RICE: Refining Instance Masks in Cluttered Environments with Graph Neural Networks

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

Abstract: Segmenting unseen object instances in cluttered environments is an important capability that robots need when functioning in unstructured environments. While previous methods have exhibited promising results, they still tend to provide incorrect results in highly cluttered scenes. We postulate that a network architecture that encodes relations between objects at a high-level can be beneficial. Thus, in this work, we propose a novel framework that refines the output of such methods by utilizing a graph-based representation of instance masks. We train deep networks capable of sampling smart perturbations to the segmentations, and a graph neural network, which can encode relations between objects, to evaluate the perturbed segmentations. Our proposed method is orthogonal to previous works and achieves state-of-the-art performance when combined with them. We demonstrate an application that uses uncertainty estimates generated by our method to guide a manipulator, leading to efficient understanding of cluttered scenes. Code, models, and video can be found at https://github.com/chrisdxie/rice.

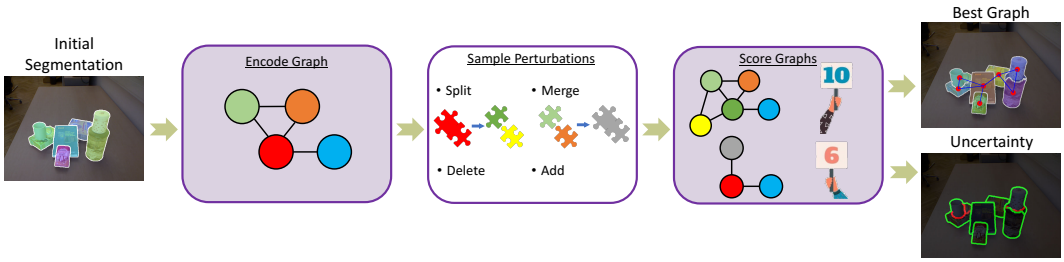


Figure 1: High-level overview of our proposed method. Given an initial segmentation, we encode it as a graph, sample perturbations, then score the resulting segmentation graphs. The highest scoring graph and/or contour uncertainty is output. Best viewed in color and zoomed in.

1 Introduction

Perception lies at the core of the ability of a robot to function in an unstructured environment. A critical component of such a perception system is its capability to solve Unseen Object Instance Segmentation (UOIS), as it is infeasible to assume all possible objects have been seen in a training phase. Proper segmentation of these unseen instances can lead a better understanding of the scene, which can then be exploited by algorithms such as manipulation [1, 2, 3] and re-arrangement [4].

Many methods for UOIS directly predict segments from raw sensory input such as RGB and/or depth images. While recent methods have shown strong results for this problem [5, 6, 7, 8], they still tend to fail when dealing with highly cluttered scenes, which are common in manipulation scenarios. A natural thought is that an architecture with relational reasoning can benefit the predictions. For example, it can potentially learn to recognize common object configurations (e.g. realizing that one object is stacked on top of another). While relational inductive biases have shown to be useful for problems such as scene graph prediction [9, 10, 11], it remains to be seen whether it can be useful in identifying objects in dense clutter. In this work, we investigate the use of graph neural networks, which can encode relations between objects, for segmenting densely cluttered unseen objects.

In this paper, we propose a novel method for Refining Instance masks in Cluttered Environments, named RICE. Given an initial instance segmentation of unseen objects, we encode it into a *segmentation graph*, where individual masks are encoded as nodes and connected with edges when they are close in pixel space. Starting from this initial graph, we build a tree of sampled segmentation graphs by perturbing the leaves in a CEM-style (Cross Entropy Method) framework, where example perturbations include splitting and merging. We learn Sampling Operation Networks (SO-Nets) that sample efficient and smart perturbations that generally lead to better segmentations. The perturbed segmentation graphs are scored with a graph neural network, denoted Segmentation Graph Scoring Network (SGS-Net). Finally, we can return the highest scoring segmentation or compute contour uncertainties, depending on the application. Figure 1 provides a high-level illustration of our method.

RICE is able to improve the results of existing techniques to deliver state-of-the-art performance for UOIS. An investigatory analysis reveals that applying SGS-Net on top of the SO-Nets results in more accurate and consistent predictions. In particular, we find that SGS-Net learns to rank segmentation graphs better than SO-Nets alone. Additionally, we provide a proof-of-concept efficient scene understanding application that utilizes uncertainties output by RICE to guide a manipulator.

In summary, our main contributions are: 1) We propose a novel framework that utilizes a new graph-based representation of instance segmentation masks in cluttered scenes, where we learn deep networks capable of suggesting smart perturbations and scoring of the graphs. 2) Our method achieves state-of-the-art results for UOIS when combined with previous methods. 3) We demonstrate that uncertainty outputs from our method can be used to perform efficient scene understanding.

2 Related Work

Instance Segmentation Traditional methods for 2D instance segmentation include GraphCuts [12], Connected Components [13], and LCCP [14]. Recently, learning-based approaches have provided more semantic solutions. For example, top-down solutions combine segmentation with object proposals in the form of bounding boxes [7, 15, 16, 17]. Mask R-CNN [7] predicts a foreground mask for each proposal produced by its region proposal network (RPN). 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 [18, 19, 20, 21, 22]. Some examples of this include contrastive losses [18] and unrolling mean shift clustering as a neural network to learn pixel embeddings [22].

Most of the afore-mentioned algorithms provide instance masks with category-level semantic labels, which do not generalize to unseen objects in novel categories. Class-agnostic methods [23, 24, 25, 26] and motion segmentation [27, 28, 29] methods have been investigated for this problem. In robotic perception, Xie et al. [30] proposed to separate the processing of depth and RGB in order to generalize their method from sim-to-real settings and provide sharp masks. Their follow-on work [5] proposed a 3D voting method to overcome the limitations of their earlier 2D method. Xiang et al. [6] showed that training a network on RGB-D with simulated data and a simple contrastive loss [18] can demonstrate strong results for this problem. While these methods show promise, they are not perfect and still admit mistakes in cluttered scenes, which can hamper downstream robot tasks that rely on such perception. Our method is orthogonal to these works, and is designed to refine their outputs by sampling perturbations to result in better instance segmentations in the cluttered environments.

Graph Neural Networks Graph neural networks (GNN) in vision and robotics have recently become a useful tool for learning relational representations. They have found applications in many standard computer vision tasks such as image classification [31, 32], object detection [33], semantic segmentation [34], and question answering [35]. GNNs have also been used to perform "scene graph generation", which requires predicting not just object detections, but also the relations between the objects [9, 10, 11]. The resulting scene graphs have been used for applications such as image retrieval [36]. GNNs have also been used to learn object dynamics, properties, and relations for applications such as differential physics engines [37, 38]. Our proposed work represents instance segmentation masks as graphs and utilizes this architecture in order to refine the predicted masks.

3 Method

Our method, RICE, is designed to Refine Instance masks of unseen objects in Cluttered Environments. Given an initial segmentation mask $S \in \mathbb{N}^{H \times W}$ of unseen objects, we first encode this as a *segmentation graph* G_S, which is described in Section 3.1. Then, in Section 3.2, we build a tree T of sampled segmentation graphs by perturbing the leaves in a CEM-style [39] framework. Section 3.3 details the sampling operations, which are parameterized by our Sampling Operation Networks (SO-Nets). Each candidate graph (tree node) is scored by a GNN named Segmentation Graph Scoring Network (SGS-Net), introduced in Section 3.4. Finally, the highest scoring graph in T and/or contour uncertainties are returned. Figure 1 provides a high-level illustration of RICE, and pseudocode can be found in the Supplement (Algorithm 2).

3.1 Node Encoder

Given a single instance mask $S_i \in \{0,1\}^{H \times W}$ for instance i, we crop the RGB image $I \in \mathbb{R}^{H \times W \times 3}$, an organized point cloud $D \in \mathbb{R}^{H \times W \times 3}$ (computed by backprojecting a depth image with camera intrinsics), and the mask S_i with some padding for context. We then resize the crops to $h \times w$ and feed these into a multi-stream encoder network which we denote as the Node Encoder. This network applies a separate convolutional neural network (CNN) to each input, and then fuses the flattened outputs to provide a feature vector $\mathbf{v}_i$ for this node. See Figure 2 for a visual illustration of the network. Note that we also encode the background mask as a node in the graph. This gives the segmentation graph $G_S = (V, E)$, where each $\mathbf{v}_i \in V$ corresponds to an individual instance mask, and nodes are connected with undirected edges $e = (i, j) \in E$ if their set distance is less than a threshold.

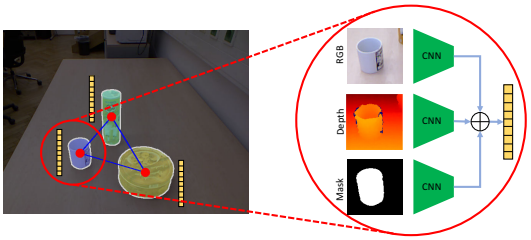


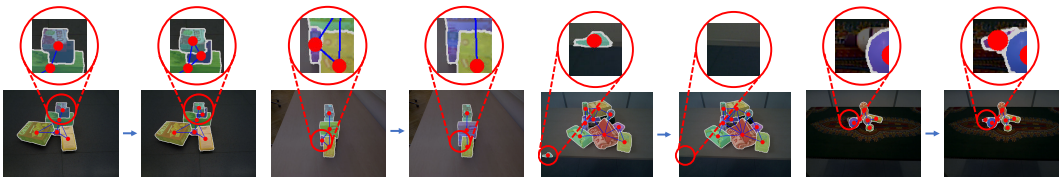
Figure 2: Given an initial instance segmentation mask (left), our segmentation graph representation encodes each individual mask as a graph node (red dots) with a corresponding feature vector $\mathbf{v}_i$ (yellow bar) output by the Node Encoder (right). Edges (blue lines) connect nearby masks.

3.2 Building the Sample Tree

Our sample tree-building procedure operates in a CEM-style fashion. CEM [39] is an iterative sampling-based optimization algorithm that updates its sampling distribution based on an "elite set" of the top k (or top percentile) samples. For more details, we refer the reader to [39]. Following this terminology, our elite set consists of the leaves of our sample tree T, each of which are guaranteed to be better with respect to our proxy objective function, SGS-Net. Then, the sampling distribution is implicitly defined by the SO-Nets; while we cannot explicitly write out the distribution, we can certainly sample from it with our sampling operations described in Section 3.3.

Our sample tree T starts off with the root G_S. We expand the tree from the leaves with K expansion iterations. For each expansion iteration, we iterate through the current leaves of T. For a leaf G, we randomly choose a sample operation from Section 3.3 and apply it to G to obtain candidate graph G'. We then compare the scores $s_G, s_{G'}$ output by SGS-Net, and add G' to T as a child of G if $s_{G'} > s_G$. Thus, any leaf of T is guaranteed to be at least as good as the root G_S w.r.t. our proxy objective function SGS-Net. We apply this procedure B times for G, such that each tree node can have a maximum of B children. Thus, B is a branching factor. Finally, due to constraints of limited GPU memory, we exit the process in an anytime fashion whenever we exceed a budget of maximum graph nodes and/or graph edges (not to be confused with tree nodes/edges). See the Supplement for pseudocode (Algorithm 2) and an example of the sample tree-building procedure (Figure 12).

It is important to note that while we utilize our learned SO-Nets and SGS-Net to build the sample tree T, they are applied in different manners (although they are trained on the same dataset). In order to add a candidate graph to the tree, they must both agree in the sense that the perturbation must be suggested via an SO-Net and SGS-Net must approve of the candidate graph via its score. This redundancy offers a level of robustness, Section 4.4 shows that the combination of these leads to more accurate performance with lower variance.



(a) Split (b) Merge (c) Delete (d) Add

Figure 3: We show real-world examples of the sampling operations and how they can refine the original segmentation. Best viewed in color on a computer screen and zoomed in.

3.3 Sampling Operations

We consider four sampling operations: 1) splitting, 2) merging, 3) deleting, and 4) adding. However, randomly performing these operations leads to inefficient samples which wastes computation time and memory. For example, it is not clear how to split or add an instance mask randomly such that it may potentially result in a better segmentation. Thus, we introduce two networks for these four operations, SplitNet and DeleteNet, which comprise our SO-Nets. They are learned to suggest smart perturbations to bias the sampling towards better graphs, lowering the amount of samples needed in order to favorably refine the segmentation. Examples of each operation can be found in Figure 3.

Split It is not clear how to randomly split a mask such that it provides an effective split. For example, a naive thing to do is to sample a straight line to split the mask, however in many cases this will not result in a reasonable split (see Figure 3a for an example). Thus, we propose to learn a deep network denoted SplitNet to handle this. SplitNet takes the output of the Node Encoder (before flattening), fuses them with concatenation followed by a convolution, then passes them through a single decoder with skip connections. Essentially it is a multi-stream encoder-decoder U-Net [40] architecture, much like Y-Net [27], except that it has three streams for RGB, depth, and the mask. The output of SplitNet is a pixel-dense probability map $p_i \in [0,1]^{h \times w}$ of split-able object boundaries. To sample a split for instance mask S_i, we first sample two end points on the contour of the original mask S_i, and calculate the highest probability path from the end points that travels through p_i, resulting in a trajectory $\tau = \{(u_t, v_t)\}_{t=1}^{L_i}$ of length L_i. We score the split with $s_\tau = \frac{1}{L_i} \sum_t p_i[u_t, v_t] \in \mathbb{R}$, which is the average probability along the sampled path. More details can be found in the Supplement (Section A.2).

Merge We exploit the fact that merging is the opposite of splitting and adapt SplitNet for this operation. For each pair (i, j) of neighboring masks, we take their union S_{ij} and pass it through SplitNet to get p_{ij}. Note that we do not consider merging disjoint masks that may belong to the same instance, which is a limitation of this work. To compute the merge score m_{ij}, we first compute the union of the boundaries of S_i and S_j, denoted $B_{ij} \in \{0,1\}^{h \times w}$. Then, we calculate the merge score as $m_{ij} = 1 - (p_{ij} \odot B_{ij} / (\mathbf{1}^\intercal p_{ij} \mathbf{1}))$ where $\odot$ is element-wise multiplication, $\mathbf{1}$ is a vector of ones. This is essentially a weighted average of B_{ij} with weights p_{ij}.This score indicates how likely SplitNet thinks S_i and S_j correspond to different objects. Figure 3b shows an ideal merge operation.

Delete We design a network, DeleteNet, to provide delete scores $d_i \in \mathbb{R}$ for every instance (graph node) i. This network is also built on top of the Node Encoder: it computes the difference $\mathbf{v}_i - \mathbf{v}_{bg}$, where $\mathbf{v}_{bg}$ is the feature vector for the background node output by the Node Encoder. This difference is then provided as input to a multi-layer perceptron (MLP) which outputs a scalar d_i. See Figure 3c for an example of how DeleteNet can help remove false positives from the segmentation.

Add Similarly to merging, we can exploit the fact that adding is the opposite of deleting. Given a candidate mask S_{N+1} to add to the graph, we can use DeleteNet to compute its delete score d_{N+1}. If d_{N+1} is below a threshold, we successfully add the mask to the graph. However, the question remains of how to generate such candidate masks. Given an external foreground mask $F \in \{0,1\}^{H \times W}$ (provided by UOIS-Net-3D [5]), we run connected components on $F \setminus \{\cup_i S_i\}$, and use the discovered components as potential new masks. A successful addition operation can be seen in Figure 3d.

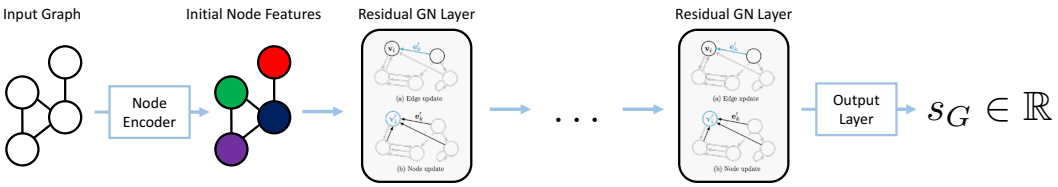


Figure 4: A high-level illustration of our Segmentation Graph Scoring Network (SGS-Net). It is composed of a Node Encoder (see Figure 2), multiple Residual GraphNet Layers, and an output layer. We borrowed elements from Figure 3 of Battaglia et al. [41].

3.4 Segmentation Graph Scoring Network

While our sample operations provide efficient samples that typically lead to better segmentation graphs, they can also suggest samples that worsen the segmentation. Thus, we learn SGS-Net which acts as a proxy for the objective function in the CEM framework. Our proposed SGS-Net learns to score a segmentation graph by considering the fused feature vectors $\mathbf{v}_i$ in context of their neighboring graph nodes (masks). We posit that this context will aid SGS-Net in predicting whether the perturbations improve the segmentation. For example, it can potentially learn to recognize common object configurations from the training set, and score such configurations higher.

A high-level illustration of SGS-Net can be found in Figure 4. The initial node features $\mathbf{v}_i^{(0)}$ are given by the Node Encoder, and we obtain initial edge features $\mathbf{e}_{ij}^{(0)}$ by running the Node Encoder on all neighboring union masks S_{ij}. Then, we pass them through multiple Residual GraphNet Layers (RGLs), which are essentially GraphNet Layers [41] with a residual connection. We refer readers to Battaglia et al. [41] for details of GraphNet Layers, and also provide a full mathematical specification of RGLs in the Supplement (Section B) for completeness. The output of SGS-Net is a scalar score in $[0, 1]$.

3.5 Training Procedure

For SplitNet, we apply a weighted binary cross entropy (BCE) loss to the probability map p: $\ell_{\text{split}} = \sum_u w_u \, \ell_{bce}\left(p_u, \hat{p}_u\right)$ where u ranges over pixels, $\hat{p} \in \{0,1\}^{h \times w}$ is ground truth boundary, and ℓ_{bce} is the binary cross entropy loss. The weight w_u is inversely proportional to the number of pixels with labels equal to $\hat{p}_u$. DeleteNet is also trained with standard BCE loss. SGS-Net is trained with ℓ_{bce} to regress to $.8F + .2F@.75$, where F is the Overlap F-measure [30] and $F@.75$ is the Overlap $F@.75$ measure [42]. The latter measures the percentage of correctly segmented instances. Thus, SGS-Net learns to predict a score based on the number of correctly identified pixels and instances. Note that this regression problem is very difficult to solve. However, the scores do not actually matter as long as the relative scoring is correct, since building the sample tree relies only on this (Section 3.2). In Section 4.5 we show that while SGS-Net may not solve the regression problem well, it learns to rank graphs accurately. Further training and implementation details can be found in the Supplement (Section C).

4 Experiments

4.1 Encoding RGB and Modality Tuning

We use ResNet50 [43] with Feature Pyramid Networks [44] (FPN) to encode RGB images before passing them to the Node Encoder. However, since we are training with (a more cluttered version of) the non-photorealistic synthetic dataset from Xie et al. [30], we perform modality tuning [45], where we fine-tune earlier convolutional layers of ResNet50 during training, and use the COCO [46] pretrained weights during inference. For all experiments, we modality tune the `conv1` and `conv2_1` blocks of ResNet. We provide an experiment in the Supplement (Section E.1) that shows this setting is optimal.

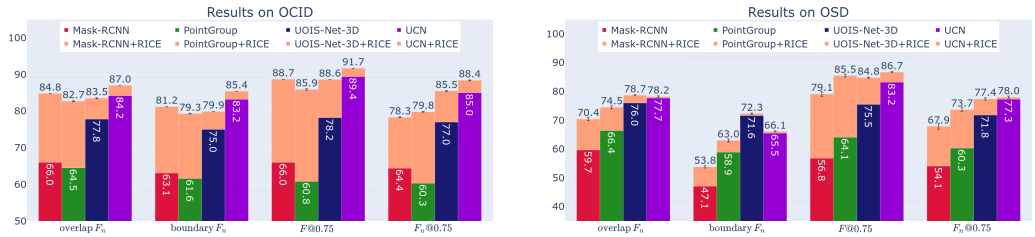


Figure 5: Applying RICE to refine results from state-of-the-art instance segmentation methods leads to improved performance across the board. Note that standard deviation bars are shown, but are very tight and difficult to see.

4.2 Datasets and Metrics

We evaluate our method on two real-world datasets of challenging cluttered tabletop scenes: OCID [47] and OSD [48], which have 2346 images of semi-automatically constructed labels and 111 manually labeled images, respectively. Our SO-Nets and SGS-Net are trained on a more cluttered version of the synthetic Tabletop Object Dataset (TOD) [30], where each scene has anywhere between 20 and 30 ShapeNet [49] objects. We use 20k scenes in total, with 5 images per scene.

Xie et al. [30] introduced the Overlap P/R/F and Boundary P/R/F measures for the problem of UOIS. However, these metrics do not weight objects equally; they are dependent on the size and larger objects tend to dominate the metrics. Thus, we introduce a variation to these metrics that equally weights the errors of individual objects regardless of their size. Given a Hungarian assignment A between the predicted instance masks $\{S_i\}_{i=1}^{N}$ and the ground truth instance masks $\{\hat{S}_j\}_{j=1}^{M}$, we compute our Object Size Normalized (OSN) P/R/F measures as follows:

$$P_n = \frac{\sum_{(i,j)\in A} P_{ij}}{N}, \quad R_n = \frac{\sum_{(i,j)\in A} R_{ij}}{M}, \quad F_n = \frac{\sum_{(i,j)\in A} F_{ij}}{\max(M,N)}, \quad F_n@.75 = \frac{\sum_{(i,j)\in A} \mathbf{1}\{F_{ij} >= 0.75\}}{\max(M,N)},$$

where P_{ij}, R_{ij}, F_{ij} are the precision, recall, and F-measure of $S_i, \hat{S}_j$. Note that the F_n@.75 penalizes both false positive and false negative instances, as opposed to the normal F@.75, which does not penalize false positives. Similarly to Xie et al. [30], we can apply the OSN metrics to the pixels and boundaries, giving us Overlap and Boundary $P_n/R_n/F_n$ measures. For comparison, we also show results with the normal Overlap and Boundary P/R/F measures in the appendix.

We run each experiment 5 times and show means and standard deviations for all metrics.

4.3 SOTA Improvements

We demonstrate how RICE can improve upon predicted instance segmentations from state-of-the-art methods. In particular, we apply it to the results of Mask R-CNN [7], PointGroup [8], UOIS-Net-3D [5], and UCN [6]. We employ RICE by returning the best segmentation from the leaves as scored by SGS-Net. For brevity, we only show Overlap F_n, Boundary F_n, F@.75, and F_n@.75 in Figure 5 on both OCID and OSD. The light orange bars show the additional performance that RICE provides over the output of the methods. Standard deviations are shown as error bars, but are in general very narrow, showing that our method provides consistent results despite its stochasticity. RICE provides substantial improvements to all methods. The largest gains occur in Mask R-CNN and PointGroup, with 21.6% and 32.3% relative gain in F_n@.75 on OCID, respectively. Additionally, on the already strong results from UOIS-Net-3D and UCN, RICE achieves 11.0% and 4.0% relative gain in F_n@.75 on OCID, respectively. These results are similar on OSD, with the gains being slightly less pronounced, which we believe is due to OSD being a smaller dataset with less clutter. Note that applying RICE increases both F@.75 and F_n@.75, indicating that not only is it capturing the object identities correctly, it is not simultaneously predicting more instances (false positives). In the appendix, we show full results for all metrics including P_n, R_n, and normal P/R/F metrics.

SO-Nets	SGS-Net	Overlap			Boundary				
		P_n	R_n	F_n	P_n	R_n	F_n	F@0.75	F_n@0.75
✗	✗	85.1 (–)	83.0 (–)	77.8 (–)	84.6 (–)	76.5 (–)	75.0 (–)	78.2 (–)	77.0 (–)
✓	✗	84.7 (1.23)	89.4 (0.19)	82.3 (1.09)	82.7 (1.37)	82.8 (0.19)	78.7 (1.10)	89.0 (0.26)	84.2 (1.26)
✓	✓	86.3 (0.03)	89.1 (0.01)	83.6 (0.05)	84.5 (0.04)	82.5 (0.04)	80.0 (0.04)	88.5 (0.02)	85.5 (0.05)

Table 1: Ablation to test the utility of SO-Nets and SGS-Net on OCID [47] starting from UOIS-Net-3D [5] masks. Only using the sample operator networks (SO-Nets) in an iterative sampling scheme already provides an increase in performance, showing that the smart samples are generally improving the initial segmentations. However, the standard deviations (shown in parentheses) are relatively high. Adding in SGS-Net boosts performance while drastically lowering the variance, demonstrating the efficacy of SGS-Net in consistently filtering out bad suggestions by the SO-Nets.

4.4 Ablation Study

We aim to answer two questions with this study: 1) how good are the samples suggested by our SO-Nets, and 2) to what degree does SGS-Net increase performance and robustness? We study these questions on the larger OCID.

Since the SO-Nets alone do not provide scores of the perturbed segmentation graphs, we structure our ablation such that this is not needed in order to answer 1). Our SO-Nets are trained to provide smart perturbations that are closer to the ground truth segmentation, so every sample is supposed to be better than the original graph. With this insight, we design an experiment where we run RICE with branch factor $B = 1$ and $K = 5$ iterations, always add the candidate graph to the tree without consulting SGS-Net, and return the final graph. Essentially, this can be seen as an iterative segmentation graph refinement procedure where the sampled graph should be better than the previous in every iteration. Starting from initial masks provided by UOIS-Net-3D [5], we see in Table 1 that applying this iterative sampling scheme with SO-Nets only provides better results on almost all metrics than without. However, adding SGS-Net back into the procedure results in better Overlap F_n, Boundary F_n, and F_n@.75, while significantly reducing the standard deviation of the results by two orders of magnitude. This demonstrates that having SGS-Net in RICE delivers not only more accurate performance, but also more robust performance with relatively small variance, which answers 2). Note that F@.75 is slightly lower with F_n@.75 higher, indicating that SO-Nets are suggesting more samples that better capture the objects, but are suggesting too many instance segments.

4.5 SGS-Net Ranking

Figure 6 shows an example of how difficult scoring the segmentations graphs is; the two slightly different segmentations have a significant difference in their ground truth scores. In fact, SGS-Net does a poor job at scoring the graphs, with a mean absolute error (MAE) of 0.184 and even higher standard deviation shown in Table 2. These values are high given that the scores are in the range $[0, 1]$. Then, this begs the question, why does SGS-Net work well within our proposed RICE framework? Recall that the score magnitudes do not matter, only the relative scoring (Section 3.2). We claim that SGS-Net learns to rank the graphs accurately, and design an experiment to test this hypothesis.

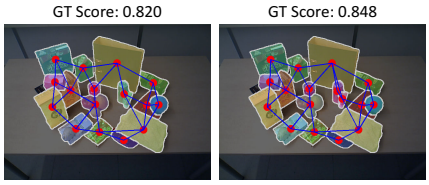


Figure 6: Can you spot the differences between the segmentations?

We leverage the normalized Discounted Cumulative Gain (nDCG) [50] which is a popular ranking metric in the information retrieval community. The DCG is computed as $\sum_{i=1}^{p} \frac{2^{\text{rel}_i}-1}{\log_2(i+1)}$ where rel_i is the numerical relevance of the item at position i (higher is better). This essentially computes a weighted sum of the relevance with a discount factor for further items, which places more emphasis on the high-ranking predictions. The normalized version divides DCG by the "ideal" version, i.e. the DCG of the correct ranking. This results in nDCG $\in [0, 1]$ with higher being better. We compute nDCG of the ranking of the iterative sampling experiment in Section 4.4, with relevance values in $\{0, ..., K\}$. The ranking for SO-Nets is given

	MAE	nDCG
Minimum	–	0.844 (0.196)
SO-Nets	–	0.944 (0.098)
SGS-Net	0.184 (0.212)	0.952 (0.095)

Table 2: Ranking study on OCID and OSD.

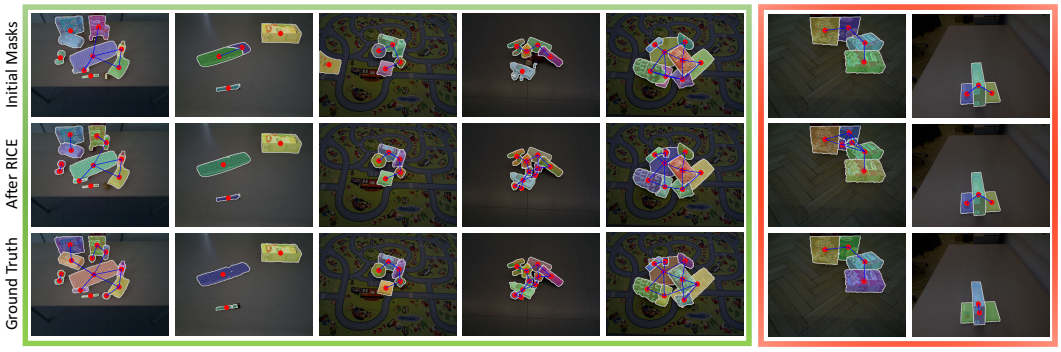


Figure 7: We demonstrate successful refinements (left, green box) for each of the sampling operations. Failure modes (right, red box) include textured objects and non-neighboring masks that belong to the same object. Best viewed in color and zoomed in on a computer screen.

by the order of the predicted graphs, and we use SGS-Net scores to compute its ranking/relevance. We also compute the nDCG of the worst ranking, denoted "minimum". In Table 2, we see that both SO-Nets and SGS-Net perform significantly better than the worst ranking. SGS-Net provides better ranking than SO-Nets with slightly lower variance, which helps to explain its effectiveness in RICE.

4.6 Visualizing Refinements

In the left side of Figure 7 (green box), we qualitatively demonstrate successful refinements from applying RICE to instance masks provided by state-of-the-art methods. The first column shows an example where many nearby objects are under-segmented. Indeed, RICE manages to find all of the necessary splits except for one. In general, RICE is quite adept at splitting under-segmented instance masks. This is quantitatively confirmed in an additional ablation in the Supplement (Section E.2) that studies the usefulness of each sampling operation. Column two shows an initial mask that is fixed with a merge operation. Column three shows a false positive mask on the textured background, which is suppressed by RICE's deletion sampling operation. In the fourth column, the initial mask is missing quite a few objects, and RICE is able to not only recover them but also correctly segment them, resulting in an almost perfect instance segmentation. In the last column, the bottom left segment is bleeding into a neighboring segment, which is fixed through multiple perturbations (i.e. split, then merge).

4.7 Failures and Limitations

In the right side of Figure 7 (red box), we discuss some failure modes and limitations. The first column demonstrates a failure mode where RICE tends to over-segment objects with a lot of texture (e.g. cereal box). We believe that this is due to TOD lacking texture on many of its objects [5]. The second column shows a limitation: since RICE only considers merging neighboring masks, it cannot merge non-neighboring masks that belong to the same object. RICE does nothing and the book is still incorrectly segmented in two pieces. We leave this as an interesting avenue for future work.

4.8 Guiding a Manipulator with Contour Uncertainties for Efficient Scene Understanding

Fully segmenting and understanding a scene of cluttered objects is necessary for various manipulation tasks, such as counting objects or re-arranging and sorting them. One way for doing this is to actively singulate each object [51]. However, such an approach can be extremely inefficient. Here we show how contour uncertainties extracted from RICE can help to solve this problem with potentially far less interactions. Specifically, we extract contour uncertainties by computing the standard deviation of the mask contours of each leaf graph. These uncertainties let us distinguish between objects that are already confidently segmented and those that require physical interaction to resolve segmentation uncertainty. We grasp [52] any object that has uncertain contours in order to determine its correct segmentation, and repeat this until no more uncertainty persists. Thus, interactions are only required to resolve the uncertain portions of the scene, which can potentially be much less than the number of

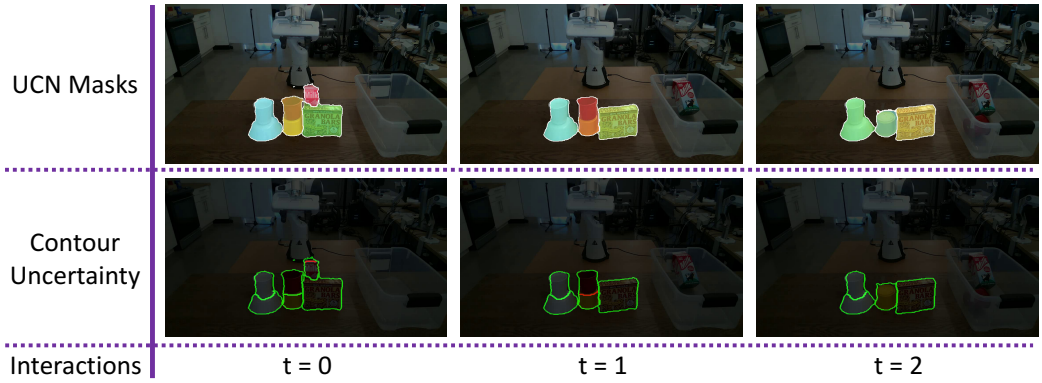


Figure 8: UCN masks [6] (top row) and contour uncertainties from RICE (bottom row, uncertainties are shown in red with average contours in green) in a trial of our scene understanding experiment. After grasping the milk carton and red cup, the scene is segmented with full certainty, indicating that the scene is fully understood. Thus, the algorithm terminates without having to singulate each object.

objects, leading to a more efficient scene understanding method. For example, in Figure 8, only two grasps are required to fully understand the scene. See the Supplemental video for more results.

5 Conclusion and Future Work

We have proposed a novel framework that utilizes a graph-based representation of instance segmentation masks. It incorporates deep networks capable of sampling smart perturbations, and a graph neural network that exploits relational inductive biases. Our experimental analysis revealed insight into why our method achieves state-of-the-art performance when combined with previous methods. We further demonstrated that our uncertainty outputs can be utilized to perform efficient scene understanding.

A main limitation of our work is the computational burden; the algorithm runs at 10-15 seconds per frame, depending on the expansion of the sample tree. Additionally, it is GPU-memory intensive as the sample tree must be stored in GPU memory. Future work will explore how to make the method more computationally efficient, along with solving the inherent limitations mentioned in Section 4.7.

References

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

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

[3] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris. Task-driven perception and manipulation for constrained placement of unknown objects. *IEEE Robotics and Automation Letters*, 2020.

[4] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox. Object rearrangement using learned implicit collision functions. *arXiv preprint arXiv:2011.10726*, 2020.

[5] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics (T-RO)*, 2021.

[6] Y. Xiang, C. Xie, A. Mousavian, and D. Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2020.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[8] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[9] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[10] V. S. Chen, P. Varma, R. Krishna, M. Bernstein, C. Re, and L. Fei-Fei. Scene graph prediction with limited labels. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[11] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

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

[13] A. Trevor, S. Gedikli, R. Rusu, and H. Christensen. Efficient organized point cloud segmentation with connected components. In *3rd Workshop on 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)*, 2014.

[15] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] A. Kirillov, Y. Wu, K. He, and R. Girshick. Pointrend: Image segmentation as rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

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

[19] D. Neven, B. De Brabandere, M. Proesmans, and L. Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi. Semi-convolutional operators for instance segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[21] L. Shao, Y. Tian, and J. Bohg. Clusternet: 3d instance segmentation in rgb-d images. *arXiv preprint arXiv:1807.08894*, 2018.

[22] S. Kong and C. Fowlkes. Recurrent pixel embedding for instance grouping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] 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 *IEEE Conference on Robotics and Automation (ICRA)*, 2019.

[24] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[25] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision (ECCV)*, 2016.

[26] W. Kuo, A. Angelova, J. Malik, and T.-Y. Lin. Shapemask: Learning to segment novel objects by refining shape priors. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[27] 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)*, 2019.

[28] A. Dave, P. Tokmakov, and D. Ramanan. Towards segmenting everything that moves. *arXiv preprint arXiv:1902.03715*, 2019.

[29] L. Shao, P. Shah, V. Dwaracherla, and J. Bohg. Motion-based object segmentation based on dense rgb-d scene flow. *IEEE Robotics and Automation Letters*, 3:3797–3804, 2018.

[30] 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.

[31] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[32] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[33] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[34] X. Liang, L. Lin, X. Shen, J. Feng, S. Yan, and E. P. Xing. Interpretable structure-evolving lstm. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[35] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[36] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[37] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29: 4502–4510, 2016.

[38] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations, (ICLR)*, 2017.

[39] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[40] 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 (MICCAI)*, 2015.

[41] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[42] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 2014.

[43] 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)*, 2016.

[44] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. Cross-modal scene networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2303–2314, 2017.

[46] 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 Computer Vision (ECCV)*, 2014.

[47] M. Suchi, T. Patten, and M. Vincze. Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets. In *IEEE Conference on Robotics and Automation (ICRA)*, 2019.

[48] 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 (IROS)*, 2012.

[49] 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, 2015.

[50] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[51] L. Chang, J. R. Smith, and D. Fox. Interactive singulation of objects from a pile. In *2012 IEEE International Conference on Robotics and Automation*, 2012.

[52] M. Sundermeyer, A. Mousavian, R. Triebel, and F. Dieter. Contact-graspnet: Efficient 6-dof grasp generation in clutteredscenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[53] Y. Wu and K. He. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018.

[54] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, (ICLR)*, 2015.

Algorithm 1 Sampling a Split

Require: Segmentation mask $S \in \{0,1\}^{h \times w}$, SplitNet output $p \in [0,1]^{h \times w}$, boundary threshold ν.

1: Compute contour of S.
2: Threshold p by ν, and compute the connected components. Create an image $\tilde{p} \in \mathbb{Z}^{h \times w}$ where $\tilde{p}_i$ is the size (in pixels) of the component at p_i for pixel i.
3: Compute contour probabilities for each contour pixel by weighted average of $\tilde{p}$ with Gaussian weights.
4: Sample start and end points on contour from contour probabilities.
5: Compute highest probability path from start to end through p, resulting in trajectory $\tau = \{(u_t, v_t)\}_{t=1}^{L}$.
6: Compute score $s_\tau = \frac{1}{L_i} \sum_t p_i[u_t, v_t]$.
7: **return** τ, s_τ

A Sampling Operation Networks details

A.1 Architecture Details

Node Encoder The Node Encoder consists of three separate CNN encoders, which consume RGB features (output by Resnet50+FPN [43, 44]), a backprojected XYZ point cloud (from a depth map and known camera intrinsics), and the mask, respectively. Each CNN encoder has 3 blocks of 2 3x3 convolutions followed by a 2x2 max pooling for resolution reduction. Lastly, there is a 7$^{\text{th}}$ convolution layer. Each convolution is immediately followed by a GroupNorm layer [53] and ReLU. The resulting features are 2x2 averaged pooled, flattened, then put through an MLP with 2 hidden layers of dimension 1024 and 512 and an output dimension of 128 (except for SplitNet, which directly consumes the fully convolutional output after the 7$^{\text{th}}$ conv layer).

SplitNet SplitNet builds off of the multi-stream CNN encoders from the Node Encoder. Given the outputs after the 7$^{\text{th}}$ conv layer of the Node Enocder, these are then concatenated and passed through another conv layer for fusion. Next, this fused output is passed to a U-Net [40] style decoder. Skip connections are fed from the multi-stream CNN encoders to the decoder. Essentially, the overall architecture (including the Node Encoder) is a multi-stream encoder-decoder UNet architecture. This is similar to Y-Net [27], except that it has three streams instead of two. Weights of the Node Encoder (multi-stream encoders) are shared with DeleteNet.

DeleteNet DeleteNet also builds off of the Node Encoder (off of the MLP outputs). In particular, for each node, it computes the difference between the Node Encoder output $\mathbf{v}_i - \mathbf{v}_{bg}$, where $\mathbf{v}_{bg}$ is the Node Encoder output of the background node. This difference is then passed through an MLP with 2 hidden layers of dimension 512 and output dimension of 1. The score is passed through a sigmoid to be in the range $[0, 1]$. Weights of the Node Encoder are shared with SplitNet.

A.2 Sampling a Split from SplitNet

We provide pseudocode of how to sample a split of mask $S \in \{0,1\}^{h \times w}$ given the output of SplitNet, which is a pixel-dense probability map $p \in [0,1]^{h \times w}$ in Algorithm 1. At a high level, we essentially compute a probability distribution over the contour of S by seeing which pixels on the contour is close to split-able boundaries given by p (more weight is given to split-able boundaries that are large components, since it is likely to find a path through that boundary). Then, start and end points are sampled and the lowest cost (where cost is 1 - p) is computed, scored and returned.

B Segmentation Graph Scoring Network Details

A high-level illustration of SGS-Net can be found in Figure 4. The initial node features $\mathbf{v}_i^{(0)}$ are given by the Node Encoder, and we obtain initial edge features $\mathbf{e}_{ij}^{(0)}$ by running the Node Encoder on all neighboring union masks S_{ij}. Then, we run multiple Residual GraphNet Layers (RGLs). Our

Residual GraphNet Layer is an adaptation of a GraphNet Layer [41] with residual connections. Our RGL first applies an edge update:

$$\mathbf{e}_{ij}^{(l+1)} = \mathbf{e}_{ij}^{(l)} + \phi_e^{(l)}\left(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}, \mathbf{e}_{ij}^{(l)}\right), \tag{1}$$

where $\phi_e^{(l)}$ is an MLP, and l describes the layer depth. This is followed by a node update:

$$\mathcal{E}_i^{(l)} = \left\{\phi_{v_1}^{(l)}\left(\mathbf{e}_{ij}^{(l+1)}, \mathbf{v}_j^{(l)}\right) : (i,j) \in E\right\} \tag{2}$$

$$\mathbf{v}_i^{(l+1)} = \mathbf{v}_i^{(l)} + \phi_{v_2}^{(l)}\left(\overline{\mathcal{E}_i^{(l)}}, \mathbf{v}_i^{(l)}\right), \tag{3}$$

where $\overline{A}$ is the mean of all elements in the set A, and $\phi_{v_1}^{(l)}, \phi_{v_2}^{(l)}$ are MLPs. We additionaly apply ReLUs after the residual connections. After passing through L levels of RGLs, we end up with the set of node and edge feature vectors $\mathcal{V} = \left\{\mathbf{v}_i^{(L)}\right\}$, $\mathcal{E} = \left\{\mathbf{e}_{ij}^{(L)}\right\}$. We pass these through an output layer that aggregates these features:

$$s_G = \sigma\left(\phi_o\left(\overline{\mathcal{V}}, \overline{\mathcal{E}}\right)\right) \in [0,1], \tag{4}$$

where ϕ_o is yet another MLP and s_G is the predicted graph score for segmentation graph G, and σ is the sigmoid function.

C Implementation Details

Our Node Encoder is shared amongst all of the networks, including SplitNet, DeleteNet, and SGS-Net. We first jointly train the SO-Nets for 200k iterations, with one segmentation graph per network per iteration (the batch sizes is the number of instances in the segmentation graph) so that the Node Encoder contains useful information for both operations. Next, we hold the Node Encoder fixed while we train SGS-Net for 100k iterations. To train SGS-Net, we take an initial segmentation and perturb it with the four proposed sampling operations and compute their ground truth scores. However, we do not use the SO-Nets, instead we randomly split masks with sampled lines, merge neighboring masks, delete masks, and add masks in the same fashion as Xie et al. [30]. Modality tuning is performed during training of the SO-Nets, and held fixed during SGS-Net training.

All images have resolution $H = 480, W = 640$. For our networks, we crop and resize the image, depth, and masks to $h = w = 64$. All training procedures use Adam [54] with an initial learning rate of 1e-4. We use $K = 3$ sample tree expansion iterations with a branching factor $B = 3$. Max nodes and edges are set to $m_n = 100, m_e = 300$ during training, and $m_n = 350, m_e = 1750$ during inference. Undirected edges are handled by including both (i,j) and (j,i) as directed edges in the graph. For each segmentation graph, edges are connected between nodes if their corresponding masks are within 10 pixels in set distance. Additionally, when sampling a candidate graph, we first randomly choose a sampling operation, compute all possible perturbation scores (e.g. split scores s_τ for each mask), and randomly select 3 of these perturbations that have a score of 0.7 or higher. This gives the opportunity to explore more segmentations within the allotted budget. All experiments are trained and evaluated on a single NVIDIA RTX2080ti GPU.

D Pseudocode for Building the Sample Tree

We provide pseudocode for building the sample tree in Algorithm 2.

E Additional Experimental Results

E.1 Modality Tuning

Inspired by Aytar et al. [45], we perform an ablation to study how to best generalize from our non-photorealistic dataset TOD to real-world data. Starting from a ResNet50 pre-trained on COCO [46], we ablate over tuning the `conv1` layer, the `conv2_1`, `conv2_2`, `conv2_3` bottleneck building blocks [43], or keeping ResNet fixed. The idea is that by fixing the rest of the layers, we can

Algorithm 2 RICE

Require: Initial instance segmentation S, RGB image I, organized point cloud D.

Build G_S with NodeEncoder applied to I, D, S
Initialize $T = \{G_S\}$
for $k \in [K]$ **do**
  for $G \in T$.leaves() **do**
    for $b \in [B]$ **do**
      Randomly choose a sampling operation and apply it to G to get candidate graph G'
      Apply SGS-Net to obtain $s_{G'}, s_G$
      if $s_{G'} > s_G$ **then**
        Add G' to T as a child of G
      end if
      if T.exceeds_budget(m_n, m_e) **then**
        Return T
      end if
    end for
  end for
end for
return Highest scoring graph in T and/or contour uncertainties

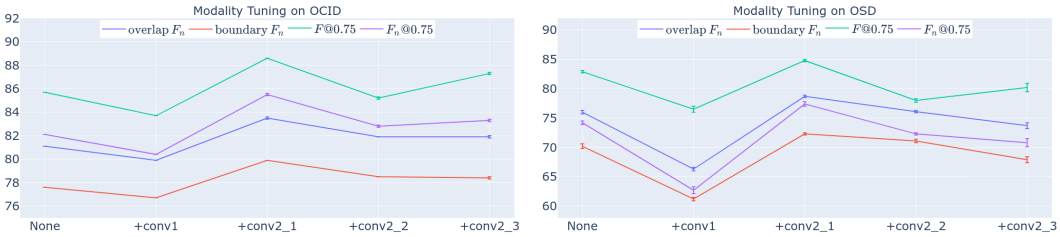


Figure 9: Modality tuning on OCID [47] and OSD [48] shows that tuning up to conv2_1 when training on simulated data generalizes best to real data. Note that standard deviation bars are shown, but are very tight and difficult to see.

encourage ResNet to learn the high level representation it has learned on COCO, on our simulated dataset. Thus, our SO-Nets and SGS-Net will learn to consume this high-level representation to provide their predictions. Then, during inference in the real-world, we resort back to the pre-trained ResNet to extract that representation from real images. In Figure 9, we show the results of our experiment. Interestingly, only modality-tuning `conv1` leads a small dip in performance compared to no tuning, while the optimal tuning for our scenario is to tune `conv1` and `conv2_1`. For the rest of this section, all networks will have been trained with this optimal setting.

E.2 Evaluating the Usefulness of Each Sampling Operation

We provide an ablation experiment where we test the efficacy of each sampling operation. In particular, we test RICE while only using one sampling operation at a time, so that the sample tree is built only by a particular operation, e.g. splitting. This allows us to determine which of the sampling operations is most helpful in comparison to the initial instance segmentation method (e.g. UOIS-Net-3D [5], UCN [6]).

We test **Split** only, **Merge** only, and **Delete/Add** only. Note that we group Delete and Add together since the Add operation is essentially the Delete operation after extracting connected components from an external foreground mask F (See Section 3.3). In Table 3, we show the results for F@.75 and F_n@.75 metrics using UCN [6], UOIS-Net-3D [5], Mask R-CNN [7], and PointGroup [8] as initial instance segmentation methods. We also show the percentage of increased performance with respect to the increased performance when using all sampling operations in parentheses. Clearly, we see that the split operation alone results in most of the performance gain compared to the full RICE method. This indicates that all four initial instance segmentation methods tend to under-segment, which is a common failure case in densely cluttered environments. UCN gains a lot from merging; the reason

Initial Instance Segmentation Method	**Split** only		**Merge** only		**Delete/Add** only	
	F@.75	F_n@0.75	F@.75	F_n@0.75	F@.75	F_n@0.75
UCN [6]	92.0 (100%)	87.2 (64.7%)	88.8 (-23.1%)	87.5 (73.5%)	89.5 (3.8%)	86.9 (55.9%)
UOIS-Net-3D [5]	88.5 (101%)	84.6 (91.6%)	78.4 (2.1%)	77.4 (4.8%)	78.3 (1.1%)	77.1 (1.2%)
Mask R-CNN [7]	79.7 (60.3%)	75.8 (82.0%)	66.0 (0.0%)	64.6 (1.4%)	69.1 (13.6%)	67.2 (20.1%)
PointGroup [8]	82.4 (86.1%)	77.4 (87.7%)	61.2 (1.6%)	60.7 (2.1%)	64.1 (13.1%)	63.1 (14.4%)

Table 3: Sampling Operation Ablation. We omit standard deviations as they are all less than 0.0005. We show results on F@.75 and F_n@.75. In parentheses, we show relative gain compared to the full RICE method (with all sampling operations).

for this is that a common failure case from their pixel-clustering procedure is that the boundaries of the objects tend to be clustered as a separate object whichh results in over-segmentation. Merging can easily solve this issue. Lastly, Mask R-CNN and PointGroup also benefit from delete/add, which suggests that they are either predicting false positives and/or false negatives.

E.3 Full Results on P/R/F

We provide full results on all Object Size Normalized (OSN) metrics for OCID [47] and OSD [48] in Figures 10a and 10b. For OCID, we can see increases in performance across all metrics in light orange. When RICE underperforms the initial segmentation, we color the bar underneath as light orange. On OSD, we can see that overlap P_n and boundary P_n are slightly worse than the initial method, while the recall R_n and F_n measures are still higher.

Figures 11a and 11b show Overlap and Boundary P/R/F measures on OCID and OSD. They show similar results to the OSN measures, but the numbers are higher. This suggests that common mistakes for SOTA methods more commonly occur in smaller objects (investigations of failure cases from Xie et al. [5], Xiang et al. [6] suggest this is the case). These numbers are directly comparable with previously published works.

F More Details for: Guiding a Manipulator with Contour Uncertainties for Efficient Scene Understanding

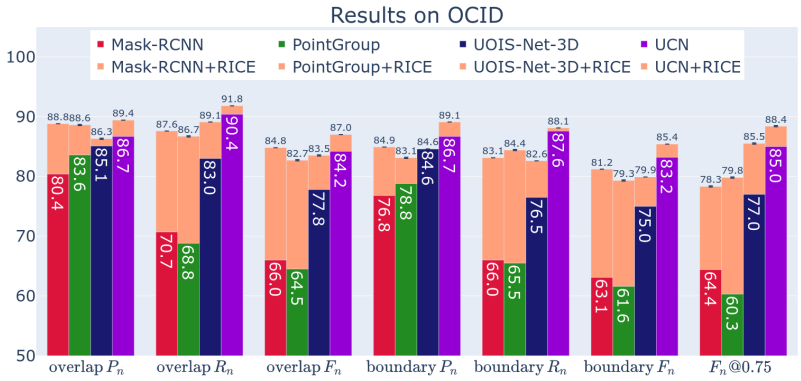
To extract contour uncertainties, we exploit the fact that RICE is a stochastic algorithm by design. Each leaf is essentially a sampled trajectory of states (segmentations) and actions (perturbations) from the initial segmentation S. These trajectories may have explored different parts of the state space (e.g. perturbed different object masks in the scene). We compute the standard deviation of the contours of each leaf graph in order to provide contour uncertainty estimates, which are shown in red in Figure 8 and the Supplemental video. Additionally, we visualize the confident contours (which are present in each leaf graph) in green.

The contour uncertainties depict certain segmentations that not all of the trajectories explored. It reflects which objects RICE is not as confident about. If a mask S_i in the initial segmentation is split the same way in all leaf graphs, then RICE is confident that S_i should be split, and there will be no uncertainty. However, if S_i is only split in some of the leaf graphs, then RICE is not as confident about whether S_i truly represents more than 1 object, and an interaction is required to resolve such uncertainty. For example, in Trial 2 in the Supplemental video, the cup and the bowl it is on top of (far left) are constantly under-segmented together by UCN [6]. RICE splits it correctly each time, and there is no uncertainty about splitting that mask, as evidenced by the uncertainty contours.

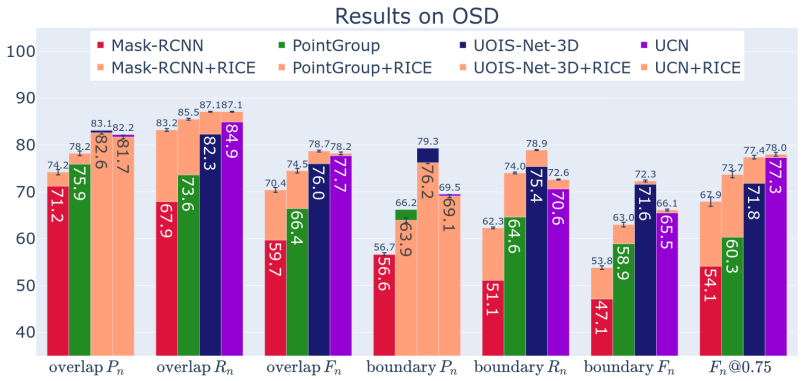
We provide a short description of the grasping algorithm with pseudocode in Algorithm 3.

G Example Sample Tree

In Figure 12, we show an example of a sample tree with branch factor $B = 2$ and $K = 2$ expansion iterations. We also visualize the ground truth score ($.8F + .2F$@.75) and the predicted score from SGS-Net. Note that the SGS-Net scores improve as the graph node gets further away from the root. In this particular example, SGS-Net would return the bottom left graph, which is also the most accurate graph.



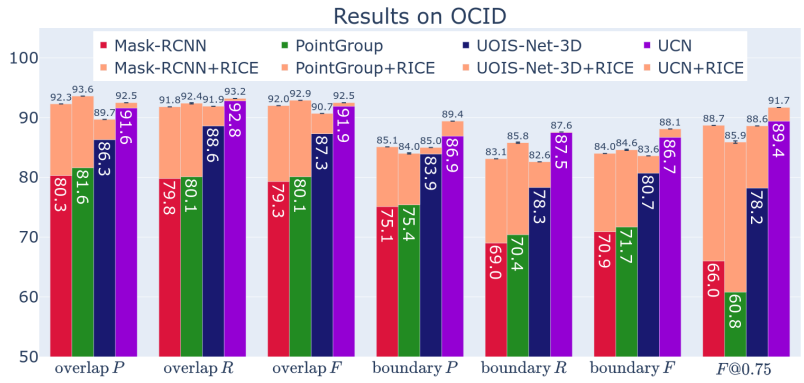
(a) Results on OCID [47] using OSN metrics.



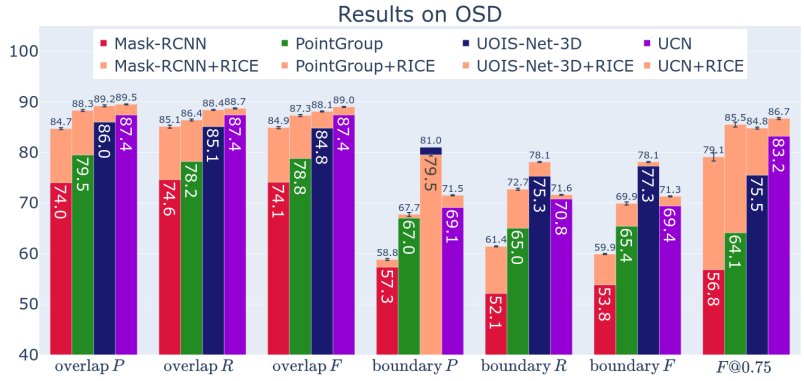
(b) Results on OSD [48] using OSN metrics.

Algorithm 3 Guiding a Manipulator with RICE

repeat
  Get S from initial instance segmentation method (e.g. UCN [6])
  Run RICE to get best masks and contour uncertainty
  if Contour Uncertainty is present **then**
    Sample a grasp with Sundermeyer et al. [52] from the uncertain masks, and execute it
  end if
until No Contour Uncertainty



(a) Results on OCID [47] using Overlap and Boundary P/R/F metrics as defined by Xie et al. [30]. Note that the UCN [6] numbers are slightly different than in the published paper, due to the authors finding a bug in the reporting code. This is the case for Xie et al. [5] as well.



(b) Results on OSD [48] using Overlap and Boundary P/R/F metrics as defined by Xie et al. [30].

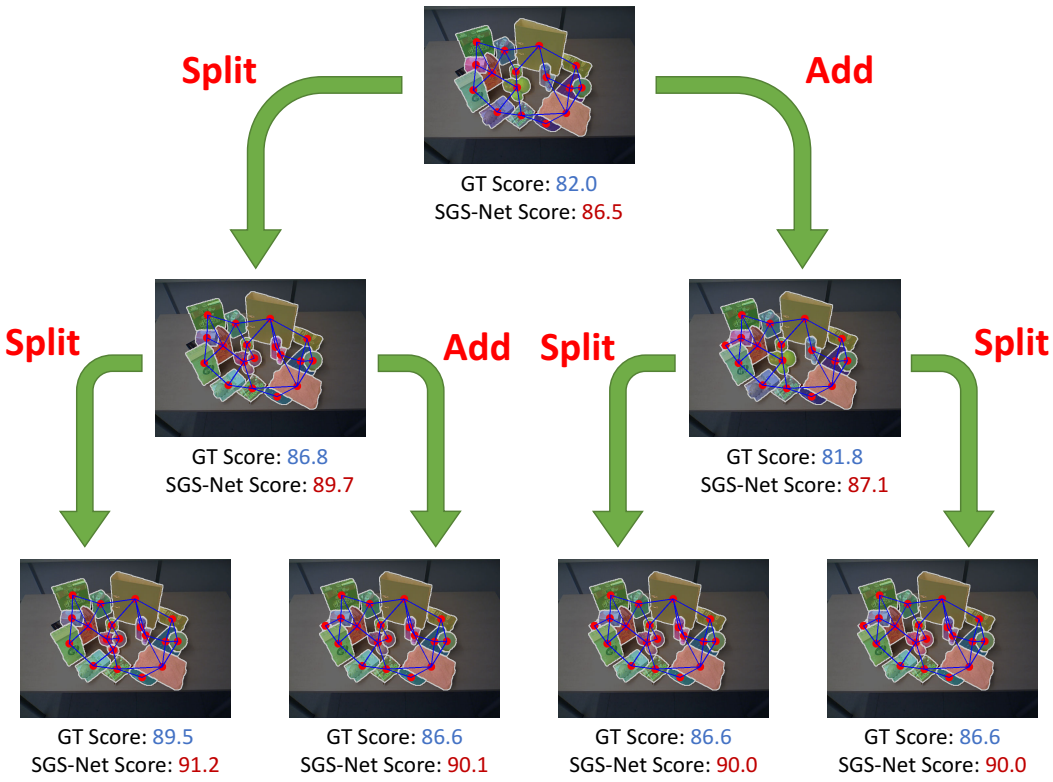


Figure 12: Example of a sample tree. Ground Truth and SGS-Net scores are shown, along with the chosen sampling operations. In this example, all leaves improve upon the initial segmentation graph, with the highest ranking graph also being the closest to the ground truth segmentation. Very similar splits and adds are investigated in the leaf trajectories.