Preprint version of the IEEE 21st International Conference on Advanced Robotics (ICAR), Abu Dhabi, UAE 5 – 8 Dec. 2023

Collision-free Volume Estimation Algorithm for Robot Motion Deformation

Nicola Miotto[1,*], Alberto Gottardi[1,2,*,†], Nicola Castaman[3], Emanuele Menegatti[1]

***Abstract*— The collaborative transport of objects between humans and robots is one of the main areas of focus in physical Human-Robot Interaction (pHRI). Ensuring the operator's safety and maintaining collision-free motion of the robot during transportation are crucial challenges in this context. Consider a collaborative co-manipulation scenario where the operator modifies the trajectory being executed by the robot. In such cases, the robot may deviate from its previously calculated path, potentially resulting in collisions.**

In this work, we propose a method to estimate the maximum collision-free volume around the path of the robot. This volume represents the permissible deviation introduced by the human worker while ensuring that no collisions occur. To evaluate the effectiveness of the proposed algorithm, we test it in a real industrial scenario.

***Index Terms*— Physical Human-Robot Interaction, Deformation Boundaries, Collision-free Volume Estimation**

I. INTRODUCTION

Human-robot co-manipulation is one of the most challenging tasks in the physical Human-Robot Interaction (pHRI), especially in the industrial fields. The advantages of physical interaction between humans and robots include increased task complexity, improved handling capabilities, flexibility, adaptability and the creation of a collaborative and trusting working environment. These benefits contribute to more efficient and effective collaborative operations and improve performance and productivity. Collaborative object transportation is one of the scenarios that benefit most from this interaction. In that scenario, humans and robots can collaborate in real-time, responding to unexpected environmental changes or variations. Humans can provide on-the-fly adjustments, ensuring the transport process remains efficient and effective even in unpredictable situations. This flexibility allows for the seamless integration of human and robot capabilities, overcoming challenges that arise during transportation: for example, the user can intervene to prevent accidental damage to the object or material being transported with the robot. Specifically, one possible interaction mode for collaborative transport is the following: the robot is moving along a pre-computed path and the human follows it. In this situation, the operator can monitor and follow the robot's movement and intervene when necessary by modifying the robot's path while at the same time ensuring compliance with safety protocols and preventing potential accidents such as collisions with other objects in the environment.

This paper proposes an algorithm for estimating a collision-free deformation volume within which the user can modify the robot's movement without causing collisions with static objects in the environment. More specifically, the proposed method is based on a precomputed collision-free path for the robot, called *nominal path*. That path is computed by a path planner, and then the maximum collision-free deformation volume is estimated along this path. These boundaries define the limits within which joint values can be adjusted during movement to account for expected deviations from the *nominal path*. These boundaries must prevent collisions while allowing paths resembling the nominal one. The aim of this research is to improve collaboration between humans and robots during movements by establishing a subspace within the configuration space that allows deviations from the nominal path, thereby enhancing the robot's adaptability to the human operator.

The main contributions of this paper are as follows:

- An algorithm for estimating the collision-free volume of deformation that allows the user to modify the robot's movement without causing collisions with static objects.
- A representation method of the volume through boundaries expressed in configuration space.

The method proposed in this paper has been tested in two simulated environments within the EU project DrapeBot[1], which aims to develop a Human-Robot Collaborative system capable of assisting an operator working on the draping of carbon fiber parts. Among the various assistance tasks is the collaborative transport of the carbon ply from the pick table to the mould for the subsequent draping.

II. RELATED WORKS

In recent years, the estimation of a collision-free volume has been investigated. A related approach is based on the concept of swept-volume. The swept-volume is defined as the volume generated from the sweeping of an object [1]. This approach has been used to generate more robust trajectories [2] or to enhance the collision detection module [3]. De Mont-Marin et al. in [2] formulated the minimum swept-volume distance and its geodetics and used them to improve the Rapidly-exploring Random Tree (RRT) motion planning algorithm. Another work that used the swept-volume in the motion planner is [4]. The authors present a technique to generate a motion sequence for a mobile manipulator based

*Authors equally contributed to the work.

† Corresponding Author

[1] Intelligent Autonomous System Lab, Department of Information Engineering, University of Padova, 35131 Padua, Italy. gottardial, emg@dei.unipd.it

[2] IT+Robotics srl, 36100 Vicenza, Italy.

[3] Professional Industrial consultant, 36061 Bassano del Grappa, Italy.

[1] https://www.drapebot.eu/

on a desired goal-hand pose. This involved recording and selecting suitable motion sequences and associated swept-volumes, allowing for real-time modification of motion sequences using sequential quadratic programming to avoid collisions with dynamically discovered obstacles. Regarding the improvements of the collision detection module, the swept-volume is used to reduce the collision checking phase during the computation of a footstep planning for humanoid robots [5] or to represent the robot's body and perform the pairwise checking for collisions [3]. More recently, Chang et al. in [6] proposed a deep learning network approach to estimate the swept-volume. Baxter et al. in [7] also proposed a deep learning-based approach to predict the swept-volume using voxel grids. All these works share similarities in their use of the swept-volume to enhance the collision detection module or the final trajectory computed by a motion planner algorithm like (RRT). However, our method differs in that we don't want to compute a trajectory with the swept-volume. Instead, we based our approach on a precomputed collision-free path and aim to estimate the collision-free space around it. In other words, we proposed a method to expand the swept-volume and dynamically deform the robot's motion ensuring the safety within the computed volume. Another difference with our approach lies in the method used to compute the space region. Several algorithms in the literature focused on generating approximations using occupation grid-based, convex polyhedra-based and boundary-based methods. Boundary-based methods [8], [9] involve the computation of an outer boundary surface for the swept volume. This process entails creating surfaces representing the sweeping and rotational motion of object surfaces and then determining the surface of their combination. These methods primarily concentrate on calculating the boundaries of the swept volume for moving solids, but they tend to be slow, particularly when dealing with articulated robots. In occupancy grid-based methods [10], [11], the space is split into voxels. These methods then record which segments of space are touched by the robot as it moves in steps from one configuration to the next configuration. Finally, convex polyhedra-based methods involve approximating robot bodies with convex hulls [12], [13]. Typically, these methods break down the model into smaller collections of convex polyhedrons, presenting its own difficulties. As the robot transitions between configurations, extra convex hulls are incorporated, and the swept volume is determined by the union of these convex hulls [13]. In the proposed approach, we did not use this representation of the space because they are expensive in terms of computational resources and time. Instead, we formulated an approach based on Height-Map, i.e. a 2D map and the Height-function used to represent the third dimension. This choice was made in order to save computational resources and allow us to be very fast when estimating the deformation volume.

III. METHOD

As observed in the previous sections, managing the human presence in the context of pHRI raises certain challenges, as the human operator is inherently unpredictable. In addition, classical approaches such as swept-volume are used to create a robust *nominal path*. Nevertheless, they cannot ensure the robot does not collide with its surroundings when the operator adjusts and modifies the path.

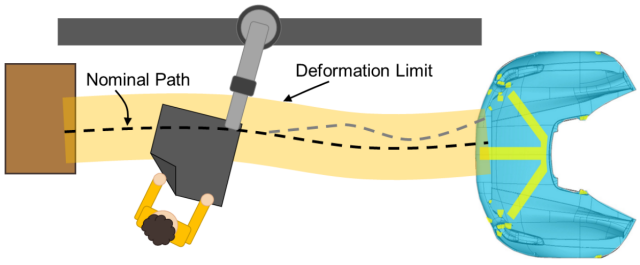


Fig. 1: Example of deformation boundaries limits in a 2D representation.

This section presents the formulation of the maximum boundaries estimation to create a collision-free volume around the *nominal path*. Starting from the *nominal path* obtained by a path planner and a model of the environment, we defined a collision-free volume around such path. The algorithm can take as input any legitimate path and return a corresponding free volume. The model of the environment, i.e. the position and dimension of the objects, can be provided by camera systems or other vision system modules. In this way, our approach is independent and adaptable to different environments. The collision-free volume represents a portion of the space where the robot can modify the *nominal path* in response to the human motion without colliding with the other objects presented in the workcell, and it is represented through boundary limits in the joint space. Fig. 1 depicts a 2D image of the boundary deformation limits around the *nominal path* to provide an idea of the proposed approach's potential.

The proposed method is entirely computed offline in the joint space before the motion execution. The overall concept is to generate a volume which synthesizes the whole workspace but is left free in the space surrounding the *nominal path* and near the goal. Then, the boundaries defining the final intervals are found by observing for which joints' value the robot collides with such volume.

Given a collision-free *nominal path* $\boldsymbol{Q}_{nominal} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_g]$ where $\boldsymbol{q}_1$ is the initial configuration and $\boldsymbol{q}_g$ is the goal configuration. Let $\boldsymbol{q}_i = [q_1, \ldots, q_k]$, where k is the number of joints of the manipulator. The *nominal path* is computed using the RRT-Connect path planner algorithm [14]. The path planner provides a path where singularity never take place. In addition, the step 3 of the algorithm checks that no singularities can occur. The method consists of the following main steps (Algorithm 1):

1) *Object Categorization*: division of the obstacles in the workspace into *goal*, *trail* and *no-goal* sets. The first represents obstacles around the target position of the robot; the second set encapsulates the information regarding the surrounding of the *nominal path* (robot's links included), and the last set the remaining

Algorithm 1 Volume estimation

Input: $\boldsymbol{q}_1, \boldsymbol{q}_g, \mathcal{O}, \delta$
Output: $\mathcal{B}$
1: $\mathcal{B} = []$
2: $\boldsymbol{Q}_{nominal} \leftarrow RRT - Connect(\boldsymbol{q}_1, \boldsymbol{q}_g)$
3: $\mathcal{O}_{ng}, \mathcal{O}_t, \mathcal{O}_g \leftarrow ObjectCategorization(\mathcal{O})$
4: $\mathcal{H} \leftarrow instantiateHeightMap(\mathcal{O}_{ng}, \mathcal{O}_t, \mathcal{O}_g)$
5: **for all** $\boldsymbol{q}_i \in \boldsymbol{Q}_{nominal}$ where $i \in [1, g]$ **do**
6: $\quad \boldsymbol{b}_i \leftarrow getJointsBounds(\boldsymbol{q}_i, \mathcal{H}, \delta)$
7: $\quad \mathcal{B}$.append($\boldsymbol{b}_i$)
8: **end for**
9: return $\mathcal{B}$

obstacles. Formally, let $\mathcal{O}$ the obstacles set, $\mathcal{O} = \mathcal{O}_{ng} \cup \mathcal{O}_t \cup \mathcal{O}_g$, where $\mathcal{O}_{ng}$ is the *no-goal* obstacles set, $\mathcal{O}_t$ is the *trail* obstacles set and $\mathcal{O}_g$ is the *goal* obstacles set. Note that $\mathcal{O}_{ng} \cap \mathcal{O}_g \cap \mathcal{O}_t = \varnothing$

2) *Volume Generation*: generation of the volume $\mathcal{V}$ through a Height-map $\mathcal{H}$ and Height-function $h(c)$ in the 3D cartesian workspace $\mathcal{W} \in \mathbb{R}^3$.
3) *Joint Space Exploration*: transforming volume $\mathcal{V}$ from $\mathcal{W}$ to joint space $\mathcal{C}$ and express it as joints' boundaries limits $\mathcal{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_g]$ where $\boldsymbol{b}_i = [b_1, \ldots, b_k]_i$, $\forall i \in [1, g]$. In other words, $\boldsymbol{b}_i$ is the boundary for $\boldsymbol{q}_i$, and $b_j =< b_l, b_u >_j$, $\forall j \in [1, \ldots, k]$ is the tuple that represents the lower and upper joint's boundary.

Note that $getJointsBounds(\boldsymbol{q}_i, \mathcal{H}, \delta)$ (Alg1 - line 6) is a function that calculates $\boldsymbol{b}_i$ for each $\boldsymbol{q}_i$. The computation is done by iteratively calling the function *Joint Bounding* (Algorithm 2) $\forall q_j \in \boldsymbol{q}_i, j = [1, \ldots, k]$.

A. Objects Categorization

This step aims to explicitly define three different categories of the objects in the working environment since, as it will be shown in the next step, different objects influence the boundaries differently. Starting from the obstacles set $\mathcal{O}$, it is necessary to divide $\mathcal{O}$ into the three different subsets $\mathcal{O}_{ng}, \mathcal{O}_g, \mathcal{O}_t$ described above. The categorization uses the distance between the obstacles and the *nominal path* (start and target pose included) along the path direction.

B. Volume Generation

This phase aims to combine the obstacles and the spatial areas around the *nominal path* in one entity to gauge the robot's moving limits. This phase requires the definition of a Height-map $\mathcal{H}$ and a related Height-function $h(c)$. The resulting volume $\mathcal{V}$ is represented through $\mathcal{H}$ as a set of collision boxes.

1) Height Map: The volume expands using a Height-map roughly representing the robot's workspace. The Height-map was preferred over a 3D structure like an Octree [15] or Octomap [16] because it allows us to save computational resources by eliminating one spatial dimension. However, the combination of Height-map and Height-function allows a representation of volume along all directions and provides the necessary robustness to changes introduced during movement. In contrast to using a Quadtree [17], the Height-map was chosen due to the potential occurrence of large empty areas in the map when using an exploratory approach. Consequently, when computing the heights of individual cells, notable discontinuities may arise between adjacent volumes. These discontinuities can give rise to unpredictable behaviours when determining the boundaries of the final joints. The height maps are typically used for reconstruction and mapping purposes, but in this work, they are used for generative purposes.

The Height-map $\mathcal{H}$ is defined by its origin $O_h \in \mathbb{R}^3$, a pair $(x_{dim},\ y_{dim})$ of the dimension of its sides and a maximum number of cells R. The side of a cell is defined as $side = \sqrt{\frac{x_{dim} y_{dim}}{R}}$; while the number of rows and columns are, respectively, $\#row = m = \frac{x_{dim}}{side}$ and $\#column = n = \frac{y_{dim}}{side}$. Finally, to instantiate the volume, it's necessary to decide how much to grow up the volume in each cell of the Height-map: the following Height function is used.

2) Height Function: The aim of this function is to set, cell by cell, the heights of the final volume. Since the volume will be used for a co-manipulation transport, where the human operator will modify the path, and at the same time, he has to guarantee the precision in the target pose, we have formulated a function that would make the volume grow higher when closer to the *non-goal* obstacles and lower when near the *goal* obstacles. By employing this approach, we enable the joint boundaries to exclude the areas surrounding the obstacles, while also ensuring they do not become excessively constrained around the nominal values or the target region. Let d_t, d_g and d_{ng} be the distance from the center of the cell c to the *trail*, *goal* and *no-goal* obstacle respectively. Let also $d_{\cdot,LB}, d_{\cdot,UB}$ be the parameters defining the floor and the top of the correspondent height-function component $\Delta_\cdot$. Thus, the mathematical formulation for the Height-function is:

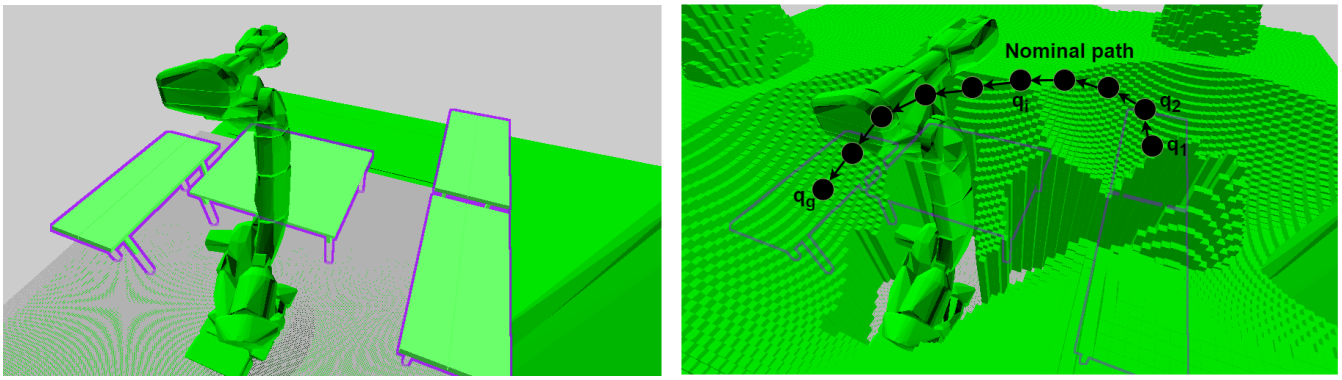
$$h(c) = A_t(c) h_{max} \left(\Delta_t^{\alpha_t} + \Delta_g^{\alpha_g} + \Delta_{ng}^{\alpha_{ng}} \right) \tag{1}$$

where $A_t(c)$ is the activation function that sets the volume corresponding to a cell c too close to the *trail* to zero, h_{max} sets the highest possible increment a single component of the function can contribute, the exponentials α represents the tightness of the boundaries and $\Delta_\cdot$ are defined as follows:

$$\Delta_t = \frac{d_t - d_{t,LB}}{d_{t,UB} - d_{t,LB}}$$
$$\Delta_g = \frac{d_g - d_{g,LB}}{d_{g,UB} - d_{g,LB}}$$
$$\Delta_{ng} = \frac{d_{ng,UB} - d_{ng}}{d_{ng,UB}}$$

In practice, for each cell c, a collision box with a square base (side $side$ and height $h(c)$) is instantiated. Fig. 2 depicts an example, visualized using the software PhysX [2]. The free space is the volume $\mathcal{V}$ computed for the nominal path, while the green boxes represent the unavailable space. The volume is truncated across the robot's nominal poses and over the goal obstacle to the left. As expected, volume peaks

[2]PhysX physics engine.
Available: www.geforce.com/hardware/technology/physx



(a) (b)

Fig. 2: Visualization of a scene in PhysX before (a) and during (b) volume generation.

are presented over the obstacles. Once the volume $\mathcal{V}$ is generated in the cartesian workspace $\mathcal{W}$, it is necessary to establish the boundaries in the joint space around the nominal values. Therefore, we have formulated an iterative approach to convert the volume from $\mathcal{W}$ to $\mathcal{C}$ as boundary limits for each joint.

C. Joint Space Exploration

The volume is used to apply "pressure" on the joints to find an upper and lower bounding interval for each joint. Upper and lower bounds are computed separately with the same procedure (Algorithm 2). The algorithm starts by setting the joint j under analysis at its absolute bound B (Alg2 - line 4), while other joints are set at their input nominal value and kept at these values in all the steps. First, the algorithm checks if a collision occurs between the robot and the volume generated in the previous section (Alg2 - line 5). If not, the algorithm halts and returns the absolute bound as a result (Alg2 - lines 6, 7). Otherwise, the joint is set to the midpoint between the input and absolute boundary values (Alg2 - lines 9, 10). After that, the algorithm enters a loop, which is depicted in Fig. 3. At each iteration, a collision check is performed as before (Alg2 - lines 12). If there's a collision, the joint is set to the midpoint between the highest collision-free value and the current joint value (Alg2 - lines 13). If there's no collision, the joint is set to the midpoint between the current and lowest colliding values (Alg2 - lines 15). The loop halts once the distance between the highest collision-free and lowest colliding values is less than a given threshold δ (Alg2 - line 11) and returns the highest collision-free value as the resultant boundary b^* (Alg2 - line 21).

A mathematical explanation of the approach follows. Let δ be the threshold parameter and d be the distance between the best value of the bound b^* and the current absolute value. At the beginning of the loop, d is the distance between the highest collision-free value and the absolute boundary B. At each step of the loop, d is cut in half. Therefore the exit condition can be written as $\frac{d}{2^s} < \delta$, where s is the number of steps inside the loop. So, the loop is exited when $s > log_2 \frac{d}{\delta}$. Therefore the whole algorithm 2 takes $s = 1$ steps if it doesn't enter the loop or $s = 1 + \lceil log_2 \frac{d}{\delta} \rceil$ if it enters the loop.

Algorithm 2 Joint Bounding

Input: $\boldsymbol{q}_i, j, B, \delta, \mathcal{H}$
Output: b^*
1: $\boldsymbol{q} \leftarrow copy(\boldsymbol{q}_i)$
2: $b^* \leftarrow \boldsymbol{q}[j]$
3: $x, y \leftarrow B$
4: $\boldsymbol{q}[j] \leftarrow B$
5: **if** !collision($\boldsymbol{q}, \mathcal{H}$) **then**
6: $b^* \leftarrow B$
7: return b^*
8: **end if**
9: $x \leftarrow (b^* + y)/2$
10: $d \leftarrow \|(x - b^*)\|$
11: **while** $d > \delta$ **do**
12: **if** collision($\boldsymbol{q}, \mathcal{H}$) **then**
13: $y \leftarrow x$
14: **else**
15: $b^* \leftarrow x$
16: **end if**
17: $x \leftarrow (b^* + y)/2$
18: $\boldsymbol{q}[j] \leftarrow x$
19: $d \leftarrow \|(x - b^*)\|$
20: **end while**
21: return b^*

D. Theoretical Time Analysis

This section will analyse the proposed algorithm from a computational perspective. Let g be the number of configurations in $\boldsymbol{Q}_{nominal}$, and $R = m \times n$ be the number of cells of $\mathcal{H}$. Let the goal position be $\boldsymbol{p_g} = [x_g, y_g, z_g]^T \in \mathcal{W}$ and the robot has k joints. Finally, let $o = |\mathcal{O}|$ the number of obstacles. Since the *Object Categorization* step subdivides and classifies obstacles, the algorithm's computation time complexity is linear. Regarding step 2, the volume generation needs to evaluate the Height-function of each cell. For each cell, it requires $1 + g + o$ iterations. As before, the computational time is linear. Finally, the *Joint Space Exploration* requires 1 or $1 + \lceil log_2 \frac{d}{\delta} \rceil$ steps. This process contains the most expensive operation in terms of computational time: collision checking. This single operation requires gR steps

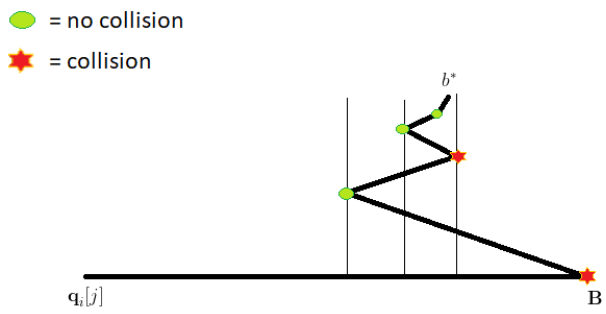


Fig. 3: Graphical description of the upper boundary searching algorithm in the joint space.

because each robot link is confronted with each cell of the volume $\mathcal{V}$. Therefore the total time T of this step:

$$gkR \leq T \leq gkR\left(1 + log_2 \frac{d}{\delta}\right) \tag{2}$$

Therefore, the discretization of the Height-map is the only main parameter under control to manage the computational time of the proposed method. This relation, represented by R, appears to be linear, as well as the other relations with the not controllable parameters like k (since the robot is given), o (since also the environment is known), and g. Based on this analysis, the expected computational time is smaller than what could have been expected with the method described in Sec. II or the cylindrical algebraic decomposition [18], which has a double exponential complexity. Notice that also g has a linear impact on the computational time.

IV. EXPERIMENTS

A. Case Study

The two simulated scenarios are from the automotive (Fig. 4a) and aerospace (Fig. 4b) industrial sectors. In the first case, the 6-DoF ABB IRB 6700-205/2.80 is used, while, in the aerospace use case, the 6-DoF KUKA Quantec KR210 R3100 robot is mounted on an eight-meter-long linear axis. Thus, we have decided to consider the robot a 7-DoF robot, with the first being a prismatic joint, i.e., the linear axis. To evaluate the proposed approach, the *Boundary Confidence* performance indicator is defined. The indicator describes the quality of the volume computed around the *nominal path* and is useful for the deformation algorithm used during the robot's motion. The metrics identified for the evaluation of the *Boundary Confidence* are:

- Time to compute the boundaries.
- Confidence Factor (CF): the CF has been computed as the average number of collision-less random joints configuration encountered in a time window. The CF aims to express the quality of the boundaries produced.

Three different start and goal states were selected for each use case to compare the performance. Since the path planner (RRT-Connect) is a sampling-based algorithm, the paths generated from the same initial and final states are not always identical. This is an advantage for testing the robustness and generalisation of the deformation volume estimation algorithm. Since the resolution of the Height-map is a crucial parameter to set, a performance analysis was done and a set R of possible values was selected. The set values are $R = \{100, 500, 1000, 2000, 5000, 10000, 15000, 20000\}$: these values represent the number of the cell on the Height-map $\mathcal{H}$. As mentioned above, the boundaries algorithm was investigated into three different paths, and each value $r \in R$ has been checked over the 20 trials for each path. So, the average computing time is used for the evaluation. As the *nominal path* can be composed of linearly connected configurations, we arbitrarily divided the path into $N_{tw} = 30$ time windows and $N_m = 10000$ random joints' configurations, with values within the time window's boundaries, are tested for collision in each time window. Then, indicating with $J(i)$ the joint boundaries within the i-th time window, the mathematical formulation of CF is:

$$CF = \frac{1}{N_{tw} N_m} \sum_{i=1}^{N_{tw}} \sum_{m=i}^{N_m} c_{J(i)} \tag{3}$$

where $c_{J(i)}$ is the function checking whether the collision of the random set in $J(i)$ occurs. In other words:

$$c_{J(i)} = \begin{cases} 1 & \text{if no-collision,} \\ 0 & \text{if collision} \end{cases} \tag{4}$$

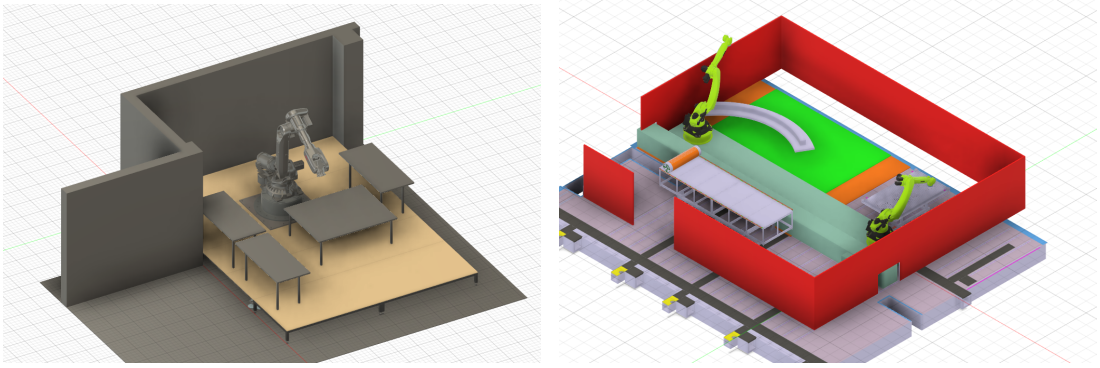
Taking equation 1 as a reference and after a fine-tuning phase, the following values were used for the parameters:

- $\alpha_t = \alpha_g = \alpha_{ng} = 1$
- $d_{t,LB} = 350\ mm$, $d_{t,UB} = 1500\ mm$
- $d_{g,LB} = 100\ mm$, $d_{g,UB} = 2000\ mm$
- $d_{ng,UB} = 500\ mm$
- $h_{max} = 1500\ mm$
- the activation function $A_t(c)$ set zero height in all the cells more distant than $2800\ mm$ for the automotive case and $3000\ mm$ for aerospace.

The activation function's parameter is set at a value such that no box is instantiated outside the robot's reachability. The exponents α_t, α_g and α_{ng} are set to 1 as it was chosen to prefer tighter boundaries. Higher exponents would make the volume scale up smoother, although these exponents have less influence on the final boundaries than the distances' UBs and LBs. Finally, the experiments were performed on a *Windows 10* pc with an Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz and 8 GB of RAM.

B. Results

1) Automotive Use Case: As can be seen from Table I, the computational time increases linearly with respect to the resolution value in all three paths. Thus, these results confirm the analysis done in Section III-D. The CF shows linear growth with respect to resolution, although less sharp than with respect to computing time. Analysing all three paths, it can be established that the best performance in terms of calculation time and confidence factor occurs with a resolution equal to 1000. In fact, the computational times are always less than $200ms$ and the CF greater than 99%.



(a) (b)

Fig. 4: Automotive use case (a) and aerospace use case (b).

Resolution	Path 1		Path 2		Path 3	
	Time [ms]	CF [%]	Time [ms]	CF [%]	Time [ms]	CF [%]
100	82	95.3839	71	93.4484	71	99.9484
500	121	98.9903	151	99.1484	119	99.9999
1000	170	99.5000	191	99.3742	173	99.9999
2000	231	99.5903	246	99.7742	202	99.9999
5000	387	99.8548	411	99.7742	365	99.9999
10000	609	99.9871	717	99.9613	598	99.9999
15000	1000	99.9226	1329	99.9161	837	99.9999
20000	1149	99.8258	1584	99.8000	1224	99.9999

TABLE I: Automotive case results.

2) Aerospace Use Case: The results obtained in this use case are similar to the automotive case: the CF values are constantly greater than 90% (except for the first three resolution values for *Path 2*), and the linear grown behaviour remains. The same linear behaviour is also present in the computational time. In this case, the resolution 2000 has the best results ($time = 562ms$ and $CF = 93.20\%$) for *Path 1*, but not for *Path 2* and *Path 3*. In *Path 2*, the highest CF is 98.19% with a resolution equal to 20000 and $2434ms$ as computing time, while the best computational time is $498ms$ with a resolution equal to 100 and a corresponding CF amount to 88.1%. Thus, analyzing the performance makes it possible to affirm that the right trade-off between the two paths is the one with a 5000 resolution because the computing time is less than $1s$ and the CF is greater than 95%.

Resolution	Path 1		Path 2		Path 3	
	Time [ms]	CF [%]	Time [ms]	CF [%]	Time [ms]	CF [%]
100	398	90.5355	498	88.1	884	93.7387
500	453	92.5355	670	87.1032	538	99.8774
1000	504	91.7452	552	87.8	557	99.9516
2000	562	93.2097	723	92.6613	640	99.9999
5000	772	92.7677	975	95.9032	984	99.9999
10000	1124	92.7194	1684	97.0613	1222	99.9999
15000	1452	92.8194	2056	98.1613	1817	99.9999
20000	1657	92.9226	2434	98.1935	2419	99.9999

TABLE II: Aerospace case results.

Compared the Table I with Table II, it is possible to notice that the time results are influenced by the workcell. This influence is due to the different sizes of the two workcells: in the automotive case, the workspace is smaller than in the aerospace case. Nevertheless, the results confirmed the proposed approach's potential to estimate a collision-free volume around a nominal path which can be very useful in applications requiring online deformation of the robot path, such as human-robot collaborative transport.

V. CONCLUSIONS

In this paper, we proposed an approach to estimate the collision-free volume of a robot's motion deformation. In fact, during collaborative transport with human and robot, the human operator could deform the current motion of the robot. This motion adjustment can, however, lead to collisions with objects in the environment. Therefore, the algorithm proposed in this work aims to be a supporting tool in this type of scenario. Starting from the *nominal path*, the algorithm analyzes the workspace and uses a Height-map and a Height-function to estimate the collision-free volume in the cartesian space. After that, the algorithm converts the volume from the cartesian space to the joint space and expresses it through boundary limits at joints along the *nominal path*. The proposed approach has been validated in two simulated industrial scenarios from the DrapeBot project. The computational time and the Confidence Factor are the two metrics used to evaluate the results. In both use cases used for the evaluation, the computing time is very fast (less than $200ms$) with a 99% of confidence for the automotive case and less than $1s$ with the 95% of confidence. Therefore, the results demonstrate the applicability and effectiveness of the approach. An extension of the algorithm here described could be to adapt it to an unknown environment, providing a detection system to build a dynamic model. Consequentially, it could be possible to change the volume at runtime due to a changing environment if the delay between one instance of the model and the next is properly managed.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 101006732, "DrapeBot – collaborative draping of carbon fiber parts".

REFERENCES

[1] T.-J. Huang, "Collision distance detection based on swept volume strategy for optimal motion plan," *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 3, no. 3, pp. 212–223, 2009.

[2] Y. de Mont-Marin, J. Ponce, and J.-P. Laumond, "A minimum swept-volume metric structure for configuration space," *arXiv preprint arXiv:2211.11811*, 2022.

[3] H. Täubig, B. Bäuml, and U. Frese, "Real-time swept volume and distance computation for self collision detection," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1585–1592.

[4] T. Iwasaki, Y. Takase, S. Arnold, K. Takeshita, and K. Yamazaki, "Online motion planning based on swept volume search with replanning using sequential quadratic programming," *Advanced Robotics*, pp. 1–14, 2023.

[5] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, 2011.

[6] H.-T. L. Chiang, J. E. Baxter, S. Sugaya, M. R. Yousefi, A. Faust, and L. Tapia, "Fast deep swept volume estimator," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1068–1086, 2021.

[7] J. Baxter, M. R. Yousefi, S. Sugaya, M. Morales, and L. Tapia, "Deep prediction of swept volume geometries: Robots and resolutions," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6665–6672.

[8] M. Campen and L. Kobbelt, "Polygonal boundary evaluation of minkowski sums and swept volumes," in *Computer Graphics Forum*, vol. 29, no. 5. Wiley Online Library, 2010, pp. 1613–1622.

[9] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha, "Fast swept volume approximation of complex polyhedral models," in *Proceedings of the eighth ACM symposium on Solid modeling and applications*, 2003, pp. 11–22.

[10] A. Von Dziegielewski, M. Hemmer, and E. Schömer, "High precision conservative surface mesh generation for swept volumes," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 183–191, 2013.

[11] J. C. Himmelstein, E. Ferre, and J.-P. Laumond, "Swept volume approximation of polygon soups," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 177–183, 2009.

[12] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3d mesh approximate convex decomposition," in *2009 16th IEEE international conference on image processing (ICIP)*. IEEE, 2009, pp. 3501–3504.

[13] A. Gaschler, R. Petrick, T. Kröger, O. Khatib, and A. Knoll, "Robot task and motion planning with sets of convex polyhedra," in *Robotics: Science and Systems (RSS) Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[14] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[15] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[16] C. Wang, L. Meng, T. Li, C. W. De Silva, and M. Q.-H. Meng, "Towards autonomous exploration with information potential field in 3d environments," in *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, 2017, pp. 340–345.

[17] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[18] R. Jhaa, D. Chablat, L. Baronc, F. Rouillier, and G. Moroz, "Workspace, joint space and singularities of a family of delta-like robot," *Elsevier*, 2018.