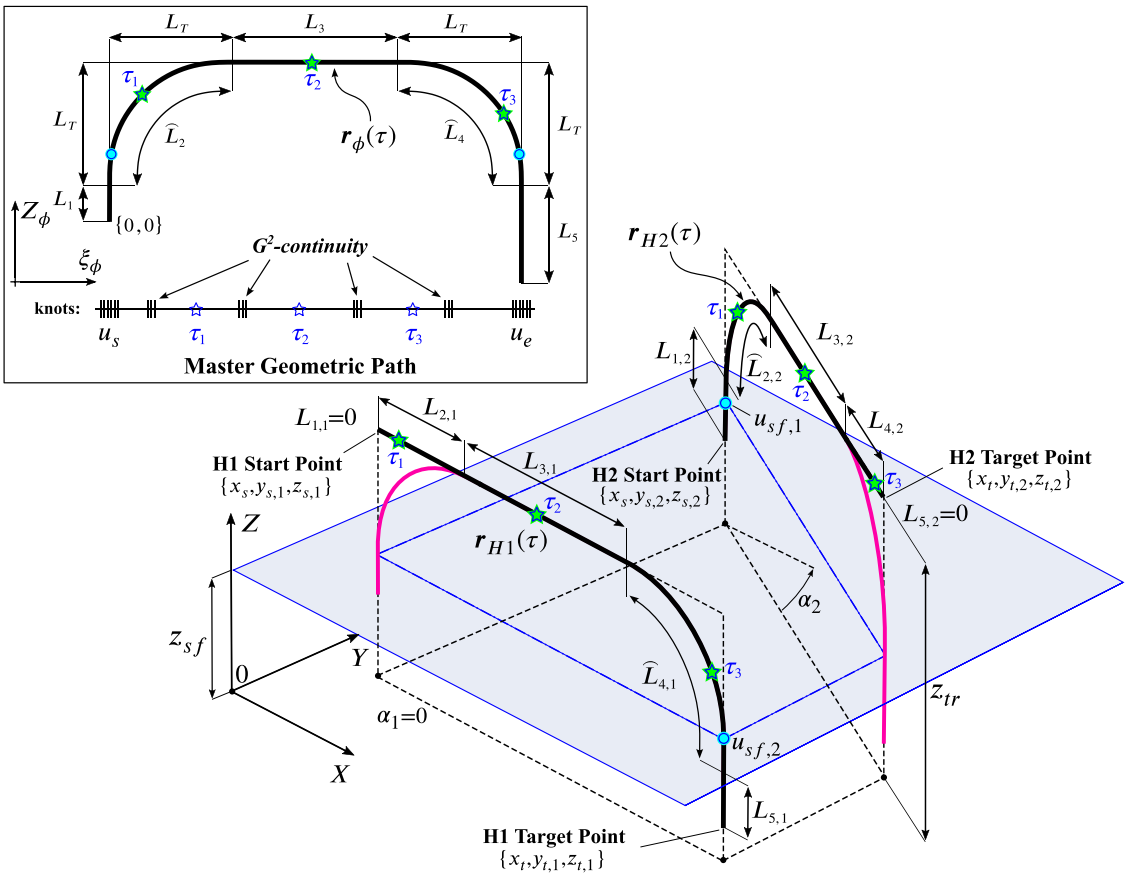


Fig. 7. Geometric path representation of the dual-arm Cartesian robot with 5th-degree spline. The circular markers correspond to the intersections of the path with the safety height calculated in accordance with (114). The star markers are examples of points calculated respectively on the master path $\boldsymbol{r}_\phi(\tau)$ and on $\boldsymbol{r}(\tau)$ for three values of the parametric variable τ.

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 Fig. 7), with the correct knots multiplicity for a spline of degree d the ith control point are uniquely determined through path scaling techniques from the relations:

$$x_i = x_{i,\phi} + x_s \tag{111}$$

$$y_{i,j} = y_{i,\phi}\frac{\Delta Y_j}{y_{m,\phi}} + y_{s,j} \tag{112}$$

$$z_{i,j} = \begin{cases} z_{i,\phi}\frac{\Delta Z_{s,j}}{(L_1+L_T)} + z_{s,j}, & \text{if } i < l \quad \text{(a)} \\ z_{tr} + \left(z_{i,\phi} - (L_1+L_T)\right)\frac{\Delta Z_{t,j}}{(L_T+L_5)}, & \text{if } i > r \quad \text{(b)} \\ z_{tr}, & \text{otherwise} \quad \text{(c)} \end{cases} \tag{113}$$

with $l = 2d-1$, $r = 4d-6$, $d = 5$ and $j = 1,2$. With reference to Fig. 7, 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 ith 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:

$$\begin{cases} \text{proj}(\boldsymbol{r}_{H1}(\tau)|_{[u_{sf,1},u_{sf,2}]},\, Z) \geq z_{sf} \\ \text{proj}(\boldsymbol{r}_{H2}(\tau)|_{[u_{sf,1},u_{sf,2}]},\, Z) \geq z_{sf} \end{cases} \tag{114}$$

The geometric path components $W_1(\tau)$ and $W_2(\tau)$ of the two end-effectors, with ith 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 (88) 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:

$$\Delta W_{max} = \max(|\Delta W_1|, |\Delta W_2|) \tag{115}$$

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

$$W_r = \min(|\Delta W_1|, |\Delta W_2|)/\Delta W_{max} \tag{116}$$

For example, if $\Delta W_1 > \Delta W_2$, $W_1(\tau)$ is computed by the routine of Section 3.3.1, while $W_2(\tau)$ is defined by the same degree and knots of $W_1(\tau)$, and the ith control point

$$w_{i,2} = \text{sgn}(\Delta W_2) w_{i,1} W_r \tag{117}$$

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 parametrization 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 transmission ratios for the Y, Z, and W of the two arms:

$$\Delta s^2_{i,\phi} = K_{r,x}\Delta x^2_{i,\phi} + K_{r,y}\Delta y^2_{i,\phi} + K_{r,z}\Delta z^2_{i,\phi} + K_{r,w}\max(\Delta w^2_{i,1}, \Delta w^2_{i,2}) \tag{118}$$

$$s_{max} = \sum_{i=0}^{m-1}\sqrt{\Delta s^2_{i,\phi}} \tag{119}$$

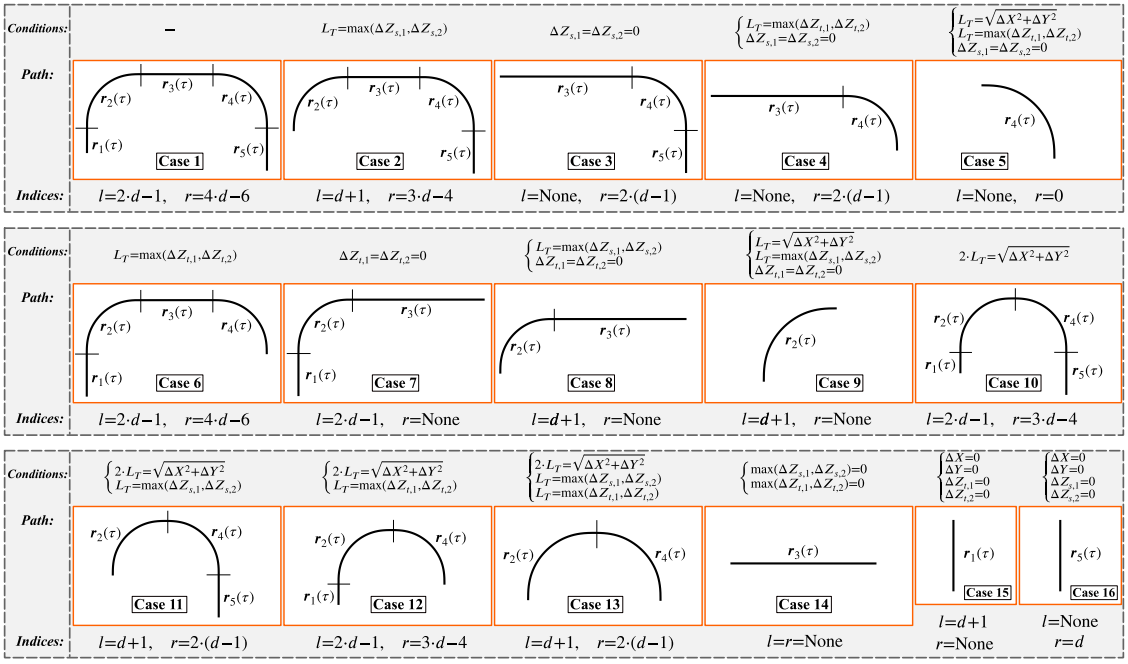


Fig. 8. Sub-cases of the five-segment path of Fig. 5 used to define the master path $\boldsymbol{r}_\phi(\tau)$. For each block; at the top the conditions defining the sub-case, in the middle the segments $\boldsymbol{r}_i(\tau)$ used to define the path, at the bottom the indices l and r, to be used in Eq. (113)a and Eq. (113)b respectively. If one of the indices l or r is set to None, the relative equation is not used in that subcase.

with

$$\Delta x_{i,\phi} = x_{i+1,\phi} - x_{i,\phi} \tag{120}$$

$$\Delta y_{i,\phi} = y_{i+1,\phi} - y_{i,\phi} \tag{121}$$

$$\Delta z_{i,\phi} = z_{i+1,\phi} - z_{i,\phi} \tag{122}$$

$$\Delta w_{i,j} = w_{i+1,j} - w_{i,j} \quad \text{with: } j = 1,2 \tag{123}$$

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 parametrized 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 Fig. 5(a), 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 (113)a and (113)b respectively. 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, Fig. 8 summarizes 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 (67) 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. An example of this case is included in the concatenation of movements that constitute the case study in Section 6.

3.3.3. 2-D collision detection solution

Before detailing the solution adopted by the collision avoidance algorithm, it is necessary to report some notions on the collision detection of dynamic elements in the working area. Fig. 9 shows a generic ith motion frame, projected onto the XY plane of the machine, where each end-effector is modelled with a regular octagon with edges of length l_e, and the loaded parts are modelled with a bounding rectangle circumscribing the external profile. The relative position of each end-effector to its loaded part (i.e. the rectangles) is uniquely defined by the particular head-part configuration in the loading task. For each point-to-point movement of the robot, a part may or may not be loaded at each end-effector. 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 constrained 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 [37], 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. It should be noted that the type of convex polygon used to circumscribe head and loaded parts is arbitrary and not limited to those adopted in this work.

In this application, we want to calculate the maximum Y-intersection between the two polygon groups, which we call κ_i, and their upper and lower overlap with respect to the working area for each frame. There are several efficient solutions in the literature for computing the intersection between convex polygons [38–41], so we do not discuss them further in this paper.

The two polygon groups are constrained to move within a working area defined geometrically by the maximum and minimum values allowed 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. This is justified because if the polygon of group 2

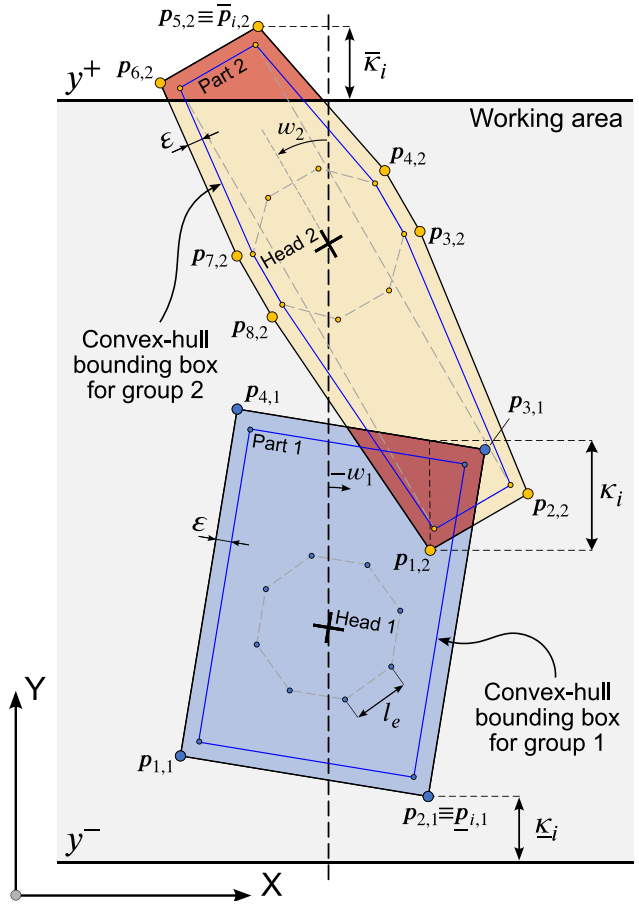


Fig. 9. Calculation of Y-intersections and bounding box grouping strategy.

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. A 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 upper and lower overlap $\overline{\kappa}_i$ and $\underline{\kappa}_i$ for the ith frame are computed with:

$$\overline{\kappa}_i = \overline{p}_{i,2,y} - y^+ \tag{124}$$

$$\underline{\kappa}_i = y^- - \underline{p}_{i,1,y} \tag{125}$$

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

Finally, the frame sampling strategy used to discretize the motion should take into account the point-to-point path length and the ε tolerance used in collision detection. In this way, an excessive number of frames can be avoided, which would increase the computation time, while ensuring a sufficient number of frames to avoid collisions. Details of the frame sampling strategy used in this work can be found in Appendix B.

3.3.4. Polygonal channel definition

Let $\bar{\boldsymbol{r}}(\tau) : \mathbb{R} \to \mathbb{R}^7$ be a 5th-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 3.3.2. 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 $\boldsymbol{\tau} = (\tau_i)_{i=0}^{(\mathrm{M})}$ computed as detailed in the previous section. This results in the definition of vectors:

- Overlap vector: $\boldsymbol{\kappa} = (\kappa_i)_{i=0}^{M}$ (126)
- Top Overrun vector: $\overline{\boldsymbol{\kappa}} = \{\max(\overline{\kappa}_i, 0),\ i \in [0, M]\}$ (127)
- Bottom Overrun vector: $\underline{\boldsymbol{\kappa}} = \{\max(\underline{\kappa}_i, 0),\ i \in [0, M]\}$ (128)

Algorithm 1 Upper Envelope Evaluation

Input:

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

Output:

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

Algorithm:

Define a function $\mathscr{E}(\tau)$ according to:

$$\mathscr{E}(\tau) := \mathcal{L}\big(\tau \mid \{\tau_0, \kappa_0\}, \{\tau_{\mathrm{M}}, \kappa_{\mathrm{M}}\}\big) \tag{129}$$

$\boldsymbol{\Delta} = \{\Delta_i\}$ with $\Delta_i = \kappa_i - \mathscr{E}(\tau_i)$
$i^* = \{i \mid i \in [0, \mathrm{M}] \ \wedge \ \Delta_i \geq \Delta_r,\ \forall \Delta_r \in \boldsymbol{\Delta}\}$
if $\Delta_{i^*} <$ Tol **then**
  return $\mathbf{I} = \{0, \mathrm{M}\}$
end if
Define a sequence $\mathbf{I}_l = \{0, \dots, i^*\}$ using Algorithm 1 with input $\boldsymbol{\Delta}_l = \{\Delta_0, \dots, \Delta_{i^*}\}$, $\boldsymbol{\tau}_l = \{\tau_0, \dots, \tau_{i^*}\}$ and Tol.
Define a sequence $\mathbf{I}_r = \{i^*, \dots, \tau_{\mathrm{M}}\}$ using Algorithm 1 with input $\boldsymbol{\Delta}_r = \{\Delta_{i^*}, \dots, \Delta_{\mathrm{M}}\}$, $\boldsymbol{\tau}_r = \{\tau_{i^*}, \dots, \tau_{\mathrm{M}}\}$ and Tol.
return $\mathbf{I} = \{0, \dots, i^*, \dots, \mathrm{M}\} = \mathbf{I}_l \cup \mathbf{I}_r$

This section proposes a post-processing method for vectors defined by (126), (127) or (128) using the algorithm illustrated in Section 3.2. 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 3.2 requires the definition of a channel in the form of a lower polyline $\underline{c}(\tau)$ and an upper polyline $\overline{c}(\tau)$. The proposed approach firstly calculates $\underline{c}(\tau)$ starting from $\boldsymbol{\kappa}$, after which $\bar{c}(\tau)$ 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 sequence of initial and final frames with no interpenetration between polygons. Therefore, the following subsets are defined:

$$\boldsymbol{\kappa}' = \{\kappa_l, \dots, \kappa_r\} \subseteq \boldsymbol{\kappa} \tag{130}$$

$$\boldsymbol{\tau}' = \{\tau_l, \dots, \tau_r\} \subseteq \boldsymbol{\tau} \tag{131}$$

with

$$l = \min(i \mid \kappa_i \neq 0, \kappa_i \in \boldsymbol{\kappa}) - 1 \tag{132}$$

$$r = \max(i \mid \kappa_i \neq 0, \kappa_i \in \boldsymbol{\kappa}) + 1 \tag{133}$$

Then, the upper envelope of $\boldsymbol{\kappa}'$ is computed with Algorithm 1.

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 3.2, 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 1 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 $\{\kappa_i, \tau_i\}$, $\forall i \in \mathbf{I}$. With reference to Fig. 10a and Fig. 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 (129), calculated in $\boldsymbol{\tau}$, from the $\boldsymbol{\kappa}$ vector. Recursion terminates on condition $\Delta_i \leq$ Tol, $\forall \Delta_i \in \boldsymbol{\Delta}$.

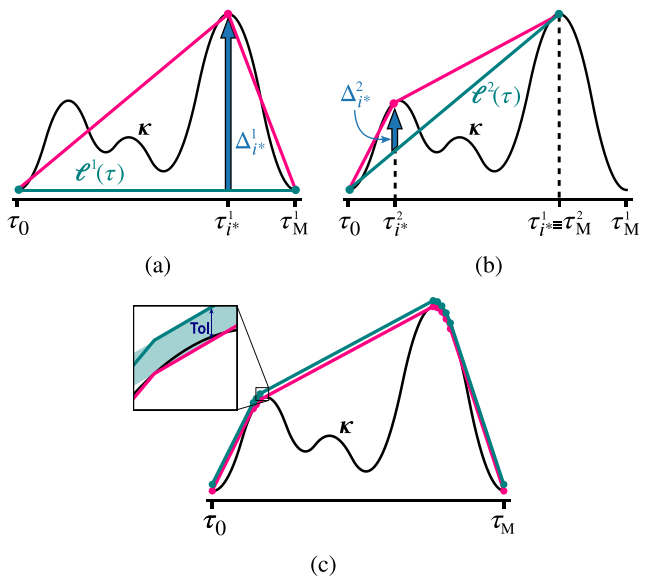


Fig. 10. Upper envelope evaluation algorithm. (a) 1st step of Algorithm 1. (b) 2nd step of Algorithm 1. (c) Tolerance parameter translation.

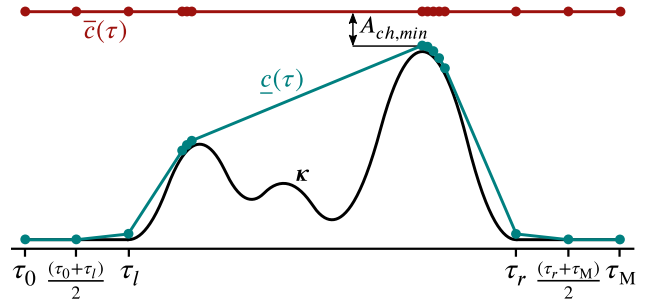


Fig. 11. Channel definition from overlap vector $\boldsymbol{\kappa}$, according to Eqs. (134) to (139).

Given the indices sequence $\mathbf{I} = (\mathrm{I}_j)_{j=0}^{n}$, the lower polyline $\underline{c}(\tau)$ is defined as follows (refer to Fig. 10c):

$$\underline{c}_0 = \underline{c}_1 = \underline{c}_{n+3} = \underline{c}_{n+4} = 0 \tag{134}$$

$$\underline{c}_{j+2} = \kappa'_{I_j} + \mathrm{Tol}, \quad j = 0, \dots, n \tag{135}$$

$$x_0 = \bar{u}_0, \quad x_{n+4} = \bar{u}_{m+6} \tag{136}$$

$$x_1 = (\bar{u}_0 + \tau_l)/2, \quad x_{n+3} = (\bar{u}_{m+6} + \tau_r)/2 \tag{137}$$

$$x_{j+2} = \tau'_{I_j}, \quad j = 0, \dots, n \tag{138}$$

$$\underline{c}(\tau) = \mathcal{L}\left(\tau | \{x_i, \underline{c}_i\}, \{x_{i+1}, \underline{c}_{i+1}\}\right) \text{ with } x_i \le \tau \le x_{i+1}, \quad \forall i \in [0, n+4] \tag{139}$$

Let $A_{ch,min}$ be the desired minimum channel amplitude the upper polyline $\overline{c}(\tau)$ adopted in this work is defined in the same form of (139) by the coefficients (see Fig. 11):

$$\overline{c}_j = \max(\underline{c}_i) + A_{ch,min}, \quad i = 0, \dots, n+4, \quad j = 0, \dots, n+4 \tag{140}$$

Note that for a constant upper polyline $\overline{c}(\tau)$, such as that defined by (140), the inequality constraints (33) and (35) reported in Section 3.2 could be greatly simplified. However, it is preferred to report the general discussion in case a different formulation from (140) would be adopted.

3.3.5. 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 Section 3.3.2, 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 3.2, with the channel defined according to 3.3.4. Depending on whether the vector of Y-intersections is defined by (126), (127) or (128), the 5th-degree 1-D spline, defined by (1), is named as follows:

- Overlap spline: $\boldsymbol{\sigma}(\tau)$ (141)
- Top Overrun spline: $\overline{\boldsymbol{\sigma}}(\tau)$ (142)
- Bottom Overrun spline: $\underline{\boldsymbol{\sigma}}(\tau)$ (143)

with control points

$$\boldsymbol{h} = (h_i)_{i=0}^{q} \quad \underline{\boldsymbol{h}} = (\underline{h}_i)_{i=0}^{\underline{q}} \quad \overline{\boldsymbol{h}} = (\overline{h}_i)_{i=0}^{\overline{q}} \tag{144}$$

and knots

$$\boldsymbol{k} = (k_i)_{i=0}^{q+6} \quad \underline{\boldsymbol{k}} = (\underline{k}_i)_{i=0}^{\underline{q}+6} \quad \overline{\boldsymbol{k}} = (\overline{k}_i)_{i=0}^{\overline{q}+6} \tag{145}$$

defined by the procedure reported in Section 3.2.3.

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 [36]:

$$\bar{y}_{i,1} \leftarrow \bar{y}_{i,1} - w_1 h_i \tag{146}$$

$$\bar{y}_{i,2} \leftarrow \bar{y}_{i,2} + w_2 h_i \tag{147}$$

Referring to Eqs. (124) and (125), be:

$$\Delta y^+ = \left\{ |\overline{\kappa}_i| \;\middle|\; \overline{\kappa}_i \le 0 \;\wedge\; \overline{\kappa}_i \ge \overline{\kappa}_j, \;\forall i,j \in [0,N] \right\} \tag{148}$$

$$\Delta y^- = \left\{ |\underline{\kappa}_i| \;\middle|\; \underline{\kappa}_i \le 0 \;\wedge\; \underline{\kappa}_i \ge \underline{\kappa}_j, \;\forall i,j \in [0,N] \right\} \tag{149}$$

$$\sigma_{max} = \max(\{\boldsymbol{\sigma}(\bar{u}_i) \mid \forall \bar{u}_i \in \bar{\boldsymbol{u}}\}) \tag{150}$$

$$w_1^* = \Delta y^- / \sigma_{max} \tag{151}$$

$$w_2^* = \Delta y^+ / \sigma_{max} \tag{152}$$

The weights w_1 and w_2 of Eqs. (146) and (147) can be calculated through:

$$\begin{cases} w_1 = w_2 = 0.5, & \text{if } w_1^* \ge 0.5 \;\wedge\; w_2^* \ge 0.5 \\ \begin{cases} w_1 = w_1^* \\ w_2 = (1 - w_1) \end{cases} & \text{if } (w_1^* < 0.5 \;\vee\; w_2^* < 0.5) \wedge w_1^* < w_2^* \\ \begin{cases} w_2 = w_2^* \\ w_1 = (1 - w_2) \end{cases} & \text{otherwise} \end{cases} \tag{153}$$

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 $\boldsymbol{\sigma}$:	Eq. (153)
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. 12 reports the pipeline of the proposed collision-free path planning strategy. First the geometric path is calculated using the strategy of Section 3.3.2 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 [18], 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 3.3.3 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

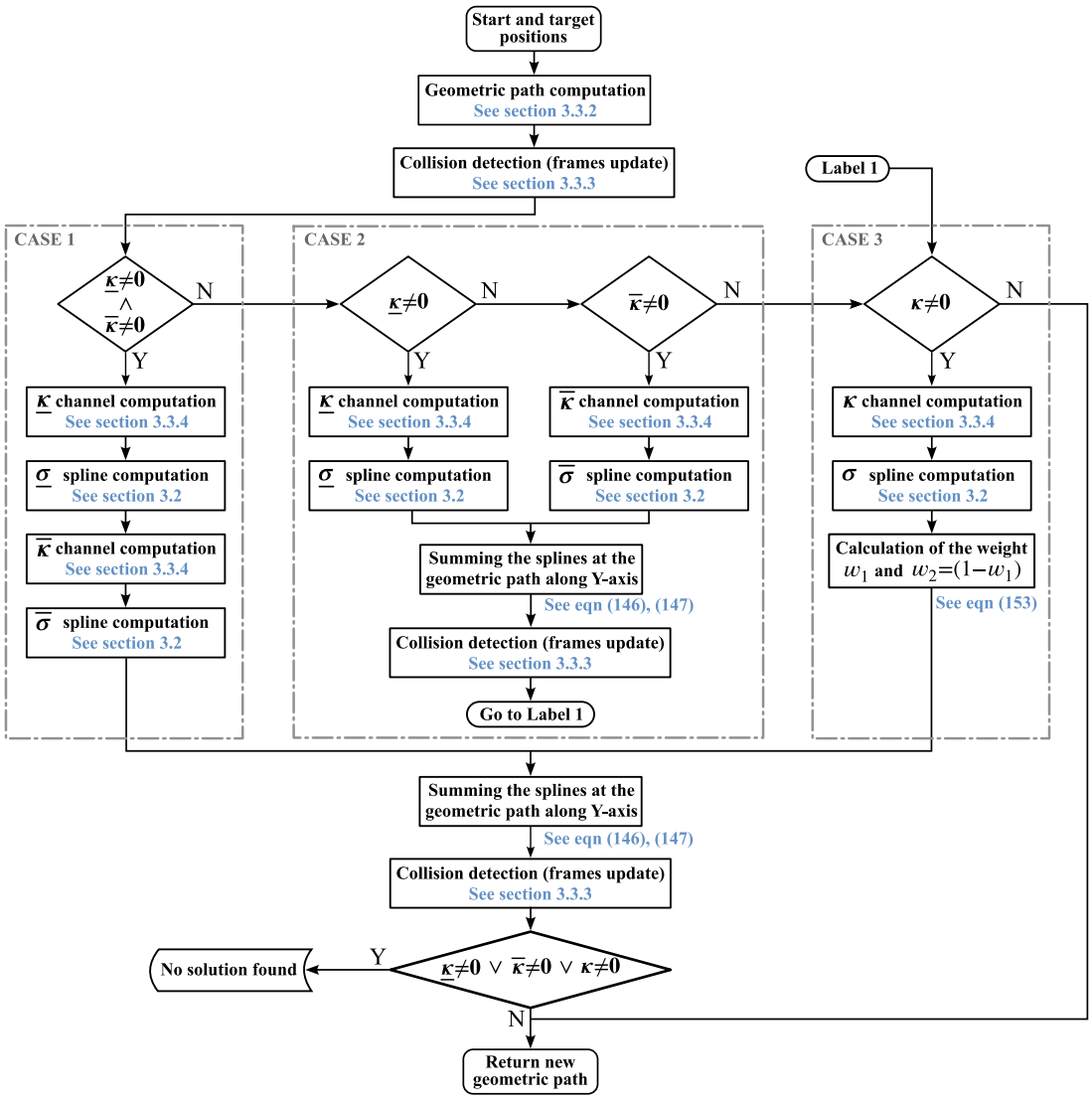


Fig. 12. Pipeline of the collision-free path planning strategy.

in Section 3.3.4 and added to the respective components of $\bar{\boldsymbol{r}}(\tau)$ 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 $\bar{\boldsymbol{r}}(\tau)$. 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 3.3.4 and summed by means of the weights w_1 and w_2 to the $\bar{r}(\tau)$ 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 occurs, 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. 12 returned a modified geometric path, in line with the proposed approach its parametrization 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. Eqs. (118) and (119) 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 Fig. 12, it is added to the Y-component of the master path before calculating s_{max} with (118) and (119). 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:

$$\bar{y}_{i,\phi} \leftarrow \bar{y}_{i,\phi} + h_i, \quad i = 0, \dots, m \tag{154}$$

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, and used in (154).

4. Motion profile with IPTP algorithm

IPTP is a trajectory planning paradigm introduced in a previous authors' work [21], 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 [21], a framework based on this paradigm is

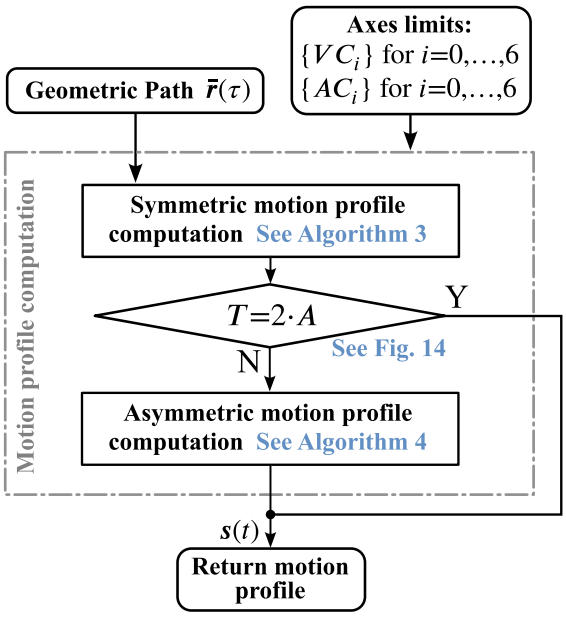


Fig. 13. Pipeline of the motion profile computation.

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 B-spline parameters, which can be obtained through the public attributes $t[0..(m+p+1)]$, $c[0..m]$ 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 easily 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 [21].

Let $s(t)$ be a 1-D time-parametrized spline of degree d, defined by (1), with unknown control points $\boldsymbol{k} = (k_i)_{i=0}^{m}$ and knots $\boldsymbol{t} = (t_i)_{i=0}^{m+d+1}$. Furthermore, $s(t)$ is strictly increasing in the domain range $[t_0, t_{m+d+1}]$, then:

$$s(t_b) > s(t_a) \quad \forall t_b > t_a, \quad \text{with: } t_b, t_a \in [t_0, t_{m+d+1}] \tag{156}$$

Let position, velocity, acceleration and jerk of the axes be defined by a 7-D vector function $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{a}, \boldsymbol{j} : \mathbb{R} \to \mathbb{R}^7$ in the time variable t. Adopting $s(t)$ as the motion profile along the geometric path described in Section 3.3.2, it follows:

$$\boldsymbol{p}(t) = \{\boldsymbol{p}_i(t)\} = (\bar{\boldsymbol{r}} \circ s)(t) \tag{157}$$

$$\boldsymbol{v}(t) = \{\boldsymbol{v}_i(t)\} = \dot{s}(t)(D\bar{\boldsymbol{r}} \circ s)(t) \tag{158}$$

$$\boldsymbol{a}(t) = \{\boldsymbol{a}_i(t)\} = \ddot{s}(t)(D\bar{\boldsymbol{r}} \circ s)(t) + \dot{s}^2(t)(D^{(2)}\bar{\boldsymbol{r}} \circ s)(t) \tag{159}$$

$$\boldsymbol{j}(t) = \{\boldsymbol{j}_i(t)\} = \dddot{s}(t)(D\bar{\boldsymbol{r}} \circ s)(t) + 3\dot{s}(t)\ddot{s}(t)(D^{(2)}\bar{\boldsymbol{r}} \circ s)(t) + \dot{s}^3(t)(D^{(3)}\bar{\boldsymbol{r}} \circ s)(t) \tag{160}$$

Algorithm 2 Symmetric motion profile computation

Input:

1. Geometric path described from 7-D spline $\bar{\boldsymbol{r}}(\tau)$ in domain $[0, s_{max}]$.
2. Kinematic limits VC_i and AC_i, with $i = 0, \dots, 6$.
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}
$A \leftarrow A_{min} := |J_l.t[0] - J_l.t[-1]|$
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 l.
    Set J_c period value:

$$A_c \leftarrow (s_{max} - 2l)/V^{peak} \tag{155}$$

    if $A_c < 0$ **then**
      Initialize an empty Point1D object P.
      $P.val \leftarrow s_{max}/2$; $P.ord \leftarrow 0$; $P.t \leftarrow A$
      $J_l \leftarrow$setValueEn(J_l, P). ▷ Refer to [21]
      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)
  $A \leftarrow A - \varepsilon$; $\varepsilon \leftarrow \varepsilon \cdot \phi$
until checkLimitsOnAxesWithTol($\bar{\boldsymbol{r}}(\tau)$, S, $\{VC_i\}$, $\{AC_i\}$, N, Tol)
return S

Defining VC_i and AC_i, $i = 0, \dots, 6$, as the maximum velocity and acceleration limits of the ith-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:

$$\boldsymbol{s}(t_0) = 0, \quad \boldsymbol{s}(t_{m+d+1}) = s_{max} \tag{161}$$

$$\dot{\boldsymbol{s}}(t_0) = 0, \quad \dot{\boldsymbol{s}}(t_{m+d+1}) = 0 \tag{162}$$

$$\ddot{\boldsymbol{s}}(t_0) = 0, \quad \ddot{\boldsymbol{s}}(t_{m+d+1}) = 0 \tag{163}$$

$$\max_{t \in [t_0, t_{m+d+1}]} \left| \boldsymbol{v}_i(t) \right| \leq VC_i, \quad \text{for } i = 0, \dots, 6 \tag{164}$$

$$\max_{t \in [t_0, t_{m+d+1}]} \left| \boldsymbol{a}_i(t) \right| \leq AC_i, \quad \text{for } i = 0, \dots, 6 \tag{165}$$

Cost function:

$$\text{minimize } T = \sum_{i=0}^{m+d} (t_{i+1} - t_i) = \sum_{i=0}^{m+d} \Delta t_i \tag{166}$$

Fig. 13 reports the pipeline of the proposed solution for the motion profile calculation. The strategy adopted first defines a symmetrical motion profile starting from the geometric path $\bar{\boldsymbol{r}}(\tau)$, within the kinematic limits from (161) to (165), using Algorithm 2. With reference to Fig. 14, the algorithm is parametrized on a Trajectory1D instance J_{in} that defines a positive acceleration impulse in the jerk order. In this

Algorithm 3 Asymmetric motion profile computation

Input:

1. Geometric path described from 7-D spline $\bar{r}(\tau)$ in domain $[0, s_{max}]$.
2. Kinematic limits VC_i and AC_i, with $i = 0, \dots, 6$, and J^{peak}.
3. Trajectory1D instance S from Algorithm 2.
4. Input jerk Trajectory1D object J_{in}.
5. Period A determined by Algorithm 2.
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.
Initialize the empty Point1D objects P_1, P_2, P_3.
repeat
  repeat
    $A_l \leftarrow A_l + \epsilon$
    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.
    $\Delta S \leftarrow S_r.c[0] - S_l.c[-1]$
    $\Delta V \leftarrow V_r.c[0] - V_l.c[-1]$
    $J_c \leftarrow$ Copy of J_{in}
    Multiply J_c by the sign of ΔV.
    $P_1.val \leftarrow \Delta V$; $P_1.ord \leftarrow 1$; $P_1.t \leftarrow J_c.t[-1]$
    $J_c \leftarrow$setValueEn(J_c,P_1) ▷ Refer to [21]
    Compute $D^{(-2)}J_c$ and save the result in V_c.
    $P_2.val \leftarrow V_l.c[-1]$; $P_2.ord \leftarrow 1$; $P_2.t \leftarrow V_c.t[0]$
    $V_c \leftarrow$setValueTr(V_c,P_2) ▷ Refer to [21]
    $P_3.val \leftarrow \Delta S$; $P_3.ord \leftarrow 0$; $P_3.t \leftarrow V_c.t[-1]$
    $V_c \leftarrow$setValueEn(V_c,P_3) ▷ Refer to [21]
    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($\bar{r}(\tau)$, S_{lc}, $\{VC_i\}$, $\{AC_i\}$, N)
  if $|J_c^{peak} - J^{peak}| <$ Tol **then**
    break
  end if
  $A_l \leftarrow A_l - \epsilon$; $\epsilon \leftarrow \epsilon \cdot \phi$
until checkLimitsOnAxesWithTol($\bar{r}(\tau)$,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

application, J_{in} is defined by a cubic spline with the following knots and control points:

$$\text{knots: } (0,\ 0.25A_{min},\ 0.5A_{min},\ 0.75A_{min},\ A_{min}) \tag{167}$$

control points:

$$(0, J^{peak}, J^{peak}, J^{peak}, -J^{peak}, -J^{peak}, -J^{peak}, 0) \tag{168}$$

with A_{min} the initial period of the acceleration section, and J^{peak} the upper limit imposed in absolute value on $\dddot{s}(t)$. As will be clarified in the remainder of this section, the control points defined by (168) limit the codomain of the function (160), in other words:

$$\exists k > 0 : |j_i(t)| \leq k \quad \forall t \in [t_0, t_{m+d+1}], \quad \text{for } i = 0, \dots, 6 \tag{169}$$

Algorithm 2 divides the motion profile into three sections; acceleration, constant velocity, and deceleration. In the jerk order these sections are referred to the Trajectory1D instances J_l, J_c, and J_r. The J_l instance is obtained in the first iteration as a copy of J_{in}. The main mechanism of the algorithm consists in iteratively enlarging J_l, of period A, by a parameter ϵ. The motion profile is computed 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 condition (161) using the relation (155). The overall period of the motion profile is therefore equal to $T = 2A + A_c$, and this becomes shorter of ΔT during the iterations. The algorithm terminates when one, or more, of the limits VC_i and AC_i are exceeded by the respective axes within a tolerance Tol. The function checkLimitsOnAxes is used to detect exceeding of the limits. Firstly it uses a brute-force method that finds the time gridpoint at which the greatest absolute value of the functions (158) and (159) occurs. In this application, an equispaced time grid of N samples is adopted. Once the point on the time grid is selected, the search of the maximum is improved by applying a local method based on the Golden Section search [42] in the range defined by the previous and the next value of the grid. The number of samples N must be high enough to ensure that, in the range where the local search method is applied, the function is unimodal. A variant of this function, called checkLimitsOnAxesWithTol, detects 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 some of the limits VC_i and AC_i are exceeded, if the most restrictive one falls outside the set tolerance, the algorithm returns to the previous step and restarts the iteration scaling the incremental step ϵ of the step ratio ϕ. If none of the kinematic limits are exceeded during the iterations, the period A_c collapses to zero and the constraint (161) is set with the macro-instruction setValueEn (please refer to [21]). If the motion profile resulting from the concatenation of J_l with J_r exceeds the kinematic axes limits, the iteration is repeated for an iteration step value ϵ scaled by ϕ. If $A_c = 0$, no further optimization is possible unless the input parameters of the algorithm are changed, so the calculated motion profile enters in the velocity override setting block. Otherwise, if $A_c \neq 0$, an optimization can be applied to further reduce the overall period T. In this case, the pipeline of Fig. 13 involves post-processing the previously calculated motion profile.

It is common knowledge that a symmetrical motion profile has the peak velocity value V^{peak} strongly constrained by the maximum curvature of the geometric path. In this application case, it is distinctly evident (but not limited to) in cases where the geometric path is missing the rise or fall section, such as that shown in Fig. 15b. In these geometric paths, Algorithm 2 generally terminates due to exceeding an axis kinematic limit at the transition segment, while it may accelerate more in the XY linear section, resulting in a further reduction of the overall period T. Consequently, it is reasonable to deduce that an asymmetrical motion profile can provide significant advantages in terms of the objective function (166). For this reason, the authors propose a post-processing of the symmetrical motion profile computed with Algorithm 2, which consists of further enlarging the acceleration or deceleration section on the side where the kinematic limits allow it. The pseudo-code of this routine is reported in Algorithm 3. Fig. 15a shows an example where Trajectory1D instance J_l, of period A_l, is enlarged until the X-axis exceeds the limit AC_0, while J_r remains unchanged. This procedure defines a two-level velocity profile, and an appropriate junction to bridge the gap ΔV must replace the null Trajectory1D instance J_c. With reference to Algorithm 3, this velocity junction can easily be defined from the same input J_{in} used by the Algorithm 2. Let A_l and A_r be the periods of acceleration and deceleration of S, which defines the motion profile of overall period T at the generic iteration. Using a slicing operation, select the two sections of S corresponding to the domain $[0..A_l]$ and $[A_r - T..T]$, and call the resulting Trajectory1D instances S_l and S_r. In order to define the velocity junction describing the central section, J_c is first defined as a copy of J_{in}, then corrected in sign according to the sign of ΔV. The gap constraint ΔV is set in the jerk order through setValueEn macro-instruction. Following, J_c

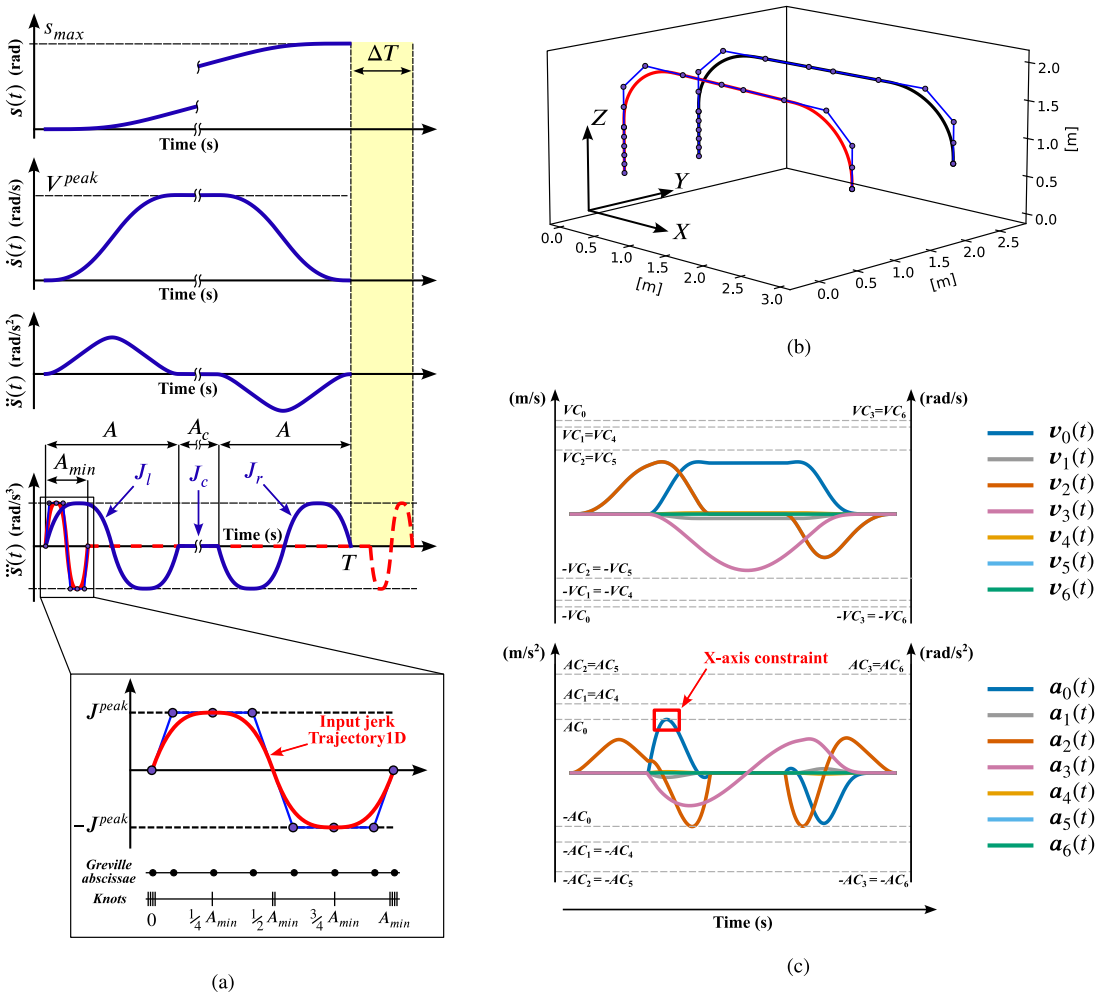


Fig. 14. Schematic representation of the symmetric motion profile generation algorithm reported in Algorithm 2. (a) Motion profile and input jerk Trajectory1D instance. (b) Geometric path $\bar{r}(\tau)$ of reference. (c) Joint trajectories obtained with (158) and (159). The algorithm is terminated for reaching the acceleration limit AC_0 on X-axis.

is integrated twice obtaining V_c, which is translated vertically with setValueTr macro-instruction (please refer to [21]) to ensure velocity continuity in the concatenation with the acceleration and deceleration sections. Finally V_c is enlarged to set the constraint of ΔS, again using setValueEn macro-instruction. Iteration terminate when at least one of the limits AC_i or VC_i is exceeded within the tolerance Tol, or if the maximum amplitude J_c^{peak} of J_c reaches the limit J^{peak} within the same tolerance.

5. Case study

This section presents a use case of the proposed motion planning solution for a classic nest sorting task. Fig. 16 shows the 2-D layout of the machine utilized 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. 16 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 Table 2. Regarding the dimensions of the parts involved in the sorting operation, Table 3 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 analysis are subsequently defined as reported in Section 3.3.3 by adopting an edge length of the head octagon of $l_e = 0.0746\,\text{m}$ and an outer offset of $\varepsilon = 0.025\,\text{m}$. The transmission ratios' vector $\boldsymbol{K}_r$, necessary for the geometric path parametrization, can be found in Table 4. According with the proposed approach, the parameters that still must be defined to uniquely determine the geometric paths of the end-effectors are: $z_{sf} = 1.13\,\text{m}$, $z_{tr} = 2\,\text{m}$, $L_{T,min} = 0.225\,\text{m}$, $A_{ch,min} = 0.05\,\text{m}$.

The geometric paths computed using the approach detailed in Section 3 are shown in Fig. 17. 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. 8). 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. 8). 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 3.3.2 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 (67), lower than the threshold $L_{T,min}$. The

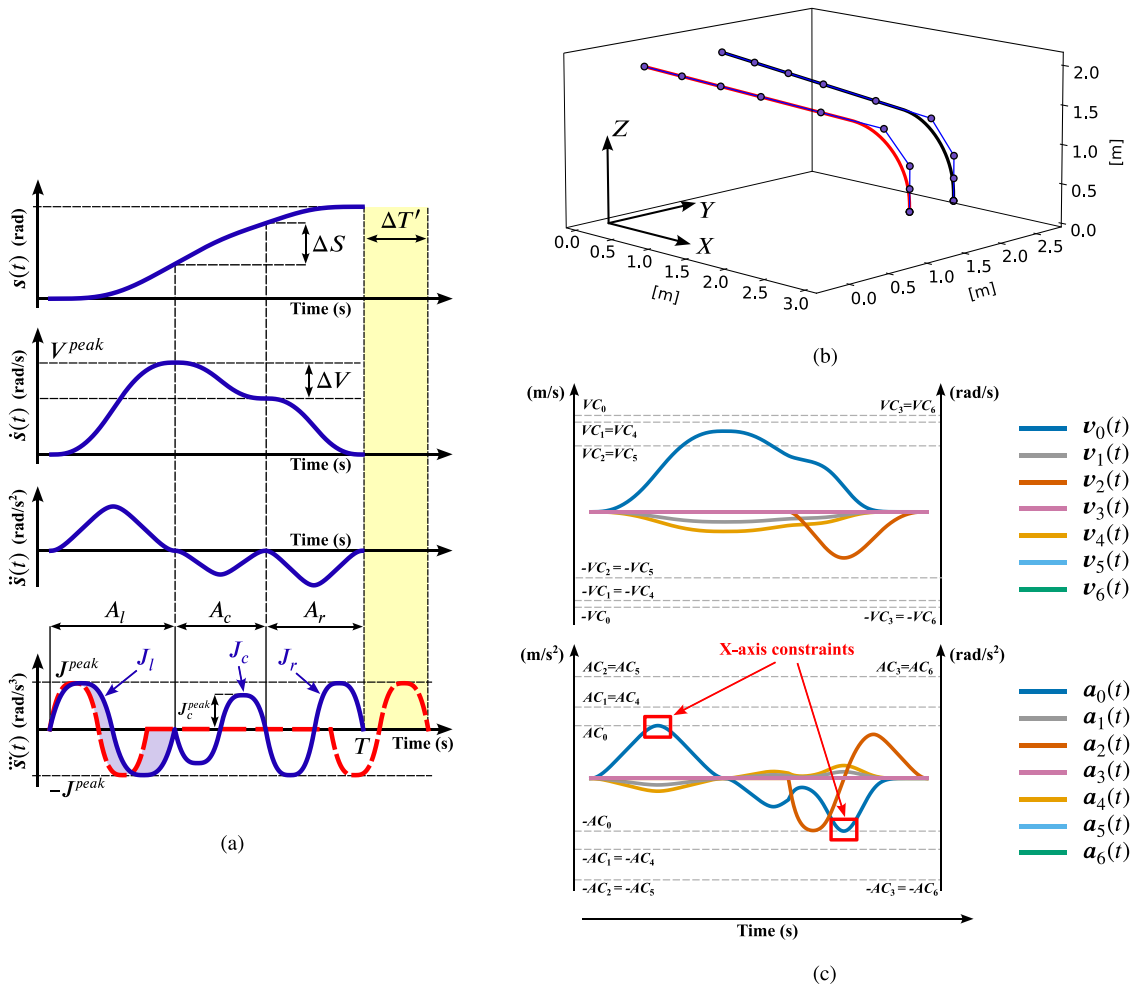


Fig. 15. Schematic representation of the asymmetrical motion profile generation algorithm reported in Algorithm 3. (a) Asymmetrical motion profile compared with the symmetrical solution computed with Algorithm 2. (b) Geometric path $\bar{r}(\tau)$ of reference. (c) Joint trajectories obtained with (158) and (159).

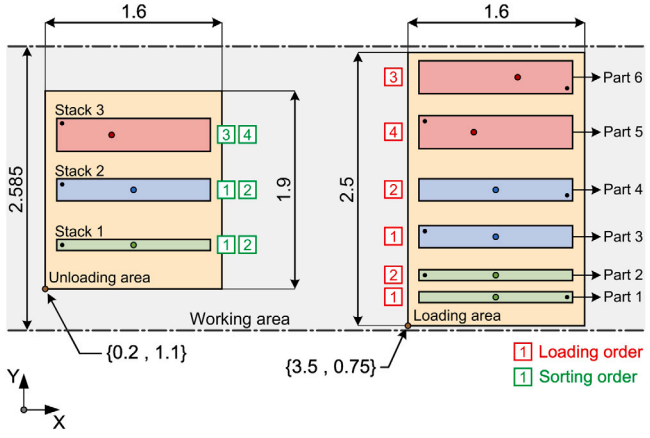


Fig. 16. Working area 2-D layout referred to the case study. All dimensions are expressed in metres.

master path is therefore separated into three sub-paths, corresponding to cases 15, 14, and 16 in Fig. 8. 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. 8. 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 Table 2, each of these movements performs a

Table 2
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 3
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}

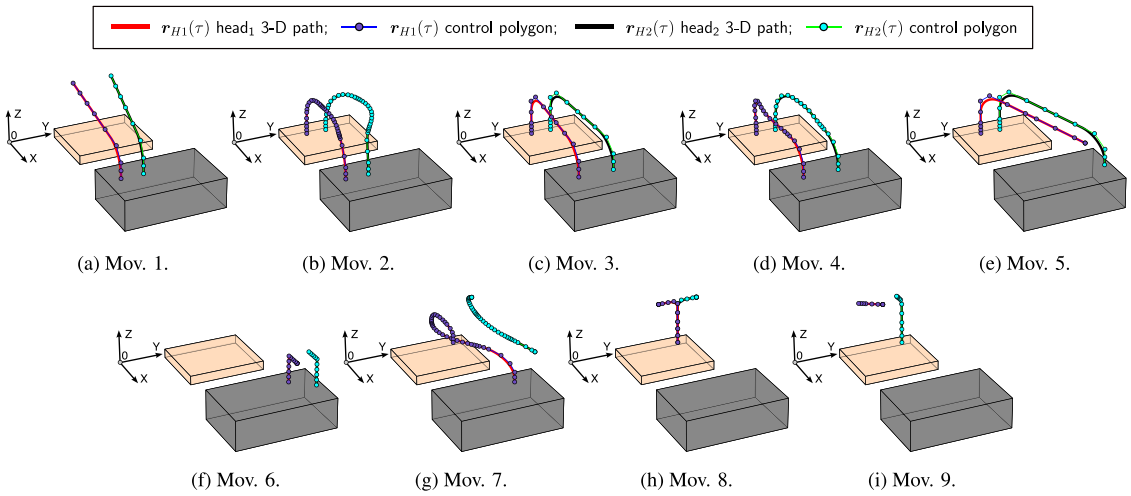


Fig. 17. End-effector geometric paths for the nine movement of the case study.

Table 4
Transmission ratios 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)	2.14 (rad/s)

Table 5
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

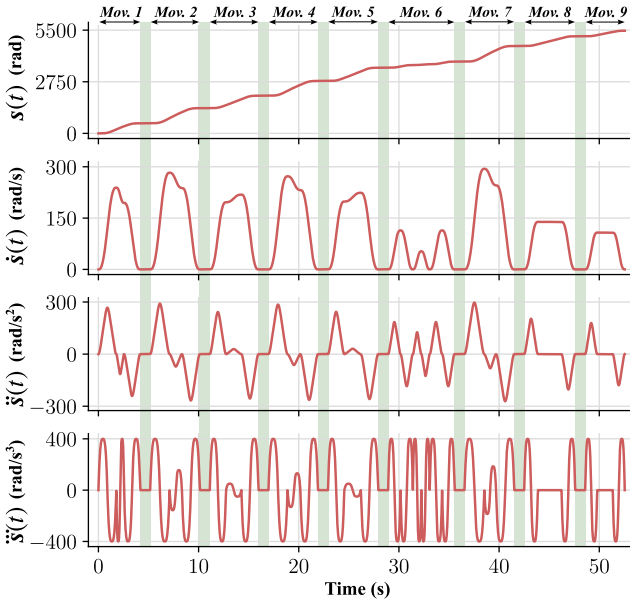


Fig. 18. Motion profile of the sorting sequence under study. Between each movement, a pause of 1 s is inserted. A 100% velocity override is used.

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 Table 6. 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 Table 5.

In movement 2, the bounding box of the group 1 exceeds the working area during the rototranslation. According to the pipeline of Fig. 12, this movement falls into case 2 because the frame sampling analysis returns $\underline{\kappa} \neq 0$. The bottom overrun spline $\underline{\sigma}(\tau)$ is computed with the method detailed in Section 3, and summed to the Y-axis of $r_{H1}(\tau)$ using (146). After that, the frame sampling is updated, and the overlap spline $\sigma(\tau)$ is computed and summed to the Y-axis of $r_{H2}(\tau)$ using (147). In movement 4, the rotation of head 1 leads to $\kappa \neq 0$, which corresponds to case 3 in the pipeline reported in Fig. 12. The overlap spline $\sigma(\tau)$ is then computed and summed to the Y-axes of $\bar{r}(\tau)$ using (146) and (147) using weights $w_1 = w_2 = 0.5$. Similarly to the 2nd movement, movement 7 also falls in case 2 of the pipeline of Fig. 12, with the difference that it is the bounding box of group 2 that exceeds the working area, resulting in $\overline{\kappa} \neq 0$. The top overrun spline $\overline{\sigma}(\tau)$ is computed and summed to the Y-axis of $r_{H2}(\tau)$ using (147). Hereafter, the frame sampling is updated, and the overlap spline $\sigma(\tau)$ is computed and summed to the Y-axis of $r_{H1}(\tau)$ using (146).

The motion profile computed for the complete paths sequence using the pipeline reported in Section 4 is shown in Fig. 18. A pause of 1 s is inserted between successive movements. The axis limits adopted to define the constraints (164) and (165) can be found in Table 4. The following parameters are used for both Algorithm 2 and 3: $J^{peak} = 400$ rad/s^3, number of samples N = 150, Tol= 10^{-9}, initial incremental step $\varepsilon = 0.1$, step ratio $\phi = 0.1$. Table 5 details the periods T for each movement, corresponding to the cost function (166). In Fig. 19 the joint trajectories computed with (157), (158) and (159) are reported, highlighting the axis limits that terminate the algorithms of the pipeline of Fig. 13. 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 3, while Algorithm 2 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

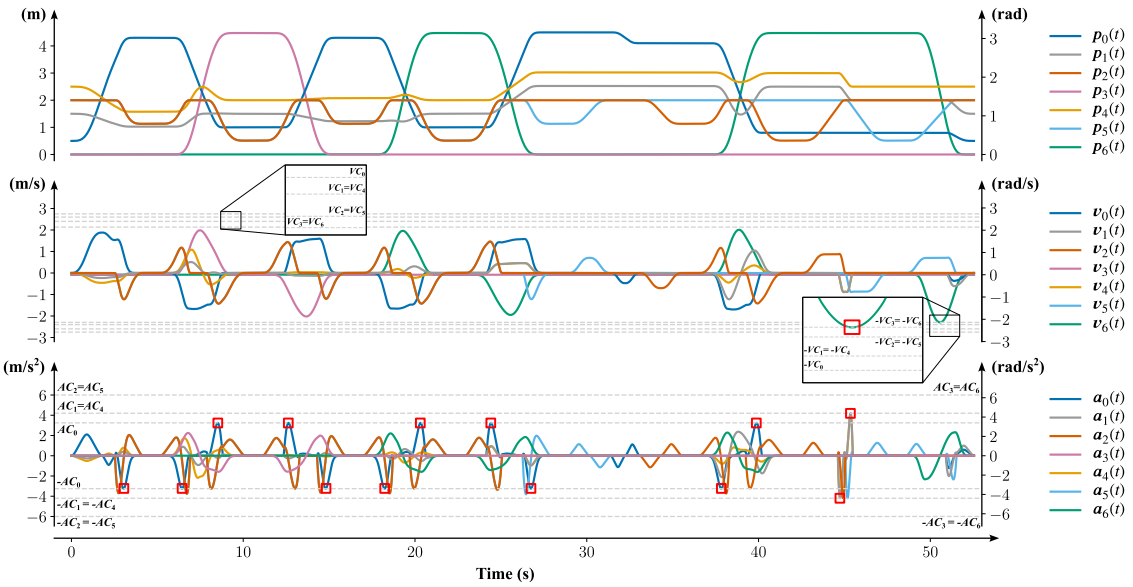


Fig. 19. Joint trajectory of the movement sequence of the case study. The red square markers are the axis limits that terminate the algorithms of the pipeline in Fig. 13. Between each movement, a pause of 1 s is inserted. A 100% velocity override is used.

both transition segments. The motion profile is asymmetrical due to the different height of the loading and unloading positions. In particular, Algorithm 2 terminates at the transition segment corresponding to the loading zone, while Algorithm 3 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 2 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. For example, in the real application documented in the attachment Video1.mp4, $J^{peak} = 2000\,\mathrm{rad/s^3}$ is used. 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. In this regard, please refer to Video1.mp4 and Video2.mp4 in the supplementary material. Fig. 20 shows the geometric paths trails in the simulated environment, with the end-effectors in target position for each movement.

Fig. 21 reports the resulting trails of the vertex positions of the parts bounding boxes, corresponding to the movements 2, 4, and 7. Finally, Fig. 22 shows the photos of the test machine in the start positions for each movement of the sequence.

The real-time control strategy adopted in the real application is strongly influenced by the hardware configuration installed on the test machine, which uses Mitsubishi Electric®motors and drives that communicate, through CANopen®protocol, with the programmable logic controller (PLC). 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. In the test, the axes are controlled by position feedback and torque feedforward using a B-spline set-point generator and a cycle times of 2 ms.

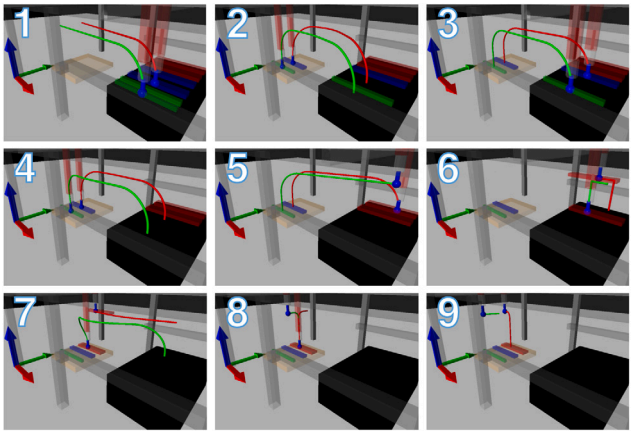


Fig. 20. The process of sorting of the nest under study in the simulated environment. Sub-pictures 1 to 9 represent the target position and the end-effectors trails.

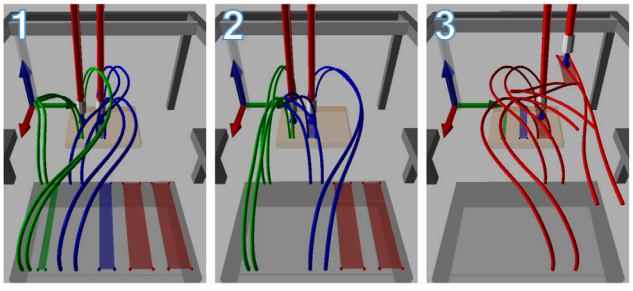
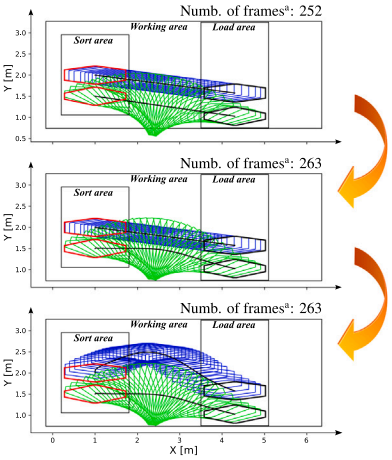
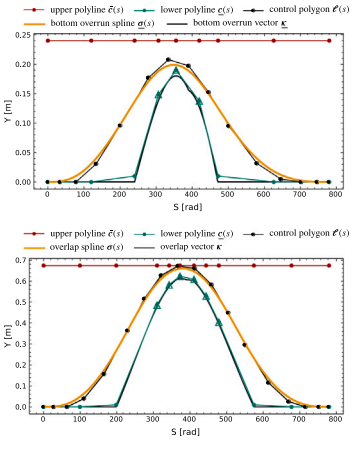
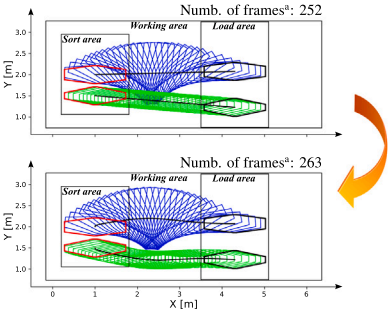
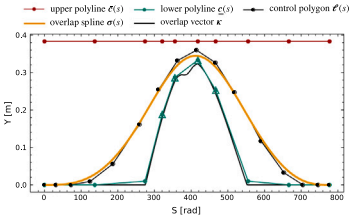
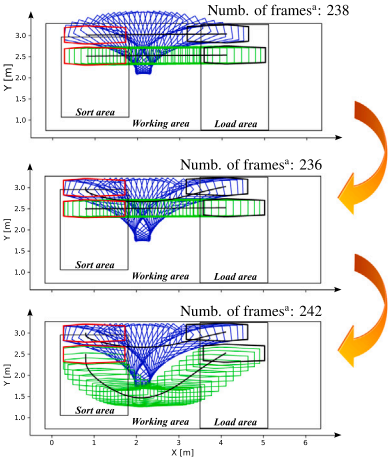
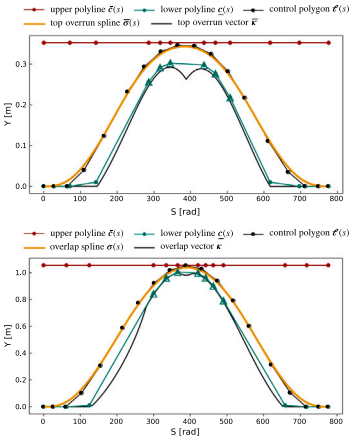


Fig. 21. Vertex position trails of part bounding boxes. Sub-pictures 1 to 3 correspond to movements 2, 4, and 7.

6. Conclusions

This paper proposes a new path-based motion planning approach for 7-d.o.f. dual-arm 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), characterizing this work in the field of pick-and-place application. The geometric path is described by a 7-D spline in its B-form, defining the two end-effectors paths through a classical five-segment planar pick-end-place trajectories, where the adjacent straight lines are blended by quintic Bézier curves with minimized curvature.

Table 6
Results of the three movements in which the collision avoidance algorithm detailed in Section 3.3.5 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 $\epsilon = 0.025$ 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.

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-parametrized spline. The problem of maximum geometric path curvature is partially addressed through asymmetrical motion profile planning, which defines two velocity levels. 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. The proposed solution allows easy expansion of the kinematics to a dual-gantry configuration, thereby considering a total of four end-effectors sharing the same workspace. In the particular industrial field, this 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. 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 may extend this work to other applications by exploring collision-free path planning in 3-D space and extending the system to a dual-gantry configuration, as well as evaluating the benefits of adopting a corner smoothing algorithm that allows for C^3-continuity of axis trajectories.

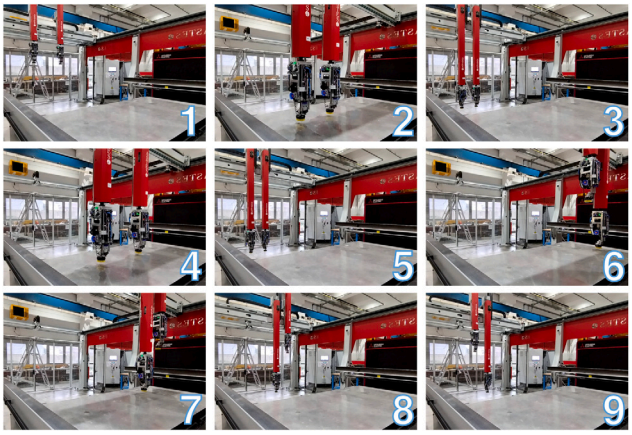


Fig. 22. The process of sorting of the nest under study in the real world. Sub-pictures 1 to 9 represent the starting position of the point-to-point movements, according to Table 2.

CRediT authorship contribution statement

Marco Riboli: Conceptualization, Methodology, Software, Data curation, Validation, Writing – original draft, Writing – review & editing. **Matthieu Jaccard:** Writing – original draft, Software, Validation. **Marco Silvestri:** Conceptualization, Project administration. **Alessandra Aimi:** Formal analysis, Supervision. **Cesare Malara:** Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marco Silvestri reports financial support was provided by Innosuisse - Swiss Innovation Agency (funding application no. 57715.1 IP-ICT). Cesare Malara reports a relationship with Astes4 SA that includes: employment.

Data availability

No data was used for the research described in the article.

Acknowledgements

This research was funded by Innosuisse - Swiss Innovation Agency, funding application no. 57715.1 IP-ICT.

Appendix A. Definition of cost function coefficients

In the following, the cost function matrix P of Eq. (18) is derived. From (10):

$$\begin{aligned}\Delta_2 b_i &= \frac{b_{i+1}-b_i}{u^*_{i+1}-u^*_i} - \frac{b_i-b_{i-1}}{u^*_i-u^*_{i-1}} \\ &= \frac{1}{\Delta u^*_{i+1}\Delta u^*_i}\left[(b_{i+1}-b_i)\Delta u^*_i - (b_i-b_{i-1})\Delta u^*_{i+1}\right] \\ &= \frac{1}{K_i}\left[b_{i+1}\Delta u^*_i - b_i(\Delta u^*_i + \Delta u^*_{i+1}) + b_{i-1}\Delta u^*_{i+1}\right] \end{aligned} \tag{170}$$

with:

$$\Delta u^*_i = u^*_i - u^*_{i-1} > 0 \tag{171}$$

$$K_i = \Delta u^*_{i+1}\Delta u^*_i > 0 \tag{172}$$

It follow that:

$$\begin{aligned}(\Delta_2 b_i)^2 = \frac{1}{K_i^2}\left[b_{i+1}^2\Delta u_i^{*2} + b_i^2(\Delta u_i^* + \Delta u_{i+1}^*)^2 + \right. \\ b_{i-1}^2\Delta u_{i+1}^{*2} - 2b_{i+1}b_i\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*) - \\ \left. 2b_i b_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^* + 2b_{i+1}b_{i-1}\Delta u_i^*\Delta u_{i+1}^*\right] \\ = g(b_{i-1}, b_i, b_{i+1})\end{aligned} \tag{173}$$

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

$$\begin{aligned}\frac{\delta f(\boldsymbol{b})}{\delta b_i} = \frac{\delta(\Delta_2 b_1)^2}{\delta b_i} + \cdots + \underbrace{\frac{\delta(\Delta_2 b_{i-1})^2}{\delta b_i}}_{(1^{\text{st}}\text{comp.})} + \underbrace{\frac{\delta(\Delta_2 b_i)^2}{\delta b_i}}_{(2^{\text{nd}}\text{comp.})} + \underbrace{\frac{\delta(\Delta_2 b_{i+1})^2}{\delta b_i}}_{(3^{\text{rd}}\text{comp.})} \\ \text{For (173) only these component are not null} \\ + \cdots + \frac{\delta(\Delta_2 b_{m-1})^2}{\delta b_i}\end{aligned} \tag{174}$$

1st component:

$$\begin{aligned}\frac{\delta(\Delta_2 b_{i-1})^2}{\delta b_i} &= \frac{\delta}{\delta b_i}\left[\frac{1}{K_{i-1}^2}\left[b_i^2\Delta u_{i-1}^{*2} + b_{i-1}^2(\Delta u_{i-1}^* + \Delta u_i^*)^2 \right.\right. \\ &\quad + b_{i-2}^2\Delta u_i^{*2} - 2b_i b_{i-1}\Delta u_{i-1}^*(\Delta u_{i-1}^* + \Delta u_i^*) - \\ &\quad \left.\left. 2b_{i-1}b_{i-2}(\Delta u_{i-1}^* + \Delta u_i^*)\Delta u_i^* + 2b_i b_{i-2}\Delta u_{i-1}^*\Delta u_i^*\right]\right] \\ &= \frac{2}{K_{i-1}^2}\left[b_i\Delta u_{i-1}^{*2} - b_{i-1}\Delta u_{i-1}^*(\Delta u_{i-1}^* + \Delta u_i^*) + \right. \\ &\quad \left. b_{i-2}\Delta u_{i-1}^*\Delta u_i^*\right]\end{aligned} \tag{175}$$

2nd component:

$$\begin{aligned}\frac{\delta(\Delta_2 b_i)^2}{\delta b_i} &= \frac{\delta}{\delta b_i}\left[\frac{1}{K_i^2}\left[b_{i+1}^2\Delta u_i^{*2} + b_i^2(\Delta u_i^* + \Delta u_{i+1}^*)^2 + \right.\right. \\ &\quad b_{i-1}^2\Delta u_{i+1}^{*2} - 2b_{i+1}b_i\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*) - \\ &\quad \left.\left. 2b_i b_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^* + 2b_{i+1}b_{i-1}\Delta u_i^*\Delta u_{i+1}^*\right]\right] \\ &= \frac{2}{K_i^2}\left[b_i(\Delta u_i^* + \Delta u_{i+1}^*)^2 - b_{i+1}\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*) \right. \\ &\quad \left. - b_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^*\right]\end{aligned} \tag{176}$$

3rd component:

$$\begin{aligned}\frac{\delta(\Delta_2 b_{i+1})^2}{\delta b_i} &= \frac{\delta}{\delta b_i}\left[\frac{1}{K_{i+1}^2}\left[b_{i+2}^2\Delta u_{i+1}^{*2} + b_{i+1}^2(\Delta u_{i+1}^* + \Delta u_{i+2}^*)^2 \right.\right. \\ &\quad + b_i^2\Delta u_{i+2}^{*2} - 2b_{i+2}b_{i+1}\Delta u_{i+1}^*(\Delta u_{i+1}^* + \Delta u_{i+2}^*) - \\ &\quad \left.\left. 2b_{i+1}b_i(\Delta u_{i+1}^* + \Delta u_{i+2}^*)\Delta u_{i+2}^* + 2b_{i+2}b_i\Delta u_{i+1}^*\Delta u_{i+2}^*\right]\right] \\ &= \frac{2}{K_{i+1}^2}\left[b_i\Delta u_{i+2}^{*2} - b_{i+1}(\Delta u_{i+1}^* + \Delta u_{i+2}^*)\Delta u_{i+2}^* \right. \\ &\quad \left. + b_{i+2}\Delta u_{i+1}^*\Delta u_{i+2}^*\right]\end{aligned} \tag{177}$$

Summing the three components (175), (176) and (177):

$$\begin{aligned}\frac{\delta f(\boldsymbol{b})}{\delta b_i} &= \frac{2b_i\Delta u_{i-1}^{*2}}{K_{i-1}^2} - \frac{2b_{i-1}\Delta u_{i-1}^*(\Delta u_{i-1}^* + \Delta u_i^*)}{K_{i-1}^2} + \frac{2b_{i-2}\Delta u_{i-1}^*\Delta u_i^*}{K_{i-1}^2} \\ &\quad + \frac{2b_i(\Delta u_i^* + \Delta u_{i+1}^*)^2}{K_i^2} - \frac{2b_{i+1}\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*)}{K_i^2} \\ &\quad - \frac{2b_{i-1}(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^*}{K_i^2} + \frac{2b_i\Delta u_{i+2}^{*2}}{K_{i+1}^2} \\ &\quad - \frac{2b_{i+1}(\Delta u_{i+1}^* + \Delta u_{i+2}^*)\Delta u_{i+2}^*}{K_{i+1}^2} + \frac{2b_{i+2}\Delta u_{i+1}^*\Delta u_{i+2}^*}{K_{i+1}^2} \\ &= 2b_{i-2}A_i - 2b_{i-1}B_i + 2b_iC_i - 2b_{i+1}D_i + 2b_{i+2}E_i\end{aligned} \tag{178}$$

with:

$$A_i = \frac{\Delta u_{i-1}^*\Delta u_i^*}{K_{i-1}^2} > 0 \tag{179}$$

$$B_i = \frac{\Delta u_{i-1}^*(\Delta u_{i-1}^* + \Delta u_i^*)}{K_{i-1}^2} + \frac{(\Delta u_i^* + \Delta u_{i+1}^*)\Delta u_{i+1}^*}{K_i^2} > 0 \tag{180}$$

$$C_i = \frac{\Delta u_{i-1}^{*2}}{K_{i-1}^2} + \frac{(\Delta u_i^* + \Delta u_{i+1}^*)^2}{K_i^2} + \frac{\Delta u_{i+2}^{*2}}{K_{i+1}^2} > 0 \tag{181}$$

$$D_i = \frac{\Delta u_i^*(\Delta u_i^* + \Delta u_{i+1}^*)}{K_i^2} + \frac{(\Delta u_{i+1}^* + \Delta u_{i+2}^*)\Delta u_{i+2}^*}{K_{i+1}^2} > 0 \tag{182}$$

$$E_i = \frac{\Delta u^*_{i+1}\Delta u^*_{i+2}}{K^2_{i+1}} > 0 \tag{183}$$

from (178):

$$\left.\frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_j}\right|_{j\notin[i-2,i-1,i,i+1,i+2]} = 0 \tag{184}$$

$$\frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i-2}} = 2A_i, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i-1}} = -2B_i, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_i} = 2C_i,$$

$$\frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i+1}} = -2D_i, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_{i+2}} = 2E_i \tag{185}$$

The special cases $\delta f(\boldsymbol{b})/\delta b_1$ and $\delta f(\boldsymbol{b})/\delta b_{m-1}$ remain to be defined. From (174) it follows that

$$\frac{\delta f(\boldsymbol{b})}{\delta b_1} = -2b_0 B_1 + 2b_1 C_1 - 2b_2 D_1 + 2b_3 E_1 \tag{186}$$

with

$$B_1 = \frac{(\Delta u^*_1 + \Delta u^*_2)\Delta u^*_2}{K^2_1} > 0 \tag{187}$$

$$C_1 = \frac{(\Delta u^*_1 + \Delta u^*_2)^2}{K^2_1} + \frac{\Delta u^{*2}_3}{K^2_2} > 0 \tag{188}$$

$$D_1 = \frac{\Delta u^*_1(\Delta u^*_1 + \Delta u^*_2)}{K^2_1} + \frac{(\Delta u^*_2 + \Delta u^*_3)\Delta u^*_3}{K^2_2} > 0 \tag{189}$$

$$E_1 = \frac{\Delta u^*_2 \Delta u^*_3}{K^2_2} > 0 \tag{190}$$

and

$$\frac{\delta f(\boldsymbol{b})}{\delta b_{m-1}} = 2b_{m-3}A_{m-1} - 2b_{m-2}B_{m-1} + 2b_{m-1}C_{m-1} - 2b_m D_{m-1} \tag{191}$$

with

$$A_{m-1} = \frac{\Delta u^*_{m-2}\Delta u^*_{m-1}}{K^2_{m-2}} > 0 \tag{192}$$

$$B_{m-1} = \frac{\Delta u^*_{m-2}(\Delta u^*_{m-2} + \Delta u^*_{m-1})}{K^2_{m-2}} + \frac{(\Delta u^*_{m-1} + \Delta u^*_m)\Delta u^*_m}{K^2_{m-1}} > 0 \tag{193}$$

$$C_{m-1} = \frac{\Delta u^{*2}_{m-2}}{K^2_{m-2}} + \frac{(\Delta u^*_{m-1} + \Delta u^*_m)^2}{K^2_{m-1}} > 0 \tag{194}$$

$$D_{m-1} = \frac{\Delta u^*_{m-1}(\Delta u^*_{m-1} + \Delta u^*_m)}{K^2_{m-1}} > 0 \tag{195}$$

In our case, $b_0 = b_m = 0$, so the first component of (186) and the last component of (191) are null. From (186) and (191):

$$\left.\frac{\delta^2 f(\boldsymbol{b})}{\delta b_1 \delta b_j}\right|_{j>3} = 0, \quad \left.\frac{\delta^2 f(\boldsymbol{b})}{\delta b_{m-1}\delta b_j}\right|_{j<m-3} = 0 \tag{196}$$

$$\frac{\delta^2 f(\boldsymbol{b})}{\delta^2 b_1} = 2C_1, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_1 \delta b_2} = -2D_1, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_i \delta b_3} = 2E_1 \tag{197}$$

$$\frac{\delta^2 f(\boldsymbol{b})}{\delta^2 b_{m-1}} = 2C_{m-1}, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_{m-1}\delta b_{m-2}} = -2B_{m-1}, \quad \frac{\delta^2 f(\boldsymbol{b})}{\delta b_{m-1}\delta b_{m-3}} = 2A_{m-1} \tag{198}$$

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

$$\nabla^2 f(\boldsymbol{b}) = 2\underbrace{\begin{bmatrix} C_1 & -D_1 & E_1 & & & & & 0 \\ -B_2 & C_2 & -D_2 & E_2 & & & & \\ A_3 & -B_3 & C_3 & -D_3 & E_3 & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & A_{m-3} & -B_{m-3} & C_{m-3} & -D_{m-3} & E_{m-3} \\ & & & & A_{m-2} & -B_{m-2} & C_{m-2} & -D_{m-2} \\ 0 & & & & & A_{m-1} & -B_{m-1} & C_{m-1} \end{bmatrix}}_{\text{Equal to matrix } P \text{ of Eq. (17)}} \tag{199}$$

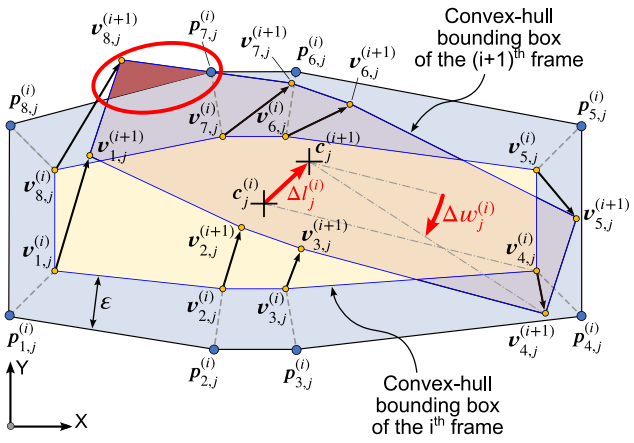


Fig. 23. Convex polygon of group j in the frames i and $i+1$. The expanded polygon in the ith frame is also reported. In this case, an intermediate frame must be inserted since vertex $v^{(i+1)}_{8,j}$ exceeds the expanded polygon.

It follows that the matrix P of Eq. (17) is defined by

$$P = \frac{1}{2}\nabla^2 f(\boldsymbol{b}) \tag{200}$$

Appendix B. Frame sampling strategy

Section 3.3.3 detailed the calculation of Y-intersections κ_i, $\overline{\kappa}_i$ and $\underline{\kappa}_i$, relative to a frame i corresponding to the ith element of a strictly increasing sequence $\boldsymbol{\tau} = (\tau_i)^{(\mathrm{M})}_{i=0}$. This section details the methodology used in this work to define the $\boldsymbol{\tau}$ vector.

With reference to Fig. 23, the following definitions are given for frame i and group j:

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

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

$$\begin{bmatrix} v^{(i+1)}_{k,j,x} \\ v^{(i+1)}_{k,j,y} \\ 1 \end{bmatrix} = Q^{(i)}_j \begin{bmatrix} v^{(i)}_{k,j,x} \\ v^{(i)}_{k,j,y} \\ 1 \end{bmatrix} \quad \text{with: } k = 1, \ldots, \mathrm{N}_j \tag{201}$$

with:

$$Q^{(i)}_j = \begin{bmatrix} C_{\Delta w^{(i)}_j} & -S_{\Delta w^{(i)}_j} & x^{(i)}_{c,j}(1 - C_{\Delta w^{(i)}_j}) + y^{(i)}_{c,j} S_{\Delta w^{(i)}_j} + \Delta x^{(i)}_j \\ S_{\Delta w^{(i)}_j} & C_{\Delta w^{(i)}_j} & -x^{(i)}_{c,j} S_{\Delta w^{(i)}_j} + y^{(i)}_{c,j}(1 - C_{\Delta w^{(i)}_j}) + \Delta y^{(i)}_j \\ 0 & 0 & 1 \end{bmatrix} \tag{202}$$

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 summarized as follows. Define $\boldsymbol{\tau}_s$ the initial frames sequence. For each ith span of the vector $\boldsymbol{\tau}_s$ the following condition is checked:

$$\forall j \in \{1,2\}, \forall k \in \{1, \ldots, \mathrm{N}_j\}, \forall r \in \{1, \ldots, \mathrm{N}_j\}, \quad \mathcal{F}\left(\boldsymbol{v}^{(i+1)}_{k,j} \;\middle|\; \boldsymbol{p}^{(i)}_{r,j}, \boldsymbol{p}^{(i)}_{(r+1)\bmod(\mathrm{N}_j+1),j}\right) \geq 0 \tag{203}$$

with $\mathcal{F}(\{x,y\} \mid \{x_1,y_1\},\{x_2,y_2\})$ direction of the cross product between vectors $\{x_2-x_1, y_2-y_1\}$ and $\{x-x_1, y-y_1\}$, defined by $\mathrm{sgn}((x_2-x_1)(y-$

$y_1) - (y_2 - y_1)(x - x_1)$). Condition (203) 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 (203) 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 (203).

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104534.

References

[1] S. Lee, H. Moradi, 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, http://dx.doi.org/10.1109/ISATP.1997.615376.

[2] Yanqiong Fei, Ding Fuqiang, Zhao Xifang, Collision-free motion planning of dual-arm reconfigurable robots, Robot. Comput.-Integr. Manuf. (ISSN: 0736-5845) 20 (4) (2004) 351–357, http://dx.doi.org/10.1016/j.rcim.2004.01.002.

[3] Steven M. LaValle, Rapidly-Exploring Random Trees : a New Tool for Path Planning, The Annual Research Report, 1998.

[4] Jun Kurosu, Ayanori Yorozu, Masaki Takahashi, Simultaneous dual-arm motion planning for minimizing operation time, Appl. Sci. 7 (2017) 1210, http://dx.doi.org/10.3390/app7121210.

[5] Wubin Shi, Ke Wang, Chong Zhao, Mengqi Tian, Obstacle avoidance path planning for the dual-arm robot based on an improved RRT algorithm, Appl. Sci. 12 (2022) 4087, http://dx.doi.org/10.3390/app12084087.

[6] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. (1959) http://dx.doi.org/10.1007/BF01386390.

[7] Sertac Karaman, Emilio Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robotic Res. - IJRR 30 (2011) 846–894, http://dx.doi.org/10.1177/0278364911406761.

[8] Richard Bellman, Dynamic Programming, Dover Publications, ISBN: 9780486428093, 1957.

[9] Andreas Völz, Knut Graichen, An optimization-based approach to dual-arm motion planning with closed kinematics, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and System, IROS, 2018, pp. 8346–8351, http://dx.doi.org/10.1109/IROS.2018.8593927.

[10] Tomas Berglund, Andrej Brodnik, HÅkan Jonsson, Mats Staffanson, Inge Soderkvist, Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles, IEEE Trans. Autom. Sci. Eng. 7 (1) (2010) 167–172, http://dx.doi.org/10.1109/TASE.2009.2015886.

[11] Youdong Chen, Ling Li, Collision-free trajectory planning for dual-robot systems using B-splines, Int. J. Adv. Robot. Syst. 14 (2017) 172988141772802, http://dx.doi.org/10.1177/1729881417728021.

[12] Wolfgang Boehm, Hartmut Prautzsch, The insertion algorithm, Comput. Aided Des. (ISSN: 00104485) 17 (2) (1985) 58–59, http://dx.doi.org/10.1016/0010-4485(85)90246-5.

[13] L.A. Piegl, W. Tiller, The NURBS Book, second ed., Springer-Verlag, New York, NY, USA, ISBN: 3540615458, 1996.

[14] Younsung Choi, Donghyung Kim, Soonwoong Hwang, Hyeonguk Kim, Namwun Kim, Changsoo Han, Dual-arm robot motion planning for collision avoidance using B-spline curve, Int. J. Precis. Eng. Manuf. 18 (2017) 835–843, http://dx.doi.org/10.1007/s12541-017-0099-z.

[15] A. Gasparetto, V. Zanotto, Optimal trajectory planning for industrial robots, Adv. Eng. Softw. (ISSN: 09659978) 41 (4) (2010) 548–556, http://dx.doi.org/10.1016/j.advengsoft.2009.11.001.

[16] Paolo Boscariol, Dario Richiedei, Energy-efficient design of multipoint trajectories for cartesian robots, Int. J. Adv. Manuf. Technol. 102 (2019) http://dx.doi.org/10.1007/s00170-018-03234-4.

[17] Wei Lin, XiaoQi Chen, Dual-arm cartesian robotic system for parallel tasking, in: Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium, Vol. 1 (Cat. No.01CH37180), 2001, pp. 458–463 vol.1, http://dx.doi.org/10.1109/IROS.2001.973399.

[18] Josh Barnett, Mike Duke, Chi Kit Au, Shen Him Lim, Work distribution of multiple cartesian robot arms for kiwifruit harvesting, Comput. Electron. Agric. (ISSN: 0168-1699) 169 (2020) 105202, http://dx.doi.org/10.1016/j.compag.2019.105202.

[19] T. Huang, P.F. Wang, J.P. Mei, X.M. Zhao, D.G. Chetwynd, Time minimum trajectory planning of a 2-DOF translational parallel robot for pick-and-place operations, CIRP Ann. (ISSN: 0007-8506) 56 (1) (2007) 365–368, http://dx.doi.org/10.1016/j.cirp.2007.05.085.

[20] J.-F. Gauthier, J. Angeles, S. Nokleby, Optimization of a test trajectory for SCARA systems, in: Jadran Lenarčič, Philippe Wenger (Eds.), Advances in Robot Kinematics: Analysis and Design, Springer Netherlands, Dordrecht, ISBN: 978-1-4020-8600-7, 2008, pp. 225–234, http://dx.doi.org/10.1007/978-1-4020-8600-7_24.

[21] Marco Riboli, Fabio Corradini, Marco Silvestri, Alessandra Aimi, A new framework for joint trajectory planning based on time-parameterized B-splines, Comput. Aided Des. (ISSN: 0010-4485) (2022) 103421, http://dx.doi.org/10.1016/j.cad.2022.103421.

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

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

[24] David Lutterkort, Jörg Peters, Smooth Paths in a Polygonal Channel, 1999, pp. 316–321, http://dx.doi.org/10.1145/304893.304985.

[25] David Lutterkort, Jörg Peters, Tight linear envelopes for splines, Numer. Math. 89 (2001) 735–748, http://dx.doi.org/10.1007/s002110100181.

[26] L.A. Piegl, W. Tiller, Curve interpolation with arbitrary end derivatives, Eng. Comput. (ISSN: 0177-0667) 16 (1) (2000) 73–79, http://dx.doi.org/10.1007/s003660050038.

[27] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, OSQP: An operator splitting solver for quadratic programs, Math. Program. Comput. 12 (4) (2020) 637–672, http://dx.doi.org/10.1007/s12532-020-00179-2.

[28] Yousef Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, 2003, http://dx.doi.org/10.1137/1.9780898718003.

[29] Bartolomeo Stellato, OSQP, 2023, URL https://osqp.org/.

[30] Jixiang Yang, Dingwei Li, Congcong Ye, Han Ding, An analytical C3 continuous tool path corner smoothing algorithm for 6R robot manipulator, Robot. Comput.-Integr. Manuf. (ISSN: 0736-5845) 64 (2020) 101947, http://dx.doi.org/10.1016/j.rcim.2020.101947.

[31] Zhang Yongbin, Taiyong Wang, Jingchuan Dong, Peng Peng, Yangfan Liu, Runji Ke, An analytical G3 continuous corner smoothing method with adaptive constraints adjustments for five-axis machine tool, Int. J. Adv. Manuf. Technol. 109 (2020) http://dx.doi.org/10.1007/s00170-020-05402-x.

[32] Jingfu Peng, Pengsheng Huang, Ye Ding, Han Ding, An analytical method for decoupled local smoothing of linear paths in industrial robots, Robot. Comput.-Integr. Manuf. (ISSN: 0736-5845) 72 (2021) 102193, http://dx.doi.org/10.1016/j.rcim.2021.102193.

[33] Burak Sencer, Eiji Shamoto, A curvature optimal sharp corner smoothing algorithm for high-speed feed motion generation of NC systems along linear tool paths, Int. J. Adv. Manuf. Technol. 76 (2014) 1977–1992, http://dx.doi.org/10.1007/s00170-014-6386-2.

[34] Les Piegl, Wayne Tiller, Software-engineering approach to degree elevation of B-spline curves, Comput. Aided Des. (ISSN: 00104485) 26 (1) (1994) 17–28, http://dx.doi.org/10.1016/0010-4485(94)90004-3.

[35] W. Tiller, Knot-removal algorithms for NURBS curves and surfaces, Comput. Aided Des. (ISSN: 00104485) 24 (8) (1992) 445–453, http://dx.doi.org/10.1016/0010-4485(92)90012-Y.

[36] Les Piegl, Wayne Tiller, Symbolic operators for NURBS, Comput. Aided Des. (ISSN: 00104485) 29 (5) (1997) 361–368, http://dx.doi.org/10.1016/S0010-4485(96)00074-7.

[37] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, fourth ed., MIT Press, ISBN: 9780262046305, 2022.

[38] Philip J. Schneider, David H. Eberly, Geometric Tools for Computer Graphics, in: The Morgan Kaufmann Series in Computer Graphics, Morgan Kaufmann, San Francisco, ISBN: 978-1-55860-594-7, 2003, http://dx.doi.org/10.1016/B978-155860594-7/50004-7.

[39] Christer Ericson, Real-Time Collision Detection, in: The Morgan Kaufmann Series in Interactive 3D Technology, Morgan Kaufmann, San Francisco, ISBN: 978-1-55860-732-3, 2005, pp. 1–6, http://dx.doi.org/10.1016/B978-1-55860-732-3.50006-1.

[40] Mark Berg, Otfried Cheong, Marc Kreveld, Mark Overmars, Computational Geometry, Algorithms and Applications, third ed., Springer Berlin, Heidelberg, ISBN: 978-3-540-77973-5, 2008, http://dx.doi.org/10.1007/978-3-540-77974-2.

[41] Franco P. Preparata, Michael Ian Shamos, Computational Geometry, An Introduction, first ed., Springer, New York, NY, ISBN: 978-1-4612-1098-6, 2012, http://dx.doi.org/10.1007/978-1-4612-1098-6.

[42] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, third ed., Cambridge University Press, USA, ISBN: 0521880688, 2007.

Marco Riboli is a Ph.D. candidate at the school of Industrial Engineering, University of Parma. His research interests include issues related to motion planning with kinematic and dynamic limits of robotic systems.

Matthieu Jaccard is a researcher at University of Applied Sciences and Arts of Southern Switzerland (SUPSI). He received his M.S. Industrial technologies with orientation in Energy systems and Electronics from the University of Applied Sciences and Arts of Western Switzerland (HES-SO). His research interests include developments in various fields related to electronics, mechatronics, and control.

Marco Silvestri worked in automation and industrial vision fields as developer and as head of R&D of the automated equipment manufacturer Tecnomec srl. Associate professor at University of Parma and docent-researcher at SUPSI. Principal investigator of several research projects in the fields of advanced automation and intelligent manufacturing. Serving as expert at Innosuisse – Swiss Innovation Agency.

Alessandra Aimi is Associate Professor in Numerical Analysis at the University of Parma. Her scientific interests range from BEM and BEM-FEM coupling for elliptic and hyperbolic equations to quadrature schemes for singular integrals and restriction matrices for symmetry invariant problems. Recently, she has worked both on B-spline based IgA-BEM and on spline QI projectors for log-singular integral equations.

Cesare Malara works as Software Director in Astes4. has a solid experience in the field of automation, industrial robotics and in the application and implementation of optimization algorithms for systems subject to constraints.