Inferring Occluded Geometry Improves Performance when Retrieving an Object from Dense Clutter

Andrew Price, Linyi Jin, Dmitry Berenson

***Abstract*—The ability to find objects in cluttered scenes is essential for enabling many robotics applications in warehouse and household environments. However, cluttered environments entail that objects often occlude one another, making it difficult to segment objects and infer their shapes and properties. We do not assume CAD or other explicit models of objects are available. Instead, we augment a manipulation planner for cluttered environments with a state-of-the-art deep neural network for shape completion as well as a volumetric memory system. These components allow the robot to reason about what may be contained in occluded areas without explicit object models. We test the system in a variety of tabletop manipulation scenes composed of household items, highlighting its applicability to realistic domains. Our results suggest that shape completion allows the robot to reduce the amount of occluded space to explore by removing occluded regions that are a part of visible objects. Likewise, volumetric memory allows the robot to track previously-seen shapes and free space that are now occluded, again allowing a better estimate of the unknown space. Finally, we show that incorporating both components into a manipulation planning framework significantly reduces the number of actions needed to find a hidden object in dense clutter.**

I. Introduction

The ability to retrieve objects in cluttered environments is an important requirement for many robotics applications, from warehouse retrieval to household chores. Yet cluttered scenes inherently impose a limitation on visibility: objects occlude one another in close proximity, and often the range of feasible viewpoints is limited. Another frequent characteristic of such scenarios is object novelty: in unstructured environments such as homes, new objects appear frequently and it is very restrictive to require that all objects in the scene have corresponding CAD models available. In spite of these difficulties, an understanding of the physical extents of objects is important for object manipulation and action sequencing.

Thus this paper focuses on a key topic that has been largely overlooked in the clutter manipulation domain: inferring occluded geometry. The goal of this paper is *not* to claim a significant novelty in terms of fundamental algorithms for inferring occluded geometry, nor is to present the world's best manipulation-in-clutter system. Rather this paper seeks to answer the following question: *Is inferring occluded geometry useful for retrieving objects from clutter?* By not inferring occluded geometry, the current state-of-the-art [1]–[3], which shows impressive performance, may suggest that it is not. However, when object models are not available, we hypothesize that this kind of inference significantly improves performance in dense clutter.

To test our hypothesis we constructed a manipulation planning system for a bimanual robot (see Figure 1). We do not assume CAD or other explicit models of objects are available. Instead we work directly with the output of an RGBD sensor to infer a volumetric segmentation along with the occluded areas, which we call *shadows*. The baseline system plans actions directly from these segmented objects and shadows. If sufficient target object voxels (as identified by a given classifier) are visible and graspable, we attempt to retrieve the object directly. Otherwise, to search for the object, we sample a shadow voxel and attempt to move the object which occludes that voxel using one of a set of motion primitives. By using the segmentation and occlusion information without further processing, the robot cannot reason about occluded geometry as it only observes the surfaces of visible objects. As a result, it can greatly overestimate the shadow of a given object, moving it unnecessarily, which is very likely in dense clutter where many objects are present. The system also has no way to remember object geometry or free space that was seen before but is now occluded as a result of the robot's actions, which can lead to repeatedly moving an object with a large shadow without gaining new information.

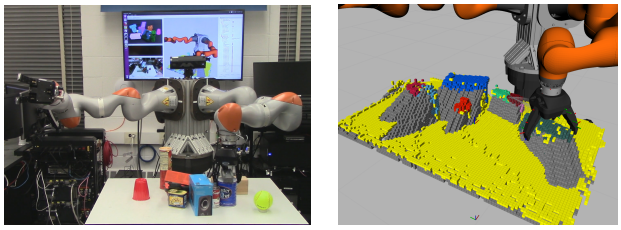


Fig. 1. Left: Our robot sliding an object in a cluttered scene to reveal the target (the yellow ball). Right: The robot's representation of the world, including shape-completed objects and memory of previously seen shapes and free space.

To evaluate whether the lack of inference of occluded geometry significantly hinders the system's performance, we developed two modules to augment the baseline system (see Figure 2):

1) *Shape completion*: A method, based on previous work [4], to infer the shape of a partially-occluded object.
2) *Shape memory*: A method to track free space seen earlier in the interaction, as well as previously observed occupied space that has been moved into an occluded region.

Again, these modules are not the main contribution of the paper, but are used to evaluate the hypothesis that inferring occluded geometry improves performance in dense clutter. While improvements to these modules and the system as a whole are no doubt possible, our results suggest that these modules are indeed sufficient to verify our hypothesis.

To evaluate the performance of the two modules and compare the baseline system to the augmented one, we conducted 182 manipulation experiments in eight tabletop cluttered scenes (see Figure 7). We first evaluated whether our modified shape completion method outperformed previous work on a dataset specialized to our application. We then tested the hypothesis that each module independently improved performance in specially-constructed scenarios. Finally, we tested the performance of the baseline vs. the augmented system in scenarios with varying amounts of clutter. In densely cluttered scenes we found that using shape completion and memory significantly improved the performance of the system.

II. Related Work

A. Object Retrieval from Clutter

Manipulation of movable obstacles in cluttered scenes has been a longstanding goal of robotics research [5], [6]. While these earlier examples assumed full knowledge of the scene in question, leveraging manipulation to discover hidden objects has also been a significant focus. In [7], Wong et al. use spatial visibility constraints and object semantic information to plan manipulation sequences. [8] use revealed volume in utility function and generate connected component networks from object occlusions, [9] frame the problem as a POMDP, and employ a similar image processing pipeline to the one presented here. The above methods make important contributions to the literature, however they make simplifying assumptions (e.g. discrete planning space, known object models, or sparse clutter) that are more restrictive than ours.

In recent years, much work about vision and manipulation in clutter has been driven by the scenarios presented by the Amazon Picking Challenge (APC), producing numerous publications on both full frameworks and isolated components. In APC 2015, [1] developed a segmentation algorithm based on explicit image features (color, edge, missing 3D, distance to shelf, height etc). [2] from APC 2016 combined object detection using a fine-tuned network on Visual Genome and semantic segmentation using a pretrained pixel-level CNN model on ImageNet. Starting in APC 2017, researchers were trying to manipulate novel items that appeared in the challenge. [3] exploited feature maps at different levels of detail to produce high-resolution semantic maps. [10] used a multi-affordance grasping framework which operates without a prior object segmentation and classification to enable grasping without object identity. However, these methods do not reason about occluded space.

Until recently, most systems seeking to infer volumetric information about the hidden parts of the scene have relied on CAD model matching [11], [12] or semantic matching [13]. However, progress in reconstructing objects from single 2.5D views [14], [4], [15] has enabled manipulation planning on unseen parts of the space [16]. We extend this work in order to reason about occlusion in a cluttered scene.

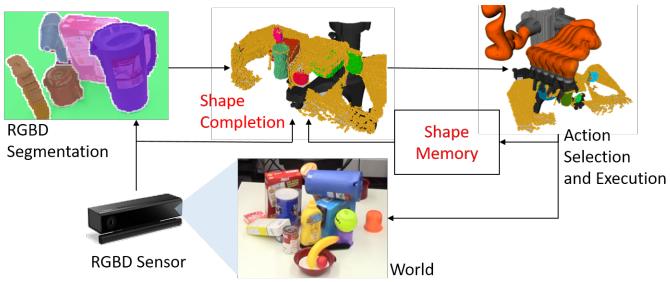


Fig. 2. Diagram of our framework. Components in red text reason about occlusion.

B. Volumetric Shape Reconstruction

Reconstructing a 3D model of a scene is both a major challenge and powerful tool for robotic manipulation. Recently, deep neural network approaches have achieved impressive success in shape completion and 3D scene reconstruction. [13] estimated 6D pose of object by a convolutional neural network from RGB images. [17] proposed a convolutional neural network-based approach to predict object pose and shape of indoor scenes from RGB-D data. However, those methods of reconstruction require object recognition, which may not apply to novel objects. Several methods exist for shape completion without object recognition [4], [14], [16], [18]. We chose to build on [4] because it obtained good performance on challenging objects and it was clear how to incorporate free-space information into the network.

III. Problem Statement

We seek to retrieve a specific object from a cluttered scene using robotic manipulator(s). The domain is designed to represent household applications, where previously-unseen objects can be present in the scene and CAD models are not always available.

Sensing: To sense the environment we assume that the robot is endowed with a single RGB-D sensor. We assume that the objects in the environment are arranged on a flat surface (such as a table) and objects may be stacked on each other. A key difference between our domain and that of much previous work is that we *do not* assume that we possess CAD models or other explicit object representations for the objects on the table. The robot may have observed some of the objects during training of its perception algorithms, but other objects are completely new. Furthermore, the robot has no way to identify objects with which it has been trained. The robot receives an observation of the state of the environment o_t before it acts. $o_t \in O$, where O is an $w \times l$ grid of RGBD values.

The robot also has no explicit representation for the target object, but is endowed with a classifier $L : o_t \rightarrow \{0, 1\}$ which determines if a given pixel in the RGBD image is likely to be a part of the target object. This is meant to handle queries that may come from a user, such as "Bring me the yellow ball", where no explicit model of the object is given.

Acting: The robot may manipulate the objects in any way it chooses, however, unlike much previous work, we assume we are *not* able to command the robot to remove objects from the

Fig. 3. Segmentation results from SceneCut. 3(a) Live view from Kinect. 3(b) Segmentation result.

scene. We make this restriction to consider scenarios where the robot has a limited work-surface (i.e. a confined space).

Further, we assume that the robot has only a limited knowledge of contact mechanics and physics. Contacts between the robot model and environment or between a grasped object and a tabletop object can be computed, but their behavior after contact is difficult to predict because physical properties such as mass, pressure distribution, and friction, are not known.

Problem: The robot is endowed with a set of possible actions it can apply $\mathcal{A}$ and must choose which actions $a_{1\ldots n} \in \mathcal{A}$ to take to find and retrieve the target object. This problem can be formulated as a Partially-Observable Markov Decision Process (POMDP) by defining a belief state over the environment and computing a policy of the form $\pi(a_{1:t}, o_{1:t}) = a_{t+1}$ which maximizes the probability of success given any starting state. While a POMDP policy would be desirable, this is clearly intractable as the belief over environments is too high-dimensional for a POMDP solver to handle and we do not have models of the transition and observation uncertainty. Instead, we focus on a greedy approach: we seek to design a π that takes the next best action given $a_{1:t}$ and $o_{1:t}$. A key challenge is how to use $a_{1:t}$ and $o_{1:t}$ to infer object geometry in occluded regions, so that this information can be used to inform action selection.

IV. System Framework

This section describes the components of our system, shown in Figure 2. We first describe our methods for perception, including shape completion and memory, and then describe our approach to action selection.

A. RGB-D Segmentation

The pipeline begins by processing o_t to produce a segmentation of the observed scene into distinct objects. Many state-of-the-art semantic segmentation approaches require object class annotations [19], [20], which violates our stipulation that the scene may contain novel objects or classes. Therefore, we investigated class-agnostic segmentation approaches, adapting SceneCut [21] for our scenario. The method begins with an ultrametric contour map (UCM) [22], a hierarchical segmentation tree derived from the RGB-D image of the scene. The UCM is generated by a Convolutional Oriented Boundary (COB) [23], [24] network trained on NYUD-v2 dataset [25]. SceneCut then utilizes a tree cut to minimize an energy function over objectness and geometric fitting. Figure 3 is a segmentation results using SceneCut.

Each segment of the RGB-D image, given the camera intrinsics, corresponds to a point cloud representing a potential object. Points belonging to the table surface, robot arms, or outside the table region of interest are rejected, and the surviving point clouds are passed to shape completion. We also try to find the target in o_t, as well as extracting occupied and free volumes (detailed below).

1) Target Object Detection: Given a collection of image/point cloud segments, we next determine whether any of them is the target in question. We assume a classifier of the form $L : o_t \to \{0, 1\}$ is available, as object recognition is not a focus of this work. Our implementation uses color matching to classify whether a pixel belongs to the target object. A mask of the RGB image is generated by filtering in HSV color space, then compared with the segmentation results. Given a match threshold τ_{target}, if $> \tau_{\text{target}}$ of a segment is masked, and $> \tau_{\text{target}}$ of the mask is within that segment, that segment is considered to be the target.

2) Occupied and Free-Space Volumes: Using the segmentation and full point cloud, we can compute a voxelized representation of the free, occupied, and unknown state of the world. Occupied and free space computation is provided by feeding the point cloud data to OctoMap [26] to generate an octree representation of the scene. The shape completion results from the following section are fed back into object-specific OctoMaps that are used for end-effector collision computations.

B. Shape Completion

Consider an occupancy map $V\colon \mathbb{R}^3 \to \{\,0, 1\,\}$ carrying 3D points to a binary occupancy value, empty or filled. Letting $\mathbb{N}_n \doteq \{\,0, 1, \ldots, n-1\,\}$ represent the first n natural numbers, we can define a *voxel grid* as a discrete version of V:

$$V_n\colon \mathbb{N}_n^3 \to \{\,0, 1\,\} \tag{1}$$

Let $\because V_n$ represent the pointwise representation of V: the list of points $\mathbf{p}$ that are at the center of each voxel and $V(\mathbf{p}) = 1$. We now define a distance between voxel grids based on the *Chamfer distance* as defined by [18]:

$$\begin{aligned} D_{\mathrm{C}}(X, Y) \doteq & \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 \\ & + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2 \end{aligned} \tag{2}$$

So, for two voxel grids U_n and V_n, we define $D(U_n, V_n) \doteq D_{\mathrm{C}}(\because U_n, \because V_n)$. Let the true voxel occupancy of an object be $V_n^{obj} \subset V_n$ and the observed free space be $V_n^{free} \subset V_n$. Then, given a partial scan of the object represented as $V_n^{partial} \subset V_n^{obj}$ and V_n^{free}, shape completion seeks to solve the following problem:

$$\begin{aligned} \underset{V_n^{completed}}{\text{argmin}} \quad & D(V_n^{completed}, V_n^{obj}) \\ \text{subject to} \quad & V_n^{partial} \subset V_n^{completed}, \\ & V_n^{partial} \cap V_n^{free} = \emptyset. \end{aligned}$$

However, it is not possible to solve this problem at runtime because V_n^{obj} (the true shape of the object) is unknown. Instead, we apply learning methods which train a deep neural network on multiple views of objects in simulation, where the true

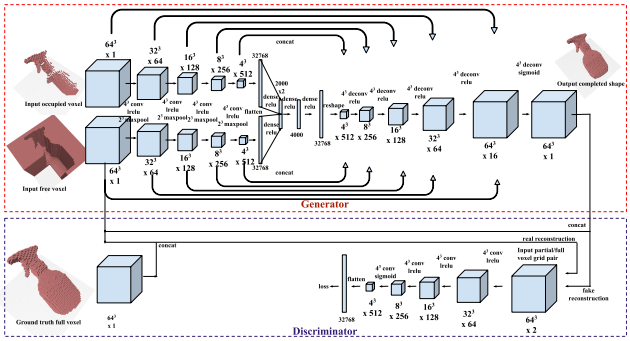
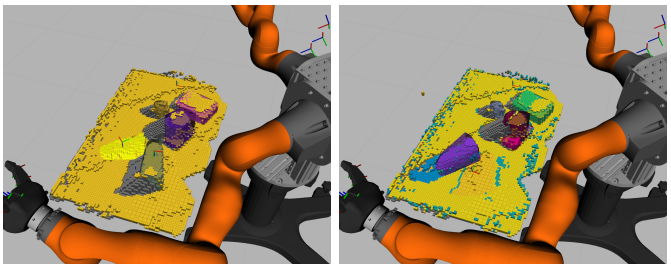


Fig. 4. Our network architecture for shape completion.



(a) (b)

Fig. 5. Types of volumetric memory. In 5(a), the bright yellow voxels are believed to be the future location of the bottom object in the scene after it is moved with a given action. In 5(b), no shadow points are generated behind the purple object, as that space was seen to be free earlier.

shape is used as ground-truth. We then use the learned network to predict a likely $V_n^{completed}$ for an object we encounter at runtime.

To tackle the learning problem, we begin with a base model of the 3D-RecGAN architecture [4].

3D-RecGAN is a combination of a generative autoencoder a and Generative Adversarial Network (GANs)[27]. By training with a GAN, the autoencoder can generate high-resolution 3D shapes that capture key features (such as handles). Compared to previous approaches which generate a 3D shape from RGB or RGB-D information, this architecture does not require object class labels and is able to generalize to unseen objects. This approach performs well, but does not have the ability to include V_n^{free}, so it may generate voxels in known free space. We thus build on this method by incorporating two main modifications: 1) restructuring the network architecture to include known-free space and 2) using a dataset that includes occlusions for training.

1) Architecture: 3D-RecGAN consists of two main networks: the generator and the discriminator. We improve on the original network by augmenting the input space with V_n^{free}. Figure 4 shows the detailed architecture of our modified generator in 3D-RecGAN. Both the occupancy voxels and free voxels are encoded using the five 3D convolutional layers used by 3D-RecGAN. In latent space, the two latent vectors are concatenated together. The decoder comprises six up-convolutional layers followed by ReLU activations except for the last layer which uses a sigmoid function. All encoder layers are concatenated to the decoder by skip-connections to preserve local structures. The discriminator and loss functions are the same as those used in 3D-RecGAN.

2) Synthesizing a dataset with occlusions: Many large synthetic datasets generated from 3D models exist for the purpose of 3D reconstruction from a single view. Most existing cluttered datasets are generated on large and complex objects such as furniture in an indoor scene. Existing datasets for robotic manipulation in cluttered scene are limited. [16] generated a dataset for shape completion for the purpose of robot grasping using objects from the YCB dataset [28] and Grasping dataset [29]. However, this dataset only includes a complete single-view occupancy grid with self occlusion only. [14] introduced a tabletop dataset which consists of a complete RGB-D occluded scene with objects occluding each other, but their TSDF encoding is different from our requirement in the network architecture.

In order to train a network to reconstruct occluded parts from cluttered scenes, we modify and augment the dataset synthesis steps used by [16] so that the objects are not only self-occluded but also occluded by an obstacle. Our dataset contains three kinds of 3D voxel grids for each example: $V_n^{partial}$, V_n^{free}, and V_n^{obj} (the ground truth).

12 objects from the YCB dataset and ShapeNet [30] are collected and occupancy grids are generated from the object meshes using binvox [31]. After that, rotations are uniformly sampled in roll-pitch-yaw space. Instead of directly generating depth images from different angles of rotations, an obstacle mask is placed in between the camera and the mesh, occluding part of V_n^{obj} and thus generating $V_n^{partial}$.The voxels between the camera and the $V_n^{partial}$ are V_n^{free}. $V_n^{partial}$ is then centered in the reconstruction grid in order to remove information about the original object extents so that the input is similar to a real scenario, where the true extent of the object is unclear. The recentered voxels are then shifted towards the camera to a fixed offset to provide more space for reconstruction. In the experiment section, we show that training using this occluded data set boosts the performance when reconstructing objects in the presence of occlusion.

C. Volumetric Memory

Although the dynamics of manipulation are difficult to predict, there are cases, such as when the robot has a stable grasp on a manipulated object, where we wish to inform the next scene of past interactions. With *positive memory*, we compute the pose transformation due to the manipulator motion, then add the object octree at time $t-1$ into the scene octree at time t using the new pose (Figure 5(a)). With *negative memory*, we assume that space that was previously free and is now occluded is likely to remain free, unless the shape completion indicates otherwise (Figure 5(b)). As both of these assumptions can be violated by unanticipated object interact (e.g. objects slipping in the grasp or collisions knocking objects behind others), unobserved space is set to decay to the occupancy threshold

$\tau_{\text{occupancy}}$ with rate $0 < \alpha < 1$:

$$V_t(\mathbf{x}) = \alpha V_{t-1}(\mathbf{x}) + (1-\alpha)\tau_{\text{occupancy}}, \quad \forall \mathbf{x} \in \text{Unobserved}(V_t) \tag{3}$$

D. Motion Planning

After segmentation and reconstruction, we are left with a collection of voxel maps approximately representing individual objects which must be rearranged to facilitate target retrieval. We employ a randomized kinodynamic motion planner with heterogeneous action types to find an action to perform in the current scene. Acting in clutter often restricts the feasibility of traditional pick-and-place actions due to limited reachability around objects. For these reasons, this work follows others, including [32], [33], [6], in employing action primitives to act in constricted space.

1) Domain Definitions: When planning for the motion of rigid bodies in 3D, the natural planning space is the Cartesian product of n_{obj} copies of $\mathbb{SE}(3)$, where n_{obj} is the number of rigid bodies. Combined with the robot's joint configuration space $\mathcal{Q}$, we can form the full configuration space

$$\mathcal{C} \doteq \underbrace{\mathbb{SE}(3) \times \cdots \times \mathbb{SE}(3)}_{n_{\text{obj}}} \times \mathcal{Q} \tag{4}$$

representing the state of the robot and all manipulable rigid bodies in the scene.

The action space $\mathcal{A}$ contains all the possible control actions the system can take. Each action $a \in \mathcal{A}$ has an associated parameter space $\mathcal{P}_a$ and performs the mapping

$$a \colon \mathcal{C} \times \mathcal{P}_a \to \mathcal{C} \tag{5}$$

Each action is associated with a predicate function Φ_a which determines the feasibility of performing a at state $\mathbf{x} \in \mathcal{C}$

$$\Phi_a \colon \mathcal{C} \times \mathcal{P}_a \to \{\, 0, 1 \,\} \tag{6}$$

The set of feasible actions from a state is then represented by

$$\mathcal{A}(\mathbf{x}) \doteq \{\, a \in \mathcal{A} \mid \Phi_a(\mathbf{x}) = 1 \,\} \tag{7}$$

Each action may also be equipped with a steering policy,

$$\pi_a \colon \mathcal{C} \times \mathcal{C} \to \mathcal{A} \times \mathcal{P}_a \tag{8}$$

which takes an initial and goal configuration and returns the parameterized action to locally advance toward the goal state. For actions without steering, it is necessary to sample from the parameter space directly.

In the absence of a known terminal state, we instead supply a reward function that determines the most promising action given the current and predicted next state.

$$\nu \colon \mathcal{C} \times \mathcal{P}_a \times \mathcal{C} \to \mathbb{R} \tag{9}$$

The function ν rewards actions that are likely to exhibit high information gain and penalizes trajectories with high incidence of collision. Collisions between the robot and scene objects during action execution are not forbidden in the framework, as the objects are movable, but actions with lower contact are rewarded.

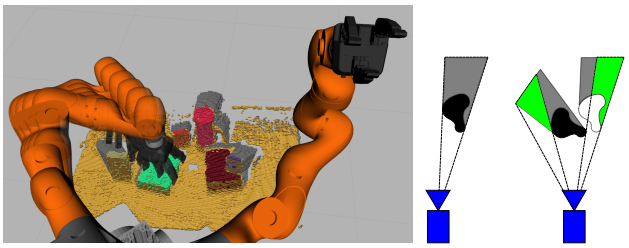
2) Object Selection: After segmentation and shape completion, occluded voxels ("shadows") in the field of view are computed by raycasting from unknown cells back to the camera origin. The object to move is determined by uniformly sampling from the set of shadow voxels, then casting back to select the object that occluded that volume. This provides a heuristic for sampling objects that are more likely to be hiding the target object. In the case where the target object is partially visible, but unreachable due to gripper collisions with its neighbors, the selector has a τ_{greedy} chance of choosing one of those colliding objects (with probability proportional to the number of gripper poses it obstructs and the number of target voxels occluded by the object), and a $1 - \tau_{\text{greedy}}$ chance of proceeding normally. Function SelectObject in Algorithm 1 highlights this process.

Algorithm 1 Object Selection and Motion Generation

```
function ComputeOcclusions(Octree T, Camera c)
    Octree T_occl
    for all p ∈ T do
        T_occl[p] ← (RayCast(T, c, p) ≠ p)
    end for
    return T_occl
end function
function SelectObject(Octree T, Octree T_occl, Camera c,
SegmentLookup S)
    p̃ ← RandomOccupiedNode(T_occl)
    p ← RayCast(T, c, p̃))
    return S[p]
end function
function GenerateMotion(Parameters P_a)
    PriorityQueue q
    for i ∈ [1 ... n_samples] do
        a ← SampleAction(P_a)
        obj ← SelectObject
        G ← GetGrasp(obj)
        r ← ν(a)
        ξ ← GenerateTrajectory(a, G)
        if ξ then Push(q, r, ξ)
        end if
    end for
    return Pop(q)
end function
```

3) Action Specifications: For the tabletop manipulation domain, we have chosen three heterogeneous actions representing a taxonomy based on the controllable subspace of the full state space: PUSH, SLIDE, and PICK. Each action operates on a single selected object, although in clutter this will likely influence the neighborhood of objects around it. PUSH, parameterized by $\mathcal{P}_{\text{PUSH}}$, represents a 1D palm-push motion with a magnitude and direction in the plane of the table surface. SLIDE is implemented by grasping the selected object and dragging in the table plane, for 3 controllable dimensions. It is parameterized by $\mathcal{P}_{\text{SLIDE}}$, which contains the $\mathbb{SE}(2)$ transform of the object motion. Finally, PICK and $\mathcal{P}_{\text{PICK}}$ represent a full grasp of the object, and can move it in $\mathbb{SE}(3)$.

Each task-space motion of an object has a generating policy that produces full joint-space motions of the robot. This consists



(a) (b) (c)

Fig. 6. Motion planning scene and motion rewards. 6(a) Unclassified occupied space is represented with gold voxels, shape-completed segments with random colors, and shadow voxels with gray. A SLIDE motion primitive is displayed with the robot motion trail. The right hand side shows dis-occluded voxels after an object motion. 6(b) shows a shape with its shadow given a camera pose. 6(c) shows the scene after an object motion, with the newly visible regions highlighted in green. The volume of these regions determines the reward.

of planning a collision free path to the start of the trajectory, then solving coherent inverse kinematics for a Cartesian path of the end-effector.

4) Feasibility Function: The feasibility function Φ_a is composed of two components. First, the entire generated trajectory must be kinematically feasible. Second, the initial pose of the hand must be free from collisions. Collisions between the remainder of the robot trajectory and the scene objects are permitted, and are handled by the reward function.

5) Reward Function: Given limited knowledge of the scene's dynamics and state, the reward function ν plays a dominant role in enabling progress toward locating the target object. The reward used here is a linear combination of a number of heuristic value functions ν_i:

$$\nu = \begin{bmatrix}\nu_1 & \nu_2 & \nu_3 & \nu_4\end{bmatrix} \mathbf{w} \tag{10}$$

For our constituent reward functions, we use the following elements:

Information ν_1: The number of previously occluded voxels that should be revealed by this motion.

Dispersion ν_2: The standard deviation of the centroids of detected objects.

Direction ν_3: The motion of the object toward or away from the center of mass of the scene.

Collision ν_4: The number of collisions between the robot trajectory and non-manipulated objects in the current motion.

No penalty is assessed for disturbing or toppling other objects, other than the collision metric. The weight vector $\mathbf{w}$ will in general depend highly on the resolution of the voxel map.

The Information heuristic requires some additional explanation. When moving a partially-visible object with shadow, the hidden voxels could "belong" either to the object in motion or to the remainder of the scene. If the object motion is represented by a rigid transform $T^{\mathbf{s}}_{\mathbf{s}'}$ and a shadowed voxel coordinate by $\mathbf{p}_{\text{occl}}$, then we need to check whether either $\mathbf{p}_{\text{occl}}$ or $\mathbf{p}'_{\text{occl}} = T^{\mathbf{s}}_{\mathbf{s}'}\mathbf{p}_{\text{occl}}$ are visible given the new object position. Figures 6(b) and 6(c) show this process.

6) Grasp Generation: For both the cases with and without shape completion, a convex-hull-based grasping generator was used. First, a convex prism of the object was generated by extruding the 2D convex hull of the object's downprojected (X-Y) occupancy map between its minimum and maximum extents in the table z-coordinate. For each edge of the 2D convex hull, the end-effector's palm frame was placed at the midpoint of the edge, plus the midpoint of the projection of the rest of the shape onto that edge's normal, with an additional static offset.Grasp poses were ranked by edge length before passing on to the motion planning.

V. EXPERIMENTAL RESULTS

Experimental Equipment To evaluate the system described, we employed a custom bimanual robot equipped with two KUKA LBR iiwa 7 R800 arms, two Robotiq 3-Finger Adaptive Grippers, and a Microsoft Kinect 2 for vision. External localization of the robot, camera, and table was provided by eight Vicon Bonita 10 motion capture cameras (no motion capture was used for the manipulated objects). Primary scene processing and motion planning was performed on a PC with an Intel 4.7GHz i7-8700K CPU and NVIDIA GTX 1080Ti GPU. Shape completion was also performed on a 1080Ti, and segmentation was performed on an NVIDIA Tesla V100-SXM2. The system, spanning six PCs including hardware-facing machines, used ROS for interprocess communication, sensor data acquisition, and trajectory transmission.

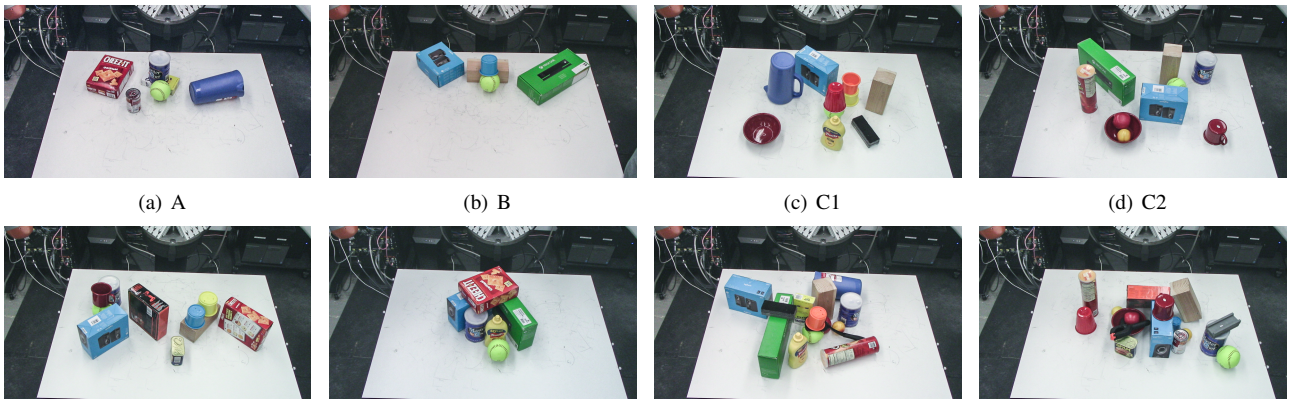
Experimental Parameters Throughout the preceding sections, several threshold and weight parameters were employed. For this experiment configuration, we used values of $\tau_{\text{target}} = 0.5$, $\tau_{\text{greedy}} = 0.9$, $\tau_{\text{occupancy}} = 0.5$, and from Equation 10 $\mathbf{w} = \begin{bmatrix}\frac{1}{2000} & 1 & 3 & -5\end{bmatrix}^{\mathsf{T}}$.

A. Shape Completion

a) Experiment setup: To benchmark the shape completion modifications, we generated voxel grids for 16 objects in a variety of previously unseen orientations, and where four of the objects were previously unseen by the network. One quadrant of the view was then occluded to mimic the conditions found in realistic scenes. 864 data points are generated for each object and randomly split into training and testing sets with the ratio of 4:1. The testing dataset also include 3 new objects which were not in the training set. The resulting reconstructions were then compared using the chamfer distance metric from Equation 2. These error statistics are collected for all 8,448 data points in Figure 8. We then evaluated two hypotheses about our shape completion method:

Hypothesis 1 *Training shape completion with a dataset of occluded objects significantly improves performance.*

As seen in Figure 8, augmenting the training data with occlusions provides a dramatic improvement in performance (note the log scale on the y-axis). Comparing the 3D-RecGAN model with the 3D-RecGAN + occlusions model yields a t-statistic of 36.752 and a p-value of ≈ 0. Examples of some of these improvements can be seen in Figure 9. This hypothesis is strongly supported by the results.



(a) A (b) B (c) C1 (d) C2

(e) C3 (f) D1 (g) D2 (h) D3

Fig. 7. Experimental configurations. In each scene, the target object is the yellow/green softball. The robot is located at the top of the image where it cannot see the softball.

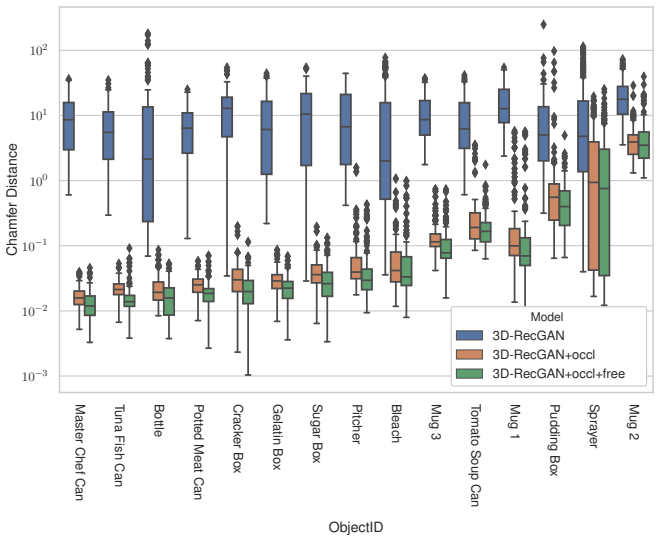
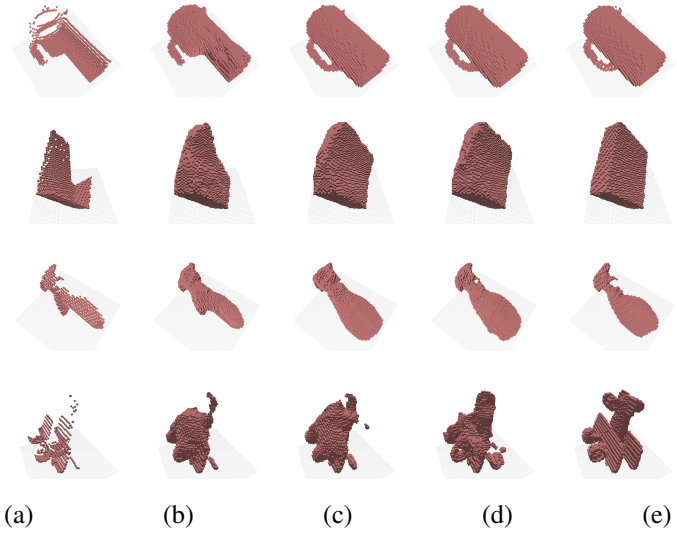


Fig. 8. Chamfer distances of various shape completion methods.



(a) (b) (c) (d) (e)

Fig. 9. Visualization of results of shape completion on several objects. (a) Input voxels; (b) 3D-RecGAN trained on unoccluded dataset; (c) 3D-RecGAN trained on occluded dataset; (d) 3D-RecGAN trained on occluded dataset with free space as augmented input. (e) Ground truth. The pitcher is in the training set. The box, sprayer and toy airplane are not in the training set.

Hypothesis 2 *Including known free voxel information in shape completion significantly improves performance.*

On top of the additions to the training dataset, Section IV-B1 described a modification to the original 3D-RecGAN architecture to account for known free space. Figure 8 shows these results as well, showing a modest improvement from the non-freespace network. Computing the t-statistics for the freespace and non-freespace leads to a t-value of 1.87543, and a p-value of 0.06079. Thus, the data does support an improvement in the performance of the network, but it falls short of the commonly accepted p-value benchmark of 0.05.

B. Tests in manually-designed scenes

In order to show the capabilities of shape completion and memory, we test each of these components in manually-designed scenarios where that component is beneficial. We then compare the results to using the baseline (the framework without either of these components). These test scenarios are shown in Figures 7(a) and 7(b).

In all scenes, the target object is the yellow ball, and the scene is considered to be successfully solved when the robot picks the ball from the scene using either hand. An attempt is marked as a failure if the target object is ejected from the workspace during the course of the attempt, or if the number of actions taken is more than three times the number of objects in the scene.

Hypothesis 3 *Memory significantly reduces the number of actions necessary to retrieve a target object when other objects are likely to be investigated first.*

Scene A was designed to explore the benefits of volumetric memory in locating a hidden object in the scene. Here the pitcher casts a much larger shadow than the coffee can behind which the target is hiding, but after one or two moves the system should realize that the target is not there, and should prioritize other objects. However, the memoryless system continues to

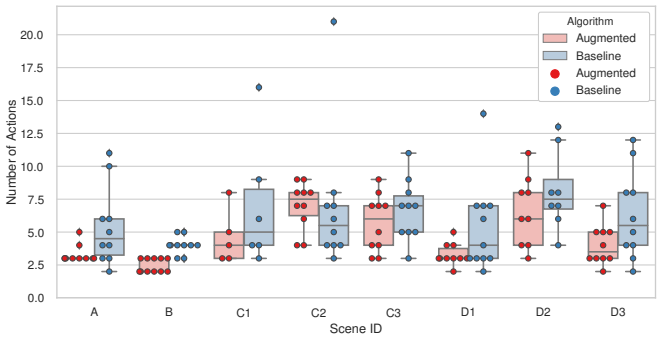


Fig. 10. Recorded action depths required to successfully retrieve the target object from each scene.

investigate the pitcher, leading to poor performance. Figure 10 shows that in most cases, the memory-enabled system first checked the largest occluded region (the pitcher) and then moved on to check the cylinder. Table I shows strong improvement as well, with a t-statistic of 5.8 and a p-value of 0.00002. This furnishes a compelling argument that memory greatly improves the system performance.

Hypothesis 4 *Shape completion significantly reduces the number of actions necessary to retrieve a target object when large occluded areas are a part of visible objects.*

Scene B was constructed to demonstrate the capability of shape completion to rule out some of the occluded area for exploration because it is a part of visible objects. In the scenario, the target is hidden behind a small cylinder, while there are two large boxes as distractions. The boxes have large shadows, so the baseline would be biased to move those to find the hidden object. However, with accurate shape completion, the system should realize that most of the volume shadowed by the boxes is likely part of the boxes themselves, and thus should move the cylinder. Figure 10 shows that about half of the time the augmented system decides to move the cylinder first, retrieving the target in the optimal two moves. In the other cases, the augmented system selects one of the side boxes first, or needs to clear an obstacle in contact with the target before grasping. Table I shows improvement over the baseline, with a t-statistic of 2.1 and a p-value of 0.056. Thus, this hypothesis is supported by the experiment, though not strongly.

C. Tests in arbitrary cluttered scenes

To assess the performance of the framework as a whole (including both memory and shape completion), we tested the full framework vs. the baseline (without shape completion or memory) on arbitrary sparsely-cluttered scenes and densely-cluttered scenes. We generated a collection of arbitrary sparse and dense clutter scenes, shown in Figure 7 as C1-C3 and D1-D3. For our purposes, "dense" clutter is defined to be where most or all objects in the scene are in contact with one another.

Hypothesis 5 *Our full framework significantly reduces the number of actions necessary to retrieve a target object in sparsely-cluttered scenarios.*

TABLE I
PERFORMANCE STATISTICS OF AUGMENTED VS. BASELINE

	Moves				Success Ratio	
	Augmented		Baseline			
	Mean	StdErr	Mean	StdErr	Augmented	Baseline
A	**3.300**	0.213	5.400	0.945	1.000	1.000
B	**2.455**	0.157	4.000	0.211	1.000	1.000
C1	**4.600**	0.927	7.000	2.000	0.833	**1.000**
C2	7.000	0.577	6.900	1.650	**1.000**	0.769
C3	**5.700**	0.684	6.700	0.731	**1.000**	0.909
D1	**3.300**	0.260	5.364	1.038	**1.000**	0.917
D2	**6.556**	0.884	8.125	1.060	**0.900**	0.800
D3	**4.000**	0.471	6.300	1.065	1.000	1.000

Scenes C1-3 were constructed to resemble typical household clutter. Figure 10 and Table I show strong improvement on C1 and C3, but C2 shows a minor regression. Thus, the hypothesis is only weakly supported, with a t-statistic of 1.5 and a p-value of 0.14.

Hypothesis 6 *Our full framework significantly reduces the number of actions necessary to retrieve a target object in densely-cluttered scenarios.*

Scenes D1-3 were constructed to resemble typical household clutter that is more densely distributed than C. In all of these cases, the augmented system showed strong improvement, with a t-statistic of 2.6 and a p-value of 0.012, showing good support for the hypothesis.

D. Computation Time

Our framework is a proof-of-concept and has not been optimized for fast computation or execution. However, to gauge the practicality of our method, we collected statistics on average computation time used for each component: Preprocessing: 4.34s; Segmentation: 7.30s; Shape Completion: 1.67s; Memory $\approx$ 0; Action Selection: 7.32s, and Execution: 34.98s. These results show the the benefits of shape completion and memory come at a low computational cost as compared to the rest of the framework.

VI. CONCLUSIONS AND FUTURE WORK

This paper has presented a system for the volumetric completion of partially observed scenes and demonstrated that such a system can significantly reduce the number of motions required to retrieve a hidden object from dense clutter. In addition, we have shown that modifications to existing network architectures and datasets to include partially-occluded views and known free-space volumes can boost the performance of such networks in cluttered environments. Our future work focuses on enhancing the segmentation component by inferring better segmentations from videos of interactions.

REFERENCES

[1] R. Jonschkowski, C. Eppner, S. Höfer, R. Martín-Martín, and O. Brock, "Probabilistic multi-class segmentation for the amazon picking challenge," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 1–7.

[2] M. Schwarz, A. Milan, C. Lenz, A. Munoz, A. S. Periyasamy, M. Schreiber, S. Schüller, and S. Behnke, "Nimbro picking: Versatile part handling for warehouse automation," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 3032–3039.

[3] A. Milan, T. Pham, K. Vijay, D. Morrison, A. W. Tow, L. Liu, J. Erskine, R. Grinover, A. Gurman, T. Hunn, *et al.*, "Semantic segmentation from limited training data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1908–1915.

[4] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3d object reconstruction from a single depth view with adversarial learning," in *International Conference on Computer Vision Workshops (ICCVW)*, 2017.

[5] M. Stilman, J. U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3327–3332, 2007, ISSN: 10504729. DOI: 10.1109/ROBOT.2007.363986.

[6] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, 2012, ISSN: 09295593. DOI: 10.1007/s10514-012-9306-z.

[7] L. L. Wong, L. P. Kaelbling, and T. Lozano-Perez, "Manipulation-based active search for occluded objects," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2814–2819, 2013, ISSN: 10504729. DOI: 10.1109/ICRA.2013.6630966.

[8] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa, "Object search by manipulation," *Autonomous Robots*, vol. 36, no. 1-2, pp. 153–167, 2014, ISSN: 09295593. DOI: 10.1007/s10514-013-9372-x.

[9] J. K. Li, D. Hsu, and W. S. Lee, "Act to see and see to act: POMDP planning for objects search in clutter," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-November, pp. 5701–5707, 2016, ISSN: 21530866. DOI: 10.1109/IROS.2016.7759839.

[10] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," 2018.

[11] M. Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa, "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *International Journal of Robotics Research*, vol. 31, no. 8, pp. 951–973, 2012, ISSN: 02783649. DOI: 10.1177/0278364911436018.

[12] C. Choi and H. I. Christensen, "3D pose estimation of daily objects using an RGB-D camera," *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=6386067.

[13] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2018.

[14] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5431–5440.

[15] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen, "Dense 3d object reconstruction from a single depth view," in *TPAMI*, 2018.

[16] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017, pp. 2442–2447.

[17] S. Tulsiani, S. Gupta, D. Fouhey, A. A. Efros, and J. Malik, "Factoring shape, pose, and layout from the 2d image of a 3d scene," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image.," in *CVPR*, vol. 2, 2017, p. 6.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, IEEE, 2017, pp. 2980–2988.

[20] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *CVPR*, Jul. 2017.

[21] T. Pham, T.-T. Do, N. Sünderhauf, and I. Reid, "SceneCut: Joint Geometric and Object Segmentation for Indoor Scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[22] P. Arbelaez, "Boundary extraction in natural images using ultrametric contour maps," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, IEEE, 2006, pp. 182–182.

[23] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool, "Convolutional oriented boundaries," in *European Conference on Computer Vision (ECCV)*, 2016.

[24] ——, "Convolutional oriented boundaries: From image segmentation to high-level tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[25] P. K. Nathan Silberman Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[26] 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, Feb. 2013, ISSN: 0929-5593. DOI: 10.1007/s10514-012-9321-0.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[28] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *Advanced Robotics (ICAR), 2015 International Conference on*, IEEE, 2015, pp. 510–517.

[29] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 4304–4311.

[30] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[31] P. Min, *Binvox*, `http://www.patrickmin.com/binvox`, Accessed: 2018-05-01, 2004 - 2017.

[32] M. Gupta, J. Müller, and G. S. Sukhatme, "Using Manipulation Primitives for Object Sorting in Cluttered Environments," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 608–614, 2015, ISSN: 15455955. DOI: 10.1109/TASE.2014.2361346.

[33] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, AAAI, Jan. 2015.