Learning RGB-D Feature Embeddings for Unseen Object Instance Segmentation

Yu Xiang[1] **Christopher Xie**[2] **Arsalan Mousavian**[1] **Dieter Fox**[1,2]
[1]NVIDIA [2]University of Washington
{yux,amousavian,dieterf}@nvidia.com chrisxie@cs.washington.edu

Abstract: Segmenting unseen objects in cluttered scenes is an important skill that robots need to acquire in order to perform tasks in new environments. In this work, we propose a new method for unseen object instance segmentation by learning RGB-D feature embeddings from synthetic data. A metric learning loss function is utilized to learn to produce pixel-wise feature embeddings such that pixels from the same object are close to each other and pixels from different objects are separated in the embedding space. With the learned feature embeddings, a mean shift clustering algorithm can be applied to discover and segment unseen objects. We further improve the segmentation accuracy with a new two-stage clustering algorithm. Our method demonstrates that non-photorealistic synthetic RGB and depth images can be used to learn feature embeddings that transfer well to real-world images for unseen object instance segmentation.

Keywords: Unseen Object Instance Segmentation, Robot Perception, Sim-to-Real, Metric Learning

1 Introduction

In order to perform complex tasks in different environments autonomously, robots need to learn various skills in perception, planning and control. Among the perception skills, the ability to segment unseen objects in cluttered scenes is critical. Imagine that a robot is tasked to clean a kitchen. It may encounter objects it has never seen before. Object recognition approaches that rely on building 3D models of objects [1, 2] cannot be applied to these scenarios.

To segment unseen objects, robots need to learn the concept of an "object" and generalize it to new objects. This is usually achieved by training an object perception method with many different objects. However, there is no large scale dataset of real images that contains many objects in robotic manipulation scenes. Existing large scale datasets such as ImageNet [3] or COCO [4] are collected from web images, which are quite different from robotic manipulation settings such as tabletop scenes. In addition, there is no depth image available in these datasets, while depth information has proven to be very useful in robotic manipulation [5, 6].

As a result, previous works on Unseen Object Instance Segmentation (UOIS) [7, 8] utilize synthetic data for training. 3D CAD models of objects are used to compose a scene that is rendered into RGB-D images from different viewpoints. The benefit of using synthetic data is that a large number of images can be generated. For example, [7] generated 40,000 scenes with 7 RGB-D images per scene. However, models trained on synthetic data may not work well on real images due to the sim-to-real domain gap, especially when the RGB images are non-photorealistic. A common recipe to solve this sim-to-real problem is to use depth images for training as shown in [7, 8]. Adding non-photorealistic RGB images in training typically degrades performance for these methods.

In this work, we propose a new method for UOIS by learning RGB-D feature embeddings directly from synthetic data. Different from previous methods, we show that our method is able to utilize both depth images and non-photorealistic RGB images to produce feature embeddings for every pixel, which can be used in a clustering algorithm to segment unseen objects. More importantly, adding the non-photorealistic RGB images in our training improves the segmentation accuracy due to our metric learning formulation.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

Specifically, a fully convolutional network is used to process an RGB-D image to produce a dense feature map with the same size as the input image. In training, our goal is to learn to produce feature embeddings such that pixels from the same object are close to each other, and pixels from different objects are separated in the embedding space. To achieve this goal, we apply a metric learning loss based on cosine distance to the unit-length normalized feature embeddings. In testing, we utilize the von Mises-Fisher mean shift algorithm [9] to cluster these feature embeddings, which automatically discovers the number of objects in the image and generates a segmentation mask for each object. In addition, we introduce a two-stage clustering process to further improve the segmentation accuracy. The first stage clusters all the pixels from the input image. Then, for each segmentation mask from the first stage, a Region of Interest (RoI) of the mask is extracted from the input image and the second stage clustering operation is applied to it. The second stage clustering can refine the segmentation boundaries and separate objects that are close to each other if the first stage clustering fails to separate them. We conduct experiments on two real-world RGB-D image datasets [10, 11] to evaluate our method named Unseen Clustering Network (UCN), and demonstrate the state-of-the-art results on these datasets.

In summary, our contributions in this work are i) we show that learning RGB-D feature embeddings with a metric learning formulation can better utilize the synthetic RGB images even if they are non-photorealistic; ii) we experimentally investigate different ways of fusing RGB and Depth; iii) we introduce a new two-stage clustering algorithm for UOIS; iv) our method achieves the state-of-the-art performance on two commonly used datasets for UOIS.

2 Related Work

Unseen Object Instance Segmentation. Earlier works for UOIS apply image segmentation methods to segment objects such as GraphCut [12], Connected Components [13], LCCP [14] and SceneCut [15]. These methods are based on low-level image cues such as edges, contours, connectivity, convexity or symmetry to group pixels into objects. Consequently, they tend to segment every thing in an image and objects are usually over-segmented due to textures or concave shapes. This is mainly because there is no object-level supervision, and these methods cannot learn the concept of object.

On the other hand, recent learning-based approaches generate synthetic data to train neural networks for unseen object segmentation [16, 8, 7], where a large number of images can be rendered using 3D CAD models. By providing object-level supervision, these methods achieve better performance than image-segmentation-based methods. However, using synthetic data for training requires dealing with the sim-to-real gap. Previous works [8, 7] rely on using depth images to bridge the sim-to-real gap. We show that our method can also utilize the non-photorealistic RGB images to learn a feature representation for effectively segmenting unseen objects.

Deep Metric Learning. Recently, combining deep neural networks and metric learning to learn powerful feature representations has been successful in many applications such as image recognition [17], image clustering [18], face recognition [19], reinforcement learning [20] and robotic manipulation [21]. Most works focus on learning visual representations using real images. In this work, we show that it is possible to learn feature embedding networks directly from synthetic RGB-D data that robustly transfer to real images for segmenting unseen objects, even if the synthetic RGB images are non-photorealistic.

3 Method

Given an RGB-D image, our goal is to segment individual objects in the image, where these objects are unseen during training. Since the concept of "objects" is quite general, in this work, we consider objects in robotic manipulation environments and focus on segmenting objects in tabletop scenes, which provides useful information for manipulating these objects as shown in [6, 22].

3.1 Learning from Synthetic Data

We employ deep neural networks for unseen object instance segmentation. In order to segment unseen objects, a network needs to learn the concept of objects and be able to generalize it to new objects. We achieve this by training the network with a large number of objects in tabletop scenes. However, there does not exist a large scale dataset of real-world images for many objects in tabletop

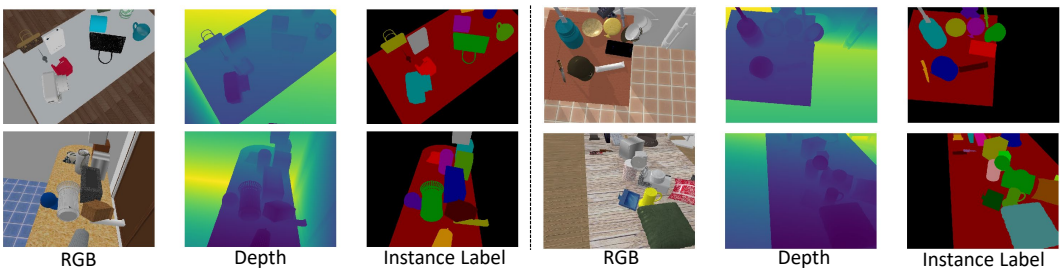

Figure 1: Example RGB-D images and the corresponding instance labels from the Tabletop Object Dataset [7].

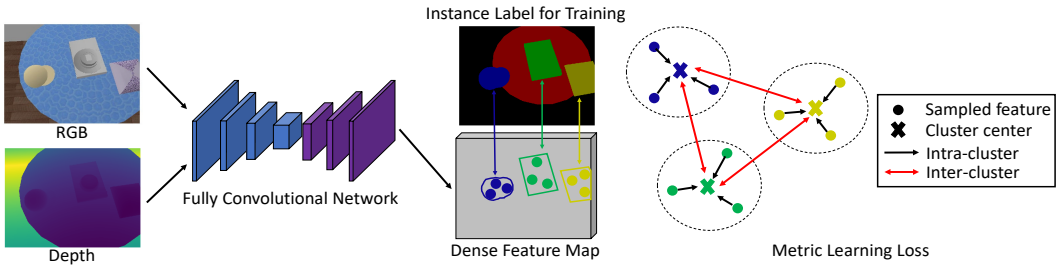

Figure 2: Illustration of our method for learning RGB-D feature embeddings using a fully convolutional network and a metric learning loss, where the loss function pushes pixels on the same object close to the cluster center and pushes the cluster centers of different objects far from each other in the embedding space.

scenes. Therefore, we resort to using synthetic data for training, where it is easy to generate a large scale dataset of many different objects.

Specifically, we utilize the Tabletop Object Dataset generated from [7] for training. This dataset consists of 40,000 synthetic scenes of cluttered objects on a tabletop in home environments. For each scene, a home environment is sampled from the SUNCG house dataset [23], and a table and arbitrary objects are sampled from the ShapeNet dataset [24]. The number of objects for each scene is between 5 to 25. The PyBullet [25] physics simulator is used to place objects on the table until they come to rest. After that, 7 RGB-D images are captured for each scene from different viewpoints. Fig. 1 shows some example images from the dataset.

3.2 Learning RGB-D Feature Embeddings

We can see that from Fig. 1, the synthetic RGB images are non-photorealistic, which causes a domain gap when applying the trained network to real images. As a result, previous works [7, 8] mainly rely on using the depth images for segmentation, since it has been shown that networks trained on synthetic depth images can generalize to real images well [5]. In our work, we investigate how to utilize these non-photorealistic synthetic RGB images combined with depth images to improve unseen object instance segmentation. Our solution is to learn RGB-D feature embeddings for clustering in a metric learning framework.

Specifically, given an RGB image $I \in \mathbb{R}^{H \times W \times 3}$ and depth image $D \in \mathbb{R}^{H \times W}$, where H and W are the image height and width, respectively, we first back-project the depth image D into an "organized" point cloud $P \in \mathbb{R}^{H \times W \times 3}$ using the camera intrinsics. Then a Fully Convolutional Network (FCN) Ψ takes the RGB image and the point cloud image as input, and computes a dense feature map $F = \Psi(I, P) \in \mathbb{R}^{H \times W \times C}$, where C is the dimension of the feature embeddings. Our FCN consists a backbone network and a set of deconvolutional layers to generate a dense feature map. Different backbone networks can be used such as VGG [26], U-Net [27] or ResNet [28].

During inference, we simply apply a clustering algorithm after computing the feature map F to group pixels together using their feature embeddings. Consequently, the goal of training is to make sure pixels from the same object are close to each other while pixels from different objects are far from each other in the embedding space. We apply a metric learning loss function to achieve this goal similar to [29, 30]. First, suppose there are K objects in the input image. For each object, we randomly sample N pixels to compute the loss, where $N = 1000$ in our experiments. Note that background is treated as one of the objects. We found that sampling the same number of pixels per object improves performance by balancing the importance of each object regardless of their

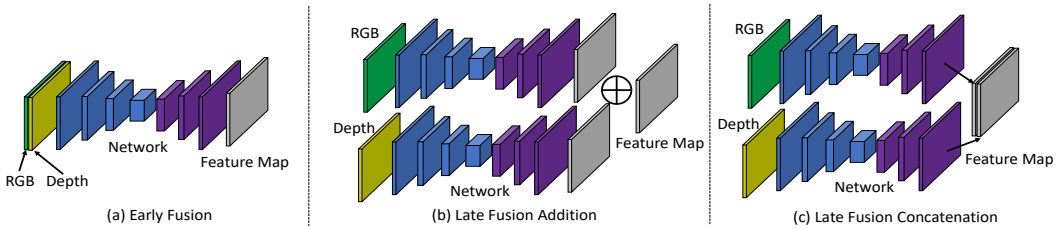

Figure 3: Three different ways of fusing RGB and depth in computing the feature embeddings.

sizes in the image. Second, assuming all feature embeddings are normalized to have unit length, we compute the spherical mean as defined in [30] for the k^{th} object as $\mu^k = \frac{\sum_{i=1}^N \mathbf{x}_i^k}{\|\sum_{i=1}^N \mathbf{x}_i^k\|}$, where $\mathbf{x}_i^k \in \mathbb{R}^C$ denotes the feature embedding of the i^{th} pixel of the k^{th} object. Third, we define the intra-cluster loss function as

$$\ell_{\text{intra}} = \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{N}\frac{\mathbb{1}\left\{d(\mu^k, \mathbf{x}_i^k) - \alpha \geq 0\right\}\ d^2(\mu^k, \mathbf{x}_i^k)}{\sum_{i=1}^{N}\mathbb{1}\left\{d(\mu^k, \mathbf{x}_i^k) - \alpha \geq 0\right\}}, \tag{1}$$

where $d(\mu^k, \mathbf{x}_i^k) = \frac{1}{2}(1 - \mu^k \cdot \mathbf{x}_i^k)$ is the cosine distance between the two unit-length feature vectors, α is the margin for the intra-cluster distance, and $\mathbb{1}$ denotes the indicator function. The above intra-cluster loss function pushes feature embeddings of pixels on the same object close to the cluster center. Fourth, we define the inter-cluster loss function as

$$\ell_{\text{inter}} = \frac{2}{K(K-1)}\sum_{k<k'}\left[\delta - d(\mu^k, \mu^{k'})\right]_+^2, \tag{2}$$

where $[x]_+ = \max(x, 0)$, and δ is the margin for the inter-cluster distance. The inter-cluster loss function pushes the cluster centers of different objects far away from each other in the embedding space. Finally, our loss function for learning RGB-D feature embeddings is $\mathcal{L} = \lambda_{\text{intra}}\ell_{\text{intra}} + \lambda_{\text{inter}}\ell_{\text{inter}}$, where we simply set $\lambda_{\text{intra}} = \lambda_{\text{inter}} = 1$ in our experiments. Fig. 2 illustrates our network and loss function for learning RGB-D feature embeddings.

3.3 Fusing RGB and Depth

Fig. 3 illustrates three different ways of fusing RGB and Depth (the point cloud image in our case) to compute the feature map. We will experiment with them for comparison. In Early Fusion, the RGB image I and the point cloud image P are concatenated before feeding them into the network. In this case, the input of the network is $\phi_{\text{early}}(I, P) \in \mathbb{R}^{H\times W\times 6}$. In Late Fusion Addition, I and P are fed into two towers of the same network architecture to compute two feature maps, and then the two feature maps are summed together. The two towers are trained to have different weights to account for the differences in color and depth. In Late Fusion Concatenation, the two feature maps are concatenated to generate a feature map $F \in \mathbb{R}^{H\times W\times 2C}$ instead.

3.4 Mean Shift Clustering in Embedding Space

We employ the mean shift algorithm [31] to cluster pixels in the embedding space due to its ability to automatically discover the number of clusters by seeking local maxima of the underlying probability distribution. Furthermore, since our feature embeddings are normalized to have unit length, we apply the von Mises-Fisher mean shift (vMF-MS) algorithm [9], where the von Mises-Fisher (vMF) distribution is used. Its probability density function is defined as

$$p(\mathbf{x}; \mu, \kappa) = C(\kappa)\exp(\kappa \mathbf{x}^T \mu), \tag{3}$$

where both $\mathbf{x}$ and μ are unit-length vectors, μ indicates the center of the distribution, κ controls the concentration of the distribution to the vector μ, and $C(\kappa)$ is a normalization constant. Algorithm 1 summarizes the vMF-MS clustering algorithm. We reshape the feature map $F \in \mathbb{R}^{H\times W\times C}$ into a feature embedding matrix $\mathbf{X} \in \mathbb{R}^{n\times C}$, where $n = H \times W$, and feed $\mathbf{X}$ into Algorithm 1.

3.5 Zoom-in Cluster Refinement

We propose a two-stage clustering process to further improve the segmentation accuracy. The motivation is to tackle challenging scenes with objects close to each other or on top of each other. In

Algorithm 1: von Mises-Fisher mean shift clustering

Input: Feature embedding matrix $\mathbf{X} \in \mathbb{R}^{n \times C}$, κ, ϵ, number of seed m, number of iteration T
Sample m initial clustering centers from $\mathbf{X}$ as the m furthest points, denote it as $\mu^{(0)} \in \mathbb{R}^{m \times C}$;
for $t \leftarrow 1$ **to** T **do**
    Compute weight matrix $\mathbf{W} \leftarrow \exp(\kappa \mu^{(t-1)} \mathbf{X}^T)$;
    Update $\mu^{(t)'} \leftarrow \mathbf{W}\mathbf{X}$;
    Normalize each row vector in $\mu^{(t)'}$ to obtain $\mu^{(t)}$;
end
Merge cluster centers in $\mu^{(T)}$ with cosine distance smaller than ϵ ;
Assign each pixel to the closest cluster center ;

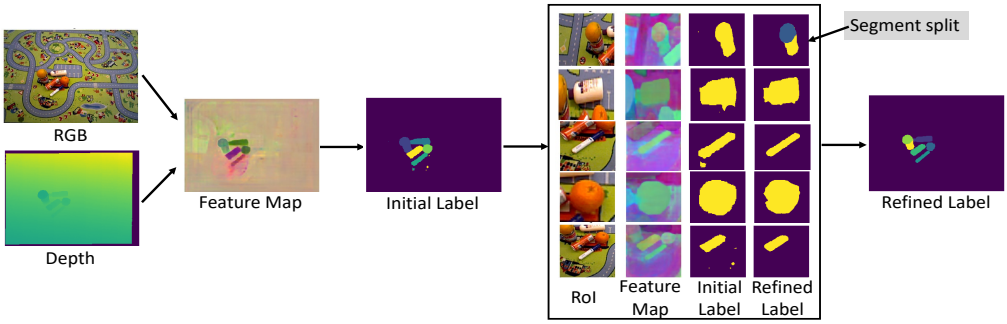

Figure 4: The two-stage clustering process in our method. The first stage clusters feature embeddings of all the image pixels. The second stage refines the segment for each RoI by clustering feature embeddings of the RoI.

these cases, clustering feature embeddings of all the image pixels may under-segment objects, since the network may treat two objects close to each other as one object. Therefore, we introduce a second stage of clustering, which we denote as *zoom-in cluster refinement* as illustrated in Fig. 4.

Specifically, for each cluster from the first stage, we crop an RGB-D Region of Interest (RoI), which is obtained by padding the cluster segment and resizing it to an image patch with size 224×224. Then, feature embeddings are computed for the RoI with an additional network which is trained with synthetic RoIs from the Tabletop Object Dataset. According to our experiments, training a separate network at the RoI-level is necessary. Directly applying the network trained on the whole images does not work well for the RoIs. Finally, the vMF-MS algorithm is used to cluster the feature embeddings of the RoIs.

Since there could be multiple objects in an RoI, for each RoI, we only keep segments which have overlap larger than a predefined threshold (0.5 in our experiments) with the original segment from the first stage, i.e., the target segment to be refined. In this way, the target segment can be refined to have sharper boundary or divided into two separate objects in the under-segmented cases as shown in Fig. 4. The final segmentation label for the whole image is computed by simply aggregating the segments from all the RoIs, and assigning a different object ID for each segment.

4 Experiments

We first conduct ablation studies of several components in our method: Unseen Clustering Network (UCN). For comparison, we adopt Mask R-CNN [32] which is trained to detect foreground objects with bounding boxes and then segment an object inside each bounding box. [8] has successfully shown that Mask R-CNN achieves strong performance on segmenting unknown objects by training with synthetic depth images. After the ablation studies, we present the comparison of our method with the state-of-the-art methods for UOIS, and disucss failure cases from our method.

4.1 Implementation Details

Our network consists of a 34-layer, stride-8 ResNet (ResNet34-8s) with bilinearly upsampling to produce a full resolution 640x480 feature map with embedding dimension $C = 64$. ResNet34-8s achieves better performance than networks using VGG or U-Nets as backbones in our experiments. Pretrained weights of ResNet34-8s on ImageNet [3] are used to initialize the RGB tower in Late

Method	Input	OCID [11] (2390 images)							OSD [10] (111 images)						
		Overlap			Boundary				Overlap			Boundary			
		P	R	F	P	R	F	%75	P	R	F	P	R	F	%75
MRCNN	RGB	77.6	67.0	67.2	65.5	53.9	54.6	55.8	64.2	61.3	62.5	50.2	40.2	44.0	31.9
UCN (Ours)	RGB	54.8	76.0	59.4	34.5	45.0	36.5	48.0	57.2	73.8	63.3	34.7	50.0	39.1	52.5
MRCNN	Depth	85.3	85.6	84.7	**83.2**	76.6	**78.8**	72.7	77.8	85.1	80.6	52.5	57.9	54.6	77.6
UCN (Ours)	Depth	83.1	90.7	86.4	77.7	74.3	75.6	75.4	78.7	83.8	81.0	52.6	50.0	50.9	72.1
MRCNN	RGBD early	78.7	79.0	78.1	73.4	70.3	70.8	62.2	78.3	78.4	78.3	65.2	62.2	63.2	61.2
UCN (Ours)	RGBD early	78.8	89.2	82.8	66.9	69.7	67.2	73.5	77.4	81.8	79.2	53.9	53.0	53.0	69.0
MRCNN	RGBD add	79.6	76.7	76.6	68.7	63.7	64.3	62.9	66.4	64.8	65.5	53.7	43.8	47.5	37.1
UCN (Ours)	RGBD add	**86.0**	**92.3**	**88.5**	80.4	**78.3**	**78.8**	**82.2**	**84.3**	**88.3**	**86.2**	**67.5**	**67.5**	**67.1**	**79.3**
MRCNN	RGBD concat	79.6	76.2	76.0	68.2	63.5	63.7	63.0	67.0	63.8	65.3	53.1	42.7	46.5	37.1
UCN (Ours)	RGBD concat	79.2	87.8	82.9	70.6	67.5	68.5	68.3	76.4	83.3	79.7	50.5	48.5	48.8	67.5

Table 1: Evaluation of our method and Mask R-CNN [32] trained on different input modes.

Fusion Addition and Late Fusion Concatenation (Fig. 3(b)(c)). For early fusion and the depth tower in late fusion, no pretrained weights are used. The network is trained with the Adam optimizer [33] for 16 epochs on the Tabletop Object Dataset with batch size 16 and learning rate 1e-5.

We train Mask R-CNN with the same number of epochs on the Tabletop Object Dataset using the SGD optimizer. We modified Mask R-CNN to support training with RGB-D images, where the two-tower structure is used in late fusion. Similarly, pretrained weights on ImageNet are only used for the RGB tower in Mask R-CNN.

In our metric learning loss, we set margin $\alpha = 0.02$ in the intra-cluster loss, and margin $\delta = 0.5$ in the inter-cluster loss. In the von Mises-Fisher mean shift clustering, we set $\kappa = 20$ and $\epsilon = 2\alpha$. We run the clustering for 10 iterations on 100 initial seeds. The run time of our method is on average 0.2s for the 1st stage and 0.05s per object in the 2nd stage on a TITAN Xp GPU for 640x480 images.

4.2 Datasets and Evaluation Metrics

We evaluate our methods on the Object Clutter Indoor Dataset (OCID) [11] and the Object Segmentation Database (OSD) [10] for UOIS in tabletop scenes. OCID contains 2,390 RGB-D images with up to 20 objects in an image, while OSD has 111 RGB-D images with up to 15 objects in an image. On average, there are 7.5 objects per image in OCID and 3.3 objects per image in OSD. Both datasets contain challenging images of cluttered scenes. The main difference between the two datasets is that, the images in OSD are manually annotated, while the annotations in OCID are obtained in a semi-automatic way by adding objects to each scene one by one and then checking the depth differences. As a result, the segmentation masks from OCID are prone to the noise from the depth images. We use all images from the two datasets for testing.

Following [7], we use precision, recall and F-measure to evaluate the object segmentation performance. These three metrics are first computed between all pairs of predicted objects and ground truth objects. Then the Hungarian method with pairwise F-measure is used to compute a matching between predicted objects and ground truth. Given this matching, the final precision, recall and F-measure are computed by $P = \frac{\sum_i |c_i \cap g(c_i)|}{\sum_i |c_i|}$, $R = \frac{\sum_i |c_i \cap g(c_i)|}{\sum_j |g_j|}$, $F = \frac{2PR}{P+R}$, where c_i denotes the set of pixels belonging to predicted object i, $g(c_i)$ is the set of pixels of the matched ground truth object of c_i, and g_j is the set of pixels for ground truth object j.

We denote the above three metrics as Overlap P/R/F since the true positives are counted by the pixel overlap of the whole object. [7] also introduces the Boundary P/R/F to evaluate how sharp the predicted boundary matches against the ground truth boundary, where the true positives are counted by pixel overlap of the two boundaries. Additionally, we employ the percentage of segmented objects with Overlap F-measure $\geq 75\%$ as a metric to indicate the percentage of objects that can be segmented with a certain accuracy [34].

4.3 Effect of Input Mode

We first study the effect of different input modes in training our method and Mask R-CNN on the Tabletop Object Dataset for UOIS. We train different networks of both methods using RGB, depth, and RGB-D images as input. When using RGB-D images, three different ways for fusing RGB and depth as shown in Fig. 3 are investigated.

Table 1 presents the segmentation results on OCID and OSD. From the table, we can see that i) using depth information significantly boosts the performance for both methods. This is mainly because training with the non-photorealistic RGB images only does not work well for real world images;

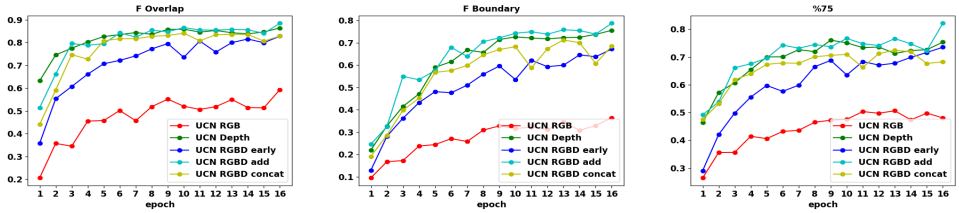

Figure 5: Learning curves of our method on the OCID dataset in terms of different number of training epochs.

UCN (Ours)	OCID [11] (2390 images)							OSD [10] (111 images)						
	Overlap			Boundary				Overlap			Boundary			
	P	R	F	P	R	F	%75	P	R	F	P	R	F	%75
RGB	54.8	76.0	59.4	34.5	45.0	36.5	48.0	57.2	73.8	63.3	34.7	50.0	39.1	52.5
RGB + Zoom-in	55.5	71.1	58.1	37.2	52.1	40.8	55.2	59.1	71.7	63.8	34.3	53.3	39.5	52.6
Depth	83.1	90.7	86.4	77.7	74.3	75.6	75.4	78.7	83.8	81.0	52.6	50.0	50.9	72.1
Depth + Zoom-in	84.7	88.4	86.4	79.7	80.4	79.6	81.5	83.5	86.0	84.5	57.2	56.8	56.5	80.6
RGBD early	78.8	89.2	82.8	66.9	69.7	67.2	73.5	77.4	81.8	79.2	53.9	53.0	53.0	69.0
RGBD early + Zoom-in	82.3	87.6	84.0	70.7	77.2	72.5	81.6	79.4	79.7	79.2	57.3	58.6	57.1	67.9
RGBD add	86.0	**92.3**	**88.5**	80.4	78.3	78.8	82.2	84.3	**88.3**	86.2	67.5	67.5	67.1	79.3
RGBD add + Zoom-in	**87.4**	88.7	87.8	**82.2**	**83.3**	**82.3**	**85.6**	**87.4**	87.4	**87.4**	**69.1**	**70.8**	**69.4**	**83.2**
RGBD concat	79.2	87.8	82.9	70.6	67.5	68.5	68.3	76.4	83.3	79.7	50.5	48.5	48.8	67.5
RGBD concat + Zoom-in	84.2	86.4	85.1	78.2	79.5	78.3	78.7	82.0	85.0	83.4	57.8	59.2	57.9	76.5

Table 2: Evaluation of the proposed zoom-in cluster refinement algorithm on different input modes.

ii) For Mask R-CNN, using depth as input achieves the best performance. Using RGB-D images as input is worse than just using depth for any of the three fusion ways; iii) For our method, using RGB-D images with Late Fusion Addition achieves the best performance, which is also better than Mask R-CNN trained on depth. This result demonstrates that, unlike Mask R-CNN, our method is able to utilize the non-photorealistic RGB images in addition to the depth images to learn powerful feature embedding networks that can be effectively transferred to real world images. Fig. 5 shows the learning curves of our method in different input modes on the OCID dataset.

4.4 Effect of the Two-stage Clustering Algorithm

We investigate the effect of our two-stage clustering algorithm to see how much the zoom-in cluster refinement can help. Table 2 shows the results before and after refinement for the five different input modes. As we can see from the table, the zoom-in cluster refinement algorithm significantly improves the Boundary P/R/F and the percentage of objects with F-measure larger than 0.75 on both datasets. This result confirms that the second stage clustering generates sharper object boundaries and helps separate objects that are under-segmented from the first stage. For the Overlap P/R/F, the refinement scarifies a small amount of recall in the trade off of better precision.

Fig. 6 shows some qualitative results of our method on OCID and images from our lab using RGB-D images as input with Late Fusion Addition. We can see how the object boundaries are refined and

Figure 6: Examples of the feature maps, initial labels from the first stage clustering and the refined labels after the second stage clustering from our method. The last two columns show results on images from our lab.

Method	OCID [11] (2390 images)							OSD [10] (111 images)						
	Overlap			Boundary				Overlap			Boundary			
	P	R	F	P	R	F	%75	P	R	F	P	R	F	%75
MRCNN Depth [32]	85.3	85.6	84.7	**83.2**	76.6	78.8	72.7	77.8	85.1	80.6	52.5	57.9	54.6	77.6
UOIS-Net-2D [7]	**88.3**	78.9	81.7	82.0	65.9	71.4	69.1	80.7	80.5	79.9	66.0	67.1	65.6	71.9
UOIS-Net-3D [35]	86.5	86.6	86.4	80.0	73.4	76.2	77.2	85.7	82.5	83.3	**75.7**	68.9	**71.2**	73.8
UCN (Ours)	87.4	**88.7**	**87.8**	82.2	**83.3**	**82.3**	**85.6**	**87.4**	**87.4**	**87.4**	69.1	**70.8**	69.4	**83.2**

Table 3: Comparison with the state-of-the-art methods for unseen object instance segmentation.

objects are correctly separated from each other after refinement in these examples. To visualize the feature map with dimension 64, we first sum along channels with interval three to obtain an image with dimension 3 and then normalize all the values between 0 and 255 for visualization. We can clearly see that from the feature map, objects are separated in the embedding space.

4.5 Comparison to the State-of-the-art Methods

We compare our method to the state-of-the-art methods for UOIS in the literature. We use our best model: the network trained with RGB-D images in Late Fusion Addition plus the zoom-in refinement. Table 3 presents the results. Our approach achieves the best Overlap F-measure and Boundary F-measure on the OCID dataset, and the best Overlap F-measure on the OSD dataset. Additionally, we significantly outperform the other methods on the percentage of objects segmented with F-measure larger than 0.75. Note that all methods are trained on the Tabletop Object Dataset.

The main difference between our method and Mask R-CNN [32] and UOIS-Net [35] is that, our method is a bottom-up approach and we learn a distance metric in the embedding space to compare pixels. Both Mask R-CNN and UOIS-Net have softmax functions to classify pixels into foreground and background. As a result, our method can utilize the non-photorealistic synthetic RGB images to help learning the distance metric, while preventing overfitting to the synthetic images. Due to the nature of a bottom-up approach in grouping pixels together, our method obtains higher recall compared to Mask R-CNN and UOIS-Net.

4.6 Discussion of Failure Cases

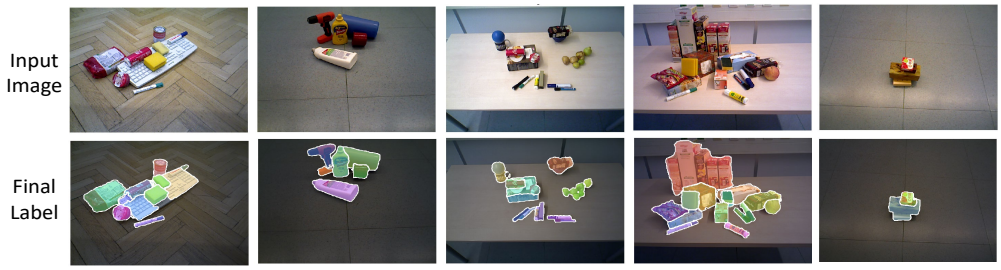

Figure 7: Examples of segmentation failure cases from our method.

We discuss segmentation failure cases from our method and show some examples in Fig. 7. The majority failures are under-segmentation. In some cases, our method cannot separate similar objects that are very close to each other or on top of each other, even with the zoom-in refinement step. As we can see in Fig. 7, the method fails to separate two pens, a group of similar fruits or a row of boxes in these examples. Another failure is over-segmentation, and we see this mainly happens for the power drill and some keyboards in the OCID dataset. Future work could investigate how to fix there failures using multi-view images or active perception with robots.

5 Conclusion

We have introduced a simple but effective approach for UOIS by learning RGB-D feature embeddings from synthetic data. Different from previous approaches that mainly rely on synthetic depth for segmentation, our approach can utilize the non-photorealistic RGB images jointly with the depth images to learn a distance metric for clustering. We also introduce a new two-stage clustering algorithm to tackle challenging scenes when objects are close to each other. Our experiments demonstrate the effectiveness of our approach for UOIS on real-world images.

References

[1] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018.

[2] 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 *Conference on Robot Learning (CoRL)*, 2018.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[5] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[6] A. Mousavian, C. Eppner, and D. Fox. 6-DOF graspnet: Variational grasp generation for object manipulation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2901–2910, 2019.

[7] 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 *Conference on Robot Learning (CoRL)*, 2019.

[8] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg. Segmenting unknown 3D objects from real depth images using mask R-CNN trained on synthetic data. In *International Conference on Robotics and Automation (ICRA)*, pages 7283–7290, 2019.

[9] T. Kobayashi and N. Otsu. Von mises-fisher mean shift for clustering on a hypersphere. In *International Conference on Pattern Recognition (ICPR)*, pages 2130–2133, 2010.

[10] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4796, 2012.

[11] M. Suchi, T. Patten, D. Fischinger, and M. Vincze. EasyLabel: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets. In *International Conference on Robotics and Automation (ICRA)*, pages 6678–6684, 2019.

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.

[13] A. J. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, 2013.

[14] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter. Object partitioning using local convexity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311, 2014.

[15] T. T. Pham, T.-T. Do, N. Sünderhauf, and I. Reid. SceneCut: Joint geometric and object segmentation for indoor scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2018.

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

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[18] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016.

[19] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, pages 815–823, 2015.

[20] A. Srinivas, M. Laskin, and P. Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[21] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning (CoRL)*, pages 373–385, 2018.

[22] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox. 6-DOF grasping for target-driven object manipulation in clutter. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[23] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1754, 2017.

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

[25] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

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

[30] C. Xie, Y. Xiang, Z. Harchaoui, and D. Fox. Object discovery in videos as foreground motion clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9994–10003, 2019.

[31] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619, 2002.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2961–2969, 2017.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(6):1187–1200, 2013.

[35] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. Unseen object instance segmentation for robotic environments. *arXiv preprint arXiv:2007.08073*, 2020.