Unseen Object Instance Segmentation for Robotic Environments

Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox, *Fellow, IEEE*

***Abstract*—In order to function in unstructured environments, robots need the ability to recognize unseen objects. We take a step in this direction by tackling the problem of segmenting unseen object instances in tabletop environments. However, the type of large-scale real-world dataset required for this task typically does not exist for most robotic settings, which motivates the use of synthetic data. Our proposed method, unseen object instance segmentation (UOIS)-Net, separately leverages synthetic RGB and synthetic depth for unseen object instance segmentation. UOIS-Net is composed of two stages: first, it operates only on depth to produce object instance center votes in 2D or 3D and assembles them into rough initial masks. Second, these initial masks are refined using RGB. Surprisingly, our framework is able to learn from synthetic RGB-D data where the RGB is nonphotorealistic. To train our method, we introduce a large-scale synthetic dataset of random objects on tabletops. We show that our method can produce sharp and accurate segmentation masks, outperforming state-of-the-art methods on unseen object instance segmentation. We also show that our method can segment unseen objects for robot grasping.**

***Index Terms*—Robot perception, sim-to-real, unseen object instance segmentation.**

I. Introduction

FOR a robot to work in an unstructured environment, it must have the ability to recognize new objects that have not been seen before. Assuming every object in the environment has been modeled is infeasible and impractical. Recognizing unseen objects is a challenging perception task since the robot needs to learn the concept of "objects" and generalize it to unseen objects. Building such a robust object recognition module is valuable for robots interacting with objects, such as picking up unseen objects or learning to use new tools [1]–[3]. A common environment in which manipulation tasks take place is on tabletops. Thus, we approach this by focusing on the problem of unseen object instance segmentation (UOIS), where the goal is to segment every arbitrary (and potentially unseen) object instance, in tabletop environments.

In order to ensure the generalization capability of the module to recognize unseen objects, we need to learn from data that contains large amounts of various objects. However large-scale datasets with this property do not exist. Since collecting a large dataset with manual annotations is expensive and time-consuming, it is appealing to utilize synthetic data for training, such as using the ShapeNet repository which contains thousands of 3D objects [4]. However, there exists a domain gap between synthetic data and real-world data as many simulators do not provide realistic-looking images. Training directly on such synthetic data only usually does not work well in the real world [5], and synthesizing photo-realistic images with physics-based rendering can be computationally expensive [6], making large photorealistic synthetic datasets impractical to obtain.

Consequently, recent efforts in robot perception have been devoted to the problem of sim-to-real, where the goal is to transfer capabilities learned in simulation to real-world settings. For instance, some works have used domain adaptation techniques to bridge the gap when unlabeled real data is available [7], [8]. Domain randomization [9] was proposed to diversify the rendering of synthetic data for training. While these techniques attempt to fix the discrepancy between synthetic and real-world RGB, models trained with synthetic depth have been shown to generalize reasonably well for simple settings such as bin-picking [10], [11]. However, in more complex settings, noisy depth sensors can limit the application of such methods, and models trained on RGB have been shown to produce accurate masks [12]. An ideal method should combine the generalization capability of training on synthetic depth and the ability to produce sharp masks by utilizing RGB.

In this work, we investigate how to utilize synthetic RGB-D images for UOIS in tabletop environments. We show that simply combining synthetic RGB images and synthetic depth images as inputs does not generalize well to the real world. To tackle this problem, we propose a two-stage network architecture called UOIS-Net that separately leverages the strengths of RGB and depth for UOIS. Our first stage is a depth seeding network (DSN) that utilizes only depth to produce object instance center votes, which are then used to compute rough initial instance masks. We compare multiple architectures for the DSN that produce center votes in 2D and 3D. Training the DSN with depth images allows for better generalization to real-world data. However, the initial

Manuscript received July 14, 2020; revised November 30, 2020; accepted February 8, 2021. This work was supported in part by the National Science Foundation under Contract NSF-NRI-1637479, in part by the Office of Naval Research under ONR Grant 63-6094, and in part by Honda Research Institute as part of the Curious Minded Machine initiative. This paper was recommended for publication by Associate Editor J. Bohg and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: Christopher Xie.)*

Christopher Xie is with the School of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: chrisxie@cs.washington.edu).

Yu Xiang and Arsalan Mousavian are with the NVIDIA, Santa Clara, CA 95051 USA (e-mail: yux@nvidia.com; arsalan.mousavian@gmail.com).

Dieter Fox is with the School of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA, and also with the NVIDIA, Santa Clara, CA 95051 USA (e-mail: fox@cs.washington.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2021.3060341.

Digital Object Identifier 10.1109/TRO.2021.3060341

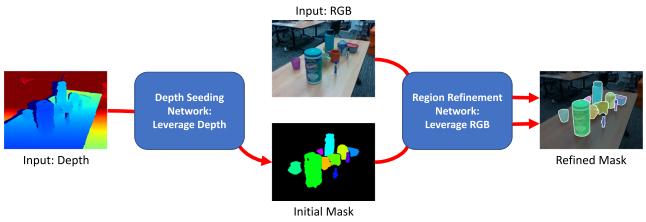


Fig. 1. High level overview of our proposed two-stage framework. The first stage leverages depth only to produce rough initial masks. The second stage then leverages RGB to refine the initial masks to produce accurate, sharp instance masks.

masks from the DSN may contain inaccurate object boundaries due to depth senor noise. In this case, exploiting textures in RGB images can significantly help.

Thus, our second stage is a region refinement network (RRN) that takes an initial mask from the DSN and an RGB image as input and outputs a refined mask. Our surprising result is that, conditioned on initial masks, our RRN can be trained on non-photorealistic synthetic RGB images without adopting any of the afore-mentioned Sim-to-Real solutions. We posit that mask refinement is an easier problem than directly using RGB as input to produce masks, mainly because the mask refinement uses a local image patch as input and focuses on a single object. We empirically show robust generalization across many different objects in cluttered real-world data. In fact, our RRN works almost as well as if it were trained on real data. Our framework produces sharp and accurate masks even when the depth images are noisy. We show that it outperforms state-of-the-art (SOTA) methods including Mask R-CNN [12] and PointGroup [13]. Fig. 1 illustrates our two-stage framework.

To train our method, we introduce a synthetic dataset of tabletop objects in home environments, which we name tabletop object dataset (TOD). Our dataset consists of indoor scenes of random ShapeNet [4] objects on random ShapeNet tables. We use the PyBullet physics simulator [14] to generate the scenes and render depth and nonphotorealistic RGB. Training our proposed method on this dataset results in SOTA results on multiple real-world datasets for UOIS.

We extend our previous work, UOIS-Net-2D [15], by introducing a novel DSN architecture that reasons in 3D space. As we show in Section VIII, this architecture solves many limitations of producing these center votes in 2D (our previous work), thus providing stronger performance. Additionally, we propose a novel loss function that encourages separation of the center vote clusters and show that this loss function is crucial to achieving strong performance in cluttered environments.

II. Related Works

A. Category-Level Object Segmentation

2D semantic segmentation involves assigning pixels in an image to a set of known classes. Deep learning has emerged as the most popular tool for solving this problem [16]–[20]. Shelhamer *et al.* [16] first introduced the concept of using a fully convolutional architecture (FCN). Chen *et al.* [17] designed an architecture that utilizes dilated convolutions in order to increase the receptive field. Refs. [18], [19] further improve performance by introducing decoder architectures on top of the encoders. Lin *et al.* [20] proposed a multipath refinement network with long-range residual connections to enable high-resolution predictions. These methods have demonstrated strong performance on datasets such as PASCAL [21] and COCO [22].

Much work has been devoted to solving the semantic segmentation problem in 3D as well. A common representation of 3D space is voxels; however, operating with voxel grids as input can be expensive both computationally and memory-wise. Thus, Graham *et al.* [23] introduced a submanifold sparse convolutional operator to preserve spatial sparsity of the input. Choy *et al.* [24] further generalized these sparse convolutions to arbitrary kernel shapes, improving performance. Other 3D methods utilize point clouds. Qi *et al.* [25] proposed PointNet, a permutation-invariant network architecture to handle point clouds, and Qi *et al.* [26] extended this to a hierarchical network that recursively applies PointNet in order to obtain multiresolution features, similar to deep convolutional networks with decreasing resolutions via strides/max pooling.

The advent of RGB-D sensors such as Kinect allowed the research community to utilize both modalities for semantic segmentation, and drove the creation of datasets such as [27]. Ren *et al.* [28] investigated a combination of kernel descriptors, support vector machines, and Markov random fields for indoor scene segmentation. Both [29] and [30] leverage RGB-D videos, extracting 2D features from each RGB frame and integrating them with a reconstructed voxel representation of the scene. Deep learning-based approaches include [31]–[33]. Gupta *et al.*[31] proposed the Horizontal disparity, Height above ground, and the Angle the pixel's local surface normal makes with the inferred gravity direction (HHA) encoding of depth images. The authors used this encoding to design an object detection system, which they further exploited to improve semantic segmentation performance. Shelhamer *et al.*[16] also used the HHA in their pioneering work on FCNs. Wang and Neumann [32] proposed a depth-aware convolution and pooling mechanism to incorporate geometry into the convolution operators to build a depth-aware receptive field. Qi *et al.* [33] used the output of a 2D segmentation network to initialize node features of a graph neural network applied on a 3D point cloud which was backprojected from a depth image. In this work, we also leverage RGB-D images, but focus on segmenting each individual object with unknown object class.

B. Instance-Level Object Segmentation

2D object instance segmentation is the problem of segmenting every object instance in an image. Many approaches for this problem involve top-down solutions that combine segmentation with object proposals in the form of bounding boxes [12], [34]–[36], typically produced by a region proposal network (RPN). Fully convolutional instance-aware semantic segmentation (FCIS) [34] utilizes position-sensitive inside/outside score maps for fully end-to-end convolutional instance segmentation. Mask R-CNN [12], a prominent work in the field, predicts a foreground mask for each object proposal. Chen *et al.* [35] build

on top of both FCIS and Mask R-CNN and exploit semantic segmentation and direction predictions to assemble foreground masks. Kirillov *et al.* [36] propose a module that iteratively refines segmentation predictions at adaptively selected locations, which can be used in conjunction with Mask R-CNN.

However, when bounding boxes contain multiple objects (e.g., cluttered robot manipulation setups), the true instance mask is ambiguous and these methods struggle. Recently, a few methods have investigated bottom-up methods which assign pixels to object instances [37]–[40]. Other methods examine dense sliding-window instance segmentation on 4D tensors [41], combining top-down and bottom-up methods via blending modules [42], and alternative mask representations such as contours [43]. Additionally, interactive instance segmentation has shown strong results with few user inputs [44].

Most of the aforementioned algorithms provide instance masks with category-level semantic labels, which do not generalize to unseen objects in novel categories. One approach to adapting these techniques to unseen objects is to employ "class-agnostic" training, which treats all object classes as one foreground category [11]. One family of methods exploits motion cues with class-agnostic training in order to segment arbitrary moving objects [45], [46]. Another family of methods are class-agnostic object proposal algorithms [47]–[49]. However, these methods will segment everything and require some postprocessing method to select the masks of interest. [50] jointly estimates instance segmentation masks and rigid scene flow, similar to [51], [52]. We also train our proposed method in a class-agnostic fashion, but instead focus our notion of unseen objects in particular environments such as tabletop settings.

In 3D instance segmentation, researchers have recently been investigating architectures to apply on point clouds/voxel grids. Hou *et al.* [53] introduced the first deep learning method to fuse RGB and geometric information from RGB-D scans. Han *et al.* [54] proposed an occupancy term, which greatly aids supervoxel clustering, leading to strong results. A few of these methods embrace center voting-based techniques [13], [55]–[58]. Lahoud *et al.* [55] utilize metric learning to learn abstract features, and predicts center votes which are postprocessed by meanshift clustering. Qi *et al.* [56] use the center votes with a simple grouping mechanism to detect 3D bounding boxes of objects from point clouds only. Their follow-up work [57] incorporates RGB information by lifting 2D votes and features into 3D. Engelmann *et al.* [58] follow a similar architecture but stack a graph convolutional network to refine proposal features. Jiang *et al.* [13] also perform clustering on votes and semantic features for targeted performance on certain object classes. Our method takes inspiration from these voting-based methods, but is targeted to cluttered robot environments.

C. Sim-to-Real Perception

Training a model on synthetic RGB and directly applying it to real data typically fails [5]. Many methods employ some level of rendering randomization [9], [59]–[63], including lighting conditions and textures. However, they typically assume specific object instances and/or known object models. Another family of methods employ domain adaptation to bridge the gap between simulated and real images [7], [8]. Algorithms trained on depth have been shown to generalize reasonably well for simple settings [10], [11]. However, noisy depth sensors can limit the application of such methods. Our proposed method is trained purely on (nonphotorealistic) synthetic RGB-D data and is accurate even when depth sensors are inaccurate, and can be trained without adapting or randomizing the synthetic RGB.

III. Method Overview

Given a single RGB-D image, the goal of our algorithm is to produce object instance segmentation masks for all objects on a tabletop, where the object instances (or even the semantic class) are arbitrary and are not assumed to have been seen during a training phase. These masks do not have any notion of class categorization or semantics. These masks can be employed by robots for interacting with unseen object instances in downstream applications such as grasping and/or manipulation.

Our framework consists of two separate networks that process Depth and RGB separately to produce instance segmentation masks. First, we design a depth seeding network (DSN) that takes a depth image as input and outputs *initial* object instance segmentation masks. These initial masks can be quite noisy for a number of reasons, thus we design an initial mask processor (IMP) to robustify them with standard image processing techniques. We further refine the processed initial masks using our region refinement network (RRN), which is designed to snap the noisy initial mask edges to object edges in RGB, providing sharp and accurate final instance masks. The full architecture is shown in Fig. 2.

Because the DSN incorporates non-differentiable techniques in order to build the initial masks, our DSN and RRN are trained separately as opposed to end-to-end. Both the DSN and RRN can be trained fully in simulation with no fine-tuning on real-world data, allowing our framework to capitalize on large amounts of simulated scenes and objects without resorting to the expensive process of annotating data. Our framework generalizes remarkably well to real-world scenarios despite being trained only on nonphotorealistic simulated data, enabling robotic tasks with unseen objects.

IV. Depth Seeding Network

It has been shown that depth generalizes reasonably well for sim-to-real problems [10], [11], [64]. Inspired by this concept, we focus the first stage of our framework on depth only to produce initial class-agnostic instance segmentation masks. At a high level, the DSN takes as input a three-channel organized point cloud, $D \in \mathbb{R}^{H \times W \times 3}$, of XYZ coordinates, and outputs initial instance segmentation masks. Note that D can be computed by backprojecting a depth map given camera intrinsics.

We examine two methods of structuring the DSN. First, we investigate building initial masks by predicting centers in 2D pixel space. While this method provides SOTA results, it has some obvious pitfalls (examined in Section VIII) that motivates a novel architecture that builds masks by predicting centers in 3D space.

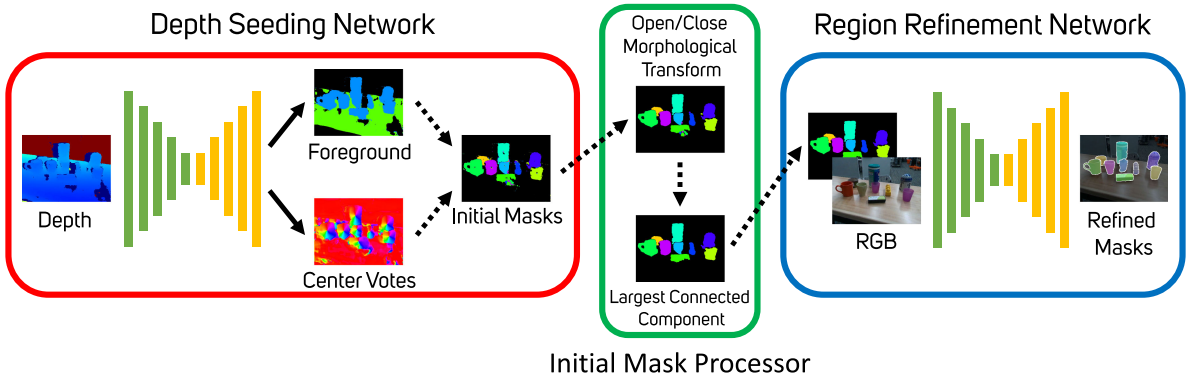


Fig. 2. Overall architecture. The depth seeding network (DSN) is shown in the red box, the initial mask processor (IMP) in the green box, and the region refinement network (RRN) in the blue box. The images come from a real example taken by an RGB-D camera in our lab. Despite the level of noise in the depth image (due to reflective table surface), our method is able to produce sharp and accurate instance masks. Gradients do not flow backwards through dotted lines.

A. Reasoning in 2D

1) Network Architecture: The organized point cloud D is passed through an encoder–decoder architecture to produce two outputs: a semantic segmentation mask $F \in \mathbb{R}^{H \times W \times C}$, where C is the number of semantic classes, and 2D directions to object centers $V \in \mathbb{R}^{H \times W \times 2}$. We use $C = 3$ for our semantic classes: background, tabletop, and tabletop objects. Each pixel of V encodes a two-dimensional unit vector pointing to the 2D center of the object. We define the center of the object to be the mean pixel location of the observable mask (part of mask that is unoccluded). Although we do not explicitly make use of the tabletop label in Section VIII, it can be used in conjunction with RANSAC [65] in order to better estimate the table for downstream applications. For the encoder–decoder architecture, we use a U-Net [66] architecture where each 3×3 convolutional layer is followed by a GroupNorm layer [67] and ReLU. The output of the U-Net is a feature map of shape $\mathbb{R}^{H \times W \times 64}$. Sitting on top of this is two parallel branches of convolutional layers that produce the foreground mask F and center directions V (Fig. 2).

In order to compute the initial segmentation masks from F and V, we design a Hough voting layer similar to [59]. We describe the pseudocode detailed in Algorithm 1. First, we discretize the space of angles $[0, 2\pi]$ into A equally spaced bins. For every pixel, we compute the percentage of discretized directions from all other foreground pixels that point to it and use this as a score for how likely the pixel is an object center (lines 3–10). We threshold when a foreground pixel points to it with an inlier threshold and distance threshold. We then threshold the percentages and apply nonmaximum suppression (NMS) to select object centers (line 11). Given these object centers, each pixel is assigned to the closest center it points to (line 12), which gives the initial masks as shown in the red box of Fig. 2. Note that the inlier, distance, and percentage thresholds provide the Hough voting layer with robustness. For example, if not enough foreground pixels from all directions point towards a potential object center, that center is not selected. This robustifies the algorithm by protecting against false positives. We qualitatively show the efficacy of these design choices in Section VIII-E.

2) Loss Functions: To train the DSN, we apply two different loss functions on the semantic segmentation F and the direction prediction V.

Foreground Loss: For the semantic segmentation F, we use a weighted cross-entropy as this has been shown to work well in detecting object boundaries in imbalanced images [68]. The loss is $\ell_{fg} = \sum_i w_i \ \ell_{ce}(F_i, \bar{F}_i)$, where i ranges over pixels, $F_i, \bar{F}_i$ are the predicted and ground truth probabilities of pixel i, respectively, and ℓ_{ce} is the cross-entropy loss. The weight w_i is inversely proportional to the number of pixels with labels equal to $\bar{F}_i$, normalized to sum to 1.

Direction Loss: We apply a weighted cosine similarity loss to the direction prediction V. The cosine similarity is focused on the tabletop object pixels, but we also apply it to the background/tabletop pixels to have them point in a fixed direction to avoid false positives. The loss is given by

$$\ell_{dir} = \sum_{i \in \mathcal{O}} \alpha_i \left(1 - V_i^{\intercal} \bar{V}_i\right) + \frac{\lambda_{bt}}{|\mathcal{B} \cup \mathcal{T}|} \sum_{i \in B \cup T} \left(1 - V_i^{\intercal} \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) \tag{1}$$

where $V_i, \bar{V}_i$ are the predicted and ground truth unit directions of pixel i, respectively. $\mathcal{B}, \mathcal{T}$, and $\mathcal{O}$ are the sets of pixels belonging to background, table, and object/foreground classes, respectively. Note that $\mathcal{B} \cup \mathcal{T} \cup \mathcal{O} = \Omega$, where Ω is the set of all pixels. α_i is inversely proportional to the number of pixels with the same *instance* label as pixel i, which gives equal weight to each instance regardless of size. We set $\lambda_{bt} = 0.1$. The total loss for our 2D DSN is given by $\ell_{fg} + \ell_{dir}$.

B. Reasoning in 3D

Reasoning in 2D has some failure cases that can be mitigated by reasoning in 3D. For example, if the center of an object is occluded by another object, the 2D center voting procedure will not detect that object (examples of this can be found in Section VIII-E). Thus, we propose a new architecture to the DSN to better handle these cases and provide stronger results. This formulation requires more sophisticated loss functions. In particular, we introduce a novel separation loss that significantly improves accuracy in cluttered scenes.

Algorithm 1: Hough Voting Procedure Pseudocode.

Require: F, V, cosine distance d_c, number of angle bins A. Robustness parameters: inlier threshold ϵ_{it}, distance threshold ϵ_d, percentage threshold ϵ_{pt}.

return Initial instance masks S
Initialize $\mathcal{H} \in \mathbb{R}^{H \times W \times A}$ to zeros
for potential center $p_c \in \Omega$ **do**
  for $p \in F$ **do**
    if $d_c(p_c - p, V_p) < \epsilon_{it}$ AND $d(p_c, p) < \epsilon_d$ **then**
      $a = \lfloor A \cdot d_c(p_c - p, [0, 1]) \rfloor$ # discretized angle
      $\mathcal{H}_{p_c,a} = 1$
    end if
  end for
end for
Compute local maximums $\mathcal{C}$ of $\mathcal{H}$ using NMS and ϵ_{pt}
Compute S with $\mathcal{C}, V$

1) Network Architecture: The network architecture of this 3D DSN is almost the same as the 2D DSN. The input is the same, which is the organized point cloud D of XYZ coordinates. The main difference between the 2D DSN and the 3D DSN is the output. Our 3D DSN still outputs the semantic segmentation mask F, but produces 3D offsets to object centers $V' \in \mathbb{R}^{H \times W \times 3}$ instead of 2D directions V. Note that elements in V' are not unit vectors, which is the case with V. Since V' are 3D offsets, $D + V'$ is the predicted object centers for each pixel, which we will refer to as "center votes."

We propose to modify the architecture to use dilated convolutions [69] in order to provide the DSN with a higher receptive field. We replace the 6th, 8th, and 10th convolution layers with efficient spatial pyramid (ESP) modules [70]. An ESP module is a lightweight module consisting of a *reduction* operation, a *split/transform* component that applies convolutions with different dilation rates to get a spatial pyramid, and a *merge* process that hierarchically fuses the feature maps of the spatial pyramid [70]. The ESP module has less parameters than the convolution layer it replaces, making it more computationally efficient. Details of the implementation can be found in the public code release at the project website.[1] We show in Section VIII-F that adding this module provides a boost in performance.

To compute initial masks, we perform mean shift clustering in 3D space over our center votes $D + V'$. Mean shift clustering is an iterative procedure to find the modes of a distribution approximated by a kernel density esimate (KDE). The number of clusters (objects, in our case) is not determined beforehand, but instead by the number of modes in the KDE. We use the Gaussian kernel $K(x, y) = \exp(\frac{1}{\sigma^2}\|x - y\|_2^2)$, which results in Gaussian mean shift (GMS) clustering. $\sigma > 0$ is a hyperparameter which affects the number of modes (objects) in the KDE. Thus, the choice of σ is crucial and depends on the relative distance between objects, which is low in clutter. A detailed review of mean shift clustering algorithms can be found in [71]. After clustering, each pixel is assigned to the cluster ID of its center vote to generate the initial masks. The clustering is only applied to the foreground pixels. Note that this method of producing initial masks lacks the thresholds such as $\epsilon_{it}, \epsilon_d, \epsilon_{pt}$ from the 2D DSN Hough voting layer that provide robustness.

2) Loss Functions: We apply four loss functions on the semantic segmentation F and center offsets V' to train the 3D-reasoning version of the DSN.

a) *Foreground Loss*: We utilize the same foreground loss as in Section IV-A2 for the 2D DSN, ℓ_{fg}.

b) *Center Offset Loss*: We apply a Huber loss ρ (Smooth L1 loss) to the center offsets V' to penalize the distance of the center votes to their corresponding ground truth object centers.

$$\ell_{co} = \sum_{i \in \Omega} w_i \rho \left(D_i + V'_i - c_i \right) \tag{2}$$

where c_i is the 3D coordinate of the ground truth object center for pixel i. Like ℓ_{fg}, the weight w_i is inversely proportional to the number of pixels with the same instance label y_i. For object centers that are out of view of the camera, we project them to the camera's field of view.

c) *Clustering Loss*: We adopt a clustering loss that unrolls GMS for a few iterations and applies a loss on the clustered points, very similar to [72]. GMS iteratively shifts a set of S 3D seed points, $Z \in \mathbb{R}^{S \times 3}$, to higher density regions of the KDE in a gradient ascent-type fashion [71]. Let $Z^{(l)}$ be the points at the lth iteration of GMS. $Z^{(0)}$ is initialized as the center votes $X = D + V' \in \mathbb{R}^{|\mathcal{O}| \times 3}$ of the foreground pixels. One iteration of GMS amounts to $Z^{(l+1)} = \tilde{D}^{-1} K X$ where $K \in \mathbb{R}^{S \times |\mathcal{O}|}$ is the kernel matrix s.t. $[K]_{ij} = K(Z_i^{(l)}, X_j)$, and $\tilde{D} = \text{diag}(K\mathbf{1})$ with $\mathbf{1} \in \mathbb{R}^{|\mathcal{O}|}$ being a vector of all ones. Note that K depends on σ. This iteration can be seen as a layer in the network with no parameters for learning. We apply the following loss function to $Z^{(l)}$ and X, with the corresponding object instance labels $Y \in \mathbb{R}^{|\mathcal{O}|}$:

$$\ell_{cl}^{(l)}(Z^{(l)}, X, Y) = \sum_{i=1}^{S} \sum_{j \in \mathcal{O}} w_{ij} \mathbb{1}\{y_i = y_j\} d^2(Z_i^{(l)}, X_j) + w_{ij} \mathbb{1}\{y_i \neq y_j\} [\delta - d(Z_i^{(l)}, X_j)]_+^2 \tag{3}$$

where w_{ij} are inverse proportional weights with respect to class size, $d(\cdot, \cdot)$ is the Euclidean distance, and $[\cdot]_+ = \max(\cdot, 0)$. This loss function influences the KDE modes to be close to its points, and at least δ away from points not belonging to the cluster, encouraging the points $X = D + V'$ to be more cluster-like.

Applying (3) to all points, i.e., $S = |\mathcal{O}|$, results in excessive memory usage, and thus we instead adopt a stochastic version of this loss function. We randomly sample an index set $\mathcal{I} \subset \{1, 2, \ldots, |\mathcal{O}|\}$ and set $Z^{(0)} = X_{\mathcal{I}}$, and run GMS clustering only on these points. We unroll GMS for L iterations, and apply $\ell_{cl}^{(l)}$ at each iteration, giving the full cluster loss $\ell_{cl} = \ell_{cl}^{(1)} + \cdots + \ell_{cl}^{(L)}$.

d) *Separation Loss*: We introduce a novel separation loss that encourages the center votes to not necessarily be at the center of an object, as long as it is far away from other object center votes in order to ease the postprocessing GMS clustering phase.

[1][Online]. Available: https://rse-lab.cs.washington.edu/projects/unseen-object-instance-segmentation/.

To do this, we consider the following tensor:

$$M_{ij} = \frac{\exp\left(-\tau\, d(c_j, D_i + V_i')\right)}{\sum_{j'=1}^{J} \exp\left(-\tau\, d(c_{j'}, D_i + V_i')\right)} \tag{4}$$

where c_j is the jth ground truth object center, $i \in \mathcal{O}$, and $\tau > 0$ is a hyperparameter. This is simply the distance from center vote $D_i + V_i'$ to all object centers scaled by τ, with a softmax applied. We apply a cross entropy loss $\ell_{sep}(M_{ij}) = -\sum_{j=1}^{N} \mathbb{1}\{y_i = j\} \log(M_{ij})$ in order to maximize M_{ij} when $y_i = j$.

Maximizing M_{ij} for a foreground pixel i and its corresponding GT object center j encourages 1) the center vote $D_i + V_i'$ to be close to c_j, and 2) to be far from $\{c_j \mid y_i \neq y_j\}$. While the ℓ_{co} also enforces property 1, property 2 is quite desirable in heavy clutter. If two objects are situated in a way such that their 3D object centers are very close, postprocessing clustering will be difficult. This separation loss encourages the network to predict object center votes that are not close to each other, making the task of postprocessing clustering easier. In Section VIII-F, we show that this loss is crucial for strong performance in heavy clutter.

In summary, the total loss used to train the 3D DSN is given by $\lambda_{fg}\ell_{fg} + \lambda_{co}\ell_{co} + \lambda_{cl}\ell_{cl} + \lambda_{sep}\ell_{sep}$.

V. Initial Mask Processing Module

Computing the initial masks from F and V/V' often results in noisy masks (see an example of initial masks computed from the 2D DSN using our Hough voting layer in Fig. 2). For example, these instance masks often exhibit salt/pepper noise and erroneous holes near the object center (see Section VIII-E for examples). As shown in Section VIII-D, the RRN has trouble refining the masks when they are scattered as such. To robustify the algorithm, we propose to use two simple image processing techniques to clean the masks before refinement.

For a single instance mask, we first apply an opening operation, which consists of mask erosion followed by mask dilation [73], removing the salt/pepper noise issues. Next we apply a closing operation, which is dilation followed by erosion, which closes up small holes in the mask. Finally, we select the largest connected component and discard all other components. Note that these operations are applied to each instance mask separately. These simple image processing techniques are immensely helpful in robustifying the system.

VI. Region Refinement Network

While depth generalizes reasonably well from sim-to-real, the initial masks (after IMP) are still subject to many errors due to noisy depth sensors. The RRN is designed to *snap* the initial mask edges to the object edges in RGB to provide accurate and sharp instance masks.

A. Network Architecture

Inspired by [44], this network takes as input a cropped 4-channel image, which consists of RGB concatenated with a single initial instance mask. The RGB image is cropped around the instance mask with some padding for context, concatenated with the (cropped) mask, and then resized to 224×224. This gives an input image $I \in \mathbb{R}^{224\times224\times4}$. The output of the RRN is the refined mask probabilities $R \in \mathbb{R}^{224\times224}$, which we threshold to get the final output. We use the same U-Net architecture as in the DSN. To train the RRN, we apply the loss ℓ_{fg} with two classes (foreground vs. background) instead of three.

B. Mask Augmentation

Recall that the DSN and RRN are trained separately. In order to train the RRN, we need examples of perturbed instance masks. While we could train the RRN with the outputs of the DSN, we found that they are typically too clean on our synthetic dataset and we achieved better results by perturbing the ground truth masks instead. This problem can be seen as a data augmentation task where we augment the mask into something that resembles an initial mask (after the IMP). We detail the different augmentation techniques used below.

1) Translation/Rotation: We translate the mask by sampling a displacement vector proportionally to the mask size. Rotation angles are sampled uniformly in $[-10°, 10°]$.
2) Adding/Cutting: For this augmentation, we choose a random part of the mask near the edge, and either remove it (cut) or copy it outside of the mask (add). This reflects the setting when the initial mask egregiously overflows from the object, or is only covering part of it.
3) Morphological Operations: We randomly choose multiple iterations of either erosion or dilation of the mask. The erosion/dilation kernel size is set to be a percentage of the mask size, where the percentage is sampled from a beta distribution. This reflects inaccurate boundaries in the initial mask, e.g. due to noisy depth sensors.
4) Random Ellipses: We sample the number of ellipses to add or remove in the mask from a Poisson distribution. For each ellipse, we sample both radii from a gamma distribution and a random rotation angle. This augmentation requires the RRN to learn to remove irrelevant blots outside of the object and close up small holes within it.

VII. Tabletop Object Dataset

Many desired robot environment settings (e.g., kitchens, cabinets) lack large-scale training data to train deep networks. To our knowledge, there is no large-scale dataset for unseen tabletop objects. To remedy this, we generate our own synthetic dataset which we name the TOD. This dataset is composed of 40 k synthetic scenes of cluttered ShapeNet [4] objects on a (ShapeNet) tabletop in SUNCG home environments [74]. We only use ShapeNet tables that have convex tabletops and filter the ShapeNet object classes to roughly 25 classes of objects that could potentially be on a table. Example classes include jar, mug, helmet, and pillow.

Each scene in the dataset is of a random room chosen from a random SUNCG house loaded without any furniture. We sample a ShapeNet table and scale it such that its height is in the range 0.75–1 m, and places it in the room so that it is not colliding with walls or other fixtures in the room. Next, we randomly sample anywhere between 5 and 25 objects to put on the table,

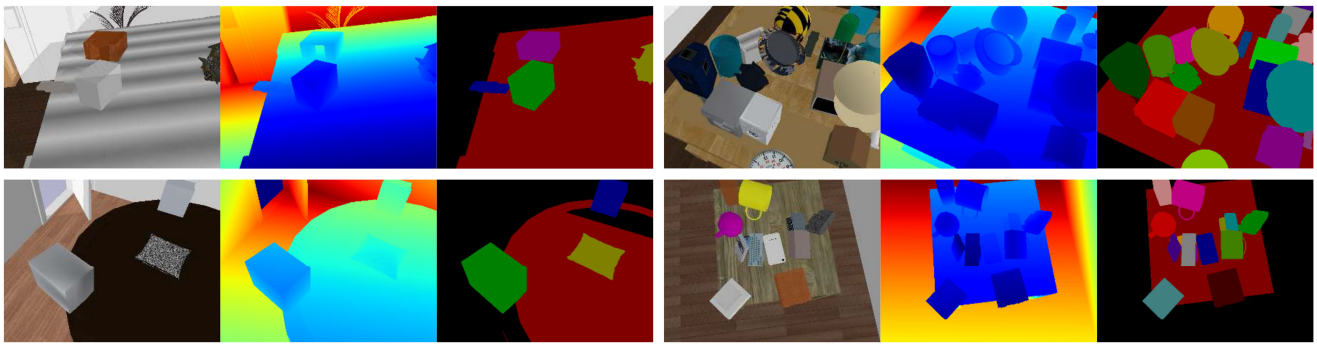


Fig. 3. Examples from our tabletop object dataset (nonphotorealistic). RGB, depth, and instance masks are shown.

rescaling them such that they are not larger than $\frac{1}{4}\min\{t_h, t_l\}$, where t_h, t_l are the height and length of the table, respectively. The objects are either randomly placed on the table, on top of another object (stacked), or generated at a random height and orientation above the table. We use PyBullet [14] to simulate physics until the objects come to rest and remove any objects that fell off the table. Next, we generate seven views: one view is of only background, another is of just the table in the room, and the rest are taken with random camera viewpoints with the tabletop objects in view. The viewpoints are sampled at a height of between 5 and 1.2 m above the table and rotated randomly with an angle in $[-12°, 12°]$. The images are generated at a resolution of 640×480 with vertical FOV of 45 degrees. The segmentation has a tabletop (table plane only) label and instance labels for each object.

We show some example images of our dataset in Fig. 3. The rightmost two examples show that some of our scenes are heavily cluttered. Note that the RGB looks nonphotorealistic. In particular, PyBullet is unable to load textures of some ShapeNet objects (see gray objects in leftmost two images). PyBullet was built for reinforcement learning, not computer vision, and thus its rendering capabilities are insufficient for photorealistic tasks [14]. Despite this, our RRN can learn to snap masks to object boundaries from this synthetic dataset.

VIII. Experiments

We evaluate our method on real datasets against the SOTA methods Mask R-CNN [12] and PointGroup [13]. We denote our method as *UOIS-Net-2D* or *UOIS-Net-3D*, depending on whether the DSN reasons in either 2D or 3D.

A. Implementation Details

All images have resolution $H = 480, W = 640$.

1) DSN: We train all 2D DSN models for 100 k iterations with stochastic gradient descent (SGD) using a fixed learning rate of 1e$-$2. We use a batch size of 8 and set $\lambda_{fg} = \lambda_{\text{dir}} = 1$. For the Hough voting layer, we discretize the angles into $A = 100$ bins, and set $\epsilon_{it} = 0.9, \epsilon_d = 20, \epsilon_{pt} = 0.5$. We also process every 10th pixel instead of every pixel in line 4 of Algorithm 1 for computational efficiency.

All 3D DSNs are trained for 150 k iterations with Adam [81] with initial learning rate of 1e$-$4. We use a batch size of 8, with $\lambda_{fg} = 3, \lambda_{co} = 5, \lambda_{cl} = \lambda_{sep} = 1$. In training, we roll out the cluster loss ℓ_{cl} $L = 5$ times and use $|\mathcal{I}| = 150$ seeds, while we use $L = 10, |\mathcal{I}| = 200$ during test time. δ is set to 0.1. We remove any cluster of pixels that is smaller than 500. Unless stated otherwise, we use $\tau = 15, \sigma = 0.02$. Note that we tried learning σ as an output of the network; however it did not perform well, and thus we opted to keep it fixed.

During DSN training, we augment depth maps with multiplicative gamma noise similar to [10], and add Gaussian Process noise to the backprojected point clouds.

2) RRN: All RRNs are trained with SGD for 100 k iterations with a fixed learning rate of 1e$-$2 and batch size of 16. Inputs to the RRN are padded by 25% of the initial mask's bounding box size in each dimension. Our entire pipeline runs at approximately 3–5 frames per second on a single NVIDIA RTX 2080ti.

3) Baselines: For the baselines in [79], we use the results graciously provided by the authors. We follow the official detectron schedule when training Mask R-CNN [12] on TOD, and train for 100 k iterations using SGD with a batch size of 8. We train PointGroup [13] for 300 k iterations with a batch size of 4 on TOD. We remove clustering of the semantic scores since our problem only has one meaningful semantic class, foreground, which makes no sense to cluster.

Note that the training procedure for 2D DSNs and Mask R-CNN (reported in [15]) differs slightly from the training procedure for 3D DSNs. We replicated these experiments using the same conditions as 3D DSNs (150 k iterations with Adam/SGD), but observed very similar results to [15]. Thus, we report numbers from [15].

B. Datasets

We evaluate quantitatively and qualitatively on two real-world datasets: OCID [79] and OSD [80], which have 2346 images of semi-automatically constructed labels and 111 manually labeled images, respectively. OSD is a small dataset that was manually annotated, so the annotation quality is high. OCID, which is much larger, uses a semi-automatic process of annotating the labels. It incrementally builds up the scene by adding one object at a time, and computes labels by calculating difference in depth.

However, this process is subject to depth sensor noise, so while the majority of the instance label is accurate, the label boundaries are noisy. Additionally, OCID contains images with objects on a tabletop, and images with objects on a floor. Despite our method being trained in synthetic tabletop settings, it generalizes to floor settings as well.

Finally, we use the Google Open Images Dataset v5 [82] (OID) in an experiment to test the sim-to-real gap of the RRN. OID contains approximately 9 million "in-the-wild" images with image-level annotations, bounding boxes, and segmentations. In particular, it contains 2.8 million segmentations for 350 object classes. We filtered the 350 object object classes down to 156 classes that could potentially be on a tabletop, resulting in roughly 220 k instance masks on real RGB images.

C. Metrics

We use the precision/recall/F-measure (P/R/F) metrics as defined in [46]. These metrics promote methods that segment the desired objects and penalize methods that provide false positives. Specifically, the F-measure is computed between all pairs of predicted and ground truth masks, which are matched via the Hungarian method. Given a matching, the final P/R/F is computed by

$$P = \frac{\sum_i |s_i \cap g(s_i)|}{\sum_i |s_i|}, R = \frac{\sum_i |s_i \cap g(s_i)|}{\sum_j |g_j|}, F = \frac{2PR}{P+R}$$

where s_i denotes the set of pixels belonging to predicted object i, $g(s_i)$ is the set of pixels of the matched ground truth object of s_i, and g_j is the set of pixels for ground truth object j. We denote this as Overlap P/R/F. See [46] for more details.

Segmentations with sharp or fuzzy boundaries will obtain similar overlap P/R/F scores, which will not reflect the efficacy of the RRN. To remedy this, we introduce a boundary P/R/F measure to complement the overlap P/R/F. Using the same Hungarian matching used to compute overlap P/R/F, we compute boundary P/R/F by

$$P = \frac{\sum_i |s_i \cap D\left[g(s_i)\right]|}{\sum_i |s_i|}, R = \frac{\sum_i |D\left[s_i\right] \cap g(s_i)|}{\sum_j |g_j|}$$

where we overload notation and denote s_i, g_j to be the set of pixels belonging to the boundaries of predicted object i and ground truth object j, respectively. $D[\cdot]$ denotes the dilation operation, which allows for some slack in the prediction. Note that these metrics are sensitive to the amount of allowed slack. However, without this, these numbers can look deceivingly low as no slack requires exact pixel boundary matching which can lead to issues with either noisy manual annotations or noisy semi-automatic annotations [79]. For the dilation operation, we use a circular kernel where the diameter of the kernel depends on the size of the image. Roughly, this metric is a combination of the $\mathcal{F}$-measure in [83] along with the overlap P/R/F as defined in [46].

We report all P/R/F measures in the range [0100] (P/R/F $\times 100$).

TABLE I
COMPARISON WITH BASELINES ON ARID20 AND YCB10 SUBSETS OF OCID [79].

Method	Overlap			Boundary		
	P	R	F	P	R	F
GCUT [75]	21.5	51.5	25.7	10.2	46.8	15.7
SCUT [76]	45.7	72.5	43.7	43.1	65.1	42.6
LCCP [77]	58.4	89.1	63.8	53.6	82.6	60.2
V4R [78]	65.3	81.4	69.5	62.5	81.4	66.6
UOIS-Net-2D	88.8	81.7	84.1	83.0	67.2	73.3
UOIS-Net-3D	88.2	88.0	87.9	81.1	74.3	77.3

RED indicates the best performance.

D. 2D Quantitative Results

Comparison to Baselines. We compare to baselines shown in [79], which include GCUT [75], SCUT [76], LCCP [77], and V4R [78]. In [79], these methods were only evaluated on the ARID20 and YCB10 subsets of OCID, so we compare our results these subsets as well. These baselines are designed to provide oversegmentations (i.e., they segment the whole scene instead of just the objects of interest). To allow a more fair comparison, we set all predicted masks smaller than 500 pixels to background, and set the largest mask to table label (which is not considered in our metrics). Results are shown in Table I. Because the baselines aim to oversegment the scene, the precision is in general low while the recall is high. LCCP is designed to segment convex objects (most objects in OCID are convex), but its predicted boundaries are noisy due to utilizing depth only. Recall that OCID labels suffer from depth sensor noise, which explains why LCCP's boundary recall is quite high (see Section VIII-E for a visual example). Both SCUT and V4R utilize models trained on real data, putting them at an advantage with respect to UOIS-Net-2D. V4R was trained on OSD [80] which has an extremely similar data distribution to OCID, giving V4R a substantial advantage and making it the strongest baseline in [79]. However, our UOIS-Net-2D (line 5 in Table I), despite never having seen any real data, significantly outperforms these baselines on F-measure.

Comparison to SOTA: In Table II, we compare UOIS-Net-2D to two SOTA methods, Mask R-CNN [12] and PointGroup [13], both trained on RGB-D from TOD. While Mask R-CNN is a general method for detection that can be applied to different types of input modes, the more recent PointGroup requires point clouds and utilizes a sparse convolutional backbone. We see that UOIS-Net-2D (line 6) outperforms Mask R-CNN (line 3) on OSD and slightly on OCID. On the other hand, PointGroup (line 4) provides comparable performance to UOIS-Net-2D. Note, however, that PointGroup reasons in 3D with center voting while UOIS-Net-2D does not. In that sense, Mask R-CNN is more similar to UOIS-Net-2D.

Note that the performance of UOIS-Net-2D is similar to Mask R-CNN and PointGroup on OCID in terms of boundary F-measure. This result is misleading: it turns out that using the RRN to refine the initial masks produced by UOIS-Net results in degraded quantitative performance on OCID, while

TABLE II
EVALUATION OF OUR METHODS AGAINST SOTA METHODS TRAINED ON DIFFERENT INPUT MODES.

Method	Input	OCID [79]						OSD [80]					
		Overlap			Boundary			Overlap			Boundary		
		P	R	F	P	R	F	P	R	F	P	R	F
Mask R-CNN [12]	RGB	66.0	34.0	36.6	58.2	25.8	29.0	63.7	43.1	46.0	46.7	26.3	29.5
Mask R-CNN [12]	Depth	82.7	78.9	79.9	79.4	67.7	71.9	73.8	72.9	72.2	49.6	40.3	43.1
Mask R-CNN [12]	RGB-D	79.2	78.6	78.0	73.6	67.2	69.2	74.0	74.6	74.1	57.3	52.1	53.8
PointGroup [13]	RGB-D	81.6	80.1	80.1	75.4	70.4	71.7	79.5	78.2	78.8	67.0	65.0	65.4
UOIS-Net-2D	DSN: RGB-D	84.2	57.6	62.2	72.9	44.5	49.9	72.2	63.7	66.1	58.5	43.4	48.4
UOIS-Net-2D	DSN: Depth	88.3	78.9	81.7	82.0	65.9	71.4	80.7	80.5	79.9	66.0	67.1	65.6
UOIS-Net-3D	DSN: Depth	86.5	86.6	86.4	80.0	73.4	76.2	85.7	82.5	83.3	75.7	68.9	71.2

Red indicates the best performance.

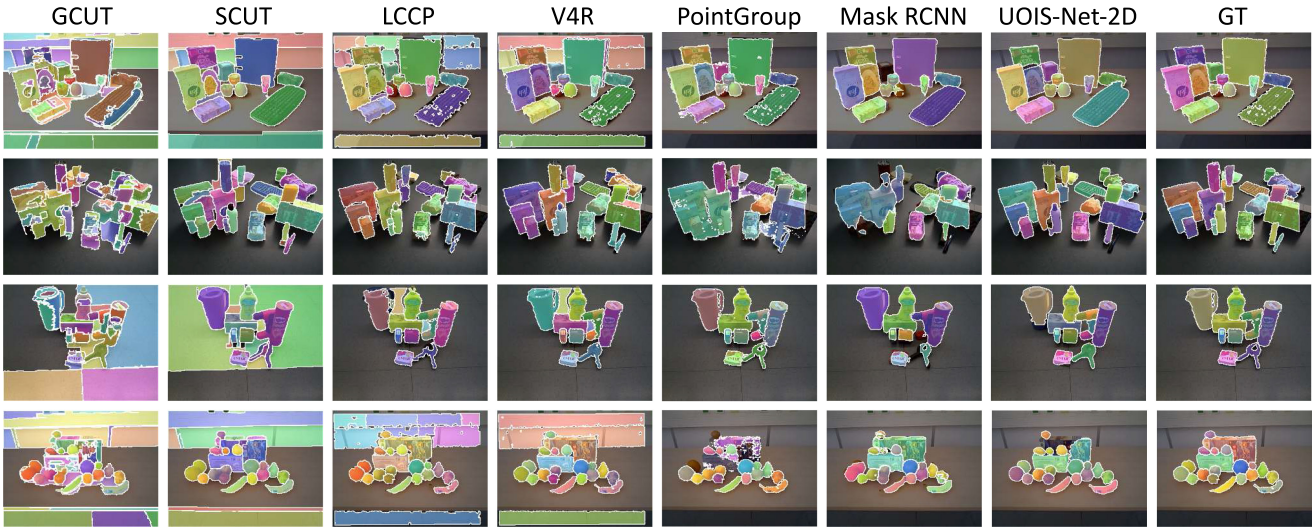


Fig. 4. Comparison of UOIS-Net-2D with baselines, Mask R-CNN, and PointGroup on OCID [79]. LCCP and V4R operate on depth only, thus are subject to noise from depth sensors. However, this is also true for the ground truth segmentation produced by OCID [79]. Our proposed UOIS-Net-2D provides sharp and accurate masks in comparison to all of the baselines.

TABLE III
PERFORMANCE OF UOIS-NET WITHOUT RRN

Method	OCID [79]						OSD [80]					
	Overlap			Boundary			Overlap			Boundary		
	P	R	F	P	R	F	P	R	F	P	R	F
UOIS-Net-2D	90.9	79.4	83.0	85.2	70.8	75.7	80.3	81.1	80.1	53.4	53.3	52.8
UOIS-Net-3D	88.8	89.2	88.8	86.9	80.9	83.5	85.7	82.1	83.0	74.3	67.5	70.0

the qualitative results are better. This is due to OCID's noisy label boundaries. An illustration of this can be found in the first example (row) of Fig. 4. Table III shows the performance of UOIS-Net without applying RRN (DSN and IMP only) on OCID and OSD. This method utilizes only depth and predicts segmentation boundaries that are aligned with the sensor noise. In this setting, UOIS-Net-2D outperforms both SOTA methods, with a significant gain in boundary F-measure and a minor gain in overlap F-measure. In fact, UOIS-Net-3D also gains performance on OCID when removing the RRN which further demonstrates the issue with OCID labels. On the other hand, OSD has manually annotated labels and this issue is not present, and our methods' performances deteriorate in the absence of the RRN. In particular, UOIS-Net-2D drops almost 20% relatively without it.

Effect of Input Mode: To evaluate how different input modes affect results, we train Mask R-CNN on RGB, depth, and RGB-D and compare it to UOIS-Net-2D in Table II. Training Mask R-CNN on synthetic RGB only poorly generalizes from sim-to-real. Training on depth drastically boosts this generalization, which is in agreement with [10], [11], [64]. When training on RGB-D, we posit that Mask R-CNN relies heavily on depth as adding RGB to depth results in little change. However, UOIS-Net-2D exploits RGB and depth separately, leading to better

TABLE IV
COMPARISON OF RRN WHEN TRAINING ON TOD AND REAL IMAGES FROM GOOGLE OID [82]

DSN reasoning	RRN training data	OCID [79]						OSD [80]					
		Overlap			Boundary			Overlap			Boundary		
		P	R	F	P	R	F	P	R	F	P	R	F
2D	TOD	88.3	78.9	81.7	82.0	65.9	71.4	80.7	80.5	79.9	66.0	67.1	65.6
	OID	87.9	79.6	81.7	84.0	69.1	74.1	81.2	83.3	81.7	69.8	73.7	70.8
3D	TOD	86.5	86.6	86.4	80.0	73.4	76.2	85.7	82.5	83.3	75.7	68.9	71.2
	OID	86.0	88.2	86.9	83.1	77.6	79.9	86.0	85.1	84.8	81.0	75.3	77.3

TABLE V
(LEFT) ABLATION EXPERIMENTS FOR UOIS-NET-2D ON OSD [80], AND (RIGHT) REFINING MASK R-CNN RESULTS WITH RRN (TRAINED ON TOD) ON OSD.

DSN	IMP		RRN	Boundary		
	O/C	CCC		P	R	F
✓				35.0	58.5	43.4
✓			✓	36.0	48.1	39.6
✓	✓			49.2	55.3	51.7
✓	✓		✓	59.0	64.1	60.7
✓	✓	✓		53.4	53.3	52.8
✓	✓	✓	✓	66.0	67.1	65.6

Method	Input	RRN	Boundary		
			P	R	F
Mask R-CNN	RGB		46.7	26.3	29.5
Mask R-CNN	RGB	✓	52.8	28.8	33.3
Mask R-CNN	Depth		49.6	40.3	43.1
Mask R-CNN	Depth	✓	69.0	55.2	59.8
Mask R-CNN	RGB-D		57.3	52.1	53.8
Mask R-CNN	RGB-D	✓	63.4	57.0	59.2

O/C denotes the open/close morphological transform, while CCC denotes the closest connected component.

results on OSD while being trained on the exact same synthetic dataset. Furthermore, when our DSN is trained directly on RGB-D (line 4, Table II), we see a drop in performance, suggesting that training directly on (nonphotorealistic) RGB is not the best way of utilizing synthetic data.

Degradation of Training on Nonphotorealistic Simulated RGB: To quantify how much nonphotorealistic RGB degrades performance, we train an RRN on real data. This serves as an approximate upper bound on how well the synthetically trained RRN can perform. We use the instance masks from OID [82] and show results in Table IV. Both models share the same DSN and IMP. The overlap measures are roughly the same, while the RRN trained on OID has slightly better performance on the boundary measures. This suggests that while there is still a gap, our method is surprisingly not too far off, considering that we train with nonphotorealistic synthetic RGB. We conclude that mask refinement with RGB is an easier task to transfer from sim-to-real than directly segmenting from RGB.

Ablation Studies: We report ablation studies on OSD to evaluate each component of our proposed method in Table V (left). We omit the overlap P/R/F results since they follow similar trends to boundary P/R/F. Running the RRN on the raw masks output by DSN without the IMP module actually hurts performance as the RRN cannot refine such noisy masks. Adding the open/close morphological transform and/or the closest connect component results in much stronger results, showing that the IMP is crucial in robustifying our proposed method. In these settings, applying the RRN significantly boosts boundary P/R/F. In fact, Table V (right) shows that applying the RRN to the Mask R-CNN results effectively boosts the boundary P/R/F for all input modes on OSD, showing the efficacy of the RRN. Note that even with this refinement, the Mask R-CNN results are outperformed by our method.

E. 2D Qualitative Results

Comparison to Baselines and SOTA: First, we show qualitative results on OCID of baseline methods, Mask R-CNN (trained on RGB-D), PointGroup, and UOIS-Net-2D in Fig. 4. It is clear that the baseline methods suffer from oversegmentation issues; they segment the table and background into multiple pieces. This is especially the case for methods that utilize RGB as an input (GCUT and SCUT); the objects are often oversegmented as well. In example (row) 1 of Fig. 4, the ground truth label for the keyboard is riddled with holes. Methods that operate only on depth (LCCP and V4R) mirror these noisy boundaries, leading to inflated boundary P/R/F measures. Despite this, our quantitative results still outperform these baselines.

The main failure mode for Mask R-CNN is that it tends to undersegment objects. This is the typical failure mode of top-down instance segmentation algorithms in clutter. A close inspection of Fig. 4 shows that Mask R-CNN frequently segments multiple objects as one. Examples 1 and 4 show undersegmentation of many neighboring small objects, and example 2 shows undersegmentation of larger objects. Since PointGroup requires point clouds, it is susceptible to degradation from depth sensor noise as well. Additionally, some masks (e.g., example 4) can be quite noisy, which we hypothesize is due to the rudimentary breadth-first search clustering algorithm.

On the other hand, our method utilizes depth and RGB separately to provide sharp and accurate masks. Because UOIS-Net-2D leverages RGB after depth, it can fix the issues with depth sensors shown in example 1. Additionally, our method can segment complicated structures such as stacks (examples 2 and 4) and cluttered environments (example 4).

RRN Refinements: In Fig. 5, we qualitatively show the effect of the RRN. The top row shows the masks before refinement (after IMP), and the bottom row shows the refined masks. These

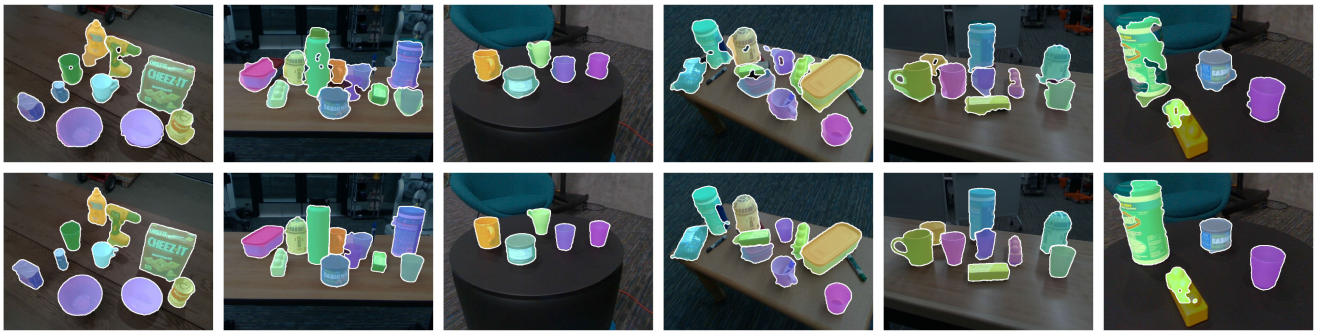


Fig. 5. Mask refinements with RRN: before (top) refinement (after IMP), and after refinement (bottom).

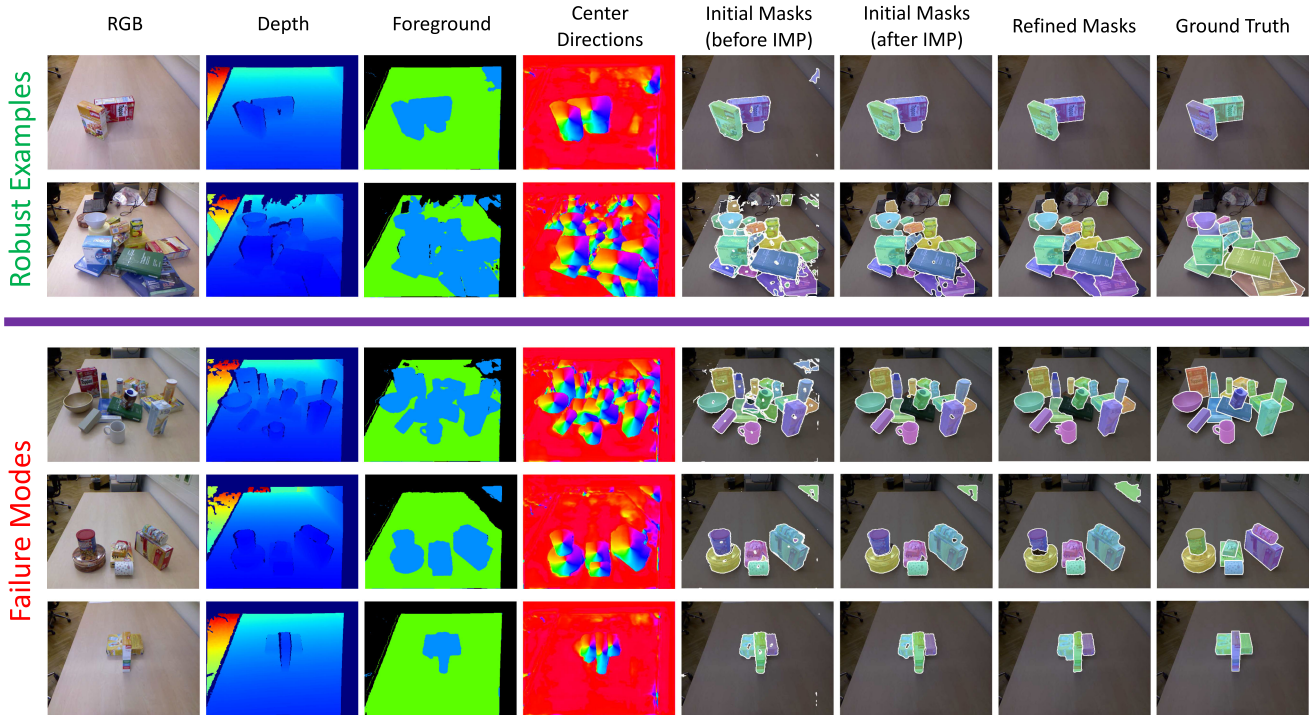


Fig. 6. Outputs (and intermediate) of UOIS-Net-2D are visualized. We demonstrate the robustness of the Hough voting layer and the IMP (top) and show common failure modes (bottom). See text for details. Best viewed in color on a computer screen.

images were taken in our lab with an Intel RealSense D435 RGB-D camera to demonstrate the robustness of our method to camera viewpoint variations and distracting backgrounds, as OCID and OSD have relatively simple backgrounds. Due to noise in the depth sensor, it is impossible to get sharp and accurate predictions from depth alone without using RGB. Our RRN can provide sharp masks even when the boundaries of objects are occluding other objects (examples 2 and 5). Our RRN is able to fix squiggly mask boundaries and patch up large missing chunks in the initial mask.

Robustness: In Fig. 6 (top), we demonstrate how our method is robust to errors in the pipeline. In row 1, the DSN produces a false positive foreground region in the top right of the image. However, there is not enough evidence in the center directions (not enough discretized angles are present), and thus the Hough voting layer suppresses this potential object. In row 2, there are many spurious center direction predictions; however, these are not considered as potential object centers by the Hough voting layer because they are not detected by the foreground mask. Additionally, one can see the many holes and spurious mask components in column 5. As seen in the ablation studies in Section VIII-D, applying the RRN here actually degrades the performance. The IMP cleans this image, making the refinement task easier for the RRN.

Failure Modes: We show some failure modes of our DSN in Fig. 6 (bottom) on the OSD dataset. In row 1, we see an undetected object (green book) because the center of the mask is occluded. In the center direction image, there is no convergent point for this object which would be considered as the object center with all discretized angles pointing to it. In row 2, there is a false positive region detected by the DSN in the top right corner, which cannot be undone by the RRN. Finally, in row 3, the DSN cannot correctly segment an object whose mask has been split into two by an occluding object.

TABLE VI
ABLATION STUDY OVER LOSS FUNCTIONS OF UOIS-NET-3D.

Loss Functions				OCID [79]						OSD [80]					
				Overlap			Boundary			Overlap			Boundary		
ℓ_{fg}	ℓ_{co}	ℓ_{cl}	ℓ_{sep}	P	R	F	P	R	F	P	R	F	P	R	F
✓	✓			77.9	79.8	78.6	71.2	68.0	68.9	79.4	80.1	79.8	74.5	66.5	69.8
✓	✓	✓		77.0	81.3	78.7	71.9	68.2	69.0	72.8	74.9	72.8	67.2	61.0	62.2
✓	✓		✓	85.5	87.9	86.5	80.9	77.8	78.8	84.0	85.1	84.5	75.0	74.9	74.6
✓	✓	✓	✓	86.0	88.2	86.9	83.1	77.6	79.9	86.0	85.1	84.8	81.0	75.3	77.3

Our novel separation loss ℓ_{sep} is crucial to obtaining state-of-the-art performance. Note that we are using an RRN trained on real data.

Our RRN also has some common failure modes. Column 4 in Fig. 5 shows that the RRN can fail when the object has a complex texture. Column 6 demonstrates that if there is not enough padding to the initial mask to see the entire object, the RRN cannot segment the entire object (lego block). In column 2, the RRN has a tough time fixing the segmentation mask when the DSN has incorrectly undersegmented a few objects together (the two cups to the left of the eraser).

F. 3D Quantitative Results

Comparison to Baselines: We compare the full UOIS-Net-3D with an RRN trained on TOD to all previous baselines from [79] and our UOIS-Net-2D in Table I. Discussion of all baselines and UOIS-Net-2D is given in Section VIII-D. Compared to UOIS-Net-2D, the 3D version substantially increases the recall in both the overlap and boundary metrics, leading to a relative increase of 4.5% in overlap F-measure and 5.5% in boundary F-measures. In fact, UOIS-Net-3D, despite no convexity assumptions, gets quite close to the overlap recall of LCCP, which segments objects based on convexity. Note that only the DSN structure is changed when moving from 2D to 3D, and thus this extra recall is mainly due to the 3D center voting structure and better initial masks.

Comparison to SOTA: In Table II, we show comparisons of UOIS-Net-3D with SOTA methods Mask R-CNN [12] and PointGroup [13], and the previous UOIS-Net-2D on OCID and OSD. Again, the trend of performance increase from UOIS-Net-2D is similar to before, where a significant increase in recall leads to a boost in F-measure. On OCID, UOIS-Net-3D receives a relative increase of 5.8% in overlap F-measure and a 6.7% in boundary F-measure. Compared to the best performing Mask R-CNN, our performance provides a relative increase of 8.1% on overlap F-measure and 6.0% on boundary F-measure, while being trained on the exact same dataset (TOD). Additionally, we outperform PointGroup by 7.9% and 6.3% on overlap and boundary F-measures, respectively. Note that UOIS-Net-3D performs center voting in a similar fashion to PointGroup; however, we believe our novel loss functions target cluttered environments more effectively. On OSD overlap F-measure, we provide a relative increase of 4.3% over UOIS-Net-2D, 12.4% over Mask R-CNN, and 5.7% over PointGroup. For boundary F-measure, we show 8.5% over UOIS-Net-2D, 32.3% over Mask R-CNN, and 8.9% over PointGroup. Note that while our recall jumps modestly in comparison to UOIS-Net-2D, the precision increase is more sizable.

RRN Ablation: We refer readers to Table IV to show the full power of our method. When using an RRN trained on OID [82], our full UOIS-Net-3D further increases the boundary F-measures, leading to 79.9 points on OCID and 77.3 points on OSD, which is 7.8% and 9.2% higher than the full UOIS-Net-2D on OCID and OSD, respectively. In the rest of this section, all UOIS-Net-3D's will use an RRN trained on OID in order to show the full performance of our method.

Loss Function Ablation: At the minimum, we require ℓ_{fg} and ℓ_{co} to train UOIS-Net-3D. In Table VI, UOIS-Net-3D actually performs poorer than UOIS-Net-2D (see Table IV) with these two losses only. When adding the cluster loss ℓ_{cl}, we get roughly the same performance on OCID. Our intuition is that this loss encourages the same behavior as ℓ_{co}, namely that the points should cluster near the ground truth object center, and thus it does not introduce anything new. However, adding this loss significantly hurts performance on OSD; we posit that this result is noisy due to the small size of OSD. In the third row, we show that adding our novel ℓ_{sep} loss relatively boosts overlap F-measure by 10.1% and boundary F-measure by 14.4% compared to the base model ($\lambda_{fg}\ell_{fg} + \lambda_{co}\ell_{co}$), demonstrating it is crucial for obtaining good performance. ℓ_{sep} pushes center votes away from center votes of other objects in order to make the postprocessing GMS clustering easier. Note that the center votes do not necessarily need to be close to the ground truth object centers; they simply need to be easily clustered, and this loss allows for that. This property is crucial to getting strong performance in clutter as the numbers demonstrate. We will also visually illustrate this phenomena in Section VIII-G. Finally, since the object clusters are not necessarily near the ground truth object centers, ℓ_{cl} now encourages something that ℓ_{co} does not: it encourages that the center votes are tightly packed wherever they are placed, further easing the job of the postprocessing GMS clustering. In line 4, we see it provides some extra performance gain.

ESP Module Ablation: We evaluate the significance of using the ESP module [70] to obtain a higher receptive field by testing both UOIS-Net-3D and UOIS-Net-2D (which normally does not include the ESP module) with and without the ESP modules embedded into the DSN in Table VII. We immediately see that obtaining a higher receptive field is beneficial to the performance. For UOIS-Net-3D, we see relative gains on OCID of 3.0% and 3.5% on overlap and boundary F-measures, respectively. On OSD, we see a similar performance. For UOIS-Net-2D, we see a large relative gain on OCID of 4.9% and 5% on overlap

TABLE VII
ABLATION STUDY TO TEST THE SIGNIFICANCE OF A WIDER RECEPTIVE FIELD.

DSN reasoning	ESP Module	OCID [79]						OSD [80]					
		Overlap			Boundary			Overlap			Boundary		
		P	R	F	P	R	F	P	R	F	P	R	F
2D	✗	87.9	79.6	81.7	84.0	69.1	74.1	81.2	83.3	81.7	69.8	73.7	70.8
	✓	87.6	85.0	85.7	84.4	73.4	77.8	84.6	82.3	82.8	75.5	71.6	72.4
3D	✗	83.9	85.6	84.4	79.1	76.4	77.2	85.1	85.7	85.1	78.9	76.5	77.3
	✓	86.0	88.2	86.9	83.1	77.6	79.9	86.0	85.1	84.8	81.0	75.3	77.3

Using the ESP module [70] in the DSN architecture improves performance for both UOIS-Net-2D and UOIS-Net-3D. Note that we are using an RRN trained on real data.

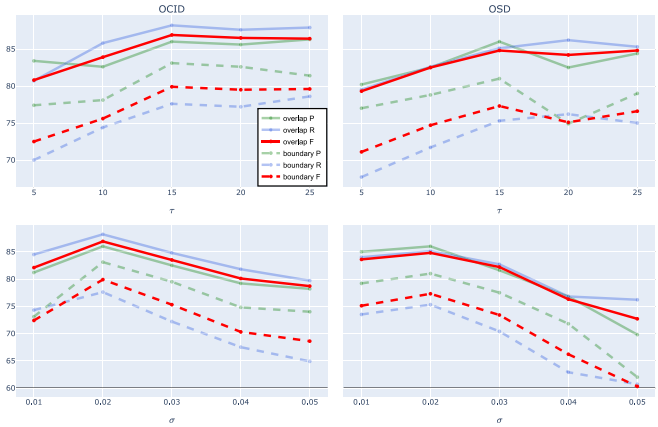


Fig. 7. Ablation study to test the sensitivity of UOIS-Net-3D with respect to τ, σ. Best viewed by zooming in on a computer.

and boundary F-measures, respectively. Note that the DSN with ESP modules has slightly fewer parameters than without them, and thus these experiments highlight the usefulness of a higher receptive field.

τ Ablation: In the top row of Fig. 7, we test the sensitivity of our model to the settings of τ over $[5, 10, 15, 20, 25]$, which is used in ℓ_{sep}. We visualize both overlap and boundary P/R/F measures. Note that as τ is essentially a multiplicative factor on the Euclidean distance in (4); thus, as τ becomes larger, the Euclidean distance becomes more inflated and objects do not need to be pushed as far in order to minimize ℓ_{sep}. It is clear to see Fig. 7 that all metrics increase as τ increases, and they plateau after τ hits a high enough value, in this case 15. This suggests that we only need a small level of encouragement to push the object centers away from each other in order to get strong performance, and that setting τ too low can degrade performance, potentially by making ℓ_{sep} too difficult to minimize.

σ Ablation: In the bottom row of Fig. 7, we test the sensitivity of UOIS-Net-3D to σ over $[0.01, 0.02, 0.03, 0.04, 0.05]$, which is used in ℓ_{cl}. The value of σ determines how tightly packed the center votes of an object need to be in order to minimize ℓ_{cl}. Additionally, it determines how much spread the center votes can have and still be clustered together during the post-processing GMS clustering step. When σ is too small, the DSN typically oversegments the objects. On the other hand, when σ is too large, center votes from multiple objects get clustered together and undersegmentation occurs. Thus, σ should be chosen to reflect the level of clutter in the scenes, i.e., how close the objects are. Our ablation study hints at this idea: the bottom row of Fig. 7 that $\sigma = 0.02$ works the best for both OCID [79] and OSD [80], as they are similar in distribution. In fact, the trends of the graph in both datasets are also similar. Note that we tried learning σ as a parameter of the network; however, these experiments did not work well.

G. 3D Qualitative Results

2D vs. 3D Comparison: In Fig. 8, we qualitatively compare the predictions of UOIS-Net-2D and UOIS-Net-3D to understand how reasoning in 3D can solve the 2D issues mentioned beforehand (in Section VIII-E). The first row of Fig. 6 (bottom) exhibits the 2D issue of false negative detection when the 2D center of the object is occluded. However, this is easily corrected when voting for 3D centers, as seen in columns 2 and 5 of Fig. 8 In columns 1, 3, and 4, we see that UOIS-Net-3D detects and segments more small, thin objects such as pens and bananas. The 2D Hough voting procedure typically fails to detect enough discretized directions for objects with such shape. This issue is ameliorated when reasoning in 3D, as there are no approximations made with discretized directions. Finally, in columns 3, 4, and 6, we show that the 3D DSN architecture performs better due to a higher receptive field provided by the ESP modules.

Center Votes: As shown in Table VI, ℓ_{sep} is crucial to ensuring UOIS-Net-3D has strong performance. We visually examine the reason for this in Fig. 9. In the first and second columns, we visualize the point cloud of the scene with the predicted center votes overlaid on the same image. Both the point cloud and center votes are color coded with respect to their instance segmentation masks. It is clear to see that when training without ℓ_{sep}, the center votes are more spread out and messy, which makes the postprocessing GMS clustering more difficult. On the other hand, the full model has much more tightly packed and separated center votes, which leads to better performance. Looking at example (row) 5, there are many small objects such as fruits and bananas which are close to each other in spatial proximity. UOIS-Net-3D trained without ℓ_{sep} shows that the center votes are mostly jumbled together, while the full model nicely separates the center votes which leads to almost perfect segmentation. Note that the center votes do not need to be at the center of the object, as long as they can be clustered correctly. In the upper right of example 2, the cylinder and the bowl have very close 3D centers. The 3D DSN predicts the center votes in a

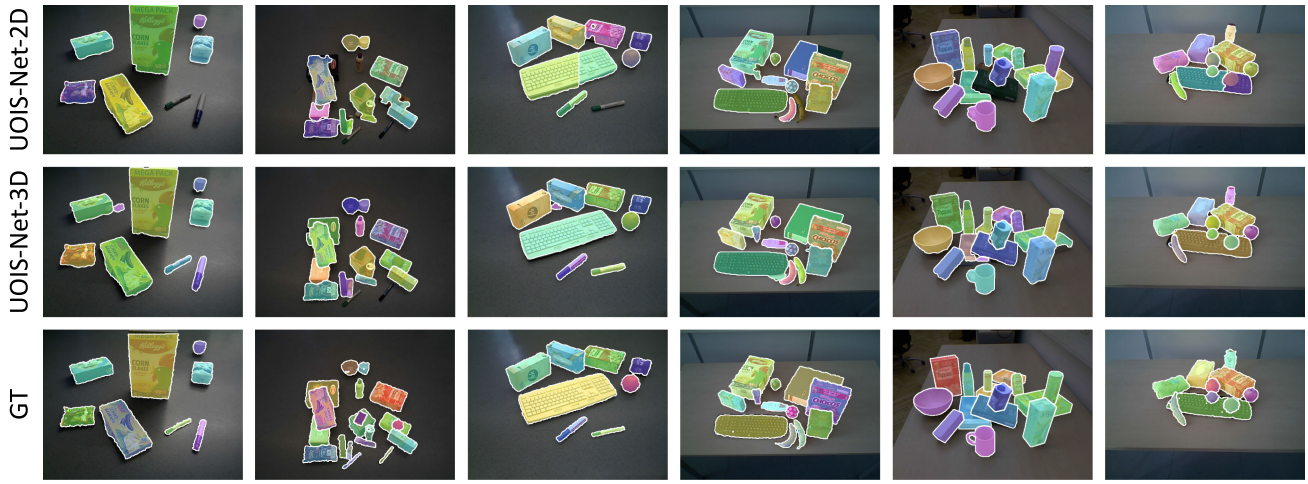


Fig. 8. Qualitative comparison of UOIS-Net-2D to UOIS-Net-3D on OCID.

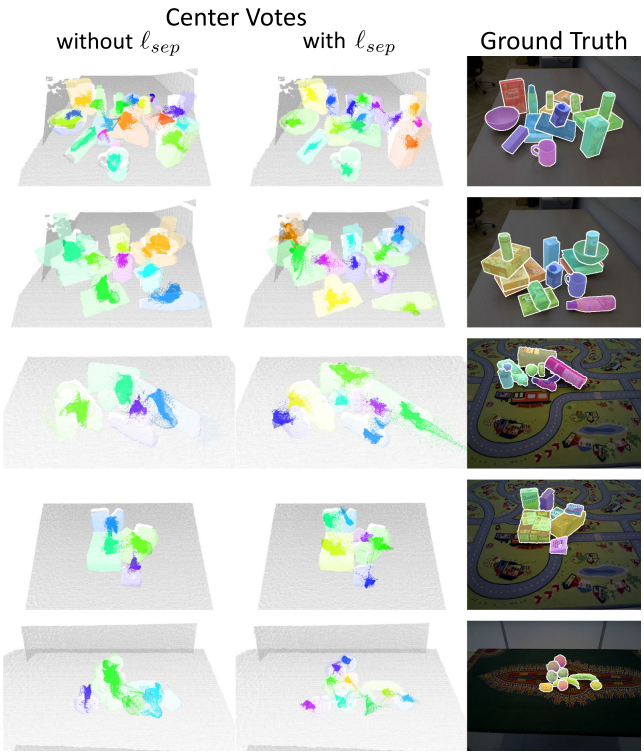


Fig. 9. Effect of ℓ_{sep} on center votes. Columns 1 and 2 show the point cloud (visualized with Open3D [84]) and center votes overlaid on the image, which are color-coded according to their instance ID. Best viewed in color and zoomed in.

way that pushes them apart to ease the postprocessing clustering and allows (the full) UOIS-Net-3D to correctly segment these objects.

Failure Modes: In Fig. 10, we demonstrate common failure modes of UOIS-Net-3D. Row 1 shows that when objects are close together, they may undersegmented into a single segment. The center votes show that the three small fruits have center votes that are too close to separate in the post-processing clustering step. Another case of undersegmentation is shown in row 3, where multiple objects (cereal boxes) with flat surfaces are lined up such that the depth map shows a large flat surface. It is difficult to correctly separate these boxes from depth alone; appearance is key in providing the correct segmentation in such situations. Row 4 shows that highly nonconvex objects such as power drills can be oversegmented into pieces. The center votes image clearly shows that the DSN believes this object to be two objects. Finally, row 2 indicates that the 2D issue of attempting to segment a mask that is split into multiple pieces by an occluding object is still present when reasoning in 3D.

TABLE VIII
PERFORMANCE ON TOD TEST SET (20 K IMAGES)

Method	Overlap			Boundary		
	P	R	F	P	R	F
UOIS-Net-2D	94.3	89.0	90.9	89.7	80.6	84.1
UOIS-Net-3D	92.1	92.1	91.9	85.0	87.1	85.5
Mask R-CNN	95.4	95.2	95.1	92.3	90.0	90.9
PointGroup	97.7	95.1	96.1	93.3	89.8	91.1

H. Quantifying Generalization From Sim to Real

In order to quantify generalization from simulation to the real world, we show performance of our methods and SOTA methods Mask R-CNN and PointGroup on the TOD test set. This dataset of 20 k images was generated using instances that are not present in the training set. In Table VIII, we show that both Mask R-CNN and PointGroup outperform UOIS-Net on all metrics on this test set. However, as seen in Table II, UOIS-Net outperforms Mask R-CNN and PointGroup on real-world data, indicating that our methods handle the distribution shift better, which is ultimately what we care about.

I. Application in Grasping Unknown Objects

We use our model to demonstrate manipulation of unknown objects in a cluttered environment using a Franka robot with panda gripper and wrist-mounted RGB-D camera. The task is to collect objects from a table and put them in a bin. Object instances are segmented using our method and the point cloud of the closest object to the camera is fed to 6-DOF GraspNet [1],

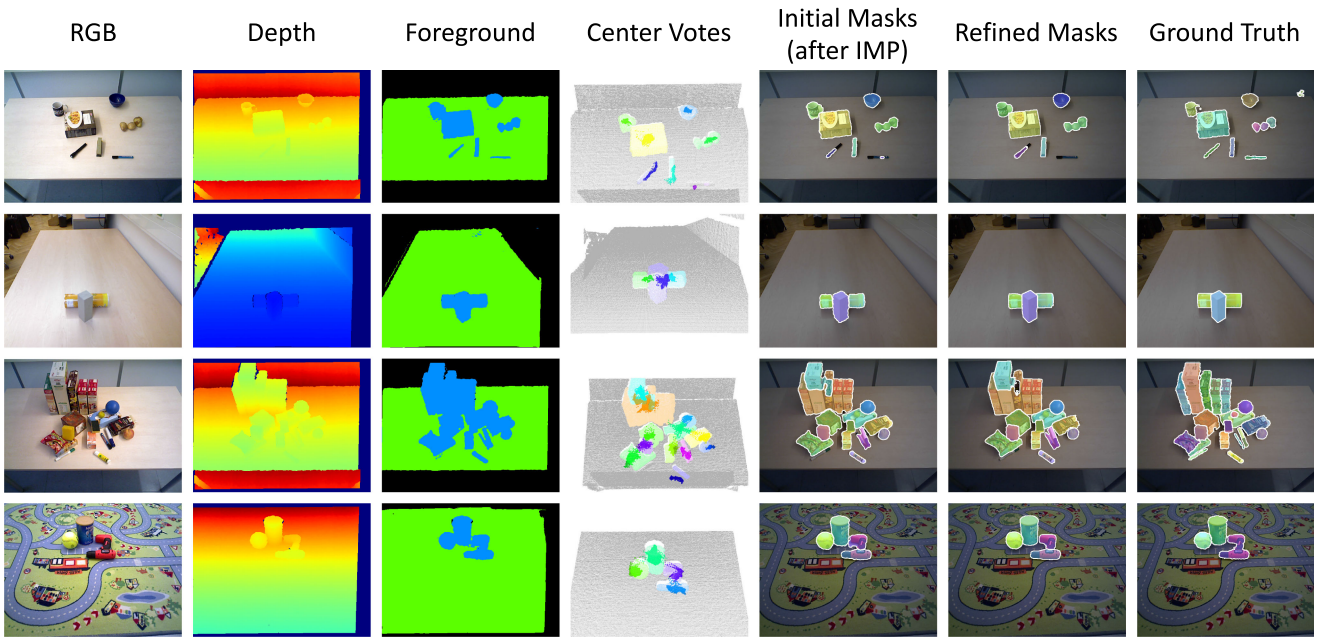


Fig. 10. Common failure modes of UOIS-Net-3D. See text for details. Best viewed in color and zoomed in.

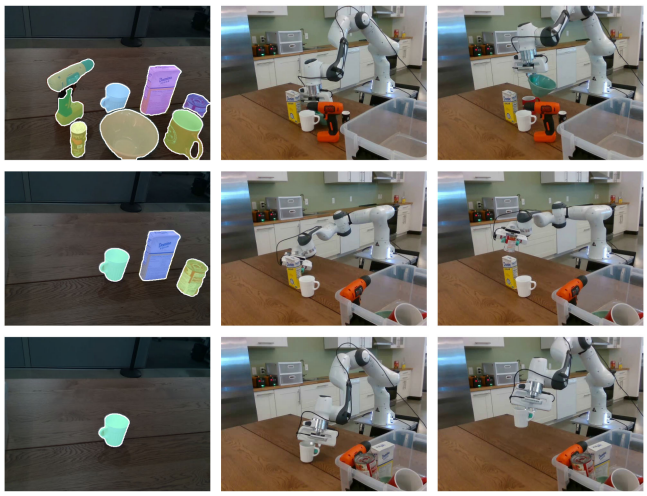


Fig. 11. Visualization of clearing table using our instance segmentation and 6-DOF GraspNet [1].

[2] to generate diverse grasps, with other objects considered obstacles. Other objects are represented as obstacles by sampling fixed number of points using farthest point sampling from their corresponding point cloud. The grasp that has the maximum score and has a feasible path is chosen to execute. Fig. 11 shows the instance segmentation at different stages of the task and also the execution of the robot grasps. Video of the experiments can be found at the project website.[2] Our method segments the objects correctly most of the time but fails sometimes, such as the oversegmentation of the drill in the scene. Our method considers the top of the drill as one object and the handle as an obstacle. This is because the object is highly nonconvex as discussed in the previous section. We conducted the experiment to collect 51 objects from nine different scenes. Each object is considered to be successfully grasped if the robot can pick it up in maximum two attempts. Otherwise, we count that object as failure case and remove it from the scene manually and proceed to other objects. In our trials, the robot successfully grasped 41/51 objects (80.3% success rate). The failures stem from either imperfections in segmentation or inaccurate generated grasps. Note that neither our method nor 6-DOF GraspNet [1] is trained on real data.

[2][Online]. Available: https://rse-lab.cs.washington.edu/projects/unseen-object-instance-segmentation/.

IX. Conclusion

We proposed a deep network, UOIS-Net, that separately leverages RGB and depth to provide sharp and accurate masks for unseen object instance segmentation. Our two-stage framework produces rough initial masks using only depth by regressing center votes in either 2D or 3D, then refines those masks with RGB. Surprisingly, our RRN can be trained on nonphotorealistic RGB and generalize quite well to real-world images. We demonstrated the efficacy of our approach on multiple tabletop environment datasets and showed that our model can provide strong results for unseen object instance segmentation. Finally, we also addressed the weaknesses of our method which will serve as the base of future work.

References

[1] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2901–2910.

[2] A. M. A. Murali, C. Eppner, C. Paxton, and D. Fox, "6-DOF grasping for target-driven object manipulation in clutter," in *Proc. IEEE Conf. Robot. Automat.*, 2020, pp. 6232–6238.

[3] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris, "Task-driven perception and manipulation for constrained placement of unknown objects," *IEEE Robot. and Automat. Lett.*, vol. 5, no. 4, pp. 5605–5612, Oct. 2020.

[4] A. X. Chang *et al.*, "ShapeNet: An information-rich 3D model repository," Stanford Univ.—Princeton University.—Toyota Technol. Inst. at Chicago.

[5] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," in *Proc. Australas. Conf. Robot. Automat.*, 2015, pp. 1–8.

[6] T. Hodaň *et al.*, "Photorealistic image synthesis for object instance detection," *in Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 66–70.

[7] E. Tzeng *et al.*, "Adapting deep visuomotor representations with weak pairwise constraints," in *Proc. Int. Workshop Algorithmic Foundat. Robot.*, 2016, pp. 688–703.

[8] K. Bousmalis *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4243–4250.

[9] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 23–30.

[10] J. Mahler *et al.*, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Conf. Robot. Sci. Syst.*, 2017.

[11] M. Danielczuk *et al.*, "Segmenting unknown 3D objects from real depth images using mask R-CNN trained on synthetic data," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 7283–7290.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[13] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3D instance segmentation," in *Proc. IEEE Conf. Comput. Visi. Pattern Recognit.*, 2020, pp. 4867–4876.

[14] E. Coumans and Y. Bai, "Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning," 2016–2019 [Online]. Available: http://pybullet.org

[15] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "The best of both modes: Separately leveraging RGB and depth for unseen object instance segmentation," in *Proc. Conf. Robot Learn.*, 2019, pp. 1369–1378.

[16] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFS," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder–decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.

[20] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5168–5177.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2999–3007.

[22] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[23] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Visi. Pattern Recognit.*, 2018, pp. 9224–9232.

[24] C. Choy, J. Gwak, and S. Savarese, "4D Spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3070–3079.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.

[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.* (NeurIPS), 2017, pp. 5105–5114.

[27] P. K. Nathan Silberman, D. Hoiem, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.

[28] X. Ren, L. Bo, and D. Fox, "RGB-(D) scene labeling: Features and algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2759–2766.

[29] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *Proc. IEEE Conf. Robot. Automat.*, 2014, pp. 3050–3057.

[30] Y. Xiang and D. Fox, "DA-RNN: Semantic mapping with data associated recurrent neural networks," in *Robot. Sci. Syst.*, 2017.

[31] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 345–360.

[32] W. Wang and U. Neumann, "Depth-aware CNN for RGB-D segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 135–150.

[33] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3D graph neural networks for RGBD semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5209–5218.

[34] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4438–4446.

[35] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "Masklab: Instance segmentation by refining object detection with semantic and direction features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4013–4022.

[36] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 9796–9805.

[37] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," 2017, *arXiv:1708.02551*.

[38] D. Neven, B. De Brabandere, M. Proesmans, and L. Van Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8829–8837.

[39] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi, "Semi-convolutional operators for instance segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 86–102.

[40] L. Shao, Y. Tian, and J. Bohg, "ClusterNet: 3D instance segmentation in RGB-D images," 2018, *arXiv:1807.08894*.

[41] X. Chen, R. Girshick, K. He, and P. Dollár, "Tensormask: A foundation for dense object segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2061–2069.

[42] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "Blendmask: Top-down meets bottom-up for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8573–8581.

[43] S. Peng, W. Jiang, H. Pi, X. Li, H. Bao, and X. Zhou, "Deep snake for real-time instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8530–8539.

[44] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool, "Deep extreme cut: From extreme points to object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp.616–625.

[45] C. Xie, Y. Xiang, Z. Harchaoui, and D. Fox, "Object discovery in videos as foreground motion clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9986–9995.

[46] A. Dave, P. Tokmakov, and D. Ramanan, "Towards segmenting everything that moves," in *Proc. IEEE/CVF Int. Conf. Comput. Vision (ICCV) Workshops*, 2019.

[47] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *Proc. Adv. Neural Inf. Process. Syst.* (NeurIPS), 2015, pp. 1990–1998.

[48] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 75–91.

[49] W. Kuo, A. Angelova, J. Malik, and T.-Y. Lin, "Shapemask: Learning to segment novel objects by refining shape priors," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9207–9216.

[50] L. Shao, P. Shah, V. Dwaracherla, and J. Bohg, "Motion-based object segmentation based on dense RGB-D scene flow," in *Proc. IEEE Robot. Automat. Lett. Sel. IROS'18 Program Committee Presentation Conf.*, vol. 3, no. 4, pp. 3797–3804, 2018.

[51] A. Byravan and D. Fox, "SE3-Nets: Learning rigid body motion using deep neural networks," in *Proc. IEEE Conf. Robot. Automat.*, 2017, pp. 173–180.

[52] A. Byravan, F. Leeb, F. Meier, and D. Fox, "SE3-Pose-Nets: Structured deep dynamics models for visuomotor planning and control," in *Proc. IEEE Conf. Robot. Automat.*, 2018, pp. 3339–3346.

[53] J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D semantic instance segmentation of RGB-D scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4421–4430.

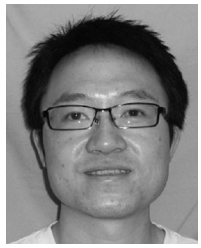
[54] L. Han, T. Zheng, L. Xu, and L. Fang, "OccuSeg: Occupancy-aware 3D instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2937–2946.

[55] J. Lahoud, B. Ghanem, M. Pollefeys, and M. R. Oswald, "3D instance segmentation via multi-task metric learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9256–9266.

[56] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9276–9285.
[57] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "ImVoteNet: Boosting 3D object detection in point clouds with image votes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4404–4413.
[58] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3D-MPA: Multi-proposal aggregation for 3D semantic instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9031–9040.
[59] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *Robot.: Sci. Syst.*, 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p19.html
[60] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proc. Conf. Robot Learn.*, 2018, pp. 306–316.
[61] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 683–698.
[62] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *Robot.: Sci. Syst.*, 2018.
[63] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," *Robot. Sci. Syst.*, 2017. [Online]. Available: http://www.roboticsproceedings.org/rss13/p34.html
[64] D. Seita *et al.*, "Deep transfer learning of pick points on fabric for robot bed-making," in *Proc. Int. Symp. Robot. Res. (ISRR)*, 2019.
[65] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[66] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Interv.*, 2015, pp 234–241.
[67] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.
[68] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1395–1403.
[69] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. Int. Conf. Learn. Representat.*,, 2016, pp. 240–245.
[70] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 552–568.
[71] M. A. Carreira-Perpinán, "A review of mean-shift algorithms for clustering," 2015, *arXiv:1503.00687*.
[72] S. Kong and C. Fowlkes, "Recurrent pixel embedding for instance grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9018–9028.
[73] J. Serra, *Image Analysis and Mathematical Morphology*. New York, NY, USA: Academic Press, Inc., 1983.
[74] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 190–198.
[75] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.
[76] T. T. Pham, T.-T. Do, N. Sünderhauf, and I. Reid, "SceneCut: Joint geometric and object segmentation for indoor scenes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3213–3220.
[77] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter, "Object partitioning using local convexity," in *Proc. IEEE Conf. Comput, Vision Pattern Recognit.*, 2014, pp. 304–311.
[78] E. Potapova, A. Richtsfeld, M. Zillich, and M. Vincze, "Incremental attention-driven object segmentation," in *Proc. IEEE-RAS Int. Conf. Humanoid Robot.*, 2014, pp. 252–258.
[79] M. Suchi, T. Patten, and M. Vincze, "EasyLabel: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets," in *Proc. IEEE Conf. Robot. Automat.*, 2019, pp. 6678–6684.
[80] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2012, pp. 4791–4796.
[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representat.*, 2015. [Online]. Available: https://openreview.net/forum?id=8gmWwjFyLj
[82] R. Benenson, S. Popov, and V. Ferrari, "Large-scale interactive object segmentation with human annotators," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11692–11701.
[83] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 724–732.
[84] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," Tech. Rep., 2018, *arXiv:1801.09847*. [Online]. Available: http://www.open3d.org/

Christopher Xie received the B.Sc. degree in electrical engineering and computer science from the University of California, Berkeley, CA, USA, in 2015, and the M.Sc. degree in computer science from the University of Washington, Seattle, WA, USA, in 2017. He is currently working toward the Ph.D. degree at the Paul G. Allen School of Computer Science and Engineering, University of Washington.

From 2016 to 2019, he was a National Defense Science and Engineering Graduate (NDSEG) Fellow. His research interests include applying machine learning to solve robot perception problems.



Yu Xiang received the B.S. and M.S. degrees in computer science from Fudan University, Shanghai, China, in 2007 and 2010, respectively, and the Ph.D. degree in electrical engineering from the University of Michigan at Ann Arbor, MI, USA, in 2016.

He is a Senior Research Scientist with NVIDIA, Santa Clara, CA, USA. He was a Postdoctoral Researcher with Stanford University, Stanford, CA, USA, and with the University of Washington, Seattle, WA, USA from 2016 to 2017, and was a Visiting Student Researcher with the Artificial Intelligence Lab, Stanford University, from 2013 to 2016. His research interests include robotics and computer vision.

Arsalan Mousavian received his B.Sc. and M.Sc. degrees in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2010, the M.Sc. degree from the University of Tehran, Tehran, in 2013, and the Ph.D. degree in computer science from George Mason University, Fairfax, VA, USA, in 2018.

He is a Senior Research Scientist with NVIDIA, Santa Clara, CA, USA. His research interests include 3D perception methods that help robots accomplish robot manipulation tasks in the real world.

Dieter Fox received his M.S. and Ph.D. degrees in computer science from the University of Bonn, Germany, in 1993 and 1998, respectively.

He is a Professor with the Paul G. Allen School of Computer Science and Engineering, University of Washington, Seatle, WA, USA, where he heads the UW Robotics and State Estimation Lab. He is also Senior Director of Robotics Research with NVIDIA, Santa Clara, CA, USA. His research interests include robotics and artificial intelligence, with a focus on state estimation and perception applied to problems such as mapping, object detection and tracking, manipulation, and activity recognition. He has authored/co-authored more than 200 technical papers and is co-author of the textbook "Probabilistic Robotics."

Dr. Fox is a Fellow of the AAAI. He has received the IEEE RAS Pioneer Award. He was an Editor of the IEEE TRANSACTIONS ON ROBOTICS, Program Co-Chair of the 2008 AAAI Conference on Artificial Intelligence, and Program Chair of the 2013 Robotics: Science and Systems conference.