Grasp It Like a Pro 3.0: An Expert-Based Data-Driven Algorithm for Grasping Unknown Objects in Cluttered Environments

Gabriele Gambino, Simone Tolomei, *Graduate Student Member, IEEE*, Franco Angelini, *Member, IEEE*, and Manolo Garabini

***Abstract*—Grasping unknown objects in clutter remains challenging due to partial occlusions, self-occlusions, and frequent object interactions during execution. In this paper we present Grasp It Like a Pro 3.0 (GILP 3.0), a lightweight learning-from-demonstration pipeline for closed-loop clutter clearing with unknown objects. The method segments the scene from raw RGB-D point clouds, approximates candidate regions via MVBB decomposition, and predicts grasp pose and interaction wrench from compact MVBB descriptors using two histogram-based gradient-boosted decision-tree regressors trained from a limited number of human demonstrations. Grasp candidates are ranked with execution-oriented feasibility and collision-aware scoring, and the pipeline iteratively reacquires and replans after each attempt to handle object motion and occlusions in clutter. Real-robot experiments on a Franka Emika Panda with Franka Hand in 12 cluttered scenes (47 objects) achieve 95.7% per-object success and a 91.7% scene clearing rate, demonstrating a replicable and data-efficient solution for grasping unknown everyday objects in cluttered environments.**

***Note to Practitioners*—We present a grasping pipeline for cluttered, unknown scenes that outputs collision-aware, executable poses with high first-attempt success. It suits industrial cells handling varied, unlabeled items and frequent product changes, since it relies on coarse box descriptors rather than precise CAD. Deep learning–based method often need large training and extra feasibility checks; purely analytic methods depend on accurate geometry. The pipeline segments the scene (3D clustering and MVBB) and uses two histogram-based GBDT regressors to map box dimensions to a 6-DoF pose and an interaction-wrench metric. It returns a single scored pose that can be processed by standard planners. Requirements: RGB-D/3D sensing with basic calibration; typical tuning is limited to density thresholds and a few score weights. Known failure cases are transparent/specular surfaces and heavy occlusions; light filtering and periodic calibration help.**

***Index Terms*—Grasping, perception for grasping and manipulation, intelligent and flexible manufacturing, human-driven grasping.**

I. Introduction

GRASPING, securely holding, or manipulating objects is a critical capability for robotic systems operating in both simple and cluttered environments. Successful grasping depends on accurate perception, reliable decision-making, and fine motor control to cope with varying shapes and poses, and it becomes especially challenging when objects overlap, are partially occluded, or are randomly arranged. Research has therefore focused on improving algorithms that detect and plan grasp poses and on integrating sensor feedback to ensure robust manipulation under clutter. Analytic methods for grasping [1], [2], [3], [4], which rely on prior knowledge of an object's shape and geometry, offer deterministic solutions for selecting contact points and are efficient when the object model is known. Their main limitation is reduced flexibility under uncertainties—e.g., discrepancies between the object and its model or partial observations—where performance degrades in dynamic or cluttered settings [5].

A key challenge is the variability of object shapes and orientations, which demands robust algorithms capable of generalizing across different geometric features. For this reason, finding a solution to this challenge is non-trivial [6], [7], [8], [9], [10], [11], [12], [13], [14]. Prior work addresses this via model-driven stability analysis [15] and data-driven grasp synthesis [7], but reliability under severe occlusions and tight clearances is still difficult to guarantee during execution. These advances illustrate the growing role of optimization and machine learning in real-world grasping. Model-based methods deliver high precision in structured scenes but degrade under noise and occlusion, whereas perception-driven approaches improve adaptability at the cost of potential losses in accuracy and reliability. In dense clutter—typified by bin picking—overlapping items and partial visibility make detection and execution particularly challenging [16], [17].

At the perception layer, pipelines typically use (i) geometry-driven 3D clustering, (ii) RGB-guided segmentation with

Received 7 October 2025; revised 21 January 2026; accepted 8 March 2026. Date of publication 16 March 2026; date of current version 20 March 2026. This article was recommended for publication by Associate Editor X. Li and Editor H. Moon upon evaluation of the reviewers' comments. This work was supported in part by the European Union's (EU) HORIZON-MSCA-2023-SE-01-01—MSCA Staff Exchanges 2023 Program under Grant 101182891 (NEUTRAWEED), in part by EU by the Next Generation EU Project "Ecosistema dell'Innovazione" Tuscany Health Ecosystem (THE, PNRR, Spoke 9: Robotics and Automation for Health) under Grant ECS00000017, in part by the Italian Ministry of University and Research (MUR)—Fondo Italiano per la Scienza Applicata (FISA) through the OCCAM Project CUP I53C25000700001 under Grant FISA 2023-00324, and in part by MUR in the Framework of the Future-oriented Research Laboratory (FoReLab) Project (Departments of Excellence). The work of Simone Tolomei was supported by the Italian Doctorate in Robotics and Intelligent Machines. *(Corresponding author: Gabriele Gambino.)*

The authors are with the Centro di Ricerca "Enrico Piaggio" and the Dipartimento di Ingegneria dell'Informazione, Università di Pisa, 56126 Pisa, Italy (e-mail: gabriele.gambino@phd.unipi.it).

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2026.3674643, provided by the authors.

Digital Object Identifier 10.1109/TASE.2026.3674643

3D back-projection [18], or (iii) segmentation-free 6-DoF predictors from depth/RGB-D [19], [20]; These techniques respectively emphasize geometric coherence through clustering, semantic richness via RGB-guided segmentation (with the overhead of fusion and back-projection), or end-to-end prediction by direct 6-DoF estimators, which reduce reliance on segmentation but demand larger datasets and careful calibration. Complementary to deep models, learning-from-demonstration (LfD) offers data-efficient and explainable pipelines—regressors trained from human demonstrations combined with analytic scoring—though most evidence comes from single-object or structured scenarios [6], [12].

This paper presents Grasp it Like a Pro 3.0 (GILP 3.0), a data-driven grasping algorithm for closed-loop manipulation in cluttered multi-object scenes. In particular, we target closed-loop tabletop clutter clearing, i.e., the iterative removal of multiple unknown objects from a shared planar support surface under partial occlusions. In dense clutter, grasp success is often dominated by execution feasibility (collisions with neighboring objects and workspace constraints) rather than grasp quality alone, and the scene changes after each attempt, requiring re-acquisition and re-planning in scenes with stacked and, in some cases, nested objects, where occluded targets become reachable only after removing occluders. The intended application scenarios include household/assistive table clearing as well as light industrial pick-and-place on workbenches under occlusions and tight inter-object spacing. Unlike GILP 2.0 [6], which targets isolated single-object grasping, GILP 3.0 addresses overlapping objects and partial occlusions by iteratively reacquiring the scene and re-planning until the workspace is cleared. It introduces an execution-oriented grasp ranking that accounts for scene-level collisions and robot/gripper feasibility, and replaces the single-tree regressor with two lightweight histogram-based GBDT models (pose and wrench) trained on compact MVBB-based geometric descriptors, improving data-efficiency and robustness with a compact set of human demonstrations. Overall, GILP 3.0 maintains low computational cost and a bounded per-attempt runtime without large-scale training, and it is validated through an expanded real-robot clutter-clearing evaluation reported in Sec. IV-B. The remainder of this paper is organized as follows. In Sec. II, we present the relevant works - either model-based or data-driven - on grasping cluttered objects. In Sec. III, we describe in detail the proposed method. Sec. IV describes the experimental setup, and finally Sec. V presents the experimental results and in Sec. VI, we discuss the results.

II. Related Work

Manipulating objects in cluttered environments is a major challenge in modern robotics, requiring advanced perception, planning, and control capabilities. These environments introduce difficulties such as unpredictable object shapes, varying orientations, and complex layouts. Research in this domain has evolved along two main directions: knowledge-based and perception-based approaches. The work of Katz, Kenney, and Brock [21] discusses the challenges robots face in unstructured environments, such as cluttered warehouses and home kitchens, and highlights the role of machine learning and real-time perception to handle incomplete and dynamic information. This provides a foundation for understanding how different approaches tackle clutter.

Knowledge-based methods rely on prior object information (shapes, positions, properties) and are suited to structured environments. For example, [17] presents a mobile bin-picking system integrating navigation, manipulation, and object recognition with noise-resistant detection and active view planning to handle occlusions. Despite promising results, performance depends heavily on model quality, and grasp stability and last-object detection remain challenging. In a similar vein, [22] proposes a physics-based trajectory optimization method for grasp planning in clutter; leveraging simulation improves success rates but incurs substantial computational cost, suggesting the need for parallel computation for real-time use. Finally, [23] proposes an efficient grasp synthesis method that dynamically adjusts approach direction and finger placement with a cost function and genetic algorithms; however, its reliance on precise scene geometry limits flexibility under incomplete or uncertain data.

Since complete knowledge of the environment is often unavailable, learning-based (perception-driven) approaches have gained traction for adaptability and real-time decision making, especially under partial observability. ContactGraspNet [19] predicts 6-DoF parallel-jaw grasps directly from depth, with fast inference and good generalization in moderately cluttered scenes. However, collision handling is typically applied after prediction rather than enforced during generation, and robustness under severe multi-object occlusions is less established. Related deep methods explicitly target clutter and occlusions using pre-existing datasets [24] or self-supervision for partially occluded objects [25]; while they improve robustness or label efficiency, the lack of explicit reachability predictors and the reliance on synthetic data can hamper reliability in heavily cluttered real scenes.

A complementary direction extends deep grasp *generation* to dexterous hands. DexGraspNet 2.0 [20] uses a diffusion-based generator to produce diverse 6-DoF grasp candidates and reports strong per-object performance. In practice it remains largely open-loop and, in dense clutter, predicted grasps may require additional feasibility/collision filtering and replanning to be reliably executable. In addition, training depends on a large synthetic dataset that is expensive to reproduce. Segmentation-driven grasping also includes methods based on segment anything model (SAM) such as GraspSAM [18], which adapt SAM to grasping to predict object masks and a 2D grasp map from RGB with lightweight adapters. They note that full SAM is computationally heavy and therefore use *MobileSAM [26]*/*EfficientSAM* [27] to increase throughput; however, the 2D output still requires back-projection and 3D fusion to obtain 6-DoF poses, adding latency and exposing an accuracy–speed trade-off compared to the full model. Interactive and feedback-based systems offer yet another angle.

Closed-loop, multi-view strategies [28] improve robustness through viewpoint refinement and reactive execution, though they typically require pre-trained components and additional feasibility filters; Human-in-the-Loop solutions [29] combine operator input with autonomous grasping and motion planning

TABLE I
COMPARISON ACROSS METHODS. "N/R" = NOT REPORTED

Method	Clutter	Occlusion level in tests	Training / input data	Time for pose prediction	Per-object success
Contact-GraspNet [19] (CGN)	Yes	low	17.7M samples	∼0.28 s	∼90%
GLP 2.0 [6]	No	No	630 demos	n/r	85%
DexGraspNet 2.0 [20]	Yes	high	427M samples	<0.7 s	92.5%
GILP 3.0 (this work)	**Yes**	**high**	**630 demos**	∼0.8 s	**95.7%**

to handle unstructured scenes, but the dependency on human oversight limits full autonomy. Data-driven pipelines from demonstrations (LfD) remain attractive for their interpretability and modest data requirements. Single-demonstration grasp learning has been demonstrated [30], and methods leveraging human expertise with cuboid approximations and Decision Tree Regression have been proposed [6], [12]. In particular, GLP 2.0 [6] refines grasp generation and introduces a selection policy that improves success rates at the expense of execution time; while effective, evidence is mostly from single-object or structured setups, with limited treatment of multi-object occlusions and robot workspace interactions. Since [6], [12] adopt a Decision Tree Regressor (DTR), it is worth noting that other non-deep pipelines employ gradient-boosted decision trees (GBDT) to *score* grasp proposals from RGB/RGB-D—e.g., LightGBM [31] on Cornell-style rectangles—favoring candidate ranking over full 6-DoF pose regression.

To address the limitation of existing works, we propose GILP 3.0. Our method provides a data-efficient and explainable pipeline tailored to cluttered, occluded scenes, treating feasibility as a design principle rather than a post-hoc filter. Concretely, we couple scene clustering with an extended grasp score that penalizes interactions with surrounding objects and the workspace, together with a planning-aware selection policy. The method reports both per-object and scene-level outcomes, positioning GILP 3.0 as a practical choice when accountability, safety margins, and predictable computation are required, while remaining compatible with hybrid pipelines in which learned predictions are filtered and prioritized by an explicit score. As summarized in Table I, deep-learning pipelines such as Contact-GraspNet and DexGraspNet typically rely on large-scale training datasets, whereas GILP 3.0 achieves strong per-object performance in clutter with only 630 human demonstrations, highlighting its data efficiency.

III. GRASPING ALGORITHM

This section introduces the GILP 3.0 algorithm 1, whose pipeline is illustrated in Fig. 1. The algorithm begins by acquiring a point cloud of the entire environment, which is segmented into clusters. A policy derived from the segmentation is used to select a specific cluster for further processing. The selected cluster is then decomposed into N Minimum Volume Bounding Boxes (MVBBs) using the algorithm in [32]. These MVBBs are then processed to facilitate the generation of feasible grasp pose candidates, which is performed using two histogram-based GBDT regressors—one for pose and one for wrench—trained on data from human experts. Part of the algorithm parameters are automatically tuned to reduce computational cost and execution time.

Algorithm 1 GILP 3.0

```
Initialize: z_t ← Threshold, h ← End Effector ;
while true do
    Q ← Acquisition ;
    [C*, z_max] ← QProcessing(Q, z_t);
    if C* is EMPTY then
        break
    X = |C*| foreach i ∈ X do
        [G*] ← GraspComputation(C*_i, z_max, Q,h);
        flag ← MotionPlanning(G*, Q, C);
        if flag == SUCCESS then
            break
```

The generated candidates are scored, ranked and filtered to remove potentially harmful solutions. To address the challenges of cluttered environments, where objects may move or fall between grasps, the algorithm incorporates an iterative loop. This ensures that the system adapts to changes and consistently selects a high-scoring grasp pose.

A. Point Cloud Processing and Clustering

The first step involves acquiring a point cloud to detect and group objects in the scene. Among the available options, we choose to use the RGB-D cameras to acquire the point cloud of the set of to-be-grasped objects. These sensors provide color and depth information and are often used to obtain rich 3D data in complex, cluttered scenes [33]. The point cloud is defined as a set of points $Q = \{q_i = [x_i, y_i, z_i]^\top\}$. Since Q usually includes more than $10^5 \sim 10^6$ elements, its processing is demanding. To reduce the computational cost and the overall algorithm execution time, a subset of Q is considered, specifically.

$$\bar{Q} = \left\{ q_i = [x_i, y_i, z_i]^\top \in Q | z_i \geq \frac{z_{\max}}{2} \right\}, \tag{1}$$

with

$$z_{\max} = \begin{cases} \max\limits_i(z_i) & \text{if } \max_i(z_i) \geq z_t \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where $z_i \in q_i$, $q_i \in Q$, and z_t is a threshold parameter to avoid reducing point clouds which describe scenes including only short or flat objects (and therefore with an already limited number of points); in this latter case, $\bar{Q} = Q$. In the real-robot pipeline, before clustering we additionally apply a voxel-grid downsampling to $\bar{Q}$ to suppress isolated noise and homogenize point density.

Since the scene is composed of multiple objects, the next step is to identify them to grasp them one at a time. To this end, we apply clustering algorithms to the reduced point cloud $\bar{Q}$

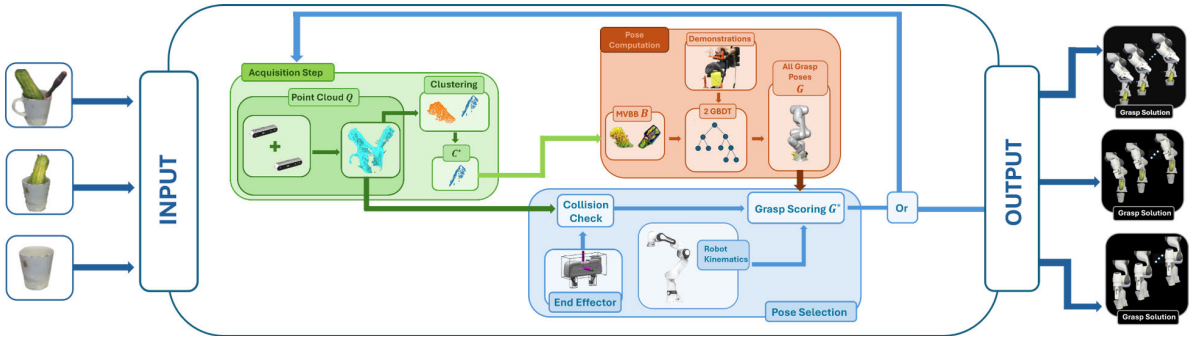

Fig. 1. GILP 3.0 pipeline for closed-loop clutter clearing. From a raw RGB-D point cloud, the scene is clustered and a target cluster is selected; MVBB decomposition generates candidate boxes that are refined and pruned to limit computation in clutter. For each box, grasp pose and interaction wrench are predicted from MVBB descriptors using two lightweight GBDT regressors trained from human demonstrations, and grasp orientations can be aligned to local surface normals to handle flat cases and reduce collision risk in tight scenes. Candidates are ranked with execution-oriented scoring that accounts for gripper/robot feasibility and collision/workspace constraints. After each attempt, the scene is re-acquired and the loop repeats until clearing.

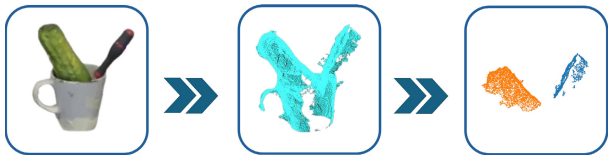

Fig. 2. Point-cloud acquisition and preprocessing. Starting from the raw point cloud of the scene, a height-based filtering step reduces the data to the relevant workspace region; the filtered cloud is then clustered to obtain object hypotheses used in the subsequent MVBB decomposition and grasp computation.

(Fig. 2), which segment and label objects based on geometric and topological features. These algorithms group data into clusters $C_j \subseteq \bar{Q}$, keeping points within the same cluster as close as possible while maximizing the separation between different clusters. A detailed overview of these algorithms is provided in [34] and [35]. Among the possible methods, we use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [36]; Affinity Propagation (AP) [37]; and Spectral Clustering (SP) [38]. In our pipeline, 3D point-cloud clustering is used as a standard preprocessing module for its training-free design, maintaining low computational cost and a bounded per-attempt runtime, and volumetric partitions, rather than as a primary research contribution. By contrast, as discussed in [18], SAM-based segmentation is computationally heavy, whereas MobileSAM/EfficientSAM increase throughput but reduce mask fidelity, reflecting an inherent speed accuracy trade-off.

At the end of the clustering algorithm execution, L clusters C_j are obtained such that $\cap^L C_j = \emptyset$ and $\cup^L C_j \subseteq \bar{Q}$. Each cluster represents the visible part of one of the objects in the scene. To facilitate the grasping phase, the objects are selected starting from the highest, reducing potential collision occurrence. To this end, we first compute that maximum $z_{\max,j}$ value for each cluster C_j, i.e., $z_{\max,j} = \max_i(z_i)$ with $z_i \in q_i, q_i \in C_j$. Then, clusters are ordered in a list with decreasing values of $z_{\max,j}$. The first analyzed cluster in the next step is one with the maximum $z_{\max,j}$, i.e., $C^* = C_j$ with $j = \arg\max_j(z_{\max,j})$.

B. Object Shape Approximation

After acquiring and pre-processing the point cloud, and dividing it into clusters, the next goal is to approximate the object shape into basic elements (i.e., boxes) to facilitate the grasp pose generation. To this end, we rely on MVBB decomposition, which has already been used in literature by [6], [12], [32], and [39]. The boxes represent a simplified form of the objects, enclosing their respective volumes. This decomposition step facilitates learning the grasping poses from human experts, providing a structured representation of the scene and reducing the number of required human examples. To reduce computational load, the bounding boxes are not computed for the whole point cloud, but only for the cluster C^* selected as described in Sec. III-A. Each bounding box is associated with relevant information such as position, orientation, and size, which are used for grasp planning. Each box is defined as $b \triangleq \langle T_\mathrm{b}, \lambda_\mathrm{b} \rangle$, where

$$T_\mathrm{b} \triangleq \begin{bmatrix} R_\mathrm{b} & y_\mathrm{b} \\ 0 & 1 \end{bmatrix} \triangleq \begin{bmatrix} \hat{r}_{\mathrm{b},1} & \hat{r}_{\mathrm{b},2} & \hat{r}_{\mathrm{b},3} & y_\mathrm{b} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

is the transformation matrix of the pose in the world frame, while $\lambda_b \triangleq [\lambda_{b,1}, \lambda_{b,2}, \lambda_{b,3}]^\top$, is the vector of the box dimensions. All the boxes are identified in a set B, so $b \in B$. The number and size of boxes generated by the MVBB decomposition algorithm is influenced by a set of parameters: the gain h, the minimum v_m and the maximum volume v_M of the box, the minimum number of points μ_b within the box, and the minimum point density ρ_b.

In [12], μ_b was fixed, which led to over-segmentation of small objects and overly large boxes for large objects, exceeding the gripper's range. [6] addressed this by defining μ_b as a logarithmic function of the number of points X in C^*, i.e., $X = |C^*|$, specifically

$$\mu_\mathrm{b} = [a_1 \log(a_2 X + 1)], \tag{4}$$

where a_1 and a_2 are design parameters, chosen from experimental trials. Eq. (4) balances the decomposition, ensuring high-quality grasps for both small and large objects. However, in some cases this may also lead to an overly large number of boxes, which is useful to achieve finer grasps, but it also increases computational cost with complexity $O(N)$, where N is the number of boxes. To limit over-segmentation, we set μ_b as in (4) and tune the minimum box volume $v_m = 10^{-t}$ with $t \in [3, 5]$ (step 0.5), balancing geometric fidelity and runtime. The smallest threshold considered in our experiments

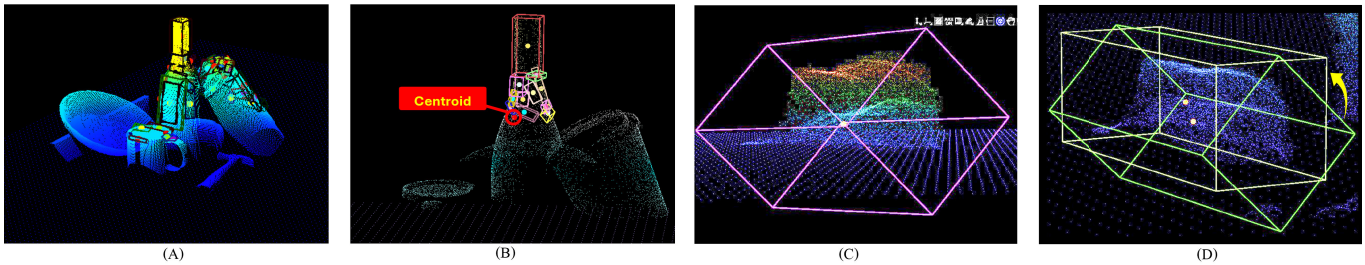

Fig. 3. Effect of MVBB post-processing. (A) Initial sloped MVBBs. (B) Orientation adjustment aligning the boxes with the world XY plane. (C) MVBB decomposition before centroid-based reduction. (D) Centroid-based box reduction, which prunes lower boxes and decreases the number of candidates used by GILP 3.0.

is $v_m = 10^{-5}\,\mathrm{m}^3$, which corresponds to a characteristic length of $\approx 2.2\,\mathrm{cm}$ and is well above the gripper finger thickness scale (about 5 mm), thus avoiding boxes that are too small to yield reliable pinches. This choice keeps the number of MVBBs bounded while discarding geometrically insignificant partitions, thereby improving computational efficiency. To further reduce the computational cost, the grasp calculation is computed only on the upper boxes of the cluster C^* (Fig. 3). Specifically, for each box its centroid is computed such as $c_\mathrm{b} = T_\mathrm{b}\frac{\lambda_\mathrm{b}}{2}$. Then, the vertical component of $c_\mathrm{b} = [x_\mathrm{cb}, y_\mathrm{cb}, z_\mathrm{cb}]^\top$ is compared with the mean of the vertical component of all points $q_i = [x_i, y_i, z_i]^\top \in C^*$, i.e., $z^* = \frac{1}{X}\sum_{i=1}^{X} z_i$, with $X = |C^*|$. This means that a box is analyzed only if $z_\mathrm{cb} \geq z^*$, reducing the computational range by half or more. However, this logic is not applied to small objects, where the best grasp is often obtained with a single box, corresponding to an exponent of 3. To handle such cases, the centroid-based reduction is applied only if more than one box is calculated. Otherwise, no reduction is performed to ensure the grasp remains valid.

During the grasp selection, it is of paramount importance to consider and avoid collisions. Vertical grasps help in reducing this risk, especially in cluttered scenarios or in the case of small objects. Indeed, in this latter scenario, it is very likely that a side grasp may cause contact between the end-effector and the working surface.

To avoid this, the boxes generated by the MVBB decomposition are re-orientated to align one of the box faces with the XY-plane of the world frame, as shown in Fig. 3. To this end, given a box $b_i \in B$ generated from cluster C^* and its orientation matrix R_i defined as in (3), the new orientation matrix R'_i is given by

$$R'_i = R_i R_\mathrm{x}(\theta_i)\ , \tag{5}$$

$R_\mathrm{x}(\theta)$ is the rotation matrix about the x-axis, while the angle θ_i is computed such as

$$\theta_i = \left(\arctan 2\left(|\hat{r}_{\mathrm{b},3,i} \times n_t|,\ \hat{r}_{\mathrm{b},3,i}\, n_t\right)\right)\ , \tag{6}$$

with $\hat{r}_{\mathrm{b},3,i}$ defined as in (3), and n_t is the normal to the plan, i.e., $n_\mathrm{t} = [0, 0, 1]^\top$. The method is shown in the algorithm 2 This strategy has proven effective in handling orientation-critical cases, improving the algorithm's robustness and generalization across different object geometries.

Algorithm 2 Rotation of Bounding Boxes

```
Function BoundingBoxRotation(B):
    n_b ← size(B, 1) ;  // Bounding Box number
    rot_b ← cell(n_b, 1) ;  // Initialization
    t_n ← [0; 0; 1] ;  // Plane Normal
    for i ← 1 to n_b do
        b ← B[i];
        b_n ← b.T(1 : 3, 3) ;  // Bounding Boxes normals
        θ_x ← atan2(‖cross(b_n, t_n)‖, dot(b_n, t_n));
        R_b ← b.T(1 : 3, 1 : 3) R' ← R_b * R_x(θ_x)
        b.T(1 : 3, 1 : 3) ← R' ;  // Rotation Matrix Update
        B'[i] ← b ;  // Save the rotated box
    return B';
```

C. Grasp Computation From Human Examples

Once the boxes approximating the object have been properly created, the next step is to generate potential grasps and select the best suited for the specific scene. To achieve this, we follow the approach proposed in [6] with the difference that [6] focuses on single-object scenarios, while in this work we consider multiple objects settings. In the following, we summarize here the method presented in [6] highlighting the main differences.

As described in Sec. III-B, a list of boxes B for the selected cluster C^* is generated and appropriately modified. Then, for each box $b \in B$, a list of grasps G is computed. Each candidate grasp is defined as $g = \langle h, p, b, w\rangle$, where h is the specifically employed end-effector, p is the predicted hand pose, and w is the predicted interaction wrench metric associated with pose p, which is employed to select grasps with better quality. p and w are predicted based on example data generated by a human operator using the specific robotic end-effector h to grasp boxes [6], [12].

The robotic end-effector is defined as $h = \langle O_\mathrm{h}, \mathcal{O}, \mathcal{C}, \delta\rangle$, where O_h represents the end-effector reference frame (position and orientation), $\mathcal{O}$ is the gripper shape, $\mathcal{C}$ is the region swept by the fingers during closure [40], and δ is the thresholds for maximum size, alignment, and collision constraints. The method can be applied to various robotic end-effectors, for instance to the Franka Emika Hand [41]. In this case, the elements of h are depicted in Fig. 4.

Given the specific hand, grasp computation is generated using human-collected data. In the case of the Franka Emika Hand, 630 grasp poses and the associated wrench were gathered across various cuboid configurations [6]. Specifically, a

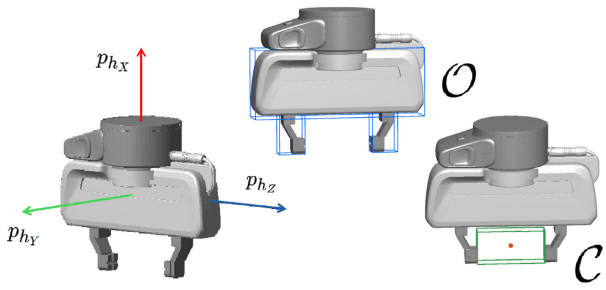

Fig. 4. The reference Franka Emika Hand frame O_h is designed as $O_h = [p_{h_x}, p_{h_y}, p_{h_z}]$. The gripper shape $\mathcal{O}$, which is approximated with three blue boxes, one for the main body, and two for the fingers. The closure $\mathcal{C}$ of the gripper is shown with a green box.

human operator -using the robot's hand-guiding interface - guided the gripper to a grasping position for the specific cuboid. Then, the end-effector pose was saved through robot kinematics, while the interaction forces were measured with a force/torque sensor mounted below the cuboid. Differently from boosted-tree approaches [31] that *score* grasp candidates, we generalize these data using two histogram-based GBDT regressors—one for pose and one for wrench—that directly regress the grasp pose p and the interaction-wrench metric w from the box size λ_b. Specifically,

$$\begin{bmatrix} p \\ w \end{bmatrix} = \begin{bmatrix} \psi(\lambda_b) \\ \nu(\lambda_b) \end{bmatrix} , \tag{7}$$

where $\psi(\lambda_b)$ and $\nu(\lambda_b)$ are the predictive model for the grasp pose $p \in \mathbb{R}^6$ and the interaction wrench metric $\omega \in \mathbb{R}^2$, respectively.

Differently from [6], which employed a single Decision Tree Regressor (DTR), here we adopt two histogram-based GBDT regressors—one for pose and one for wrench. Despite the relatively small size of the dataset (630 demonstrations on cuboidal proxies), this combination of geometric box descriptors and boosted trees supports transfer to unseen object shapes, as evidenced by the real-robot experiments in clutter.

In summary, the grasp computation phase generates a set of grasps G related to all the boxes considered B that approximate (part of) the selected cluster C^*. Each grasp candidate $g \in G$ is then ranked based on a specific grasp score $S_g(g)$, whose maximum and best value is 1. $S_g(g)$ is evaluated as the multiplication of several sub-scores, each one modeling a different aspect of grasp quality. $S_g(g)$ was introduced in [6], such as

$$S_g(g) = J_b J_w J_a J_c , \tag{8}$$

where J_b (box score), evaluates the density of points within box b and its distance from the point cloud centroid C^*, favoring boxes that better approximate the object's geometry. J_w (wrench score), incorporates the interaction wrench metric w, prioritizing grasps that can generate higher forces for more robust handling, and J_a (alignment score) measures how well the robotic hand frame O_h aligns with the box one R'_b, penalizing grasps that exceed the gripper's physical limits. Finally, J_c (collision score) takes into account potential end-effector-object collision only in single-object scenarios.

In this work, grasps are ranked using S_g such as in [6]. Besides modifying the collision score J_c for multi-object cluttered environments, we also refine the alignment score J_a. As in [6] (Eq. 11), J_a enforces the gripper physical limits through the orientation and size thresholds δ_θ and δ_λ; in GILP 3.0 we additionally introduce a minimum pinch-thickness constraint ℓ_{min} to avoid degenerate edge grasps under partial views and clutter. Formally, we apply the gate

$$J_a = \begin{cases} J_a, & \ell \geq \ell_{min}, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

where ℓ denotes the MVBB thickness along the chosen pinch direction and $\mathbb{I}(\cdot)$ is the indicator function. We then modify J_c to handle multi-object cluttered environments, integrating additional penalties for environmental congestion, workspace constraints, and gripper interference. Specifically, the collision score proposed in this work is

$$J_c = \kappa_{\mathcal{O}} \kappa_{\mathcal{C}} \kappa_{\mathcal{R}} , \tag{10}$$

where $\kappa_{\mathcal{O}}$, $\kappa_{\mathcal{C}}$, and $\kappa_{\mathcal{R}}$ are indexes designed to take into account collisions between the whole gripper and all the objects, collisions of the fingers during the closure; and collisions of the robot with the surrounding environment, respectively.

The index $\kappa_{\mathcal{O}}$ ensures that grasp candidates are not selected in highly cluttered regions. Its formulation follows the structure of [6], where a threshold-based decision determines the feasibility of each grasp

$$\kappa_{\mathcal{O}} = \begin{cases} 1, & \text{if } \rho_{\mathcal{O}} \geq \delta_{\mathcal{O}} \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $\rho_{\mathcal{O}}$ is the density of the occupancy volume, while $\delta_{\mathcal{O}}$ is one of the elements of the vector δ in h, which represents the maximum allowable density before the grasp is discarded. Unlike [6], where density computation was restricted to the target object, in this work $\rho_{\mathcal{O}}$ includes the evaluation of the entire point cloud of the environment, including all objects, i.e., point cloud Q, and the surface of the workspace, which is defined as a point cloud W. Thus, the point cloud of the environment E is defined as $E = Q \cup W$. Considering this point cloud ensures that grasp candidates are not selected in poses that could lead to collisions with the working surface or any other objects in the scene. The collision density $\rho_{\mathcal{O}}$ is computed as

$$\rho_{\mathcal{O}} = \frac{|E \cap \mathcal{O}|}{|\mathcal{O}|} , \tag{12}$$

where $|E \cap \mathcal{O}|$ defines the number of points of E which are inside the hand shape occupancy region $\mathcal{O}$, while $|\mathcal{O}|$ is the volume of $\mathcal{O}$.

The index $\kappa_{\mathcal{C}}$ takes into account potential obstructions during the gripper closure. In this regard, we consider the closure collision density ρ_c, which is computed as

$$\rho_c = \frac{|Q \cap \mathcal{C} \setminus b|}{\mathcal{C}} \tag{13}$$

where $\mathcal{C}$ is the gripper's closure region. This metric allows us to measure the ability of the gripper to close around the object without obstruction caused by other parts of the

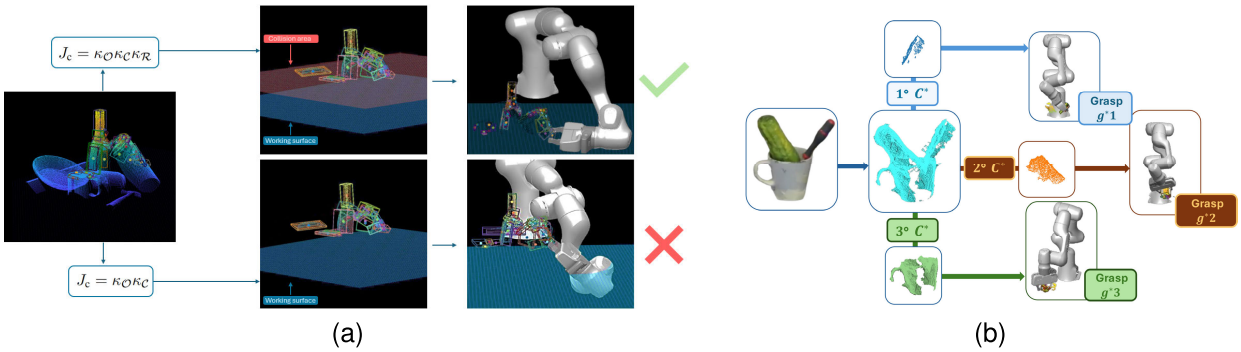

Fig. 5. (a) The introduction of $\kappa_{\mathcal{R}}$ in the sub-score J_c prevents the selection of a candidate grasp pose potentially colliding with the working table. Effect of adding the workspace-awareness term $\kappa_{\mathcal{R}}$ in the collision sub-score J_c: without $\kappa_{\mathcal{R}}$, a high-scoring candidate may be selected even if it leads to collisions with the support plane; with $\kappa_{\mathcal{R}}$, such candidates are penalized and the selected grasp remains feasible with respect to the working surface. (b) The grasping candidates g, computed starting from the cluster C^*, are ranked with a score algorithm. This approach have been for each cluster after every grasp done. Ranked grasp candidate g selection in clutter. For a selected cluster C^*, the method generates a set of candidates and sorts them into an ordered list G^* according to the grasp score. If the top candidate is infeasible (e.g., collision or planning failure), the algorithm automatically tries the next ranked candidate, ensuring robust execution without recomputing the full candidate set.

same or other objects in the scene. A high value ρ_c indicates obstruction within the closure region, while a low value confirms unobstructed closure, improving grasp stability.

To improve robustness in cluttered environments, we replace the cubic spline penalty used in [6] with a sigmoidal function

$$\kappa_C = 1 - \frac{1}{1 + e^{-k(\rho_c - \delta_C)}}, \tag{14}$$

where k controls the transition steepness and δ_C represents the penalization threshold. The sigmoid replaces the cubic spline to provide smoother and more predictable penalization, reducing instability caused by partial occlusions. This is especially beneficial in cluttered environments, where small changes in object position can affect grasp feasibility without abruptly rejecting otherwise valid grasps. Moreover, reducing the number of parameters simplifies tuning and improves computational efficiency.

Finally, collisions may occur also due to the positioning of the robot's body w.r.t. the surrounding environment. However, the evaluation of these potential collisions for each grasp candidate is highly demanding. To reduce computational load, we include the index $\kappa_{\mathcal{R}}$, which penalizes low grasps, that is, grasps that present a gripper position close to the environment, e.g. the table. Given the gripper pose $p \in \mathbb{R}^6$, the index $\kappa_{\mathcal{R}}$ is defined as

$$\kappa_{\mathcal{R}} = \begin{cases} 1, & \text{if } p_3 > \dfrac{z_{\max}}{2} \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where $z_{\max}$ is defined as in (2). If $\kappa_{\mathcal{R}} = 0$ the grasp is penalized, promoting configurations that avoid workspace interference. When $z_{\max=0}$, the full point cloud is considered, and the algorithm discards only the candidate grasps which falls below the working surface. The effect of the index κ_R is shown in the Fig. 5a.

Given $S_g(g)$ in (8) and the set of all grasp candidate G, the list of ranked grasp candidates is created as

$$G^* = \text{sort}(G, S_g(g)), \tag{16}$$

where the first element, $g_1^* \in G^*$, corresponds to the grasp pose with the highest score, i.e.

$$g_1^* = \arg\max_{g \in G} S_g(g). \tag{17}$$

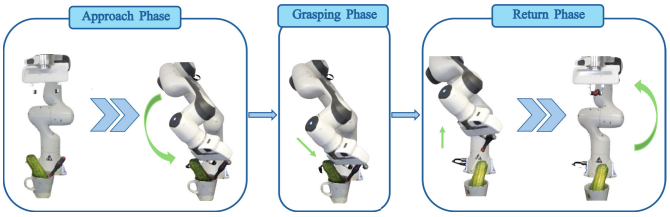

Fig. 6. Three main phases of the robot motion. 1) In the approach phase, the robot reaches an intermediate pose, $g_{a,1}$, before grasping to ensure a safe movement and avoid collisions. 2) In the grasping phase, the end-effector moves to the grasping pose g_1 along the local Z-axis, and the gripper closes. 3) In the return phase, the robot safely moves away from the grasping area and returns to its initial configuration.

The creation of the ordered list G^* ensures that if the highest-scoring grasp is discarded due to safety constraints or execution issues, the next-best candidate can be selected without recomputation. An example of the grasps poses found for a set is shown in Fig. 5b.

D. Motion Planning Algorithm

Once the grasp candidates have been ranked as G^*, the best candidate grasp g_1^* needs to be executed on the physical robot. After ranking, we validate the g_1^* pose through motion planning by checking a collision-free trajectory to the approach and grasp pose w.r.t. the full reconstructed scene (including neighboring objects and the support surface). To this end, the movement sequence, shown in Fig. 6, consists of three main phases: i) the approach phase, ii) the grasp phase, and iii) the return phase, where the robot safely moves back to its initial configuration. During these three phases, the surrounding environment is perceived as an obstacle. This includes the fixed objects (e.g., table, camera supports, etc.), and the to-be-grasped objects too. Fig. 7a, 7b, 7c shows the approximation as obstacles of the elements surrounding the robot, within the motion planning framework. The fixed objects are approximated as rectangular boxes b_{fo}, depicted in Fig. 7a, and they are considered as obstacles during all three phases. The to-be-grasped objects are represented by the point cloud Q, which can be divided into the target cluster

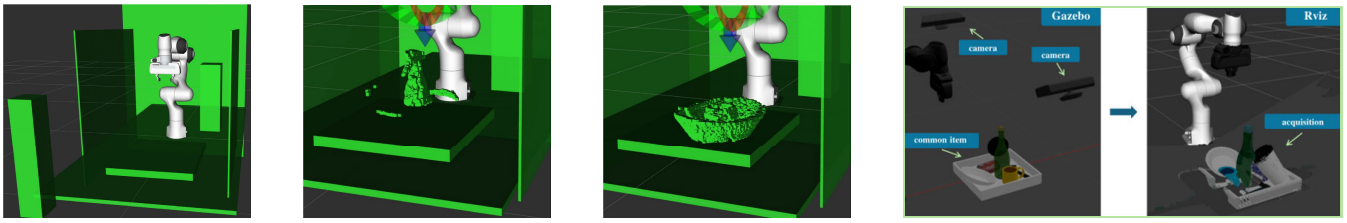

Fig. 7. The collision of fixed objects (a) in the work environment is represented by boxes that maintain their dimensions. Two walls are placed on either side of the manipulator to encourage more linear movements during motion planning. The point cloud is approximated using small boxes and is divided into two parts during the experiment: the cluster under examination and the remaining point cloud, shown in (b) and (c), respectively. (d)The simulation environment is built in Gazebo to replicate a real-world scenario, including two cameras positioned on top and common objects arranged in a cluttered scene. The camera acquisition is then visualized in Rviz.

C^*, treated as an obstacle only during the approach phase, and the remaining part, defined as $Q_{\mathrm{d}} = Q \setminus C^*$, considered as an obstacle in all three phases. Unlike fixed objects, for the to-be-grasped objects a finer approximation is used (Fig. 7b and Fig. 7c). For each point $c^* \in C^*$ and each point $q_{\mathrm{d}} \in Q_{\mathrm{d}}$, a small bounding box $b_{c^*} \in B_{c^*}$ and $b_{q_{\mathrm{d}}} \in B_{q_{\mathrm{d}}}$ is created, defined as

$$b_{c^*} = \left(c^*, \lambda_{b_{c^*}}\right), \quad b_{q_{\mathrm{d}}} = \left(q_{\mathrm{d}}, \lambda_{b_{q_{\mathrm{d}}}}\right), \tag{18}$$

where $\lambda_{b_{c^*}}$ and $\lambda_{b_{q_{\mathrm{d}}}}$ represent the size vectors, following the same notation as λ_{b}. This approximation enables motion planner to process the obstacles. After these preliminary considerations, the robot begins its movement sequence to perform the grasp, ensuring both its safety and that of the surrounding environment.

i) Approach phase: In this phase the robot end-effector is moved from the initial configuration to the approaching pose, namely $g^*_{\mathrm{a},1}$, which is a pose close to the grasping one g^*_1. The addition of this intermediate step is necessary to reduce collisions during movements close to the target cluster. The approaching pose is defined as

$$g^*_{\mathrm{a},\ 1}(d) = g^*_1 - [0, 0, d]^\top, \tag{19}$$

where d is an incremental offset ranging from 0.005 m to 0.05 m. During this phase, the RRT Connect algorithm is used to compute the trajectory. If a potential collision is encountered with the bounding boxes b_{c^*} or b_{d}, the offset d is increased and the motion computation is restarted. If no feasible path is found with $d = 0.05m$, then the candidate grasp g^*_1 is discarded, and the approach phase planning is restarted with the next best grasp candidate g^*_2.

ii) Grasping phase: In this phase the robot end-effector is moved from the approaching pose $g^*_{\mathrm{a},\ 1}(d)$ to the grasping pose g^*_1 using a waypoint approach. In this phase, the cluster collision boxes b_{c^*} are removed, allowing unobstructed access to the grasp pose g^*_1. Conversely, the bounding boxes b_{d} remain active throughout the process to prevent collisions with the remaining part of the point cloud. This phase ends with the closure of the gripper.

iii) Return phase: The robot end-effector is moved from the grasping pose g^*_1 to the home configuration using a two-step motion. First, the robot executes a vertical movement along the Z-axis to mitigate potential constraints with nearby objects, and then the robot moves back to its initial configuration (home configuration). This motion is planned using the RRT Connect algorithm.

Fig. 8. Comparison of clustering performance, in terms of cluster identified and of execution time of each algorithms.

TABLE II

PARAMETERS INTRODUCED/MODIFIED IN GILP 3.0 W.R.T. GILP 2.0 [6] AND THE EXACT VALUES USED IN OUR EXPERIMENTS (DBSCAN CLUSTERING, MVBB DECOMPOSITION, AND EXECUTION-ORIENTED SCORING/FEASIBILITY). UNITS: METERS (M) AND CUBIC METERS (m^3); REMAINING PARAMETERS ARE UNITLESS OR POINT-COUNT THRESHOLDS

Clustering & MVBB		Grasp scoring & feasibility	
Symbol	**Value**	**Symbol**	**Value**
ε	0.02 m	k	10
ζ	2500	δ_C	0.5
h	0.98	δ_θ	0.4363 rad
μ_b	10	δ_λ	0.09 m
ρ_b	0.01	$\ell_{\min}$	0.015 m
v_M	10^{-1} m^3		

In phases (ii) and (iii), potential execution failures are handled through a fallback strategy. If the robot is unable to execute the grasp g^*_i, the algorithm attempts the next ranked candidate g^*_{i+1} in G^*, iterating until a feasible grasp is found. This ensures that the system explores the ranked candidates until a valid solution is identified while avoiding collisions.

For reproducibility, Table II reports the exact values used in our experiments for the parameters introduced or modified in GILP 3.0 with respect to GILP 2.0. DBSCAN parameters set the clustering scale (ε) and minimum point support (ζ),

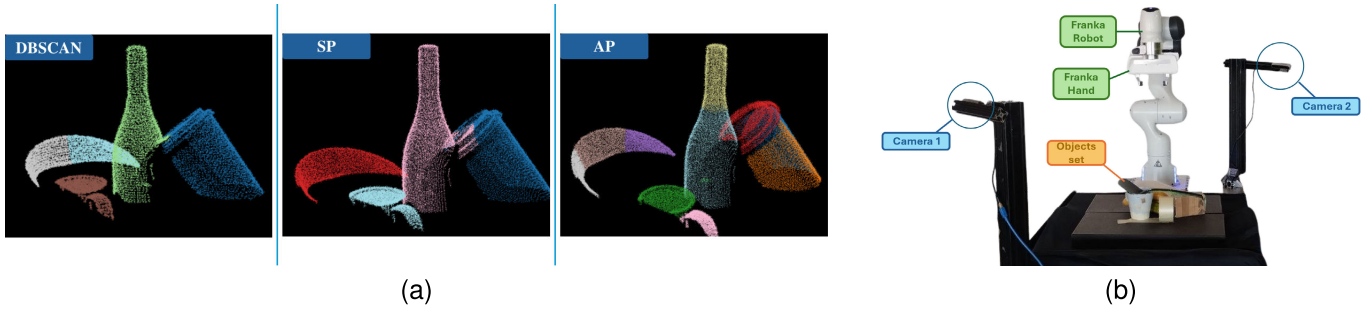

Fig. 9. (a) The different results obtained from the three tested algorithms: DBSCAN, spectral clustering (SP), and affinity propagation (AP). (b) The real-world environment designed for the experiments.

MVBB parameters (h, μ_b, ρ_b, v_M) regulate the granularity of the box decomposition, and the scoring parameters map closure-collision density through a sigmoid (k, δ_C) and enforce basic orientation/size feasibility. In particular, δ_θ and δ_λ keep the same physical meaning as in GILP 2.0 (Eq. 11) but are instantiated here with values consistent with our end-effector and setup, while $\ell_{\min}$ is introduced in GILP 3.0 to reject candidates that are too thin along the pinch direction and thus avoid degenerate grasps in clutter. We set $\ell_{\min} = 0.015$ m based on the Franka Hand finger/contact geometry as a conservative minimum reliable pinch thickness, which filters thin MVBB candidates that under partial views and depth noise often produce edge contacts and slip during closure.

IV. Validation

To address the challenge of grasping in cluttered environments, GILP 3.0 uses an initial segmentation of the fused point cloud combined with a data-driven approach to accurately determine the best grasping pose, as described in Sec. III. To evaluate the validity of our method, we divided the experiment into two parts. The first part (Sec. IV-A) performs a design-time comparison of three standard clustering algorithms, DBSCAN, SP, and AP (see Sec. III-A), in order to select a suitable segmentation module for our pipeline; it is not intended as a novel contribution. The second part (Sec. IV-B) evaluates the effectiveness of grasping cluttered objects using the complete GILP 3.0 system.

A. Clustering Tests

Setup: The clustering tests evaluated the performance of three algorithms: DBSCAN, AP, and SP. The entire experiment was carried out in a simulated environment built using Gazebo [42]. Different sets of objects were used to test the validity of the three clustering algorithms. These sets were designed progressively, starting from simple configurations to more complex arrangements of common household objects such as bottles, cups, and dishes, as shown in Fig. 7d. Point-cloud acquisition in simulation and the clustering tests were implemented in Python/ROS (RViz for visualization). Experiments were run on a laptop workstation with an Intel i7-11800H CPU and an NVIDIA GeForce RTX 3060 GPU.

Results: As illustrated in Fig. 8, the clustering performance comparison evaluates two key parameters: the number of clusters identified and the execution time in seconds. A visual

Fig. 10. (c) The 12 experimental sets, consisting of both simple and complex configurations, in terms of occlusion, designed to evaluate the algorithm under different conditions.

representation of these segmentation outcomes is shown in Fig. 9a, where it can be observed that the portion of the point cloud under examination consists of approximately 4 objects. DBSCAN identified 5 clusters, SP identified 4 clusters, and AP identified 10 clusters. Regarding execution time, DBSCAN completed the clustering process in 0.5 seconds, while SP and AP required 1.5 and 10 seconds, respectively.

Discussion: The results highlight notable differences in clustering behavior. DBSCAN and SP produced a limited number of clusters, which is preferable for structured segmentation. In contrast, AP produced more clusters, leading to fragmented segmentation. Furthermore, SP requires knowledge of the number of clusters to be identified. For this reason, DBSCAN is preferred as it also offers greater generality. In terms of execution time, DBSCAN demonstrated superior efficiency due to its computational complexity of $O(n\ log(n))$, making it the most suitable choice for real-time applications.

Among the tested methods, DBSCAN emerged as the most efficient clustering algorithm, both in terms of segmentation quality and execution time. This comparison is therefore used only to motivate the choice of DBSCAN as the segmentation backend for the grasping tests, rather than as a standalone clustering contribution.

B. Real-World Experiments

Setup: The subsequent step after IV-A, aimed at selecting the most suitable segmentation algorithm, involves conducting

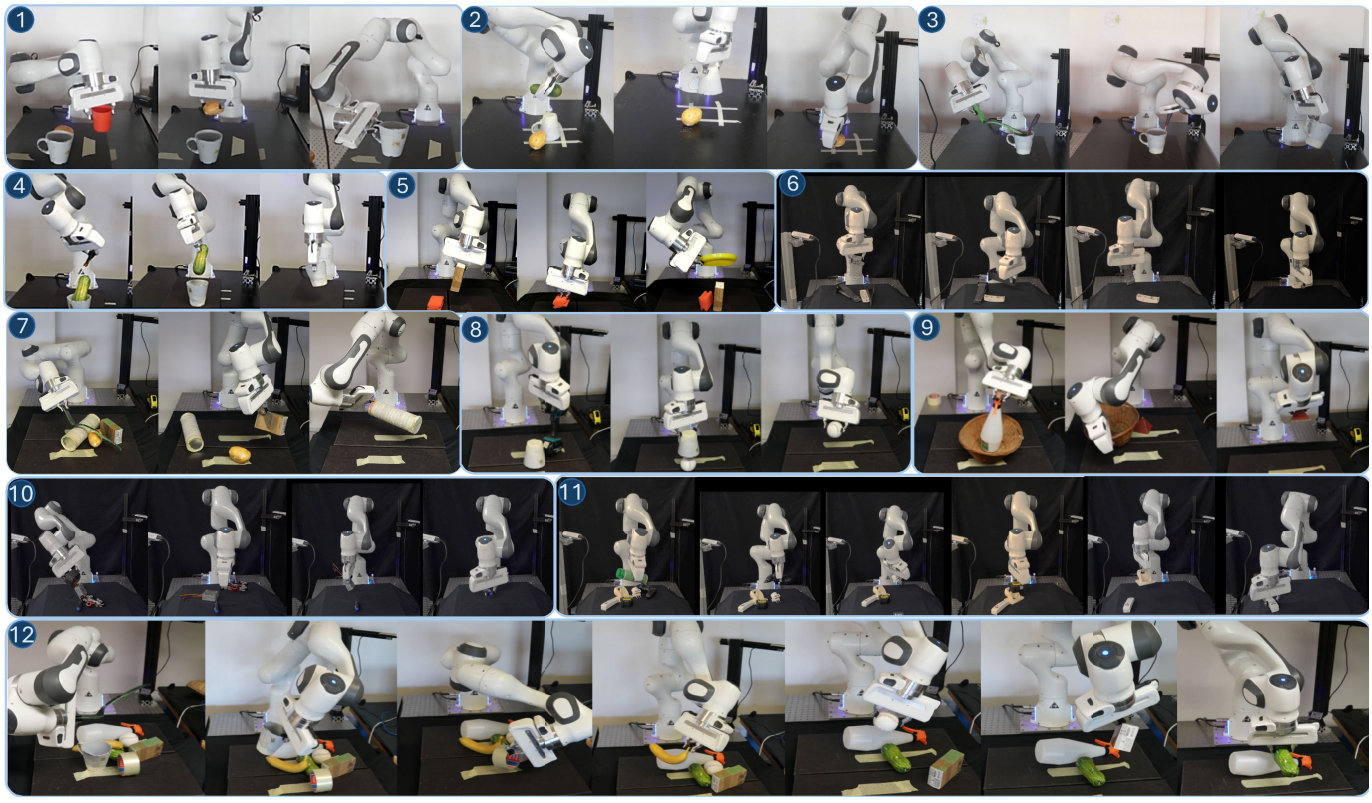

Fig. 11. The 12 real-robot cluttered sets used for evaluation, ordered from simpler to more challenging configurations. Early sets contain separated or lightly interacting objects; mid sets increase inter-object contact and tight spacing; later sets progressively increase occlusion and scene density. Set 12 represents the most challenging condition, where multiple objects are packed with tight clearances and mutual occlusions, making collision-free approach directions scarce and requiring closed-loop reacquisition/re-planning.

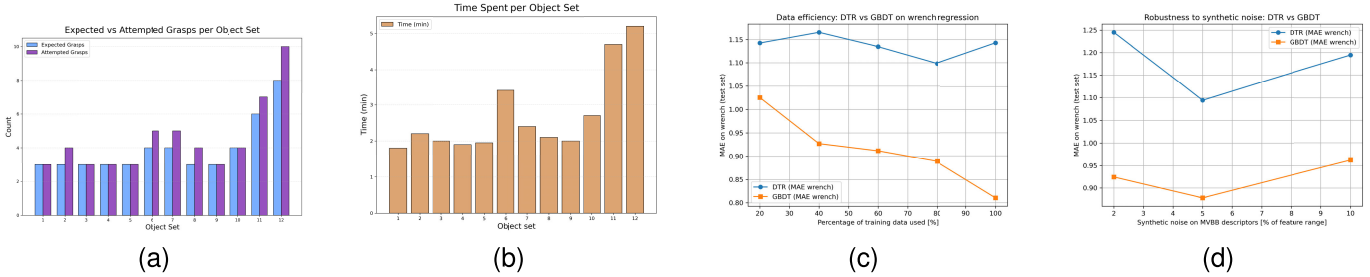

Fig. 12. (a) In cyan the grasp per-object success rates across different sets of items built, and in purple the grasp attempts of each set. (b) Grasping execution time of each set (c) data-efficiency analysis for wrench regression: the GBDT achieves consistently lower MAE than the DTR across different fractions of the training demonstrations. (d) Robustness to synthetic perturbations of the MVBB descriptors: the GBDT degrades more gracefully as noise increases, reflecting improved stability under realistic point cloud distortions.

grasping tests. These tests are designed to validate GILP 3.0's ability to accurately determine effective grasping poses, even in complex and unknown scenarios. These tests were performed in both simulated environments and real-world scenarios. The simulation phase served exclusively to validate the computed grasping poses and assess the reliability of the proposed approach. Two object sets were designed in *Gazebo* [42], starting with a simple composition (e.g., cluttered scenarios with objects not too close to each other) and progressing to more complex configurations. Once the point cloud was acquired and segmented, the grasping poses were computed.

For real-world experiments, the setup included two *Intel RealSense D415 cameras* [43] for point cloud acquisition and a *Franka Emika Panda* [41], [44] robotic arm equipped with a *Franka Emika Hand* gripper, as shown in Fig. 9b. To evaluate effectiveness, nine object sets were designed, as shown in Fig. 10, organized from the simplest (Set 1) to the most complex in terms of composition and grasping difficulties (Set 9). These tests aim to clear the working surface by sequentially removing the constructed sets of items. A failure occurs when an object is not successfully grasped throughout the whole test. The test take into account also the number of attempts to grasp an object. The main differences between simulated and real-world grasping experiments include the need for camera calibration, filtering real-world data to remove noise, and the actual execution of grasping actions. First, the images captured by the two Intel RealSense D415 cameras [43] were calibrated with an ARuco marker to align the camera frames with the workspace frame. GILP 3.0 then captured the point cloud Q, reduced it to $\bar{Q}$ by discarding points below

$z_{\max}/2$ as described in Sec. III-A, and applied a Voxel Grid filter from the Point Cloud Library (PCL) [45] to remove invalid points, homogenize density, and attenuate outliers. The filtered cloud was finally processed by DBSCAN (Sec. IV-A) to obtain the segmentation C^*, which was used to compute grasping poses, with all computations performed in Python. The system was executed with the same laptop used for IV-A. Finally, trajectory planning, as described in III-D, was managed via MoveIt! [46] using a Python script.

Results: We tested 12 cluttered sets comprising 47 objects in total. Their execution are shown in Fig. 11. Out off 47 objects, 45 were successfully grasped in 52 attempts, and per-object success rate of 95.7%. The robot cleared 11 out of 12 scenes, and clearing rate of 91.7%. Fig. 12a reports, for each set, the expected number of grasps (equal to the number of objects) and the number of attempted grasps. The clearing time per set ranged from 1.8 to 5.20 min (Fig. 12b).

Pose prediction averaged 0.8 s per attempt, while the execution time per grasp did not exceed 1 minute. Beyond the real-robot evaluation, we performed an offline comparison of the original DTR and the boosted regressors. For pose regression, both models achieve comparable accuracy on the ideal demonstration dataset, as expected given the geometric regularity of the cuboidal proxies. In contrast, the wrench regression task reveals a clear numerical advantage for the GBDT: on the full test set, the mean absolute error (MAE) decreases from 1.14 to 0.81, and Fig. 12c shows that this gap is preserved across all fractions of the available demonstrations, confirming better data efficiency. Furthermore, Fig. 12d reports the behaviour under synthetic perturbations of the MVBB descriptors (2)–10% of the feature range), where the GBDT consistently maintains a lower MAE than the DTR, indicating increased robustness to geometric noise. For reproducibility, we use a fixed train/test split and a fixed pseudo-random seed. The histogram-based boosted regressor is trained with a squared-error objective; we set the learning rate to 0.05, limit the maximum tree depth to 3, and use 200 boosting iterations. Feature values are quantized into 255 histogram bins, and we fix the random seed (`random_state=0`) to make the results deterministic. *Discussion:* The variation in completion time across the different sets suggests that both object composition and spatial configuration significantly affected grasp efficiency. In particular, the densest scene (set 12) yielded the longest execution time and the highest number of grasp attempts, reflecting a more challenging clutter regime. Conversely, sets requiring fewer attempts and shorter completion times likely provided more favorable object layouts for collision-free approaches and stable grasps. Furthermore, scenes containing more complex, non-convex objects (e.g., gamepad-like geometries and mechanical components) increased occlusions and collision risk in tight arrangements, making closed-loop reacquisition and re-planning especially beneficial. The system dynamically adapted to different environments, updating the scene after each grasp to ensure subsequent actions were based on the most recent information. This adaptability was essential to maintain grasping efficiency and reliability. These results were achieved using a grasp policy trained from a compact set of human demonstrations, confirming the ability of the proposed non-deep, data-driven model to generalize to diverse everyday objects in clutter. While, for pose regression, the DTR and the boosted model achieve comparable accuracy on the ideal demonstration dataset, the wrench regression task reveals a clear advantage for the GBDT, as shown in Fig. 12c, 12d. Its prediction error remains consistently lower across different fractions of the available demonstrations and under synthetic perturbations of the MVBB descriptors, indicating better data efficiency and robustness to geometric noise. Replacing the DTR with the two GBDT models mainly improved wrench prediction, resulting in more reliable first-attempt grasps. As detailed in Sec. IV-B, the point cloud processing pipeline already suppresses most isolated outliers and local depth noise. However, transient calibration drift of one camera in the dual D415 setup can warp the fusion of the two point clouds and distort the geometry of smooth objects, creating artificial ridges, as in the potato example of Fig. 13, which GILP 3.0 may incorrectly interpret as graspable handles. Future work will focus on increasing robustness to these calibration-induced artifacts, for example through stronger filtering, consistency checks between the two views, and regular re-calibration.

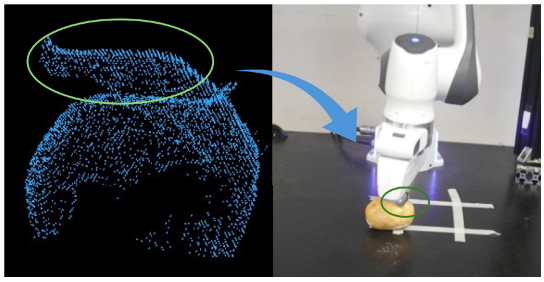

Fig. 13. Illustration of calibration quality in the dual real sense setup. The figure shows a failed and a successful attempt to grasp a smooth potato in the third setup. In the failed case, a transient calibration drift between the two cameras warps the fused point cloud and creates an artificial ridge on the potato surface, leading GILP 3.0 to select a spurious grasp candidate.

V. Conclusion

GILP 3.0 represents a significant advancement in robotic grasping, particularly for cluttered and unstructured environments. By combining clustering algorithms, bounding box decomposition, and human demonstrations, the algorithm effectively identifies grasping poses in complex scenarios. Additional policies and heuristics enhance its ability to handle overlapping objects or those in challenging positions. Overall, GILP 3.0 demonstrated notable improvements in computational efficiency, grasping accuracy, and flexibility. The algorithm successfully managed a variety of object poses, dynamically adapted to environmental changes, and efficiently operated in cluttered sets of objects. These capabilities were validated through rigorous testing, achieving a per-object success of 95.7% and a clearing rate of 91.7%, confirming its effectiveness in real-world applications. Despite these strengths, severe miscalibration or depth corruption in the fused point cloud from the dual RealSense setup can distort object geometry and yield spurious grasp candidates, a limitation that should be addressed in future iterations. Nevertheless, the advancements introduced in GILP 3.0 represent a

meaningful step forward in robotic grasping, offering robust, efficient, and adaptable solutions for diverse industrial and domestic tasks. Its ability to deliver high-quality grasps in challenging conditions positions it as a valuable tool for real-world deployment in complex environments.

REFERENCES

[1] A. Bicchi, "On the closure properties of robotic grasping," *Int. J. Robot. Res.*, vol. 14, no. 4, pp. 319–334, Aug. 1995.

[2] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automation. Symp.*, vol. 1, Mar. 2000, pp. 348–353.

[3] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 886–900, Jun. 2012.

[4] G. J. Pollayil, M. J. Pollayil, M. G. Catalano, A. Bicchi, and G. Grioli, "Sequential contact-based adaptive grasping for robotic hands," *Int. J. Robot. Res.*, vol. 41, no. 5, pp. 543–570, Apr. 2022.

[5] Y. Bi, S. Lin, and C. Yang, "Grasp planning based on geometric fitting with high-quality initial configurations," in *Proc. 10th Int. Conf. Electr. Eng., Control Robot. (EECR)*, Guangzhou, China, Mar. 2024, pp. 98–103.

[6] A. Palleschi et al., "Grasp it like a pro 2.0: A data-driven approach exploiting basic shape decomposition and human data for grasping unknown objects," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 4016–4036, Oct. 2023.

[7] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—A survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[8] J. Mahler et al., "Dex-net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1957–1964.

[9] A. Ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *Int. J. Robot. Res.*, vol. 36, nos. 13–14, pp. 1455–1473, Dec. 2017.

[10] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 2901–2910.

[11] C. D. Santina et al., "Learning from humans how to grasp: A data-driven architecture for autonomous grasping with anthropomorphic soft hands," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1533–1540, Apr. 2019.

[12] C. Gabellieri et al., "Grasp it like a pro: Grasp of unknown objects with robotic hands based on skilled human expertise," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2808–2815, Apr. 2020.

[13] H. Zhang, X. Zhou, X. Lan, J. Li, Z. Tian, and N. Zheng, "A real-time robotic grasping approach with oriented anchor box," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 5, pp. 3014–3025, May 2021.

[14] R. Xu, F.-J. Chu, and P. A. Vela, "GKNet: Grasp keypoint network for grasp candidates detection," *Int. J. Robot. Res.*, vol. 41, no. 4, pp. 361–389, Apr. 2022.

[15] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 4679–4684.

[16] A. Cordeiro, L. F. Rocha, C. Costa, P. Costa, and M. F. Silva, "Bin picking approaches based on deep learning techniques: A state-of-the-art survey," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Santa Maria da Feira, Portugal, Apr. 2022, pp. 110–117.

[17] M. Nieuwenhuisen et al., "Mobile bin picking with an anthropomorphic service robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 2327–2334.

[18] S. Noh, J. Kim, D. Nam, S. Back, R. Kang, and K. Lee, "GraspSAM: When segment anything model meets grasp detection," 2024, *arXiv:2409.12521*.

[19] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-GraspNet: Efficient 6-DoF grasp generation in cluttered scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 13438–13444.

[20] J. Zhang et al., "DexGraspNet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes," in *Proc. 8th Annu. Conf. Robot Learn.*, 2024, p. 28.

[21] D. Katz et al., "How can robots succeed in unstructured environments?," in *Proc. RSS Workshop Robot Manipulation, Intell. Hum. Environments*, 2008, p. 6.

[22] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 3102–3109.

[23] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2008, pp. 189–196.

[24] D.-C. Hoang, J. A. Stork, and T. Stoyanov, "Context-aware grasp generation in cluttered scenes," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 1492–1498.

[25] Y. Yang, H. Liang, and C. Choi, "A deep learning approach to grasping the invisible," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2232–2239, Apr. 2020.

[26] E. Zhou et al., "Mobile SAM-based motion target detection," in *Proc. 43rd Chin. Control Conf. (CCC)*, Jul. 2024, pp. 7854–7859.

[27] Y. Xiong et al., "EfficientSAM: Leveraged masked image pretraining for efficient segment anything," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 16111–16121.

[28] D. Son, "Grasping as inference: Reactive grasping in heavily cluttered environment," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7193–7200, Jul. 2022.

[29] A. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, "Strategies for human-in-the-loop robotic grasping," in *Proc. 7th ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, Mar. 2012, pp. 1–8.

[30] M. Kopicki, S. I. Ansary, S. Tolomei, F. Angelini, M. Garabini, and P. Skrzypczyłski, "Underactuated dexterous robotic grasping with reconfigurable passive joints," *IEEE Robot. Autom. Lett.*, vol. 10, no. 1, pp. 48–55, Nov. 2024.

[31] S. Lin, C. Zeng, and C. Yang, "Robot grasping based on object shape approximation and LightGBM," *Multimedia Tools Appl.*, vol. 83, no. 3, pp. 1–17, Jan. 2024.

[32] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1628–1633.

[33] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3D object recognition in cluttered scenes with local surface features: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2270–2287, Nov. 2014.

[34] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.

[35] R. Xu and D. WunschII, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[36] T. N. Tran, K. Drab, and M. Daszykowski, "Revised DBSCAN algorithm to cluster data with dense adjacent clusters," *Chemometric Intell. Lab. Syst.*, vol. 120, pp. 92–96, Jan. 2013.

[37] D.-Y. Xia, F. Wu, X.-Q. Zhang, and Y.-T. Zhuang, "Local and global approaches of affinity propagation clustering for large scale data," *J. Zhejiang Univ.-Sci. A*, vol. 9, no. 10, pp. 1373–1381, Oct. 2008.

[38] M. C. V. Nascimento and A. C. P. L. F. de Carvalho, "Spectral methods for graph clustering—A survey," *Eur. J. Oper. Res.*, vol. 211, no. 2, pp. 221–231, Jun. 2011.

[39] M. Bonilla, D. Resasco, M. Gabiccini, and A. Bicchi, "Grasp planning with soft hands using bounding box object decomposition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 518–523.

[40] A. T. Pas, *Using Geometry To Detect Grasp Poses 3D Point Clouds*. Cham, Switzerland: Springer, 2018.

[41] S. Haddadin et al., "The franka emika robot: A reference platform for robotics research and education," *IEEE Robot. Autom. Mag.*, vol. 29, no. 2, pp. 46–64, Jun. 2022.

[42] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Sep. 2004, pp. 2149–2154.

[43] F. Lourenço and H. Araujo, "Intel RealSense SR305, D415 and l515: Experimental evaluation and comparison of depth estimation," in *Proc. 16th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2021, p. 8.

[44] R. A. B. Petrea, M. Bertoni, and R. Oboe, "On the interaction force sensing accuracy of franka emika panda robot," in *Proc. 47th Annu. Conf. IEEE Ind. Electron. Soc.*, Toronto, ON, Canada, Oct. 2021, pp. 1–6.

[45] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.

[46] S. Chitta, I. Sucan, and S. Cousins, "MoveIt! [ROS topics]," *IEEE Robot. Autom. Mag.*, vol. 19, no. 1, pp. 18–19, Mar. 2012.

Open Access funding provided by 'Università di Pisa' within the CRUI CARE Agreement