

IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

Robot Cell Modeling via Exploratory Robot Motions

Gaetano Meli and Niels Dehio

Abstract—Generating a collision-free robot motion is crucial for safe applications in real-world settings. This requires an accurate model of all obstacle shapes within the constrained robot cell, which is particularly challenging and time-consuming. The difficulty is heightened in flexible production lines, where the environment model must be updated each time the robot cell is modified. Furthermore, sensor-based methods often necessitate costly hardware and calibration procedures, and can be influenced by environmental factors (e.g., light conditions or reflections). To address these challenges, we present a novel data-driven approach to modeling a cluttered workspace, leveraging solely the robot's internal joint encoders to capture exploratory motions. By computing the corresponding swept volume, we generate a (conservative) mesh of the environment that is subsequently used for collision checking within established path planning and control methods. Our method significantly reduces the complexity and cost of classical environment modeling by removing the need for CAD files and external sensors. We validate the approach with the KUKA LBR iisy collaborative robot in a pick-and-place scenario. In less than three minutes of exploratory robot motions and less than four additional minutes of computation time, we obtain an accurate model that enables collision-free motions. Our approach is intuitive, easy-to-use, making it accessible to users without specialized technical knowledge. It is applicable to all types of industrial robots or cobots.

Index Terms—Collision Avoidance, Physical Human-Robot Interaction, Software Tools for Robot Programming.

I. INTRODUCTION

AN accurate environment model is paramount for successfully deploying and operating robot systems without compromising hardware integrity. The process of environment modeling involves creating a (digital) representation of the physical world. Typically, it encompasses data acquisition through (expensive) sensors [1] and integrating this data into coherent models [2], [3]. The output of this process may be a dense point cloud, a 3D map, or 3D meshes [4], [5]. Alternatively, objects in the robot cell are modeled separately through Computer-Aided Design (CAD) files, incorporating shape and position information. This approach can yield imprecise results due to the sim-to-real gap. Moreover, necessary data are often unavailable. The challenge intensifies in robot cells frequently modified to meet changing production needs. Based on our experience, currently, many enterprises avoid modeling the robot surroundings due to the time-consuming, expensive, and complex nature of the task. However, this hinders the implementation of applications involving autonomous robots that generate collision-free paths and adapt motions in

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

The authors are with KUKA Deutschland GmbH, Augsburg, Germany.

©2026 IEEE

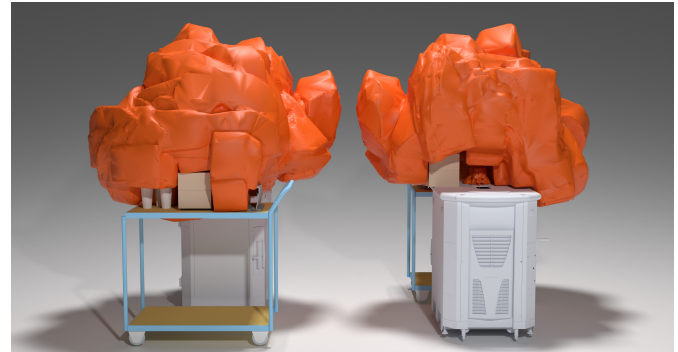


Fig. 1: Swept volume (in orange) of a KUKA LBR iisy robot (left: front view, right: back view). The exploration of the free workspace has been performed through hand guidance with a cube-shaped exploration tool.

real-time. Instead, operators program sub-optimal trajectories offline, leading to inefficiencies in execution.

Our main contribution is a novel data-driven approach to modeling a static constrained robot cell using exploratory robot motions, relying *solely* on the integrated joint encoders. Based on the collected data, we compute the robot swept volume, a non-convex hull represented as a 3D mesh that encapsulates the explored obstacle-free space, focusing on fixed obstacles. This mesh represents a conservative approximation of the accessible space. It can be utilized to verify if a particular robot configuration is safe. Hence, it can be integrated into state-of-the-art frameworks to enable automatic, collision-free motion planning and control [6], [7]. The assumption of a static environment is valid for many industrial applications, such as welding, painting, gluing, milling, deburring, and inspection. Our approach offers a compelling alternative to CAD-based modeling by eliminating sim-to-real gaps and reducing the need for complex modeling tasks. Moreover, it is fast, cost-effective, and straightforward – making it well-suited for supporting flexible production lines in modern Industry 4.0 scenarios. We validate the effectiveness of our approach in a pick-and-place scenario with the KUKA LBR iisy (see Fig. 1), highlighting the intuitive, user-friendly interface that does not require advanced technical knowledge in robotics. Note that the approach is robot-agnostic, making it applicable to both collaborative and industrial robots.

As a secondary contribution, we propose integrating an *exploration tool* that simplifies and accelerates the exploration phase, while reducing computational load. This tool is inexpensive as it contains no electronics. When combined with a commercially available tool change system, multiple

IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

exploration tools with different shapes can be easily used, without significantly extending the exploration process.

A demonstration video of our work is provided as supplementary material and is available at <https://youtu.be/p5DA41EtNUA>.

In the following, Sec. II presents the state of the art related to environment modeling and swept volume computation. Sec. III describes our approach to obtaining a representation of the obstacle-free space within the constrained robot cell. It also addresses the limitations of our method and potential resolution strategies to overcome them. Experimental results are reported in Sec. IV, while Sec. V concludes.

II. RELATED WORK

At its core, our approach to *environment modeling* is based on the *swept volume* of an exploratory robot motion. This section reviews the current state of the art in these two areas.

A. Environment Modeling

To ensure autonomous robots can safely execute tasks and avoid unintended collisions with their surroundings, an accurate environment model is essential. *Simultaneous Localization and Mapping* (SLAM) is a well-established paradigm that allows navigating through an unknown environment, while simultaneously localizing the robot pose thanks to onboard sensors such as cameras, laser scanners, GPSs, sonars, or LiDARs. The algorithm proposed in [1] has proven to be robust, accurate, and flexible in many applications with various sensor setups. However, it faces challenges maintaining high performance within sparsely textured and dynamic environments. Remarkable progress in camera localization and map reconstruction has been achieved by integrating deep learning techniques, considerably improving the underlying feature extraction [2], [3]. In recent years, there has been a growing need for continuous surface modeling and finding solutions for occlusions and sparse observations. This has increased research interest in Neural Radiance Fields (NeRF) [4] and 3D Gaussian Splatting (3DGS) [5]. These methods can produce 3D meshes from dense and compact environment maps. However, their practical applications may be hindered by limitations in real-time processing, hardware demands, and training duration. In addition to these automatic methods, CAD models describe the geometry of individual objects in a scene. They are also used to visualize, simulate and optimize large production line processes in 3D, demanding significant technical expertise. Although this approach can provide high precision and control when performed accurately, it may suffer from sim-to-real-gaps and is time-consuming.

B. Swept Volume

Swept volume (SV) refers to the three-dimensional space encompassing all points that a rigid object motion occupies. This concept is now widely utilized in several application fields, including numerically controlled machining verification (e.g., for a milling process), modeling of complex solids, robot reachable and dexterous workspace analysis, collision detection/avoidance, and ergonomics.

Abdel-Malek et al. [8] compared several methods for SV computation. An *explicit* representation, based on the geometric properties of the moving object, is typically obtained via voxel grid approximations or by using a triangle mesh to approximate the boundary of the SV. This approach struggles to generalize to all motions and object types, and the error, closely tied to computational power, is difficult to control. Alternatively, an *implicit* representation describes a mathematical function determining whether a point is inside or outside the SV. Even though the mathematical formulation is straightforward, the SV computation may result in a relevant computational load and provide sub-optimal solutions. Such numerical issues are avoided by the *stamping* method, which samples the object's motion in space and time. The accuracy of the final result heavily depends on the complexity of the object's motion and on the sampling time. Moreover, it scales poorly with the volume size. To the best of our knowledge, Sellán et al. [9] describe the current best-performing method for SV computation by combining the implicit representation with a numerical continuation method.

In robotics, swept volumes have been employed for collision detection and collision-free path planning. To ensure safe motions, [10] checked pairwise self-collisions for all robot links utilizing swept convex hulls extended by a buffer radius. Baxter et al. [11] introduced a neural network that predicts the SV geometry for a robot moving from a start to goal joint configuration. The method outputs discretized voxel grids, where each voxel indicates either free or swept space. In the same scenario, [12] overcame the accuracy limitation given by the voxel discretization by learning a neural implicit SV model as a signed distance function, requiring large amounts of training data associated with a desired motion type. This approach, however, does not apply to hand guidance, teleoperation or any other human-guided motions.

We are not aware of any prior work that utilize SV in the context of environment modeling.

III. FREE WORKSPACE EXPLORATION

This section describes the entire pipeline for modeling the constraints imposed by static obstacles in the robot cell environment. The proposed method is entirely based on data acquired through exploratory motions, typically lasting only a few minutes, and does not require any additional *external* sensor. Our pipeline consists of four main steps (see Fig. 2), which will be detailed in the following subsections: 1) sweep through free space, 2) swept volume, 3) volume decimation, and 4) obstacle representation. The outcome is a mesh describing the boundary of the explored collision-free space. Steps 1–3 can optionally be repeated to explore additional free space and enhance the resulting model.

It is important to note that, similarly to CAD-based environment modeling, our approach does not address self-collisions or collision-free motion planning. Instead, it provides an environment model of the robot cell that can be integrated into established trajectory planning and control frameworks to generate safe robot motions.

IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

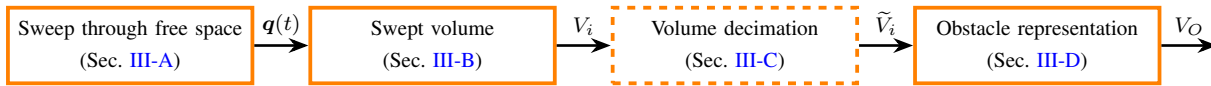


Fig. 2: The proposed pipeline consists of 4 steps. First, exploratory robot motions sweeps the free space of the constrained robot cell. The recorded joint trajectories $q(t)$ are utilized to compute the robot link poses and, thus, the corresponding link swept volumes V_i . These 3D meshes can optionally be decimated to obtain a simplified volume \tilde{V}_i while preserving the overall shape. We obtain a representation of the unexplored and potentially occupied space V_O by carving out the link swept volumes from a bounding volume that covers the entire robot workspace. Steps 1–3 can be optionally repeated in additional exploration sessions to improve the representation of V_O . The 3D mesh associated with V_O can subsequently be utilized within established methods for a collision-free trajectory planning and control.

Assumptions. The approach involves a fixed-based robot with N non-static rigid links interconnected by joints, and is based on the following assumptions:

- A1. The robot cell is static, i.e., there are no moving obstacles or humans within the workspace;
- A2. Joint encoders are calibrated, and the configuration-dependent position and orientation of all links are accessible through precise forward kinematics;
- A3. Accurate meshes for all robot links and tools are available.

The pipeline steps will be illustrated throughout this section through an ideal planar robot manipulator with $N = 3$ non-static links and a KUKA LBR iisy collaborative robot.

A. Sweep through Free Space

The novel idea is to utilize the robot swept volume (SV) resulting from exploratory motions to identify the collision-free space within the constrained robot cell. During the exploratory phase, the robot’s volume may be intentionally modified, i.e., by mounting an additional rigid body on the robot flange. Increasing the robot’s overall shape allows for exploring more space in the same amount of time or the same space in less time without changing the robot’s velocity. The modification of the robot volume also supports industrial scenarios requiring robot tool changes to satisfy the desired task scenario.

1) *Exploratory Robot Motion:* The exploration of the constrained workspace can be performed by a human operator through hand guidance, tele-operation, jogging, or other means. Alternatively, the robot may also explore its cell autonomously, i.e., it moves (randomly) through free space and reverts its direction of motion upon contact detection. The robot’s velocity during this exploratory phase does not affect the subsequent steps of our pipeline as the robot’s SV only depends on the joint configurations. The recorded joint trajectories $q(t)$ of the exploratory motions are continuous and smooth.

2) *Exploration Tool:* The end-effector tool(s) required for the actual task scenario can be used in the exploratory phase. However, the CAD models of such tools are typically very detailed, even when simplified, thus leading to high computation times and, possibly, to numerical instabilities during the subsequent steps of our pipeline. Similar limitations can occur when utilizing a convex decomposition. Moreover, the exploration phase may become time-consuming in the case of small-size end-effector tools. Therefore, in the exploratory

phase, the robot’s shape may be modified by mounting an exploration tool on the flange, which increases the robot’s overall volume and speeds up the exploration process. In this way, the exploration tool, rigidly connected to the robot flange, is considered part of the last link mesh. Such an exploration tool can be customized for the exploratory motion phase. For its design, we recommend simple geometric shapes made of lightweight material without any electronics involved. Furthermore, choosing a form that encloses the tools (e.g., gripper or pneumatic suction cup) used later to achieve the actual task may be beneficial. The dimensions of the exploration tool may also be determined by considering the specific robot cell setup. If, for example, narrower gaps are of interest, the dimensions of the exploration tool should be designed accordingly. Moreover, the exploratory phase may also involve multiple exploration tools. Note that modern tool change systems allow users to quickly and flexibly switch between different tools. In the future, robot manufacturers might deliver new manipulators together with a set of such exploration tools that are cheap to produce.

Utilizing a bounding box or a convex hull to represent the actual tool during the exploration phase constitutes a *non-conservative* approximation within our approach that can lead to severe problems. Both representations encompass spatial regions that are not part of the real tool – such as the gap between the two fingers of a parallel jaw gripper. Hence, utilizing such representations would erroneously label some regions as obstacle-free, even though they have not been traversed during the exploration phase (see Fig. 3). Consequently, this can lead to an incorrect representation of the obstacle-free space, which is unacceptable in safety-critical applications. The proposed exploration tool effectively addresses and overcomes this problem.

B. Swept Volume

The swept volume is the space a rigid body occupies as it moves. By definition, it encompasses all points within the object at any given moment of the motion. Given a rigid body B moving on a path in the time interval $[0, T]$, its swept volume V_S is

$$V_S = \bigcup_{t \in [0, T]} V_B(\mathbf{T}(t)), \quad (1)$$

where $V_B \subset \mathbb{R}^3$ is the volume occupied by the rigid body, whose pose along the path is described by the homogeneous transformation matrix $\mathbf{T}(t)$.

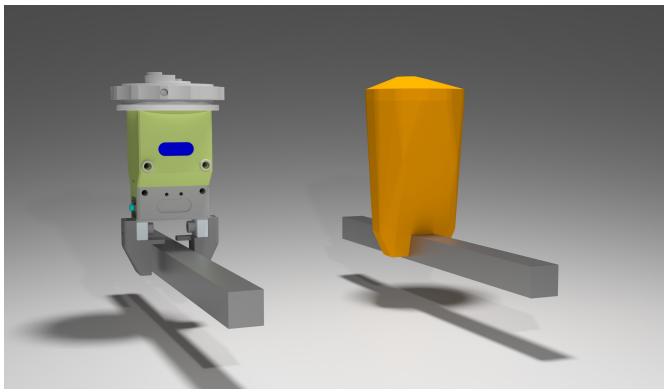


Fig. 3: Example scenario where a gripper moves along a static metal bar, positioning its fingers on either side of the obstacle (left). Using the gripper’s convex hull (or bounding box) would incorrectly label the volume between the two gripper fingers as obstacle-free, despite being occupied (right).

Given the joint trajectories $\mathbf{q}(t)$ recorded in the previous step, the corresponding i -th robot link pose $\mathbf{T}_i(t)$ at time t is obtained by utilizing the robot kinematic model. This allows computing the swept volume V_i of the i -th non-static robot link using (1). The computation can be performed for all N non-static links – possibly including the exploration tool as part of the last link – in parallel improving computational efficiency¹. Note that the SV computation is not negatively affected by clustered or heterogeneously distributed data (e.g., same or similar joint configurations recorded in the exploration phase).

A volume can generally be represented through a 3D mesh, whose faces can be triangles, quadrilaterals, or polygons. In the following, we will consider a triangle mesh to represent a volume without loss of generality.

Figs. 4b–4d show the highly non-convex swept volumes associated with the three links of a planar robot, representing altogether the collision-free space discovered during the exploratory phase.

Fig. 5 illustrates examples of three-dimensional SVs of a KUKA LBR iisy manipulator performing three motions related to gluing and inspection tasks. The mesh details including vertices, faces, and volume², as well as the computation times, are reported in Table I.

C. Volume Decimation (Optional)

The usage of complex mesh representations in the context of computer graphics and 3D modeling has encouraged research into new techniques to (conservatively) simplify and reduce the size of a given mesh. This has resulted in the development of several decimation algorithms [13]. These methods modify the 3D model by reducing the number of vertices, edges, and/or faces without (or conservatively) altering its overall shape and volume. This is achieved by identifying elements

¹Note that the base link of a fixed-base manipulator does not move, and, hence, its swept volume is identical to its body shape.

²The volume of the SV meshes has been computed with the 3D PRINT TOOLBOX of Blender.

not strictly necessary to describe the object mesh. Decimation algorithms iterate until a user-specified termination condition is met, typically defined as a face reduction percentage. In the following, the decimated volume of the i -th link SV is referred to as \tilde{V}_i . This step is optional within our pipeline and often beneficial to reduce overall computation times (see Sec. IV-F).

D. Obstacle Representation

Next, we compute a representation of the unexplored and (potentially) occupied volume V_O . It is obtained through

$$V_O = (((V_{BV} \setminus V_1) \setminus V_2) \setminus \dots) \setminus V_N, \quad (2)$$

where V_{BV} denotes a bounding volume covering the entire robot workspace. An intuitive choice for the bounding volume in the case of a robot manipulator could be a sphere or cube parameterized according to the robot size. The sequential operations in (2) are performed using the Boolean difference operator. Initially defined in the Boolean algebra, Boolean operations have extended to computer graphics and 3D modeling. The Boolean difference of two solid models $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^3$ is defined as $\mathcal{A} \setminus \mathcal{B} := \{x \in \mathbb{R}^3 \mid x \in \mathcal{A} \text{ and } x \notin \mathcal{B}\}$. Given N swept volumes associated with the non-static robot links, the Boolean difference operation in (2) is performed N times³. The resulting volume V_O corresponds to the bounding volume V_{BV} without a portion of the inner part. Hence, V_O can be interpreted as a conservative mesh representation including all obstacles and unexplored areas in the confined workspace. The robot is guaranteed to be collision-free as long as it does not penetrate V_O .

Fig. 4e shows the unexplored and potentially occupied space obtained by considering the link swept volumes of the planar robot previously mentioned.

E. Repeat (Optional)

Suboptimal explorations can occur when the robot only partially sweeps the free workspace of interest. Hence, unexplored areas will automatically be considered obstacles in the representation V_O obtained from (2), negatively affecting the subsequent motion planning and control. Therefore, after inspecting V_O (e.g., potentially using Augmented Reality glasses), the operator can perform additional exploration sessions (steps 1–3). This way, the volume V_O will be further reduced, facilitating the collision-free motion planning.

F. Discussion on Formal Guarantees

Our approach is closely related to [14], describing a purely data-driven technique to environment modeling. Given similar exploratory robot motions, a neural network for collision-free inverse kinematics is trained and a graph representation in the task-space is built. Each node in the graph represents a collision-free configuration. Edges are introduced based on heuristics to connect neighboring nodes, however, this process is not supported by any formal guarantee of a collision-free transition between nodes⁴. Therefore, the authors suggest

³The static base link of a fixed-base manipulator is not considered in (2).

⁴Refer also to the figures 4 and 5 in [14].

IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

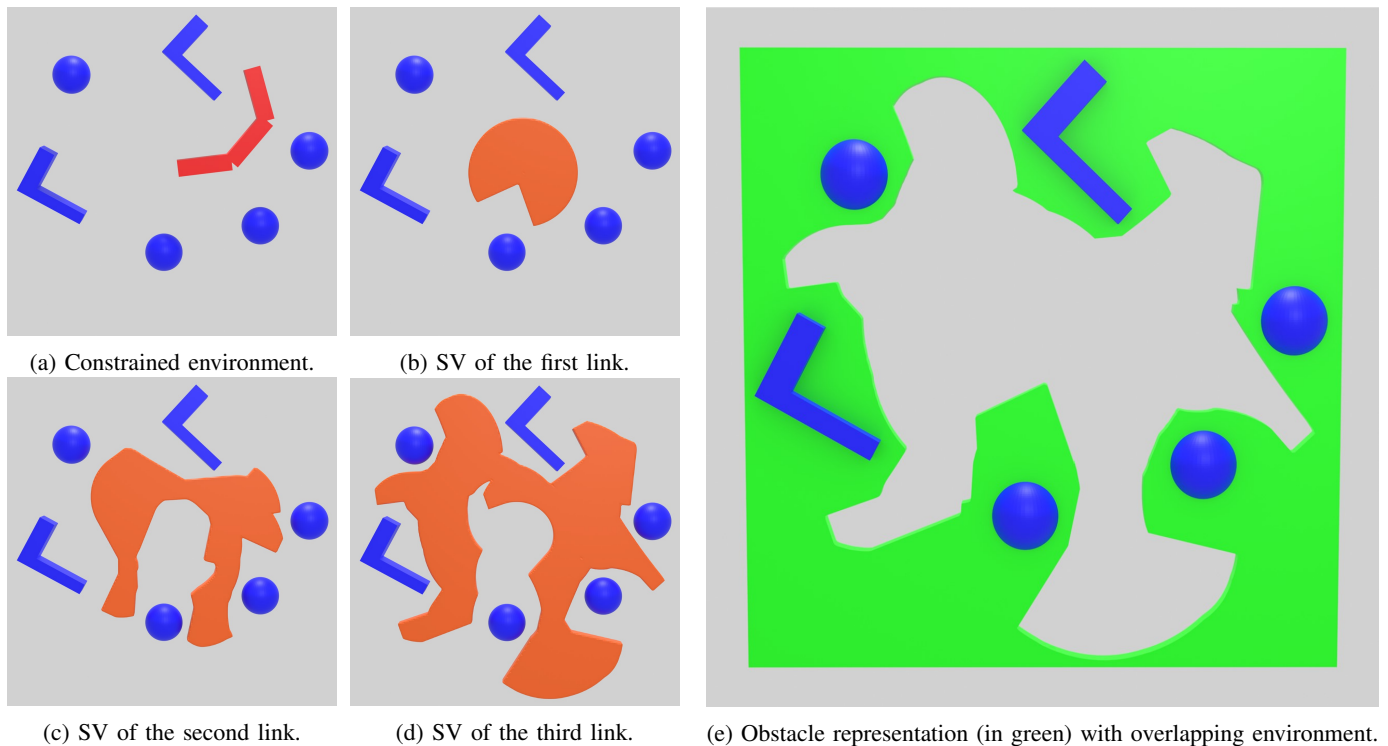


Fig. 4: Results obtained with step 2 and 4 of our proposed pipeline for a planar robot (in red) with three non-static links.

not to collect training data in proximity of the obstacles. Furthermore, the method relies on a non-trivial distance metric that integrates both task- and joint-space information, and requires the tuning of several hyper-parameters. Together, the neural network and graph, implicitly form an environment model, which is utilized for autonomous motion planning. It is, however, not suitable for reactive control.

In contrast, our approach explicitly generates an accurate environment model represented as a mesh, thus providing formal guarantees: as long as the robot moves within the previously explored space, collision-free motions are guaranteed. This is achieved through algorithms for computing SVs and boolean operations that produce correct or conservative results. Another advantage of our mesh representation is that it can be integrated with state-of-the-art optimization techniques to ensure proven collision-free trajectory planning and control, which is crucial for safety-critical industrial applications.

G. Limitations

Our method is explicitly designed for deployment in static robot cell environments (Assumption A1). However, many real-world applications involve dynamic environments with moving obstacles and objects (e.g., on a conveyor), which can lead to unintended robot collisions. To automate processes in these environments, we suggest integrating the environment model generated by our approach with real-time data from additional sensors to detect and track dynamic obstacles and/or objects. On the other hand, it is worth noticing that the assumptions A2 and A3 generally hold in industrial applications.

Currently, our software implementation is not capable of real-time processing. Therefore, it is not possible to provide

TABLE I
SWEEP VOLUME MESH DETAILS AND COMPUTATION TIME (CT) FOR A KUKA LBR IISY 3 PERFORMING THREE DIFFERENT MOTIONS.

Motion	Vertices	Faces	Volume	CT
rect. path	19399	38810	0.43 m ³	24.53 s
semi-circ. path	13093	26182	0.29 m ³	14.35 s
shelf expl.	26724	53502	0.69 m ³	57.90 s

visual feedback of the growing robot swept volume during the exploration phase, though such feedback would be beneficial for the operator. We expect that future advancements in swept volume computation will help address this limitation.

IV. EXPERIMENT

We validate the proposed pipeline within a pick-and-place scenario, confirming its effectiveness.

A. Software Libraries

In this work, the SV computation (1) is based on the algorithm described in [9]. It is implemented within the GPYTOOLBOX library⁵, which leverages the LIBIGL library⁶. The algorithm inputs are the triangle mesh of the solid of interest and its discretized sequence of poses, resulting in a high-quality 3D mesh.

We selected the VISUALIZATION TOOLKIT [15] for the decimation algorithm. The input is a triangle mesh, and the

⁵S. Sellán, O. Stein et al., “gpytoolbox: A python geometry processing toolbox,” 2023, <https://gpytoolbox.org/>.

⁶A. Jacobson, D. Panozzo et al., “libigl: A simple C++ geometry processing library,” 2018, <https://libigl.github.io/>.

IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

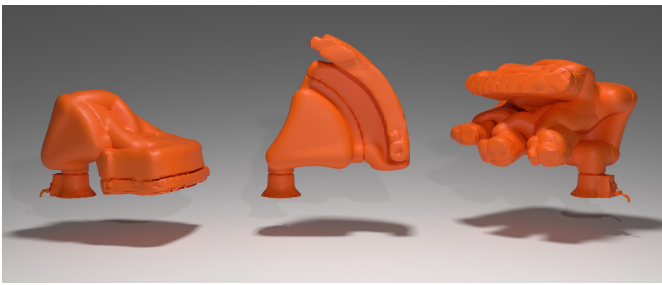


Fig. 5: Swept volumes of a KUKA LBR ii3 for three different motions: a rectangular (left) and semi-circular (middle) path for a gluing task, and the exploration of a shelf (right) for an inspection task.

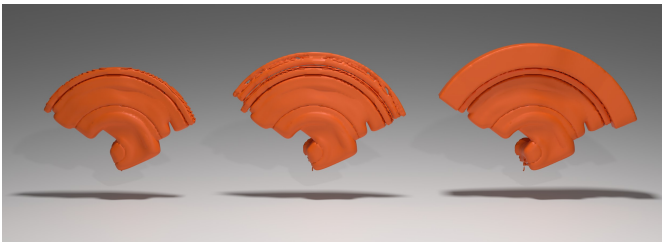


Fig. 6: KUKA LBR ii3 swept volume of a simple trajectory: robot as it is (left), with a parallel gripper (middle), and with a cube-shaped exploration tool (right).

parameters defining the decimation process (e.g., the target percentage of the triangle reduction, the maximum allowed error, and whether the mesh topology should be preserved). It returns a triangle mesh.

Finally, we used the software library LIBIGL for the Boolean difference operation, which is based on [16] and CGAL library⁷. The algorithm inputs are two triangle meshes, provided as a collection of vertices and faces.

B. Hardware Setup

The experimental platform is a KUKA LBR ii3 R760 with $N = 6$ non-static links. The robot is equipped with a SCHUNK change system (FWA series), enabling a quick switch between the exploration tool and the gripper required for the task. The latter is a SCHUNK gripper (GEI FWA-50 series) with parallel 3D-printed fingertips. Overall, it approximately measures $(7 \times 7 \times 12)$ cm. We designed a cube-shaped exploration tool to speed up the exploration phase and avoid high computation times or numerical instabilities. Fig. 9 shows the exploration tool, a lightweight cardboard box of $(20 \times 20 \times 20)$ cm, which encloses the gripper entirely. Fig. 6 highlights the advantages of using this tool by comparing the robot SV mesh for a planar trajectory in three different scenarios: the robot as it is, with a parallel gripper, and with the exploration tool. The SV mesh generated with the gripper shows imperfections such as holes and irregularities. In contrast, the use of the exploration tool results in a smoother

⁷CGAL, Computational Geometry Algorithms Library,” <https://www.cgal.org/>.

and more expansive SV. This underlines our secondary contribution. Fig. 7 displays additional SV examples involving three distinct exploration tool geometries within a simple trajectory. Utilizing Robot Operating System (ROS) 2 to exchange data, all computations are performed on a laptop system with an Intel Core i7-12800H (2.4 GHz) CPU and 32 GB of RAM.

C. Constrained Robot Cell Environment

Fig. 8 shows an industrial cart positioned in front of the robot, with three glasses and a ramp holding six orange blocks. The cell contains four additional obstacles: two boxes between the ramp and the glasses and two boxes next to the robot, limiting its elbow motions. While operating, the robot must avoid self-collisions and collisions with the environment.

D. Pick-and-Place Task

The robot must pick the orange blocks from the ramp and place them into the glasses. Whenever the block at the bottom of the ramp is removed, the remaining blocks shift downward, preserving a constant picking location. This process is repeated six times until the ramp is empty. The first three blocks go into different glasses, and the next three follow the same order. In the end, each glass contains two blocks.

1) *Workspace Exploration*: The operator explores the collision-free space with the collaborative robot, equipped with the exploration tool, utilizing the hand guidance mode proposed in [17]. We record the trajectory at 25 Hz and collect 4009 joint configurations in less than three minutes. Fig. 9 shows selected video frames of the supplementary material.

2) *Robot Link Swept Volumes and Decimation*: The six swept volumes of the non-static links are decimated and jointly visualized in Fig. 1 together with the static robot base. The volume decimation step has reduced both the number of vertices and faces by 63.64%.

3) *Obstacle Representation*: We represent the bounding volume V_{BV} as a cube whose dimensions have been chosen according to the maximum robot length. The mesh of the obstacle representation volume V_O is obtained by iteratively subtracting the mesh representing the link swept volumes \tilde{V}_i from the mesh of the bounding volume V_{BV} .

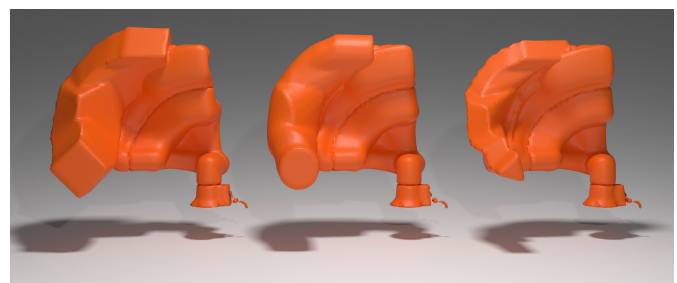


Fig. 7: Swept volumes of a KUKA LBR ii3 executing a simple trajectory with three distinct exploration tool geometries: a cube (left), a cylinder (middle), and a triangular prism (right).

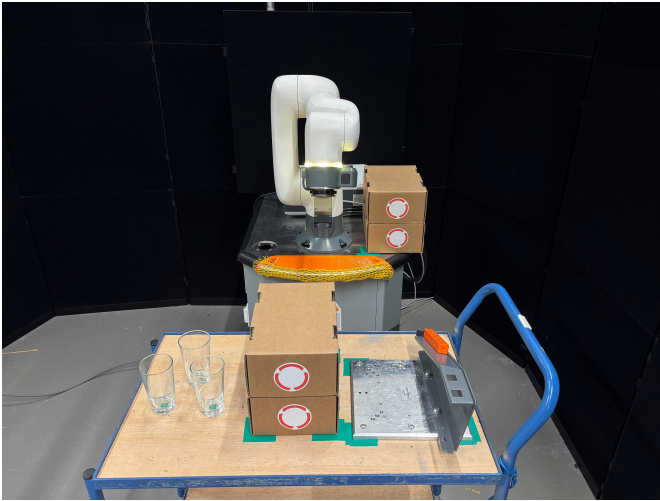


Fig. 8: Constrained cell with a KUKA LBR iisy performing a pick-and-place task. The robot must pick the orange blocks from the ramp and place them into the three glasses by exploiting the proposed obstacle representation.

4) *Motion Planning and Control:* Thanks to the change system, once the exploratory phase has been performed, the gripper quickly replaces the exploration tool. The pick-and-place task is modeled as a simple finite state machine, specifying the pick-and-place poses through the hand guidance mode. Moreover, since a collision is likely detected when the robot picks up an orange block, we record an additional *pre-pick* joint configuration 16 cm above the ramp.

Operators can flexibly select from a range of motion planning frameworks, including QP-based methods, AI-based planners, global planners, or reactive strategies, depending on the demands of the particular application. In this experiment, we rely on the framework presented in [6] and the FLEXIBLE COLLISION LIBRARY (FCL) [18] to generate optimal collision-free trajectories through fast collision checks. Potential collisions are evaluated given the mesh of the robot, gripper, and the environment represented as V_O . The motion framework also considers robot self-collisions.

5) *Task Execution:* The robot executes the planned trajectories without collisions, moving consistently within the previously explored space. For further details, refer also to the video in the supplementary material.

E. Execution Time Analysis

A thorough exploration (performed by the first author) of the free space surrounding the robot took 162.27 s (≈ 2.70 min). Table II reports the execution times of each pipeline step (see Sec. IV-D) averaged over ten independent pipeline executions. From a computational perspective, the most time-consuming operation is the computation of the robot link swept volumes, which takes 185.81 s (≈ 3.1 min). On the other hand, the computation of the volume decimation is almost negligible, taking only a few seconds. Overall, the manual robot workspace exploration and the automatic generation of the environment model V_O took 368.49 s (≈ 6.14 min). Additionally, programming the particular application required approximately 1.5 min

TABLE II
EXECUTION TIMES IN SECONDS AVERAGED OVER TEN INDEPENDENT EXECUTIONS OF THE PIPELINE STEPS FOR A PICK-AND-PLACE TASK WITH A KUKA LBR IISY COLLABORATIVE ROBOT.

Exploration	Swept Volume	Volume Decimation	Obstacle Repr.	Total
162.27 s	185.81 s	2.16 s	18.25 s	368.49 s

to record the home robot configuration, set the pick-and-place poses, and generate a collision-free trajectory. In summary, the robot operator spent 7.64 min to model the robot cell and setup the collision-free robot program.

F. The Role of the Volume Decimation

In this ablation study, we evaluate the role of the volume decimation step in the pipeline from a computational load point of view. Therefore, we have repeated the execution time analysis conducted in Sec. IV-E without performing the volume decimation step. The obstacle representation step works directly with the SV meshes V_i produced by the previous step. The execution times to obtain the obstacle representation increase from 18.25 s to 37.40 s. Hence, the optional and almost costless execution of 2.16 s for the volume decimation step implies a reduction of 16.99 s ($\approx 4.41\%$) on the whole execution time. It is worth noticing that computational optimization may be significantly higher in the case of longer explorations or a more complex SV mesh.

V. CONCLUSION

This letter proposes a novel data-driven and robot-agnostic approach to modeling obstacles within a cluttered robot cell. The method does not rely on additional external sensors, making the environment modeling process cost-effective and immediate. It supports novice users, who can gather the necessary data by hand guiding the robot. After performing exploratory robot motions for few minutes, the unexplored and potentially occupied space is modeled by leveraging the robot's kinematic structure and the swept volume of the non-static links. Our method is capable of effectively managing clustered or heterogeneously distributed data. We obtain a triangular mesh, which is finally used to plan and execute collision-free trajectories safely. Showcasing the method's potential to streamline industrial processes, we validated the intuitive interface in a pick-and-place scenario. Our execution time analysis highlighted that the user can model a robot cell and perform a task in less than eight minutes. Moreover, the ablation study showed the beneficial role of the optional volume decimation (step 3), which further optimizes the computational efficiency almost at no cost.

Future work involves integrating an autonomous exploration mode, where the robot changes its direction of motion upon contact detection. Furthermore, we would like to evaluate less accurate but faster AI-accelerated techniques similar to [11], [12] for visualizing the swept volume in real-time during an exploratory session (e.g., utilizing immersive augmented reality hardware). Finally, a user study demonstrating the programming efficiency compared to conventional methods could offer valuable insights to the community.

IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.



Fig. 9: Exploration phase through hand guidance of a KUKA LBR iiwa equipped with a cube-shaped exploration tool: front view (top row) and side view (bottom row). The corresponding swept volume is shown in Fig. 1.

VI. ACKNOWLEDGMENTS

This work was partly supported by KUKA Deutschland GmbH and the state of Bavaria through the OPERA project DIK-2107-0004/DIK0374/01.

REFERENCES

- [1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [2] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6243–6252.
- [3] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam – learning a compact, optimisable representation for dense visual slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2560–2568.
- [4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [5] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.
- [6] J. D. Muñoz, A. Abdelazim, F. Allmendinger, and U. E. Zimmermann, "Unilateral constraints for torque-based whole-body control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7623–7628.
- [7] M. D. Fiore, G. Meli, A. Ziese, B. Siciliano, and C. Natale, "A general framework for hierarchical redundancy resolution under arbitrary constraints," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2468–2487, 2023.
- [8] K. Abdel-Malek, J. Yang, D. Blackmore, and K. Joy, "Swept volumes: foundation, perspectives, and applications," *International Journal of Shape Modeling*, vol. 12, no. 1, pp. 87–127, 2006.
- [9] S. Sellán, N. Aigerman, and A. Jacobson, "Swept volumes via spacetime numerical continuation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–11, 2021.
- [10] 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.
- [11] 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.
- [12] D. Joho, J. Schwinn, and K. Safronov, "Neural implicit swept volume models for fast collision detection," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 15 402–15 408.
- [13] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, vol. 22, no. 1, pp. 37–54, 1998.
- [14] D. Seidel, C. Emmerich, and J. J. Steil, "Model-free path planning for redundant robots using sparse data from kinesthetic teaching," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 4381–4388.
- [15] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit (4th ed.)*. Kitware, 2006.
- [16] Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson, "Mesh arrangements for solid geometry," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–15, 2016.
- [17] J. D. Muñoz, F. Allmendinger, M. D. Fiore, U. E. Zimmermann, and T. Ortmaier, "Physical human-robot interaction under joint and cartesian constraints," in *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019, pp. 185–191.
- [18] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3859–3866.