IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

I Get the Hang of It! A Learning-Free Method to Predict Hanging Poses for Previously Unseen Objects

Wanze Li[1], Lexin Pan[1], Boren Jiang[1], Yuwei Wu [1], Weixiao Liu [1,2] Gregory S. Chirikjian[1,3]

***Abstract*— The action of hanging previously unseen objects remains a challenge for robots due to the multitude of object shapes and the limited number of stable hanging arrangements. This paper proposes a learning-free framework that enables robots to infer stable relative poses between the object being hung (object) and the supporting item (supporter). Our method identifies potential hanging positions and orientations on previously unseen supporters and objects by analyzing the hanging mechanics and geometric properties. An evaluation policy is designed to match potential hanging positions and directions and to optimize the relative hanging poses. Experiments were conducted in both simulation and real-world scenarios. The success rates of our strategy outperform the state-of-the-art baseline method. The proposed method was also tested on unhangable pairs of objects and supporters and results show that our algorithm can reject false positive hanging properly. Finally, we ran experiments under different scanning conditions. Experimental results indicate that although the success rate decreases as the quality of the scan decreases, it remains at a high level. More details and Supplementary Material can be found in our project webpage and our RA-L paper.**

I. Introduction

Over the past few decades, the field of robotics has made tremendous strides, leading to a significant impact on our daily lives. Household robotics, a critical sector within the robotics industry, has gained increasing attention. It is widely believed that in the near future, robots will be capable of assisting humans in performing routine tasks, such as housework and elderly care [1]. However, certain everyday tasks, such as hanging objects, remain challenging for robots. Hanging objects is a routine activity in our daily lives, involving tasks such as placing mugs on mug trees, clothes on racks, and pans on hooks. Therefore, enabling robots to hang everyday objects on various supporting items could increase convenience and improve domestic robotics. However, the diverse geometry of everyday objects and the limited number of stable hanging postures bring significant challenges for robotic hanging, especially in unseen environments.

Compared to popular topics such as grasping and planning, the problem of hanging objects with robots has received relatively little attention. Object hanging is typically used to validate methods for other robotics problems [2]–[10], like objects placement and affordance representation. For instance, Jiang et al. [2] proposed a Support Vector Machines (SVM) based supervised learning algorithm to identify object placement. The algorithm fits a function of features to estimate the stability and orientation of a placement. They tested their method via hanging objects on hooks, but the rate of success was only 40%. These works generally focus on a specific pair of object and supporter, such as hanging a mug on a mug tree, lacking universality in hanging arbitrary objects.

Fig. 1. Examples of daily objects hanging with single contact point

In recent years, some groups have developed methods that specifically address the problem of hanging objects [11]–[13]. You et al. [11] proposed a reinforcement learning based approach that can hang a wide range of objects on various supporting items. Takeuchi et al. [12], [13] used a Generative Adversarial Network (GAN) to generate 3D models with different shapes. Then they trained a deep neural network with these 3D models to estimate hanging points of an unknown object. Both of these methods require training on large amounts of data.

Existing research on objects hanging mainly uses learning-based methods. These methods require extensive training data and often have limited adaptability to novel situations that differ significantly from the training data. In contrast, as shown in Fig. 1 when humans hang objects we can recognize and extract suitable geometric shapes for hanging such as hooks and holes on both the object and the supporter [14]. To adapt the philosophy of human perception and action in hanging objects, in this paper we propose a three-stage learning-free algorithm capable of predicting a stable hanging pose for previously unseen objects and supporters. The workflow of the proposed method is shown in Fig. 2. The algorithm takes the mesh model of the object and the point cloud of the supporter as inputs, both of which can be obtained by 3D scanning. Then the mesh and point cloud are processed separately to detect the suitable hanging positions and hanging directions on both object and supporter. Each position is represented as a 3D keypoint and each direction

[1] Department of Mechanical Engineering, National University of Singapore, Singapore
[2] Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD, USA
[3] Department of Mechanical Engineering, University of Delaware, Newark, DE, USA

† This work was supported by the National Research Foundation, Singapore, under its Medium Sized Centre Programme - Centre for Advanced Robotics Technology Innovation (CARTIN), subaward A-0009428-08-00, and AME Programmatic Fund Project MARIO A-0008449-01-00. The authors thank Xin Meng in NUS for programming assistance in the experiments.

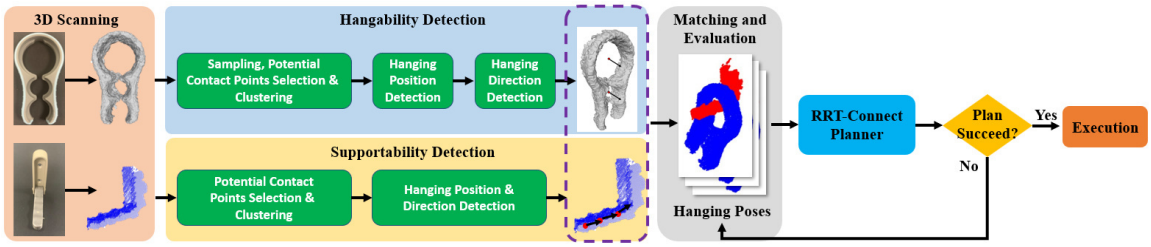

Fig. 2. The workflow of the proposed method. The entire procedure includes 3D scanning, hangability detection, supportability detection and matching and evaluation. In this figure, red points represent hanging positions and black arrows represent hanging directions.

is represented as a key-vector. Subsequently, all keypoints and key-vectors on the object and supporter are aligned with each other to identify all possible matches. These matches are evaluated and ranked to determine proper hanging poses. Finally the hanging poses prediction method is integrated with an RRT-connect planner [15] for hanging execution. Compared to existing methods for robotic hanging objects, the proposed work eliminates the need for training data. Moreover, our method is based on the analysis of the mechanics and geometric properties, resulting in greater accuracy in unseen objects and supporters.

II. Methods

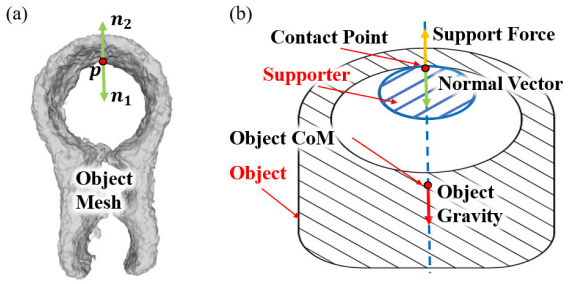

Fig. 3. (a) The direction of the normal vector. $\mathbf{n}_1$ and $\mathbf{n}_2$ are two possible directions of the normal vector at point $\mathbf{p}$ on the mesh. (b) The cross-section of the object and the supporter in a stable hanging pose.

A. Support Force Analysis

When an object is stably hung on a supporter, the support force must balance the gravity of the object. Since there is only one contact point, the support force should be colinear with the center of mass (CoM) of the object. Therefore, as illustrated in Fig. 3b, the direction of the support force on the object should be opposite to the normal vector pointing outside at the contact point [16]. A possible contact point $\mathbf{p}_h$ on object satisfies:

$$\frac{(\mathbf{h}_c - \mathbf{p}_h)\cdot \mathbf{n}_h}{\|\mathbf{h}_c - \mathbf{p}_h\|} = 1 \tag{1}$$

where $\mathbf{n}_h$ is the normal vector at $\mathbf{p}_h$ with a unit length, $\mathbf{h}_c$ is the 3D position of the object CoM. In addition, the direction of the support force offered by the supporter is opposite to the gravity and along the normal vector at the contact point on the supporter. Therefore, a potential contact point $\mathbf{p}_s$ on supporter satisfies:

$$-\mathbf{g}\cdot \mathbf{n}_s = 1 \tag{2}$$

Here, $\mathbf{g}$ is the direction of gravity and $\mathbf{n}_s$ is the unit normal vector at $\mathbf{p}_s$. While these conditions are necessary but not sufficient, they are enough to provide a proper initial guess for the following procedures. In this paper, the z direction is antiparallel with the gravitational direction.

B. Hangability Detection

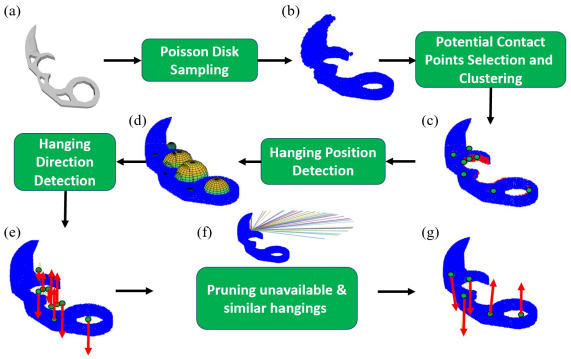

Fig. 4. The procedure for hangability detection.

The goal of hangability detection is to find all possible structures on the object that can afford hanging and represents each of them as a position and a direction. The workflow is shown in Fig. 4.

Sampling, Potential Contact Points Selection and Clustering: To leverage the rules described in Eq.1, a point cloud (Fig. 4b) $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^{N} \in R^{3\times N}$ is sampled from the mesh via Poisson disk sampling [17]. $N = 10000$ is the number of points in the point cloud. Then points satisfying the following relationship are considered as potential contact points (red points in Fig. 4c) and form a subset $\mathbf{p}_h$ of $\mathbf{H}$:

$$\mathbf{P}_h \triangleq \{\mathbf{h}\in\mathbf{H} | \frac{(\mathbf{h}_c - \mathbf{h})\cdot \mathbf{n}_h}{\|\mathbf{h}_c - \mathbf{h}\|} > \alpha_h, \mathbf{h}_c = \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i\} \tag{3}$$

Here, $\alpha_h \in (0, 1]$ is a threshold. To reduce the computation and accelerate following steps, $\mathbf{P}_h$ is then clustered using the Mean Shift Algorithm. Only clusters with enough number of points are kept for the following steps, *i.e.:*

$$\mathbf{P}_h^i \subseteq \mathbf{P}_h, i = 1, ..., M, s.t. |\mathbf{P}_h^i| > l_c \tag{4}$$

M is the number of remained clusters and l_c is a threshold. Then the algorithm selects the closest point to the centroid of each cluster as its substitute. These points (green dots in Fig. 4c) are denoted as $\{\mathbf{p}_h^i\}_{i=1}^{M}$.

Hanging Position Detection: As shown in Fig. 3b, in the stable state, the supporter is below the object at the contact

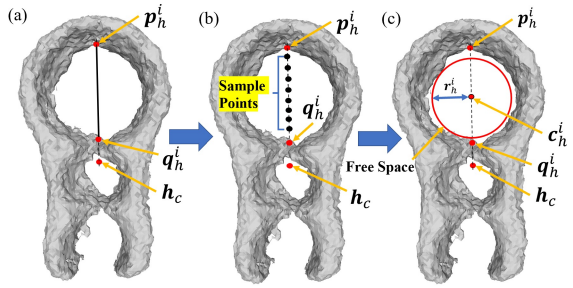

Fig. 5. The procedure of hanging position detection on the object.

point. Consequently, there must be sufficient free space near the contact point ($\mathbf{p}_h^i$) to accommodate the supporter. The next step is to detect the hanging position, which is defined as the center of the free space. The procedure of hanging position detection is shown in Fig. 5. For every point $\mathbf{p}_h^i \in \{\mathbf{p}_h^k\}_{k=1}^M$, a ray is cast toward object CoM $\mathbf{h}_c$. The intersection point between the ray and mesh of object is denoted as $\mathbf{q}_h^i$ (see Fig. 5a). If there is no interaction, $\mathbf{q}_h^i$ is set to $\mathbf{h}_c$. Next, N_h points (black dots in Fig. 5b) are evenly sampled along the line segment $\overline{\mathbf{p}_h^i \mathbf{q}_h^i}$. The farthest sampled point from the object point cloud $\mathbf{H}$ is selected as the hanging position, denoted as $\mathbf{c}_h^i$ (see Fig. 5c). The distance from $\mathbf{c}_h^i$ to $\mathbf{H}$ is denoted as free space radius (FSR), represented as r_h^i. As shown in Fig.4d and Fig. 5c, the free space between $\mathbf{p}_h^i$ and $\mathbf{q}_h^i$ can be represented as a sphere with $\mathbf{c}_h^i$ as the center and r_h^i as the radius. Finally, hanging positions and corresponding free spaces are filtered based on two conditions: (1) $r_h^i > r_\alpha$ where r_α is a threshold for FSR, to ensure a sufficiently large free space for supporter placement and (2) $S[\mathbf{c}_h^i] > 0$ where $S[\mathbf{c}_h^i]$ is the signed distance function (SDF) value at $\mathbf{c}_h^i$, ensuring that $\mathbf{c}_h^i$ is outside the object to avoid collisions. Only the $(\mathbf{c}_h^i, r_h^i)$ pairs that satisfy these conditions are kept for the following procedure.

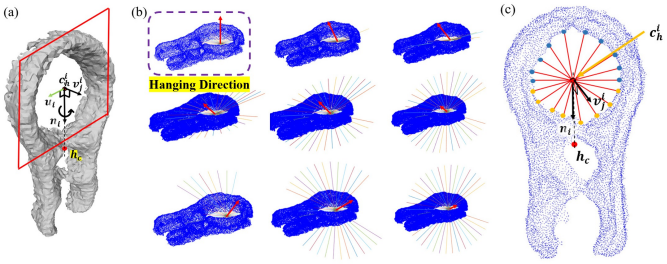

Fig. 6. (a) Schematic of hanging direction detection. (b) visualizes the hanging direction detection. (c) shows more details about how ray casting works.

Hanging Direction Detection: The next step involves determining the optimal hanging directions for hanging positions ($\mathbf{c}_h^i$) identified in the previous step. II-A shows that the vector $\mathbf{n}_i = \frac{(\mathbf{q}_h^i - \mathbf{p}_h^i)}{||\mathbf{q}_h^i - \mathbf{p}_h^i||}$ in Fig.6 should coincide with the gravity direction at a stable hanging pose. Thus, the only degree of freedom for the object orientation is the rotation about $\mathbf{n}_i$. Therefore, possible hanging directions at $\mathbf{c}_h^i$ can be enumerated by $\mathbf{v}_j^i = R_n(\beta_j)\mathbf{u}_i, j = 1, ..., N_p$, where $\mathbf{u}_i$ is a random vector that is perpendicular to $\mathbf{n}_i$ (see Fig.6a and b for more details). $R_n(\beta_j)$ rotates $\mathbf{u}_i$ about $\mathbf{n}_i$ for β_j, $\beta_j = \frac{j\pi}{N_p}$ and N_p is a predefined value.

As shown in Fig.6b, the optimal hanging direction is determined by casting rays from $\mathbf{c}_h^i$ within the plane that is vertical to $\mathbf{v}_j^i$. Then the $\mathbf{v}_j^i$ with the most number of rays that intersect with the object is selected as the hanging direction. The objective for hanging direction detection is to find the most proper direction for the supporter to 'insert' into the object at each hanging position. Fig. 3b displays that if the object is sliced by a plane that is perpendicular to the hanging direction at the hanging position $\mathbf{c}_h^i$, most range around $\mathbf{c}_h^i$ should be surrounded by the object surface to provide enough contact with the supporter. Therefore, the hanging direction can be determined by searching the sampled vector $\mathbf{v}_j^i$ at $\mathbf{c}_h^i$ with most object surfaces around. Concretely, for each potential hanging direction $\mathbf{v}_j^i$ at $\mathbf{c}_h^i$, N_r rays (thin lines in Fig.6b) are cast uniformly from $\mathbf{c}_h^i$ within the plane (marked red in Fig.6a) perpendicular to $\mathbf{v}_j^i$. N_r is a predefined sampling number. More rays intersecting with the model means that the hanging position is more completely encircled by the object in that ray surface, making the corresponding hanging direction $\mathbf{v}_j^i$ more appropriate for hanging. For instance, the hanging of the object along the direction $\mathbf{v}_j^i$ labelled by a dashed box in Fig.6b is better than other sampled directions. Then for each sampled hanging direction $\mathbf{v}_j^i$, we denote the proportion of rays that contact with object mesh as m^j. Proportions of rays that intersect with the object model above (blue points in Fig.6c) and below (orange points in Fig.6c) $\mathbf{c}_h^i$ at the stable hanging pose are represented as m_1^j and m_2^j, respectively. The direction $\mathbf{v}_j^i$ with largest m^j is selected as the hanging direction for $\mathbf{c}_h^i$, denoted as $\mathbf{v}_h^i$. Corresponding m^j, m_1^j and m_2^j are denoted as m_i, m_i^1 and m_i^2, respectively. To further remove unfeasible hangings, only $(\mathbf{c}_h^i, \mathbf{v}_h^i)$ pairs that satisfy

$$m_i^1 < \alpha_1 \ and \ m_i^2 < \alpha_2 \tag{5}$$

are retained for following steps. α_1 and α_2 are two thresholds. Here, α_1 is larger than α_2 because the portion of the object above the hanging position actually makes contact with the supporter during hanging.

Pruning Similar Hangings: Finally, potential hangings are filtered by merging the hanging positions and directions with similar free space size or hanging direction. Two pairs of potential hanging positions and directions $(\mathbf{c}_h^i, \mathbf{v}_h^i)$ and $(\mathbf{c}_h^j, \mathbf{v}_h^j)$ with FSRs r_h^i and r_h^j are considered similar if they satisfy:

$$\frac{\left|r_h^i - r_h^j\right|}{max(r_h^i, r_h^j)} < r_\beta \ and \ \left|\mathbf{v}_h^i \cdot \mathbf{v}_h^j\right| > \beta \tag{6}$$

r_β and β are two user-defined thresholds. If the conditions are met, the hanging position-direction pair with the higher m_i value is retained.

C. Supportability Detection

Similar to hangability, supportability is defined as the ability to provide support to an object hanging on it. As shown in Fig.2, the supportability detection is to identify

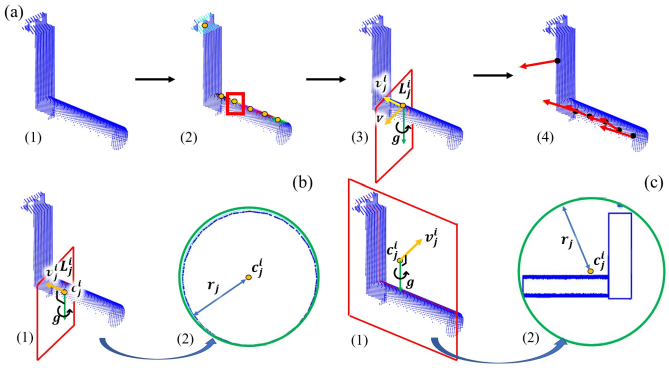

Fig. 7. (a) The procedure of supportability detection on the supporter. (b) and (c) shows examples of sliced supporter point clouds with different slicing directions.

proper hanging positions and directions on the supporter. The detailed procedure is shown in Fig.7a. Intuitively, a smaller cross-sectional area (blue shaded in Fig. 3b) of the supporter component under the object makes it easier to place the supporter along the normal vector of that cross-section. For instance, hanging objects along the orange arrow in Fig. 7b is easier than along the orange arrow in Fig. 7c. Points on the supporter point cloud are firstly selected according to Eq.2 and clustered (Fig.7a.2). Then for each cluster, the algorithm slices the supporter point cloud along different directions (Fig.7a.3) at the center of every cluster. The direction with the minimum cross-sectional area is selected as the hanging direction and the centroid of cross section is selected as the hanging position.

Specifically, given a point cloud of supporter $\mathbf{S}$ with M points, the normal vectors of potential contact points must point in the opposite direction of gravity according to Eq. 2. Hence, points with normal vectors that satisfy $-\mathbf{g} \cdot \mathbf{n}_s^i > \alpha_s$ are selected as potential contact points, where $\mathbf{n}_s^i$ is the normal vector at the point for checking. $\alpha_s \in (0, 1)$ is a hyperparameter. Similar to the process of object, potential contact points are clustered into the M_c clusters, expressed as $\{\mathbf{P}_s^i\}_{i=1}^{M_c}$ with cluster centroids $\{\mathbf{p}_s^i\}_{i=1}^{M_c}$. Since centroids of the cluster may not be present in the $\mathbf{S}$, the algorithm adjusts each $\mathbf{p}_s^i$ as the closest point to the cluster centroid (orange dots in Fig.7a.2) in $\mathbf{S}$. Results after potential contact points selection and clustering are shown in Fig.7a.2.

To determine the hanging positions and hanging directions of supporter, the algorithm employs M_s planes $\{\mathbf{L}_j^i\}_{j=1}^{M_s}$ to slice the point cloud $\mathbf{S}$ at each cluster center $\mathbf{p}_s^i$ (see Fig.7a.3). $\mathbf{L}_j^i$ contains $\mathbf{p}_s^i$ and has the normal vector $\mathbf{v}_j^i = R_g(\beta_j)\mathbf{v}, j = 1, ..., M_s$. $\mathbf{v}$ is a vector that is perpendicular to gravitational direction and $R_g(\beta_j)$ rotates $\mathbf{v}$ about gravitational direction for $\beta_j = \frac{j\pi}{M_s}$. The algorithm slices $\mathbf{S}$ with $\mathbf{L}_j^i$ by selecting all points in $\mathbf{S}$ that have a distance to $\mathbf{L}_j^i$ smaller than a threshold d_l (see Fig.7b.1 and c.1). Then these points are clustered using the Agglomerative Clustering Algorithm [18] and the cluster contains the $\mathbf{p}_s^i$ is chosen for subsequent steps. The geometric center of the selected cluster is denoted as $\mathbf{c}_j^i$ (orange dots in Fig.7b and c). Next, a boundary sphere is generated for each cluster, centered at $\mathbf{c}_j^i$. The radius r_j of the sphere is defined as the distance from $\mathbf{c}_j^i$ to the farthest point from $\mathbf{c}_j^i$ in the cluster. Fig.7b.2 and c.2 show the cross sections formed by intercepting the boundary sphere with $\mathbf{L}_j^i$ (green circles), corresponding center $\mathbf{c}_j^i$ and radius r_j. For each $\mathbf{p}_s^i$, the normal vector of the slicing plane with the smallest boundary sphere is selected as the hanging direction (yellow arrows in Fig.7a.4) of the supporter, denoted as $\mathbf{v}_s^i$. The center of the selected boundary sphere is regarded as the corresponding hanging position (red dots in Fig.7a.4), denoted as $\mathbf{c}_s^i$. The radius of the smallest boundary sphere is denoted as r_s^i. Fig. 7b and c demonstrate the situations with small and large boundary spheres, respectively. Finally, hanging positions $\{\mathbf{c}_s^i\}_{i=1}^{M_c}$, hanging directions $\{\mathbf{v}_s^i\}_{i=1}^{M_c}$ and boundary circle radii $\{r_s^i\}_{i=1}^{M_c}$ are stored for following steps.

D. Matching and Evaluation

The final step of the proposed method involves the object/supporter matching and selecting the most appropriate hanging pose. For each hanging position-direction pair of the object ($\mathbf{c}_h^i$, $\mathbf{v}_h^i$) and the supporter ($\mathbf{c}_s^j$, $\mathbf{v}_s^j$), the algorithm first aligns the hanging directions $\mathbf{v}_h^i$ and $\mathbf{v}_s^j$. Next, the hanging position $\mathbf{c}_h^i$ on the object is moved to coincide with the supporter hanging position $\mathbf{c}_s^j$. Finally, the object is rotated about the hanging direction $\mathbf{v}_s^j$ until the vector points from $\mathbf{c}_h^i$ to the object CoM is parallel to the gravitational direction. These steps result in a transformation $T_{ij} \in SE(3)$. Matches with a free space radius $r_h^i < \alpha_m r_s^j$ are ignored, where r_h^i is the FSR of the object and r_s^j is the boundary sphere radius of the supporter. $\alpha_m \in (0, 1]$ is a hyperparameter. This ensures that the object can be placed without interference in the free space around the supporter.

In practice, there may be multiple potential hanging positions on both the object and the supporter, and not all matches are feasible. Hence, it is necessary to evaluate these matches and select viable ones. The matching evaluation is implemented by calculating and ranking the value of a cost function which considers three criteria: collision, distance to the supporter boundary and contact area.

Collision: To ensure successful hanging, it is essential that the predicted hanging pose is collision-free. If the object collides with the supporter at the predicted hanging pose T_{ij}, some points of the supporter point cloud would be inside the object mesh transformed by T_{ij}. These points are denoted as $\mathbf{S}_{collision} \subseteq \mathbf{S}$. The cost of collision between the object and the supporter is defined as:

$$S_c = |\mathbf{S}_{collision}| \tag{7}$$

Distance to Supporter Boundary: Another criterion for evaluating the hanging pose is the distance between the hanging position and the boundary of the supporter. If the hanging position $\mathbf{c}_s^j$ of the supporter is very close to the edge of the supporter, the object is more likely to fall off with disturbance. Hence, it is safer for the hanging position to be far away from the boundary of the supporter. Considering $\mathbf{c}_s^j$ and corresponding hanging direction $\mathbf{v}_s^j$, the cost of distance to the boundary can be represented as:

$$S_b = \frac{max(d_1, d_2)}{d_1 + d_2} \tag{8}$$

where d_1 and d_2 are distances from $\mathbf{c}_s^j$ to the convex hull of supporter point cloud along $\mathbf{v}_s^j$. A smaller S_b indicates that the hanging position is farther away from the boundary of the supporter, making it less prone to falling off.

Contact Area: As shown in Eq.4, for each hanging position $\mathbf{c}_h^i$ of the object, there is a corresponding cluster of potential contact points $\mathbf{P}_h^i$. The number of points in $\mathbf{P}_h^i$ represents the size of the contact area that can support stable hanging. Therefore, selecting the hanging position $\mathbf{c}_h^i$ with a larger $|\mathbf{P}_h^i|$ is more likely to result in a stable hanging. The cost of the contact area is defined as:

$$S_a = 1 - \frac{|\mathbf{P}_h^i|}{max(\{|\mathbf{P}_h^k|\}_{k=1}^{N_k})} \tag{9}$$

where N_k is the number of hanging positions on the object. A smaller S_a indicates a larger contact area between the object and the supporter, which makes it more likely to result in a stable hanging.

Finally, the total cost is computed as:

$$S_{total} = \gamma_1 S_c + \gamma_2 S_b + \gamma_3 S_a \tag{10}$$

γ_1, γ_2 and γ_3 are three hyperparameters within $(0, 1]$. The success of the hanging is more likely when the S_{total} value is smaller. Hence, all predicted hanging poses are ranked based on their S_{total} value in ascending order, the smaller S_{total} value the better.

TABLE I

HANGING ACCURACY OF SIMULATED EXPERIMENT

Methods	Accuracy	Methods	Accuracy
Ours (24V + 3V)	83.3%	Ours (24V + 1V)	81%
Ours (3V + 1V)	79.6%	Ours (1V + 1V)	71.2%
Omnihang [11]	64.4%		

III. EXPERIMENT

To evaluate the accuracy and stability of the proposed method, we conduct experiments in both simulation and real-world environments. Experiments are also conducted with meshes and point clouds reconstructed from varying numbers of viewpoints to validate the robustness of the proposed method under different scanning conditions. Algorithms are implemented in Python on a computer equipped with AMD Ryzen™ 9 5950X (3.4GHz).

A. Experiment in Simulation

The simulated experiment is conducted in an open-source simulation environment PyBullet [19] with 1600 pairs of objects and supporters from the dataset introduced in [11]. Object meshes and supporter point clouds are generated from both multiple and single depth images to test the algorithm sensitivity to the scanning quality. Then our algorithm predicts the hanging poses and selects the one with the smallest cost as the final prediction. To evaluate the hanging pose stability, the object is loaded at the predicted pose and released. A predicted hanging pose is considered stable if it is collision-free and the object doesn't fall on the ground after release. The results of the simulated experiment are shown in Table I.

B. Experiment in Real-world

To further test the proposed method, we also conduct experiments with real objects. The dataset used for experiments contains 14 daily objects for hanging and 8 supporting items. 4 out of the 14 objects for hanging are actually unhangable, which enables us to test whether the method proposes false positive predictions. All of these objects and supporters are previously unseen by robot. We build a robot system with a Franka Emika robot arm mounted with a gripper and a Primesense RGB-D camera for execution. The robot system is controlled to scan the object and supporter, manipulate, and hang the object onto the supporter. In each trial, our algorithm takes the mesh and point cloud as input and predicts potential hanging poses. Since the predicted pose may be outside the workspace of the robot arm, which is not the focus of our algorithm, ten predicted poses with the lowest S_{total} value are selected. Then the robot only executes the hanging prediction with the smallest cost among the poses that the robot can reach. If none of these ten poses is reachable, the trial is considered a failure.

TABLE II

HANGING ACCURACY OF REAL-WORLD EXPERIMENT

Object Types	Total	Hanging Detected	Execution Success	Rate of Success
Hangable	80	77	61	76.3%
Not Hangable	32	1	0	96.9%

TABLE III

HANGING ACCURACY OF OBJECTS SCANNED IN DIFFERENT NUMBER OF VIEWS

View Number	Total Trials	Hanging Detected	Execution Success	Rate of Success
10	10	9	9	90%
3	10	10	8	80%
1	10	10	7	70%

TABLE IV

HANGING ACCURACY OF SUPPORTERS SCANNED IN DIFFERENT NUMBER OF VIEWS

View Number	Total Trials	Hanging Detected	Execution Success	Rate of Success
3	16	14	14	87.5%
1	16	15	11	68.8%

Three distinct experiments are launched to validate the accuracy and stability of our method. (1) Accuracy validation: We first run an experiment with 14 objects for hanging and 8 supporting items which gives a total of 80 trials for feasible hangings and 32 trials for unfeasible hangings. For each trial, the object is scanned from 10 different views and the supporter is scanned from 3 views. The outcomes are shown in Table II. (2) Sensitivity to object scanning: In the second experiment, we assess our method's resilience to variations in object mesh quality by scanning the object from different numbers of viewpoints. Specifically, we conduct three groups of experiments with 10 hangable objects and 1 supporter. In each group, objects are scanned from varying numbers of viewpoints (10/3/1 views respectively), while the supporter is consistently scanned from 3 viewpoints. The results are

shown in Table III. (3) Sensitivity to supporter scanning: Similarly, in the third experiment, we examine the robustness of our method with respect to the scanning of the supporter. Here, we conduct experiment involving 2 hangable objects and 8 supporters. The objects are consistently scanned from 10 viewpoints, while the supporters are scanned from 3 or 1 viewpoint(s). The results are shown in Table IV.

C. Discussion

Table I illustrates that the hanging accuracy of our method significantly outperforms the baseline method [11]. Particularly, with the same input (**single view** for both object and supporter), the hanging success rate of our method (71.2%) is still higher than the success rate of baseline (64.4%). In addition, the baseline model is trained on 16195 pairs of objects and supporters. On the contrary, the proposed geometric-based method doesn't require any data for training and has better stability while facing previously unseen objects and supporters. Results of real-world experiments further display the reliability and the high accuracy of our approach. As shown in Table II, in real-world experiment our method identifies hanging poses for 77 pairs out of 80 pairs of hangable objects and supporters. The robot successfully executes 61 of them, resulting in a final success rate of 76.3%. Only one hanging pose is detected for the 32 unhangable pairs, which displays the stability of our method for false positive cases. Reasons of failure include unstable hanging prediction, predicted poses with collision, no hanging detected and problem in execution.

Table I, III and IV demonstrate that the hanging accuracy reduces as scan quality decreases but still remains high. Notably, even with meshes and point clouds reconstructed from single-view scanning, our method still achieves a high success rate and outperforms the baseline method. One reason for the reduction in accuracy is that when scanning quality is low, some structural elements that can afford hanging, like the cup handle, are incomplete and corresponding hangings cannot be detected. Moreover, if the reconstructed mesh and point cloud are incomplete, the algorithm may treat a hanging pose with collision as a collision-free one. However, as long as the essential structure required for hanging remains relatively intact in meshes and point clouds, our algorithm can still detect the corresponding hanging and furnish accurate predictions. Consequently, the hanging accuracy under poor scanning conditions still remains high.

IV. Conclusion and Future Work

This paper proposes a learning-free method for hanging detection of previously unseen objects and supporters. The proposed method first finds potential hanging positions and directions on both object and supporter. Then these hanging positions and directions are matched and evaluated to obtain feasible hanging poses. We also run experiments in simulation and real-world environments to validate the proposed method. Results show that our approach achieves a high success rate, outperforming the baseline results. Moreover, the accuracy remains high under poor scanning conditions. Future work will focus on exploring more complex hanging scenarios, such as hanging with multiple contact points and the optimization of the entire pick-and-hang procedure.

References

[1] R. Bogue, "Domestic robots: Has their time finally come?" Industrial Robot: An International Journal, vol. 44, no. 2, pp. 129–136, 2017.

[2] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," The International Journal of Robotics Research, vol. 31, no. 9, pp. 1021–1043, 2012. [Online]. Available: https://doi.org/10.1177/0278364912438781

[3] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "Kpam: Keypoint affordances for category-level robotic manipulation," in Robotics Research, T. Asfour, E. Yoshida, J. Park, H. Christensen, and O. Khatib, Eds. Cham: Springer International Publishing, 2022, pp. 132–157.

[4] E. Ruiz and W. Mayol-Cuevas, "Geometric affordance perception: Leveraging deep 3d saliency with the interaction tensor," Frontiers in Neurorobotics, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2020.00045

[5] E. Ruiz and W. Mayol-Cuevas, "Where can i do this? geometric affordances from a single example with the interaction tensor," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 2192–2199.

[6] A. Simeonov, Y. Du, Y.-C. Lin, A. R. Garcia, L. P. Kaelbling, T. Lozano-Pérez, and P. Agrawal, "Se (3)-equivariant relational rearrangement with neural descriptor fields," in Conference on Robot Learning. PMLR, 2023, pp. 835–846.

[7] P. Xu, H. Cheng, J. Wang, and M. Q.-H. Meng, "Learning to reorient objects with stable placements afforded by extrinsic supports," arXiv preprint arXiv:2205.06970, 2022.

[8] J.-S. Ha, D. Driess, and M. Toussaint, "Deep visual constraints: Neural implicit models for manipulation planning from visual input," IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 10 857–10 864, 2022.

[9] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, "Learning models as functionals of signed-distance fields for manipulation planning," in Conference on Robot Learning. PMLR, 2022, pp. 245–255.

[10] J. A. Haustein, K. Hang, J. Stork, and D. Kragic, "Object placement planning and optimization for robot manipulators," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 7417–7424.

[11] Y. You, L. Shao, T. Migimatsu, and J. Bohg, "Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation," in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.

[12] K. Takeuchi, I. Yanokura, Y. Kakiuchi, K. Okada, and M. Inaba, "Automatic hanging point learning from random shape generation and physical function validation," in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 4237–4243.

[13] K. Takeuchi, I. Yanokura, Y. Kakiuchi, K. Okada, and M. Inaba, "Automatic learning system for object function points from random shape generation and physical validation," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 2428–2435.

[14] I. Biederman, "Recognition-by-components: a theory of human image understanding." Psychological review, vol. 94, no. 2, p. 115, 1987.

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

[16] H. Goldstein, C. Poole, and J. Safko, "Classical mechanics," 2002.

[17] C. Yuksel, "Sample elimination for generating poisson disk sample sets," in Computer Graphics Forum, vol. 34, no. 2. Wiley Online Library, 2015, pp. 25–32.

[18] M. L. Zepeda-Mendoza and O. Resendis-Antonio, Hierarchical Agglomerative Clustering. New York, NY: Springer New York, 2013, pp. 886–887. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7-1371

[19] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning.(2016)," URL http://pybullet. org, 2016.