AVPLUG: Approach Vector PLanning for Unicontact Grasping amid Clutter

Yahav Avigal*[1], Vishal Satish*[1], Zachary Tam[1], Huang Huang[1],
Harry Zhang[1], Michael Danielczuk[1], Jeffrey Ichnowski[1], Ken Goldberg[1]

***Abstract*— Mechanical search, the finding and extracting of a known target object from a cluttered environment, is a key challenge in automating warehouse, home, retail, and industrial tasks. In this paper, we consider contexts in which occluding objects are to remain untouched, thus minimizing disruptions and avoiding toppling. We assume a 6-DOF robot with an RGBD camera and unicontact suction gripper mounted on its wrist. With this setup, the robot can move both camera and gripper in order to identify a suitable approach vector, reach in to achieve a suction grasp of the target object, and extract it. We present AVPLUG: Approach Vector PLanning for Unicontact Grasping, an algorithm that uses an octree occupancy model and Minkowski sum computation to find a collision-free grasp approach vector. Experiments in simulation and with a physical Fetch robot suggest that AVPLUG finds an approach vector up to $20\times$ faster than a baseline search policy.**

I. Introduction

In many automation tasks, such as extracting a product from a warehouse shelf, removing an ingredient from a refrigerator, or retrieving a tool from a cluttered workbench, desired objects may be hidden behind other objects. This presents a challenge in both locating the target object and finding a grasp for it. To automate such tasks, robots need to first perform visual search, and then robustly grasp and manipulate target objects once found. Although prior work [1], [2], [3] proposed methods for grasping objects in isolation, finding a robust grasp becomes significantly more challenging [4], [5] in a cluttered environment where the target object may be partially or fully occluded.

Mechanical search [6] aims to find a target object in clutter and focuses on clearing a view to the target by pushing or removing occluding objects [6], [7], [8], [9], [10]. However, this requires planning and executing multiple collision-free motions of the arm, adding to the overall runtime in the form of both motion planning and execution. Furthermore, the placement of occluding objects is often structured, for example with objects resting on a kitchen counter or supermarket shelf [11]. In such environments, pushing or removing objects may be undesirable. In addition, when objects are in unstable poses, even glancing contacts can lead to accidental toppling, which can damage delicate objects such as glass bottles. In contrast, this work focuses on servoing a wrist-mounted camera with a unicontact suction gripper tool (see Fig. 1) to a view from which the target object is fully visible and thus extractable.

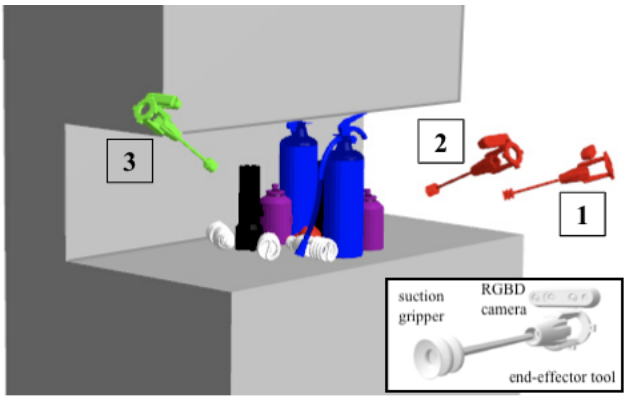


Fig. 1. AVPLUG searches for an approach vector to grasp the occluded red target object on the worksurface. AVPLUG moves the wrist-mounted tool to a view from which it can find an approach vector for unicontact grasping (successful view shown in green). AVPLUG uses an occupancy map and a Minkowski sum to track previously explored regions of the scene and to find and evaluate candidate vectors. **Inset:** The end effector used by AVPLUG is comprised of an RGBD camera with its optical axis aligned to a unicontact suction gripper.

Efficiently searching for a view of a target object is related to next-view planning, the problem of finding an additional sensor placement to improve scene reconstruction [12]. This topic has a rich history in computer vision [13], [14], [15]. Next-view planning requires keeping track of which regions of the scene have already been explored and which have not. For the task of unicontact grasp planning, a full scene reconstruction is computationally expensive and unnecessary; thus, we propose the use of an efficient 3D voxel-based occupancy map (e.g., OctoMap [16]) as it provides the required information and can be computed rapidly.

We focus on unicontact suction grasping. As opposed to parallel-jaw grasping, in which the contact points are rarely visible from the approach vector, unicontact grasp quality is highly correlated with the visible surface normals [17]. Accordingly, we propose that aligning the suction gripper contact axis to the camera optical axis can be well-suited for a unicontact grasp exploration policy, in which finding an unoccluded view of the graspable target surface corresponds to finding an approach vector.

We present Approach Vector Planning for Unicontact Grasping (AVPLUG), an algorithm that leverages an occupancy map based on an octree and Minkowski sum com-

*Equal contribution. [1]The AUTOLAB at the University of California, Berkeley (automation.berkeley.edu) {yahav_avigal, vsatish, ztam, huangr, harryhzhang, mdanielczuk, jeffi, goldberg}@berkeley.edu

putation to find an approach vector for unicontact grasping of a partially or fully occluded target object in a structured clutter scene, without changing the scene. First, AVPLUG samples potential target object locations from the unknown regions of the occupancy map. It then efficiently casts rays outwards from these candidate locations in order to identify unobstructed candidate vectors. Next, it cross-references these candidate views with a pre-computed expected grasp quality distribution (which takes into account individual grasp quality as well as uncertainty in target object pose). It moves to the view with the highest expected grasp quality. AVPLUG repeats this process until it finds a collision-free approach vector, or reports failure.

In the case of a fully occluded target object, we encounter an additional challenge, in that there is no clear signal to guide exploration. In order to narrow the search space, we propose to efficiently compute the Minkowski sum [18] between the target object and the region of the occupancy map that we have explored thus far. This constrains the potential locations of the target object on the worksurface.

Experiments in simulation and on a physical Fetch robot suggest that AVPLUG can find an approach vector in up to $20\times$ fewer steps than a baseline policy, even in the presence of dense occlusions and in tight spaces (see Fig. 6). This paper makes three contributions:

1) A formulation of the problem of efficiently finding an approach vector for unicontact grasping a target object in the presence of partial or full occlusions.
2) AVPLUG, an efficient algorithm that uses an octree-based occupancy map and Minkowski sum computation to address the above problem.
3) Experiments in simulation and on a physical robot comparing AVPLUG with a grid search baseline, which systematically visits views on a discretized grid.

II. Related Work

A. Target-Driven Grasping in Clutter

There has been significant prior work on searching for target objects in clutter, however the most common approach is to move or remove occluding objects. For example, Danielczuk et al. [6] defined the mechanical search problem and proposed a pipeline to iteratively search for a partially occluded object through a series of parallel-jaw grasping, suction, and pushing actions. Huang et al. [8] and Danielczuk et al. [7] then extended this work by learning an occupancy distribution to guide the search process to recover the occluded target. Xiao et al. [19] formulate the object search in clutter task as a POMDP and suggest an algorithm that takes into account the robot's current belief to evaluate the success of a manipulation task. Murali et al. [11] leveraged a variational autoencoder [3] to plan 6-DOF parallel-jaw grasps on a partially occluded target object in a cluttered scene, and remove occluding objects if no feasible grasp is found. Boroushaki et al. [20] identify and locate a fully occluded target object using RFID tags. In this work, we instead focus on moving a wrist-mounted camera to find clear approach vectors. We align the optical axis with these approach vectors in order to grasp the target object without affecting the rest of the scene.

B. View Planning for Grasping

In active perception [14], [15], [21], [22], we change the position of the sensor to reveal more of the scene's geometry. This is particularly useful for tasks such as 3D scene reconstruction [23] and mapping [24]. The next-best-view planning problem refers to computing the optimal next view with respect to a chosen goal. In the context of manipulation, a camera mounted on a robot end effector can guide the motion. Kahn et al. [25] model the occluded regions where the target object may be located as a mixture of Gaussians, and encourage exploration during the trajectory optimization by penalizing for uncertainty. Other works constrain the action space to top-down (4-DOF) grasps. For example, Morrison et al. [26] propose a top-down grasp planning controller that uses active perception to choose the next-best-view of the camera as it approaches the target object along the z-axis to reveal more robust grasps. Novokovic et al. [27] propose a reinforcement learning based active and interactive perception system from a top-down view to uncover a hidden target cube in a pile of cubes. In contrast, in this work we consider approach directions on a sphere centered on the clutter centroid and consider candidate 5-DOF grasps (unicontact suction grasps have symmetry about the approach vector).

C. Occupancy Maps

Occupancy maps are 3D representations of the environment that store information about which regions have already been explored and which have not. This information can be used to guide next-best-view planning. Hornung et al. [16] presented OctoMap, an efficient implementation of an octree-based occupancy map. Given a point cloud, OctoMap updates a 3D voxelized representation of the scene with one of three labels per voxel: occupied, empty, or unknown. Santos et al. [28] used an octree alongside a robotic arm and wrist-mounted camera, however they focused on 3D scene reconstruction. Octrees have also been used for grasping a target object in a cluttered scene [9], [10], [29], however in contrast to moving the camera, these works remove occluding objects from the scene to expose the target object.

III. Problem Statement

Given:

- An RGBD camera with known intrinsics, mounted in alignment with a vacuum suction cup gripper on a robot arm (see Fig. 1 inset).
- A target object of known geometry.
- An environment of unknown objects resting on a planar worksurface, partially or fully occluding the target object.
- A target object detector that returns a binary mask of the target object if it is visible from the RGBD camera.

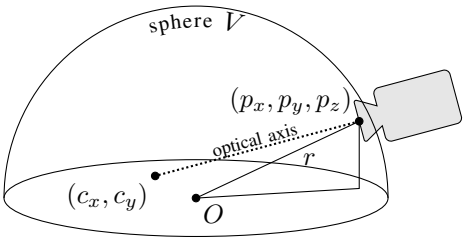


Fig. 2. States in AVPLUG consist of a camera location (p_x, p_y, p_z) on a sphere V centered on O with radius r, and a focal point (c_x, c_y) (with implicit $c_z = 0$) on the worksurface. (c_x, c_y) represents a potential target location, at which the camera's optical axis points.

- A suction grasp planner (Dex-Net 3.0 [17]) that samples candidate suction points on a depth image and returns the point with the highest associated grasp quality value.

Output: an approach vector $\mathbf{v}$ along which a collision-free linear motion can achieve a unicontact grasp of the target object. AVPLUG aims to minimize the number of steps to find such an approach vector, or reports failure.

A. Definitions

We define the following states, actions and observations:

Worksurface. A worksurface is a planar surface orthogonal to the z-axis which is aligned to gravity. The space reachable by the robot may be bounded from below by the worksurface and from above by a ceiling plane.

Sphere. Let V be a sphere with radius r centered on the origin of the worksurface (see Fig. 2).

States ($\mathcal{S}$). Let $\mathbf{s}_t \in \mathcal{S}$ denote a state at timestep t defining the position and orientation of the camera on V. We restrict the camera focal point to within the bounds of the worksurface. The camera can rotate about its placement on V to look at any point on the planar surface, but does not roll around its optical axis. The state space is thus $\mathcal{S} = S^2 \times S^2$, which we represent with a pair of Cartesian coordinates $(\mathbf{p}, \mathbf{c})$, where $\mathbf{p} \in \mathbb{R}^3$ is the location on V, and $\mathbf{c} \in \mathbb{R}^3$ is the point on the worksurface that the camera's optical axis intersects, thus $c_z = 0$ (see Fig. 2). Let $\mathbf{v} = \mathbf{c} - \mathbf{p}$ be the approach vector, defined as the direction from the camera to the target.

Actions (A). Let $\mathbf{a}_t = (\Delta p_x, \Delta p_y, \Delta p_z, \Delta c_x, \Delta c_y)$ denote the change from the state $\mathbf{s}_t$ to state $\mathbf{s}_{t+1}$, where $\mathbf{s}_{t+1}$ is restricted to remain on V.

Observations (Ω). Let $\mathbf{y}_t = \mathbb{R}_+^{H \times W \times 4}$ be an $H \times W$ RGBD image taken from state $\mathbf{s}_t$ at timestep t.

IV. Approach Vector Planning

Given an observation $\mathbf{y}_t$, AVPLUG seeks a grasp approach vector aligned within a tolerance angle ψ of the camera optical axis. After detecting and segmenting the target object, AVPLUG samples and evaluates grasps from its visible surface using a provided grasp planner, $\mathcal{G} : \mathbb{R}_+^{H \times W} \rightarrow (\mathbb{R}^3 \times S^2, \mathbb{R})$. $\mathcal{G}$ maps grasps parameterized by a 5-DOF pose $g \in \mathbb{R}^3 \times S^2$ to the corresponding grasp quality $q \in [0, 1]$. A higher value of q indicates a more robust grasp. If a termination condition $\mathcal{T}$ is not reached—i.e., there are no visible grasps in $\mathbf{y}_t$ above a certain grasp quality threshold—AVPLUG finds the next approach vector.

Scenes where the target object is fully occluded (e.g., due to inter-object and environmental occlusions) can be particularly challenging, since AVPLUG does not know the target object location. Without full knowledge of object poses and geometries, it is difficult to estimate the location and orientation of the target object, and more difficult still to estimate which views will uncover a graspable surface. We address this with an occupancy map, which we use to compile knowledge from previous views into an estimate of the scene state. This allows the policy to efficiently keep track of unexplored regions of the scene and prioritize them in subsequent steps. The occupancy map used in AVPLUG is based on an octree, $\mathcal{M} : \mathbb{R}^3 \rightarrow \{-1, 0, 1\}$, which maps voxels (minimum-size boxes in the octree) to occupancy values. In this paradigm, -1 represents unknown occupancy, 0 means known to be empty, and 1 means known to be occupied. The resolution of the octree is configurable—higher resolution allows for a more accurate search at the expense of increased processing time.

A. Updating the Octree

To update AVPLUG's representation of the occupancy map, the depth image in observation $\mathbf{y}_t$ is deprojected to a point cloud using the known camera intrinsics. It is then transformed to a global coordinate frame centered at the center of the worksurface using the known camera extrinsics. This transformed point cloud is inserted into the occupancy map $\mathcal{M}$ (Fig. 3(a)).

B. Finding Candidate Target Object Locations

If the target object is partially visible, AVPLUG approximates the translation of the target object as the center-of-mass of its visible portion. Otherwise, AVPLUG's first priority is finding the target object. With the assumption that all objects rest on a planar worksurface, AVPLUG reduces the computational complexity of the problem by limiting the search for candidate target object locations to the 2D worksurface. We define a 2D occupancy map as the 2D slice of the octree that corresponds to the worksurface, i.e., the portion at $z = 0$ in the global coordinate frame (see Fig. 3(b)). We note that AVPLUG does not project the occupancy map onto the worksurface, as this can result in missing a target object that is hidden beneath another object (e.g., a small object hidden below a large bowl). We also observe that there are only occupied and unknown voxels on the worksurface. AVPLUG searches for a set of candidate target object locations $\mathcal{U}$ in the unknown region of the worksurface (Fig. 3(b)).

Given the geometry of the target object, and assuming it has a finite set of feasible stable poses on the worksurface, we use a Minkowski sum [18] to estimate an occupancy distribution for the location of the target object. To compute the Minkowski sum, we first generate polygons from both the occupied region and the target object. For the former, we convert the 2D occupancy map to a binary image and find the

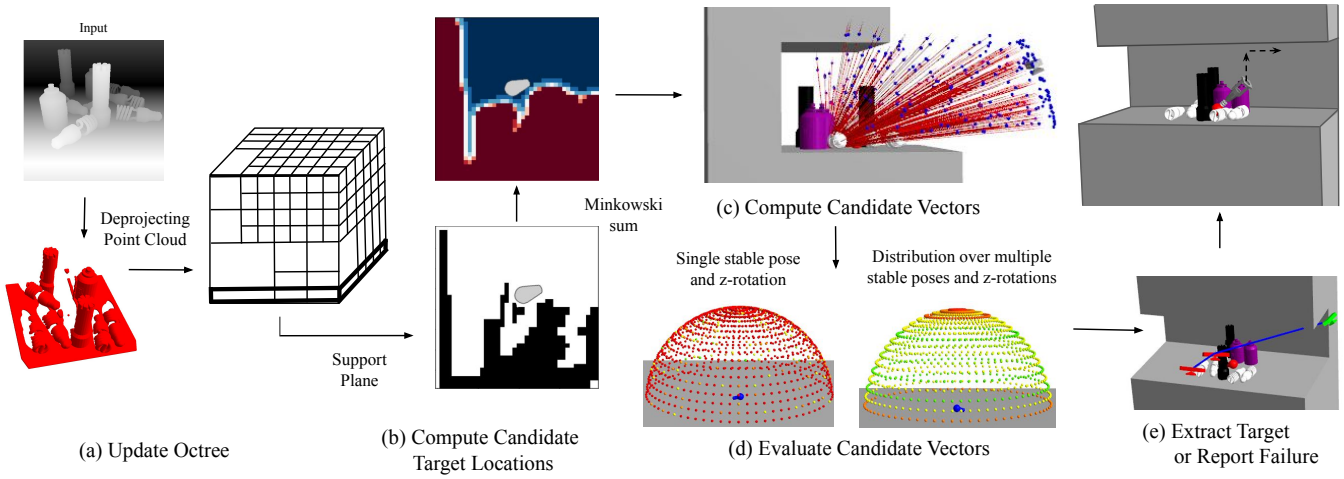


Fig. 3. **AVPLUG overview.** (a) AVPLUG updates the octree by deprojecting a depth image to a point cloud and inserting it to the octree. (b) AVPLUG queries the part of the octree corresponding to the worksurface in order to identify candidate target locations in the unknown regions (white is unknown, black is occupied). In the figure, the location of the target object is gray, although in practice its location is unknown. To reduce the number of candidate target locations, we compute the Minkowski sum between the convex hull of the known target object and the occupied section of the worksurface.. (c) AVPLUG casts rays outwards from candidate target locations to find candidate approach vectors. The corresponding camera positions (restricted to a sphere around the workspace) are marked by blue points. (d) AVPLUG uses a pre-computed expected grasp quality distribution to evaluate each candidate vector. The distribution is computed by averaging ground-truth grasp quality data from the target object model over stable poses and z-axis rotations. The average is weighted according to the relative likelihood of each stable pose. The data is then discretized into bins, with each bin represented by a colored dot in the figure (green represents the highest expected grasp quality, and red represents the lowest). (e) Once the target object is revealed, AVPLUG uses the provided grasp planner $\mathcal{G}$ to find a collision-free approach vector. It then aligns the camera's optical axis with that approach vector, moves linearly along the approach vector to grasp the target object, and extracts the grasped target with an upward motion.

contours of the occupied components (see Fig. 3(B)). If the contours form self-intersecting polygons, we smooth them using erosion and dilation [30]. To create a polygon of the target object, we project the vertices of the known mesh to the worksurface and compute the convex hull of the projected vertices. We then compute the Minkowski sum between the occupied region polygons and the target object polygon. This inflates the occupied region in a way that eliminates unknown points that are less likely to occupy the target object. Since the stable pose of the target object and its rotation about the z axis are unknown, we discretize the rotations into 8 bins and compute a Minkowski sum per stable pose and discretized rotation. We then sum the results and normalize to estimate a distribution for the location of the target object. We then use this distribution to uniformly sample a set of points $\mathcal{U}$ that maximize the likelihood of occupying a portion of the target object.

C. Finding Candidate Vectors

To find a candidate collision-free approach vector $\mathbf{v}$, AVPLUG casts rays outwards from the approximated target locations. This method draws inspiration from Lozano-Perez et al. [31], who describe an approach to fine motion synthesis by chaining backwards from a known goal toward the current position (Fig. 3(c)). If ray i in direction $\mathbf{d}$ does not intersect any occupied voxels, it represents a potentially clear line of sight, and the intersection point $\mathbf{p} \in \mathbb{R}^3$ of the ray with V is computed (see the blue points in Fig. 3(c)). There may be few or many candidate vectors, in correspondence with the levels of occlusion in the scene. A valid candidate vector consists of a point on the sphere and the negated ray direction leading to it: $\mathbf{v} = (\mathbf{p}, -\mathbf{d})$. To visit such a view, we move the camera to position $\mathbf{p}$ and align its optical axis with $-\mathbf{d}$.

D. Evaluating Candidate Vectors

Given the target object geometry and pose we could use $\mathcal{G}$ to compute grasps and check which approach vector aligns with a collision-free candidate vector. However, AVPLUG does not know the target object pose a priori; it only has access to the object geometry and a probability distribution of stable poses. Using this information, AVPLUG computes an expected grasp quality distribution for the target object before viewing any scenes. To compute this distribution, AVPLUG performs a weighted average of grasps and their associated quality, averaging over the known stable poses and all z-axis rotations. Higher likelihood stable poses are given more weight in the average. It then discretizes the data into 5° elevation $\times$ 5° azimuth bins (see Fig. 3(d)). AVPLUG evaluates each candidate vector based on the score of the bin containing its direction vector $-\mathbf{d}$. This method prioritizes views that align with a larger number of approach vectors (generally corresponding to different stable poses). Such views are more likely to find at least one approach vector that leads to a successful grasp in the given scene.

E. Finding and Evaluating Visible Grasps

Once the target object is revealed, AVPLUG queries the grasp planner $\mathcal{G}$ for visible grasps $g \in \mathbb{R}^3 \times S^2$. It then sorts the grasps by quality and iterates through the grasps in order to classify them. First, AVPLUG discards any previously seen grasps (from visible from prior views) known to

collide or be unreachable. Next, AVPLUG casts a ray in the occupancy map outward from the grasp contact point along the negated approach vector. If there are any collisions with voxels known to be occupied, it classifies this grasp as colliding. Otherwise, AVPLUG declares the grasp is collision-free, then moves to align the camera with the candidate approach vector and attempt the grasp. It caches all remaining visible grasps for later consideration if this grasp fails (e.g., by an undetected collision or obstacles preventing extraction). If none of the visible grasps is collision-free, AVPLUG finds the next view following the steps in Sections IV-C, IV-D.

V. Experiments

To evaluate AVPLUG, we run experiments in simulated and physical environments, and compare to a baseline policy.

A. Simulation Experiments

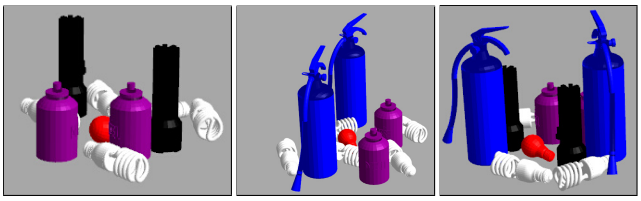
We use $R = 0.01\,\text{m}$ resolution since it empirically allows for a sufficiently accurate Minkowski sum in the fully occluded case and improves grasp collision estimation in the partially occluded case. To implement the octree for the occupancy map, we use the open-source OctoMap [16], [32].

We use ground truth segmentation to generate a binary mask of the target object. Since AVPLUG relies on an external instance segmentation algorithm, in practice, one could use an off-the-shelf object segmentation algorithm such as SD Mask R-CNN [33] with an additional matching phase for classification.

To decide whether the target object is graspable from the current state, we use a grasp planner based on Dex-Net 3.0 [17] as an oracle. Dex-Net 3.0 pre-computes suction grasps and associates quasi-static wrench-resistance quality metrics to a target object mesh, then matches these to pre-computation results at evaluation time.

B. Environments in Simulation

We first evaluate AVPLUG on two simulated scenes: 1) a tabletop, for which the potential vectors on the sphere V range between elevation angle $\theta \in [0°, 85°]$ and azimuth angle $\phi \in [-90°, 90°]$, and 2) a counter with a cabinet above that constrains the possible grasp approach directions to a slice of V ranging between elevation angle $\theta \in [55°, 85°]$ and azimuth angle $\phi \in [-90°, 90°]$. In both cases, we use a sphere V with radius $r = 0.6\,\text{m}$. To generate multiple different levels of occlusion, we use $N = 10$ objects of varying heights. Object models were selected from Thingiverse [34] and YCB [35]. We sample the locations of the objects from a uniform distribution in a bounded $0.4\,\text{m} \times 0.4\,\text{m}$ worksurface, and position each object in a stable pose. We also randomize the initial camera view. We generate 3 tiers of scene complexity (Fig. 4) by altering the relative proportions of larger objects; this affects the level of occlusions in the scene. **Tier 1** consists of 2 flashlights, 2 spray bottles and 5 spiral bulbs; **Tier 2** replaces the flashlights from tier 1 with $1.6\times$ higher and $1.7\times$ wider fire extinguishers; and **Tier 3** consists of 2 fire extinguishers, 2 flashlights, 2 spray bottles, and 3 spiral bulbs. For all tiers, the target object is a light bulb, which is easily occluded due to its small size compared to the other objects in the scene. After executing a successful grasp, the target object is extracted with an upwards motion (Fig 3(e)).



(a) Tier 1 (b) Tier 2 (c) Tier 3

Fig. 4. **Difficulty tiers with varying levels of occlusions.** The target is shown in red on a tabletop environment. The difficulty tiers define scenes with increasing levels of complexity due to increased occlusions. (a) Tier 1 includes 2 flashlights, 2 spray bottles, and 5 spiral bulbs. (b) Tier 2 includes 2 fire extinguishers, 2 spray bottles, and 5 spiral bulbs. (c) Tier 3 includes 2 fire extinguishers, 2 flashlights, 2 spray bottles, and 3 spiral bulbs.

			# Steps		Distance [m]	
Scene	**Tier**	**Policy**	Median	IQR	Median	IQR
Tabletop	1	GridSearch	16.0	22.2	0.8	1.0
		AVPLUG	**1.0**	**1.0**	**0.6**	**0.4**
	2	GridSearch	20.0	26.0	1.0	1.4
		AVPLUG	**1.0**	**1.0**	**0.7**	**0.6**
	3	GridSearch	16.0	16.0	0.8	0.9
		AVPLUG	**1.0**	**1.0**	**0.7**	**0.8**
Counter	1	GridSearch	14.5	19.0	0.7	0.9
		AVPLUG	**1.0**	**1.0**	0.7	**0.8**
	2	GridSearch	15.5	18.0	**0.7**	**0.9**
		AVPLUG	**2.0**	**2.0**	0.8	1.0
	3	GridSearch	18.0	19.5	0.9	**1.0**
		AVPLUG	**2.0**	**1.0**	0.9	1.2

TABLE I - **Simulation Experiments.** Median and interquartile range (IQR) of the number of steps to success and the distance traveled for each policy over 100 rollouts in 2 simulated environments, for successful rollouts. The success rate is 100 % for the *GridSearch* baseline and 94 % to 100 % for AVPLUG. (See Fig. 8 for description of failure modes)

C. GridSearch Baseline

We compare AVPLUG to a *GridSearch* baseline. *GridSearch* discretizes the sphere V into 212 fixed-spaced views (the distance between neighboring views is $l = 5°$ in both elevation and azimuth) and systematically visits each view until it finds a view from which it can plan a grasp. This baseline visits all the views to the right of the initial view, moves up to the next row of views once it reaches the maximum azimuth angle defined in V-B, continues the search by moving left until it reaches the next boundary, and so on. Once it reaches the top-most row and cannot move up, it continues the search from the bottom-most row. *GridSearch* stops when it finds a view from which it can plan a grasp, or after it has visited all the discretized views.

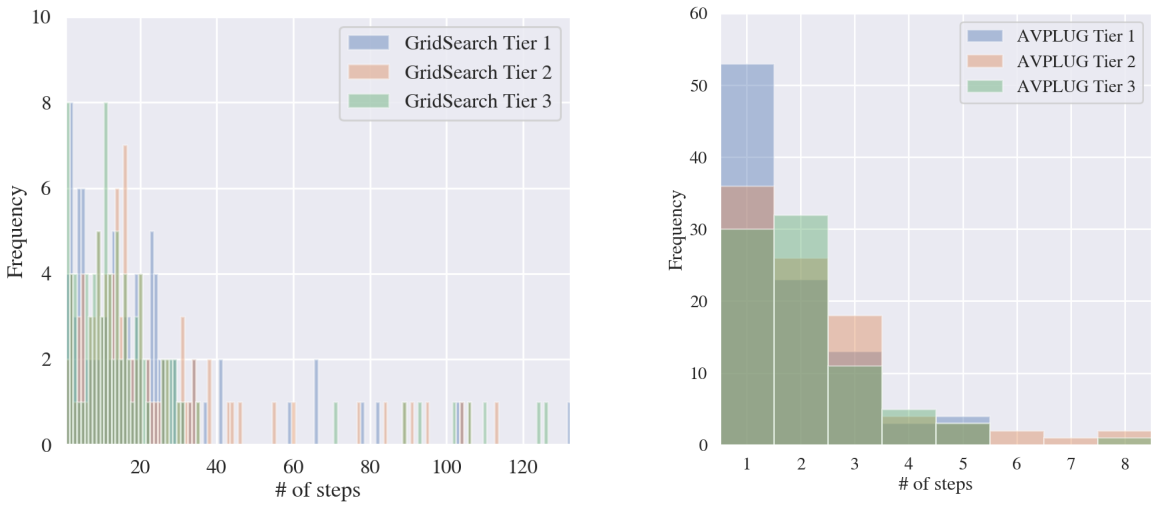


Fig. 5. Comparisons of steps to completion of successful rollouts in simulation. The scale on the horizontal axis is $15\times$ larger for the *GridSearch* policy.

D. Simulation Results

We roll out AVPLUG on 100 scenes, until the policy reaches a termination condition $\mathcal{T}$ and it finds a high-quality grasp ($q \geq 0.75$) on the target object, or it fails to find a grasp within a maximal number of steps H and the experiment fails. We set $H = 212$ to account for the total number of grid points in the countertop environment; therefore, if a successful approach vector exists, the *GridSearch* baseline will find it. We benchmark the experiments using the following metrics: median and interquartile range (IQR) for number of steps to success, and distance traveled. We use these metrics since the number of steps to success relates to the data acquisition and computation time, and the distance traveled by the robot arm may result in increased travel time and a potential loss in precision. The results are summarized in Table V-B and Fig. 5, and show that AVPLUG finds an approach vector in up to $20\times$ fewer steps (median) than the baseline. In Fig. 5 we observe that the baseline suffers from high variance, as it is sensitive to the initial view—if it starts near a successful approach vector it can terminate quickly, otherwise it may search the grid exhaustively.

The average computation time of AVPLUG for finding an approach vector is 1.05 s, benchmarked on a server with an Intel Xeon CPU @ 2.20 GHz.

E. Physical Experiments

We evaluate AVPLUG on physical scenes in the countertop setting using a Fetch mobile robot. To find grasps, we use a planarity-based grasp planner. The grasp planner first samples candidate suction points from a depth image by computing surface normals, then selects only those within 10° of the optical axis. Finally, it ranks these candidates by using a planarity metric: a suction cup-sized ring is projected around the grasp point, and the grasp is scored based on the distance from the ring to the surface depth. This distance is minimized in higher quality grasps [17]. We filter out any grasps that will collide with the scene when approaching or exiting using collision checking between the gripper mesh and the observed point cloud. We consider a grasp successful if it is not in collision and its quality value is above 0.8.

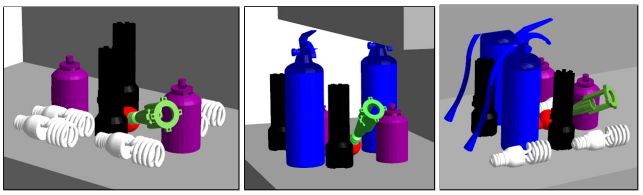


Fig. 6. **Unicontact grasping in tight spaces.** AVPLUG can find approach vectors for unicontact grasping even in tight spaces due to the high resolution of the occupancy map.

We construct 3 tiers of scenes with matching difficulty to those in simulation. For each tier, we evaluate a single scene, and for each scene we choose 5 random starting views. At each view, we evaluate the baseline once and AVPLUG 3 times, taking the average to account for inherent stochasticity. We use the elevation angle $\theta \in [45^\circ, 75^\circ]$, azimuth angle $\phi \in [-45^\circ, 45^\circ]$, and radius $r = 0.5$ m for kinematic feasibility. The target object is a red light bulb similar to the target in simulation, and the occluding objects are objects found around the house and lab. We use an HSV color detector to get the binary target segmentation mask. Figure 7 shows the experimental setup. Results in Table V-E suggest that AVPLUG can consistently find an approach vector in fewer steps (median 2.0) than the baseline (median between 5.0 and 12.0). While the number of search steps taken by baseline policy highly depends on the starting view (with a higher IQR between 3.0 and 10.0), AVPLUG is able to achieve more consistent high performance among random starting views (with a lower IQR of 1.0).

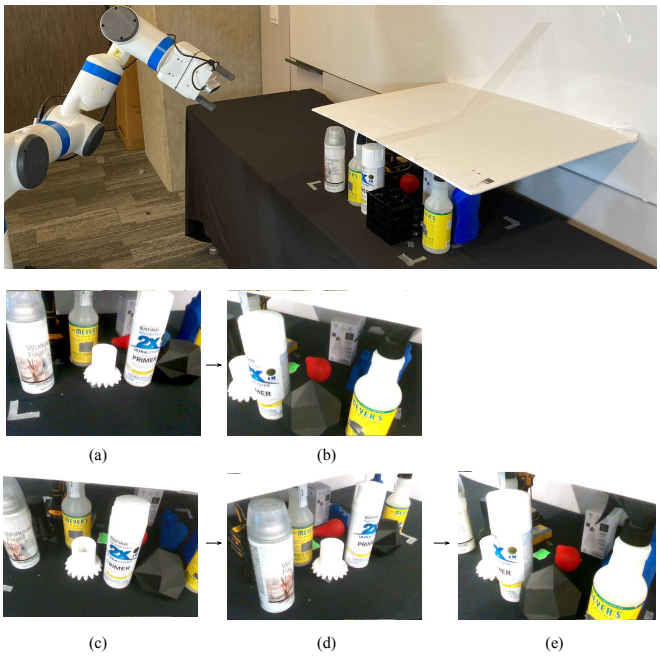


Fig. 7. **Physical experiments setup. Top:** Physical counter setup with a Fetch mobile manipulator for grasping. **Middle:** In the first experiment starting at (a), the visible part of the target object (in red) is not graspable from the initial position, but is graspable from the next position (b). **Bottom:** In the second experiment starting with view (c), although a successful grasp is found from the second position (d), it leads to a collision between the gripper and the environment. AVPLUG then finds a collision free approach vector on the following step (e).

		# Steps		Distance [m]	
Tier	**Policy**	Median	IQR	Median	IQR
1	GridSearch	5.0	10.0	**0.6**	0.5
	AVPLUG	**2.0**	**1.0**	0.7	**0.4**
2	GridSearch	6.5	9.2	0.6	0.5
	AVPLUG	**2.0**	**1.0**	0.6	**0.3**
3	GridSearch	12.0	3.0	1.1	0.6
	AVPLUG	**2.0**	**1.0**	**0.6**	**0.4**

TABLE II - **Physical Experiments Results.** Median and interquartile range (IQR) of the number of steps to success and the distance traveled for each policy over 5 rollouts in a physical counter environment. The metrics are reported for successful rollouts. The success rate is 100% for both the *GridSearch* baseline and AVPLUG.

F. Visibility vs Graspability

An visibility version of AVPLUG evaluated candidate approach vectors according to their *information gain*, defined by the number of voxel labels that changed from unknown to either empty or occupied after aligning the camera optical axis with the corresponding approach vector. In this method, AVPLUG chose an approach vector that maximized the information gain, with the goal of discovering presently hidden graspable surfaces on the target object. One limitation of this approach, however, was that it prioritized distant approach vectors over near and successful ones, since drastic view changes would generally reveal more unobserved parts of the scene. This visibility-based approach is better suited for the purpose of scene reconstruction and mapping than for finding a grasp on an occluded target object—this motivated using known grasp distributions instead.

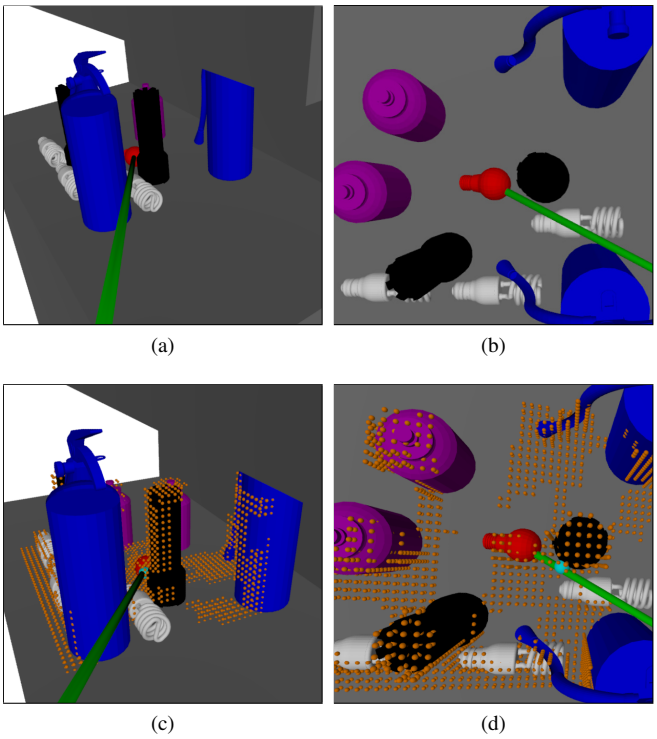


Fig. 8. **Octree resolution failure case.** AVPLUG queries the grasp planner for available grasps on the target object. The grasp approach axis (green) passes between the flashlight and the fire extinguisher with enough clearance for the long and thin end effector in (a) and (b), therefore the grasp planner declares this to be an accessible grasp. However, when AVPLUG casts this ray through the octree, it finds a collision with the flashlight at the cyan point in (c) and (d). This is because the octree's occupied voxels (denoted by orange points) extend slightly beyond the bounds of the flashlight due to the discretization.

G. Failure Cases

Since the occupancy map discretizes the scene into cubic voxels, the occupied section of the octree occasionally extends beyond the true boundaries of the occluding objects (see Fig. 8). Furthermore, due to the long and narrow structure of the end effector, valid grasp approach vectors can pass very close to occluding objects. As a result, AVPLUG's grasp evaluation step (Section IV-D) may detect collisions when there are none.

VI. Conclusion

We present AVPLUG, an algorithm that employs an octree-based occupancy map and Minkowski sum computation to find an approach vector for unicontact grasping. AVPLUG takes advantage of the strong correlation between visibility and graspability in suction grasping by servoing a wrist-mounted camera to find graspable views. It is able to find and extract fully or partially occluded known target

objects without the risk of toppling other objects. Experiments in simulation and on a physical robot suggest that AVPLUG can find an approach vector in up to $20\times$ fewer steps compared to a baseline policy, and can extract objects from tight spaces (see Fig. 6). In future work, we will utilize shape completion and pose estimation algorithms to reason about the graspable part of the target object. We will also extend this work to a tight shelf environment, from which the object cannot be easily extracted.

ACKNOWLEDGEMENTS

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, and the CITRIS "People and Robots" (CPAR) Initiative. The authors were supported in part by donations from Google, Siemens, Toyota Research Institute, Autodesk, Honda, Intel, Hewlett-Packard. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1752814. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. We thank our colleagues who provided helpful feedback and suggestions, in particular Daniel Seita, Anna Deza, and William Wong.

REFERENCES

[1] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "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. Robotics and Automation (ICRA)*, 2016, pp. 1957–1964.

[2] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robotics: Science and Systems (RSS)*, 2017.

[3] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2901–2910.

[4] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *Int. Journal of Robotics Research (IJRR)*, vol. 39, no. 2-3, pp. 183–201, 2020.

[5] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Conf. on Robot Learning (CoRL)*, 2017, pp. 515–524.

[6] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, "Mechanical search: Multi-step retrieval of a target object occluded by clutter," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 1614–1621.

[7] M. Danielczuk, A. Angelova, V. Vanhoucke, and K. Goldberg, "X-ray: Mechanical search for an occluded object by minimizing support of learned occupancy distributions," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[8] H. Huang, M. Dominguez-Kuhne, J. Ichnowski, V. Satish, M. Danielczuk, K. Sanders, A. Lee, A. Angelova, V. Vanhoucke, and K. Goldberg, "Mechanical search on shelves using lateral access x-ray," *arXiv preprint arXiv:2011.11696*, 2020.

[9] J. K. Li, D. Hsu, and W. S. Lee, "Act to see and see to act: Pomdp planning for objects search in clutter," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 5701–5707.

[10] A. Price, L. Jin, and D. Berenson, "Inferring occluded geometry improves performance when retrieving an object from dense clutter," in *Int. S. Robotics Research (ISRR)*, 2019.

[11] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 6232–6238.

[12] E. Dunn and J.-M. Frahm, "Next best view planning for active model improvement." in *BMVC*, 2009, pp. 1–11.

[13] S. D. Roy, S. Chaudhury, and S. Banerjee, "Active recognition through next view planning: a survey," *Pattern Recognition*, vol. 37, no. 3, pp. 429–446, 2004.

[14] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[15] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[16] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[17] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018, pp. 5620–5627.

[18] D. Halperin, "Robust geometric computing in motion," *Int. Journal of Robotics Research (IJRR)*, vol. 21, no. 3, pp. 219–232, 2002.

[19] Y. Xiao, S. Katt, A. ten Pas, S. Chen, and C. Amato, "Online planning for target object search in clutter under partial observability," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 8241–8247.

[20] T. Boroushaki, J. Leng, I. Clester, A. Rodriguez, and F. Adib, "Robotic grasping of fully-occluded objects using rf perception," *arXiv preprint arXiv:2012.15436*, 2020.

[21] Y. Aloimonos, *Active perception*. Psychology Press, 2013.

[22] J. Aloimonos, "Purposive and qualitative active vision," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 1990, pp. 346–360.

[23] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[24] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active slam," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 2080–2087.

[25] G. Kahn, P. Sujan, S. Patil, S. Bopardikar, J. Ryde, K. Goldberg, and P. Abbeel, "Active exploration using trajectory optimization for robotic grasping in the presence of occlusions," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 4783–4790.

[26] D. Morrison, P. Corke, and J. Leitner, "Multi-view picking: Next-best-view reaching for improved grasping in clutter," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 8762–8768.

[27] T. Novkovic, R. Pautrat, F. Furrer, M. Breyer, R. Siegwart, and J. Nieto, "Object finding in cluttered scenes using interactive perception," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 8338–8344.

[28] J. Santos, M. Oliveira, R. Arrais, and G. Veiga, "Autonomous scene exploration for robotics: A conditional random view-sampling and evaluation using a voxel-sorting mechanism for efficient ray casting," *Sensors*, vol. 20, no. 15, p. 4331, 2020.

[29] M. Gupta, T. Rühr, M. Beetz, and G. S. Sukhatme, "Interactive environment exploration in clutter," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 5265–5272.

[30] P. Soille, "Erosion and dilation," in *Morphological Image Analysis*. Springer, 2004, pp. 63–103.

[31] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *Int. Journal of Robotics Research (IJRR)*, vol. 3, no. 1, pp. 3–24, 1984.

[32] K. Wada, "Octomap-python," https://github.com/wkentaro/octomap-python, 2013.

[33] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 7283–7290.

[34] Thingiverse online 3d object database. [Online]. Available: https://www.thingiverse.com/

[35] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols," *arXiv preprint arXiv:1502.03143*, 2015.