Discovering and Segmenting Unseen Objects for Robot Perception

Christopher Xie

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2021

Reading Committee:
Dieter Fox, Chair
Siddhartha Srinivasa
Ali Farhadi

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Discovering and Segmenting Unseen Objects for Robot Perception

Christopher Xie

Chair of the Supervisory Committee:

Professor Dieter Fox
Computer Science and Engineering

Perception lies at the core of the ability of a robot to function in the real world. As robots become more ubiquitously deployed in unstructured environments such as homes and offices, it is inevitable that robots will encounter objects that they have not observed before. Thus, in order to interact effectively with such environments, building a robust object recognition module of unseen objects is valuable. Additionally, it can facilitate downstream tasks including grasping, re-arrangement, and sorting of unseen objects. This is a challenging perception task since the robot needs to learn the concept of "objects" and generalize it to unseen objects.

In this thesis, we propose different methods for learning such perception systems by exploiting different visual cues and learning data without manual annotations. First, we investigate the use of motion cues for this problem. We develop a novel neural network architecture, PT-RNN, that leverages

optical flow by casting the problem as object discovery via foreground motion clustering from videos. This network learns to produce pixel-trajectory embeddings such that clustering them results in segmenting the unseen objects into different instance masks. Next, we introduce UOIS-Net, which separately leverages RGB and depth for unseen object instance segmentation. UOIS-Net is able to learn from synthetic RGB-D data where the RGB is non-photorealistic, and provides state-of-the-art unseen object instance segmentation results in tabletop environments, which are common to robot manipulation. Lastly, we investigate the use of relational inductive biases in the form of graph neural networks in order to better segment unseen object instances. We introduce a novel framework, RICE, that refines a provided instance segmentation by utilizing a graph-based representation.

We conclude with a discussion of the proposed work and future directions, which includes a vision of future research that leverages the proposed work to bootstrap a lifelong learning mechanism that renders unseen objects as no longer unseen.

Acknowledgements

First, I would like to thank my advisor Dieter Fox for his support, advice and wisdom throughout my PhD. He graciously accepted me into his group during my fourth year, and since then, I have learned a great deal about how to conduct research. His ability to keep up-to-date in the robotics and computer vision subfields is incredibly impressive. His creativity is remarkable, which helped me navigate through the many roadblocks I encountered in my research. Perhaps most importantly, he has shown me how to ask deep, meaningful questions when digging into research problems. I would also like to thank my committee members including Siddhartha Srinivasa, who I co-authored a paper with, Ali Farhadi, and Ming-Ting Sun.

I would also like to thank the numerous mentors I have been lucky to have throughout my graduate school journey. In particular, I would like to thank Emily Fox for showing me how to effectively communicate my ideas to other researchers, and Zaid Harchaoui for providing many great nuggets of advice and introducing me to the field of computer vision. Thanks to Avleen Bijral, Matthew Brown, and Ricardo Martin-Brualla for

their mentorship during my internships at Microsoft and Google. Finally, a special thanks to Yu Xiang and Arsalan Mousavian for their long-term collaboration and mentorship that spanned many projects within this thesis.

Next, I would like to thank the collaborators I have worked with at UW, including William Agnew, Keunhong Park, and Alex Tank, and the members of the UW Robotics and State Estimation (RSE) Lab. They were the source of many great conversations and collaborations, and I learned a lot from our shared experiences.

I would like to thank friends and family, who provided an escape from the stress and pressure of a PhD program. Thanks to (listed in no particular order): Max, Rahul, Maarten, Meldrew, Mondrew, Carrison, YanMi, I-Lyds, Rohin, Lanssie, Jeannie, Jason, Justin, Phil, among many others. A big thanks to my mom, Su Li, and dad, ZuQi Xie, for their unwavering support in my pursuit of higher education, and my brother, Walt, for always being the best big brother one can ask for (and helping me land my post-graduation job!).

Last but definitely not least, none of this would be possible without the support of my amazing wife Amanda. She cheered me on during the good times and kept me sane and grounded through the rough patches. Her energy and fervor for bettering herself is a limitless source of motivation and inspiration for me. It was this source that enabled me to grow and accomplish my goals these past six years.

DEDICATION

To my amazing, supportive and beautiful wife, Amanda, who never ceases to inspire me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A future where robots are ubiquitously deployed around the world is not too far off. The International Federation of Robotics World Robotics Report in 2020 reports that a record high of 2.7 million robots are currently operating in factories around the world. In the United States alone, industrial robots grew fourfold between 1993 and 2007 [1]. This concept was predicted as far back as 1930, when John Maynard Keynes famously predicted that the rapid spread of technology will result in "technological unemployment" [80].

Robots have been flourishing in factory environments. The automotive industry employs 38% of existing robots, followed by the electronics industry (15%), plastics and chemicals (10%), and metal products (7%) [1]. These environments are typically very well structured, where objects to be manipulated (e.g. product parts such as a car door for a car assembly robot) are exactly where they are expected to be.

Robot automation in homes and offices can provide a number of advantages for the world. They can significantly ease the burden of performing mundane tasks such as making coffee, or cleaning rooms. They can provide safety by largely reducing human error in driving by replacing human drivers with autonomous vehicles. Robots can additionally be beneficial in caring for the elderly, and providing services at hospitals such as transporting medications from floor to floor or providing surgeons with enhanced control and vision, such as Intuitive Surgical's da Vinci robot. This then begs

the question: *why aren't we seeing more robots operating in unstructured spaces such as homes and/or offices?*

Let's consider a task in a home such as cleaning up a room full of toys (putting them back where they belong). What skills does a robot need in order to successfully perform this task? One may correctly identify that manipulation skills are required to solve this task. While this is true, the precursor to manipulation involves perceiving and reasoning spatially about the objects of interest (toys on the floor). We argue that the basic robot perception technologies that are required to perform this task are still lacking.

A natural thought is to apply solutions researched in the computer vision domain for this problem. While computer vision originally was meant to be a stepping stone to endow robots with visual capabilities, the current research questions they focus on today typically do not address the specific problems that arise when attempting to deploy robots in unstructured environments [154]. One such problem is the recognition of **unseen objects**. Most object detection algorithms in the computer vision domain tend to focus on detecting objects of known classes (e.g. pedestrians, cars, and bikers). However, in the above example task of cleaning a room full of toys, it is likely that the robot may run into types of toys it has never encountered before, either in a training phase or a previous interaction. A failure to recognize the toy can lead to it being left on the floor, which could be a potential danger when kids are running around the room. In fact, it is inevitable that robots will encounter unseen objects as they continue to be deployed in unstructured environments. Thus, robots will need perceptual systems that can safely reason about such objects in order to facilitate environmental interactions in unstructured spaces such as homes and offices. We focus our work onto this specific problem, which is to say that we attempt to answer the question: **"How can we imbue robots with perception systems that can reason about unseen objects?"**

This thesis starts off by laying out the challenges that emerge when building systems that can perceive and reason about unseen objects. First, we further detail current computer vision technologies and why they cannot

be directly applied. We then discuss how we can leverage the ever growing amount of repositories of 3D data to generate datasets of which we can use to learn our algorithms. It will then develop potential solutions to this problem by exploiting different visual cues and clever designs to take advantage of synthetic data.

1.1 Computer Vision vs. Robot Vision in Today's World

In the 1960s, computer vision was originally meant to be a stepping stone to endow robots with visual competence [156]. Marvin Minsky famously asked his undergraduate student Gerald Sussman to "spend the summer linking a camera to a computer and getting the computer to describe what it saw". However, the problem was infinitely harder than imagined and has been the center of much research effort to this day.

Current computer vision research touches upon many subfields including object detection, tracking, and 3D reconstruction. The research questions typically addressed in these areas are mainly concerned with "in-the-wild" images, a term that encapsulates the distribution of images found on the internet. For example, social networking sites such as Facebook or Instagram possess large amounts of photos captured by humans, and it is very much of interest to be able to detect people in order to build interesting products that better serve their customers (e.g. automated tagging of individuals). Thus a desirable quality of an object detection algorithm is to be able to adequately perform in such "in-the-wild" settings.

Because "in-the-wild" images are typically captured by humans, there is a bias to certain object categories that are frequently present in such images such as other humans, or pets such as dogs. To build computer vision systems that perform reliably in such settings, large amounts of "in-the-wild" training data is collected and labeled by humans. This leads to further bias as the classes chosen for manual labeling are typically a small subset of the objects present in the images. These labels include objects that are of interest to humans, e.g. other humans, animals, and vehicles such as cars or trucks. In fact, many of the large-scale datasets that are used for evaluation

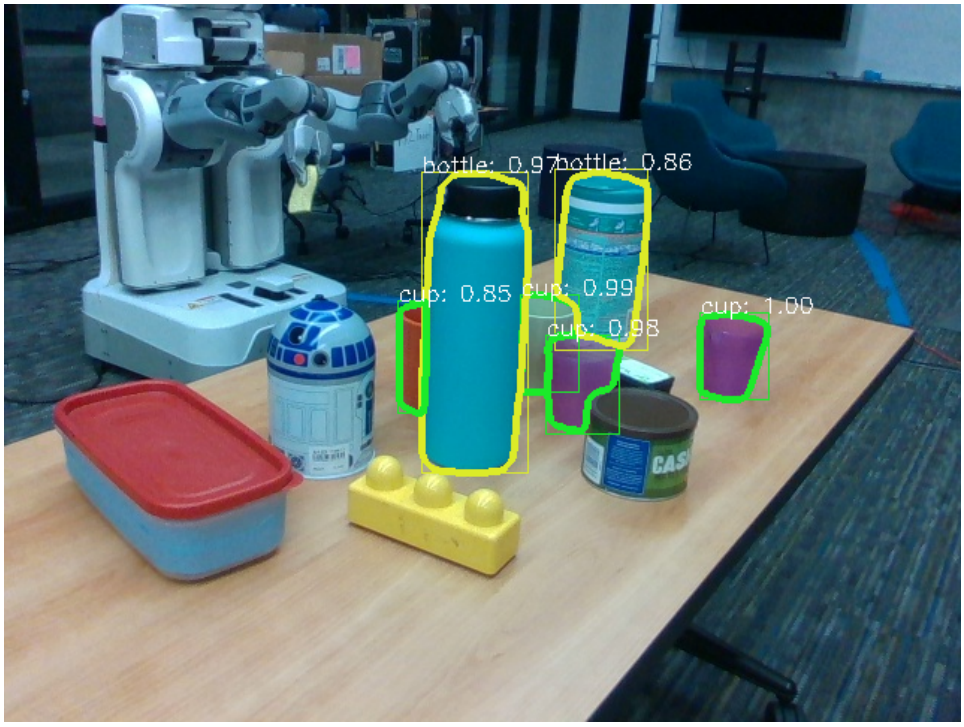


Figure 1.1: Applying a state-of-the-art pretrained object detector (Mask R-CNN [63]) to an image with objects not belonging to a pre-defined set of categories results in missed detections. The objects on the table are of interest for the robot to manipulate, yet the lego block, R2D2 can, and the tupperware are among many objects that are not detected. This demonstrates the clear need to develop solutions to recognize such *unseen objects*.

such as ImageNet (image recognition) [43] and COCO (object detection) [99] report results on a finite number of categories to detect. This leads a lot of computer vision research to focus only on these categories, and popular methods for solving these problems (e.g. Mask RCNN [133, 63]) tend to contain specific solutions for dealing with finite categories. Objects that do not belong to this pre-defined set of categories are typically ignored as shown in Figure 1.1. Neglecting the ignored categories is generally fine in many computer vision applications such as image tagging or image content manipulation (think automatic Adobe photoshop).

Robot vision, on the other hand, tends to come across a different set of challenges that are not typically addressed by the computer vision community [154], especially when compared to the "in-the-wild" settings. Robots do not just perceive their surroundings, but also take actions to either move through the environment or induce changes to it through manipulation. Thus, visual perception is only one part of the complex, embodied system that robots encompass [154]. This leads to problems including visual navigation [18], simultaneous localization and mapping (SLAM), and (inter)active perception [17], each of which has its own research communities.

In unstructured, ever-changing environments, robots will inevitably encounter instances of object classes that were not seen in a training phase nor via a previous interaction. For example, imagine a robot navigating through a home. During its navigation trajectory, it may roam through rooms full of toys, underneath tables, and potentially through hard-to-reach places. In each of these locations it may observe objects such as scattered toys, outlets, and objects lost for long periods of time (e.g. that TV remote you couldn't find for years) that do not register as objects from a known set of classes (e.g. prior information distilled form a dataset). Such "open-set" conditions [11, 140], where objects of unknown class are regularly encountered, should be expected as robots are deployed in these variable environments.

Because the ultimate goal of robot perception is to inform a choice of action in a real-world environment, it is important that unseen objects are detected and recognized, not just objects of known classes. If robots operate as if these objects are not present, unfortunate consequences can occur. Toys may be left unattended which can be dangerous for kids running around, or a manipulation trajectory may collide with the ignored objects. Even worse, self-driving cars can crash into rare or unusual-looking animals/objects that are not identified as any instance of a pre-defined set of object classes.

Additionally, some notion of uncertainty in these detections would also serve useful in a robot selecting its actions. Uncertainty can be used in fusing information together over time (e.g. Kalman filters [75]), or allow a framework such as active learning [143] or information gain [149] to select

actions that minimize the uncertainty. Thus, the question of how to obtain uncertainty estimates of unseen objects and resolve these uncertainties through interaction is an important problem to consider in getting a fully autonomous robot to function in unstructured environments.

1.2 Leveraging Large-Scale Data without Manual Annotation

The recent boom in deep learning [88] has demonstrated that training deep networks with large amounts of data can lead to very accurate prediction models. Deep networks now dominate the majority of computer vision subtasks including image recognition [88], object detection/segmentation [63, 28], and 3D reconstruction [107]. We build off of such advances and also learn deep networks in order to solve our problem of detecting and segmenting unseen objects.

In order to ensure the generalization capability of a deep network to recognize unseen objects, we need to learn from data that contains large amounts of various objects in the environments of interest (e.g. tabletop environments). However, large-scale datasets with this property currently do not exist. We could follow the paradigm of computer vision research and collect a large real-world dataset with manual annotations such as ImageNet [43] and COCO [99]. Unfortunately, this requires a substantial amount of time and money in order to collect the images and pay humans to annotate the collected images. Additionally, robots are intended to encounter many different environments (e.g. homes, offices, disaster recovery, just to name a few) which would require separately collected large-scale real-world datasets for each of them. Clearly, this approach will not scale with the amount of environments robots encounter.

Thus, it is appealing to utilize synthetic data for training, such as using the ShapeNet repository which contains thousands of 3D objects [24]. While collecting 3D models for ShapeNet took a lot of effort, using it to generate a large-scale synthetic dataset allows us to sidestep the image collection

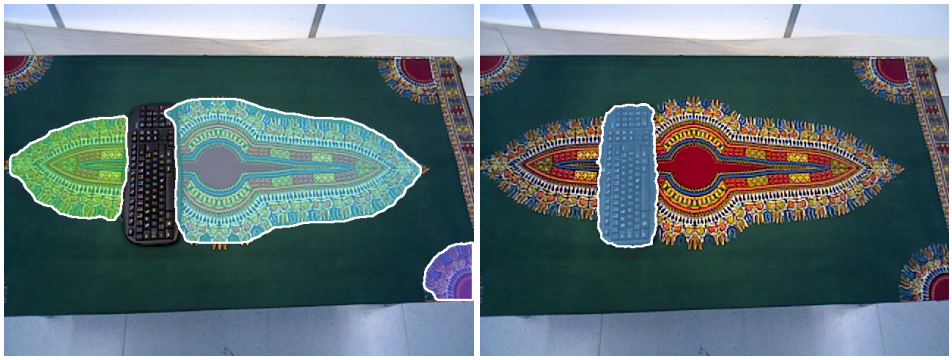


Figure 1.2: Mask R-CNN [63] trained on a synthetic dataset of tabletop objects (introduced in Chapter 3) evaluated on a real-world image *(left)* of tabletop objects [152]. The detections not only miss the single object of interest (keyboard), but they also misfire on table textures. This demonstrates the "Sim-to-Real" gap. The ground truth segmentation masks *(right)* show only one object of interest on the table, a keyboard.

and manual annotation steps, which reduces the time and cost to next to nothing. We obtain detection and segmentation annotations for free, which are expensive to label with humans. Additionally, we have full control over the object poses, which affords us more control than over the traditional image collection process in "in-the-wild" settings. However, there exists a domain gap between synthetic data and real-world data as many simulators do not provide realistic-looking images. Training directly on such synthetic data only usually does not work well in the real world [183]. In Figure 1.2, we trained Mask R-CNN on a synthetic dataset of tabletop objects (introduced in Chapter 3) and evaluated it on a real-world tabletop image where it fails to detect the single object on the table. Additionally, synthesizing photo-realistic images with physics-based rendering can be computationally expensive [64], making large photorealistic synthetic datasets impractical to obtain.

Consequently, recent efforts in robot perception have been devoted to the problem of *Sim-to-Real*, where the goal is to transfer capabilities learned in simulation to real-world settings. For instance, some works have used domain adaptation techniques to bridge the gap when unlabeled real data is

available [163, 19]. Domain randomization [159] was proposed to diversify the rendering of synthetic data for training. While these techniques attempt to fix the discrepancy between synthetic and real-world RGB, models trained with synthetic depth have been shown to generalize reasonably well for simple settings such as bin-picking [103, 37]. However, in more complex settings, noisy depth sensors can limit the application of such methods.

The above solutions can be described as augmenting the training process of deep networks to handle the irregularities of synthetic data with respect to real-world data (i.e. in order to handle the sim-to-real gap). Domain randomization assumes control over the data generation during training, and domain adaptation introduces extra network architectural components along with real-world data to the training pipeline. Instead, an interesting question to explore is whether we can leverage standard training pipelines of deep networks (e.g. stochastic gradient descent with standard loss functions such as binary cross entropy) while designing the network structure to better handle synthetic data. This would allow for a more transparent approach: simply choose the network architecture design and train it with standard loss functions and only the synthetic data. The requirements of control over the data generation or access to real-world data would be relieved. We pose this question here as an interesting thought that serves as additional motivation for the work in this thesis.

1.3 Dissertation Overview

This dissertation explores multiple directions of research of solutions to the problem of detecting and segmenting unseen objects for robot perception. In particular, we explore methods that rely heavily on the nuances of robot environments and/or large-scale synthetic data.

We begin with Chapter 2, where we investigate the use of motion cues extracted from video as the primary source of information. We formulate the problem as foreground motion clustering, where the goal is to cluster foreground pixels in videos into different objects. We propose a network architecture, PT-RNN, that is able to segment the objects consistently in time

by design, and demonstrate its ability to discover novel unseen objects from videos. We leverage a large-scale synthetic dataset of flying chairs in order to pre-train PT-RNN.

We then introduce UOIS-Net in Chapter 3, a solution that utilizes geometry cues from a single image to segment unseen object instances. UOIS-Net is designed to leverage the strengths of depth and RGB separately in order to perform the segmentation. This two-stage network extracts initial instance segmentation masks from depth alone which generalizes quite well from synthetic to real-world settings, and then utilizes RGB to sharpen the initial masks. The clever design of UOIS-Net allows for it to be trained purely on a non-photorealistic synthetic dataset that we generated, yet generalize to the real-world quite well without any fine-tuning. This work has enabled further downstream robotics tasks of unseen objects such as grasping and re-arrangement.

In Chapter 4, we propose RICE, a method for refining a set of input instance segmentation masks by utilizing a relational neural network architecture. We introduce a new graph-based representation of instance segmentation masks. RICE uses this representation to sample perturbations to the original set of masks, and uses a graph neural network to evaluate whether the perturbations are better. We show improved results when combined with previous works that directly predict instance masks. Additionally, we show that RICE can generate uncertainty at the instance segmentation level and use it to demonstrate an efficient scene understanding application.

Finally, we conclude this thesis with a conclusion and discussion of future directions of this research in Chapter 5.

Chapter 2

Object Discovery in Videos as Foreground Motion Clustering

This chapter discusses work originally published in Xie et al. [174].

Discovering objects from videos is an important capability that an intelligent system needs to have. Imagine deploying a robot to a new environment. If the robot can discover and recognize unseen objects in the environment by observing, it would enable the robot to better understand its work space. In the interactive perception setting [17], the robot can even interact with the environment to discover objects by touching or pushing objects. To define an "object", in this chapter we consider an entity that can move or be moved to be an object, which includes various rigid, deformable and articulated objects. We utilize motion and appearance cues to discover objects in videos.

Motion-based video understanding has been studied in computer vision for decades. In low-level vision, different methods have been proposed to find correspondences between pixels across video frames, which is known as optical flow estimation [65, 8]. Both camera motion and object motion can result in optical flow. Since the correspondences are estimated at a pixel level, these methods are not aware of the objects in the scene, in the sense that they do not know which pixels belong to which objects. In high-level vision, object detection and object tracking in videos has been well-studied [5, 74, 61, 184, 13, 171]. These methods train models for specific object

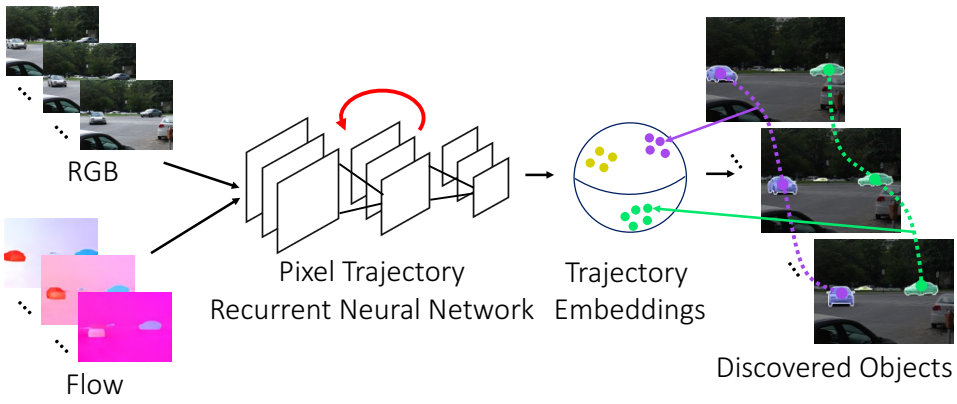


Figure 2.1: Overview of PT-RNN. RGB images and optical flow are fed into a recurrent neural network, which computes embeddings of pixel trajectories. These embeddings are clustered into different foreground objects.

categories using annotated data. As a result, they are not able to detect nor track unseen objects that have not been seen in the training data. In other words, these methods cannot discover new objects from videos. In contrast, motion segmentation methods [20, 78, 14, 118] aim at segmenting moving objects in videos, which can be utilized to discover unseen objects based on their motion.

In this chapter, we formulate the unseen object instance segmentation problem as foreground motion clustering, where the goal is to cluster pixels in a video into different objects based on their motion. There are two main challenges in tackling this problem. First, how can foreground objects be differentiated from background? Based on the assumption that moving foreground objects have different motion as the background, we design a novel encoder-decoder network that takes video frames and optical flow as inputs and learns a feature embedding for each pixel, where these feature embeddings are used in the network to classify pixels into foreground or background. Compared to traditional foreground/background segmentation methods [33, 68], our network automatically learns a powerful feature representation that combines appearance and motion cues from images.

Secondly, how can we consistently segment foreground objects across

video frames? We would like to segment individual objects in each video frame and establish correspondences of the same object across video frames. Inspired by [20] that clusters pixel trajectories across video frames for object segmentation, we propose to learn feature embeddings of pixel trajectories with a novel Recurrent Neural Network (RNN), and then cluster these pixel trajectories with the learned feature embeddings. Since the pixel trajectories are linked in time, our method automatically establishes the object correspondences across video frames by clustering the trajectories. Different from [20] that employs hand-crafted features to cluster pixel trajectories, our method automatically learns a feature representation of the trajectories, where the RNN controls how to combine pixel features along a trajectory to obtain the trajectory features. Figure 2.1 illustrates our framework for object motion clustering.

Since our problem formulation aims to discover objects based on motion, we conduct experiments on five motion segmentation datasets to evaluate our method: Flying Things 3D [105], DAVIS [120, 125], Freiburg-Berkeley motion segmentation [115], ComplexBackground [111] and CamouflagedAnimal [15]. We show that our method is able to segment potentially unseen foreground objects in the test videos and consistently across video frames. Comparison with the state-of-the-art motion segmentation methods demonstrates the effectiveness of our learned trajectory embeddings for object discovery. In summary, this chapter has the following key contributions:

- We introduce a novel encoder-decoder network to learn feature embeddings of pixels in videos that combines appearance and motion cues.
- We introduce a novel recurrent neural network to learn feature embeddings of pixel trajectories in videos.
- We use foreground masks as an attention mechanism to focus on clustering of relevant pixel trajectories for object discovery.
- We achieve state-of-the-art performance on commonly used motion

segmentation datasets.

2.1 Related Work

Video Foreground Segmentation. Video foreground segmentation is the task of classifying every pixel in a video as foreground or background. This has been well-studied in the context of video object segmentation [15, 116, 161, 161, 70], especially with the introduction of unsupervised challenge of the DAVIS dataset [120]. [15] uses a probabilistic model that acts upon optical flow to estimate moving objects. [116] predicts video foreground by iteratively refining motion boundaries while encouraging spatio-temporal smoothness. [161, 160, 70] adopt a learning-based approach and train Convolutional Neural Networks (CNN) that utilize RGB and optical flow as inputs to produce foreground segmentations. Our approach builds on these ideas and uses the foreground segmentation as an attention mechanism for pixel trajectory clustering.

Instance Segmentation. Instance segmentation algorithms segment individual object instances in images. Many instance segmentation approaches have adopted the general idea of combining segmentation with object proposals [63, 123]. While these approaches only work for objects that have been seen in a training set, we make no such assumption as our intent is to discover objects. Recently, a few works have investigated the instance segmentation problem as a pixel-wise labeling problem by learning pixel embeddings [42, 114, 85, 49]. [114] predicts pixel-wise features using translation-variant semi-convolutional operators. [49] learns pixel embeddings with seediness scores that are used to compose instance masks. [42] designs a contrastive loss and [85] unrolls mean shift clustering as a neural network to learn pixel embeddings. We leverage these ideas to design our approach of learning embeddings of pixel trajectories.

Motion Segmentation. Pixel trajectories for motion analysis were first introduced by [153]. [20] used them in a spectral clustering method to produce motion segments. [115] provided a variational minimization to produce pixel-wise motion segmentations from trajectories. Other works that build

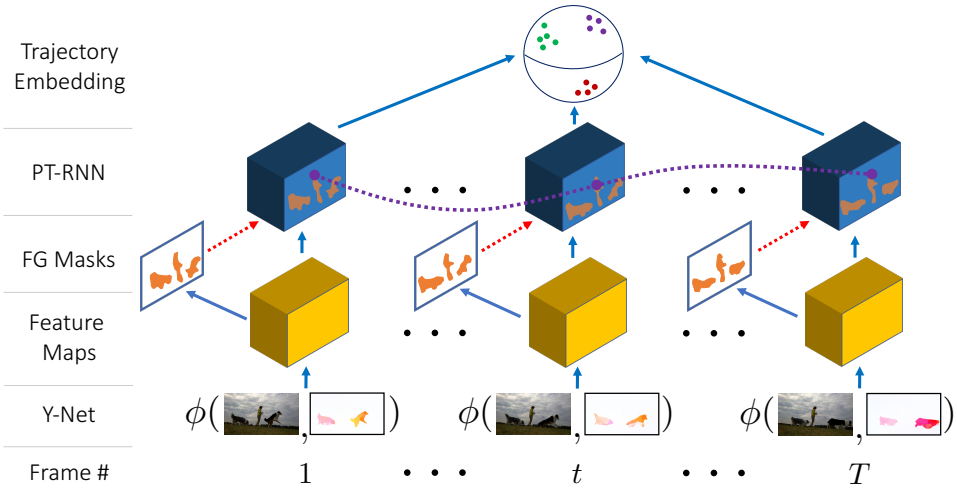
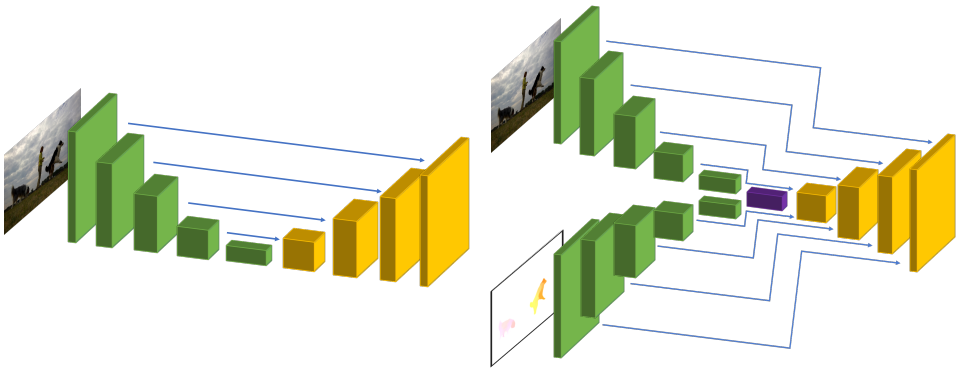


Figure 2.2: Overview of PT-RNN architecture. First, feature maps of each frame are extracted from the Y-Net. Next, foreground masks are computed, shown in orange. The PT-RNN uses these foreground masks to compute trajectory embeddings (example foreground trajectory from frame 1 to T shown in purple), which are normalized to produce unit vectors. Backpropagation passes through the blue solid arrows, but not through the red dashed arrows.

off this idea include formulating trajectory clustering as a multi-cut problem [77, 78, 79] or as a density peaks clustering [166], and detecting discontinuities in the trajectory spectral embedding [53]. More recent approaches include using occlusion relations to produce layered segmentations [157], combining piecewise rigid motions with pre-trained CNNs to merge the rigid motions into objects [16], and jointly estimating scene flow and motion segmentations [144]. We use pixel trajectories in a recurrent neural network to learn trajectory embeddings for motion clustering.

2.2 Method

Our approach takes video frames and optical flow between pairs of frames as inputs, which are fed through an encoder-decoder network, resulting in



(a) U-Net architecture (b) Y-Net architecture

Figure 2.3: We show U-Net [136] and our proposed Y-Net to visually demonstrate the difference. Y-Net has two encoding branches (shown in green) for each input modality, which is fused (shown in purple) and passed to the decoder (shown in yellow). Skip connections are visualized as blue arrows.

pixel-wise features. These features are used to predict foreground masks of moving objects. In addition, a recurrent neural network is designed to learn feature embeddings of pixel trajectories inside the foreground masks. Lastly, the trajectory embeddings are clustered into different objects, giving a consistent segmentation mask for each discovered object. The network architecture is visualized in Figure 2.2.

2.2.1 Encoder-Decoder: Y-Net

Let $I_t \in \mathbb{R}^{H \times W \times 3}, F_t \in \mathbb{R}^{H \times W \times 2}$ be an RGB image and forward optical flow image at time t, respectively. Our network receives these images from a video as inputs and feeds them into an encoder-decoder network separately at each time step, where the encoder-decoder network extracts dense features for each video frame. Our encoder-decoder network is an extension of the U-Net architecture [136] (Figure 2.3a) to two different input types, i.e., RGB images and optical flow images, by adding an extra input branch. We denote this mid-level fusion of low-resolution features as Y-Net. We illustrate the Y-Net architecture in Figure 2.3b.

In detail, our network has two parallel encoder branches for the RGB

and optical flow inputs. Each encoder branch consists of four blocks of two 3×3 convolutions (each of which is succeeded by a GroupNorm layer [169] and ReLU activation) followed by a 2×2 max pooling layer. The encodings of the RGB and optical flow branches are then concatenated and input to a decoder network, which consists of a similar architecture to [136] with skip connections from both encoder branches to the decoder.

We argue that this mid-level fusion performs better than early fusion and late fusion (using completely separate branches for RGB and optical flow, similar to two-stream networks [148, 50, 160]) of encoder-decoder networks while utilizing less parameters, and show this empirically in Section 2.3.1. The output of Y-Net, $\phi(I_t, F_t) \in \mathbb{R}^{H \times W \times C}$, is a pixel-dense feature representation of the scene. We will refer to this as pixel embeddings of the video.

2.2.2 Foreground Prediction

The Y-Net extracts a dense feature map for each video frame that combines appearance and motion information of the objects. Using these features, our network predicts a foreground mask for each video frame by simply applying another convolution on top of the Y-Net outputs to compute foreground logits. These logits are passed through a sigmoid layer and thresholded at 0.5. For the rest of the paper, we will denote m_t to be the binary foreground mask at time t.

The foreground masks are used as an attention mechanism to focus on the clustering of the trajectory embeddings. This results in more stable performance, as seen in Section 2.3.1. Note that while we focus on moving objects in our work, the foreground can be specified depending on the problem. For example, if we specify that certain objects such as cars should be foreground, then we would learn a network that learns to discover and segment car instances in videos.

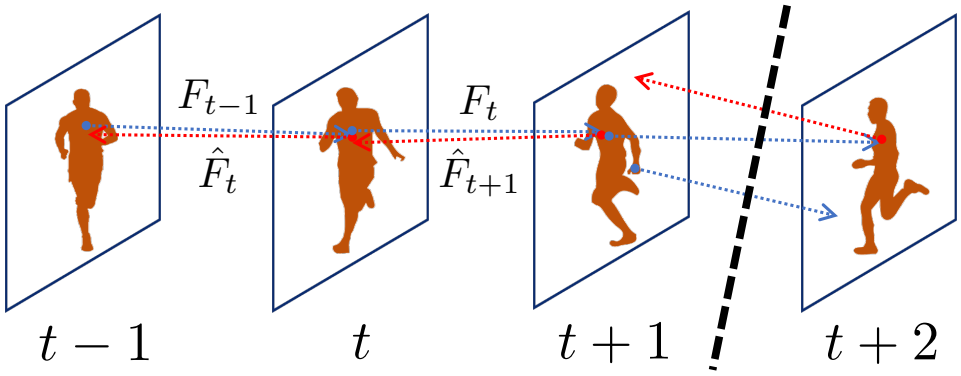


Figure 2.4: We illustrate pixel linking in foreground pixel trajectories. The foreground mask is shown in orange, forward flow is denoted by the blue dashed arrow, and backward flow is denoted by the red dashed arrow. The figure shows a trajectory that links pixels in frames $t-1, t, t+1$. Two failure cases that can cause a trajectory to end are shown between frames $t+1$ and $t+2$: 1) Eq. (2.1) is not satisfied, and 2) one of the pixels is not classified as foreground.

2.2.3 Trajectory Embeddings

In order to consistently discover and segment objects across video frames, we propose to learn deep representations of *foreground pixel trajectories* of the video. Specifically, we consider dense pixel trajectories throughout the videos, where trajectories are defined as in [153, 20]. Given the outputs of Y-Net, we compute the trajectory embedding as a weighted sum of the pixel embeddings along the trajectory.

Linking Foreground Trajectories

We first describe the method to calculate pixel trajectories according to [153]. Denote $F_{t-1} \in \mathbb{R}^{H \times W \times 2}$ to be the forward optical flow field at time $t-1$ and $\hat{F}_t \in \mathbb{R}^{H \times W \times 2}$ to be the backward optical flow field at time t. As defined in [153], we say the optical flow for two pixels (i, j) at time $t-1$ and (i', j') at

time t is consistent if

$$\left\| F_{t-1}^{i,j} + \hat{F}_t^{i',j'} \right\|^2 \leq 0.01 \left(\left\| F_{t-1}^{i,j} \right\|^2 + \left\| \hat{F}_t^{i',j'} \right\|^2 \right) + 0.5, \tag{2.1}$$

where $F_{t-1}^{i,j}$ denotes the i,j-th element of F_{t-1}. Essentially, this condition requires that the backward flow points in the inverse direction of the forward flow, up to a tolerance interval that is linear in the magnitude of the flow. Pixels (i,j) and (i',j') are linked in a pixel trajectory if Eq. (2.1) holds.

To define *foreground pixel trajectories*, we augment the above definition and say pixels (i,j) and (i',j') are linked if Eq. (2.1) holds and both pixels are classified as foreground. Using this, we define a foreground-consistent warping function $g : \mathbb{R}^{H\times W} \to \mathbb{R}^{H\times W}$ that warps a set of pixels $\mathbf{v} \in \mathbb{R}^{H\times W}$ forward in time along their foreground trajectories:

$$g(\mathbf{v})^{i',j'} = \begin{cases} \mathbf{v}^{i,j} & \text{if } (i,j), (i',j') \text{ linked} \\ 0 & \text{otherwise.} \end{cases}$$

This can be achieved by warping $\mathbf{v}$ with $\hat{F}_t$ with bilinear interpolation and multiplying by a binary consistency mask. This mask can be obtained by warping the foreground mask m_{t-1} with $\hat{F}_t$ using Eq. (2.1) and intersecting it with m_t, resulting in a mask that is 1 if (i',j') is linked to a foreground pixel at time $t-1$. Figure 2.4 demonstrates the linking of pixels in a foreground pixel trajectory.

Pixel Trajectory RNN

After linking foreground pixels into trajectories, we describe our proposed Recurrent Neural Network (RNN) to learn feature embedings of these trajectories. Denote $\{(i_t, j_t)\}_{t=1}^{L}$ to be the pixel locations of a foreground trajectory, $\{\mathbf{x}_t^{i_t,j_t} \in \mathbb{R}^C\}_{t=1}^{L}$ to be the pixel embeddings of the foreground trajectory (Y-Net outputs, i.e. $\mathbf{x}_t = \phi(I_t, F_t)$), and L as the length of the trajectory. We define the foreground trajectory embeddings to be a weighted sum of the

standard	*conv*	*convGRU*
$\mathbf{c}_t^{i,j} = \text{ReLU}\left(W_c \left[\begin{array}{cc} \widetilde{\mathbf{h}}_{t-1}^{i,j} & \mathbf{x}_t^{i,j} \\ \widetilde{\mathbf{W}}_{t-1}^{i,j} & \end{array}\right]\right)$ $\mathbf{w}_t^{i,j} = \sigma\left(W_w \mathbf{c}_t^{i,j}\right)$	$\mathbf{c}_t = \text{ReLU}\left(W_c * \left[\begin{array}{cc} \widetilde{\mathbf{h}}_{t-1} & \mathbf{x}_t \\ \widetilde{\mathbf{W}}_{t-1} & \end{array}\right]\right)$ $\mathbf{w}_t = \sigma\left(W_w * \mathbf{c}_t\right)$	$\mathbf{z}_t = \sigma\left(W_z * \left[\begin{array}{cc} \widetilde{\mathbf{h}}_{t-1} & \mathbf{x}_t \\ \widetilde{\mathbf{W}}_{t-1} & \end{array}\right]\right)$ $\mathbf{r}_t = \sigma\left(W_r * \left[\begin{array}{cc} \widetilde{\mathbf{h}}_{t-1} & \mathbf{x}_t \\ \widetilde{\mathbf{W}}_{t-1} & \end{array}\right]\right)$ $\hat{\mathbf{c}}_t = \text{ReLU}\left(W_{\hat{c}} * \left[\begin{array}{cc} \mathbf{r}_t \odot \widetilde{\mathbf{h}}_{t-1} & \mathbf{x}_t \\ \widetilde{\mathbf{W}}_{t-1} & \end{array}\right]\right)$ $\mathbf{c}_t = (1 - \mathbf{z}_t) \odot \widetilde{\mathbf{c}}_{t-1} + \mathbf{z}_t \odot \hat{\mathbf{c}}_t$ $\mathbf{w}_t = \sigma\left(W_w * \mathbf{c}_t\right)$
$\mathbf{h}_t = \widetilde{\mathbf{h}}_{t-1} + \mathbf{w}_t \odot \mathbf{x}_t$ $\mathbf{W}_t = \widetilde{\mathbf{W}}_{t-1} + \mathbf{w}_t$		

Table 2.1: PT-RNN variants. For *standard*, we show the equations for pixel (i, j), while for the others we show equations in terms of the entire $H \times W \times C$ feature map. Note that for *standard*, $W_c \in \mathbb{R}^{1 \times 2C}$, $W_w \in \mathbb{R}^{1 \times C}$, while for *conv* and *convGRU*, $W_c, W_w, W_z, W_r, W_{\hat{c}}$ are 3×3 convolution kernels. $*$ denotes convolution and σ is the sigmoid nonlinearity.

pixel embeddings along the foreground trajectory. Specifically, we have

$$\psi\left(\{\mathbf{x}_t^{i_t,j_t}\}_{t=1}^L\right) = \frac{\sum_{t=1}^L \mathbf{w}_t^{i_t,j_t} \odot \mathbf{x}_t^{i_t,j_t}}{\sum_{t=1}^L \mathbf{w}_t^{i_t,j_t}}, \tag{2.2}$$

where $\odot$ denotes element-wise multiplication, the division sign denotes element-wise division, and $\mathbf{w}_t^{i_t,j_t} \in [0,1]^C$.

To compute the trajectory embeddings, we encode $\psi(\cdot)$ as a novel RNN architecture which we denote Pixel Trajectory RNN (PT-RNN). In its hidden state, PT-RNN stores

$$\left\{\mathbf{h}_t^{i_t,j_t} := \sum_{\tau=1}^t \mathbf{w}_\tau^{i_\tau,j_\tau} \odot \mathbf{x}_\tau^{i_\tau,j_\tau}, \mathbf{W}_t^{i_t,j_t} := \sum_{\tau=1}^t \mathbf{w}_\tau^{i_\tau,j_\tau}\right\}, \tag{2.3}$$

which allows it to keep track of the running sum and total weight throughout the foreground trajectory. While Eq. (2.3) describes the hidden state at each pixel location and time step, we can efficiently implement the PT-RNN for all pixels by doing the following: at time step t, PT-RNN first applies the foreground consistent warping function to compute $\widetilde{\mathbf{h}}_{t-1} := g\left(\mathbf{h}_{t-1}\right), \widetilde{\mathbf{W}}_{t-1} = g\left(\mathbf{W}_{t-1}\right)$. Next, we compute $\mathbf{w}_t$. We design three variants of PT-RNN to compute $\mathbf{w}_t$, named *standard* (based on simple RNNs), *conv* (based on convRNNs), and *convGRU* (based on [7]), shown in detail in Table

2.1. For *standard*, we show the equations for a single pixel trajectory. It computes weights based on the pixel embeddings along that trajectory without knowledge of any other trajectories. For *conv*, it uses a 3×3 convolution kernel instead of the standard matrix multiply to include information from neighboring trajectories. Lastly, for *convGRU*, we design this architecture based on the convGRU architecture [7] which has an explicit memory state to capture longer-term dependencies. For all three variants, the hidden state is $\{\mathbf{h}_t, \mathbf{W}_t\}$. However, in the RNN we propagate $\frac{\widetilde{\mathbf{h}}_t}{\widetilde{\mathbf{W}}_t}$, which is the intermediate weighted sum at time t. This allows the network to use knowledge of the previous weights and pixel embeddings to calculate $\mathbf{w}_{t+1}$.

When a trajectory is finished, i.e., pixel (i, j) does not link to any pixel in the next frame, PT-RNN outputs $\mathbf{h}_t^{i,j}/\mathbf{W}_t^{i,j}$, which is equivalent to Eq. (2.2). This results in a C-dimensional embedding for every foreground pixel trajectory, regardless of its length, when it starts, or when it ends. Note that these trajectory embeddings are pixel-dense, removing the need for a variational minimization step [115]. The embeddings are normalized so that they lie on the unit sphere.

A benefit to labeling the trajectories is that we are enforcing consistency in time, since consistent forward and backward optical flow usually means that the pixels are truely linked [153]. However, issues can arise around the motion and object boundaries, which can lead to trajectories erroneously drifting and representing motion of two different objects or an object and background [153]. In this case, the foreground masks are beneficial and able to sever the trajectory before it drifts. We also note the similarity to the DA-RNN architecture [170] that uses data association in a RNN for semantic labeling.

Spatial Coordinate Module

The foreground trajectory embeddings incorporate information from the RGB and optical flow images. However, they do not encode information about the location of the trajectory in the image. Thus, we introduce a spatial coordinate module which computes location information for each foreground trajectory. Specifically, we compute a 4-dimensional vector

consisting of the average x, y pixel location and displacement for each trajectory and pass it through two fully connected (FC) layers to inflate it to a C-dimensional vector, which we add to the output of $\psi(\cdot)$ (before the normalization of the foreground trajectory embeddings).

2.2.4 Loss Function

To train our proposed network, we use a loss function that is comprised of three terms

$$\mathcal{L} = \lambda_{\text{fg}}\ell_{\text{fg}} + \lambda_{\text{intra}}\ell_{\text{intra}} + \lambda_{\text{inter}}\ell_{\text{inter}} \ ,$$

where we set $\lambda_{\text{fg}} = \lambda_{\text{intra}} = \lambda_{\text{inter}} = 1$ in our experiments. ℓ_{fg} is a pixel-wise binary cross-entropy loss that is commonly used in foreground prediction. We apply this on the predicted foreground logits. ℓ_{intra} and ℓ_{inter} operate on the foreground trajectory embeddings. Inspired by [42], its goal is to encourage trajectory embeddings of the same object to be close while pushing trajectories that are different objects apart. For simplicity of notation, let us overload notation and define $\left\{\mathbf{x}_i^k\right\}, k = 1, \ldots, K,\ i = 1, \ldots, N_k$ to be a list of trajectory embeddings of dimension C where k indexes the object and i indexes the embedding. Since all the feature embeddings are normalized to have unit length, we use the cosine distance function $d(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\left(1 - \mathbf{x}^\intercal\mathbf{y}\right)$ to measure the distance between two feature embeddings $\mathbf{x}$ and $\mathbf{y}$.

Proposition 1. *Let $\{\mathbf{y}_i\}_{i=1}^N$ be a set of unit vectors such that $\sum_{i=1}^n \mathbf{y}_i \neq 0$. Define the* spherical mean *of this set of unit vectors to be the unit vector that minimizes the cosine distance*

$$\mu := \underset{\|\mathbf{w}\|_2=1}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} d\left(\mathbf{w}, \mathbf{y}_i\right) \tag{2.4}$$

Then $\mu = \frac{\sum_{1=1}^n \mathbf{y}_i}{\left\|\sum_{1=1}^n \mathbf{y}_i\right\|}$.

Proof. We note the the following:

$$\begin{aligned}
\underset{\|\mathbf{w}\|_2=1}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} d(\mathbf{w},\mathbf{y}_i) &= \underset{\|\mathbf{w}\|_2=1}{\operatorname{argmin}} \ \frac{1}{2n}\sum_{i=1}^{n}(1-\mathbf{w}^\intercal\mathbf{y}_i) \\
&= \underset{\|\mathbf{w}\|_2=1}{\operatorname{argmin}} \ \left[1-\frac{1}{n}\sum_{i=1}^{n}\mathbf{w}^\intercal\mathbf{y}_i\right] \\
&= \underset{\|\mathbf{w}\|_2=1}{\operatorname{argmax}} \ \sum_{i=1}^{n}\mathbf{w}^\intercal\mathbf{y}_i \\
&= \underset{\|\mathbf{w}\|_2=1}{\operatorname{argmax}} \ \mathbf{w}^\intercal\sum_{i=1}^{n}\mathbf{y}_i
\end{aligned}$$

Note that the unit vector that maximizes the inner product with a given vector $\mathbf{v}$ is simply the normalized version of $\mathbf{v}$ (if $\mathbf{v} \neq 0$). Thus, the solution to the above problem is $\frac{\sum_{i=1}^{n}\mathbf{y}_i}{\left\|\sum_{i=1}^{n}\mathbf{y}_i\right\|_2}$. □

The goal of the intra-object loss ℓ_{intra} is to encourage these learned trajectory embeddings of an object to be close to their spherical mean. This results in

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

where μ_k is the spherical mean of trajectories $\left\{\mathbf{x}_i^k\right\}_{i=1}^{N_k}$ for object k, and $\mathbb{1}$ denotes the indicator function. Note that μ_k is a function of the embeddings. The indicator function acts as a hard negative mining that focuses the loss on embeddings that are further than margin α from the spherical mean. In practice, we do not let the denominator get too small as it could result in unstable gradients, so we allow it to reach a minimum of 50.

Lastly, the inter-object loss ℓ_{inter} is designed to push trajectories of different objects apart. We desire the clusters to be pushed apart by some margin δ, giving

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

where $[x]_+ = \max(x, 0)$. This loss function encourages the spherical means of different objects to be at least δ away from each other. Since our embeddings lie on the unit sphere and our distance function measures cosine distance, δ does not need to depend on the feature dimension C. In our experiments, we set $\delta = 0.5$ which encourages the clusters to be at least 90 degrees apart.

2.2.5 Trajectory Clustering

At inference time, we cluster the foreground trajectory embeddings with the von Mises-Fisher mean shift (vMF-MS) algorithm [84]. This gives us the clusters as well as the number of clusters, which is the estimated number of objects in a video. vMF-MS finds the modes of the kernel density estimate using the von Mises-Fisher distribution. The density can be described as $p(\mathbf{y}; \mathbf{m}, \kappa) = C(\kappa) \exp(\kappa \mathbf{m}^\intercal \mathbf{y})$ for unit vector $\mathbf{y}$ where κ is a scalar parameter, $\|\mathbf{m}\|_2 = 1$, and $C(\kappa)$ is a normalization constant. κ should be set to reflect the choice of α. If the training loss is perfect and $d(\mu_k, \mathbf{x}_i^k) < \alpha, \forall i = 1, \ldots, N_k$, then all of the $\mathbf{x}_i^k$ lie within a ball with angular radius $\cos^{-1}(1 - 2\alpha)$ of μ_k. In our experiments, we set $\alpha = 0.02$, giving $\cos^{-1}(1 - 2\alpha) \approx 16$ degrees. Thus, we set $\kappa = 10$, resulting in almost 50% of the density being concentrated in a ball with radius 16 degrees around $\mathbf{m}$ (by eyeing Figure 2.12 of [151]).

Running the full vMF-MS clustering is inefficient due to our trajectory representation being pixel-dense. Instead, we run the algorithm on a few randomly chosen seeds that are far apart in cosine distance. If the network learns to correctly predict clustered trajectory embeddings, then this random initialization should provide little variance in the results. Furthermore, we use a PyTorch-GPU implementation of the vMF-MS clustering for efficiency.

2.3 Experiments

Datasets. We evaluate our method on video foreground segmentation and multi-object motion segmentation on five datasets: Flying Things 3d (FT3D) [105], DAVIS2016 [120], Freibug-Berkeley motion segmentation [115],

Complex Background [111], and Camouflaged Animal [15]. For FT3D, we combine object segmentation masks with foreground labels provided by [161] to produce motion segmentation masks. For DAVIS2016, we use the $\mathcal{J}$-measure and $\mathcal{F}$-measure for evaluation. For FBMS, Complex Background, and Camouflaged Animal, we use precision, recall, and F-score, and ΔObj metrics for evaluation as defined in [115, 16].

It is well-understood that the original FBMS labels are ambiguous [14]. Some labels exhibit multiple segmentations for one aggregate motion, or segment the (static) background into multiple regions. Thus, [14] provides corrected labels which we use for evaluation.

Implementation Details. We train our networks using stochastic gradient descent with a fixed learning rate of 1e-2. We use backpropagation through time with sequences of length 5 to train the PT-RNN. Each image is resized to 224×400 before processing. During training (except for FT3D), we perform data augmentation, which includes translation, rotation, cropping, horizontal flipping, and color warping. We set $C = 32, \alpha = 0.02, \delta = 0.5, \kappa = 10$. We extract optical flow via [69].

Labels for each foreground trajectory are given by the frame-level label of the last pixel in the trajectory. Due to sparse labeling in the FBMS training dataset, we warp the labels using Eq. (2.1) so that each frame has labels. Lastly, due to the small size of FBMS (29 videos for training), we leverage the DAVIS2017 dataset [125] and hand select 42 videos from the 90 videos that roughly satisfy the rubric of [14] to augment the FBMS training set. We denote this as DAVIS-m.

When evaluating the full model on long videos, we suffer from GPU memory constraints. Thus, we devise a sliding window scheme to handle this. First, we cluster all foreground trajectories within a window. We match the clusters of this window with the clusters of the previous window using the Hungarian algorithm. We use distance between cluster centers as our matching cost, and further require that matched clusters must have $d(\mu_k, \mu_{k'}) < 0.2$. When a cluster is not matched to any of the previous clusters, we declare it a new object. We use a 5-frame window and adopt this scheme for the FBMS and Camouflaged Animal datasets.

	FT3D	DAVIS	FBMS
Y-Net	0.905	0.701	0.631
Early Fusion	0.883	0.636	0.568
Late Fusion	0.897	0.631	0.570

Table 2.2: Fusion ablation. Performance is measured in IoU.

In Section 2.3.2, we use the *conv* PT-RNN variant of Figure 2.2, trained for 150k iterations on FT3D, then fine-tuned on FBMS+DAVIS-m for 100k iterations.

Our implementation is in PyTorch, and all experiments run on a single NVIDIA TitanXP GPU. Given optical flow, our algorithm runs at approximately 15 FPS. Note that we do not use a CRF post-processing step for motion segmentation.

2.3.1 Ablation Studies

Fusion ablation. We show the choice of mid-level fusion with Y-Net is empirically a better choice than early fusion and late fusion of encoder-decoder networks. For early fusion, we concatenate RGB and optical flow and pass it through a single U-Net. For late fusion, there are two U-Nets: one for RGB and one for optical flow, with a conv layer at the end to fuse the outputs. Note that Y-Net has more parameters than early fusion but less parameters than late fusion. Table 2.2 shows that Y-Net outperforms the others in terms of foreground IoU. Note that the performance gap is more prominent on the real-world datasets.

Architecture ablation. We evaluate the contribution of each part of the model and show results in both the multi-object setting and the binary setting (foreground segmentation) on the FBMS testset. All models are pre-trained on FT3D for 150k iterations and trained on FBMS+DAVIS-m for 100k iterations. Experiments with the different PT-RNN variants shows that *conv* PT-RNN performs the best empirically in terms of F-score, thus we use this in our comparison with state-of-the-art methods. *Standard* performs similarly, while *convGRU* performs worse perhaps due to overfitting to the small dataset. Next, we remove the PT-RNN architecture (per-frame embed-

	Multi-object				Foreground		
	P	R	F	ΔObj	P	R	F
conv PT-RNN	75.9	66.6	67.3	4.9	90.3	87.6	87.7
standard PT-RNN	72.2	66.6	66.0	4.27	88.1	89.3	87.5
convGRU PT-RNN	73.6	63.8	64.8	4.07	89.6	85.8	86.3
per-frame embedding	79.9	56.7	59.7	11.2	92.1	85.4	87.4
no FG mask	63.5	60.3	59.6	1.97	82.5	85.7	82.1
no SCM	70.4	65.5	63.2	3.70	89.3	89.1	88.1
no pre-FT3D	70.2	63.6	63.1	3.66	87.6	88.2	86.3
no DAVIS-m	66.9	63.6	62.1	2.07	87.1	86.9	85.2

Table 2.3: Architecture and Dataset ablation on FBMS testset.

ding) and cluster the foreground pixels at each frame. The F-score drops significantly and ΔObj is much worse, which is likely due to this version not labeling clusters consistently in time. Because the foreground prediction is not affected, these numbers are still reasonable. Next, we remove foreground masks (no FG mask) and cluster all foreground and background trajectories. The clustering is more sensitive; if the background trajectories are not clustered adequately in the embedding space, the performance will suffer. Lastly, we removed the spatial coordinate module (no SCM) and observed lower performance. Similar to the per-frame embedding experiment, foreground prediction is not affected.

Dataset ablation. We also study the effects of the training schedule and training dataset choices. In particular, we first explore the effect of not pre-training on FT3D, shown in the bottom portion of Table 2.3. Secondly, we explore the effect of training the model only on FBMS (without DAVIS-m). Both experiments show a noticeable drop in performance in both the multi-object and foreground/background settings, showing that these ideas are crucial to our performance.

2.3.2 Comparison to State-of-the-Art Methods

Video Foreground Segmentation. For FBMS, ComplexBackground and CamouflagedAnimal, we follow the protocol in [16] which converts the

		Video Foreground Segmentation							Multi-object Motion Segmentation			
		PCM [15]	FST [116]	NLC [47]	MPNet [161]	LVO [160]	CCG [16]	Ours	CVOS [157]	CUT [78]	CCG [16]	Ours
FBMS	P	79.9	83.9	86.2	87.3	92.4	85.5	90.3	72.7	74.6	74.2	75.9
	R	80.8	80.0	76.3	72.2	85.1	83.1	87.6	54.4	62.0	63.1	66.6
	F	77.3	79.6	77.3	74.8	87.0	81.9	87.7	56.3	63.6	65.0	67.3
	ΔObj	-	-	-	-	-	-	-	11.7	7.7	4.0	4.9
CB	P	84.3	87.6	79.9	86.8	74.6	87.7	83.1	60.8	67.6	64.9	57.7
	R	91.7	85.0	69.3	77.5	77.0	93.1	89.7	44.7	58.3	67.3	61.9
	F	86.6	80.6	73.7	78.2	70.5	90.1	83.5	45.8	60.3	65.6	58.3
	ΔObj	-	-	-	-	-	-	-	3.4	3.4	3.4	3.2
CA	P	81.9	73.3	82.3	77.8	77.6	80.4	78.5	84.7	77.8	83.8	77.2
	R	74.6	56.7	68.5	62.0	51.1	75.2	79.7	59.4	68.1	70.0	77.2
	F	76.3	60.4	72.5	64.8	50.8	76.0	77.1	61.5	70.0	72.2	75.3
	ΔObj	-	-	-	-	-	-	-	22.2	5.7	5.0	5.4
All	P	80.8	82.1	84.7	85.3	87.4	84.7	87.1	73.8	74.5	75.1	74.1
	R	80.7	75.8	73.9	70.7	77.2	82.7	86.2	54.3	62.8	65.0	68.2
	F	78.2	75.8	75.9	73.1	77.7	81.5	85.1	56.2	64.5	66.5	67.9
	ΔObj	-	-	-	-	-	-	-	12.9	6.8	4.1	4.8

Table 2.4: Results for FBMS, ComplexBackground (CB), CamouflagedAnimal (CA), and averaged over all videos in these datasets (ALL). Best results are highlighted in red with second best in blue.

motion segmentation labels into a single foreground mask and use the metrics defined in [115] and report results averaged over those three datasets. We compare our method to state-of-the-art methods including PCM [15], FST [116], NLC [47], MPNet [161], LVO [160], and CCG [16]. We report results in Table 2.4. In terms of F-score, our model outperforms all other models on FBMS and CamouflagedAnimal, but falls just short on ComplexBackground behind PCM and CCG. Looking at all videos, we show a relative gain of 4.4% on F-score compared to the second best method CCG, due to our high recall.

Additionally, we report results of our model on FT3D and the validation set of DAVIS2016. We compare our model to state-of-the-art methods: including LVO [160], FSEG [70], MPNet [161], and FST [116] in Table 2.5. For this experiment only, we train a Y-Net with $C = 64$ channels on FT3D for 100k iterations, resulting in outperforming MPNet by a relative gain of 5.6%. We then fine-tune for 50k iterations on the training set of DAVIS2016 and use a CRF [86] post-processing step. We outperform all methods in terms of $\mathcal{F}$-measure and all methods but LVO on $\mathcal{J}$-measure. Note that unlike LVO, we do not utilize an RNN for video foreground segmentation, yet we still achieve performance comparable to the state-of-the-art. Also, LVO [160] reports a $\mathcal{J}$-measure of 70.1 without using a CRF, while our method attains

		FST [116]	FSEG [70]	MPNet [161]	LVO [160]	Ours
DAVIS	$\mathcal{J}$	55.8	70.7	70.0	75.9	74.2
	$\mathcal{F}$	51.1	65.3	65.9	72.1	73.9
FT3D	IoU	-	-	85.9	-	90.7

Table 2.5: Results on Video Foreground Segmentation for DAVIS2016 and FT3D. Best results are highlighted in red.

a $\mathcal{J}$-measure of 71.4 without using a CRF. This demonstrates the efficacy of the Y-Net architecture.

Multi-object Motion Segmentation. We compare our method with state-of-the-art methods CCG [16], CUT [78], and CVOS [157]. We report our results in Table 2.4. We outperform all models on F-score on the FBMS and CamouflagedAnimal datasets. On FBMS, we dominate on precision, recall, and F-score with a relative gain of 3.5% on F-score compared to the second best method CCG. Our performance on ΔObj is comparable to the other methods. On CamouflagedAnimal, we show higher recall with lower precision, leading to a 4.4% relative gain in F-score. Again, our result on ΔObj is comparable. However, our method places third on the ComplexBackground dataset. This small 5-sequence dataset exhibits backgrounds with varying depths, which is hard for our network to correctly segment. However, we still outperform all other methods on F-score when looking at all videos. Similarly to the binary case, this is due to our high recall.

To illustrate our method, we show qualitative results in Figure 2.5. We plot RGB, optical flow [69], groundtruth, results from the state-of-the-art CCG [16], and our results on 4 sequences (*goats01*, *horses02*, and *cars10* from FBMS, and *forest* from ComplexBackground). On *goats01*, our results illustrate that due to our predicted foreground mask, our method is able to correctly segment objects that do not have instantaneous flow. CCG struggles in this setting. On *horses02*, we show a similar story, while CCG struggles to estimate rigid motions for the objects. Note that our method provides accurate segmentations without the use of a CRF post-processing step. We show two failure modes for our algorithm: 1) if the foreground

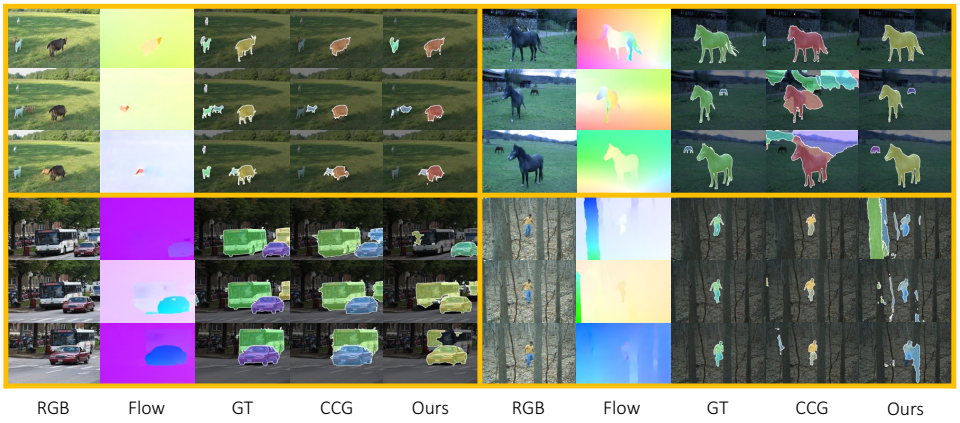


Figure 2.5: Qualitative results for motion segmentation. The videos are: *goats01*, *horses02*, and *cars10* from FBMS, and *forest* from ComplexBackground.

mask is poor, the performance suffers as shown on *cars10* and *forest*, and 2) cluster collapse can cause multiple objects to be segmented as a single object as shown in *cars10*.

2.4 Discussion

In this chapter, we proposed a novel deep network architecture, PT-RNN, for solving the problem of discovering unseen object instances using object motion cues. We formulated the problem as foreground motion clustering, and introduced an encoder-decoder network that learns representations of video frames and optical flow along with a novel recurrent neural network that learns feature embeddings of pixel trajectories inside foreground masks. By clustering these embeddings, we are able to discover and segment potentially unseen objects in videos. We demonstrated the efficacy of our approach on several motion segmentation datasets for object discovery.

A strong assumption of this chapter is that objects of interest undergo motion. In our recurring example of a robot being deployed in a new unstructured environment such as a house or office, the objects to be manip-

ulated are typically inanimate objects (e.g. toys), which will not experience motion unless physically manipulated by an external force. This implies that the robot will have to physically touch each object in order to observe motion cues for the unseen object before potentially applying a method such as PT-RNN. As the number of unseen objects may be very large, this clearly will not scale and we will need some other priors for detecting unseen objects that do not require motion. In the next chapter, we will investigate a method that exploits a different visual cue for doing so from static images, namely geometry cues.

Chapter 3

Unseen Object Instance Segmentation for Robotic Environments

This chapter discusses work originally published in Xie et al. [178].

For a robot to function in an unstructured environment, it must have the ability to recognize new objects that have not been seen before. Assuming every object in the environment has been modeled is infeasible and impractical. Recognizing unseen objects is a challenging perception task since the robot needs to learn the concept of "objects" and generalize it to unseen objects. Building such a robust object recognition module is valuable for robots interacting with objects, such as picking up unseen objects or learning to use new tools [109, 110, 108]. A common environment in which manipulation tasks take place is on tabletops. Thus, we approach this by focusing on the problem of Unseen Object Instance Segmentation (UOIS), where the goal is to segment every arbitrary (and potentially unseen) object instance, in tabletop environments.

In order to ensure the generalization capability of the module to recognize unseen objects, we need to learn from data that contains large amounts of various objects. However large-scale datasets with this property do not

exist. Since collecting a large dataset with manual annotations is expensive and time-consuming, it is appealing to utilize synthetic data for training, such as using the ShapeNet repository which contains thousands of 3D objects [24]. However, there exists a domain gap between synthetic data and real-world data as many simulators do not provide realistic-looking images. Training directly on such synthetic data only usually does not work well in the real world [183], which is also evidenced by Figure 1.1. Additionally, synthesizing photo-realistic images with physics-based rendering can be computationally expensive [64], making large photorealistic synthetic datasets impractical to obtain.

Consequently, recent efforts in robot perception have been devoted to the problem of Sim-to-Real, where the goal is to transfer capabilities learned in simulation to real-world settings. For instance, some works have used domain adaptation techniques to bridge the gap when unlabeled real data is available [163, 19]. Domain randomization [159] was proposed to diversify the rendering of synthetic data for training. While these techniques attempt to fix the discrepancy between synthetic and real-world RGB, models trained with synthetic depth have been shown to generalize reasonably well for simple settings such as bin-picking [103, 37]. However, in more complex settings, noisy depth sensors can limit the application of such methods and models trained on RGB have been shown to produce accurate masks [63]. An ideal method should combine the generalization capability of training on synthetic depth and the ability to produce sharp masks by utilizing RGB.

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

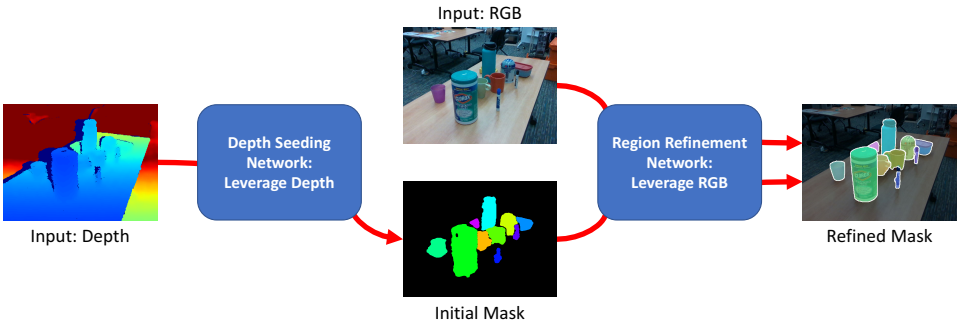


Figure 3.1: High level overview of the proposed two-stage framework of UOIS-Net. The first stage leverages depth only to produce rough initial masks. The second stage then leverages RGB to refine the initial masks to produce accurate, sharp instance masks.

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

Thus, our second stage is a Region Refinement Network (RRN) that takes an initial mask from the DSN and an RGB image as input and outputs a refined mask. Our surprising result is that, conditioned on initial masks, our RRN can be trained on non-photorealistic synthetic RGB images without adopting any of the afore-mentioned Sim-to-Real solutions. We posit that mask refinement is an easier problem than directly using RGB as input to produce masks, mainly because the mask refinement uses a local image patch as input and focuses on a single object. We empirically show robust generalization across many different objects in cluttered real-world data. In fact, our RRN works almost as well as if it were trained on real data. Our framework produces sharp and accurate masks even when the depth images are noisy. We show that it outperforms state-of-the-art methods including Mask R-CNN [63] and PointGroup [72]. Figure 3.1 illustrates our two-stage framework.

To train our method, we introduce a synthetic dataset of tabletop objects in home environments, which we name Tabletop Object Dataset (TOD). Our dataset consists of indoor scenes of random ShapeNet [24] objects on

random ShapeNet tables. We use the PyBullet physics simulator [36] to generate the scenes and render depth and non-photorealistic RGB. Training our proposed method on this dataset results in state-of-the-art results on multiple real-world datasets for UOIS.

3.1 Related Works

3.1.1 Category-level Object Segmentation

2D semantic segmentation involves assigning pixels in an image to a set of known classes. Deep learning has emerged as the most popular tool for solving this problem [147, 28, 6, 30, 98]. [147] first introduced the concept of using a fully convolutional architecture (FCN). [28] designed an architecture that utilizes dilated convolutions in order to increase the receptive field. [6, 30] further improves performance by introducing decoder architectures on top of the encoders. [98] proposed a multi-path refinement network with long-range residual connections to enable high-resolution predictions. These methods have demonstrated strong performance on datasets such as PASCAL [101] and COCO [99].

Much work has been devoted to solving the semantic segmentation problem in 3D as well. A common representation of 3D space is voxels; however operating with voxel grids as input can be expensive both computationally and memory-wise. Thus, [57] introduced a submanifold sparse convolutional operator to preserve spatial sparsity of the input. [34] further generalized these sparse convolutions to arbitrary kernel shapes, improving performance. Other 3D methods utilize point clouds. [127] proposed PointNet, a permutation-invariant network architecture to handle point clouds, and [130] extended this to a hierarchical network that recursively applies PointNet in order to obtain multi-resolution features, similar to deep convolutional networks with decreasing resolutions via strides/max pooling.

The advent of RGB-D sensors such as Kinect allowed the research community to utilize of both modalities for semantic segmentation, and drove

the creation of datasets such as [112]. [134] investigated a combination of kernel descriptors, support vector machines, and Markov random fields for indoor scene segmentation. Both [91] and [170] leverage RGB-D videos, extracting 2D features from each RGB frame and integrating them with a reconstructed voxel representation of the scene. Deep learning-based approaches include [58, 165, 131]. [58] proposed the HHA encoding of depth images. The authors used this encoding to design an object detection system, which they further exploited to improve semantic segmentation performance. [147] also used the HHA in their pioneering work on FCNs. [165] proposed a depth-aware convolution and pooling mechanism to incorporate geometry into the convolution operators to build a depth-aware receptive field. [131] used the output of a 2D segmentation network to initialize node features of a graph neural network applied on a 3D point cloud which was backprojected from a depth image. In this chapter, we also leverage RGB-D images, but focus on segmenting each individual object with unknown object class.

3.1.2 Instance-level Object Segmentation

2D object instance segmentation is the problem of segmenting every object instance in an image. Many approaches for this problem involve top-down solutions that combine segmentation with object proposals in the form of bounding boxes [63, 95, 29, 83], typically produced by a region proposal network (RPN). FCIS [95] utilizes position-sensitive inside/outside score maps for fully end-to-end convolutional instance segmentation. Mask R-CNN [63], a prominent work in the field, predicts a foreground mask for each object proposal. [29] builds on top of both FCIS and Mask R-CNN and exploits semantic segmentation and direction predictions to assemble foreground masks. [83] proposes a module that iteratively refines segmentation predictions at adaptively selected locations, which can be used in conjunction with Mask R-CNN.

However, when bounding boxes contain multiple objects (e.g. cluttered robot manipulation setups), the true instance mask is ambiguous

and these methods struggle. Recently, a few methods have investigated bottom-up methods which assign pixels to object instances [42, 113, 114, 146]. Other methods examine dense sliding-window instance segmentation on 4D tensors [32], combining top-down and bottom-up methods via blending modules [27], and alternative mask representations such as contours [119]. Additionally, interactive instance segmentation has shown strong results with few user inputs [104].

Most of the afore-mentioned algorithms provide instance masks with category-level semantic labels, which do not generalize to unseen objects in novel categories. One approach to adapting these techniques to unseen objects is to employ "class-agnostic" training, which treats all object classes as one foreground category [37]. One family of methods exploits motion cues with class-agnostic training in order to segment arbitrary moving objects [174, 40], such as PT-RNN introduced in Chapter 2. Another family of methods are class-agnostic object proposal algorithms [122, 123, 89]. However, these methods will segment everything and require some post-processing method to select the masks of interest. [145] jointly estimates instance segmentation masks and rigid scene flow, similar to [21, 22]. We also train our proposed method in a class-agnostic fashion, but instead focus our notion of unseen objects in particular environments such as tabletop settings.

In 3D instance segmentation, researchers have recently been investigating architectures to apply on point clouds/voxel grids. [66] introduced the first deep learning method to fuse RGB and geometric information from RGB-D scans. [60] proposed an occupancy term which greatly aids supervoxel clustering, leading to strong results. A few of these methods embrace center voting-based techniques [90, 128, 129, 46, 72]. [90] utilizes metric learning to learn abstract features, and predicts center votes which are post-processed by meanshift clustering. [128] uses the center votes with a simple grouping mechanism to detect 3D bounding boxes of objects from point clouds only. Their follow up work [129] incorporates RGB information by lifting 2D votes and features into 3D. [46] follows a similar architecture but stacks a graph convolutional network to refine proposal features. [72] also performs clustering on votes and semantic features for targeted perfor-

mance on certain object classes. Our method takes inspiration from these voting-based methods, but is targeted to cluttered robot environments.

3.1.3 Sim-to-Real Perception

Training a model on synthetic RGB and directly applying it to real data typically fails [183]. Many methods employ some level of rendering randomization [172, 162, 96, 159, 124, 137], including lighting conditions and textures. However, they typically assume specific object instances and/or known object models. Another family of methods employ domain adaptation to bridge the gap between simulated and real images [163, 19]. Algorithms trained on depth have been shown to generalize reasonably well for simple settings [103, 37]. However, noisy depth sensors can limit the application of such methods. Our proposed method is trained purely on (non-photorealistic) synthetic RGB-D data and is accurate even when depth sensors are inaccurate, and can be trained without adapting or randomizing the synthetic RGB.

3.2 Method

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

Our framework consists of two separate networks that process Depth and RGB separately to produce instance segmentation masks. First, we design a Depth Seeding Network (DSN) that takes a depth image as input and outputs *initial* object instance segmentation masks. These initial masks can be quite noisy for a number of reasons, thus we design an Initial Mask Pro-

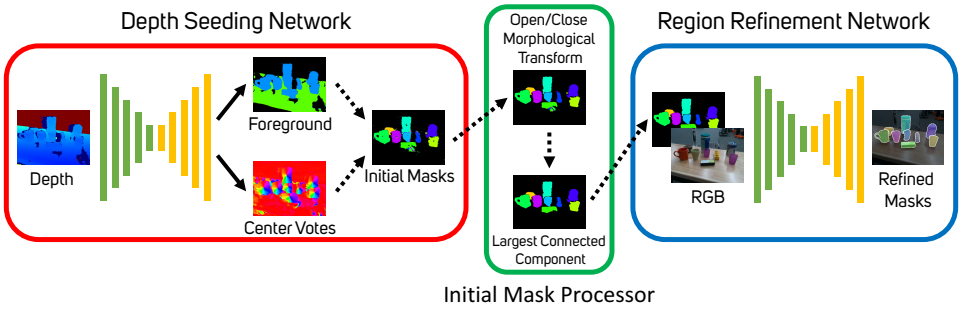


Figure 3.2: Overall architecture. The Depth Seeding Network (DSN) is shown in the red box, the Initial Mask Processor (IMP) in the green box, and the Region Refinement Network (RRN) in the blue box. The images come from a real example taken by an RGB-D camera in our lab. Despite the level of noise in the depth image (due to reflective table surface), our method is able to produce sharp and accurate instance masks. Gradients do not flow backwards through dotted lines.

cessor (IMP) to robustify them with standard image processing techniques. We further refine the processed initial masks using our Region Refinement Network (RRN), which is designed to snap the noisy initial mask edges to object edges in RGB, providing sharp and accurate final instance masks. The full architecture is shown in Figure 3.2.

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

3.2.1 Depth Seeding Network

It has been shown that depth generalizes reasonably well for Sim-to-Real problems [103, 37, 141]. Inspired by this concept, we focus the first stage

of our framework on depth only to produce initial class-agnostic instance segmentation masks. At a high level, the DSN takes as input a 3-channel organized point cloud, $D \in \mathbb{R}^{H \times W \times 3}$, of XYZ coordinates, and outputs initial instance segmentation masks. Note that D can be computed by backprojecting a depth map given camera intrinsics.

We examine two methods of structuring the DSN. First, we investigate building initial masks by predicting centers in 2D pixel space. While this method provides state-of-the-art results, it has some obvious pitfalls (examined in Section 3.4) that motivates a novel architecture that builds masks by predicting centers in 3D space.

Reasoning in 2D

Network Architecture The organized point cloud D is passed through an encoder-decoder architecture to produce two outputs: a semantic segmentation mask $F \in \mathbb{R}^{H \times W \times C}$, where C is the number of semantic classes, and 2D directions to object centers $V \in \mathbb{R}^{H \times W \times 2}$. We use $C = 3$ for our semantic classes: background, tabletop, and tabletop objects. Each pixel of V encodes a 2-dimensional unit vector pointing to the 2D center of the object. We define the center of the object to be the mean pixel location of the observable mask (part of mask that is unoccluded). Although we do not explicitly make use of the tabletop label in Section 3.4, it can be used in conjunction with RANSAC [52] in order to better estimate the table for downstream applications. For the encoder-decoder architecture, we use a U-Net [136] architecture where each 3×3 convolutional layer is followed by a GroupNorm layer [169] and ReLU. The output of the U-Net is a feature map of shape $\mathbb{R}^{H \times W \times 64}$. Sitting on top of this is two parallel branches of convolutional layers that produce the foreground mask F and center directions V (Figure 3.2). While we use U-Net for the DSN architecture, our framework is not limited to this and can replace it with any network architecture.

In order to compute the initial segmentation masks from F and V, we design a Hough voting layer similar to [172]. We describe the pseudocode detailed in Algorithm 1. First, we discretize the space of angles $[0, 2\pi]$ into

Algorithm 1 Hough Voting Procedure Pseudocode

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

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

A equally spaced bins. For every pixel, we compute the percentage of discretized directions from all other foreground pixels that point to it and use this as a score for how likely the pixel is an object center (lines 3-10). We threshold when a foreground pixel points to it with an inlier threshold and distance threshold. We then threshold the percentages and apply non-maximum suppression (NMS) to select object centers (line 11). Given these object centers, each pixel is assigned to the closest center it points to (line 12), which gives the initial masks as shown in the red box of Figure 3.2. Note that the inlier, distance, and percentage thresholds provide the Hough voting layer with robustness. For example, if not enough foreground pixels from all directions point towards a potential object center, that center is not selected. This robustifies the algorithm by protecting against false positives. We qualitatively show the efficacy of these design choices in Section 3.4.5.

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

Foreground Loss For the semantic segmentation F, we use a weighted cross entropy as this has been shown to work well in detecting object boundaries in imbalanced images [179]. The loss is $\ell_{fg} = \sum_i w_i \, \ell_{ce} \left(F_i, \bar{F}_i\right)$ where i ranges over pixels, $F_i, \bar{F}_i$ are the predicted and ground truth probabilities of pixel i, respectively, and ℓ_{ce} is the cross-entropy loss. The weight w_i is inversely proportional to the number of pixels with labels equal to $\bar{F}_i$, normalized to sum to 1.

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

$$\ell_{dir} = \sum_{i \in \mathcal{O}} \alpha_i \left(1 - V_i^\intercal \bar{V}_i\right) + \frac{\lambda_{bt}}{|\mathcal{B} \cup \mathcal{T}|} \sum_{i \in B \cup T} \left(1 - V_i^\intercal \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right), \tag{3.1}$$

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

Reasoning in 3D

Reasoning in 2D has some failure cases that can be mitigated by reasoning in 3D. For example, if the center of an object is occluded by another object, the 2D center voting procedure will not detect that object (examples of this can be found in Section 3.4.5). Thus, we propose a new architecture to the DSN to better handle these cases and provide stronger results. This formulation requires more sophisticated loss functions. In particular, we introduce a novel separation loss that significantly improves accuracy in cluttered scenes.

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

We propose to modify the architecture to use dilated convolutions [181] in order to provide the DSN with a higher receptive field. We replace the 6^{th}, 8^{th}, and 10^{th} convolution layers with ESP modules [106]. An ESP module is a lightweight module consisting of a *reduction* operation, a *split/transform* component that applies convolutions with different dilation rates to get a spatial pyramid, and a *merge* process that hierarchically fuses the feature maps of the spatial pyramid [106]. The ESP module has less parameters than the convolution layer it replaces, making it more computationally efficient. Details of the implementation can be found in the public code release at the project website[1]. We show in Section 3.4.6 that adding this module provides a boost in performance.

To compute initial masks, we perform mean shift clustering in 3D space over our center votes $D + V'$. Mean shift clustering is an iterative procedure to find the modes of a distribution approximated by a kernel density esimate (KDE). The number of clusters (objects, in our case) is not determined beforehand, but instead by the number of modes in the KDE. We use the Gaussian kernel $K(x, y) = \exp\left(\frac{1}{\sigma^2}\|x - y\|_2^2\right)$, which results in Gaussian mean shift (GMS) clustering. $\sigma > 0$ is a hyperparameter which affects the number of modes (objects) in the KDE. Thus, the choice of σ is crucial and depends on the relative distance between objects, which is low in clutter. A detailed review of mean shift clustering algorithms can be found in [23]. After clustering, each pixel is assigned to the cluster ID of its center vote to generate the initial masks. The clustering is only applied to the

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

foreground pixels. Note that this method of producing initial masks lacks the thresholds such as $\epsilon_{it}, \epsilon_d, \epsilon_{pt}$ from the 2D DSN Hough voting layer that provide robustness.

Loss Functions We apply four loss functions on the semantic segmentation F and center offsets V' to train the 3D-reasoning version of the Depth Seeding Network.

Foreground Loss We utilize the same foreground loss as in Section 3.2.1 for the 2D DSN, ℓ_{fg}.

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

$$\ell_{co} = \sum_{i \in \Omega} w_i \rho \left(D_i + V'_i - c_i \right) , \tag{3.2}$$

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

Clustering Loss We adopt a clustering loss that unrolls GMS for a few iterations and applies a loss on the clustered points, very similar to [85]. GMS iteratively shifts a set of S 3D seed points, $Z \in \mathbb{R}^{S \times 3}$, to higher density regions of the KDE in a gradient ascent-type fashion [23]. Let $Z^{(l)}$ be the points at the l^{th} iteration of GMS. $Z^{(0)}$ is initialized as the center votes $X = D + V' \in \mathbb{R}^{|\mathcal{O}| \times 3}$ of the foreground pixels. One iteration of GMS amounts to $Z^{(l+1)} = \tilde{D}^{-1} K X$ where $K \in \mathbb{R}^{S \times |\mathcal{O}|}$ is the kernel matrix s.t. $[K]_{ij} = K(Z_i^{(l)}, X_j)$, and $\tilde{D} = \text{diag}(K\mathbf{1})$ with $\mathbf{1} \in \mathbb{R}^{|\mathcal{O}|}$ being a vector of all ones. Note that K depends on σ. This iteration can be seen as a layer in the network with no parameters for learning. We apply the following

loss function to $Z^{(l)}$ and X, with the corresponding object instance labels $Y \in \mathbb{R}^{|\mathcal{O}|}$:

$$\begin{aligned}\ell_{cl}^{(l)}(Z^{(l)}, X, Y) = \sum_{i=1}^{S} \sum_{j \in \mathcal{O}} & w_{ij} \mathbb{1}\{y_i = y_j\} d^2(Z_i^{(l)}, X_j) \\ & + w_{ij} \mathbb{1}\{y_i \neq y_j\} [\delta - d(Z_i^{(l)}, X_j)]_+^2\end{aligned} \tag{3.3}$$

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

Applying Eq. (3.3) to all points, i.e. $S = |\mathcal{O}|$, results in excessive memory usage, thus we instead adopt a stochastic version of this loss function. We randomly sample an index set $\mathcal{I} \subset \{1, 2, \ldots, |\mathcal{O}|\}$ and set $Z^{(0)} = X_{\mathcal{I}}$, and run GMS clustering only on these points. We unroll GMS for L iterations, and apply $\ell_{cl}^{(l)}$ at each iteration, giving the full cluster loss $\ell_{cl} = \ell_{cl}^{(1)} + \ldots + \ell_{cl}^{(L)}$.

Separation loss We introduce a novel separation loss that encourages the center votes to not necessarily be at the center of an object, as long as it is far away from other object center votes in order to ease the post-processing GMS clustering phase. To do this, we consider the following tensor:

$$M_{ij} = \frac{\exp\left(-\tau \, d(c_j, D_i + V_i')\right)}{\sum_{j'=1}^{J} \exp\left(-\tau \, d(c_{j'}, D_i + V_i')\right)} \tag{3.4}$$

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

Maximizing M_{ij} for a foreground pixel i and its corresponding GT object center j encourages 1) the center vote $D_i + V_i'$ to be close to c_j, and 2) to be far from $\{c_j \mid y_i \neq y_j\}$. While the ℓ_{co} also enforces property 1, property 2 is quite desirable in heavy clutter. If two objects are situated in a way such

that their 3D object centers are very close, post-processing clustering will be difficult. This separation loss encourages the network to predict object center votes that are not close to each other, making the task of post-processing clustering easier. In Section 3.4.6, we show that this loss is crucial for strong performance in heavy clutter.

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

3.2.2 Initial Mask Processing Module

Computing the initial masks from F and V/V' often results in noisy masks (see an example of initial masks computed from the 2D DSN using our Hough voting layer in Figure 3.2). For example, these instance masks often exhibit salt/pepper noise and erroneous holes near the object center (see Section 3.4.5 for examples). As shown in Section 3.4.4, the RRN has trouble refining the masks when they are scattered as such. To robustify the algorithm, we propose to use two simple image processing techniques to clean the masks before refinement.

For a single instance mask, we first apply an opening operation, which consists of mask erosion followed by mask dilation [142], removing the salt/pepper noise issues. Next we apply a closing operation, which is dilation followed by erosion, which closes up small holes in the mask. Finally, we select the largest connected component and discard all other components. Note that these operations are applied to each instance mask separately. These simple image processing techniques are immensely helpful in robustifying the system.

3.2.3 Region Refinement Network

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

Network Architecture

Inspired by [104], this network takes as input a cropped 4-channel image, which consists of RGB concatenated with a single initial instance mask. The RGB image is cropped around the instance mask with some padding for context, concatenated with the (cropped) mask, then resized to 224×224. This gives an input image $I \in \mathbb{R}^{224\times224\times4}$. The output of the RRN is the refined mask probabilities $R \in \mathbb{R}^{224\times224}$, which we threshold to get the final output. We use the same U-Net architecture as in the DSN. To train the RRN, we apply the loss ℓ_{fg} with two classes (foreground vs. background) instead of three.

Mask Augmentation

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

- Translation/rotation: We translate the mask by sampling a displacement vector proportionally to the mask sizefrom a beta distribution. Rotation angles are sampled uniformly in $[-10°, 10°]$.
- Adding/cutting: For this augmentation, we choose a random part of the mask near the edge, and either remove it (cut) or copy it outside of the mask (add). This reflects the setting when the initial mask egregiously overflows from the object, or is only covering part of it.
- Morphological operations: We randomly choose multiple iterations of either erosion or dilation of the mask. The erosion/dilation kernel size is set to be a percentage of the mask size, where the percentage is

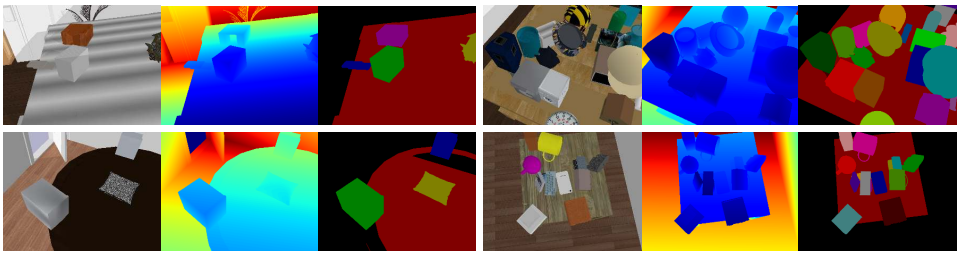


Figure 3.3: Examples from our Tabletop Object Dataset. (Non-photorealistic) RGB, depth, and instance masks are shown.

sampled from a beta distribution. This reflects inaccurate boundaries in the initial mask, e.g. due to noisy depth sensors.

- Random ellipses: We sample the number of ellipses to add or remove in the mask from a Poisson distribution. For each ellipse, we sample both radii from a gamma distribution and a random rotation angle. This augmentation requires the RRN to learn to remove irrelevant blots outside of the object and close up small holes within it.

3.3 Tabletop Object Dataset

Many desired robot environment settings (e.g. kitchens, cabinets) lack large scale training data to train deep networks. To our knowledge, there is no large scale dataset for unseen tabletop objects. To remedy this, we generate our own synthetic dataset which we name the Tabletop Object Dataset (TOD). This dataset is comprised of 40k synthetic scenes of cluttered ShapeNet [24] objects on a (ShapeNet) tabletop in SUNCG home environments [150]. We only use ShapeNet tables that have convex tabletops and filter the ShapeNet object classes to roughly 25 classes of objects that could potentially be on a table. Example classes include: jar, mug, helmet, and pillow.

Each scene in the dataset is of a random room chosen from a random SUNCG house loaded without any furniture. We sample a ShapeNet table

and scale it such that its height is in the range [0.75m, 1m], and place it in the room so that it is not colliding with walls or other fixtures in the room. Next, we randomly sample anywhere between 5 and 25 objects to put on the table, rescaling them such that they are not larger than $\frac{1}{4}\min\{t_h, t_l\}$ where t_h, t_l are the height and length of the table, respectively. The objects are either randomly placed on the table, on top of another object (stacked), or generated at a random height and orientation above the table. We use PyBullet [36] to simulate physics until the objects come to rest and remove any objects that fell off the table. Next, we generate seven views: one view is of only background, another is of just the table in the room, and the rest are taken with random camera viewpoints with the tabletop objects in view. The viewpoints are sampled at a height of between .5m and 1.2m above the table and rotated randomly with an angle in $[-12°, 12°]$. The images are generated at a resolution of 640×480 with vertical field-of-view of 45 degrees. The segmentation has a tabletop (table plane only) label and instance labels for each object.

We show some example images of our dataset in Figure 3.3. The rightmost two examples show that some of our scenes are heavily cluttered. Note that the RGB looks non-photorealistic. In particular, PyBullet is unable to load textures of some ShapeNet objects (see gray objects in leftmost two images). PyBullet was built for reinforcement learning, not computer vision, thus its rendering capabilities are insufficient for photorealistic tasks [36]. Despite this, our RRN can learn to snap masks to object boundaries from this synthetic dataset.

3.4 Experiments

We evaluate our method on real datasets against the state-of-the-art (SOTA) methods Mask R-CNN [63] and PointGroup [72]. We denote our method as *UOIS-Net-2D* or *UOIS-Net-3D*, depending on whether the DSN reasons in either 2D or 3D.

3.4.1 Implementation Details

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

DSN

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

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

During DSN training, we augment depth maps with multiplicative gamma noise similar to [103], and add Gaussian Process noise to the back-projected point clouds.

RRN

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

Baselines

For the baselines in [152], we use the results graciously provided by the authors. We follow the official Detectron schedule when training Mask

R-CNN [63] on TOD, and train for 100k iterations using SGD with a batch size of 8. We train PointGroup [72] for 300k iterations with a batch size of 4 on TOD. We remove clustering of the semantic scores since our problem only has one meaningful semantic class, foreground, which makes no sense to cluster.

Note that the training procedure for 2D DSNs and Mask R-CNN (reported in [175]) and differs slightly from the training procedure for 3D DSNs. We replicated these experiments using the same conditions as 3D DSNs (150k iterations with Adam/SGD), but observed very similar results to [175]. Thus, we report numbers from [175].

3.4.2 Datasets

We evaluate quantitatively and qualitatively on two real-world datasets: OCID [152] and OSD [135], which have 2346 images of semi-automatically constructed labels and 111 manually labeled images, respectively. OSD is a small dataset that was manually annotated, so the annotation quality is high. OCID, which is much larger, uses a semi-automatic process of annotating the labels. It incrementally builds up the scene by adding one object at a time, and computes labels by calculating difference in depth. However, this process is subject to depth sensor noise, so while the majority of the instance label is accurate, the label boundaries are noisy. Additionally, OCID contains images with objects on a tabletop, and images with objects on a floor. Despite our method being trained in synthetic tabletop settings, it generalizes to floor settings as well.

Lastly, we use the Google Open Images Dataset v5 [12] (OID) in an experiment to test the Sim-to-Real gap of the RRN. OID contains approximately 9 million "in-the-wild" images with image-level annotations, bounding boxes, and segmentations. In particular, it contains 2.8 million segmentations for 350 object classes. We filtered the 350 object object classes down to 156 classes that could potentially be on a tabletop, resulting in roughly 220k instance masks on real RGB images.

3.4.3 Metrics

We use the precision/recall/F-measure (P/R/F) metrics as defined in [40]. These metrics promote methods that segment the desired objects and penalize methods that provide false positives. Specifically, the F-measure is computed between all pairs of predicted and ground truth masks, which are matched via the Hungarian method. Given a matching, the final P/R/F is computed by

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

where s_i denotes the set of pixels belonging to predicted object i, $g(s_i)$ is the set of pixels of the matched ground truth object of s_i, and g_j is the set of pixels for ground truth object j. We denote this as Overlap P/R/F. See [40] for more details.

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

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

where we overload notation and denote s_i, g_j to be the set of pixels belonging to the boundaries of predicted object i and ground truth object j, respectively. $D[\cdot]$ denotes the dilation operation, which allows for some slack in the prediction. Note that these metrics are sensitive to the amount of allowed slack. However, without this, these numbers can look deceivingly low as no slack requires exact pixel boundary matching which can lead to issues with either noisy manual annotations or noisy semi-automatic annotations [152]. For the dilation operation, we use a circular kernel where the diameter of the kernel depends on the size of the image. Roughly, this metric is a combination of the $\mathcal{F}$-measure in [120] along with the Overlap P/R/F as

Method	Overlap			Boundary		
	P	R	F	P	R	F
GCUT [51]	21.5	51.5	25.7	10.2	46.8	15.7
SCUT [121]	45.7	72.5	43.7	43.1	65.1	42.6
LCCP [35]	58.4	89.1	63.8	53.6	82.6	60.2
V4R [126]	65.3	81.4	69.5	62.5	81.4	66.6
UOIS-Net-2D	88.8	81.7	84.1	83.0	67.2	73.3
UOIS-Net-3D	88.2	88.0	87.9	81.1	74.3	77.3

Table 3.1: Comparison with baselines on ARID20 and YCB10 subsets of OCID [152]. Red indicates the best performance.

defined in [40].

We report all P/R/F measures in the range $[0, 100]$ (P/R/F $\times 100$).

3.4.4 2D Quantitative Results

In this section, we quantitatively compare our UOIS-Net-2D to baselines and state-of-the-art algorithms. In all experiments with UOIS-Net-2D, we use an RRN trained on TOD, except when explicitly stated otherwise.

Comparison to baselines. We compare to baselines shown in [152], which include GCUT [51], SCUT [121], LCCP [35], and V4R [126]. In [152], these methods were only evaluated on the ARID20 and YCB10 subsets of OCID, so we compare our results these subsets as well. These baselines are designed to provide over-segmentations (i.e., they segment the whole scene instead of just the objects of interest). To allow a more fair comparison, we set all predicted masks smaller than 500 pixels to background, and set the largest mask to table label (which is not considered in our metrics). Results are shown in Table 3.1. Because the baselines aim to over-segment the scene, the precision is in general low while the recall is high. LCCP is designed to segment convex objects (most objects in OCID are convex), but its predicted boundaries are noisy due to utilizing depth only. Recall that OCID labels suffer from depth sensor noise, which explains why LCCP's boundary recall is quite high (see Section 3.4.5 for a visual example). Both SCUT and V4R utilize models trained on real data, putting them at an advantage

Method	Input	OCID [152]						OSD [135]					
		Overlap			Boundary			Overlap			Boundary		
		P	R	F	P	R	F	P	R	F	P	R	F
Mask R-CNN [63]	RGB	66.0	34.0	36.6	58.2	25.8	29.0	63.7	43.1	46.0	46.7	26.3	29.5
Mask R-CNN [63]	Depth	82.7	78.9	79.9	79.4	67.7	71.9	73.8	72.9	72.2	49.6	40.3	43.1
Mask R-CNN [63]	RGB-D	79.2	78.6	78.0	73.6	67.2	69.2	74.0	74.6	74.1	57.3	52.1	53.8
PointGroup [72]	RGB-D	81.6	80.1	80.1	75.4	70.4	71.7	79.5	78.2	78.8	67.0	65.0	65.4
UOIS-Net-2D	DSN: RGB-D	84.2	57.6	62.2	72.9	44.5	49.9	72.2	63.7	66.1	58.5	43.4	48.4
UOIS-Net-2D	DSN: Depth	88.3	78.9	81.7	82.0	65.9	71.4	80.7	80.5	79.9	66.0	67.1	65.6
UOIS-Net-3D	DSN: Depth	86.5	86.6	86.4	80.0	73.4	76.2	85.7	82.5	83.3	75.7	68.9	71.2

Table 3.2: Evaluation of our methods against SOTA methods trained on different input modes. Red indicates the best performance.

with respect to UOIS-Net-2D. V4R was trained on OSD [135] which has an extremely similar data distribution to OCID, giving V4R a substantial advantage and making it the strongest baseline in [152]. However, our UOIS-Net-2D (line 5 in Table 3.1), despite never having seen any real data, significantly outperforms these baselines on F-measure.

Comparison to SOTA. In Table 3.2, we compare UOIS-Net-2D to two SOTA methods, Mask R-CNN [63] and PointGroup [72], both trained on RGB-D from TOD. While Mask R-CNN is a general method for detection that can be applied to different types of input modes, the more recent PointGroup requires point clouds and utilizes a sparse convolutional backbone. We see that UOIS-Net-2D (line 6) outperforms Mask R-CNN (line 3) on OSD and slightly on OCID. On the other hand, PointGroup (line 4) provides comparable performance to UOIS-Net-2D. Note, however, that PointGroup reasons in 3D with center voting while UOIS-Net-2D does not. In that sense, Mask R-CNN is more similar to UOIS-Net-2D.

Note that the performance of UOIS-Net-2D is similar to Mask R-CNN and PointGroup on OCID in terms of boundary F-measure. This result is misleading: it turns out that using the RRN to refine the initial masks produced by UOIS-Net results in degraded quantitative performance on OCID, while the qualitative results are better. This is due to the semi-automatic labeling procedure in OCID that leads to ground truth segmentation labels aligning with noise from the depth camera [152]. An illustration of this can be found in the first example (row) of Figure 3.4. Table 3.3 shows the performance of UOIS-Net without applying RRN (DSN and IMP only)

Method	OCID [152]						OSD [135]					
	Overlap			Boundary			Overlap			Boundary		
	P	R	F	P	R	F	P	R	F	P	R	F
UOIS-Net-2D	90.9	79.4	83.0	85.2	70.8	75.7	80.3	81.1	80.1	53.4	53.3	52.8
UOIS-Net-3D	88.8	89.2	88.8	86.9	80.9	83.5	85.7	82.1	83.0	74.3	67.5	70.0

Table 3.3: Performance of UOIS-Net without RRN.

on OCID and OSD. This method utilizes only depth and predicts segmentation boundaries that are aligned with the sensor noise. In this setting, UOIS-Net-2D outperforms both SOTA methods, with a significant gain in boundary F-measure and a minor gain in overlap F-measure. In fact, UOIS-Net-3D also gains performance on OCID when removing the RRN which further demonstrates the issue with OCID labels.. On the other hand, OSD has manually annotated labels and this issue is not present, and our methods' performances deteriorate in the absence of the RRN. In particular, UOIS-Net-2D drops almost 20% relatively without it.

Effect of input mode. To evaluate how different input modes affect results, we train Mask R-CNN on RGB, depth, and RGB-D and compare it to UOIS-Net-2D in Table 3.2. Training Mask R-CNN on synthetic RGB only poorly generalizes from Sim-to-Real. Training on depth drastically boosts this generalization, which is in agreement with [103, 37, 141]. When training on RGB-D, we posit that Mask R-CNN relies heavily on depth as adding RGB to depth results in little change. However, UOIS-Net-2D exploits RGB and depth separately, leading to better results on OSD while being trained on the exact same synthetic dataset. Furthermore, when our DSN is trained directly on RGB-D (line 4, Table 3.2), we see a drop in performance, suggesting that training directly on (non-photorealistic) RGB is not the best way of utilizing synthetic data.

Degradation of training on non-photorealistic simulated RGB. To quantify how much non-photorealistic RGB degrades performance, we train an RRN on real data. This serves as an approximate upper bound on how well the synthetically-trained RRN can perform. We use the instance masks from OID [12] and show results in Table 3.4. Both models share the same DSN and IMP. The Overlap measures are roughly the same, while the RRN

DSN reasoning	RRN training data	OCID [152]						OSD [135]					
		Overlap			Boundary			Overlap			Boundary		
		P	R	F	P	R	F	P	R	F	P	R	F
2D	TOD	88.3	78.9	81.7	82.0	65.9	71.4	80.7	80.5	79.9	66.0	67.1	65.6
	OID	87.9	79.6	81.7	84.0	69.1	74.1	81.2	83.3	81.7	69.8	73.7	70.8
3D	TOD	86.5	86.6	86.4	80.0	73.4	76.2	85.7	82.5	83.3	75.7	68.9	71.2
	OID	86.0	88.2	86.9	83.1	77.6	79.9	86.0	85.1	84.8	81.0	75.3	77.3

Table 3.4: Comparison of RRN when training on TOD and real images from Google OID [12].

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

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

Table 3.5: (left) Ablation experiments for UOIS-Net-2D on OSD [135]. O/C denotes the Open/Close morphological transform, while CCC denotes Closest Connected Component. (right) Refining Mask R-CNN results with RRN (trained on TOD) on OSD.

trained on OID has slightly better performance on the Boundary measures. This suggests that while there is still a gap, our method is surprisingly not too far off, considering that we train with non-photorealistic synthetic RGB. We conclude that mask refinement with RGB is an easier task to transfer from Sim-to-Real than directly segmenting from RGB.

Ablation studies. We report ablation studies on OSD to evaluate each component of our proposed method in Table 3.5 (left). We omit the Overlap P/R/F results since they follow similar trends to Boundary P/R/F. Running the RRN on the raw masks output by DSN without the IMP module actually hurts performance as the RRN cannot refine such noisy masks. Adding the open/close morphological transform and/or the closest connect component results in much stronger results, showing that the IMP is crucial in robustifying our proposed method. In these settings, applying the RRN significantly boosts Boundary P/R/F. In fact, Table 3.5 (right) shows that applying the RRN to the Mask R-CNN results effectively boosts the Boundary P/R/F for all input modes on OSD, showing the efficacy of the RRN despite be-

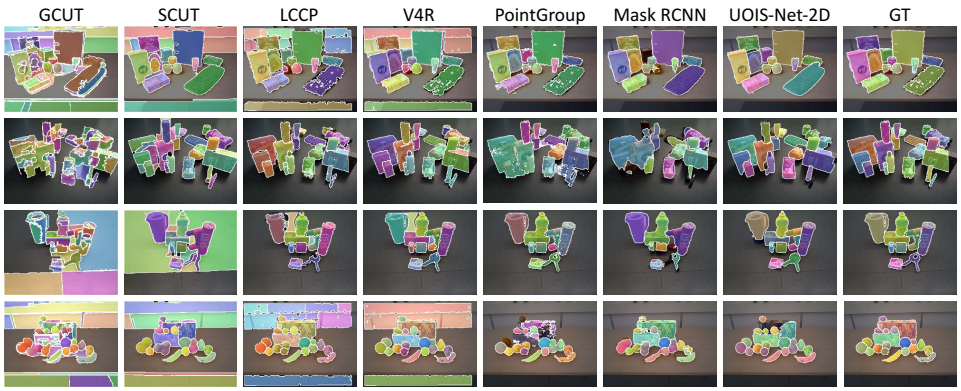


Figure 3.4: Comparison of UOIS-Net-2D with baselines, Mask R-CNN, and PointGroup on OCID [152]. LCCP and V4R operate on depth only, thus are subject to noise from depth sensors. However, this is also true for the ground truth segmentation produced by OCID [152]. Our proposed UOIS-Net-2D provides sharp and accurate masks in comparison to all of the baselines.

ing directly trained on non-photorealistic RGB. Note that even with this refinement, the Mask R-CNN results are outperformed by our method.

3.4.5 2D Qualitative Results

In this subsection, we qualitatively compare UOIS-Net-2D to the baselines given in [152] along with the state-of-the-art baseline Mask R-CNN. We also visually demonstrate the efficacy of our RRN, showcasing its capability for refinement. Finally, we investigate the robustness of the algorithm, along with common failure modes.

Comparison to baselines and SOTA. First, we show qualitative results on OCID of baseline methods, Mask R-CNN (trained on RGB-D), PointGroup, and UOIS-Net-2D in Figure 3.4. It is clear that the baseline methods suffer from over-segmentation issues; they segment the table and background into multiple pieces. This is especially the case for methods that utilize RGB as an input (GCUT and SCUT); the objects are often over-segmented as well. In example (row) 1 of Figure 3.4, the ground truth label for the keyboard is riddled with holes. Methods that operate only

Figure 3.5: Mask refinements with RRN: before (top) refinement (after IMP), and after refinement (bottom).

on depth (LCCP and V4R) mirror these noisy boundaries, leading to inflated boundary P/R/F measures. Despite this, our quantitative results still outperform these baselines.

The main failure mode for Mask R-CNN is that it tends to undersegment objects. This is the typical failure mode of top-down instance segmentation algorithms in clutter. A close inspection of Figure 3.4 shows that Mask R-CNN frequently segments multiple objects as one. Examples 1 and 4 show undersegmentation of many neighboring small objects, and example 2 shows undersegmentation of larger objects. Since PointGroup requires point clouds, it is susceptible to degradation from depth sensor noise as well. Additionally, some masks (e.g. example 4) can be quite noisy, which we hypothesize is due to the rudimentary breadth-first search clustering algorithm.

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

RRN Refinements. In Figure 3.5, we qualitatively show the effect of the RRN. The top row shows the masks before refinement (after IMP), and the bottom row shows the refined masks. These images were taken in our lab with an Intel RealSense D435 RGB-D camera to demonstrate the robustness of our method to camera viewpoint variations and distracting backgrounds,

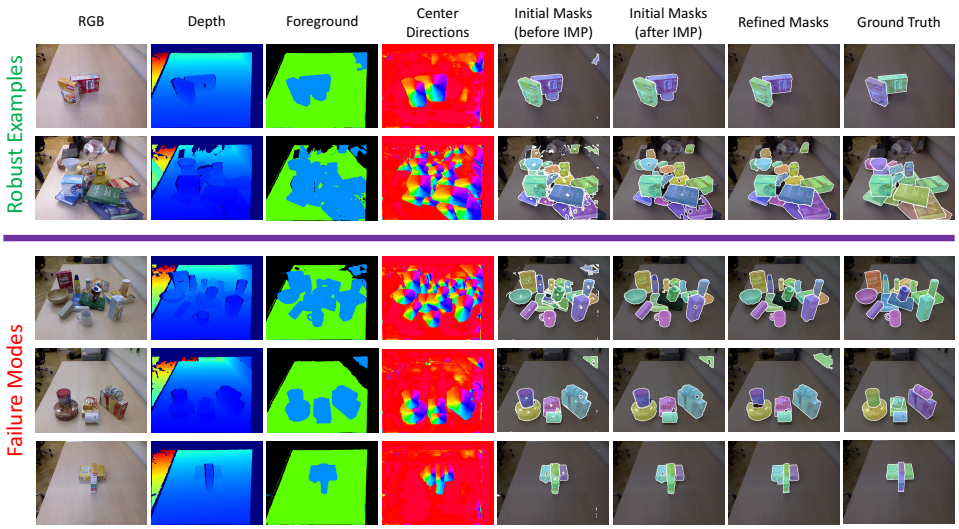


Figure 3.6: Outputs (and intermediate) of UOIS-Net-2D are visualized. We demonstrate the robustness of the Hough voting layer and the IMP (top) and show common failure modes (bottom). See text for details. Best viewed in color on a computer screen.

as OCID and OSD have relatively simple backgrounds. Due to noise in the depth sensor, it is impossible to get sharp and accurate predictions from depth alone without using RGB. Our RRN can provide sharp masks even when the boundaries of objects are occluding other objects (examples 2 and 5). Additionally, it is able to fix squiggly mask boundaries and patch up large missing chunks in the initial mask (example 6).

Robustness. In Figure 3.6 (top), we demonstrate how our method is robust to errors in the pipeline by visualizing the inputs, outputs, and intermediate outputs of the entire system. In row 1, the DSN produces a false positive foreground region in the top right of the image. However, there is not enough evidence in the center directions (not enough discretized angles are present), thus the Hough voting layer suppresses this potential object. The IMP further removes small masks components that are disconnected from the largest mask. In row 2, there are many spurious center direction predictions, however these are not considered as potential object centers by

the Hough voting layer because they are not detected by the foreground mask. Additionally, one can see the many holes and spurious mask components in column 5. As seen in the ablation studies in Section 3.4.4, applying the RRN here (without the IMP) actually degrades the performance. The IMP cleans this image, making the refinement task easier for the RRN.

Failure Modes. We show some failure modes of our DSN in Figure 3.6 (bottom) on the OSD dataset. In row 1, we see an undetected object (green book) because the center of the mask is occluded. In the center direction image, there is no convergent point for this object which would be considered as the object center with all discretized angles pointing to it. This is the main limitation of the 2D center voting procedure. In row 2, there is a false positive region detected by the DSN in the top right corner, which cannot be undone by the RRN. Lastly, in row 3, the DSN cannot correctly segment an object whose mask has been split in two by an occluding object.

Our RRN also has some common failure modes. Column 4 in Figure 3.5 shows that the RRN can fail when the object has a complex texture. Column 6 demonstrates that if there is not enough padding to the initial mask to see the entire object, the RRN cannot segment the entire object (lego block). In column 2, the RRN has a tough time fixing the segmentation mask when the DSN has incorrectly undersegmented a few objects together (the two cups to the left of the eraser).

3.4.6 3D Quantitative Results

In this section, we quantitatively compare UOIS-Net-3D to all of the previous baselines and our proposed UOIS-Net-2D, showing superior performance across all datasets and experiments.

Comparison to baselines. We compare the full UOIS-Net-3D with an RRN trained on TOD to all previous baselines from [152] and our UOIS-Net-2D in Table 3.1. Discussion of all baselines and UOIS-Net-2D is given in Section 3.4.4. Compared to UOIS-Net-2D, the 3D version substantially increases the recall in both the overlap and boundary metrics, leading to a relative increase of 4.5% in overlap F-measure and 5.5% in boundary F-

measures. In fact, UOIS-Net-3D, despite no convexity assumptions, gets quite close to the overlap recall of LCCP, which segments objects based on convexity. In the OCID dataset, the majority of objects are convex, thus this method performs quite well. Our method, without any such assumption and trained only on simulated data, achieves similar recall while focusing all the segmentation effort on the relevant tabletop objects, resulting in much stronger performance in precision and F-measure. The boundary recall also improves with respect to UOIS-Net-2D, but does not reach the quality of LCCP and V4R, because of the noisy ground truth object boundaries provided by the semi-automatic annotation capture by OCID [152] (see Section 3.4.5 for discussion and Figure 3.4 for visual examples). Note that only the DSN structure is changed when moving from 2D to 3D, thus this extra recall is mainly due to the 3D center voting structure and better initial masks.

Comparison to SOTA. In Table 3.2, we show comparisons of UOIS-Net-3D with SOTA methods Mask R-CNN [63] and PointGroup [72], and the previous UOIS-Net-2D on OCID and OSD. Again, the trend of performance increase from UOIS-Net-2D is similar to before, where a significant increase in recall leads to a boost in F-measure. On OCID, UOIS-Net-3D receives a relative increase of 5.8% in overlap F-measure and a 6.7% in boundary F-measure. Compared to the best performing Mask R-CNN, our performance provides a relative increase of 8.1% on overlap F-measure and 6.0% on boundary F-measure, while being trained on the exact same dataset (TOD). Additionally, we outperform PointGroup by 7.9% and 6.3% on overlap and boundary F-measures, respectively. Note that UOIS-Net-3D performs center voting in a similar fashion to PointGroup, however we believe our novel loss functions target cluttered environments more effectively. On OSD overlap F-measure, we provide a relative increase of 4.3% over UOIS-Net-2D, 12.4% over Mask R-CNN, and 5.7% over PointGroup. For boundary F-measure, we show 8.5% over UOIS-Net-2D, 32.3% over Mask R-CNN, and 8.9% over PointGroup. Note that while our recall jumps modestly in comparison to UOIS-Net-2D, the precision increase is more sizable.

RRN ablation. We refer readers to Table 3.4 to show the full power of

Loss Functions				OCID [152]						OSD [135]					
				Overlap			Boundary			Overlap			Boundary		
ℓ_{fg}	ℓ_{co}	ℓ_{cl}	ℓ_{sep}	P	R	F	P	R	F	P	R	F	P	R	F
✓	✓			77.9	79.8	78.6	71.2	68.0	68.9	79.4	80.1	79.8	74.5	66.5	69.8
✓	✓	✓		77.0	81.3	78.7	71.9	68.2	69.0	72.8	74.9	72.8	67.2	61.0	62.2
✓	✓		✓	85.5	87.9	86.5	80.9	77.8	78.8	84.0	85.1	84.5	75.0	74.9	74.6
✓	✓	✓	✓	86.0	88.2	86.9	83.1	77.6	79.9	86.0	85.1	84.8	81.0	75.3	77.3

Table 3.6: Ablation study over loss functions of UOIS-Net-3D. Our novel separation loss ℓ_{sep} is crucial to obtaining state-of-the-art performance. Note that we are using an RRN trained on real data.

our method. When using an RRN trained on OID [12], our full UOIS-Net-3D further increases the boundary F-measures, leading to 79.9 points on OCID and 77.3 points on OSD, which is 7.8% and 9.2% higher than the full UOIS-Net-2D on OCID and OSD, respectively. In the rest of this section, all UOIS-Net-3D's will use an RRN trained on OID in order to show the full performance of our method.

Loss function ablation. At the minimum, we require ℓ_{fg} and ℓ_{co} to train UOIS-Net-3D. In Table 3.6, UOIS-Net-3D actually performs poorer than UOIS-Net-2D (see Table 3.4) with these two losses only. When adding the cluster loss ℓ_{cl}, we get roughly the same performance on OCID. Our intuition is that this loss encourages the same behavior as ℓ_{co}, namely that the points should cluster near the ground truth object center, thus it doesn't introduce anything new. However, adding this loss significantly hurts performance on OSD; we posit that this result is noisy due to the small size of OSD. In the third row, we show that adding our novel ℓ_{sep} loss relatively boosts overlap F-measure by 10.1% and boundary F-measure by 14.4% compared to the base model ($\lambda_{fg}\ell_{fg} + \lambda_{co}\ell_{co}$), demonstrating it is crucial for obtaining good performance. ℓ_{sep} pushes center votes away from center votes of other objects in order to make the post-processing GMS clustering easier. Note that the center votes don't necessarily need to be close to the ground truth object centers; they simply need to be easily clustered, and this loss allows for that. This property is crucial to getting strong performance in clutter as the numbers demonstrate. We will also visually illustrate this phenomena in Section 3.4.7. Lastly, since the object clusters are not necessarily near the

DSN reasoning	ESP Module	OCID [152]						OSD [135]					
		Overlap			Boundary			Overlap			Boundary		
		P	R	F	P	R	F	P	R	F	P	R	F
2D	✗	87.9	79.6	81.7	84.0	69.1	74.1	81.2	83.3	81.7	69.8	73.7	70.8
	✓	87.6	85.0	85.7	84.4	73.4	77.8	84.6	82.3	82.8	75.5	71.6	72.4
3D	✗	83.9	85.6	84.4	79.1	76.4	77.2	85.1	85.7	85.1	78.9	76.5	77.3
	✓	86.0	88.2	86.9	83.1	77.6	79.9	86.0	85.1	84.8	81.0	75.3	77.3

Table 3.7: Ablation study to test the significance of a wider receptive field. Using the ESP module [106] in the DSN architecture improves performance for both UOIS-Net-2D and UOIS-Net-3D. Note that we are using an RRN trained on real data.

ground truth object centers, ℓ_{cl} now encourages something that ℓ_{co} does not: it encourages that the center votes are tightly packed wherever they are placed, further easing the job of the post-processing GMS clustering. In line 4 we see it provides some extra performance gain.

ESP module ablation. We evaluate the significance of using the ESP module [106] to obtain a higher receptive field by testing both UOIS-Net-3D and UOIS-Net-2D (which normally does not include the ESP module) with and without the ESP modules embedded into the DSN in Table 3.7. We immediately see that obtaining a higher receptive field is beneficial to the performance. For UOIS-Net-3D, we see relative gains on OCID of 3.0% and 3.5% on overlap and boundary F-measures, respectively. On OSD, we see similar performance. For UOIS-Net-2D, we see a large relative gain on OCID of 4.9% and 5% on overlap and boundary F-measures, respectively. Note that the DSN with ESP modules has slightly fewer parameters than without them, thus these experiments highlight the usefulness of a higher receptive field.

τ ablation. In the top row of Figure 3.7, we test the sensitivity of our model to the settings of τ over $[5, 10, 15, 20, 25]$, which is used in ℓ_{sep}. We visualize both overlap and boundary P/R/F measures. Note that as τ is essentially a multiplicative factor on the Euclidean distance in Eq. (3.4); thus, as τ becomes larger, the Euclidean distance becomes more inflated and objects do not need to be pushed as far in order to minimize ℓ_{sep}. It is clear to see Figure 3.7 that all metrics increases as τ increases, and they plateau

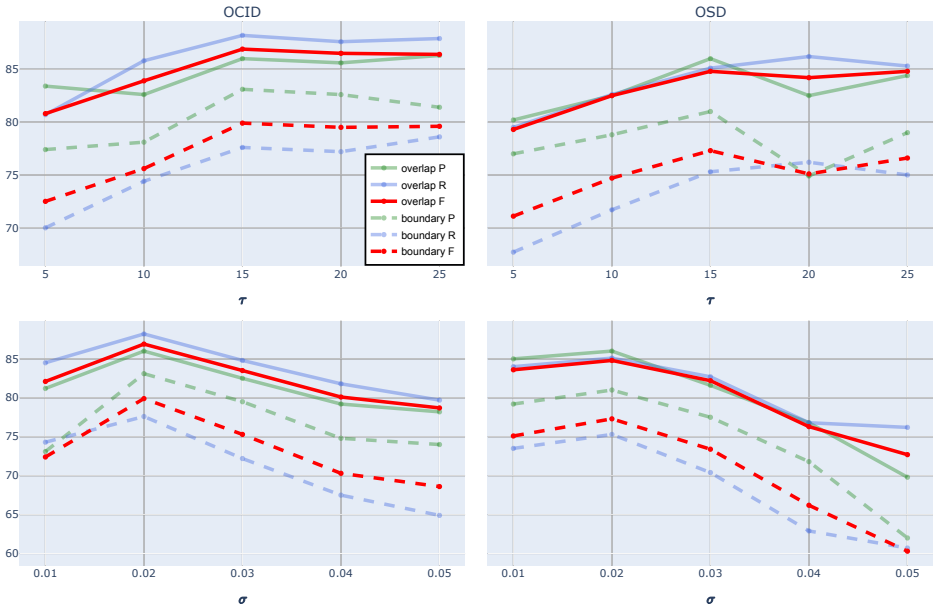


Figure 3.7: Ablation study to test the sensitivity of UOIS-Net-3D with respect to τ, σ. Best viewed by zooming in on a computer.

after τ hits a high enough value, in this case 15. This suggests that we only need a small level of encouragement to push the object centers away from each other in order to get strong performance, and that setting τ too low can degrade performance, potentially by making ℓ_{sep} too difficult to minimize.

σ ablation. In the bottom row of Figure 3.7, we test the sensitivity of UOIS-Net-3D to σ over $[0.01, 0.02, 0.03, 0.04, 0.05]$, which is used in ℓ_{cl}. The value of σ determines how tightly packed the center votes of an object need to be in order to minimize ℓ_{cl}. Additionally, it determines how much spread the center votes can have and still be clustered together during the post-processing GMS clustering step. When σ is too small, the DSN typically oversegments the objects. On the other hand, when σ is too large, center votes from multiple objects get clustered together and undersegmentation occurs. Thus, σ should be chosen to reflect the level of clutter in the scenes, i.e. how close the objects are. Our ablation study hints at this idea: the bottom row of Figure 3.7 that $\sigma = 0.02$ works the best for both OCID [152]

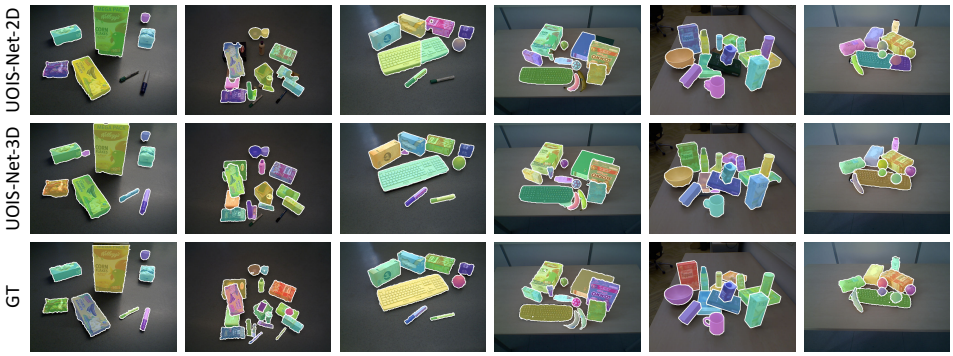


Figure 3.8: Qualitative comparison of UOIS-Net-2D to UOIS-Net-3D on OCID.

and OSD [135], as they are similar in distribution. In fact, the trends of the graph in both datasets are also similar. Note that we tried learning σ as a parameter of the network, however these experiments did not work well.

3.4.7 3D Qualitative Results

In this section, we qualitatively investigate the efficacy of UOIS-Net-3D. We compare it to UOIS-Net-2D to show how reasoning in 3D can solve the issues of reasoning in 2D pixel space, show the importance of using ℓ_{sep} by visualizing the center votes, and illustrate common failure modes of this architecture.

2D vs. 3D comparison. In Figure 3.8, we qualitatively compare the predictions of UOIS-Net-2D and UOIS-Net-3D to understand how reasoning in 3D can solve the 2D issues mentioned beforehand (in Section 3.4.5). The first row of Figure 3.6 (bottom) exhibits the 2D issue of false negative detection when the 2D center of the object is occluded. However, this is easily corrected when voting for 3D centers, as seen in columns 2 and 5 of Figure 3.8 In columns 1, 3, and 4, we see that UOIS-Net-3D detects and segments more small, thin objects such as pens and bananas. The 2D Hough voting procedure typically fails to detect enough discretized directions for objects with such shape. This issue is ameliorated when reasoning in 3D,

Figure 3.9: Effect of ℓ_{sep} on center votes. Rows 2 and 3 show the point cloud (visualized with Open3D [185]) and center votes overlaid on the image, which are color-coded according to their instance ID. Best viewed in color and zoomed in.

as there are no approximations made with discretized directions. Lastly, in columns 3, 4, and 6, we show that the 3D DSN architecture performs better due to a higher receptive field provided by the ESP modules. In columns 3 and 6, the long keyboards are only half segmented by UOIS-Net-2D, and in column 4, the binder (behind the cereal box) is also only half segmented.

Center Votes. As shown in Table 3.6, ℓ_{sep} is crucial to ensuring UOIS-Net-3D has strong performance. We visually examine the reason for this in Figure 3.9. In the second and third rows, we visualize the point cloud of the scene with the predicted center votes overlaid on the same image. Both the point cloud and center votes are color coded with respect to their instance segmentation masks. In the second row, we show point clouds/center votes from UOIS-Net-3D without training with ℓ_{sep}, and the third row is the full model. It is clear to see that when training without ℓ_{sep}, the center votes are more spread out and messy, which makes the post-processing GMS clustering more difficult. On the other hand, the full model has much more tightly

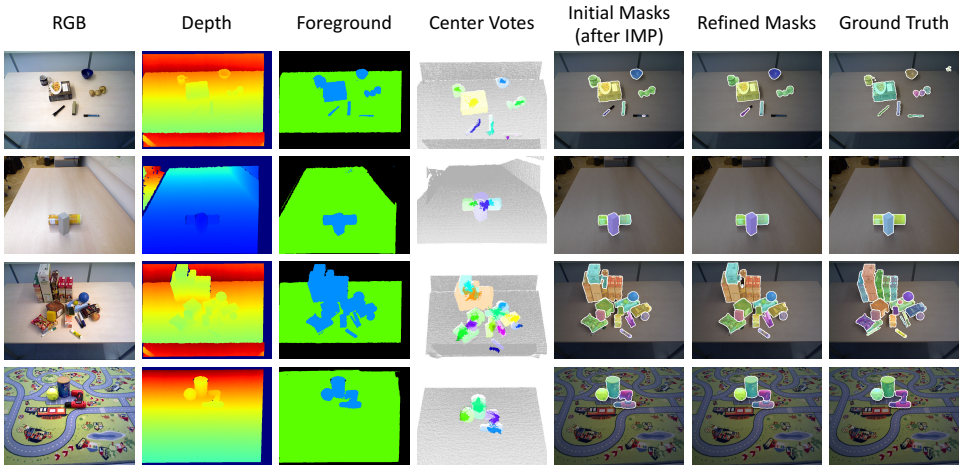


Figure 3.10: Common failure modes of UOIS-Net-3D. See text for details. Best viewed in color and zoomed in.

packed and separated center votes, which leads to better performance. Looking at example (column) 5, there are many small objects such as fruits and bananas which are close to each other in spatial proximity. UOIS-Net-3D trained without ℓ_{sep} shows that the center votes are mostly jumbled together, while the full model nicely separates the center votes which leads to almost perfect segmentation. Note that the center votes do not need to be at the center of the object, as long as they can be clustered correctly. In the upper right of example 2, the cylinder and the bowl have very close 3D centers. The 3D DSN predicts the center votes in a way that pushes them apart to ease the post-processing clustering and allows (the full) UOIS-Net-3D to correctly segment these objects.

Failure modes. In Figure 3.10, we demonstrate common failure modes of UOIS-Net-3D. Row 1 shows that when objects are close together, they may under-segmented into a single segment. The center votes shows that the three small fruits have center votes that are too close to separate in the post-processing clustering step. Another case of under-segmentation is shown in row 3, where multiple objects (cereal boxes) with flat surfaces are lined up such that the depth map shows a large flat surface. It is difficult to correctly

separate these boxes from depth alone; appearance is key in providing the correct segmentation in such situations. Row 4 shows that highly nonconvex objects such as power drills can be over-segmented into pieces. The center votes image clearly shows that the DSN believes this object to be two objects. Lastly, row 2 indicates that the 2D issue of attempting to segment a mask that is split into multiple pieces by an occluding object is still present when reasoning in 3D.

3.4.8 Quantifying Generalization from Sim to Real

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

Table 3.8: Performance on TOD test set (20k images).

In order to quantify generalization from simulation to the real-world, we show performance of our methods and SOTA methods Mask R-CNN and PointGroup on the TOD test set. This dataset of 20k images was generated using instances that are not present in the training set. In Table 3.8, we show that both Mask R-CNN and PointGroup outperform UOIS-Net on all metrics on this test set. However, as seen in Table 3.2, UOIS-Net outperforms Mask R-CNN and PointGroup on real-world data, indicating that our methods handle the distribution shift better, which is ultimately what we care about.

3.4.9 Application in Grasping Unknown Objects

We use our model to demonstrate manipulation of unknown objects in a cluttered environment using a Franka robot with panda gripper and wrist-mounted RGB-D camera. The task is to collect objects from a table and put them in a bin. Object instances are segmented using our method

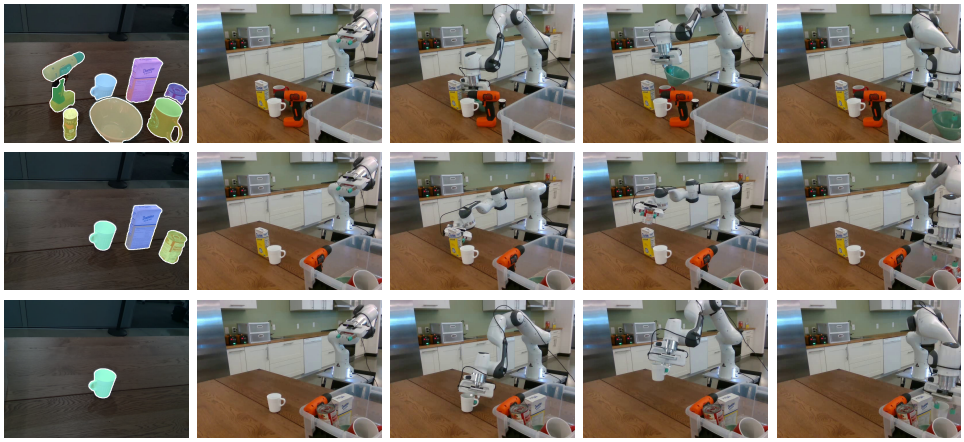


Figure 3.11: Visualization of clearing table using our instance segmentation and 6-DOF GraspNet [109].

and the point cloud of the closest object to the camera is fed to 6-DOF GraspNet [109, 110] to generate diverse grasps, with other objects considered obstacles. Other objects are represented as obstacles by sampling fixed number of points using farthest point sampling from their corresponding point cloud. The grasp that has the maximum score and has a feasible path is chosen to execute. Figure 3.11 shows the instance segmentation at different stages of the task and also the execution of the robot grasps. Video of the experiments can be found at the project website[2]. Our method segments the objects correctly most of the time but fails sometimes, such as the over-segmentation of the drill in the scene. Our method considers the top of the drill as one object and the handle as an obstacle. This is because the object is highly nonconvex as discussed in the previous section. We conducted the experiment to collect 51 objects from 9 different scenes. Each object is considered to be successfully grasped if the robot can pick it up in maximum 2 attempts. Otherwise, we count that object as failure case and remove it from the scene manually and proceed to other objects. In our trials, the robot successfully grasped 41/51 objects (80.3% success rate). The failures stem

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

from either imperfections in segmentation or inaccurate generated grasps. Note that neither our method nor 6-DOF GraspNet [109] are trained on real data.

3.5 Discussion

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

In parallel to our work presented in this chapter and in [178], we also worked on an alternate solution using a different network architecture. We introduced Unseen Clustering Network (UCN) in [173]. UCN adopts a standard encoder-decoder network architecture, much like UOIS-Net, and trains directly on the non-photorealistic TOD dataset introduced in Section 3.3. The key difference with UCN is that it utilizes a contrastive loss on pixel-dense embeddings. This loss function is almost identical to the one introduced in Section 2.2.4. UCN trains for a much longer time than UOIS-Net, and generalizes very well from simulation to the real world without any tricks, similarly to UOIS-Net. However, UCN directly predicts instance segmentations from RGB-D, which is in contrast to UOIS-Net which directly predicts instance segmentations from depth, and refines them with RGB. UCN gives state-of-the-art performance, and we build on top of the results in the next chapter.

While the results of UOIS-Net are quite promising, there is still lots of room for improvement. First, as discussed in Section 3.4.7, in heavily cluttered settings, our method tends to under-segment objects together as shown in example 1 of Figure 3.10. In the RGB image, there are clear delineations

of the different objects, whereas the depth may be less so. Perhaps then, a useful postprocessing step would be able to split the mask along those object boundaries. Secondly, another shortcoming of UOIS-Net is its absence of uncertainty. Like many deep learning-based segmentation algorithms, it produces a single output with a forward pass of the model. We explore these two threads in the next chapter, where we devise a novel method of sampling perturbations to initial instance segmentations to produce both refined instance segmentation masks and uncertainty estimates.

Chapter 4

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

This chapter discusses work originally published in Xie et al. [176].

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 [109, 110, 108] and re-arrangement [38]. Additionally, understanding the object identities is a useful piece of information when planning to re-arrange objects.

Many methods for UOIS directly predict segments from raw sensory input such as RGB and/or depth images. Recent methods, including UOIS-Net introduced in Chapter 3, have shown strong results for this problem [178, 173, 63, 72]. For example, Xiang et al. [173] showed that training a network on RGB-D with simulated data and a simple contrastive loss [42] can demonstrate strong results for this problem. While these methods show

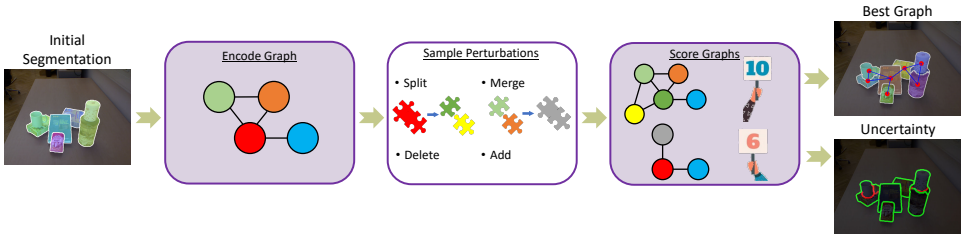


Figure 4.1: High-level overview of RICE. Given an initial segmentation, we encode it as a segmentation graph, sample perturbations, then score the resulting segmentation graphs. The highest scoring graph and/or contour uncertainty is output. Best viewed in color and zoomed in.

promise, they are not perfect and still admit mistakes in cluttered scenes, which are common in manipulation scenarios. This can hamper downstream robot tasks that rely on such perception.

A natural thought is that an architecture with relational reasoning can benefit the predictions. Graph neural networks (GNN) in vision and robotics have recently become a useful tool for learning relational representations. Motivated by the success of convolutional neural networks (CNNs) [93], GNNs were first introduced by [139], and many variants have been proposed that incorporate properties analogous to CNNs such as locality [82, 59], attention [164], and residual connections [94, 62]. They have found applications in many standard computer vision tasks such as image classification [55, 167], object detection [67], semantic segmentation [97], and question answering [138]. GNNs have also been used to perform "scene graph generation", which requires predicting not just object detections, but also the relations between the objects [180, 31, 182]. The resulting scene graphs have been used for applications such as image retrieval [73]. GNNs have also been used to learn object dynamics, properties, and relations for applications such as differential physics engines [9, 26]. While relational inductive biases have shown to be useful for many problems as discussed above, it remains to be seen whether it can be useful in identifying objects in dense clutter.

In this chapter, 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 4.1 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, which is a supporting insight as to why it aids performance. Additionally, we provide a proof-of-concept efficient scene understanding application that utilizes uncertainties output by RICE to guide a manipulator.

In summary, the main contributions in this chapter are:

- 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.
- Our method achieves state-of-the-art results for UOIS when combined with previous methods.
- We demonstrate that uncertainty outputs from our method can be used to perform efficient scene understanding.

4.1 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 4.1.1. Then, in Section 4.1.2, we build a tree T of sampled segmentation graphs by perturbing the leaves in a CEM-style [41] framework. Section 4.1.3 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 4.1.4. This process is repeated until a given budget is depleted (e.g. number of tree nodes). Finally, the highest scoring graph in T and/or contour uncertainties are returned. Figure 4.1 provides a high-level illustration of RICE, and pseudocode can be found in Algorithm 2.

4.1.1 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

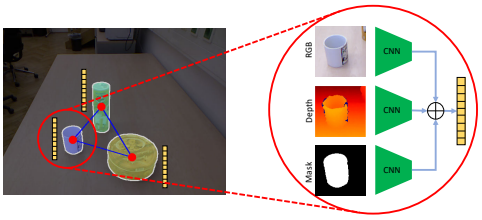


Figure 4.2: Given an initial instance segmentation mask (left), our segmentation graph representation encodes each individual mask as a graph node (red dots) with a corresponding feature vector $\mathbf{v}_i$ (yellow bar) output by the Node Encoder (right). Edges (blue lines) connect nearby masks.

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

node. See Figure 4.2 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.

4.1.2 Building the Sample Tree

Our sample tree-building procedure operates in a CEM-style fashion. CEM [41] 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 [41]. 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 4.1.3.

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 4.1.3 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 pseudocode in Algorithm 2 and an example of the sample tree-building procedure (Figure 4.3).

Given the fully-expanded sample tree T, we return the highest scoring graph. Since the process is stochastic, we can also compute uncertainty of the segmentation graphs. Each leaf is essentially a sampled trajectory of states (segmentations) and actions (perturbations) from the initial segmentation. These trajectories may have explored different parts of the state space (e.g. perturbed different object masks in the scene). We can average the contours of each leaf graph and compute the standard deviation in order to provide uncertainty estimates of the object contours.

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 4.2.5 shows that the combination of these leads to more accurate performance with lower variance.

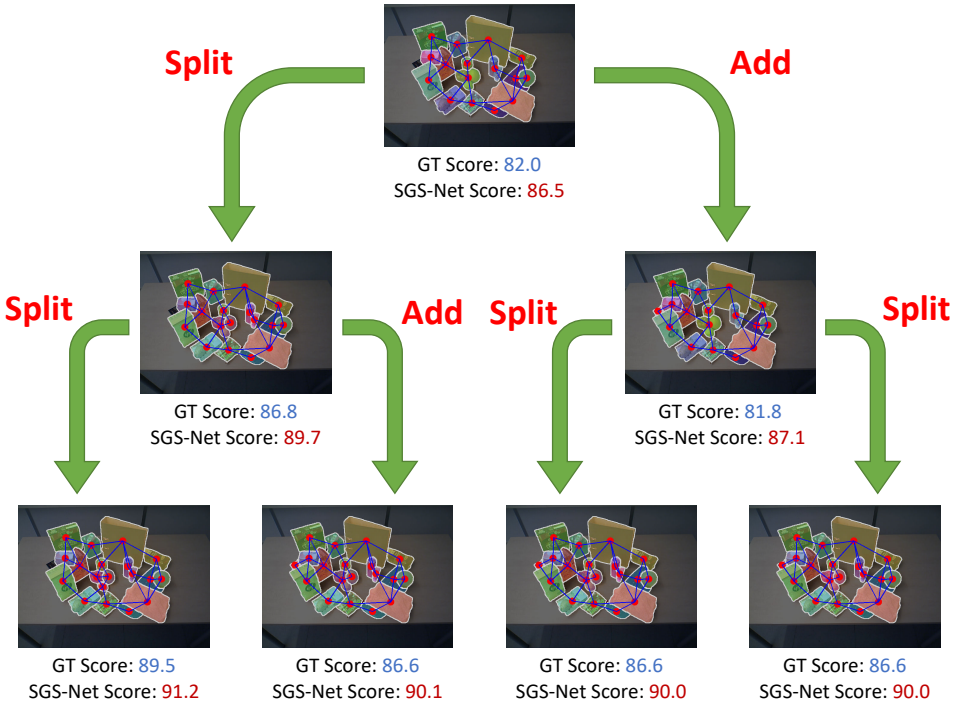
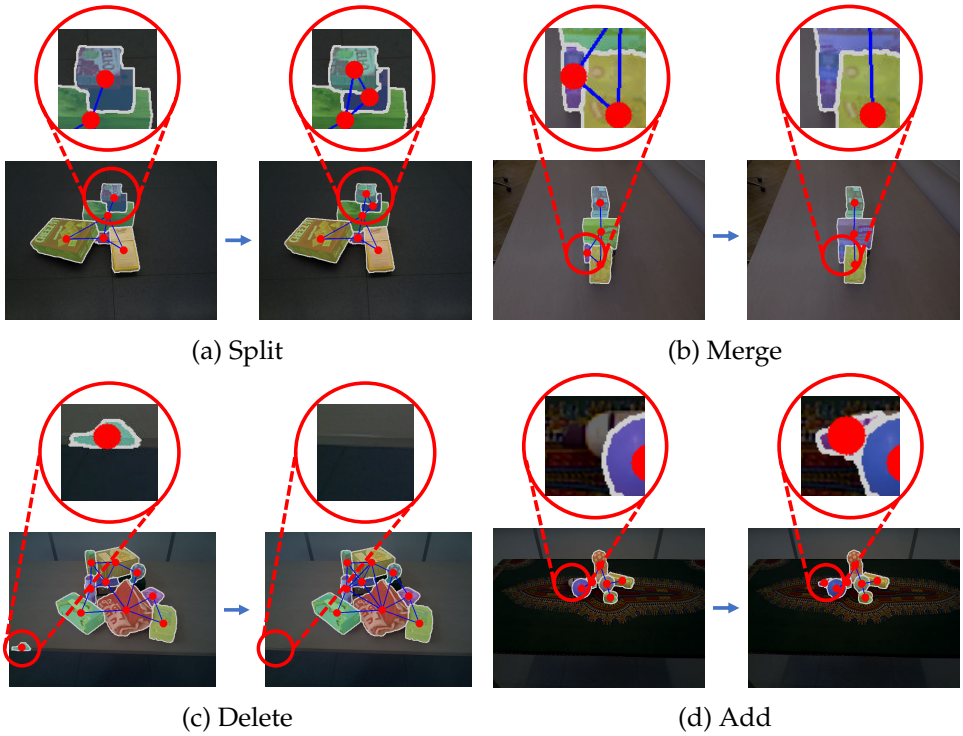


Figure 4.3: Example of a sample tree. Ground Truth and SGS-Net scores are shown, along with the chosen sampling operations. In this example, all leaves improve upon the initial segmentation graph, with the highest ranking graph also being the closest to the ground truth segmentation. Very similar splits and adds are investigated in the leaf trajectories.

4.1.3 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



(a) Split (b) Merge

(c) Delete (d) Add

Figure 4.4: 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.

operation can be found in Figure 4.4.

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 4.4a 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 [136] architecture, much like Y-Net [174], except that it has three streams for RGB, depth, and the mask. The output of SplitNet is a pixel-dense probability

Algorithm 3 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

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. Pseudocode for this operation can be found in Algorithm 3.

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 4.4b 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 4.4c 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 [178]), 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 4.4d.

To summarize, we characterize four sampling operations, which use two networks, SplitNet and DeleteNet, which we denote as our Sampling Operation Networks (SO-Nets).

4.1.4 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.

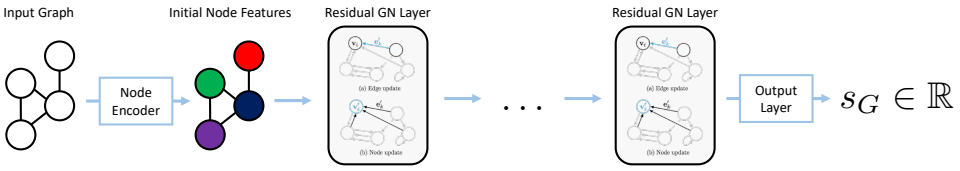


Figure 4.5: A high-level illustration of our Segmentation Graph Scoring Network (SGS-Net). It is composed of a Node Encoder (see Figure 4.2), multiple Residual GraphNet Layers, and an output layer. We borrowed elements from Figure 3 of Battaglia et al. [10].

A high-level illustration of SGS-Net can be found in Figure 4.5. 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 an adaptation of a GraphNet Layer [10] with residual connections. Our RGL first applies an edge update:

$$\mathbf{e}_{ij}^{(l+1)} = \mathbf{e}_{ij}^{(l)} + \phi_e^{(l)}\left(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}, \mathbf{e}_{ij}^{(l)}\right), \tag{4.1}$$

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

$$\mathcal{E}_i^{(l)} = \left\{\phi_{v_1}^{(l)}\left(\mathbf{e}_{ij}^{(l+1)}, \mathbf{v}_j^{(l)}\right) : (i,j) \in E\right\} \tag{4.2}$$

$$\mathbf{v}_i^{(l+1)} = \mathbf{v}_i^{(l)} + \phi_{v_2}^{(l)}\left(\overline{\mathcal{E}_i^{(l)}}, \mathbf{v}_i^{(l)}\right), \tag{4.3}$$

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:

$$s_G = \sigma\left(\phi_o\left(\overline{\mathcal{V}}, \overline{\mathcal{E}}\right)\right) \in [0,1], \tag{4.4}$$

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. The output of SGS-Net is a scalar score in $[0, 1]$.

4.1.5 Training Procedure

We use simple loss functions to train our networks. 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 $\hat{p}_u$, normalized to sum to 1. DeleteNet is also trained with standard BCE loss. For SGS-Net, we train it to regress to a certain score in the range $[0, 1]$. For a given segmentation, we compute the Overlap F-measure, F [175] and Overlap F@.75 measure [115], where the latter is a metric that indicates the percentage of correctly segmented objects with a certain accuracy. We train SGS-Net with $\ell_{SGS} = \ell_{bce}(s_G, \ .8F + .2F@.75)$, thus it 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 on this only (see line 8 of Algorithm 2). In Section 4.2.7 we show that while SGS-Net may not solve the regression problem well, it learns to rank graphs accurately.

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. [175].

Lastly, we use ResNet50 [62] combined with Feature Pyramid Networks [100] (FPN) to encode RGB images before passing them to the Node Encoder. However, since we are training with (a slightly more cluttered version of) the non-photorealistic synthetic dataset from Xie et al. [175], we perform modality tuning [4], where we fine-tune earlier convolutional layers of ResNet50 during training, and use the COCO [99] pretrained ResNet50 during inference. Modality tuning is performed during training of the SO-Nets, and held fixed during SGS-Net training. This helps with generalization from simulation to the real-world and we show the optimal amount of ResNet50 to modality tune for generalization in Section 4.2.3.

4.2 Experiments

4.2.1 Implementation Details

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 [81] 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.

4.2.2 Datasets and Metrics

We evaluate our method on two real-world datasets of challenging cluttered tabletop scenes: OCID [152] and OSD [135], 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) [175] (introduced in Chapter 3), where each scene has anywhere between 20 and 30 ShapeNet [24] objects. We use 20k scenes in total, with 5 images per scene.

Xie et al. [175] 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:

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

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. [175], we can apply the OSN metrics to the pixels and boundaries, giving us Overlap and Boundary $P_n/R_n/F_n$ measures.

Since RICE is a stochastic method, we run each experiment 5 times and show means and standard deviations for all metrics.

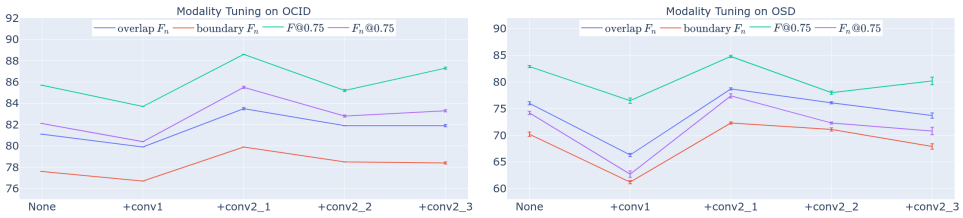


Figure 4.6: Modality tuning on OCID [152] and OSD [135] shows that tuning up to conv2_1 when training on simulated data generalizes best to real data. Note that standard deviation bars are shown, but are very tight and difficult to see.

4.2.3 Encoding RGB and Modality Tuning

Inspired by Aytar et al. [4], 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 [99], we ablate over tuning the `conv1` layer, the `conv2_1`, `conv2_2`, `conv2_3` bottleneck building blocks [62], or keeping ResNet fixed. The idea is that by fixing the rest of the layers, we can 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 4.6, 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.

4.2.4 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 [63], PointGroup [72], UOIS-Net-3D [178], and UCN [173]. We employ RICE by returning the best segmentation from the leaves as

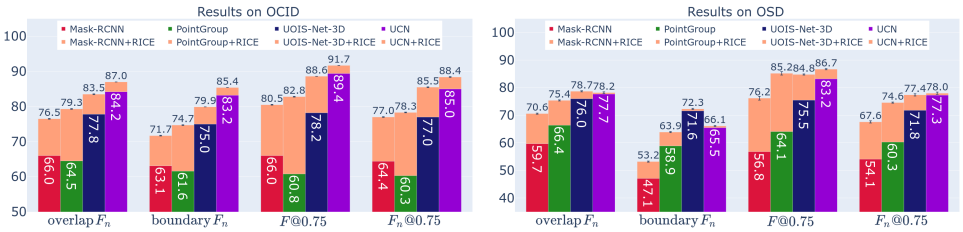


Figure 4.7: Applying RICE to refine results from state-of-the-art instance segmentation methods leads to improved performance across the board. Note that standard deviation bars are shown, but are very tight and difficult to see.

scored by SGS-Net. For brevity, we only show Overlap F_n, Boundary F_n, F@.75, and F_n@.75 in Figure 4.7 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).

4.2.5 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 segmen-

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 4.1: Ablation to test the utility of SO-Nets and SGS-Net on OCID [152] starting from UOIS-Net-3D [178] 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.

tation 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 [178], we see in Table 4.1 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_n@.75 is slightly lower with F@.75 higher, indicating that SO-Nets are suggesting more samples that better capture the objects, but are suggesting too many instance segments.

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 [173]	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 [178]	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 [63]	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 [72]	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 4.2: 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).

4.2.6 Evaluating the Usefulness of Each Sampling Operation

We provide another 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 [178], UCN [173]).

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 4.1.3). In Table 4.2, we show the results for F@.75 and F_n@.75 metrics using UCN [173], UOIS-Net-3D [178], Mask R-CNN [63], and PointGroup [72] 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 for this is that a common failure case from its pixel-clustering procedure is that the boundaries of the objects tend to be clustered as a separate object which results in over-segmentation. Merging can easily solve this issue. Lastly, Mask R-CNN and PointGroup also ben-

efit from delete/add, which suggests that they are either predicting false positives and/or false negatives.

4.2.7 SGS-Net Ranking

Figure 4.8 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 4.3. 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 4.1.2). We claim that SGS-Net learns to rank the graphs accurately, and design an experiment to test this hypothesis.

Figure 4.8: Can you spot the differences between the segmentations?

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

Table 4.3: Ranking study on OCID and OSD.

We leverage the normalized Discounted Cumulative Gain (nDCG) [71] 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 4.2.5, with relevance values in $\{0, ..., K\}$. The ranking for SO-Nets is given 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 4.3, 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.

4.2.8 Visualizing Refinements

In the left side of Figure 4.9 (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 as discussed in Section 4.2.6. 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). These examples demonstrate RICE's capabilities to split, merge, delete, and add masks appropriately to refine initial instance masks.

4.2.9 Failures and Limitations

In the right side of Figure 4.9 (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 [178]. The second

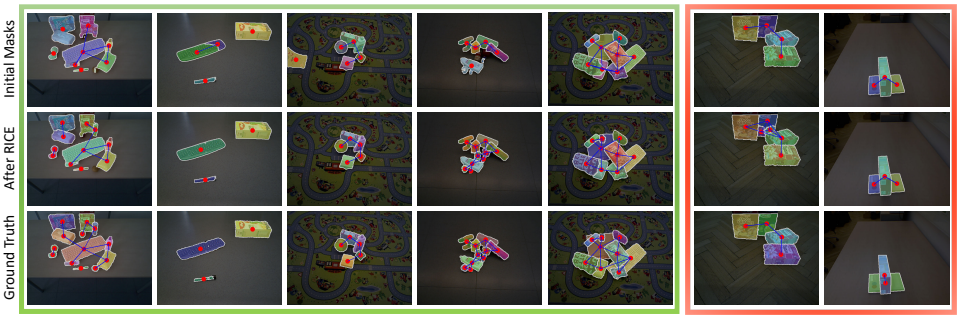


Figure 4.9: We demonstrate successful refinements (left, green box) for each of the sampling operations. Failure modes (right, red box) include textured objects and non-neighboring masks that belong to the same object. Best viewed in color and zoomed in on a computer screen.

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.

4.2.10 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 [25]. 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 [155] 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

Algorithm 4 Guiding a Manipulator with RICE

```
repeat
    Get S from initial instance segmentation method (e.g. UCN [173])
    Run RICE to get best masks and contour uncertainty
    if Contour Uncertainty is present then
        Sample a grasp with Contact-GraspNet [155] from the uncertain
        masks, and execute it
    end if
until No Contour Uncertainty
```

required to resolve the uncertain portions of the scene, which can potentially be much less than the number of objects, leading to a more efficient scene understanding method. For example, in Figure 4.10, only two grasps are required to fully understand the scene. A video at the project website[1] shows the full set of grasping trials.

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 4.10. 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 video, the cup and the bowl it is on top of (far left) are constantly

[1] https://github.com/chrisdxie/rice

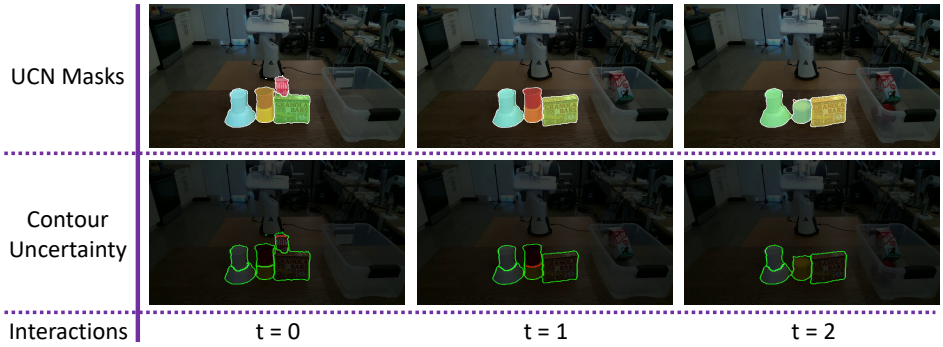


Figure 4.10: UCN masks [173] and contour uncertainties from RICE in a trial of our scene understanding experiment. Uncertainties are shown in red with average contours in green. After grasping the milk carton and red cup, the scene is segmented with full certainty, indicating that the scene is fully understood. Thus, the algorithm terminates without having to singulate each object. Best viewed in color and zoomed in.

under-segmented together by UCN [173]. RICE splits it correctly each time, and there is no uncertainty about splitting that mask, as evidenced by the uncertainty contours. Pseudocode of the grasping algorithm can be found in Algorithm 4.

4.3 Discussion

In this chapter, we proposed a novel framework that utilizes a new 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.

Our proposed work comes with limitations (aside from those discussed in Section 4.2.9). One of the main ones 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. At a higher-level, the uncertainty estimates that we generate from RICE are rudimentary and do not interact with the grasping algorithm in any way. For example, grasping some parts of the scene, such as the objects at the bottom of a pile, may result in large changes to the current scene structure that may not be ideal. An improved model would produce uncertainty estimates that not only convey the uncertainty of the segmentation from a vision perspective, but also which grasping (or other interaction primitives) actions would best allow for understanding the scene, such as maximizing information gain [87].

In the next chapter, we discuss future directions that encompass such ideas.

Chapter 5

Conclusion

As we begin to deploy robots into unstructured environments such as homes and/or offices, they will need to be equipped with the ability to perceive and reason about unseen objects. Different from "in-the-wild" computer vision, this will require that any potential object of interest, including the unseen objects, be identified as opposed to a pre-defined set of object categories. As this problem is quite difficult, methods with accompanying uncertainty estimates are desirable in assisting robots with choosing safe and efficient actions. Solutions based on data-hungry deep learning methods, which recently have surged to become the dominant and most successful methods for solving visual perception problems, will require large amounts of data for training. Collecting such datasets in the real-world is both expensive and time-consuming as it requires a data collection phase along with a manual annotation phase. Thus, it is appealing to look towards utilizing large amounts of synthetic data, however the Sim-to-Real gap is another hurdle that must be overcome in this case.

5.1 Contributions

In this thesis, we designed solutions for the problem of detecting and segmenting unseen object instances based on deep networks primarily trained on large scale synthetic datasets. Our proposed methods investigated the

use of different visual cues including motion cues (in the form of optical flow) and geometry cues (in the form of depth maps). We designed our networks to be able to generalize from simulation to the real world with minimal adaptations.

Beginning with Chapter 2, we presented a novel neural network architecture, denoted Pixel Trajectory Recurrent Neural Network (PT-RNN), that primarily exploited motion cues in order to segment any entity in a video that can move or be moved. PT-RNN is designed based on pixel trajectories, which are pixels in time-adjacent frames that are linked together by optical flow (extracted from an external method such as Flow-Net-2 [69]). The problem was formulated as foreground motion clustering, where feature embeddings of the pixel trajectories are clustered in order to reveal the different object instances. Unseen objects can then be discovered as long as they exhibit coherent motion evidenced by optical flow. We pre-trained PT-RNN on a large-scale synthetic dataset of flying chairs and showed that this pre-training was necessary in order to achieve state-of-the-art results on multiple motion segmentation datasets.

However, a perception module that requires each object to exhibit motion before recognizing it will not scale to a large number of objects, as the robot will have to physically manipulate all inanimate objects of interest before applying such a perception module. Thus, in Chapter 3, we introduced UOIS-Net, a method that instead exploits geometry cues to segment unseen objects from static RGB-D images. UOIS-Net is designed to separately leverage RGB and Depth in a way that allows for training on non-photorealistic synthetic data while generalizing to real-world settings without any fine-tuning. To train the method, we generated a large-scale non-photorealistic dataset of random objects on random tabletops. UOIS-Net is able to detect and segment arbitrary unseen objects in tabletop environments, which are common to robot manipulation scenarios. We demonstrated state-of-the-art performance on multiple tabletop object datasets, along with the ability to clear a table of unseen objects by combining the outputs of UOIS-Net with a grasping algorithm [109].

The strong performance and ease-of-use of UOIS-Net has facilitated

further research of robot manipulation of unseen objects. In the original published work [175], we demonstrated grasping of unseen objects for table clearing. This has been extended to grasping in clutter [110], and further advancements in the grasping community has also built off of unseen object instance segmentations [155]. Reconstruction of unseen objects for general manipulation planning including grasping, pushing and re-arrangement has also been explored [2]. Re-arrangement of unseen objects in particular has seen some interesting advancements [132, 39]. Additionally, UOIS-Net has been utilized as a core piece of interesting solutions to the difficult problem of unseen object pose estimation [117].

While UOIS-Net and competitor methods have shown promising performance for segmenting unseen objects, they still tend to fail in heavily cluttered scenes and do not provide uncertainty estimates. To combat this, in Chapter 4, we introduced a novel framework for refining initial instance masks, called RICE. RICE consumes an initial instance mask output by a method such as UOIS-Net, and samples perturbations to it in order to refine it. This is done in a novel graph-based representation, where a graph neural network is trained to evaluate the perturbed instance masks, and sampling operation deep networks are learned to suggest smart perturbations. RICE outputs both the best scoring graph according to the GNN and an uncertainty estimate of the instance masks, which we showed could be used for an efficient scene understanding application. Similarly to UOIS-Net, we trained RICE purely on non-photorealistc synthetic data, using modality tuning [4] as a means for generalizing from simulation to the real world. We demonstrated that RICE improves performance amongst multiple direct segmentation methods, including UOIS-Net.

In summary, we have designed multiple solutions to attack the problem of discovering and segmenting unseen object instances for robot perception. The work in this thesis is just the beginning of building and learning perceptual systems that imbue robots with the ability to perceive and reason about unseen objects in unstructured settings. We conclude this thesis with a discussion of interesting future directions of this work.

5.2 Future Directions

There are many interesting avenues of future work in the realm of perceiving and reasoning about unseen objects. We discuss a few here:

5.2.1 Incorporating Interaction and Self-Supervised Learning

In Chapter 4, we briefly demonstrated an interactive application that performs scene understanding by resolving uncertainties output by RICE. The manipulation of the scene can lead to valuable signals for training, which we did not make use of in our proof-of-concept demonstration. For example, interactions such as poking or grasping will produce motion in the scene. This signal can be potentially exploited by ideas such as those from Chapter 2 to help learn where the original segmentation made mistakes. Additionally, constantly re-arranging the scene to not only induce motion, but view the objects from different viewpoints and poses can lead to more training data to fine-tune the network weights. This idea bares many similarities to the self-supervised pipeline of Deng et al. [44]. Such an interactive feedback loop can allow the robot perception system to overfit to a particular scene. Although overfitting typically has a negative connotation in machine learning, in this situation it would be advantageous since the robot will be performing tasks in the same environment (e.g. a kitchen cleaning robot), and overfitting to the objects present in the scene will only make the perception of the specific scene more accurate.

5.2.2 Better Uncertainty Estimate Representations

Obtaining uncertainty estimates from deep neural networks has recently been gaining more attention. A couple of methods of doing this include Monte Carlo dropout as an approximation to the posterior distribution [54] and training multiple networks on the same dataset in parallel [92]. These ideas have been used to generate uncertainty estimates in computer vision and robot perception tasks [3, 76, 102].

These methods, similar to RICE, tend to give uncertainty predictions at

the pixel level. However, instance segmentations are at the object level, and there arises a question of whether uncertainty would be more useful at the object level? In this case, these object-level uncertainties could potentially be predicted in a way that helps the robot select an action to resolve the uncertainty. For example, in our efficient scene understanding application in Chapter 4, it would make sense to manipulate the objects that have the most uncertainty and that do not support the pile of objects, so that the induced motion in the scene is restricted to only the object of interest. On the other hand, pixel-dense uncertainty allows the robot to focus on parts of objects that are more uncertain than others, which could also be useful in tasks such as 3D object reconstruction and understanding. Finding the right representation for uncertainty in segmentation masks remains an open problem.

In addition to the many questions yet to be answered in this domain, there are many question yet to be discovered as well. Eventually, robots should be able to leverage reliable uncertainty estimates produced by deep learning perception systems.

5.2.3 Generalizing Outside of Tabletop Settings

An ideal ability for robots is to be able to recognize unseen objects in any environment. In the majority our work (Chapters 3 and 4), we focused on unseen objects in tabletop environments. We dealt with objects that were graspable and on a planar surface, thus our definition of "unseen object" was easily defined. However, the definition of "object" can span many different definitions, which likely will depend on the application of interest. For example, COCO [99] object classes include small items such as silverware (fork, knife, cup, and spoon) and larger classes such as vehicles (car, motorcycle, airplane, and bus), while KITTI [56] includes object classes that autonomous vehicles would care about such as cars, trucks, and pedestrians. In these settings, it is more difficult to define "unseen object" as there is a lack of similarity and/or contextual structure amongst the objects. Thus, there is an open question of how to define "unseen objects" in general

settings. Perhaps an interesting way of extracting similarity amongst a known set of objects and generalizing that to unseen objects is to develop a method that can detect whether a potential object is similar to the known set of classes but is not confident enough to classify it, which bears ideas that are somewhat similar to ShapeMask [89].

5.2.4 Connecting Actions to Segmentations

The intermediate outputs of robotic vision (e.g. predicted instance masks, or depth maps) ultimately will be used to compute actions in real-world settings [154]. In all of the work we presented in this thesis, we have not considered linking our segmentation masks directly with actions. An interesting avenue of future work involves predicting unseen object instance masks such that a corresponding downstream task is easy to solve. For example, if the task is grasping, then only grasp points (e.g. a cup handle) needs to be segmented with high accuracy. To set down an object on a table safely, the robot needs to understand the spatial extent of the bottom of the object so that it does not crash the object into the table at a high velocity. Additionally, segmenting parts of the object that may potentially be stable hyperplanes can allow robots to safely put down objects in a variety of configurations, which can aid in packing tasks. The idea of using downstream tasks to inform the intermediate representation has shown interesting results in grasping [48], however has yet to be applied to segmentation. Such an idea can be combined with affordance-based segmentation [45] to further enhance the information present.

5.2.5 Lifelong Learning

When a robot is first deployed in a new environment and sees an unseen object for the first time, technically that is the only time the object is unseen. All subsequent times that the object is observed, it will have been seen in a previous interaction phase, thus it should no longer be unseen. Thus, it makes sense then that when perceiving this object, the robot should no longer rely on methods such as the ones presented in this thesis, but instead

one that incrementally builds up specific detection models over time. In other words, the robot should follow a lifelong learning approach [158] that renders unseen objects as no longer unseen. Examples of interesting directions here involve understanding the object from multiple viewpoints with object-specific segmentors [172], few-shot learning approaches [168], and 3D object understanding of these unseen objects [2]. This can even be applied to understand an entirely new object class [177] if the set of unseen objects exhibits a similar structure. The work presented in this thesis provides the basis for which such methods can be developed for unseen objects in order to render them as no longer unseen.

Final Thoughts

In order for us to see more robots operating in our homes, they will need the ability to recognize and manipulate unseen objects. This thesis has put forward a few initial attempts at solving this while assuming certain problem structures for the solutions to exploit. While the results are promising, there is much work to be done and it will be exciting to see the development of more methodologies for this research question. We hope to see future research that builds off of our presented work, especially the ones such as incremental or lifelong learning that will eventually make the central problem of this thesis obsolete.

Bibliography

[1] Daron Acemoglu and Pascual Restrepo. Robots and jobs: Evidence from us labor markets. *Journal of Political Economy*, 128(6):2188–2244, 2020.

[2] William Agnew, Christopher Xie, Aaron Walsman, Octavian Murad, Caelen Wang, Pedro Domingos, and Siddhartha Srinivasa. Amodal 3d reconstruction for robotic manipulation via stability and connectivity. In *Conference on Robot Learning (CoRL)*, 2020.

[3] Vijay Badrinarayanan Alex Kendall and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *British Machine Vision Conference (BMVC)*, 2017.

[4] Yusuf Aytar, Lluis Castrejon, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Cross-modal scene networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2303–2314, 2017.

[5] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 2011.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39 (12):2481–2495, 2017.

[7] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *International Conference on Learning Representations (ICLR)*, 2016.

[8] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12 (1):43–77, 1994.

[9] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29:4502–4510, 2016.

[10] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[11] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[13] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 2011.

[14] Pia Bideau and Erik Learned-Miller. A detailed rubric for motion segmentation. *arXiv preprint arXiv:1610.10033*, 2016.

[15] Pia Bideau and Erik Learned-Miller. It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In *European Conference on Computer Vision (ECCV)*, pages 433–449. Springer, 2016.

[16] Pia Bideau, Aruni RoyChowdhury, Rakesh R Menon, and Erik Learned-Miller. The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[18] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296, 2008.

[19] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[20] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision (ECCV)*, 2010.

[21] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *IEEE Conference on Robotics and Automation (ICRA)*, 2017.

[22] Arunkumar Byravan, Felix Leeb, Franziska Meier, and Dieter Fox. Se3-pose-nets: Structured deep dynamics models for visuomotor planning and control. In *IEEE Conference on Robotics and Automation (ICRA)*, 2018.

[23] Miguel A Carreira-Perpinán. A review of mean-shift algorithms for clustering. *arXiv preprint arXiv:1503.00687*, 2015.

[24] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012, 2015.

[25] Lillian Chang, Joshua R Smith, and Dieter Fox. Interactive singulation of objects from a pile. In *2012 IEEE International Conference on Robotics and Automation*, 2012.

[26] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations, (ICLR)*, 2017.

[27] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. Blendmask: Top-down meets bottom-up for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40 (4):834–848, 2017.

[29] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference Computer Vision (ECCV)*, 2018.

[31] Vincent S Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Re, and Li Fei-Fei. Scene graph prediction with limited labels. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[32] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[33] Jian Cheng, Jie Yang, Yue Zhou, and Yingying Cui. Flexible background mixture models for foreground segmentation. *Image and Vision Computing*, 2006.

[34] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[35] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[36] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[37] Michael Danielczuk, Matthew Matl, Saurabh Gupta, Andrew Li, Andrew Lee, Jeffrey Mahler, and Ken 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.

[38] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object rearrangement using learned implicit collision functions. *arXiv preprint arXiv:2011.10726*, 2020.

[39] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object rearrangement using learned implicit collision functions. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[40] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting everything that moves. *arXiv preprint arXiv:1902.03715*, 2019.

[41] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[42] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[44] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[45] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *2018 IEEE international conference on robotics and automation (ICRA)*, 2018.

[46] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] Alon Faktor and Michal Irani. Video segmentation by non-local consensus voting. In *British Machine Vision Conference (BMVC)*, 2014.

[48] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020.

[49] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[50] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[52] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[53] Katerina Fragkiadaki, Geng Zhang, and Jianbo Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[54] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016.

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

[56] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun.

Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[57] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[58] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference Computer Vision (ECCV)*, 2014.

[59] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin*, 2017.

[60] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[61] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[64] Tomáš Hodaň, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. *IEEE International Conference on Image Processing (ICIP)*, 2019.

[65] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[66] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[67] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[68] Weiming Hu, Xi Li, Xiaoqin Zhang, Xinchu Shi, Stephen Maybank, and Zhongfei Zhang. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *International Journal of Computer Vision*, 2011.

[69] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[70] Suyog Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv preprint arXiv:1701.05384*, 2017.

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

[72] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[73] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using

scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[74] Zdenek Kalal, Krystian Mikolajczyk, Jiri Matas, et al. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 2012.

[75] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[76] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[77] Margret Keuper. Higher-order minimum cost lifted multicuts for motion segmentation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4252–4260, 2017.

[78] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[79] Margret Keuper, Siyu Tang, Bjorn Andres, Thomas Brox, and Bernt Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[80] John Maynard Keynes. Economic possibilities for our grandchildren. In *Essays in persuasion*, pages 321–332. Springer, 2010.

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

[82] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[83] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[84] Takumi Kobayashi and Nobuyuki Otsu. Von mises-fisher mean shift for clustering on a hypersphere. In *International Conference on Pattern Recognition (ICPR)*, 2010.

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

[86] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[87] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *IEEE Conference on Robotics and Automation (ICRA)*, 2011.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[89] Weicheng Kuo, Anelia Angelova, Jitendra Malik, and Tsung-Yi Lin. Shapemask: Learning to segment novel objects by refining shape priors. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[90] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[91] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *IEEE Conference on Robotics and Automation (ICRA)*, 2014.

[92] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[93] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[94] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[95] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[96] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *European Conference Computer Vision (ECCV)*, 2018.

[97] Xiaodan Liang, Liang Lin, Xiaohui Shen, Jiashi Feng, Shuicheng Yan, and Eric P Xing. Interpretable structure-evolving lstm. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[98] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[99] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference Computer Vision (ECCV)*, 2014.

[100] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object

detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[101] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[102] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[103] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017.

[104] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[105] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[106] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *European Conference Computer Vision (ECCV)*, 2018.

[107] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[108] Chaitanya Mitash, Rahul Shome, Bowen Wen, Abdeslam Boularias, and Kostas Bekris. Task-driven perception and manipulation for constrained placement of unknown objects. *IEEE Robotics and Automation Letters*, 2020.

[109] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[110] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter. In *IEEE Conference on Robotics and Automation (ICRA)*, 2020.

[111] Manjunath Narayana, Allen Hanson, and Erik Learned-Miller. Coherent motion segmentation in moving camera videos using optical flow orientations. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[112] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference Computer Vision (ECCV)*, 2012.

[113] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[114] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Semi-convolutional operators for instance segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[115] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 2014.

[116] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[117] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[118] Deepak Pathak, Ross B Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[119] Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[120] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[121] Trung T Pham, Thanh-Toan Do, Niko Sünderhauf, and Ian Reid. Scenecut: Joint geometric and object segmentation for indoor scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[122] Pedro O. Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[123] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision (ECCV)*, 2016.

[124] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech

Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems (RSS)*, 2018.

[125] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[126] Ekaterina Potapova, Andreas Richtsfeld, Michael Zillich, and Markus Vincze. Incremental attention-driven object segmentation. In *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[127] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[128] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[129] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[130] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[131] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[132] Ahmed H Qureshi, Arsalan Mousavian, Chris Paxton, Michael C Yip, and Dieter Fox. Nerp: Neural rearrangement planning for unknown objects. In *Robotics: Science and Systems (RSS)*, 2021.

[133] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[134] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[135] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[136] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[137] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. In *Robotics: Science and Systems(RSS)*, 2017.

[138] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[139] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[140] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.

[141] Daniel Seita, Nawid Jamali, Michael Laskey, Ron Berenstein, Ajay Kumar Tanwani, Prakash Baskaran, Soshi Iba, John Canny, and Ken

Goldberg. Robot bed-making: Deep transfer learning using depth sensing of deformable fabric. *arXiv preprint arXiv:1809.09810*, 2018.

[142] Jean Serra. *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

[143] Burr Settles. Active learning literature survey. 2009.

[144] Lin Shao, Parth Shah, Vikranth Dwaracherla, and Jeannette Bohg. Motion-based object segmentation based on dense rgb-d scene flow. In *IEEE Robotics and Automation Letters*, volume 3, pages 3797–3804. IEEE, October 2018.

[145] Lin Shao, Parth Shah, Vikranth Dwaracherla, and Jeannette Bohg. Motion-based object segmentation based on dense rgb-d scene flow. *IEEE Robotics and Automation Letters*, 3:3797–3804, 2018.

[146] Lin Shao, Ye Tian, and Jeannette Bohg. Clusternet: 3d instance segmentation in rgb-d images. *arXiv preprint arXiv:1807.08894*, 2018.

[147] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017.

[148] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems (NIPS)*, 2014.

[149] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[150] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[151] Julian Straub. *Nonparametric Directional Perception*. PhD thesis, Massachusetts Institute of Technology, 2017.

[152] Markus Suchi, Timothy Patten, and Markus 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.

[153] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision (ECCV)*, 2010.

[154] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[155] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Fox Dieter. Contact-graspnet: Efficient 6-dof grasp generation in cluttered-scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[156] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[157] Brian Taylor, Vasiliy Karasev, and Stefano Soatto. Causal video object segmentation from persistence of occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[158] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15:25–46, 1995.

[159] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[160] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[161] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning motion patterns in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[162] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, 2018.

[163] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, and Trevor Darrell. Adapting deep visuomotor representations with weak pairwise constraints. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[164] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[165] Weiyue Wang and Ulrich Neumann. Depth-aware cnn for rgb-d segmentation. In *European Conference Computer Vision (ECCV)*, 2018.

[166] Wenguan Wang, Jianbing Shen, Jianwen Xie, and Fatih Porikli. Supertrajectory for video segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1671–1679, 2017.

[167] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[168] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[169] Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018.

[170] Yu Xiang and Dieter Fox. Da-rnn: Semantic mapping with data associated recurrent neural networks. In *Robotics: Science and Systems (RSS)*, 2017.

[171] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[172] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[173] Yu Xiang, Christopher Xie, Arsalan Mousavian, and Dieter Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2020.

[174] Christopher Xie, Yu Xiang, Zaid Harchaoui, and Dieter Fox. Object discovery in videos as foreground motion clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[175] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2019.

[176] Christopher Xie, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Rice: Refining instance masks in cluttered environments with graph neural networks. *arXiv preprint arXiv:2106.15711*, 2021.

[177] Christopher Xie, Keunhong Park, Ricardo Martin-Brualla, and Matthew Brown. Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling. *arXiv preprint arXiv:2104.08418*, 2021.

[178] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. Unseen object instance segmentation for robotic environments. *IEEE Transactions on Robotics (T-RO)*, 2021.

[179] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[180] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[181] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations, (ICLR)*, 2016.

[182] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[183] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. In *Australasian Conference on Robotics and Automation (ACRA)*, 2015.

[184] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[185] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.