Title

Object-Centric Perception for Real-World Robotics

Permalink

https://escholarship.org/uc/item/89z3q4b6

Author

Mishra, Nikhil

Publication Date

2024

Peer reviewed|Thesis/dissertation

Object-Centric Perception for Real-World Robotics

By

Nikhil Mishra

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair
Professor Angjoo Kanazawa
Professor Trevor Darrell
Igor Mordatch

Spring 2024

Object-Centric Perception for Real-World Robotics

Abstract

Object-Centric Perception for Real-World Robotics

by

Nikhil Mishra

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Deep learning has resulted in incredible progress in many applications of artificial intelligence. However, these techniques often fall short when applied to robotics, due to their inability to reason about the ambiguity that often arises in the real world. Much of this ambiguity stems from the real world's long-tail visual diversity – in particular, the huge variety of objects that robots must interact with. Such shortcomings are only exacerbated by the strict requirements for autonomous, high-throughput operation that deployed systems must meet, as well as the cost and difficulty of obtaining the large-scale training datasets that modern deep learning methods require.

In this thesis, we explore two primary avenues of addressing these challenges. First, we introduce models that can better express uncertainty in challenging or ambiguous situations, across a variety of 2D and 3D perception tasks. Real-world robots can incorporate these models to reason explicitly about ambiguity, in flexible ways depending on their specific tasks. Second, we extend the capabilities of neural renderers to develop a sim2real2sim method that can drastically reduce the amount of data needed to train such models. From only a handful of in-the-wild examples, our method learns to generate synthetic scenes, targeted to specific real objects and environments, that can be used to train downstream perception models for a variety of tasks.

To all my friends, family, collaborators, and the Covariant team.

Contents

Chapter 1

Introduction

1.1 Robotics for the Real-World

Although robots have been used in key manufacturing segments for many decades, they have largely been constrained to carefully structured settings that do not demand much intelligence from them. With traditional automation techniques, a robot's behavior often is fully specified in advance, to the extent that it can simply execute pre-programmed motions blindly and repetitively. However, there is huge and growing demand for robotic systems that can intelligently perceive their environments and adapt their behavior accordingly. In particular, industrial applications in logistics and e-commerce seek robots that can perform general-purpose manipulation in more dynamic environments, but these applications are especially challenging due to the long-tail visual diversity of the objects and environments the robots must interact with. The performance requirements are often extremely stringent, as a deployed system needs to operate autonomously while achieving high accuracy and throughput, keeping pace with highly-optimized industrial processes or human co-workers. Recent work incorporating artificial intelligence (AI) into robotics has largely focused on learning specialized motor skills through reinforcement learning, or on natural language as a means of task specification. Unfortunately, this work is not immediately applicable to real-world industrial robotics, where the task description is simple but the tolerance for errors is almost zero – rather, the challenge lies in the need for robust perceptual generalization to unseen scenarios and ambiguous situations.

This thesis investigates how we can extend modern deep learning methods for perception to enable robots to succeed in real-world applications. We identify the ability to reason about uncertainty in ambiguous situations as a critical capability where state-of-the-art methods fall short, and propose novel models to address this shortcoming. These models allow deployed robotic systems to explicitly express their uncertainty in the face of real-world ambiguity, and adapt their behavior in flexible ways depending on the particulars of their task. Although reasoning about such ambiguity vastly improves a deployed system's ability to operate autonomously, the long-tail variability of the real world also necessitates systems that can continue to learn from their experience. However, such methods still require substantial amounts of training data, which may be impractically costly or difficult to obtain. To overcome this challenge, we develop a neural rendering method that can improve sim-to-real transfer by learning from real-world examples to synthesize targeted training data for a variety of perception tasks and models.

Robots have enormous potential to tangibly impact our society, and we hope that the contributions presented in this thesis help further the development of robust, intelligent robotic systems. We simultaneously hope that the lessons learned and techniques developed from this practical application of deep learning prove valuable to the broader AI community.

1.2 Overview

In the first half of this thesis, we propose novel perception models that can better reason about uncertainty in challenging or ambiguous situations. We study instance segmentation and 3D object detection, two popular perception tasks that exhibit substantial ambiguity and where state-of-the-art methods fall short. In the second half, we explore methods for improving sim-to-real transfer as a way to reduce the amount of real-world supervision needed to train such perception models. We propose a sim2real2sim method to automatically reduce the sim-to-real gap by learning object-centric NeRF representations from real data, which can then be composed in arbitrary ways to generate targeted synthetic data of various modalities.

Distributional Instance Segmentation with Latent-Mask-RCNN

Instance segmentation is the cornerstone of the perception stack for many robotics applications. By enumerating the objects present in a scene, it offers a powerful inductive bias for downstream components. However, since an instance segmentation model must predict a set of high-dimensional masks, state-of-the-art approaches utilize specialized model architectures and loss functions that are not amenable to expressing uncertainty over possible hypotheses. We propose a class of instance segmentation models that use VAE-style latent codes to more expressively model distributions over sets of object masks, and introduce Latent-MaskRCNN as a particular instantiation of this model family. During inference, Latent-MaskRCNN can sample multiple plausible segmentations, and the predictions can be incorporated in arbitrary ways depending on the downstream application. We release a real-world dataset of ambiguous scenes from an apparel picking application, and show that Latent-Mask-RCNN outperforms existing segmentation methods and drastically improves the overall task success rate. This section represents work published in: *YuXuan Liu et al. "Distributional Instance Segmentation: Modeling Uncertainty and High Confidence Predictions with Latent-MaskRCNN". International Conference on Robotics and Automation (ICRA), 2023.*

Autoregressive 3D Bounding Box Prediction

3D bounding boxes are another popular intermediate representation for robotics applications, and their prediction suffers from many of the same challenges as instance segmentation. We propose to model the distribution over 3D bounding boxes autoregressively, which enables a more nuanced understanding of the underlying uncertainty. We demonstrate the performance of our autoregressive bounding box model on popular benchmarks including SUN-RGBD [86], ScanNet [15], and KITTI [20] and show that it achieves state-of-the-art performance while also learning a well-calibrated sense of uncertainty. Like Latent-MaskRCNN, our model admits flexible sampling schemes that can be tailored to asymmetric error tolerances. We also develop and release a robotics-focused simulated dataset, COB-3D, which exhibits objects in arbitrary 3D rotations, unlike the 1D rotations common in prior work. This section represents work published in: *YuXuan Liu et al. ”Autoregressive Uncertainty Modeling for 3D Bounding Box Prediction”. European Conference on Computer Vision (ECCV), 2022.*

Shape Completion Models for Dense Packing of Complex, Novel Objects

Although instance segmentation and 3D bounding box prediction typically rely on human annotators to provide supervision, many other perception tasks require dense supervision that is impossible to obtain in the real world. For example, shape completion seeks to predict the complete 3D geometry of an object, which is inherently ambiguous due to partial observability. We release COB-3D-v2, an expansion of COB-3D with additional modalities to make it suitable for for sim-to-real transfer of shape completion models, and propose Frustum Convolutional Occupancy Networks (F-CON), a simple model architecture that outperforms state-of-the-art models for shape completion. We demonstrate F-CON’s effectiveness in a real-world bin packing application, a manipulation task which prior work has mostly studied in simulation due to real-world perceptual challenges. This section represents work published in: *Nikhil Mishra et al. ”Convolutional Occupancy Models for Dense Packing of Complex, Novel Objects”. International Conference on Intelligent Robots and Systems (IROS), 2023.*

Closing the Visual Sim-to-Real Gap with Object-Composable NeRFs

Sim-to-real transfer can achieve impressive results in many situations, as exemplified by the methods presented in the previous section, but often fails dramatically if the simulation used for learning is not faithful enough to the real world. When such a discrepancy exists, resolving it often requires extensive manual engineering and domain expertise to tweak the simulator to be more realistic. We propose a novel neural rendering method, Composable Object Volume NeRF (COV-NeRF), that is uniquely suited to remedy sim-to-real mismatch. COV-NeRF performs real-to-sim learning of structured neural representations of objects encountered in the real world, which can then be used to compose and render new simulated scenes with photorealistic images and the corresponding supervision for many 2D and 3D perceptual tasks. The rendered images and supervision automatically enjoy geometric and semantic consistency with each other and across viewpoints. We study a real-world bin picking application where state-of-the-art perception models and datasets face a large sim-to-real gap, and show that COV-NeRF allows us to rapidly close the gap to achieve application-level improvement. This section represents work published in: *Nikhil Mishra et al. "Closing the Visual Sim-to-Real Gap with Object-Composable NeRFs". International Conference on Robotics and Automation (ICRA), 2024.*

Chapter 2

Distributional Instance Segmentation with Latent-Mask-RCNN

2.1 Introduction

Instance segmentation is the cornerstone of the perception stack in many real-world robotic systems. The goal of instance segmentation is to enumerate the objects (or *instances*) that appear in an image, specifying which pixels in the image belong to each object.

Recent work has mostly focused on developing specialized neural network architectures and loss functions that make the instance segmentation task more amenable to deep learning. For example, *detect-then-segment* methods [76, 30, 74, 7, 107] rely on a cascade of classification, regression, and filtering to first identify a *bounding box* for each instance (a related problem known as *object detection*), followed by an additional step to predict each instance's mask given its bounding box. Alternatively, *pixel-embedding* methods [1, 2], which optimize pixel-level auxiliary tasks, and then use a specialized clustering procedure to extract instance predictions from the dense pixel representation. Recent methods that attempt to apply Transformers to instance segmentation [47, 12] still rely on carefully-crafted, bespoke loss functions.

We observe that existing methods are not well equipped to deal with the inherent ambiguity that exists in the real world. We posit that this stems from a phenomenon we describe as limited *distributional expressiveness* – in particular, that most instance segmentation models are designed to predict only *one* possible segmentation hypothesis (a single set of object masks). When they encounter an ambiguous situation where multiple hypotheses could be plausible, they are forced to commit to a particular one. However, a distributional instance segmentation model should be capable of expressing uncertainty on over complex hypotheses, such as identifying that an ambiguous group of pixels could be one large object or two small objects. The use of non-distributional instance segmentation models can be incredibly limiting for high-performance autonomous systems, which may have strict error tolerances and asymmetric costs for different types of errors.

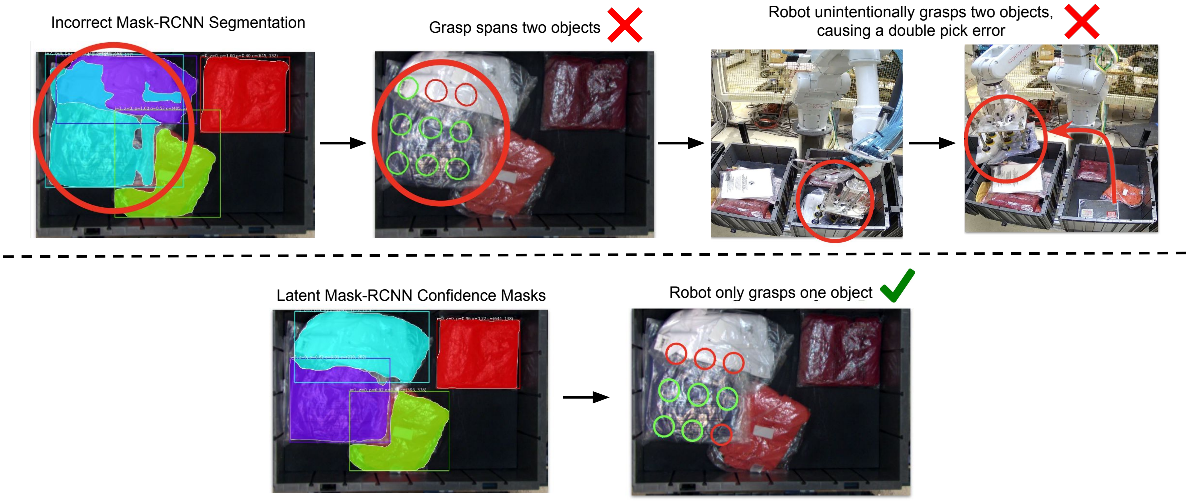

Incorrect Mask-RCNN Segmentation
Grasp spans two objects
Robot unintentionally grasps two objects, causing a double pick error
Latent Mask-RCNN Confidence Masks
Robot only grasps one object

Figure 2.1: Traditional instance segmentation methods such as Mask-RCNN cannot model uncertainty over object masks. For robotic applications, this can result in critical errors such as unintentionally picking two objects (a *double pick*). Our proposed Latent-MaskRCNN can predict multiple hypotheses, and understand the uncertainty that they reflect to make high-confidence predictions, reducing the rate of double pick errors.

To overcome these limitations, we propose *distributional instance segmentation*, where we explicitly model a distribution over possible segmentation hypotheses. The key contributions of this work are:

1. We introduce a distributional instance segmentation model, Latent-MaskRCNN, which uses latent codes to expressively model a distribution over instance segmentations. Latent-MaskRCNN can sample multiple segmentation hypotheses instead of making a single point estimate. We build on top of Mask-RCNN [30], which was a popular state-of-the-art method at the time of this work, but our latent variable formulation could be applied to any method.

2. We propose new methods for using the output of a distributional instance segmentation model. For robotic applications, we propose Confidence Masks as a method to obtain high-precision predictions from Latent-MaskRCNN. We also propose Union-NMS as a high-recall counterpart of Confidence Masks.

3. We release a dataset of over 5000 annotated images from a real-world robotics application that highlights the ambiguity in instance segmentation. We show Latent-MaskRCNN achieves high performance on this dataset, as well as on popular instance segmentation benchmarks from other domains.

4. On a real-world apparel-picking robot, we show that the Latent-MaskRCNN can be applied to significantly reduce critical errors, without sacrificing performance.

2.2 Related Work

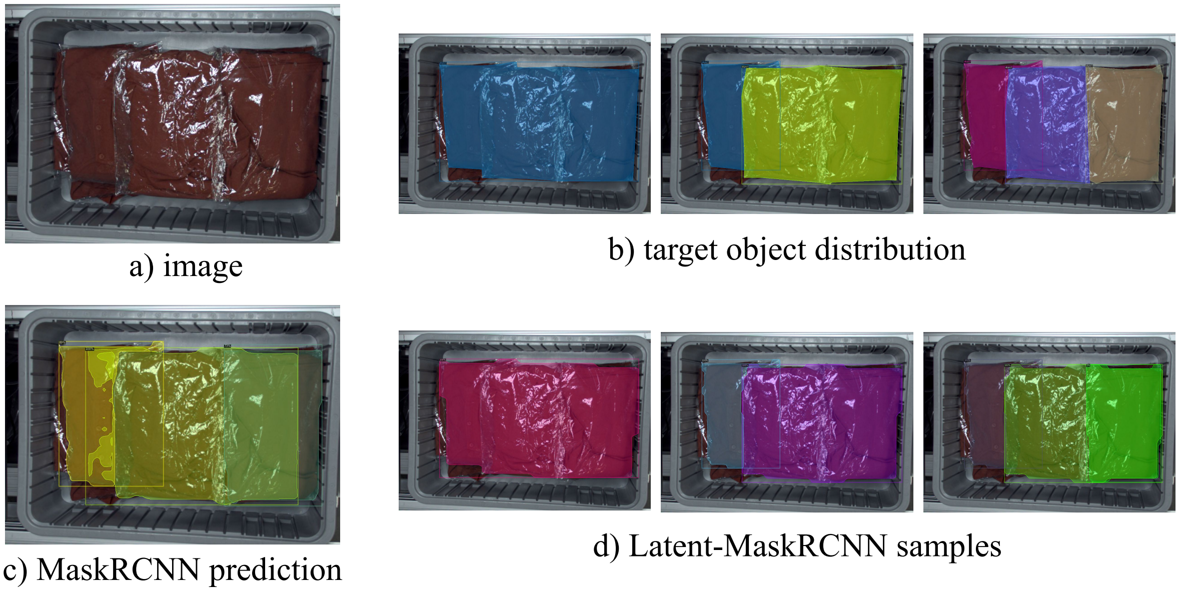
Detect-then-segment methods are the most popular instance segmentation methods, and Mask-RCNN belongs to this category. While they all first perform object detection and then segment each instance given its bounding box, there are some variations. For example, YOLACT [3] follows the same structure as MaskRCNN, but uses YOLO [74] as the object detector instead of FasterRCNN [76]. YOLO is very similar to FasterRCNN, making architectural changes that sacrifice some accuracy in exchange for real-time inference speed. Thus, we expect YOLACT to have the same distributional limitations as MaskRCNN. Other methods explore how to express uncertainty during the detection step, but they consider distributions of individual boxes rather than over sets of object masks [27, 28, 62]. Thus, they cannot express complex hypotheses, such as ones where the number of objects differs.

Mask-proposal methods [10, 97, 12] aim to circumvent bounding boxes as an intermediate representation. They are structured like FasterRCNN, but propose masks directly. Empirically, they do not behave much differently than MaskRCNN. Distributionally, they suffer from many of the same limitations as MaskRCNN: each proposal still models each pixel independently of the others, and they still rely on NMS to filter proposals.

Pixel-embedding methods [1, 2, 65, 79] work in a substantially different way than either of the above two families. They generally optimize some auxiliary task that encourages pixels in the same instance to have similar representations. Then they rely on a clustering-based inference procedure to extract instance predictions from their pixelwise representations. However, their performance has lagged quite far behind that of detect-then-segment methods, which has made them relatively unpopular. They can model per-pixel uncertainty in a manner similar to a naive semantic segmentation method, but this is likely insufficient for distributional expressiveness.

Following this work, Transformer-based methods [7, 107, 47] have matched or exceeded the performance of detect-then-segment methods. Although the underlying architecture is different, the set-matching losses that are typically used to train these models are not particularly compatible with expressive distributional output, and these methods suffer from the same distributional limitations as their non-Transformer predecessors. The latent variable method we propose in Section 2.4 could be similarly applied to any of these methods.

A number of methods explore how to express uncertainty in other structured prediction tasks. However, many of these do so by training multiple replicas of the entire model or some subset of the parameters, and modifying the training objective in a way that encourages diversity amongst the replicas [46, 25, 80, 19]. This incurs a multiplicative increase in the computational cost and memory footprint required at training time, which can be prohibitively expensive for large models. Other latent-variable formulations offer improvements on medical *semantic* segmentation and video segmentation tasks [42, 37, 49] . We find, however, that instance segmentation poses a richer set of challenges (variable size sets, high-dimensional masks) and has different application-specific uses.



a) image

b) target object distribution

c) MaskRCNN prediction

d) Latent-MaskRCNN samples

Figure 2.2: We construct a toy dataset using a single image (a) and a target distribution (b) that is uniform over threesets of objects, varying in number, size, and location. (c) When MaskRCNN is trained to fit this distribution, it blend objects from each of the three modes, and uncertainty in the mask head is expressed as spurious blobs on the left. (d) Samples from our distributionally-expressive Latent-MaskRCNN (Section) can cleanly capture each mode of the target distribution.

2.3 Distributional Instance Segmentation

Distributional instance segmentation seeks to fully model the full distribution over the set of objects present in an image. We pose it as a maximum likelihood problem, where a instance segmentation model learns the conditional distribution $p(y|x)$, letting x denote a given input image, and $y = \{y_k\}_{k=1}^{K}$ a set of instances. Each instance y_k is defined by a bounding box b_k, instance mask m_k and class c_k.

Using θ to denote the model parameters and $\mathcal{D}$ the dataset, we seek to optimize the standard maximum likelihood objective: $\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log p_\theta(y|x)]$. In the subsequent sections, we study two concrete examples where Mask-RCNN is unable to effectively optimize this objective.

Distributions over Sets

One of the main challenges in instance segmentation is modeling distributions over sets of variable size, since an image may contain an arbitrary number of objects. Traditional methods typically output only one set of objects through a deterministic inference procedure. However, a distributionally expressive model should be able to express uncertainty over hypotheses, where each hypothesis is a set of objects. Consider the example illustrated in

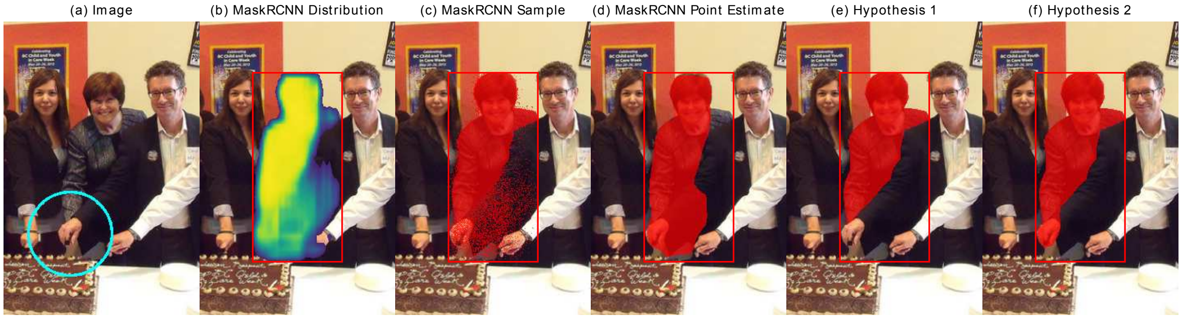


Figure 2.3: In the image shown in (a), who does the hand circled in blue belong to? (b) The pixelwise distribution predicted by MaskRCNN's mask head, which does not model the dependence between pixels. (c) A sample from this distribution, which independently samples the uncertain pixels in the hand, is not a plausible instance mask. (d) The (incorrect) point estimate that MaskRCNN would return. (e), (f) A more distributionionally expressive model might allow us to sample plausible hypotheses like these two. In this case, (e) is actually the correct segmentation.

Figure 2.2, where we might not be sure how many objects the image contains. We show three segmentations $y^{(1)}, y^{(2)}, y^{(3)}$ that might be plausible based on the image. If we construct a toy dataset where this image appears three times, once with each label, then the true conditional distribution is $p(y = y^{(j)}|x) = \frac{1}{3}$, for $j = 1, 2, 3$. This could model randomness in the annotation process, which could occur when fallible human annotators encounter challenging long-tail cases. Fitting this data requires the model to express a distribution over sets of objects. However, MaskRCNN can only predict a single set of objects, so the best it can do is to either choose a mode or blend the modes together. In Figure 2.2 (c), we see that it picks the mode $y^{(3)}$ at the bounding box level, and blends the modes together at the mask level. Meanwhile, a distributional instance segmentation model should be able to fit the true conditional distribution perfectly, and express the different possibilities as three distinct hypotheses. In Figure 2.2 (d), we show three samples from Latent-MaskRCNN, which we introduce in Section 2.2, and observe that it indeed does captures all three modes.

Expressive Pixelwise Mask Distributions

Another challenge in instance segmentation is coherently expressing the relationships between pixels. For example, Mask-RCNN's mask head models each pixel in an object's mask as independent of the others given its bounding box.

To make this concrete, consider the example show in Figure 2.3. In (a), there is a hand circled in blue: who does it belong to? At a first glance, it might seem like it belongs to the woman in the center of the frame, but a closer look reveals that it more likely belongs to the man on the right. In (b), we see that MaskRCNN predicts the correct bounding box for the woman, but is understandably uncertain about the hand (and even the rest of the man's arm). MaskRCNN expresses this uncertainty by outputting probabilities around 0.7 for pixels in the hand, compared to 1.0 for the rest of the woman. However, this is actually not an accurate reflection of the ambiguity that exists here: in (c), we plot one sample from the distribution in (b), and see that this is definitely not a plausible instance mask. In practice, MaskRCNN would return the mode of this distribution, the ultimately-incorrect point estimate shown in (d). Meanwhile, a distributional instance segmentation model would be able to capture the correlations between pixels, and might predict a distribution over the two plausible hypotheses shown in (e) or (f): either none of the hand is part of the mask, or all of it. The sample shown in (c) would be extremely unlikely under such a distribution.

2.4 Distributional Instance Segmentation with Latent-MaskRCNN

Latent Variable Formulation

How can we make instance segmentation models distributionally expressive, while retaining the inductive biases of existing model architectures? Drawing on prior work in variational inference, we consider a latent-variable formulation where we incorporate latent codes in the style of a variational autoencoder (VAE) [42, 40]. If we adopt this framework, then an instance segmentation model becomes a conditional VAE that is trained to maximize the evidence lower-bound (ELBO):

$$\log p(y|x) \geq \mathbb{E}_{z \sim q}[\log p(y|x,z)] - D_{KL}(q||p(z|x)) \tag{2.1}$$

Typically $q(z|y,x)$ is known as the encoder, $p(y|x,z)$ as the decoder, and $p(z|x)$ as the prior, and these components are all learned to maximize the ELBO. The decoder is essentially an instance segmentation model in the traditional sense, except that it is augmented to additionally consume a latent code z. This general technique allows us to reuse any existing instance segmentation model to implement our decoder (and train it in the same way), with only a slight modification to incorporate z as an input. During inference, we can sample from $p(y|x)$ by sampling different latent codes $z^{(k)} \sim p(z|x)$. Each can be decoded them into different segmentations $y^{(k)} \sim p(y|x,z)$.

Latent-MaskRCNN

In principle, the latent variable method introduced in the previous section can be applied to any existing instance segmentation model. In this section, we explore how it might be applied to Mask-RCNN. We call the resulting model *Latent-MaskRCNN*. We chose Mask-RCNN since it is one of the most popular instance segmentation models and has served as the basis for most state-of-the-art methods in recent years.

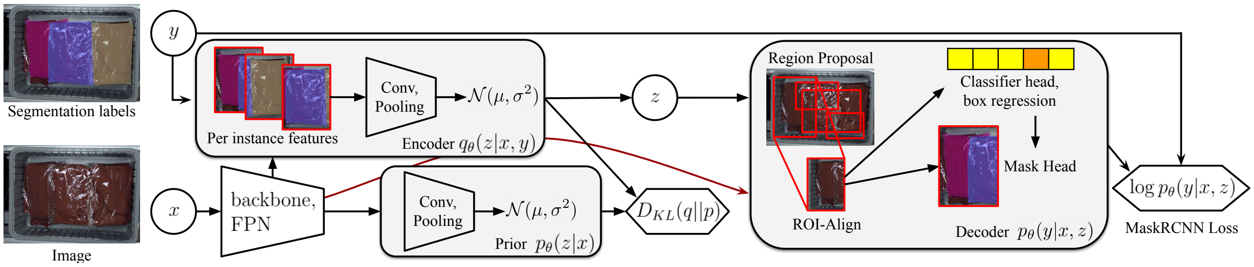


Figure 2.4: Overview of Latent-MaskRCNN: At training time, the encoder q_θ uses features extracted from the image x and labels y to sample a latent code z which is passed into the decoder. The decoder fuses with the latent code with image features, but otherwise follows the standard MaskRCNN architecture, including region proposal, non-maximum suppression (NMS), and box/mask/classifier heads. At inference time, z is sampled through the prior $p_\theta(z|x)$ which only takes the image x as input.

Figure 2.4 provides an overview of the Latent-MaskRCNN architecture, and Figure 2.5 shows the encoder and decoder in more detail. Our models and code are also released on our project page.

The decoder uses the same architecture as MaskRCNN, with the only change being the incoporation of the latent codes. To allow the latent codes to influence as much of the prediction as possible, we want to inject them as early in the model as possible, so that they can influence the every stage of the model, including the region proposal network, box refinement head, classifier head and mask head. We use latent codes of a fixed vector dimensionality $z \in \mathbb{R}^d$. We found that $d = 64$ was a reasonable choice that worked for all datasets. We tile the latent codes across the spatial dimensions of the image and concatenate them in the channel dimension with the feature maps from the Feature Pyramid Network (FPN) [50]. Then we use a few convolutional layers to project the augmented feature maps back down to their original channel dimension.

The encoder of Latent-MaskRCNN takes in an image x along with a set of ground-truth instances y, and produces a distribution over latent codes $q_\theta(z|y, x) = \mathcal{N}(\mu_\theta(y, x), \sigma^2_\theta(y, x))$. The architecture takes inspiration from the mask head of Mask-RCNN: it acts like an "inverse" mask head that operates on each ground-truth instance, and then pools features from across all instances. For each ground truth instance y_i, we extract ROI-aligned features from the FPN feature maps, and then use a small CNN to embed each one into a single feature vector. We use these per-instance features as the initial node features in graph neural network (GNN) that aggregates them into a single, globally-informative latent code for the entire image. After several graph network layers (the graph is fully-connected), we mean-pool across the final node features and use a fully-connected layer to produce a mean and log-variance for our latent distribution. The encoder is typically not used during inference, as we do not have access to the ground-truth instance masks.

The prior takes in an image x and produces a distribution over latent codes $p_\theta(z|x) =$

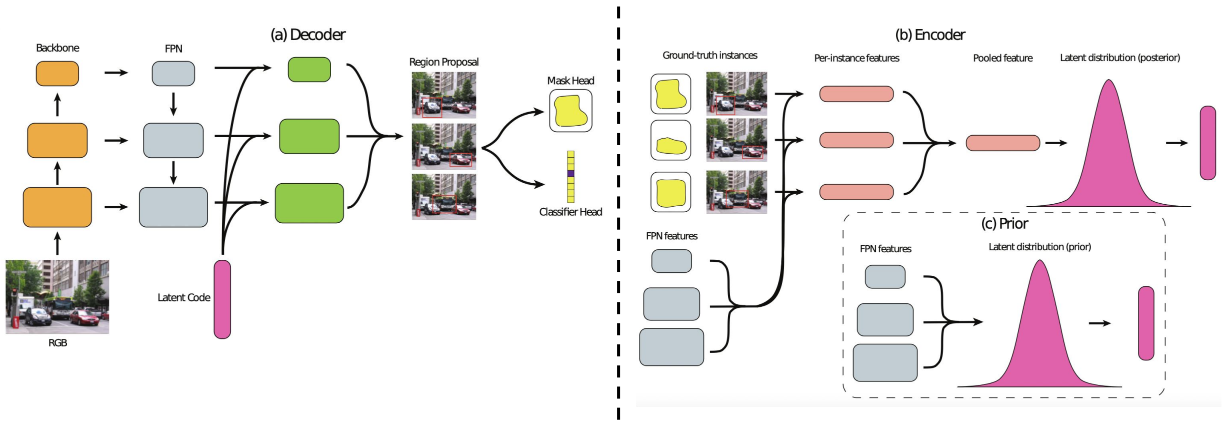


Figure 2.5: The architecture of Latent-MaskRCNN. (a) The decoder is exactly the same as MaskRCNN, except that, before region proposal, we augment the FPN feature maps with a latent code. (b) The encoder takes the (non-augmented) FPN feature maps and a list of ground-truth instances and predicts a diagonal Gaussian distribution over latent codes. (c) The prior takes the (non-augmented) FPN feature maps and predicts a diagonal Gaussian distribution over latent codes.

$\mathcal{N}(\mu_\theta(x), \sigma_\theta^2(x))$. We apply a few convolutional layers to the FPN feature maps, mean-pool across the spatial dimensions, and then predict a mean and log-variance using a small MLP. During inference, we sample latent codes from the prior (instead of from the encoder), but the decoder consumes them in the same way as during training.

During training, we use the encoder q_θ to sample latent codes z, which are passed to the Mask-RCNN decoder. We maximize the ELBO (Equation 2.1), treating Mask-RCNN's bespoke losses as the distortion (loosely corresponding to an energy-based model): $-\log p(y|x, z) \sim \mathcal{L}_{\text{MRCNN}}(x, y, z)$:

$$\mathcal{L}_{\text{MRCNN}}(x, y, z) = \mathcal{L}_{RPN} + \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{2.2}$$

We found it helpful to use a KL warm-up, as is a common practice for training VAEs [32]. The final training objective for Latent-MaskRCNN is thus:

$$\mathcal{L}(x, y) = \mathbb{E}_{z \sim q}[\mathcal{L}_{\text{MRCNN}}(x, y, z)] + \beta D_{KL}(q||p(z|x)) \tag{2.3}$$

In the first part of training, we use $\beta = 0$ and increase β towards the end of training. This allows the latent code to encode useful information early on as the rest of the model is still learning; towards the end of the training, higher β pushes the latent space to be covered by the prior for better samples.

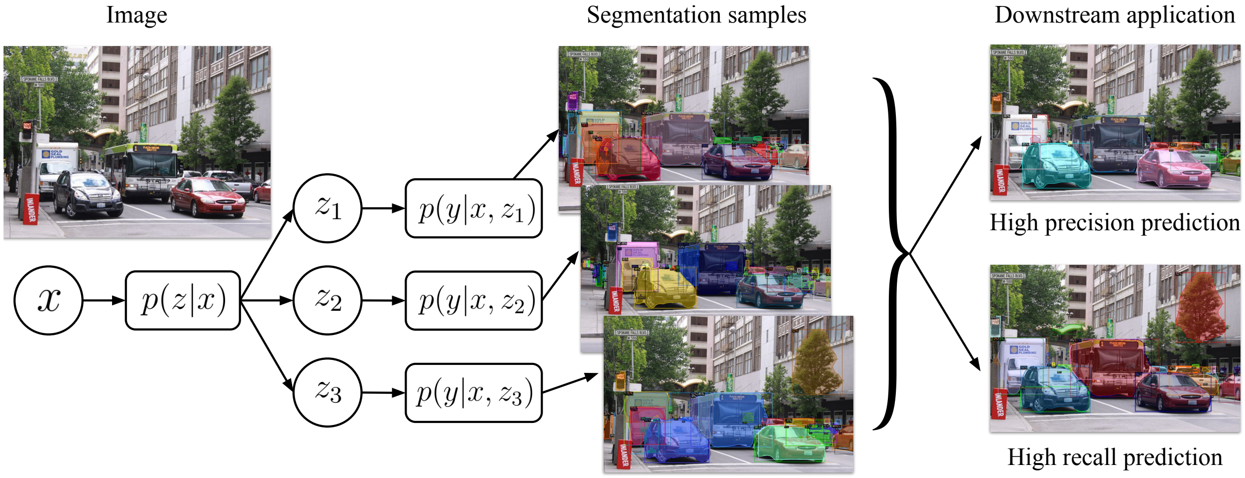


Figure 2.6: At inference time, Latent-MaskRCNN's encoder q_θ is discarded, and latent variables z_i are sampled from the image x conditioned prior $p_\theta(z|x)$. Each latent is decoded using $p_\theta(y|x, z_i)$ into a set of masks, which can be used for our high precision or recall predictions depending on the application.

2.5 Applying Distributional Instance Segmentation

A distributional instance segmentation model like Latent-MaskRCNN can express its uncertainty over multiple hypotheses. However, how should a downstream application best consume this expressive distributional output? Instance segmentation often occurs at the beginning of the perception pipeline, and there could be many ways to incorporate the samples. Moreover, each application may have asymmetric error tolerances: failing to detect an object can be catastrophic in autonomous driving but acceptable in robotic picking, while grouping two objects as one is a critical error in robotic picking but more forgivable in driving applications. In this section, we show how a single Latent-MaskRCNN model can be used flexibly across a number of applications with different requirements (Figure 2.6).

High-Precision Segmentation with Confidence Masks

In some applications, it can be very costly to make *undersegmentation* errors, when a particular instance mask prediction actually spans multiple objects. For example, consider a robotic manipulation application, where a robot must pick objects one at a time and feed them into a sortation process. If the model undersegments an instance, it may inadvertently pick multiple objects (a type of error commonly referred to as a *double pick*), which can be a particularly expensive error for the downstream application.

Suppose we draw several samples from Latent-MaskRCNN. If two pixels belong to the same instance mask in many samples, then we can be reasonably confident that they actually do belong to the same ground-truth instance. Building on this intuition, we can compute a *p-confidence mask*, consisting of pixels that are all likely to be contained in a single ground-truth instance.

For a given confidence requirement p, we define a confidence mask c_p as a mask that is fully contained in a ground truth mask m with probability at least p: $P(c_p \subseteq m) \geq p$. Using Latent-MaskRCNN, we can approximate this probability as:

$$P(c_p \subseteq m) = \mathbb{E}_{m \sim P(m|x)}[\mathbb{1}\{c_p \subseteq m\}] \approx \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}\{c_p \subseteq m_i\}$$

In the finite sample regime, $\hat{c}_p$ is an empirical confidence mask if it is contained within a sampled mask for p fraction of the samples.

Now consider any subset of masks I consisting of one mask m_j from at least kp different samples. If we take the intersection of all of the masks in I, $\hat{c}_\tau = \bigcap_{m_j \in I} m_j$, then this intersection mask must be contained in each of the masks used in the intersection $\hat{c}_p \subseteq m_j$. Therefore we have:

$$\frac{1}{k} \sum_{m_j \in I} \mathbb{1}\{\hat{c}_p \subseteq m_j\} = \frac{|I|}{k} \geq p$$

and thus $\hat{c}_p$ is an empirical confidence mask by construction.

Figure 2.7 illustrates confidence mask predictions for different p. Notice that as the confidence requirement increases, the unconfident extents of the masks shrink, and some uncertain masks are eliminated.

Since each confidence mask is an intersection of masks $c_p = \bigcap_I m_j$, how should we assign the score of a confidence mask prediction? One intuitive approach might be to take the average of the score s_i of each mask in the intersection: $\frac{1}{|I|} \sum s_j$. However, a confidence mask is not an average of masks but rather an intersection.

To formulate a better score for our confidence mask prediction, consider two scenarios. In the first scenario, the model is very confident about an object's mask so it predicts roughly the same mask in every sample. The resulting confidence mask c_p has high IoU with each of the masks used in the sample m_j. On the other hand, consider an unconfident prediction where the object's mask varies significantly across samples. Here, the confidence mask c_p represents a small but confident region of the object whose extent is highly uncertain. The resulting IoU between c_p and each m_j will be smaller than when the model is confident and masks are not varying across samples.

Formalizing this intuition, we score each confidence mask as the mean score-weighted IoU between the predicted mask c_p and every mask used in the intersection:

$$s_{c_p} = \frac{1}{|I|} \sum_{m_j \in I} s_j \frac{|c_p \cap m_j|}{|c_p \cup m_j|}$$

When s_{c_p} is large, this indicates that c_p is a confident intersection of masks with very similar IoU. On the other hand, a small s_{c_p} indicates that c_p has low score or IoU with its samples and likely does not capture the full extent of the object well.

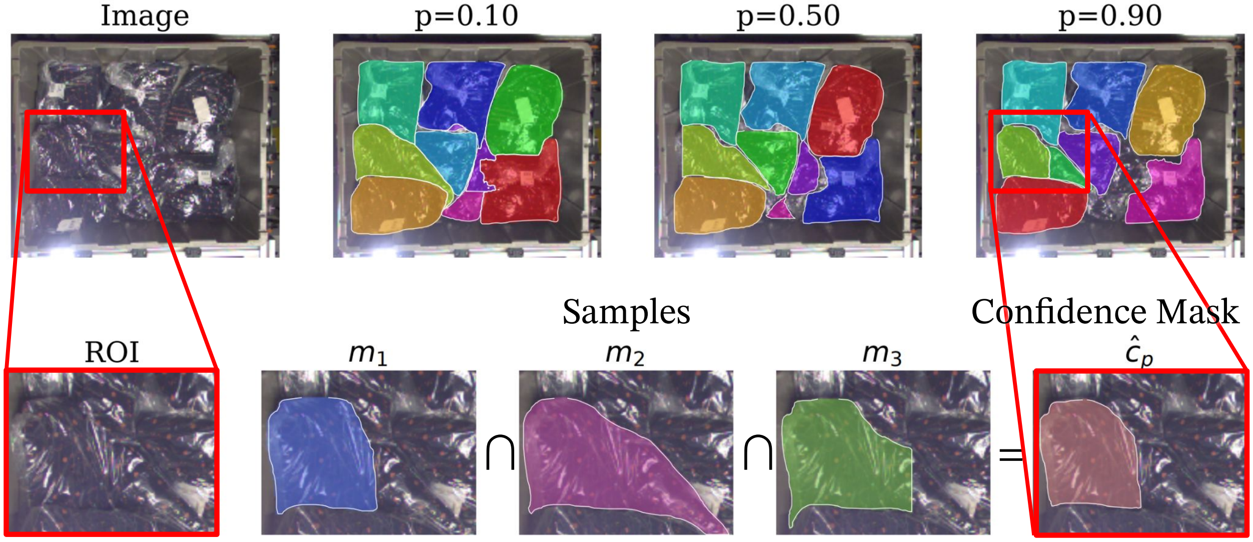


Figure 2.7: Top: Confidence Mask predictions with different p. Notice that as the confidence requirement p increases, single objects can be split into two, ambiguous object extents are reduced, and uncertain objects are eliminated entirely. Bottom: Constructing an empirical confidence mask $\hat{c}_p$ by taking intersection of samples m_1, m_2, m_3. When an object's extent is uncertain, a high confidence mask prediction will only consist of pixels that are highly likely to be contained within an object as determined by the samples.

Algorithm 1 presents a simple procedure to condense several sampled instance segmentations into a set of confidence masks. At each iteration, this scheme greedily adds the highest scoring confidence mask that can be obtained, considering the remaining masks in each sample and the confidence masks that have already been produced thus far.

High-Recall Segmentation with Union-NMS

Other applications might be concerned about *oversegmentation*, the complement of undersegmentation. For example, in autonomous driving, failing to identify a pedestrian, or predicting them to be smaller than they actually are, can lead to a catastrophic error.

To make high-recall predictions with Latent-MaskRCNN, we use a procedure called *Union-NMS*, detailed in Algorithm 2. We first sample multiple segmentations from the model, and run non-maximum suppression (NMS) on the predicted masks. NMS checks if any two masks m_i, m_j have IoU greater than some threshold, and then discards the lower-scoring one. Suppose that some mask m_i remains after we perform NMS. Then Union-NMS returns the union of m_i with every mask that it suppressed, achieving higher recall by incorporating masks that would have otherwise been ignored.

Algorithm 1: Iterative Greedy Confidence Masks

```
Given: confidence threshold $p$, sampled segmentations $y^{(1)}, \dots, y^{(K)}$, each with
  masks $m_j \in y$, cutoff threshold $\epsilon$
Initialize $M \leftarrow \{\}$ to be a set of predicted masks
while all masks in $M$ have score $s_{c_p} \geq \epsilon$ do
    for $m_h \in y^{(i)}$ do
        Compute the area of the intersection $I_{jg} = |m_h \cap m_j \setminus M|$ with all other masks
          $m_j \in y^{(g)}$
        Let $C$ be the set of masks with the $kp$ highest $I_{jg}$ where each mask must come
          from a unique $y_{(g)}$
        Compute the intersection $m_h^* = \bigcap_{m \in C} m \setminus M$ ignoring predicted masks $M$
    Add mask with highest score $\arg\max s_{m_h^*}$ to $M$
```

Algorithm 2: Union-NMS

```
Given: Masks $M = \{m_i\}_{i=1}^n$ sorted by confidence, IoU threshold $\tau$
Initialize $S \leftarrow \{\}$ to an empty map
for $i = 1 : n$ do
    if $i \in K$ then
        continue
    $S[i] \leftarrow \{\}$
    for $j = i + 1 : n$ do
        if $IoU(m_i, m_j) > \tau$ then
            Add $j$ to $S[i]$
Initialize $U \leftarrow \{\}$
for $i \in keys(S)$ do
    $m \leftarrow m_i$
    for $j \in S[i]$ do
        $m \leftarrow m \cup m_j$
    Add $m$ to $U$
Result: $U$
```

Vanilla Prediction with the Prior Mean

Some applications may not have any specific performance requirements or may have strict inference time requirements. In these cases, a point estimate may be sufficient. With Latent-MaskRCNN, we can achieve this by always decoding the mean of the prior: $z = \mu_\theta(x)$ where $\mu_\theta(x)$ is the mean of the prior $p_\theta(z|x)$ (for Gaussian $p_\theta(z|x)$, it is also the mode of the distribution). We found that this scheme typically matches or yields a small improvement over MaskRCNN predictions, suggesting that Latent-MaskRCNN strictly increases the expressiveness of MaskRCNN and no performance is lost by using a more expressive distribution.

2.6 Experiments

We conducted experiments seeking to answer the following questions:

1. How well does Latent-MaskRCNN model uncertainty over instance segmentations, across a variety of datasets?
2. How can Latent-MaskRCNN's distributional output be used effectively downstream applications with asymmetric error tolerances?
3. How effective is Latent-MaskRCNN at reducing critical double pick errors in robotic picking applications?

Datasets

To help us answer these questions, we compared MaskRCNN and Latent-MaskRCNN across several datasets, each with its own set of challenges. The complete results are in Table 2.1.

COCO [51]: This large dataset is the standard benchmark for instance segmentation. There are many object categories and a huge variety in image composition.

Cityscapes [14]: A real-world dataset from an autonomous driving application. Although it is smaller and more specialized than COCO, it is still a popular benchmark for instance segmentation. One notable challenge is that there are many background instances that are still important to segment (e.g. pedestrians), but the limited image resolution can introduce some uncertainty.

Apparel-5k: We collected this dataset of roughly 5000 images from a robot picking application. We use 4198 images in the training set and 463 in the validation set. There is only one object category, but the images exhibit a lot of inherent ambiguity due to complex occlusions, lighting, transparency, etc. The dataset is released on our project page for the broader community to build upon our work.

For each dataset, we trained both MaskRCNN and Latent-MaskRCNN on 8 NVIDIA 1080Ti GPUs using MaskRCNN's released hyperparameters and training schedules. We use the ublicly available train/val/test splits for all experiments and datasets. We used a ResNet-50 backbone [29], initialized from pretrained-Imagenet [17] weights (for COCO) or pretrained COCO weights (for other datasets).

Evaluating Confidence Masks

In Section 2.5, we proposed Confidence Masks as a method to obtain high precision predictions for applications where avoiding undersegmentation is critical. However, existing metrics for instance segmentation are not well-equipped to detect undersegmentation: IoU and mAP equally weigh oversegmentation vs undersegmentation and precision versus recall. In these

Table 2.1: Evaluation of MaskRCNN and different prediction modes of Latent-MaskRCNN across various datasets and metrics.

		COCO [51]			CityScapes [14]			Apparel-5k		
Method	Inference Mode	MR@HP	AR	mAP	MR@HP	AR	mAP	MR@HP	AR	mAP
MaskRCNN	n/a	20.0	66.0	35.0	25.3	55.1	**35.8**	23.6	39.9	26.9
Latent-MaskRCNN	Union NMS	7.4	**72.3**	26.5	17.7	**57.5**	33.6	13.6	**61.4**	34.1
Latent-MaskRCNN	Confidence Masks	**22.0**	48.1	30.5	**28.0**	48.6	34.1	**42.4**	41.8	**35.1**
Latent-MaskRCNN	Prior Mean	19.5	65.8	**35.3**	25.7	53.8	35.0	26.9	49.4	34.3

cases, we care that predictions have high Intersection-over-Prediction: $\text{IoP}(m_i, g) = \frac{|m_i \cap g|}{|m_i|}$. When IoP is high, errors due to undersegmentation are less likely to occur.

When evaluating models in this regime, we need to trade off precision (in terms of IoP) with recall, to avoid degenerate solutions. To do this, we consider the *max recall at high precision* (MR@HP):

$$\text{MR@HP} = \frac{1}{|p| \cdot |\tau|} \sum_{p_i \in p, \tau_j \in \tau} \max_{t:\text{Precision}(\tau_j) \geq p_i} \text{Recall}(t, \tau_j)$$

For a given precision threshold p_i and IoP threshold τ_j, we can compute the max recall that each model achieves (or zero, if it never achieves precision p_i). The MR@HP metric is the average of these recalls, over a range of precision threshold p and IoP thresholds τ. For high precision use-cases, we care about performance at high values of these thresholds, therefore we use $p = \tau = [0.75, 0.8, 0.85, 0.9, 0.95]$.

In Table 2.1, we evaluate Latent-MaskRCNN using both the prior-mean scheme from Section 2.5 as well as confidence masks with a confidence level $p = 0.9$. Across all three datasets, we find that confidence masks yield the best performance in terms of MR@HP. As for mAP, we find that the prior-mean prediction can match, if not exceed, the performance of MaskRCNN on all three datasets. On the challenging Apparel-5k dataset, we find that confidence masks and the prior-mean significantly outperform MaskRCNN in terms of MR@HP and mAP. Overall, we find that Latent-MaskRCNN is a strict improvement over MaskRCNN by matching overall detection performance in terms of mAP and offering the best high-precision performance in terms of MR@HP.

Evaluating Union NMS

In Section 2.5, we proposed Union-NMS as an inference procedure for applications where avoiding oversegmentation is critical. In such cases, we care that we have high recall (that we detect every instance that exists), and that each mask prediction has high *IoG* (intersection-over-ground-truth): $\text{IoG}(m_i, g) = \frac{|m_i \cap g|}{|g|}$. To capture both of these considerations, we consider the average recall (AR) [51] using IoG.

We evaluated Latent-MaskRCNN using both its prior-mean (Section 2.5) and Union-NMS. In Table 2.1, we show that the prior-mean predictions are similar to MaskRCNN on all three

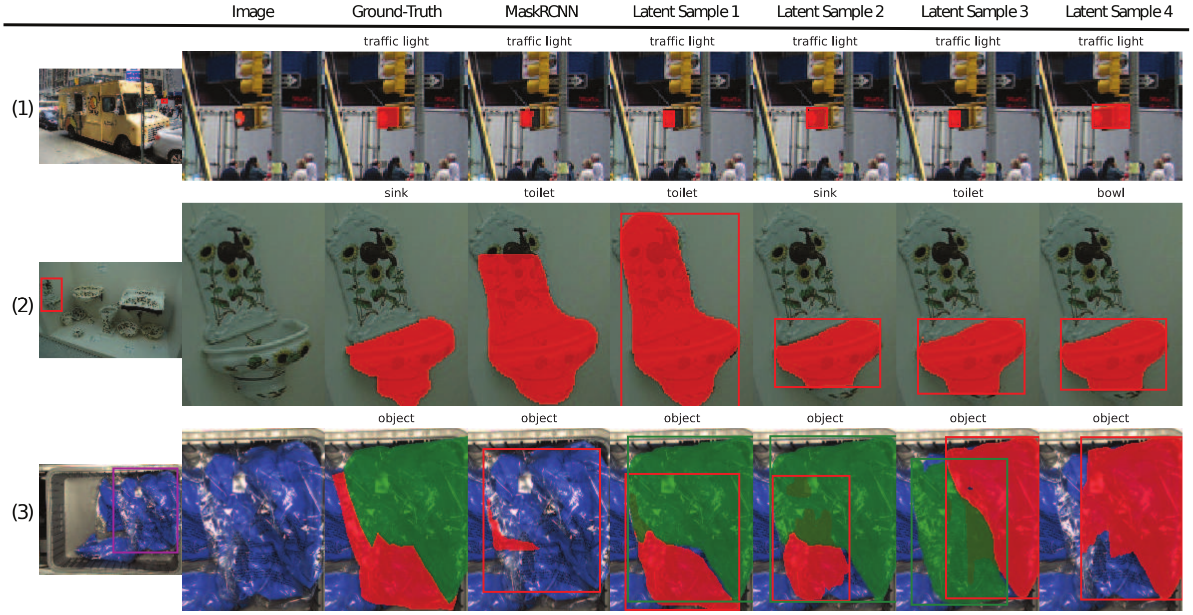


Figure 2.8: Each row corresponds to a single image. Columns 1-2 show the original image and a zoomed-in version. The remaining columns show instance masks, and the word above the image is the class of the instance. Column 3 is the ground truth, column 4 is MaskRCNN's prediction, and columns 5-8 are samples from Latent-MaskRCNN.

datasets, while Union-NMS achieves substantially higher AR (using IoG). This suggests that Latent-MaskRCNN with Union NMS can more effectively cover different modes of uncertainty, for high-recall applications.

Qualitative Evaluation of Samples

In this section, we qualitatively explore what kinds of uncertainty Latent-MaskRCNN can express. In Figure 2.8, we visualize samples from the model on images from various datasets, and observe that it does capture several distinct types of ambiguity:

Category confusion: Latent-MaskRCNN can express meaningful uncertainty in the classification head. In row 2, MaskRCNN confidently classifies this sink as a toilet, while different samples from Latent-MaskRCNN classify it as toilet, sink, and bowl.

Category imprecision: Even when the class of an instance is obvious, there still may be ambiguity in how the category is defined. For example, in row 1, both MaskRCNN and Latent-MaskRCNN are (correctly) confident that this instance is a traffic light. However, depending on how you define exactly what constitutes the extent of the traffic light, the instance mask may look very different. Latent-MaskRCNN samples a wide range of plausible possibilities, but MaskRCNN picks a single (in this case incorrect) mode.

Object and mask ambiguity: Latent-MaskRCNN lets us sample from a distribution over sets of objects. For example, in row 3, we see that the samples contain different numbers

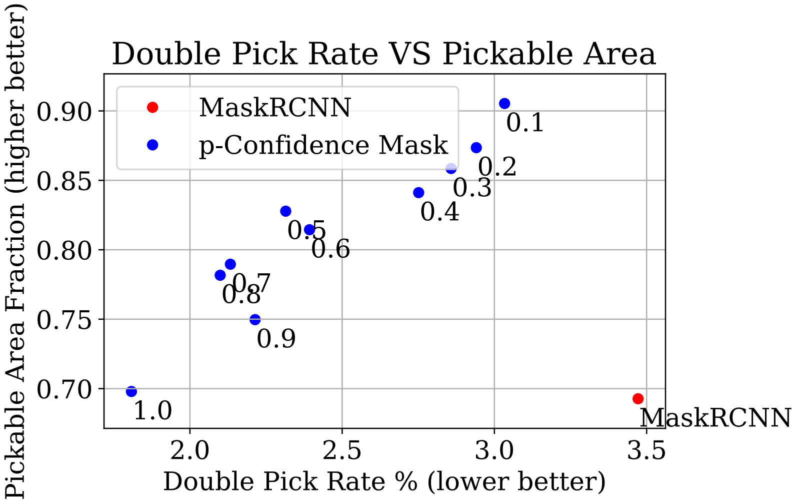


Figure 2.9: Latent confidence masks achieve lower double pick rates and generally more pickable area compared to MaskRCNN.

of objects, variety in bounding boxes, and variety in instance masks. Even though none of the samples are perfect, they are all plausible and all markedly better than MaskRCNN.

In row 2, we see examples where Latent-MaskRCNN's hypotheses express uncertainty in both mask and bounding box, while MaskRCNN picks a single mode.

Double Pick Evaluation on Apparel-5k

As discussed in Figure 2.1 and Section 2.5, double pick errors are especially costly in robotic picking applications. For example, if the robot is fulfilling e-commerce orders to be sent to a customer, a double pick can result in the customer receiving the wrong order and the warehouse having an incorrect inventory count. If the robot is sorting packages in a shipping facility, a double pick can result in packages being sent to the wrong destination.

On the Apparel-5k dataset, we can estimate the double pick rate of an instance segmentation segmentation prediction by approximating the robot's gripper as a circle with a fixed radius in pixel space (corresponding to a single suction cup). Then, we randomly sample pixel locations in the image and count the number of circles D that land within one predicted mask but more than one ground truth mask. We divide this by the number of circles N that land within one predicted mask, to arrive at the estimated double pick rate $R = \frac{D}{N}$. Empirically, we find that this simulated double pick rate is correlated with double pick rates on a real robot. In Section 2.6, we evaluate the true double pick rate with a real robot, but this evaluation can be performed more scalably and reproducibly – both offline and without access to a particular physical environment.

To ensure that we are avoiding degenerate solutions, we also consider the pickable area: the amount of visible surfaces that the robot can pick from. A model that predicts more

Table 2.2: Apparel-picking robot evaluation. * indicates a statistically significant difference

Method	Double Pick Rate	Average Sealed Cups
MaskRCNN	4.40%*	4.76
Latent-MRCNN	**0.82%***	**4.91**

accurate pickable areas enables the robot to have more flexibility in its grasping strategy (e.g. due to kinematic or other constraints). To this end, we compute the ratio between the area of all the predicted masks and the area of all the ground truth masks, and evaluate how the pickable area fraction trades off with the double pick rate.

We compare MaskRCNN and Latent-MaskRCNN with varying p-confidence masks in Figure 2.9. We find that Latent-MaskRCNN outperforms MaskRCNN in fraction of pickable area and double pick rate in all cases. Moreover, tuning the confidence parameter p s allows for application-specific tradeoffs between double pick rate and pickable area. Higher values of p tend to correspond to lower double pick rate and less pickable area, as the confidence requirement for each prediction is increased. MaskRCNN, on the other hand, can only realize a single double pick rate and pickable area fraction.

Double pick reduction on a real-world robot

To evaluate whether Latent-MaskRCNN can improve real-world application performance, we compare MaskRCNN and Latent-MaskRCNN on an apparel-picking robot. We use an ABB-1300 with a 9-cup suction gripper to pick apparel items in polybags between two totes (Figure 2.1). The robot uses two overhead camera systems to perform instance segmentation and then grasp point generation. The grasp points are optimized to land as many suction cups as possible on a single object detected by the segmentation model.

We only vary which segmentation model is used while holding other parts of the system constant, including hardware, object set, and grasp point generation. Each segmentation model is trained on the same Apparel-5k dataset. We run each model with several hundred grasps and record the number of double picks, grasps that unintentionally pick two objects. A typical high-automation warehouse can tolerate at most a 1% double-pick rate before the robot becomes prohibitively ineffective.

We also measure the average number of sealed cups on a grasped item, as an approximate measure of grasp quality. Since the suction holding force that can be exerted is proportional to the number of sealed cups, grasps that use fewer sealed cups tend to result in more dropped objects, which leads to jams, lost inventory, and costly human intervention. Naively shrinking the predicted instance masks or chapping bigger masks into smaller ones might trivially reduce the double-pick rate at the expense of grasp quality, by forcing the system to use fewer suction cups.

Table 2.2 reports the results of our apparel-picking experiments. We find that Latent-MaskRCNN with 0.9-Confidence mask significantly reduces the double pick rate. This validates our findings from the simulated experiments on Apparel-5K in Section 2.6. Moreover, Latent-MaskRCNN achieves slightly better average number of sealed cups, suggesting that grasp quality was not sacrificed. This suggests that Latent-MaskRCNN's sense of uncertainty is well-calibrated, since it is only conservative when ambiguity is present.

2.7 Discussion

We highlighted the importance of distributional expressiveness in instance segmentation, and showed where existing state-of-the-art approaches fall short. We then proposed a technique that uses latent variables to overcome this shortcoming, and showed how it can be applied to a model family like MaskRCNN. Our proposed Latent-MaskRCNN can express a wide range of uncertainty by sampling different segmentation hypotheses, and we proposed Confidence Masks and Union-NMS as two different ways that this uncertainty can be leveraged depending on the different error tolerances of the downstream application. Latent-MaskRCNN demonstrated strong performance across robotics, autonomous driving, and general object datasets. On a real apparel-picking robot, we found that it can significantly reduce the rate of critical double-pick errors while maintaining high performance. We hope that future work in instance segmentation can continue to explore the theme of distributional expressiveness and build on top of our methods and datasets developed in this chapter.

Chapter 3

Autoregresive Bounding Box Prediction

3.1 Introduction

Predicting 3D bounding boxes is a core part of the perception stack in many real-world applications, including autonomous driving, robotics, and augmented reality. The inputs to a 3D bounding box predictor typically consist of an RGB image and a point cloud; the latter is usually obtained from a 3D sensor such as LIDAR or stereo depth cameras. These 3D sensing modalities have their own idiosyncrasies: LIDAR tends to be accurate but very sparse, and stereo depth can be both sparse and noisy. When combined with the fact that objects are only seen from one perspective, the bounding-box prediction problem is fundamentally underspecified: the available information is not sufficient to unambiguously perform the task.

Imagine that a robot is going to grasp an object and manipulate it — understanding the uncertainty over its size can have a profound impact on what the robot decides to do next. For example, if it uses the predicted bounding box to avoid collisions during motion planning, then we may want to be conservative and err on the larger side. However, if the robot is trying to pack items into a shipment, then having precise, accurate dimensions may also be important.

Consider the scene depicted in Figure 3.1, which we observed in a real-world robotics application. From the image of the object in (a), it is fairly easy to gauge the width and length of the indicated object, but how tall is it? The object could be as deep as the bin, or it could be a stack of two identical objects, or even a thin object – but from the available information, it is impossible to say for sure. Formulating bounding box prediction as a regression problem results in a model that can only make a “pointwise” prediction – even in the face of ambiguity, we will only get a single predicted bounding box, shown in (b). A sufficiently expressive bounding-box model should understand the space of plausible bounding box hypotheses and make different predictions for different confidence requirements.

Setting aside partial observability, the prediction space has complexities that require

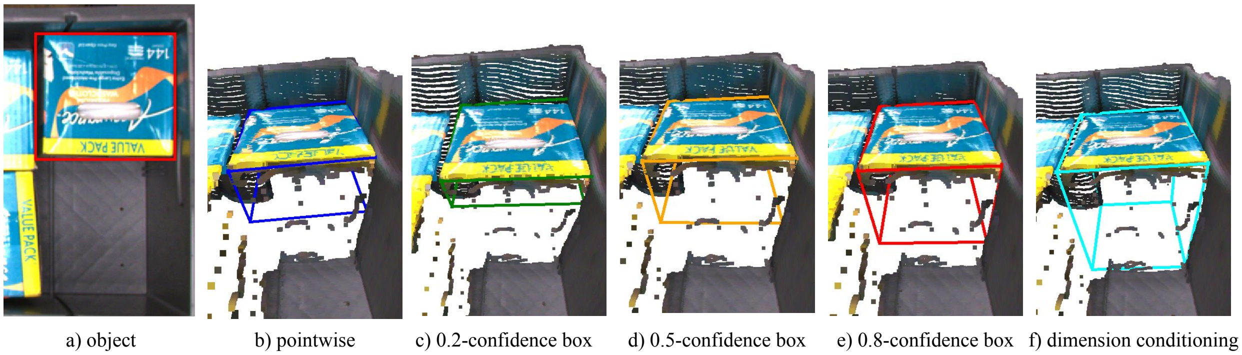


Figure 3.1: (a) In this scene from a real-world robotics application, how tall is the object highlighted in red? (b) A point estimator could output only one box prediction with no notion of uncertainty (c)-(e) Predictions from our proposed autoregressive model (Section 3.3) with difference confidence parameters (Section 3.4). Notice that the predicted box expands in the direction of uncertainty as we increase the confidence requirement. (f) Our autoregressive model can easily leverage additional information such as known dimensions to make more accurate predictions.

care in the design of a bounding-box estimator. Making accurate predictions requires the estimator to reason about rotations, which has been observed to be notoriously difficult for neural networks to predict and model uncertainty over [108, 22, 68]. Many existing methods sidestep this problem by constraining their predictions to only consider rotations about a single axis or no rotations at all. This can be sufficient for some applications but is too limiting for the general case.

A common thread that links these challenges together is the necessity to reason about uncertainty. This has been largely underexplored in existing work, but we hypothesize that it is critical to improving 3D bounding box estimators and expanding their usability in applications of interest. Taking a similar approach to the Latent-MaskRCNN method we proposed in Chapter 2 for instance segmentation, we propose to tackle this problem by predicting a more expressive probability distribution that explicitly accounts for the relationships between different box parameters. Using a technique that has proven effective in other domains, we propose to model 3D bounding boxes autoregressively: that is, to predict each box component sequentially, conditioned on the previous ones. This allows us to model multimodal uncertainty due to incomplete information, make high confidence predictions in the face of uncertainty, and seamlessly relax the orientation constraints that are popular in existing methods.

Our key contributions are:

1. We propose an autoregressive formulation to 3D bounding box prediction that can model complex, multimodal uncertainty. We show how this formulation can gracefully scale to predict complete 3D orientations, rather than the 0- or 1-D alternatives that are common in prior work.

2. We propose a method to make high confidence predictions in ambiguous scenarios and estimate useful measures of uncertainty.

3. We introduce Common Objects in Bins (COB-3D), a simulated dataset of robotics scenes that illustrates why capturing uncertainty is important for 3D bounding box prediction, as well as the benefits and challenges of predicting full 3D rotations.

4. We show that our formulation applies to both traditional 3D bounding box estimation and 3D object detection, achieving competitive results on popular indoor and autonomous driving datasets, in addition to our proposed dataset.

Our code, models, and dataset are available on our project page. COB-3D-v2, an expanded version of COB-3D, was released in the work presented in Chapter 4; please refer to COB-3D-v2 for continued interest in the dataset.

3.2 Related Work

3D Bounding-box Estimation: Early work on 3D bounding box prediction [64, 69] assumes that object detection or segmentation has already been performed, and the bounding box predictor solely needs to identify a single 3D bounding box within a filtered point cloud. In this paper, we refer to this task as *3D bounding-box estimation.* Much of this work focused on developing specialized architectures to easily consume point cloud data, which are often sparse, unstructured, and noisy when obtained from real-world data.

3D Object Detection: Recently, a number of methods have explored how to jointly perform object detection and 3D bounding box estimation, rather than treating them as two explicit steps [82, 78, 53, 63, 98, 83, 70] . This task is known as *3D object detection* and is quickly gaining popularity over the decoupled detection and estimation tasks. The main focus has been on how to take the network architectures that have proven successful at the estimation task (which have strong inductive biases for operating on point clouds), and combine them with the architectures commonly used for the 2D object detection method (which are usually based on region proposals). As we show in Section TODO, our proposed autoregressive model can be applied to both the estimation and detection tasks.

Uncertainty Modeling in Object Detection: Uncertainty modeling has been studied in the context of 2D and 3D object detection [60, 48, 59, 27]. In many cases, these methods use independent distributions, such as Gaussian or Laplace, to model uncertainty over box parameters such as corners, dimensions, and centers [31, 60, 13]. While these

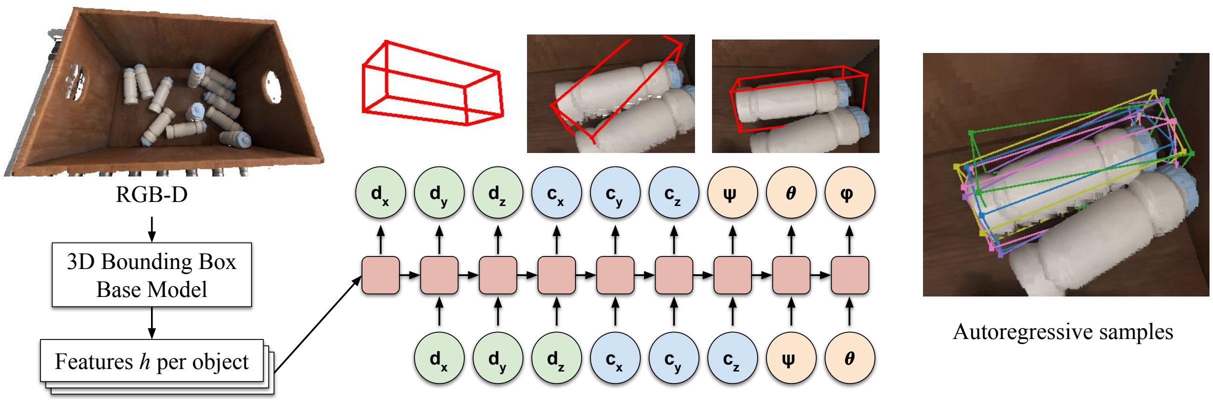


Figure 3.2: We compute per-object features h using a base model from RGB-D input. Then, we autoregressively sample dimensions, center, and rotations, each step conditioned on the previous one. We can express uncertainty through samples, such as the rotational symmetry of the bottle, whereas pointwise models could only make a single prediction.

distributions may capture some uncertainty for simple box parameterizations, they cannot capture correlations across parameters and have yet to be proven on full 3D rotations.

Autoregressive Models: Deep autoregressive models are frequently employed across a variety of domains. In deep learning, they first gained popularity for generative modeling of images [66, 92], since they can model long-range dependencies to ensure that pixels later in the autoregressive ordering are sampled consistently with the ones sampled earlier. In addition to being applied to other high-dimensional data such as audio [66], they have also been shown to offer precise predictions even for much lower-dimensional data, such as robot joint angles or motor torques [58].

3.3 Autoregressive 3D Bounding Box Prediction

3D bounding box estimation is typically formulated as a regression problem over the dimensions $d = (d_x, d_y, d_z)$, center $c = (c_x, c_y, c_z)$, and rotation $R = (\psi, \theta, \phi)$ of a bounding box, given some perceptual features h computed from the scene, e.g. from an image and point cloud. Prior work has explored various parametrizations and loss functions, but a notable salient feature to observe is that they all predict a *pointwise* estimate of the bounding box: the model simply outputs all of the box parameters at once. In 3D object detection, such regression is typically applied to every box within a set of candidates (or *anchors*), and fits into a larger cascade that includes classifying which anchors are relevant and filtering out unnecessary or duplicate anchors. In practice, this formulation can be greatly limiting, especially in the face of partial observability or symmetry.

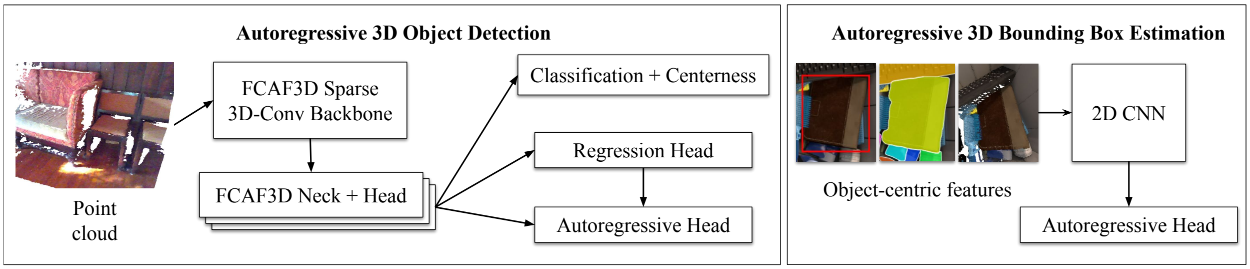

Autoregressive 3D Object Detection
Point cloud
FCAF3D Sparse 3D-Conv Backbone
FCAF3D Neck + Head
Classification + Centerness
Regression Head
Autoregressive Head
Autoregressive 3D Bounding Box Estimation
Object-centric features
2D CNN
Autoregressive Head

Figure 3.3: For indoor 3D Object Detection, we use FCAF3D as a base model with an autoregressive head for bounding box prediction. For 3D Bounding Box Estimation we take object-centric features from a 2D object detector and pass them into a 2D CNN for autoregressive bounding box prediction.

Autoregressive Modeling

We propose to tackle this problem by autoregressively modeling the components of a 3D bounding box. That is, for some ordering of the components (e.g. dimensions → center → orientation, or any permutation thereof), such a predictor will sequentially predict each component conditioned on the previous ones. In theory, the particular autoregressive ordering should not matter; empirically, we find that dimensions → center → orientation was effective, so we use this ordering for our model. Having dimension as first in the autoregressive ordering also enables us to condition on dimensions when they are known, which can be effective at improving the prediction accuracy in certain situations.

We discretize the box parameters rather than predicting continuous values, which is a well-known technique that allows the model to easily express multimodal distributions [92]. For rotations, we chose Euler angles since each dimension has a fixed range and does not to be normalized. To make discrete dimension and center predictions, we normalize those parameters so that they can fit within a fixed set of discrete bins. We normalize dimensions by some scale s so that most values of d/s are within the range $[0, 1]$, and offset the centers by c_0 so that most normalized centers $(c - c_0)/s$ are within the range $[-1, 1]$. We use 512 bins for each dimension and adjust the bin range to achieve on average ≥ 0.99 IOU with the quantized box and $< 0.1\%$ overflow or underflow due to quantization.

From RGB-D inputs we extract a fixed-dimensional feature vector h for each object. For each parameter $b = (d_x, d_y, d_z, c_x, c_y, c_z, \psi, \theta, \phi)$ in the autoregressive ordering, we model $p(b_i | b_1, \ldots, b_{i-1}, h)$ using a MLP with a few hidden layers. This autoregressive model is then trained using maximum likelihood. Figure 3.2 provides an overview.

$$\log p(b|h) = \sum_{i=1}^{9} \log p(b_i | b_1, \ldots, b_{i-1}, h) \tag{3.1}$$

Model Architectures

Our autoregressive prediction scheme can be applied to any type of 3D bounding box predictor. In this section, we discuss how it might be applied in two different contexts: 3D object detection and 3D bounding box estimation.

Autoregressive 3D Object Detection.

FCAF3D [78] is a state-of-the-art 3D object detection method that was heavily engineered to exploit sparse and unstructured point clouds. Given a colored point cloud, it applies a specialized feature extractor consisting of sparse 3D convolutions, and then proposes 3D bounding boxes following a popular single-stage detector, FCOS [88].

Autoregressive FCAF3D: We can make FCAF3D autoregressive by adding a head and training this head with maximum likelihood in addition to the FCAF3D loss $L_F(h, y)$ (Figure 3.3). We found that the pointwise box prediction was useful to condition the autoregressive prediction and estimate the scaling normalization factor $s = \max\{d'_x, d'_y, d'_z\}$, where d' is the pointwise dimension prediction of FCAF3D. Bounding box centers c are normalized by the output locations c_0 of the sparse convolutions and scaled by the same s: $(c - c_0)/s$. Since 3D object detection datasets have at most one degree of freedom for rotation, we predict only one θ parameter for box rotation.

To optimize the autoregressive prediction for higher IOU, we sample boxes $b \sim p(b|h)$ and maximize the IOUs of the samples with the ground truth box y. For this optimization, we use the conditional expectation b' where $b'_i = \mathbb{E}[b_i|b_1, \ldots, b_{i-1}, h]$ (since b' is differentiable) to maximize $IOU(b', y)$. We train autoregressive FCAF3D using the combined loss:

$$L(h, y) = L_F(h, y) - \log p(b|h) + \mathbb{E}_{b\sim p(b|h)}[1 - IOU(b', y)] \tag{3.2}$$

Autoregressive PV-RCNN: Lidar-based 3D object detectors, such as PV-RCNN [83], often have different architectures and inductive biases than indoor detection models. However, we show that our autoregressive box parameterization is agnostic to the underlying architecture by applying it to PV-RCNN. We propose Autoregressive PV-RCNN by extending the proposal refinement head to be autoregressive, modeling the residual Δr^α as discrete autoregressive $p(\Delta r^\alpha|h)$. Then, we add $-\log p(\Delta r^\alpha|h)$ to the total training loss.

Autoregressive 3D Bounding Box Estimation.

3D Bounding Box Estimation assumes that object detection has already been performed in 2D, and we simply need to predict a 3D bounding box for each detected object. To highlight that our autoregressive prediction scheme can be applied to any bounding box predictor, we chose a model architecture that is substantially different from FCAF3D. For each detected object, we take an object-centric crop of the point cloud, normals, and object mask as input to a 2D-CNN, producing a fixed feature vector h per object. This h is used as features for our autoregressive parameterization $p(b|h)$. See Figure 3.4 for more details on the architecture.

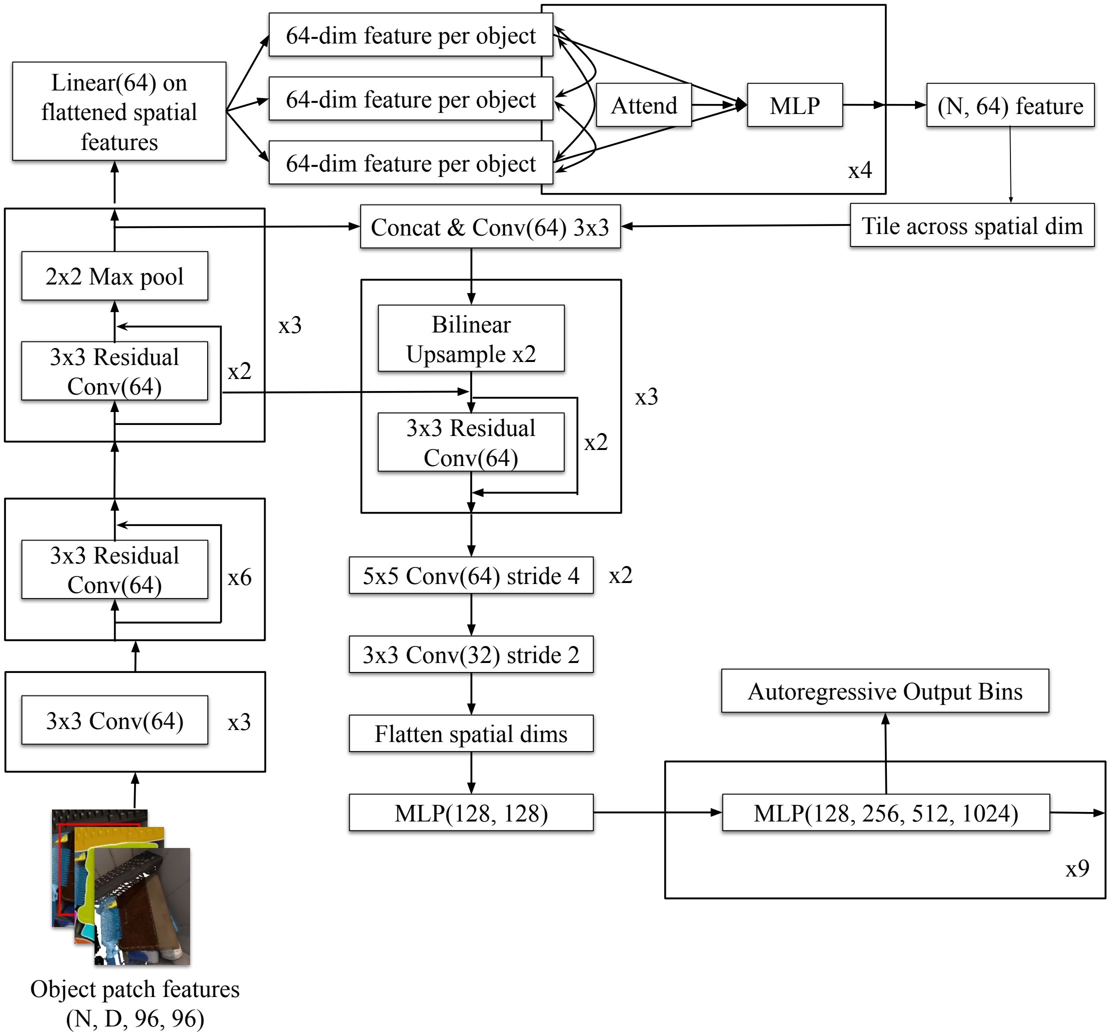
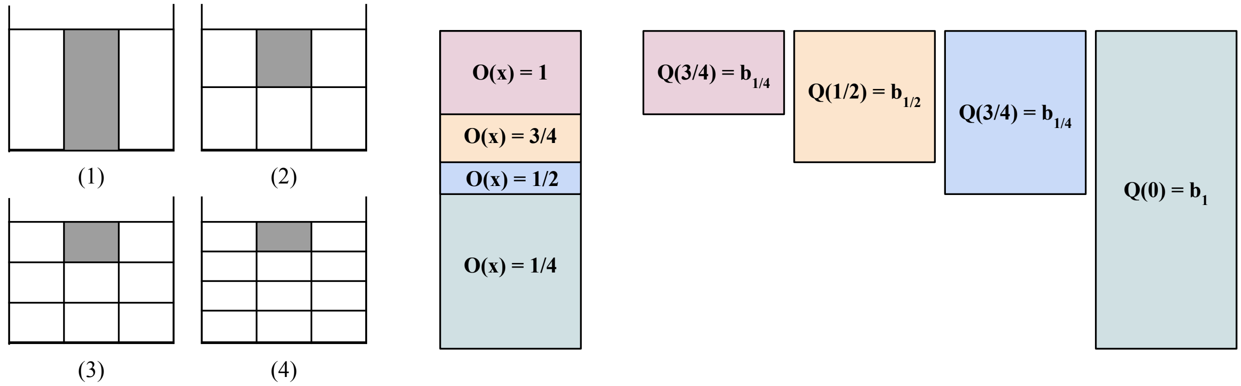


Figure 3.4: Overview of Autoregressive Bounding Box Estimation architecture

To normalize the input and box parameters, we scale by the range of the first and third quartiles of each point cloud dimension $s = Q_3 - Q_1$, and recenter by the mean of the quartiles $c_0 = \frac{Q_1+Q_3}{2}$. Within the complete space of $\mathcal{SO}(3)$ rotations, there are many box parameters that could represent the same box; for example, a box with $d = (1, 2, 3)$ is equivalent to a box with $d' = (2, 1, 3)$ and a 90° rotation. To account for this, we find all the box parameters $B = \{b^{(1)}, ..., b^{(m)}\}$ that represent the same box and supervise on all of them:

$$L(h, B) = -\frac{1}{|B|} \sum_{b^{(i)} \in B} \log(b^{(i)}|h) \tag{3.3}$$



a) **P(i) = 1/4** b) Occupancy c) Occupancy Quantiles Q(q) and Confidence Boxes $b_{1\text{-}q}$

Figure 3.5: Suppose we are estimating the bounding box of a tightly packed bin of stacked boxes. (a) There is not enough visual information to estimate the object height, however, we know that the object could have heights H/i for $i \in \{1, 2, 3, 4, \dots\}$ with equal probability. (b) We compute the occupancy $O(x)$ for different regions. (c) We visualize occupancy quantiles $Q(q)$ which correspond to confidence boxes b_{1-q}. Notice that as the confidence requirement increases, the size of the box increases to ensure we can contain the true object.

3.4 Applying Autoregressive 3D Bounding Box Models

Given a trained autoregressive bounding-box model, how should we actually obtain predictions from it? Like Latent-MaskRCNN, an autoregressive bounding box model produces expressive distributional output that enables flexibility, depending on how the downstream application plans to use the predictions.

Beam Search

In many applications, we simply want to obtain the most likely 3D bounding box given the input observation. That is, we seek the box $b^* = \arg\max_b p(b|h)$ which is most likely under the model. Finding b^* exactly can be computationally expensive, but we can approximate it using *beam search*, a technique that has proven especially popular for autoregressive models in natural language applications [18]. Beam search allows us to estimate the mode of the distribution learned by the model and serves as an effective pointwise prediction.

Quantile and Confidence Boxes

In applications such as robotics and autonomous driving, 3D bounding boxes are often used to estimate object extents and avoid collisions. To that end, we often care that an object o is fully contained in the estimated box b. For a given confidence requirement p, we define a confidence box b_p as a box that contains the true object o with probability at least p:

$\mathbb{P}(o \subseteq b_p) \geq p$. In this section we show how to extract confidence box predictions from an autoregressive bounding box model.

The motivating intuition is similar to that of Latent-MaskRCNN's confidence masks, as introduced in Section 2.5. Suppose we draw multiple samples from the model. If a point $x \in \mathbb{R}^3$ is contained in many boxes, then it is likely that point is actually part of the object. Conversely, a point that is only contained in a few sampled boxes is not likely to be part of the object. We can formalize this idea using the occupancy measure

$$O(x) = \mathbb{P}(x \in b) = \mathbb{E}_{b \sim p(b|h)}[\mathbb{1}\{x \in b\}] \approx \frac{1}{K} \sum_{i=1}^{K} \mathbb{1}\{x \in b^{(i)}\} \tag{3.4}$$

which can be approximated using samples $b^{(1)}, \dots, b^{(K)} \sim p(b|h)$ from our model.

To find regions that are very likely to be part of an object, consider the *occupancy quantile* $Q(q)$, the set of all points that have occupancy greater than q:

$$Q(q) = \{x : O(x) > q\} \tag{3.5}$$

We define the *quantile box* b_q as the minimum volume bounding box that contains the occupancy quantile $Q(q)$:

$$b_q = \arg \min_{b : Q(q) \subseteq b} \text{vol}(b) \tag{3.6}$$

Under some conditions, we can show that quantile boxes are confidence boxes.

Theorem 1 *A quantile box with quantile q is a confidence box with confidence $p = 1 - q$ when $p(b|h)$ is an ordered object distribution.*

We say that $p(b|h)$ is an ordered object distribution if for any two distinct boxes b_i, b_j in the sample space of $p(b|h)$, one box must be contained within the other, $b_i \subset b_j$ or $b_j \subset b_i$. Empirically we find that quantile boxes exhibit similar uncertainty characteristics to confidence boxes even when $p(b|h)$ is not an ordered object distribution. See Figure 3.5 for a visualization of occupancy and confidence boxes. The full proof of Theorem 1 is in Appendix B.

As described in Algorithm 3, quantile boxes can be computed extremely efficiently, in polynomial time and easily batchable on a GPU without custom CUDA kernels. We use a sample of K boxes to approximate the occupancy and a sample of KM points to approximate the occupancy quantile $Q(q)$. To find the minimum volume box, we assume that one of the sampled box rotations will be close to the optimal quantile box rotation. For each sampled rotation, we calculate the corresponding dimension/center of the minimum volume box that contains the occupancy quantile. Empirically, we find that $K = 64$, $M = 4^3$ provides a good trade-off of variance and inference time. With these parameters, quantile box computation for 15 objects takes no more than 10ms on a NVIDIA 1080Ti GPU.

Algorithm 3: Efficient Quantile Box Computation

```
Given: Desired quantile $q$, box distribution $P(b|h)$, numbers of box samples $K$,
  number of point samples $M$
Sample $b^{(1)}, ..., b^{(K)} \sim P(b|h)$ boxes
For each $b^{(i)}$, sample $M$ random points within $b^{(i)}$, adding all points to a set $T$
For all $x \in T$, estimate $O(x) = \frac{1}{k} \sum_i^K \mathbb{1}\{x \in b^{(i)}\}$
Construct the occupancy quantile $Q(q) = \{x \in T : O(x) > q\}$
for $b^{(1)}, ..., b^{(K)}$ do
    Let $R_i$ be the rotation of $b^{(i)}$
    Compute the volume of the $Q(q)$ bounding box under $R_i$,
      $v_i = \prod_{a \in x,y,z} (\max_{x \in Q(q)} (R_i^{-1} x)_a - \min_{x \in Q(q)} (R_i^{-1} x)_a)$
Find the minimum volume box $i^* = \arg\min_i v_i$
Let $s_a = \max_{x \in Q(q)} (R_{i^*}^{-1} x)_a$, $t_a = \min_{x \in Q(q)} (R_{i^*}^{-1} x)_a$
Return box $b = (d, c, R_{i^*})$ with dimensions $d = (t_x - s_x, t_y - s_y, t_z - s_z)$ and center
  $c = R_{i^*}(s_x + d_x/2, s_y + d_y/2, s_z + d_z/2)$
```

Dimension Conditioning

For some robotics applications, such as object manipulation in industrial settings, we are often presented with Stock-Keeping Unit, or SKU, information beforehand. In these scenarios, the dimensions of each SKU are provided, and the prediction task essentially boils down to correctly assigning the dimensions to a detected object instance, and predicting the pose of the 3D bounding box.

The autoregressive nature of our model allows for conveniently conditioning on the dimensions of each bounding box. However, we don't know which object in the scene corresponds to which SKU dimensions. How can we leverage dimension information from multiple SKUs without object-SKU correspondence? Our autoregressive model provides an elegant solution using conditioning and likelihood evaluation.

Given $\{d^{(1)}, ..., d^{(k)}\}$ known SKU dimensions, we can make a bounding box prediction using this information by maximizing:

$$b^* = \arg\max_b \{ \max_{d^{(1)},...,d^{(k)}} p(b|d^{(i)}, h)\} \tag{3.7}$$

We can find the optimal b^* by using beam search conditioned on each of the d_i and returning the box with the highest likelihood. Figure 3.1 illustrates how dimension conditioning can be used to greatly increase the fidelity of the predicted 3D bounding boxes.

3.5 Experiments

We designed our experiments to answer the following questions:

1. How does an autoregressive bounding box predictor perform compared to a pointwise predictor, across a variety of domains and model architectures?
2. How meaningful are the uncertainty estimates from an autoregressive model? How effective are quantile boxes for reasoning about uncertainty?

Datasets

To demonstrate the flexibility of our method, we conducted experiments on a diverse set of datasets.

SUN-RGBD [86] is a real-world dataset containing monocular images and point clouds captured from a stereo depth camera. It features a large variety of indoor scenes and is one of the most popular benchmarks in 3D object detection. The box labels only include one rotational degree of freedom θ.

ScanNet [15] is a dataset of indoor 3D reconstructions. There are 18 classes and the bounding box labels are axis-aligned (no rotation). We train on 1201 scenes and evaluate on 312 validation scenes.

KITTI [20] is a widely popular 3D detection dataset for autonomous driving. Objects in KITTI have one degree of rotational freedom θ, and we report evaluation results on the validation split.

COB-3D: *Common Objects in Bins 3D* is a simulated dataset we rendered to be more representative of the challenges encountered in real-world robotic applications. It contains 7000 scenes that aim to emulate industrial environments where robots perform manipulation-based tasks, with each scene consisting of a bin containing a variety of items. Compared to existing datasets, the objects are in a greater range of orientations than any other 3D-bounding-box dataset. In many indoor and autonomous-driving datasets, the ground-truth bounding boxes typically only exhibit one rotational degree of freedom, since the objects tend to be situated in stable poses on the ground. However, the diversity of object arrangements in COB-3D necessitates reasoning about complete 3D rotation. COB-3D also exhibits many types of ambiguity including rotation symmetry, occlusion reasoning in cluttered scenes, and tightly-pack bins with unobserved dimensions. See Appendix A for full details on this dataset, including visual examples.

Evaluation

To evaluate 3D-bounding-box predictions, *intersection-over-union*, or IoU, is commonly used to compare the similarity between two boxes. 3D object detection uses mean average precision, or mAP, to measure how well a detector trades off precision and recall. IoU is used to determine whether a prediction is close enough to a ground-truth box to constitute a true positive. For

Table 3.1: 3D Object Detection on SUN-RGBD, Scannet, and KITTI

		IoU			IoG		
Dataset	Method	$AP_{0.25}$	$AP_{0.50}$	AP_{all}	$AP_{0.25}$	$AP_{0.50}$	AP_{all}
SUN-RGBD	FCAF3D	63.8	48.2	37.42	64.72	59.82	48.75
	3DETR	59.52	32.17	31.13	63.00	53.33	44.08
	VoteNet	60.71	38.98	30.25	62.81	54.58	43.62
	ImVoteNet	**64.24**	39.38	31.12	**67.00**	57.41	45.78
	Beam Search	62.94	47.03	38.15	64.75	58.50	47.17
	Quantile 0.1	61.21	30.94	31.06	65.89	**64.34**	**60.08**
	Quantile 0.4	63.46	48.41	38.43	65.34	61.68	51.76
	Quantile 0.45	63.47	**48.64**	**38.55**	65.19	61.03	50.36
	Quantile 0.5	63.30	47.70	38.50	64.99	59.83	48.44
ScanNet	FCAF3D	68.53	**53.87**	43.32	72.05	67.63	60.66
	3DETR	64.09	47.16	39.57	68.62	59.17	49.82
	Beam Search	**69.06**	53.67	**43.85**	71.46	66.10	59.13
	Quantile 0.1	67.10	43.13	34.17	72.23	**70.01**	**66.73**
	Quantile 0.2	68.03	48.68	38.27	**72.30**	69.68	65.43
	Quantile 0.4	68.73	52.98	42.76	72.08	67.74	61.98
KITTI		AP IoU Hard Split			AP IoG Hard Split		
	Method	Car	Ped.	Cycl.	Car	Ped.	Cycl.
	PVRCNN	**82.37**	53.12	68.69	91.86	67.08	73.14
	Beam Search	**82.37**	52.28	**69.13**	91.84	66.96	73.40
	Quantile 0.1	59.75	39.26	58.38	**96.02**	**71.85**	**76.09**
	Quantile 0.4	81.98	**54.15**	68.45	93.98	70.63	74.08
	Quantile 0.5	82.32	53.78	69.03	91.84	68.14	73.52

3D bounding-box estimation, detection has already happened, so we simply measure the mean IoU between the prediction and ground-truth, averaged across objects.

Unlike 2D detection, many applications that use 3D bounding boxes especially care about underestimation more than overestimation: if the predicted bounding box is too large, that is generally a less costly error than if it is too small. In the latter case, there are parts of the object that are outside the bounding box, which may result in collisions in robotics or autonomous driving settings.

To help quantify this error asymmetry, we consider a new similarity functions, the *intersection-over-ground-truth* (IoG). IoG measure what fraction of the ground truth box is contained within the predicted box; when IoG is 1, the ground truth box is fully contained in the predicted box. With IoG and IoU, we have a more complete understanding of the types of errors that a bounding-box predictor is making. For the detection task, we compute mAP separately using IoU and IoG, and for the estimation task, we compute the mean IoG along with the mean IoU.

3D Object Detection

In Table 3.1, we evaluate the autoregressive box parameterization for 3D object detection on SUN-RGBD, ScanNet, and KITTI. We compare Autoregressive FCAF3D and Autoregressive PV-RCNN, both introduced in Section 3.3, to several other state-of-the-art methods.

We find that beam search generally matches or exceeds the baseline performance on IoU AP_{all}. With quantile boxes, we find that lower quantiles result in higher IoG mAP, which suggests that the predicted boxes are more likely to contain the ground truth box. This is consistent with Theorem 1, since lower quantiles correspond to higher confidence boxes and must contain the true object with higher probability. We find that quantile boxes 0.4-0.5 strike the best balance between IoU and IoG, achieving better mAP than baselines in most cases. The quantile parameter flexibly enables applications to trade off bounding box accuracy (as measured by IoU) with containment probability (as measured by IoG).

3D Bounding Box Estimation

We evaluate bounding box estimation on COB-3D using the model architecture described in Section 3.3. To study the effectiveness of our autoregressive parameterization, we train the same model architecture with different box parameterizations and losses. All models receive the same 2D detection results and features as input, and must make 3D bounding box predictions for each detected object. We consider 4 baseline parameterizations for this task inspired by various works in the literature:

L1 Regression: In this parameterization, the model outputs 9 real values for each of the 9 box parameters: $b = (d_x, d_y, d_z, c_x, c_y, c_z, \psi, \theta, \phi)$. The model predicts dimensions and centers in coordinates normalized around the object's point cloud. This model is trained using a L1 loss over the normalized box parameters $L(b, g) = ||b - g||_1$, where g is the ground truth box [63].

Gaussian: For this baseline, the model outputs 18 real values for the mean, μ, and log-variance, $\log \sigma^2$, of 9 Gaussian distributions $\mathcal{N}(\mu, \sigma)$ over the box parameters b [31, 60]. Predicting the variance enables the model to output uncertainty over different box parameters, independently of each other. We train this model using maximum likelihood: $L(\mu, \log \sigma^2, g) = -\sum_i \log \mathcal{N}(g_i; \mu_i, \sigma_i)$.

Discrete: In some prior works, box parameters are predicted as discrete bins but not in an autoregressive manner [72]. To evaluate this parameterization and ablate the necessity of autoregressive predictions, we predict each box parameter *independently* as discrete bins: $\log p(b|h) = \sum_{i=1}^{9} \log p(b_i|h)$

4-Point: This baseline outputs 12 real values for four 3D corner points $(p_0, p_1, p_2, p_3) \in \mathbb{R}^3$, constituting a 3D bounding box [60, 59]. We ensure that the 3D bounding box is orthogonal by applying the Gram-Schmidt process on the basis vectors $(p_1 - p_0, p_2 - p_0, p_3 - p_0)$. We use an L1-loss on the difference between the predicted points and the points of the ground truth 3D bounding box. Since there are many permutations of valid 4-point corners of a bounding box, we supervise on the permutation that induces the minimum loss.

Table 3.2: Results of the proposed method & baselines on our dataset. We also show results for conditioning our method on ground truth dimensions.

	IoU	IoG	F1	$err_{dim}[m]$	$err_{quat}[rad]$	$err_{center}[m]$
L1 Regression	0.4219	0.6113	0.4992	0.0436	0.4667	0.0138
Discrete	0.5232	0.6282	0.5709	0.0339	0.2926	**0.0105**
Gaussian	0.3169	0.5304	0.3967	0.0450	0.5154	0.0119
4-Point	0.5688	0.7113	0.6321	0.0332	0.1999	0.0132
Beam Search	**0.6296**	0.7877	0.6999	**0.0287**	**0.1598**	0.0109
Quantile 0.1	0.3821	**0.9723**	0.5486	0.0986	0.1762	0.0123
Quantile 0.4	0.5949	0.8871	0.7122	0.0377	0.1640	0.0110
Quantile 0.5	0.6275	0.8295	**0.7126**	0.0318	0.1657	0.0110
Conditioning	0.6709	0.7899	0.7215	0.0086	0.1674	0.0096

Metrics

To more quantifiably reason about the trade-off between IoG and IoU, we report the corresponding F1-score: $\text{F1}_{score} = \frac{2(IoU*IoG)}{IoU+IoG}$. We further report metrics on the dimension & pose errors, which are computed as follows:

- $err_{dim} = \text{sum}(|\mathbf{d} - \mathbf{d}_{gt}|)$, where we compute the error across all possible permutations and then choose the one with the smallest error.
- $err_{quat} = 2\arccos(|\langle \mathbf{q}, \mathbf{q}_{gt}\rangle|)$, where $\mathbf{q}$ represents the rotational part of the pose as a quaternion. We compute the error across all possible symmetries and choose the one with the smallest error.
- $err_{center} = ||\mathbf{c} - \mathbf{c}_{gt}||_2$, where $\mathbf{c}$ is the 3D-center of the bounding box.

Results

Table 3.2 compares different prediction modes of our autoregressive model to the baseline parameterizations. We find that *Beam Search* achieves the best IoU, dimension & rotation error. As for the *Quantile* methods, we find that lower quantiles achieve higher IoG while sacrificing IoU and dimension error. *Quantile 0.5* offers the best tradeoff in terms of overall performance, achieving higher IoG with similar IoU and dimension error compared to *Beam Search*. Baseline models that predict box parameters directly generally performed worse since those models cannot properly capture multimodal correlations across the box parameters. The *Discrete* baseline performs the best in terms of center error, but we can see that the best autoregressive methods are only a few millimeters worse. For bounding box predictions with full rotations in $\mathcal{SO}(3)$, we find that an autoregressive bounding box parameterization can effectively model rotation uncertainty, achieving the lowest rotation error. We can also see that conditioning

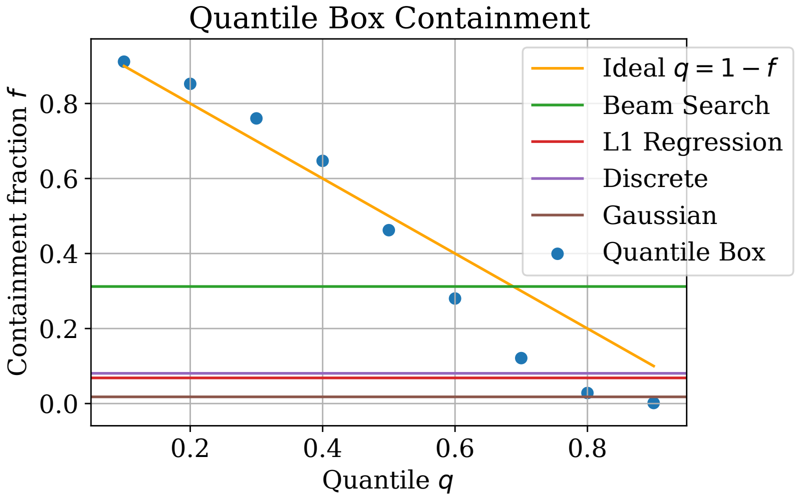


Figure 3.6: We compare fraction of predicted boxes that contain ground truth boxes f with different quantiles q and find that q-quantile boxes contain approximately $f \approx 1 - q$ fraction of ground truth boxes.

the model on known dimensions of the items in the scene increases performance in all relevant metrics (besides IoG), most notably in IoU & dimension error. Note that the dimension error is non-zero because the model is given the dimensions as an unordered set, and still needs to predict the association of each dimension tuple to the corresponding item in the scene.

Quantile and Confidence Boxes

In Section 3.4 we introduced quantile boxes as a tractable approximation for confidence boxes. In particular, we showed that when $p(b|h)$ is an ordered object distribution, a quantile box with quantile q is equivalent to a confidence box with $p = 1 - q$ and should contain the true object with probability p.

While real world objects generally do not follow an ordered distribution (and it can be hard to verify if they are or not), we can empirically evaluate whether q confidence boxes contain the ground truth object $1 - q$ fraction of the time. To test our hypothesis, we predict quantile boxes with different q and calculate the fraction of predictions f with IoG > 0.95. In Figure 3.6, we can see that $f \approx 1 - q$ and follows a generally linear relationship. This suggests that even for general object distributions, quantile boxes can be an effective approximation for confidence boxes, and that the uncertainty exposed by quantile parameter is actually well-calibrated.

3.6 Discussion

We introduced an autoregressive formulation to 3D bounding prediction that greatly expands the ability of existing architectures to express uncertainty. We showed that it can be applied to both 3D object detection and 3D bounding-box estimation, and explored different ways to extract bounding box predictions from such autoregressive models. In particular, we showed how the uncertainty expressed by these models can make high confidence predictions and meaningful uncertainty estimates. We introduced a dataset that requires predicting bounding boxes with full 3D rotations, and showed that our model naturally handles this task as well. While autoregressive models are just one class of distributionally expressive models, they are not the only option for more expressive bounding box modeling. We hope that future lines of work will continue to build upon the method, dataset, and benchmarks we introduced in this chapter.

Chapter 4

Shape Completion Models for Dense Packing of Complex, Novel Objects

4.1 Introduction

So far we have presented two techniques for improving the ability of state-of-the-art perception models to reason about uncertainty in challenging or ambiguous situations. Studying instance segmentation (Chapter 2) and 3D bounding box prediction (Chapter 3), we showed how to learn models that express distributional uncertainty over complex hypotheses, and how to flexibly incorporate this uncertainty into real-world robotics systems. However, even our improved models still require large training datasets, which can be challenging to obtain in diverse real-world environments. While human annotators can provide the supervision, at a nontrivial cost, for instance segmentation or 3D bounding box prediction, there are other perception tasks where annotation would be prohibitively impractical.

In this chapter, we delve into *shape completion*, one such task where annotation is not a feasible route to obtaining supervision. The goal of shape completion is typically to predict the entire 3D shape of an object (for example, a mesh, voxel grid, or surface point cloud) based on a partial observation, such as an incomplete or noisy point cloud. Due to the inherent partial observability, shape completion models are forced to learn strong priors about 3D object geometry, making them a good fit for real-world applications where the ability to generalize robustly in the face of inherent ambiguity is essential.

Robust real-world shape completion can form the perceptual foundation for *dense packing*, a manipulation task where a robot must pack objects into a given container, maximizing the density or number of objects in the final configuration. Dense packing is a critical capability for robotic pick-and-place systems for applications in warehouse automation and logistics, which have seen huge commercial interest in recent years. For example, in order fulfillment for e-commerce, objects must be packed into shipping boxes that will be sent from a warehouse to a customer. Suboptimal packing causes inefficiencies in the overall operation, as larger boxes, more shipments, or human interventions will be unnecessarily required.

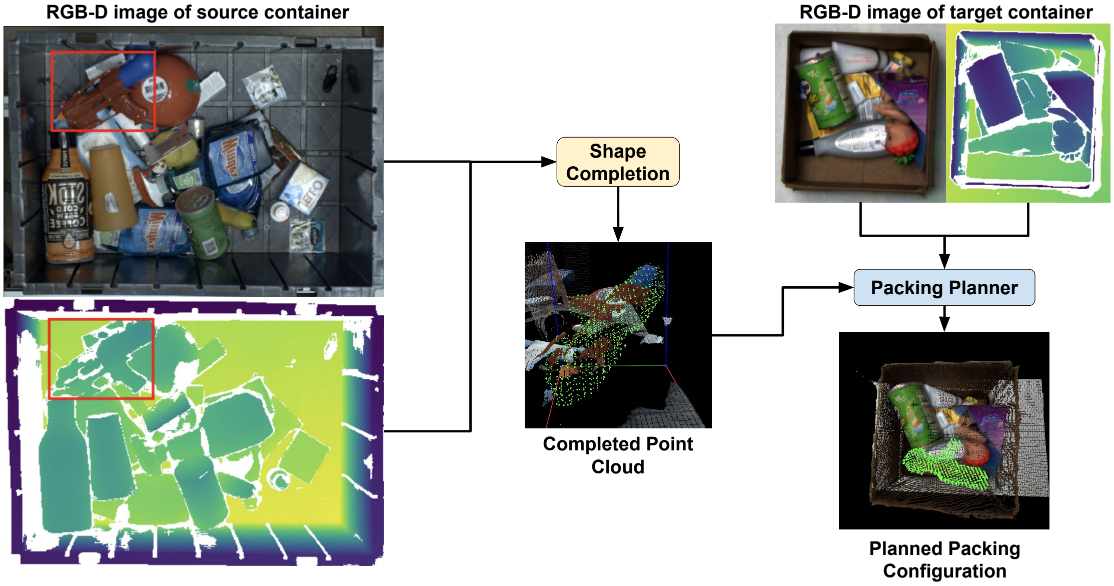

RGB-D image of source container
RGB-D image of target container
Shape Completion
Completed Point Cloud
Packing Planner
Planned Packing Configuration

Figure 4.1: Our proposed shape completion model, F-CON, is trained in simulation and predicts completed point clouds for complex, unseen objects in the real world. Off-the-shelf planners can leverage F-CON for precise dense-packing in pick-and-place applications.

Recent work in dense packing has sidestepped the real-world perception challenges, working in simulation to focus sequential-decision-making aspect of the problem: in order to achieve optimal packing densities, every object needs to be placed carefully, with regard for how it affects the subsequent objects that will need to be packed into the same container. The result of this line of work has been a series of attempts to cast dense packing as a reinforcement learning (RL) problem [45, 100, 38]. For tractability and ease of evaluation, common practice has been to operate on simplified state representations, such as by approximating all objects as cuboids, or to assume that complete state information is available, such the ground-truth geometry of all objects [95, 94].

In this chapter, we push the boundaries of *sim-to-real transfer* to develop a shape completion model that provides the necessary perceptual understanding for real-world dense packing systems. Our main contributions are as follows:

(1) We release a simulated dataset, COB-3D-v2, that can be used to train shape completion models for robotics applications. COB-3D-v2 is an expansion of the COB-3D dataset introduced in Chapter 3, and it is publically available at our project page.

(2) We propose a 3D fully-convolutional model architecture for shape completion that performs well in the robotics domain. We show that our model achieves state-of-the-art performance on COB-3D-v2.

(3) Through extensive real-world experiments, we show that our model trained on COB-3D-v2 can be combined with simple, off-the-shelf planning methods to enable state-of-the-art dense packing performance on cluttered scenes with complex, novel items.

4.2 Related Work

Dense packing has been mostly studied from a planning perspective: in what order and pose should the items be placed to maximize the packed density? Early work proposed heuristics like Deepest-Bottom-Left-First (DBLF) or Heightmap-Minimization (HM) [96, 95]. Besides their simplicity, these heuristics are attractive because they empirically perform well even in situations where the entire item set to be packed is not know in advance. More recent work has attempted to learn policies using reinforcement learning (RL), exploring state/action representations, reward functions, neural network architectures, and RL algorithms that work best for this problem domain [45, 100, 38]. However, the RL-for-packing work has mostly been limited to simulated evaluations of cuboid objects, limiting its applicability to real-world systems. Methods for 3D bounding-box estimation could help extend this work to arbitrary objects, but the imprecision of the bounding-box representation would likely result in suboptimal packing performance. Our experiments will explore this in Section 4.4.

Packing of complex objects in the real world has only been explored under restricted settings, such as where the ground-truth object geometry is known in advance, where the items are already singulated, or where the items only need to be packed in a 2D planar configuration [94, 24]. These simplifications reduce the perception requirements necessary for a packing system, but do not reflect the challenges encountered in real-world applications. In this work, we consider a more realistic setting where the items must be picked from a cluttered bin and packed into a dense 3D arrangement. As we show in Section 4.4, the strong geometric priors learned by our shape completion model enable existing planning methods to gracefully handle this difficult task. Additionally, we explicitly evaluate the achieved packing density, which, to the best of our knowledge, has not been explored in prior work.

Methods for shape completion can be roughly categorized by the particular 3D representation that they predict. Recent work has focused on implicit functions, which offer the best accuracy and resolution, but have limited ability for generalization and are computationally expensive during inference [67, 57]. While such methods are incredibly effective in many graphics applications, these properties make them a poor fit for robotic systems that need to handle unseen objects in low-latency applications. Other representations like voxel grids and unstructured point clouds are more computationally tractable to work with; voxel grids tend to be more amenable to prediction with neural networks, but scale poorly to extremely high resolutions. We will discuss how these trade-offs influence our system in Section 4.3

In robotics, shape completion has mostly seen attention in the context of grasping. There have been labor-intensive attempts to collect real-world training data by taking RGB-D captures of objects with known meshes, and then use the resulting shape completion models to plan parallel-jaw grasps on singulated objects [93]. Later work attempted to train shape completion models on existing, generic, simulated datasets, and used the predictions to evaluate both grasp quality and kinematic/collision feasibility during placement [24]. However, the poor real-world performance of their shape completion model necessitated substantial focus on how the planning algorithm could reason about perceptual errors. We evaluate that model as a baseline in Section 4.4.

4.3 Dense Packing with Convolutional Occupancy Models

Frustum-Convolutional Occupancy Networks

The shape completion problem is typically posed as follows: given a partial point cloud of an object, the goal is to produce a complete point cloud of the entire object surface, including invisible or occluded portions. This is particularly amenable to robotics, where RGB-D cameras can provide partial point clouds of varying quality. In existing benchmarks, scenes typically contain only a single object, or the partial point clouds are already segmented into objects. However, for a real-world application where objects appear in cluttered scenes, we also require access to an instance segmentation model. Model families such as Mask-RCNN [30] or DETR [7] are relatively mature and have been extensively used in robotic applications.

Given an RGB-D image and corresponding instance segmentation, our model predicts voxels for each object using a 3D fully-convolutional architecture. We find that inference with voxel representations is still efficient at the resolutions we care about, and the convolution-based network architecture is the best way to impose strong inductive biases when working with structured data like RGB-D images.

As illustrated in Figure 4.2, we first construct a trapezoidal frustum for each object. Each frustum is the projection of its 2D bounding-box into 3D, clipped between a near plane and a far plane. The near plane is chosen to be slightly closer to the camera than the nearest point in the partial point cloud in the object's instance mask, and the far plane is chosen to be conservatively far based on the working volume of the scene. We then discretize each frustum into a voxel grid, and associate a feature vector of dimension C with each voxel, resulting in a feature volume of shape $C \times D \times H \times W$ for each instance. For all experiments, we chose $D = 96, H = W = 64$ for a favorable trade-off between inference speed and performance. The voxels are trapezoidal, but they are aligned with the camera viewpoint (for a given h, w coordinate, every voxel along the D dimension is on the same ray entering the camera, and projects to the same pixel in the image plane), and they are spaced linearly along the D dimension between the near and far plane. For each point in the partial point cloud, we fill the corresponding voxel with the RGB color and a binary indicator for the instance mask ($C = 4$). The camera-centric and object-centric properties of this scheme encourage robustness to different camera viewpoints, scene composition, and object sizes, which improves the sim-to-real transfer performance. A similar frustum-based scheme was used by Mesh-RCNN [23]; however, they predict voxels jointly with instance masks, and do not condition on a partial point cloud. The latter is desirable in applications where depth is not available during inference, but depth cameras are already ubiquitous in robotics and can substantially improve performance.

The initial $C \times D \times H \times W$ feature volume is passed through a 3D-convolutional UNet to produce an updated feature volume of the same shape. We then use a 3D-conv layer to reduce the C dimension to 1, and then refine the resulting $D \times H \times W$ feature map with a

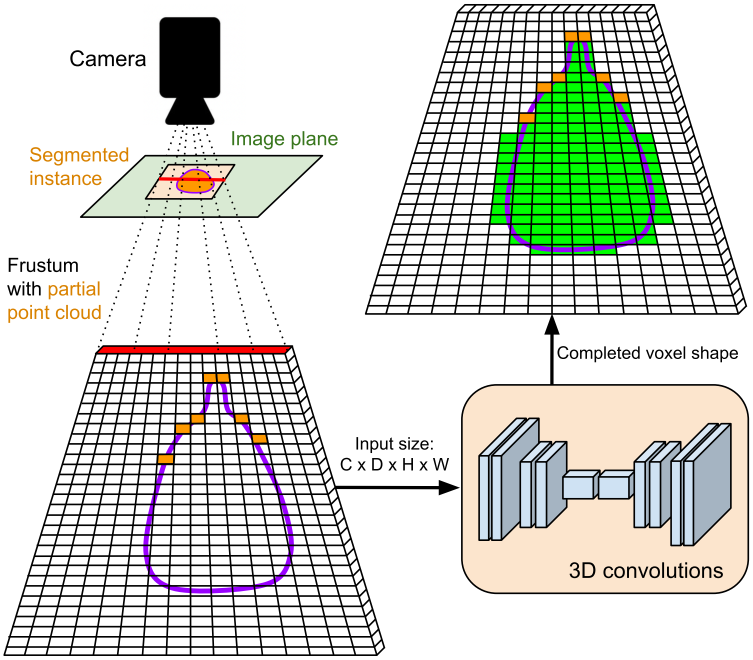


Figure 4.2: F-CON unprojects a segmented instance into a camera-aligned frustum, populates the frustum with the instance's partial point cloud (orange voxels), and then applies a series of 3D convolutions to produce a voxel grid of the completed shape (green). The ground-truth instance shape (purple contour) is used as supervision during training. Here we only visualize a single slice of the frustum (indicated by the red line in the image plane). In practice, F-CON unprojects the entire region-of-interest for each object.

2D UNet (where the D dimension is treated as the channel dimension). This is an efficient way to increase the expressiveness of the model, since 2D convolutions are cheaper than their 3D counterparts. Each element of the 2D UNet's output (still $D \times H \times W$) is the scalar probability that the corresponding voxel is occupied by the object's completed shape. During training, we label each voxel as positive if it is contained inside the object's ground-truth mesh. Each voxel is supervised independently using a class-balanced binary cross-entropy loss. During inference, we extract meshes from the voxel predictions using Marching Cubes [54]. sample points uniformly from the surface to obtain an unstructured point cloud. For more details, see our implementation.

We call this model F-CON (Frustum-Convolutional Occupancy Network). In the following sections, we discuss the dataset used to train it, and how we utilize it for dense packing in the real world.

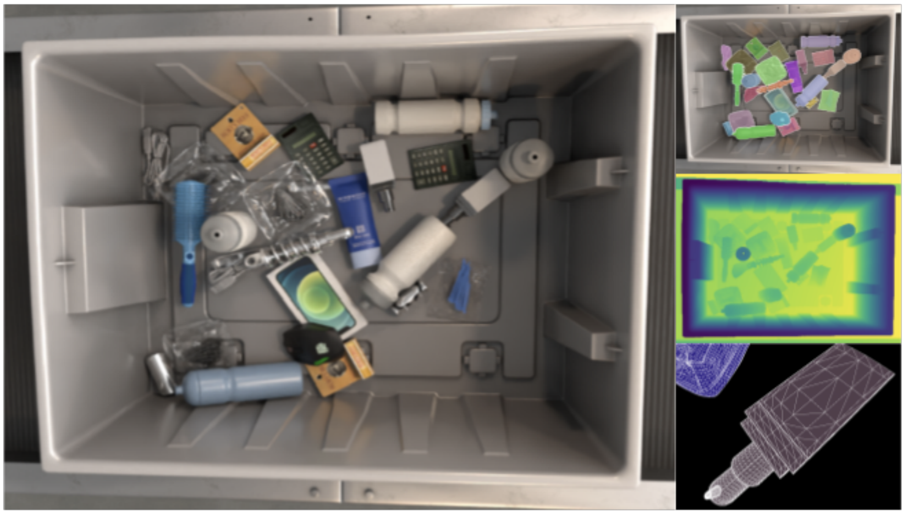


Figure 4.3: COB-3D-v2 contains high quality RGB renderings (left), instance masks (top right), and depth maps (middle right), and posed meshes for each instance (bottom right). It can be downloaded from our project page. See Appendix A for more details and examples.

Simulated Dataset

In Section 3.5, we introduced COB-3D, a simulated dataset of common objects in bins, arranged in realistic yet challenging configurations. As illustrated in Figure 4.3, the dataset contains roughly 7000 scenes of high-quality RGB renderings along with ground-truth camera calibrations, instance masks and point clouds. Each scene contains up to 30 objects, which are thrown into a bin using physics simulation. For robust sim-to-real transfer, the object sizes, camera parameters, and scene lighting are all randomized.

In this work, we released a new version of this dataset, COB-3D-v2, with ground-truth meshes and poses of each object instance. This addition allows shape completion models such as F-CON to be trained on COB-3D-v2. For more example scenes and details about the dataset format, see Appendix A.

Note that neither COB-3D nor COB-3D-v2 have object categories (all objects belong to a single category). This is a substantial departure from prior work in shape completion, where the standard practice is to either train category-specific models or condition on the object category [8]. However, we find that the lack of categories is more representative of real-world settings with novel objects, which may belong to arbitrary novel categories, or may be hard to categorize in the first place.

We trained F-CON for 125 epochs on COB-3D-v2, which took about 4 GPU-days. We used a batch size of 32 scenes and the Adam optimizer with default hyperparameters ($\alpha = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$). During training, we also randomize the near and far planes for each instance, which improves robustness after sim-to-real transfer. In Section 4.4, we evaluate F-CON against baselines for shape completion on COB-3D-v2.

Dense Packing with F-CON

Given a trained shape completion model, there can be different ways to utilize it in a dense-packing system. To highlight F-CON's perceptual capabilities, we opted for a simple planning pipeline that allows us to use off-the-shelf methods from prior work. As described in Algorithm 4, we determine a grasp pose $g^* \in SE(3)$ and placement pose $q^* \in SE(3)$ from the following inputs:

- A height-map $H[\cdot, \cdot]$ for the target container, which is computed from the captured depth map. The container is discretized into rectangular cells, where $H[u, v] = z$ if the highest sensed point in cell (u, v) is distance z from the container bottom.
- A set of candidate grasp poses $G = \{g^{(1)}, \dots, g^{(N)}\}_{i=1}^{N}, g^{(i)} \in SE(3)$. These can be generated using any method and for any grasping modality (suction, parallel-jaw, etc).
- Completed point clouds $O = \{o^{(i)}\}_{i=1}^{N}$ for each grasped object. Each $o^{(i)} \in \mathbb{R}^{K \times 3}$ is expressed in the $g^{(i)}$ frame.
- A cost function $C(g, q) \to \mathbb{R}$ that evaluates the packing quality of a candidate grasp g and placement q.

Given the returned grasp and placement poses, we use a scripted trajectory planner such that the robot's gripper retracts linearly upwards from the grasp pose, and descends linearly downwards during placement.

```
Algorithm 4: A simple planner for dense packing
Input : Height-map $H$ for the target container
        Candidate grasps $G = \{g^{(1)}, \dots, g^{(N)}\}$
        Object point clouds $O = \{o^{(1)}, \dots, o^{(N)}\}$
        Placement cost function $C(g, q) \to \mathbb{R}$
Initialize $g^* \leftarrow \text{null}, q^* \leftarrow \text{null}, c^* \leftarrow \infty$;
for $k = 1, \dots, M$ do
    Sample a grasp $g^{(k)}$ from $G$;
    Sample a placement cell $(u^{(k)}, v^{(k)})$ inside $H$;
    Sample a placement orientation $R^{(k)}$;
    Compute the lowest placeable height $z^{(k)}$ (see Figure 4.4) for object $o^{(k)}$, when
      centered at $H[u^{(k)}, v^{(k)}]$ and rotated by $R^{(k)}$;
    Compute the gripper pose $q^{(k)}$ corresponding to $(i^{(k)}, j^{(k)}, z^{(k)}, R^{(k)})$;
    Evaluate $c^{(k)} \leftarrow C(g^{(k)}, q^{(k)})$;
    if $c^{(k)} < c$ and MotionFeasible$(g^{(k)}, q^{(k)})$ then
        $g^* \leftarrow g^{(k)}, q^* \leftarrow q^{(k)}, c^* \leftarrow c^{(k)}$;
Result: Best placement $q^*$, corresponding grasp $g^*$
```

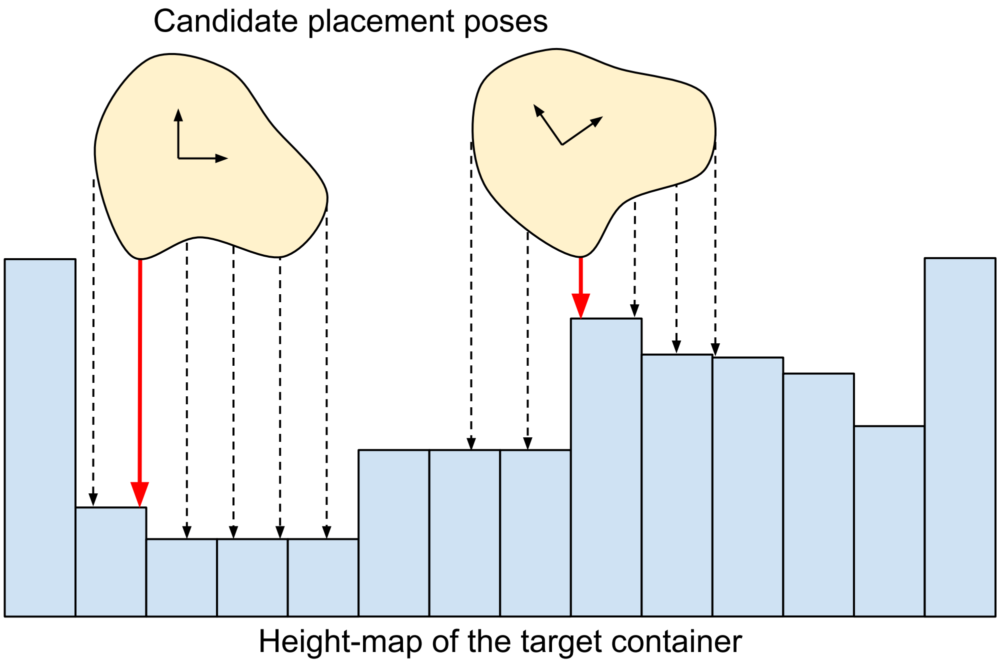


Figure 4.4: A 2D-slice of the lowest-placeable-height computation from Algorithm 4. For a candidate placement pose, every point in the completed shape (yellow) is projected (black/red arrows) onto the target container's height-map (blue). The shortest projection distance (red arrows) determines the height at which the object can be placed for that cell and orientation. In this example, we visualize the projections for two placement candidates. The left candidate results in a lower placement than the right.

This framework neatly encapsulates existing planning methods like DBLF and HM. Using $C(g, q) = q_z + \epsilon \cdot (q_x + q_y)$, for $0 < \epsilon << 1$, yields the DBLF planner. The HM planner estimates the height-map $H'(q)$ that would result from placement q, and then uses $C(g, q) = \sum_{u,v} H'(q)[u, v] - H[u, v]$. For more details about how $H'(q)$ is computed, see [95]. Combining F-CON with a model-based RL method (such as by extending the simulated packing work discussed in Section 4.2) could potentially result in a better cost function than either DBLF or HM, as well as a better sampler than the uniform one that we use. However, to focus on evaluating F-CON, we defer such explorations to future work.

Unlike some prior work, we do not consider re-grasping, where already-packed items may be removed in order to achieve a better arrangement. Re-grasping allows the system to mitigate the effects of perceptual mistakes made in the preceding timesteps, which confounds our evaluation of shape completion models.

4.4 Experiments

We conducted extensive real-world experiments to answer the following questions:

(1) How effective is F-CON at shape completion, as trained and evaluated on the realistic-but-simulated scenes in COB-3D-v2?

(2) To what extent does F-CON address the perceptual difficulties surrounding dense packing, as evaluated on novel objects in cluttered real-world scenes?

COB-3D-v2 Evaluation

To evaluate shape completion on COB-3D-v2, we considered the following metrics:

- **Chamfer distance**: This is the standard metric for comparing unstructured point clouds in benchmarks such as ShapeNet [8]. Given two point clouds X and Y, the Chamfer distance (CD) is computed as follows:

$$\mathrm{CD}(X,Y) = \frac{1}{|X|}\sum_{x\in X}\min_{y\in Y}\|x-y\|_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\min_{x\in X}\|y-x\|_2^2 \qquad (4.1)$$

  Often, an L1-variant (CD-L1), where the $\|\cdot\|_2^2$ norms are replaced by $\|\cdot\|_1$, is reported alongside the traditional Chamfer-L2 distance (CD-L2).

- **$\mathbf{F1}^\tau$**: For a given distance threshold τ, predicted point cloud X and ground-truth point cloud Y , $\mathrm{F1}^\tau(X,Y)$ is the harmonic mean of the precision at τ (fraction of points in X that are within τ of some point in Y) and the recall at τ (fraction of points in Y that are within τ of some point in X). This metric is usually considered alongside the Chamfer distance in shape completion benchmarks because it is less sensitive to outliers, and is typically reported at varying values of τ.

- **Box IoU, IoG, F1**: Given a completed point cloud, a 3D bounding-box can be fit around it and compared it to the ground-truth bounding-box, using metrics from 3D bounding-box estimation (Chapter 3). In contrast with Chamfer distances, and especially $\mathrm{F1}^\tau$, we find that bounding-box metrics are very sensitive to outliers. However, they may be more representative of packing performance, since outliers can cause a packing system to believe that an item cannot fit in a pose where it actually could have. To fit a bounding box around a point cloud, we sample rotations uniformly at random in quaternion space [43], compute the axis-aligned dimensions of the enclosing box in each sampled rotation frame, and finally choose the sampled box with the smallest volume (see Figure 4.5 for a visualization). Using this scheme, we report IoU, IoG, and F1: IoU is the standard metric for bounding-box estimation, IoG (intersection-over-ground-truth) is a form of recall to complement IoU, and F1 trades off between the two (Section 3.5). Note that IoG is not particularly meaningful in isolation, since perfect IoG can be achieved by simply predicting arbitrarily large bounding-boxes.

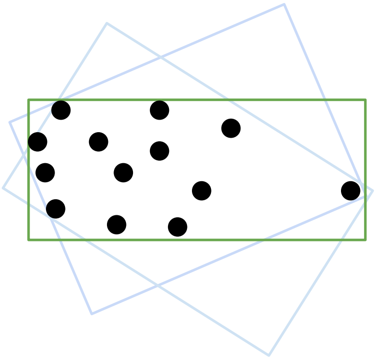


Figure 4.5: Fitting a bounding box around a point cloud (black dots). We sample several candidate boxes that contain all the points (blue), and then take the one with minimum volume (green). Notice that the single point on the far right substantially influences the dimensions of the fitted box.

We considered the following methods as baselines to F-CON. Like prior work in shape completion for grasping, we did not consider implicit functions: they must be queried extremely densely to extract surface geometry, and often require test-time optimization, making them too computationally expensive during inference for use in a real-world system [67, 57].

- PCN [105] has been used in prior work for grasp and placement planning [24]. It uses an encoder that embeds a partial cloud into a latent space, and a decoder that constructs the completed point cloud from the latent vector. Both encoder and decoder use PointNets [71] to operate directly on unstructured point clouds directly, and they are trained end-to-end using the Chamfer-L2 distance as the loss function. To train PCN on COB-3D-v2, we normalize each instance's partial point cloud using its frustum, decode the completed point cloud in the normalized coordinates, and then transform it back to the original space. For a more fair comparison with F-CON, we also improve upon the original architecture by concatenating RGB and instance masks with the partial point cloud.

- PoinTr [104] uses a similar encoder-decoder framework to PCN, but substantially improves the architecture, primarily by using Transformers. At the time of this work, it achieved state-of-the-art performance on many shape completion benchmarks, even outside of robotics. We train PoinTr using the same normalization scheme, additional inputs, and loss function as PCN.

- The autoregressive bounding-box model (AR-bbox) from Chapter 3. Since bounding boxes have been used as a simplified state representation in prior packing work (as discussed in Section 4.2), we also consider this model as a baseline, but only evaluate it on bounding-box metrics.

Table 4.1: Benchmarking Shape Completion on COB-3D-v2

	Shape completion					3D bounding-box estimation		
Method	CD-L1 (↓)	CD-L2 (↓)	$F1^{0.1}$(↑)	$F1^{0.3}$(↑)	$F1^{0.5}$(↑)	Box-IoU (↑)	Box-IoG (↑)	Box-F1 (↑)
AR-bbox (Chapter 3)	-	-	-	-	-	0.6296	0.7877	0.6999
PCN [105]	0.9835	0.5800	0.0729	0.5460	0.7968	0.5374	0.8002	0.6460
PoinTr [104]	0.4857	0.1582	0.3721	0.8717	0.9555	0.5874	**0.8394**	0.6984
F-CON (ours)	**0.4229**	**0.1157**	**0.4664**	**0.8928**	**0.9600**	**0.6809**	0.7686	**0.7485**

Both Chamfer distance and $F1^{\tau}$ generally depend on both the scaling and density of the point clouds. Following prior work [23], we scale all point clouds such that the longest edge of the ground-truth bounding-box has length 10. The ground-truth point clouds are generated by sampling 16384 points uniformly from the mesh surface. For F-CON, we sample points in the same way, but from the meshes obtained via Marching-Cubes. PCN and PoinTr always output a fixed number of points, so we train them accordingly to predict 16384 points.

In Table 4.1, we see that F-CON outperforms the baseline methods across all shape completion metrics. We find these results particularly compelling given that F-CON is not trained to minimize Chamfer distance, unlike PCN and PoinTr. We also note that F-CON performs well on 3D bounding-box metrics, even though PCN and PoinTr do not. Qualitatively, we observe PCN and PoinTr are prone to outliers in their predicted point clouds – this is well-reflected in their $F1^{\tau}$ and IoG scores.

Real-World Dense Packing

We designed real-world experiments to mimic typical order fulfillment applications, where a robot must pack items from a cluttered bin (or often, multiple bins) into a smaller container, like a cardboard box. As shown in Figure 4.6, we used an ABB-1200 with a 5-cup suction gripper, with RGB-D cameras mounted above both containers. The item set consists of a variety of household objects of diverse shapes and categories. In total, we have 35 objects, which are unseen by all models (since all models are trained purely in simulation on COB-3D-v2).

Recall that the goal of dense packing is to minimize the volume that the packed objects occupy. Thus, we evaluate our system by directly estimating this quantity at the end of each episode, using a scheme inspired by the HM heuristic. To the best of our knowledge, no prior work has evaluated real-world packing quality with a continuous volumetric measure (a common practice is simply to check whether all items were successfully placed in the container). Using the target container's height-map, as defined in Section 4.3, we can estimate the total volume occupied by the packed objects via numerical integration over the cells of the height-map. The HM planner minimizes the change in this quantity for each item to be packed; while tractable and easy to implement, this is generally not optimal when considering the entire episode.

For each shape completion model from Section 4.4, we use the planner from Algorithm 4 using either DBLF or HM as the cost function $C(g, q)$, height-map cells of 1 mm × 1 mm,

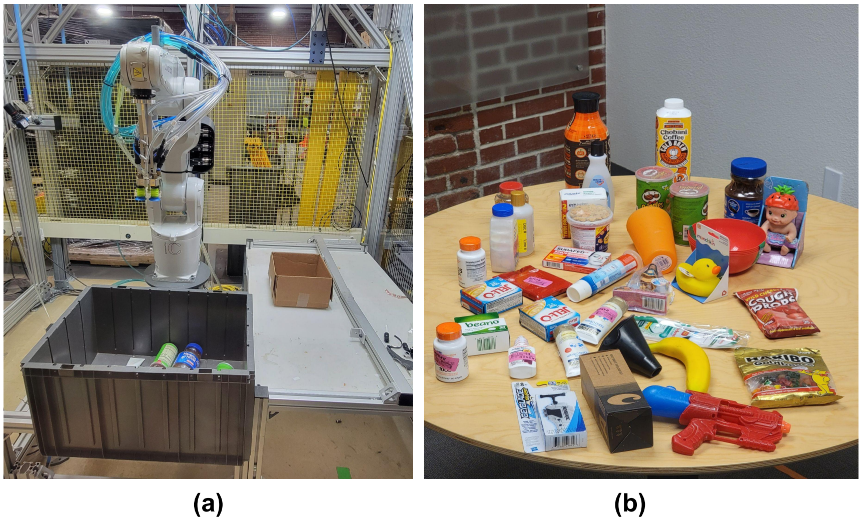


Figure 4.6: (a) Our robot picks items from the cluttered bin in the front, and packs them into the cardboard box on the the table. (b) The complete item set used in our experiments.

and a sample size of $M = 4096$ placements. Since F-CON operates in an object-centric manner, its inference time scales with the number of objects in the scene. For a large scene (16 objects), it takes about 25 milliseconds on an NVIDIA Quadro RTX 600. The entire planning process (perception, sampling placements, scoring, motion planning) takes about 300 milliseconds.

Within each trial, we select a subset ranging from 5 to 15 items uniformly at random from the overall item set, and arrange them chaotically in the source container. Each trial consists of one episode for each (*shape completion*, *planner*) configuration, wherein the system packs the sampled items one-by-one into the target container. The episode ends when either all items are placed, one is placed that causes the container to overflow, or the system cannot find an overflow-free plan. At the end of each episode, we estimate the packed volume using the height-map integration scheme discussed in the previous paragraph. Finally, we measure human performance by quickly packing the items by hand (taking 20 seconds or less), and measuring the packed volume in the same manner.

In Table 4.2, we show the performance of each shape completion model paired with different off-the-shelf packing planners, alongside human performance, across 50 trials. We report the mean and standard error for the following metrics:

- **Success Rate**: the fraction of episodes where all items were successfully packed at the end of the episode.

- **Packed Volume**: the volume measured by height-map integration at the end of the episode. We express this value as a fraction of the total container volume. In episodes that are not successes, we record a value of 1.0, which is the worst possible score and corresponds to using the entire container.

Table 4.2: Real-World Dense Packing

Shape Completion	Planner	Packed Volume (↓)	Success Rate (↑)
AR-bbox (Chapter 3)	DBLF	0.698 ± 0.047	0.48 ± 0.071
PCN [105]	DBLF	0.909 ± 0.035	0.12 ± 0.046
PoinTr [104]	DBLF	0.570 ± 0.046	0.66 ± 0.067
F-CON (ours)	DBLF	$\mathbf{0.461} \pm 0.041$	$\mathbf{0.80} \pm 0.057$
AR-bbox (Chapter 3)	HM	0.762 ± 0.045	0.38 ± 0.069
PCN [105]	HM	0.664 ± 0.049	0.50 ± 0.071
PoinTr [104]	HM	0.535 ± 0.039	0.76 ± 0.060
F-CON (ours)	HM	$\mathbf{0.440} \pm 0.032$	$\mathbf{0.88} \pm 0.046$
Human	Human	0.357 ± 0.012	1.00 ± 0.000

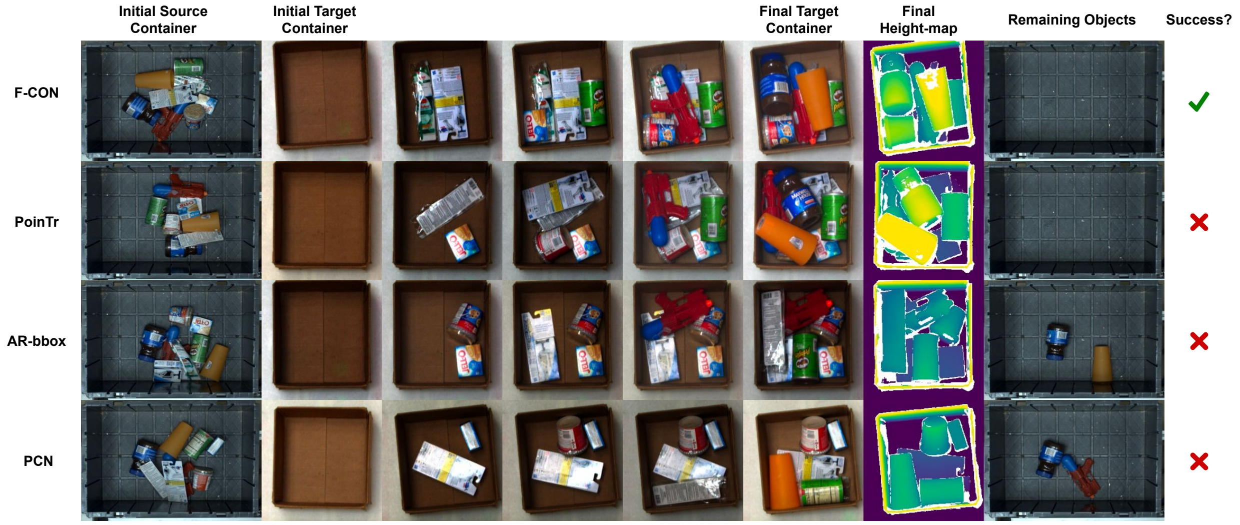


Figure 4.7: Representative episodes from our real-world packing experiments. In each row, we conduct one episode using the listed shape completion model and HM as the planning method. All rows use the same item set. The progress within the episode can be seen from left to right, along with the final height-map. With F-CON (row 1), all of the items can be packed densely. Other methods cause the packing container to overflow (row 2) or they cannot pack all the items (rows 3, 4).

For all planners, F-CON substantially outperforms the other shape completion methods, demonstrating its efficacy for real-world dense packing. In Figure 4.7, we visualize representative episodes for a qualitative understanding. With other methods, the system often cannot find a suitable placement pose or causes the target container to overflow. The former typically results from overestimation of the object's size, which is consistent with the F1$^{\tau}$ and IoG results discussed in Section 4.4.

4.5 Discussion

We presented F-CON, a voxel-based shape completion model with strong inductive biases, and validated it on the highly-realistic simulated dataset COB-3D-v2. We then conducted extensive experiments in the real-world, and showed that the strong geometric priors learned by F-CON can enable dense-packing of complex, unseen items in chaotic, cluttered scenes, without any real-world training. Although using F-CON results in substantially better performance that baseline shape completion methods, we limited our packing system to simple planning methods in order to demonstrate F-CON's efficacy. Combining F-CON with RL methods to obtain learned packing policies is an exciting direction for future work, and we hypothesize that this could close the gap with respect to human performance.

However, F-CON ultimately owes its success to its training data, the COB-3D-v2 dataset. A lot of manual engineering went into the development of this dataset, maximizing its visual and physical realism to give sim-to-real transfer the best chance of succeeding. What would we have done if the sim-trained F-CON had transferred poorly to the real-world, despite our best efforts? We will explore this theme in the next chapter.

Chapter 5

Closing the Visual Sim-to-Real Gap with Object-Composable NeRFs

5.1 Introduction

Sim-to-real transfer is the standard technique for training perception models where supervision is unreliable or impractical to obtain in the real-world. In Chapter 4, we saw a successful instantiation of this recipe that applied sim-to-real shape completion to the downstream task of dense packing, but sim-to-real transfer is generally a fickle and labor-intensive art.

Models trained in simulation often degrade when deployed in the real-world due to distribution shift; the performance discrepancy is often known as the *sim-to-real gap.* Domain randomization [89] tries to reduce the sim-to-real gap by observing that if a model is trained to generalize to different parameters of the simulator (viewpoint, lighting, material properties, etc), then it may also generalize to the real world as simply another randomization. However, this requires the training distribution to be both diverse enough to enable generalization, as well as realistic enough that the real world is a plausible sample from this distribution. This can be difficult to achieve in practice, often requiring manual asset creation (meshes, textures, materials), scene construction, and parameter tuning. When a large sim-to-real gap is observed, it can be difficult to know exactly how to improve the simulator, and actually doing so may require specialized expertise (e.g. graphics for visual tasks), or domain-specific or task-specific tuning. Even if the sim-to-real gap is initially small, the engineering effort may still need to occur continuously, as a deployed system may encounter changing environments or new scenarios that were not reflected in the original data.

In this chapter, we explore neural rendering methods as a mechanism to close the sim-to-real gap for perception tasks. The seminal NeRF [61] uses real images to build a neural representation of a scene, which can then be rendered from new viewpoints through a ray-marching process based on physical models of light transport. However, these methods typically require extensive test-time optimization (TTO), taking many GPU-hours to reconstruct a single scene. In this chapter, we introduce Composable Object Volume NeRF, or

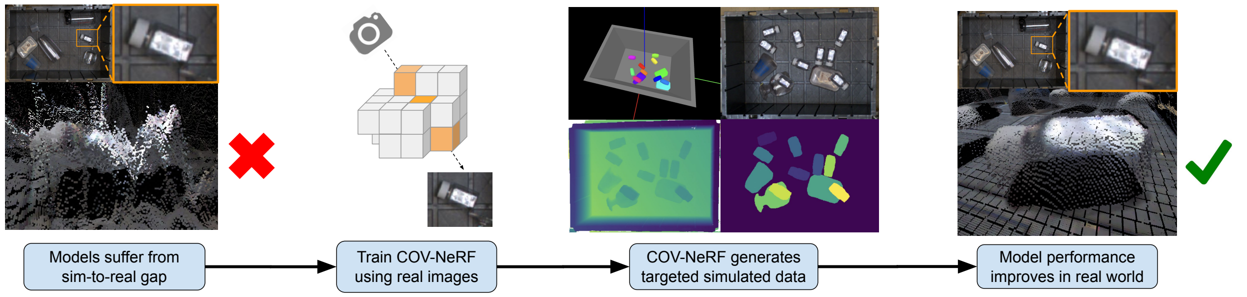

Models suffer from sim-to-real gap
Train COV-NeRF using real images
COV-NeRF generates targeted simulated data
Model performance improves in real world

Figure 5.1: Our proposed object-centric neural renderer, COV-NeRF, can be used to generate targeted supervision for other models that are brittle to sim-to-real distribution shift. After learning explicit neural representations of real objects, COV-NeRF can compose those representations into photorealistic synthetic scenes and generate many modalities of downstream supervision, including depth maps, segmentation masks, and instance meshes.

COV-NeRF, a novel neural renderer that is uniquely suited to remedy sim-to-real mismatch. Unlike prior NeRF methods, COV-NeRF explicitly represents objects while also generalizing across scenes without TTO. It performs real-to-sim learning of structured neural representations that can subsequently be used for novel scene synthesis, rendering photorealistic images along with the corresponding supervision for many perception tasks. Existing methods for domain adaptation simply re-skin simulated scenes [35, 33, 73], while COV-NeRF synthesizes new scenes *de novo*, can produce both RGB images and many types of 2D and 3D supervision, and automatically enjoys geometric and semantic consistency across viewpoints and between the rendered images and labels.

Our key contributions are as follows:

1. We introduce COV-NeRF, a novel NeRF architecture that both explicitly represents objects and generalizes across scenes. Although these properties have been explored individually in prior work; COV-NeRF is the first to exhibit both simultaneously.

2. We demonstrate COV-NeRF's ability to generate targeted synthetic data based on real-world scenes and objects. Without artificial consistency constraints, we show that COV-NeRF can generate supervision for a variety of perceptual tasks, including depth estimation, object detection, instance segmentation, and shape completion, and that training on COV-NeRF's generations improve real-world performance.

3. Finally, we apply the entire pipeline to a real world bin-picking application. We identify challenging scenarios where state-of-the-art perception models and simulated datasets face a large sim-to-real gap, and show that COV-NeRF's real-to-sim capabilities can rapidly close the gap to achieve application-level improvement.

5.2 Related Work

Neural Rendering: NeRF [61] was a breakthrough in physically-plausible differentiable rendering, yielding better visual quality and easier optimization than existing methods for inverse rendering. However, NeRF trains a new neural network for each scene, which requires 50-100 source images and several GPU-days of TTO. More recent methods, such as PixelNeRF [101], MVS-NeRF [9], and NerFormer [75], use more specialized architectures that can simultaneously represent many scenes and generalize to new scenes without TTO. Since they learn priors that transfer across scenes, they only need 3-5 source views during inference.

Object-NeRF [99] and Object Scattering Functions (OSF) [102] stayed within the single-scene paradigm, but explored object-centric decompositions that allowed scene editing, where objects can be explicitly added, re-positioned, or scaled. OSF's edited scenes are the most realistic, but it requires the lighting to be specified parametrically (which can be difficult outside of controlled settings), and an order of magnitude more compute during rendering. Subsequent work [44] [84] adds bells-and-whistles to Object-NeRF; COV-NeRF extends these methods to not require TTO.

In robotics, single-scene NeRFs have been explored for online geometry estimation, but were bottlenecked by the computational cost of TTO [36] [39]. NeRF-Supervised Deep Stereo [91] used single-scene NeRFs to generate labels for stereo depth estimation. This pipeline generated high-quality supervision, but required extensive TTO and many source views per scene. GraspNeRF [16] trained NeRF rendering in simulation as an auxiliary loss for grasp generation, but did not explore or evaluate its rendering quality, let alone its ability to mitigate sim-to-real mismatch. GraspNeRF's architecture is similar to MVS-NeRF; as such, we expect similar rendering performance, and it does not share COV-NeRF's object-composable properties.

Sim-to-Real Transfer: Domain randomization has been explored for sim-to-real transfer of both visual and physical tasks. Early visual sim-to-real work [89] focused on diversity rather than realism: these methods achieved promising results on simple tasks and environments, but could not scale to more challenging applications without manual tuning. Simple heuristics like cut-and-paste were explored for segmentation, but produce visually and geometrically implausible scenes, and cannot generate 3D supervision [21] [85].

Domain adaptation methods employed generative models to automatically improve the realism of simulated images. CyCADA [35] trained GANs to translate simulated images into real ones, but required a cycle consistency objective so that the original semantic segmentations could still serve as supervision for the translated images. Subsequent work applied this idea using object detection [33] and Q-values [73]. However, finding such a bijective mapping is an ill-posed task, and its difficulty scales poorly as more modalities are considered (e.g. if we require consistency for depth maps as well as for semantic segmentation). We take a fundamentally different approach: COV-NeRF creates datasets *de novo* that are more realistic and more meaningfully diverse than the adaptions produced by these methods, and achieves multi-view consistency for many modalities for free. We evaluate domain adaptation methods in Section 5.4.

Inspired by recent advances in vision-language alignment, CACTI [56], ROSIE [103], and GenAug [11] used text-conditioned diffusion models [34] to automatically apply augmentations, such as changing backgrounds or adding distractor objects. These methods improved the robustness of end-to-end policies, but ultimately suffer from the same limitations as the GAN-based approaches.

5.3 Generalizable Neural Rendering with Composable Object Volumes

Preliminaries

NeRF methods use differentiable volumetric rendering to train models for novel view synthesis. Given a collection of source images $\{\mathcal{I}^{(b)}\}_{b=1}^{B}$ of a scene and their corresponding camera intrinsic/extrinsic matrices, their goal is to render an image $\mathcal{I}^*$ of the scene as seen from a new viewpoint.

Recall that a ray $\mathbf{r}$ is parametrized as $\mathbf{r}(t) = o + t \cdot d, t \geq 0$, where $o, d \in \mathbb{R}^3$ are the ray's origin and direction. Each pixel $(u, v) \in \mathcal{I}^*$ corresponds to a particular ray $\mathbf{r}_{u,v}(t)$; the intrinsics and extrinsics determine o and d. To render a ray, points $\mathbf{r}(t_i), t_1 < \cdots < t_N$ are sampled along it, and the density σ_i and radiance c_i are determined for each, then accumulated to compute the ray's rendered color $\hat{C}(\mathbf{r})$:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i c_i \tag{5.1}$$

where $\delta_i = t_{i+1} - t_i$, $\alpha_i = 1 - e^{-\delta_i \sigma_i}$, $T_i = e^{-\sum_{j<i} \delta_j \sigma_j}$. The rendered color of pixel (u, v) is simply $\hat{C}(\mathbf{r}_{u,v}(t))$.

Early NeRFs used fully-connected neural networks to predict σ_i and c_i from the coordinate $\mathbf{r}_{u,v}(t_i))$ and the ray direction d. However, since they lacked direct access to the source images during inference, they had to distill visual details from the source images into the network weights via optimization, which prevented them from representing multiple scenes. Moreover, since the networks were trained from scratch for each scene, many source views were required ($B \approx 100$).

In the scene-generalizable extensions discussed in Section 5.2, the dependence on $\{\mathcal{I}^{(b)}\}$ was made explicit by making these images inputs to the network during rendering. More powerful architectures, such as 3D-CNNs and Transformers, were introduced to better leverage the additional information. Instead of memorizing visual details, the networks learned to robustly aggregate information from the source images, and could work with substantially fewer source views ($B \approx 3$).

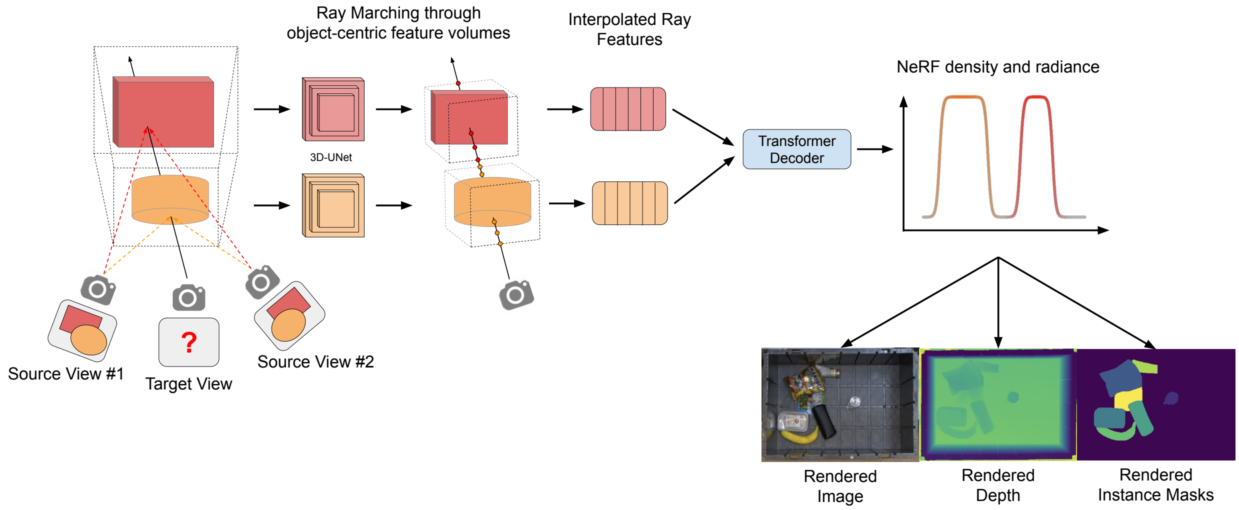


Figure 5.2: An overview of COV-NeRF's object-centric rendering process. Visual features from the source views are projected into a feature volume for each object in the scene. For each pixel to be rendered, features are interpolated along the corresponding ray from each volume that the ray intersects with. A Transformer decodes the interpolated features into the NeRF density and radiance, which are composited into RGB colors, depths and segmentation masks.

Composable Object Volumes

For synthetic data generation, we seek a NeRF method that both generalizes across scenes and explicitly represents objects. These properties ensure that we can generate supervision for different perceptual tasks, that the generated supervision is consistent with the rendered images, and that we can scalably generate diverse datasets with many objects in a variety of configurations.

COV-NeRF represents a scene as a collection of objects $\{o_k\}_{k=1}^{K}$ and a background. Each object is defined by a frustum, which is discretized into a $D \times H \times W$ voxel grid. A learned feature vector is associated with each voxel, resulting in a feature volume $V^{(k)} \in \mathbb{R}^{C \times D \times H \times W}$. The object $o^{(k)}$ is fully specified by its pose $p^{(k)} \in \mathcal{SO}^3$ and volume $V^{(k)}$.

Each $V^{(k)}$ is computed by projecting the RGB values from the source views, resulting in an initial volume of shape $B{\times}3{\times}D{\times}H{\times}W$. We aggregate features across source views via attention over the B dimension, reducing the volume to $C{\times}D{\times}H{\times}W$, and then refine it with a 3D-UNet to produce $V^{(k)}$. MVS-NeRF [9] uses a similar plane-sweep scheme, but constructs one volume for the entire scene, while we construct one for each object.

To render a ray, we gather contributions from each object and the background. For the background, we sample points $r(t_i^{(0)}), t_1^{(0)}, \ldots, t_N^{(0)}$ throughout the entire scene, and predict $\sigma_i^{(0)}, c_i^{(0)}$ using a simplified NerFormer [75], which interpolates features from each source view and processes them with a Transformer. For each object $o^{(k)}$, if the ray intersects with its frustum, we also sample points $r(t_i^{(k)}), t_1^{(k)}, \ldots, t_N^{(k)}$. We transform $r(t_i^{(k)})$ and d into the object frame $p^{(k)}$, trilinearly interpolate feature vectors from $V^{(k)}$, and use a Transformer to decode them into $\sigma_i^{(k)}$ and $c_i^{(k)}$ following [75, 52].

To accumulate these values into $\hat{C}(\mathbf{r})$, we sort all the $\{t_i^{(k)}\}_{i=1,k=0}^{i=N,k=K}$ and apply Equation 5.1. This natively handles occlusions: even if $\sigma_i^{(k)}$ is large, $T_i^{(k)}$ may be small if the ray passes through other objects before hitting object o_k.

Following prior work, the product $\alpha_i T_i$ can be interpreted as the probability that the ray terminates at distance t_i. Then the expected distance $\hat{\tau}(\mathbf{r})$ that the ray travels is:

$$\hat{\tau}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i t_i \tag{5.2}$$

$\hat{\tau}(\mathbf{r})$ is the point-to-point distance from the camera center, but can be trivially converted into a depth value in the camera frame. COV-NeRF renders depth maps by applying this process to each pixel in the image, considering all objects and the background.

This probabilistic interpretation can also be leveraged to render instance masks. In addition to standard (modal) instance masks, which indicate the pixels where an object is visible, COV-NeRF can also render amodal masks, which include both visible and occluded portions of each object.

Let $M_{u,v}^{(k)} \in [0, 1]$ be the probability that pixel (u, v) belongs to object o_k's modal mask, and let $\bar{M}_{u,v}^{(k)}$ be the corresponding probability for its amodal mask. $M_{u,v}^{(k)}$ is simply the probability that $\mathbf{r}_{u,v}$ terminates inside o_k, and $\bar{M}_{u,v}^{(k)}$ is the probability that $\mathbf{r}_{u,v}$ would terminate inside $o^{(k)}$ in the absence of other objects:

$$M_{u,v}^{(k)} = \sum_{i=1}^{N} T_i^{(k)} \alpha_i^{(k)} \tag{5.3}$$

$$\bar{M}_{u,v}^{(k)} = \sum_{i=1}^{N} \bar{T}_i^{(k)} \alpha_i^{(k)}, \text{ where } \bar{T}_i^{(k)} = \exp\big(-\sum_{j<i} \delta_j^{(k)} \sigma_j^{(k)}\big) \tag{5.4}$$

For more details about COV-NeRF's architecture and rendering, see our implementation.

Training COV-NeRF

Like other scene-generalizable NeRF methods, COV-NeRF learns visual and geometric priors that enable inference when only a few source images are available. To best learn these priors, we train COV-NeRF on a mix of simulated and real data: simulation helps bootstrap its understanding of 3D geometry, and real data exposes it to realistic textures, materials, and lighting. This gives us the best of both worlds: COV-NeRF can leverage dense geometric supervision when available, but is also robust to sim-to-real mismatch, since it can be trained on any real data that it does not initially generalize to. We jointly train the following losses:

- View synthesis: the rendered color (Equation 5.1) is trained to match the true color using an L2-loss.
- Depth estimation: the rendered depth (Equation 5.2) is trained to match the true depth using an L1 loss.
- Instance segmentation: the rendered masks (Equation 5.3) are trained to match the ground-truth masks using a cross-entropy loss.
- Voxel occupancy: we use $V^{(k)}$ to predict the occupancy of each voxel in that object's feature volume, similar to [23] and the F-CON model introduced in Chapter 4. This facilitates novel scene synthesis (see Section 5.3) since it lets us automatically pose objects into physically plausible configurations.

In practice, we found that a compute-efficient scheme is to pretrain all of the above losses in simulation, and add real data when it becomes available. We pretrain COV-NeRF for 100 epochs on the COB-3D-v2 dataset introduced in Section 4.3, which takes about 1 day using 8 NVIDIA RTX A5000s. We used a batch size of 32 scenes and the Adam optimizer with default parameters ($\alpha = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999$).

Scene Generation with COV-NeRF

Single-scene NeRF methods (see Section 5.2) anecdotally explore scene-editing as a benefit of object-centric rendering. We extend this capability to perform novel scene synthesis, procedurally generating new scenes at scale, for use in a downstream application. When sim-to-real models struggle in the real world, COV-NeRF can synthesize new training data targeting specific real-world scenes and objects.

Given a pre-trained COV-NeRF and collection of real scenes, we first perform real-to-sim finetuning using the captured images and any additional supervision that may be available (such as instance mask annotations). Although this step is not strictly required, the ability to use weak 2D supervision as learning signal for the underlying 3D semantics and geometry (for which direct real-world supervision usually cannot be obtained) is a strength unique to COV-NeRF, and is only possible because of its object- and ray-centric structure. Next, we extract feature volumes and meshes for objects from the real scenes. The meshes are extracted from the voxel occupancy predictions (Section 5.3) via Marching Cubes. To compose a new scene, we sample objects and then use a physics simulator (we use Mujoco [90]) to construct a physically plausible configuration. We sample initial poses for each object and place the meshes accordingly in the simulator, advance physics until all objects settle, and then render the scene with each object at its final pose using COV-NeRF. The process is described in Algorithm 5.

Algorithm 5: Scene Generation with COV-NeRF

```
Input : Desired number of scenes $N$
        Feature volumes $\mathcal{V} = \{V^{(1)} \ldots, V^{(K)}\}$
        Background source images $\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(B)}\}$
for $i$ = 1, ..., $N$ do
    Sample camera viewpoints
    Sample a number of objects $K^{(i)}$
    Initialize physics simulator (e.g. Mujoco)
    for $k = 1, \ldots, K^{(i)}$ do
        Sample object $V^{(k)} \sim \mathcal{V}$ and corresponding mesh
        Sample initial pose $p_0^{(k)}$
        Add mesh to simulator at pose $p_0^{(k)}$
    Advance physics for $T$ timesteps (e.g. until all objects settle)
    Render images and supervision with $V^{(k)}, p_T^{(k)}, k = 1, \ldots, K^{(i)}$ and $\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(B)}\}$
```

5.4 Experiments

We conducted experiments in both simulation and the real world to answer the following questions:

1. How does COV-NeRF compare to other NeRF methods, in terms of the visual quality of its renderings?
2. How effective is synthetic data generated by COV-NeRF for training perception models relevant to robotic applications?
3. How effective is COV-NeRF at reducing the sim-to-real gap in challenging scenarios?

View Synthesis

To evaluate COV-NeRF's rendering capabilities, we compare it to the following methods:

- MVS-NeRF [9] is a state-of-the-art scene-generalizable NeRF method. It shares architectural similarities to COV-NeRF (in particular, the mechanism for feature volume construction), but is not object-centric.
- Object-NeRF [99] is an object-centric renderer like COV-NeRF, but must be retrained for each scene. Like COV-NeRF, it considers contributions from each object and the background, but each is represented by a separate MLP. It requires many sources views and mask supervision in each.

In Table 5.1, we evaluate MVS-NeRF and COV-NeRF on the COB-3D-v2 validation set (∼600 scenes). Object-NeRF is excluded due to computational constraints. In Table 5.2,

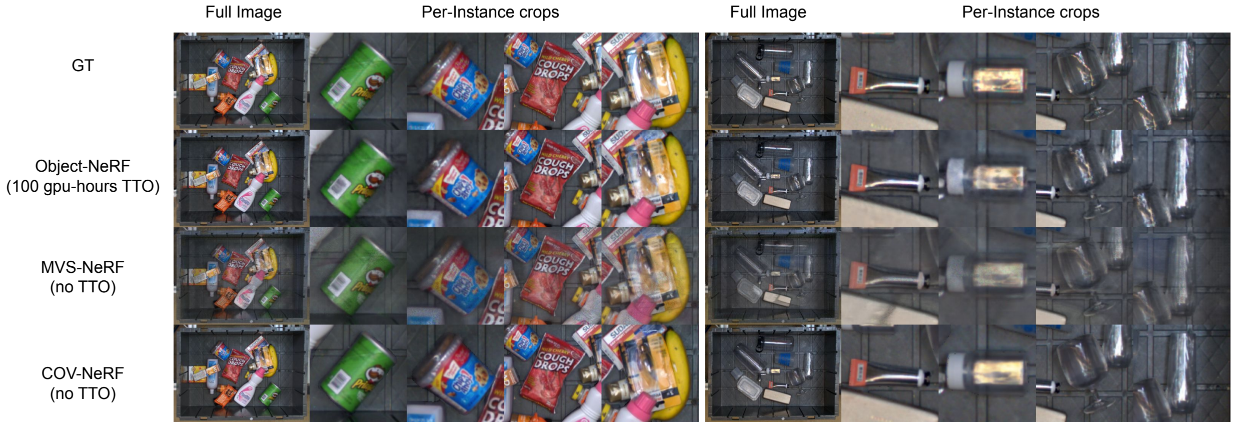


Figure 5.3: Qualitative view synthesis results on two real scenes. For each method, an image is rendered from a novel viewpoint using 4 source views (not pictured). The ground-truth image from the novel viewpoint is show in the top row. COV-NeRF matches the performance of object-centric methods that require expensive, per-scene TTO, and outperforms other scene-generalizable methods.

Table 5.1: Scene-Generalizable View Synthesis on COB-3D-v2

Method	TTO?	Object-Centric?	PSNR (↑)	SSIM (↑)	LPIPS (↓)
MVS-NeRF	N	N	25.03	0.867	0.102
COV-NeRF	N	Y	**28.32**	**0.915**	**0.082**

Table 5.2: Real-World View Synthesis

Method	TTO?	Object-Centric?	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Object-NeRF	Y	Y	23.01	0.841	0.118
MVS-NeRF	N	N	18.97	0.830	0.203
COV-NeRF	N	Y	**25.62**	**0.905**	**0.089**

we evaluate all methods on scenes from the real-world environment from Section 5.4. In Figure 5.3, we show qualitative examples from these scenes. COV-NeRF strictly outperforms MVS-NeRF, and, without TTO, matches or exceeds the performance of single-scene methods like Object-NeRF.

Sim-to-real Perception

In this section, we study COV-NeRF's effectiveness at improving visual sim-to-real transfer. Using a real-world bin picking system, we construct challenging configurations that exhibit a sim-to-real gap and evaluate COV-NeRF's effectiveness at reducing the gap. Our system consists of an ABB 1200, a 6-cup suction gripper, and 6 RGB cameras mounted over a bin. The robot must grasp objects from the bin one at a time, and transport them to an adjacent bin.

We construct a state-of-the-art bin picking system, using the following components as a representative sample of fundamental perception capabilities for robotic applications:

- Instance segmentation: we modify a SOTA method, MaskDINO [47], to predict both modal and amodal masks. We train it on COB-3D-v2, which has similar scene composition to our real environment.
- Depth estimation: we train a SOTA model for multi-view stereo, MVS-Former [6], on COB-3D-v2.
- Grasping: we use a fully-convolutional grasp-quality CNN (FC-GQCNN) [81] in the style of DexNet 3.0 [55]. It is trained in simulation to place suction cups on flat surfaces and near the object's center of mass. For each unoccluded object in the scene (as predicted by MaskDINO), we sample grasps based on a crop of the depth map predicted by MVS-Former.

We consider a few different scenarios:

- Mixed-Clutter: generic household objects are arranged chaotically inside the bin. The clutter is generally challenging for both segmentation and depth; it can be especially difficult to reason about occlusions.
- Hard-Specular: we curate a set of challenging transparent and reflective objects. These non-Lambertian surfaces cannot be accurately sensed by standard depth cameras, necessitating the use of learned methods like MVS-Former.

For each scenario, we first evaluate the baseline system. As expected, there is substantial room for improvement because it has not seen any real data, let alone these particular objects and environmental conditions. We then benchmark the following strategies for reducing the sim-to-real gap:

- COV-NeRF: We generate a synthetic dataset (following Section 5.3) and use it to finetune MaskDINO and MVS-Former. Although COV-NeRF generalizes well for view synthesis (as evidenced by Figure 5.3), we found that even tiny amounts of mask supervision vastly improve the synthetic dataset, especially the sharpness of the rendered masks. We finetune COV-NeRF using 100 real scenes encountered by the baselines system, of which 5 have mask supervision, and use the extracted objects to generate 1000 synthetic scenes.
- CyCADA [35]: This well-established domain adaptation method uses GANs to translate images between sim and real, with cycle consistency in both RGB and semantic segmentation. Deriving semantic segmentation from instance masks using three

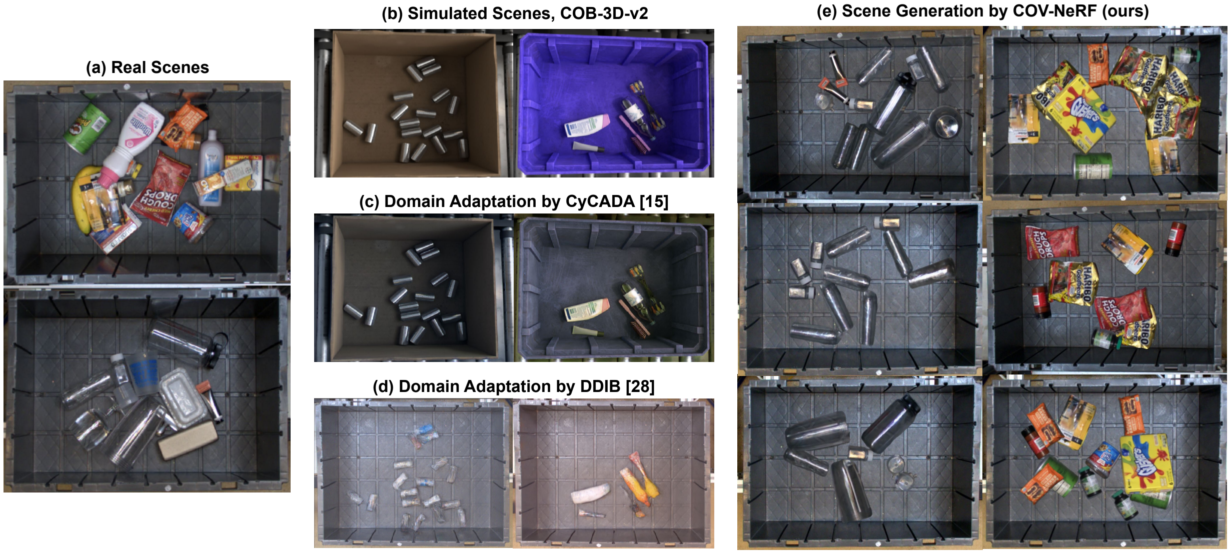

(a) Real Scenes
(b) Simulated Scenes, COB-3D-v2
(c) Domain Adaptation by CyCADA [15]
(d) Domain Adaptation by DDIB [28]
(e) Scene Generation by COV-NeRF (ours)

Figure 5.4: (a) Representative real images from the Mixed-Clutter (top) and Hard-Specular (bottom) scenarios. (b) Sample simulated scenes from COB-3D-v2. (c) CyCADA [35] adapts the scenes from (b) to more closely resemble samples from the real world, but its cycle consistency objectives result in only a mild re-styling of the sim scenes. (d) DDIB [87] produces more visually realistic adaptions of (b), but violates the original scene semantics. (e) Instead of adapting simulated scenes, COV-NeRF composes new scenes using explicit object representations extracted from (a).

classes, {*background, object, edge*}, we train CyCADA using COB-3D-v2 and 100 real scenes from each scenario. We then use the trained GANs to augment the training of MaskDINO and MVS-Former: for each simulated scene in COB-3D-v2, we use the original labels as supervision for the adapted images produced by CyCADA.

- Dual Diffusion Implicit Bridges [87]: DDIB performs image-to-image translation using diffusion models, which are the modern method of choice for image generation. Inspired by conditional diffusion models (such as StableDiffusion's img2img and depth2img modes [77]), we extend DDIB to be mask-conditioned, and train it to translate between COB-3D-v2 and our real scenarios. We then use the same procedure as for CyCADA to train MaskDINO and MVS-Former.

- Segmentation Finetuning: As a baseline, we finetune MaskDINO on real scenes using mask supervision. Note that this does not affect MVS-Former, but it allows us to evaluate whether data synthesized by COV-NeRF is visually compelling enough to reduce reliance on annotations.

In Figure 5.4, we visualize the outputs of each method, and in Table 5.3, we quantitatively evaluate grasp success and instance segmentation (mask AP for both modal and amodal predictions). Depth estimation can only be evaluated indirectly through grasp success, since

Table 5.3: Sim-to-Real Improvement

		Grasp Success Rate (↑)		Mask AP, modal / amodal (↑)	
Method	Real Supervision	Mixed-Clutter	Hard-Specular	Mixed-Clutter	Hard-Specular
Pure sim-to-real	None	0.700 ± 0.028	0.463 ± 0.035	28.5 / 27.3	13.8 / 12.4
Finetune segm	100 scenes, segm	0.742 ± 0.035	0.567 ± 0.036	59.4 / 57.3	66.5 / 65.3
CyCADA [35]	100 scenes, rgb	0.725 ± 0.035	0.453 ± 0.037	31.2 / 29.2	28.1 / 28.0
DDIB [87]	100 scenes, segm	0.727 ± 0.031	0.598 ± 0.037	36.1 / 35.1	31.7 / 31.3
COV-NeRF (ours)	100 scenes (5 segm / 95 rgb-only)	**0.929 ± 0.021**	**0.807 ± 0.022**	**62.7 / 61.8**	**72.9 / 72.4**

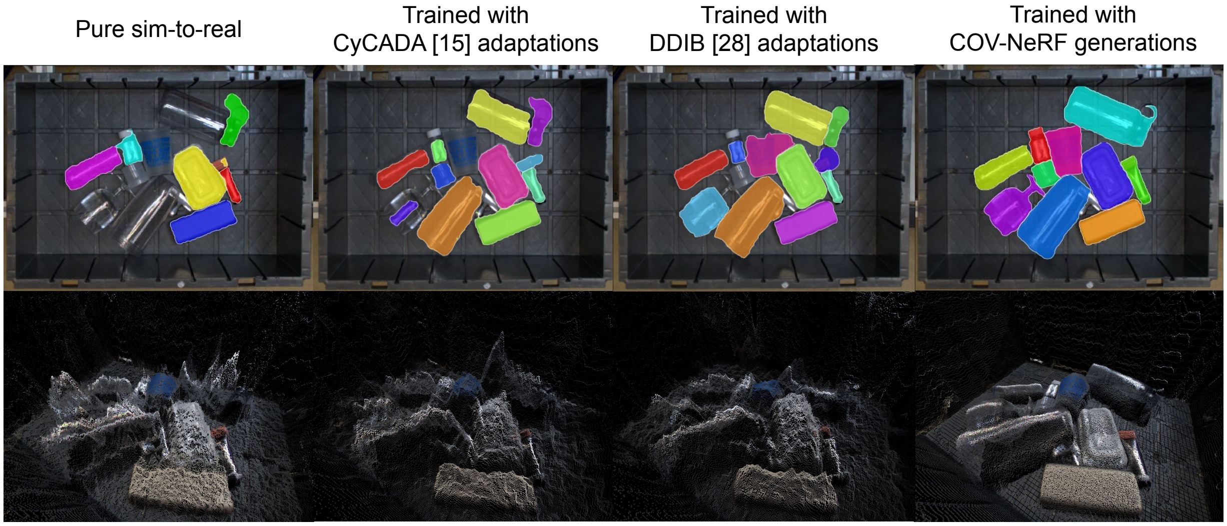

Pure sim-to-real
Trained with CyCADA [15] adaptations
Trained with DDIB [28] adaptations
Trained with COV-NeRF generations

Figure 5.5: Sample instance segmentation predictions from MaskDINO (top row) and stereo depth predictions from MVS-Former (bottom row) resulting from the sim-to-real methods evaluated in Table 5.3. COV-NeRF enables substantial improvement in both modalities.

we cannot obtain ground-truth in the real world (especially for Hard-Specular), but we study qualitative examples in Figure 5.5. With as little as 5 scenes of mask supervision, COV-NeRF enables substantial improvement in both grasp success and mask AP. CyCADA has almost no effect on the real system: the cycle consistency objective and the challenges of GAN optimization prevent it from substantially altering the simulated images. DDIB's adaptations have more realistic low-level textural details, but are still vastly less realistic than the scenes synthesized by COV-NeRF.

5.5 Discussion

We presented COV-NeRF, an object-composable and scene-generalizable neural renderer that can the close sim-to-real gap for a variety of perception modalities, including ones where direct supervision cannot be obtained in the real world. After validating COV-NeRF against existing NeRF methods, we explored its effectiveness at combating sim-to-real mismatch in challenging bin-picking scenarios. We showed that COV-NeRF can generate targeted synthetic data that is effective at improving state-of-the-art perception methods, translating to significant improvements in end-to-end application performance.

Despite its demonstrated results, COV-NeRF's rendering model suffers from a few limitations. It does not account for higher-order visual effects like reflections or variations in scene lighting, and it must imagine the appearance of the occluded portions of objects. We hope to address this in future work, as it would further enhance the diversity and realism of the scenes that COV-NeRF can generate.

Chapter 6

Conclusion

In this thesis, we have highlighted and addressed several challenges that commonly arise in real-world robotic applications. We investigated the inability of existing model families to express and reason about real-world ambiguity, and proposed novel architectures to overcome this challenge for instance segmentation and 3D object objection. We showed that our improved models learn to express the underlying uncertainty in ambiguous situations, and that their rich distributional output can be leveraged by downstream components for more flexible and robust decision making. Despite the improved performance and expanded capabilities of these methods, the difficulty of obtaining real-world supervision is still a considerable bottleneck to robotic applications, and results in a relative scarcity of relevant training data compared to other domains. We developed a neural rendering method that tackles this challenge by improving sim-to-real transfer, allowing more effective use of simulators to provide supervision. We showed that our object-centric NeRF renderer can learn from real scenes to generate synthetic training data that is targeted to real objects and environments, which can then substantially improve real-world performance for many perception modalities and for the overall application.

Our contributions help pave the way for robust and intelligent robotic systems that perform useful work in the real world. However, there is much to be done before such systems are ubiquitous, and the work presented in this thesis highlights several exciting directions for future work.

- **Natural Language and Uncertainty:** Chapters 2 and 3 presented techniques for endowing perception models with the ability to express and reason about uncertainty in ambiguous situations. Our models describe the hypothesis space by sampling, but natural language could be a more powerful and interpretable mechanism for doing so, similarly to how it has been used for task specification [4, 5]. For example, grounding Latent-MaskRCNN's latent space (Chapter 2) with language could allow the model to more compactly describe its uncertainty. This idea could also be applied to open-vocabulary detection [41, 106] where the distributional uncertainty could be extended beyond simple categorization to detailed characteristics or long-tail events.

- **Structured Intuitive Physics:** The COV-NeRF model presented in Chapter 5 combines the expressiveness of neural representations with strong inductive biases (object-centricity, pose equivariance) that enable effective interpretability, composability, and generalization. Extending this formulation to incorporate physics could result in a particularly compelling learnable simulator or world model [26]. If COV-NeRF's representations could be learned from video instead of static captures, there should be sufficient information to also learn the dynamical properties of objects in addition to their appearance. Physics is similar to rendering in that the high-level structure is well understood (e.g. Newtonian mechanics and raycasting) but the details may be complex (e.g. a particular surface's contact force model or material properties), making it a good fit for hybrid approaches like COV-NeRF. This line of work could result in a powerful fusion between system identification (where we learn a few parameters of a simple and often explicitly-specified model, but the model generalizes well within its constraints) and intuitive physics or video prediction (where no structure is imposed, but certain types of generalization may be poor even in abundant data regimes).

* * *

We hope that this work serves as a stepping stone towards fully realizing the potential of both AI and robotics. The future is bright and we are excited to see where it leads.

Bibliography

[1] Anurag Arnab and Philip H. S. Torr. "Pixelwise Instance Segmentation With a Dynamically Instantiated Network". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[2] Min Bai and Raquel Urtasun. "Deep watershed transform for instance segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[3] Daniel Bolya et al. "YOLACT: Real-time Instance Segmentation". In: *International Conference on Computer Vision (ICCV)*. 2019.

[4] Anthony Brohan et al. "RT-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[5] Anthony Brohan et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).

[6] Chenjie Cao, Xinlin Ren, and Yanwei Fu. "MVSFormer: Multi-View Stereo by Learning Robust Image Features and Temperature-based Depth". In: *Transactions on Machine Learning Research (TMLR)*. 2022.

[7] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *European Conference on Computer Vision (ECCV)*. 2020.

[8] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012*. 2015.

[9] Anpei Chen et al. "MVS-NeRF: Fast generalizable radiance field reconstruction from multi-view stereo". In: *International Conference on Computer Vision (ICCV)*. 2021.

[10] Xinlei Chen et al. "Tensormask: A foundation for dense object segmentation". In: *International Conference on Computer Vision (ICCV)*. 2019.

[11] Zoey Chen et al. "Genaug: Retargeting behaviors to unseen situations via generative augmentation". In: 2023.

[12] Bowen Cheng et al. "Masked-attention Mask Transformer for Universal Image Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[13] Jiwoong Choi et al. "Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving". In: *International Conference on Computer Vision (ICCV)*. 2019.

[14] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[15] Angela Dai et al. "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[16] Qiyu Dai et al. "Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[17] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[18] Markus Freitag and Yaser Al-Onaizan. "Beam Search Strategies for Neural Machine Translation". In: *Proceedings of the First Workshop on Neural Machine Translation*. 2017.

[19] Bin-Bin Gao et al. "Deep label distribution learning with label ambiguity". In: *IEEE Transactions on Image Processing*. 2017.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[21] Golnaz Ghiasi et al. "Simple copy-paste is a strong data augmentation method for instance segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[22] Igor Gilitschenski et al. "Deep Orientation Uncertainty Learning based on a Bingham Loss". In: *International Conference on Learning Representations (ICLR)*. 2020.

[23] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. "Mesh R-CNN". In: *International Conference on Computer Vision (ICCV)*. 2019.

[24] Marcus Gualtieri and Robert Platt. "Robotic pick-and-place with uncertain object instance segmentation and shape completion". In: *IEEE Robotics and Automation Letters (RA-L)*. 2021.

[25] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. "Multiple Choice Learning: Learning to Produce Multiple Structured Outputs." In: *Neural Information Processing Systems (NeurIPS)*. 2012.

[26] David Ha and Jürgen Schmidhuber. "World models". In: *arXiv preprint 1803.10122* (2018).

[27] David Hall et al. "Probabilistic Object Detection: Definition and Evaluation". In: *Winter Conference on Applications of Computer Vision (WACV)*. 2020.

[28] Ali Harakeh, Michael Smart, and Steven L. Waslander. "BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors". In: *International Conference on Robotics and Automation (ICRA)*. 2020.

[29] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[30] Kaiming He et al. "Mask R-CNN". In: *International Conference on Computer Vision (ICCV)*. 2017.

[31] Yihui He et al. "Bounding box regression with uncertainty for accurate object detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[32] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations (ICLR)*. 2017.

[33] Daniel Ho et al. "RetinaGAN: An object-aware approach to sim-to-real transfer". In: *International Conference on Robotics and Automation (ICRA)*. 2021.

[34] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Neural Information Processing Systems (NeurIPS)*. 2020.

[35] Judy Hoffman et al. "CyCADA: Cycle-consistent adversarial domain adaptation". In: *International Conference on Machine Learning (ICML)*. 2018.

[36] Jeffrey Ichnowski et al. "Dex-NeRF: Using a neural radiance field to grasp transparent objects". In: *Conference on Robot Learning (CoRL)*. 2021.

[37] Won-Dong Jang et al. "Learning Vector Quantized Shape Code for Amodal Blastomere Instance Segmentation". In: 2020.

[38] Jie Jia, Huiliang Shang, and Xiong Chen. "Robot Online 3D Bin Packing Strategy Based on Deep Reinforcement Learning and 3D Vision". In: *International Conference on Networking, Sensing and Control (ICNSC)*. 2022.

[39] Justin Kerr et al. "Evo-NeRF: Evolving nerf for sequential robot grasping of transparent objects". In: *Conference on Robot Learning (CoRL)*. 2022.

[40] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations (ICLR)*. 2014.

[41] Alexander Kirillov et al. "Segment anything". In: *International Conference on Computer Vision (ICCV)*. 2023.

[42] Simon A. A. Kohl et al. "A Probabilistic U-Net for Segmentation of Ambiguous Images". In: *Neural Information Processing Systems (NeurIPS)*. 2018.

[43] James J. Kuffner. “Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning”. In: *International Conference on Robotics and Automation (ICRA)*. 2004.

[44] Abhijit Kundu et al. “Panoptic neural fields: A semantic object-aware neural scene representation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[45] Olyvia Kundu, Samrat Dutta, and Swagat Kumar. “Deep-pack: A vision-based 2d online bin packing algorithm with deep reinforcement learning”. In: *International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2019.

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

[47] Feng Li et al. “Mask DINO: Towards a unified transformer-based framework for object detection and segmentation.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[48] Xiang Li et al. “Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection”. In: *Neural Information Processing Systems (NeurIPS)*. 2020.

[49] Chung-Ching Lin et al. “Video Instance Segmentation Tracking With a Modified VAE Architecture”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[50] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[51] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European Conference on Computer Vision (ECCV)*. 2014.

[52] Yuan Liu et al. “Neural rays for occlusion-aware image-based rendering”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[53] Ze Liu et al. “Group-free 3d object detection via transformers”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[54] William E Lorensen and Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *SIGGRAPH*. 1998.

[55] Jeffrey Mahler et al. “Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning”. In: *International Conference on Robotics and Automation (ICRA)*. 2018.

[56] Zhao Mandi et al. “Cacti: A framework for scalable multi-task multi-scene visual imitation learning”. In: *arXiv preprint arXiv:2212.05711*. 2022.

[57] Lars Mescheder et al. “Occupancy networks: Learning 3d reconstruction in function space”. In: 2019.

[58] Luke Metz et al. “Discrete sequential prediction of continuous actions for deep rl”. In: *arXiv preprint arXiv:1705.05035*. 2017.

[59] Gregory P. Meyer and Niranjan Thakurdesai. “Learning an Uncertainty-Aware Object Detector for Autonomous Driving”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[60] Gregory P. Meyer et al. “LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2019.

[61] Ben Mildenhall et al. “NeRF: Representing scenes as neural radiance fields for view synthesis”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[62] Dimity Miller et al. “Benchmarking Sampling-based Probabilistic Object Detectors”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2019.

[63] Ishan Misra, Rohit Girdhar, and Armand Joulin. “An End-to-End Transformer Model for 3D Object Detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[64] Arsalan Mousavian et al. “3D Bounding Box Estimation using Deep Learning and Geometry”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[65] Davy Neven et al. “Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[66] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499*. 2016.

[67] Jeong Joon Park et al. “Deepsdf: Learning continuous signed distance functions for shape representation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[68] Valentin Peretroukhin et al. “A Smooth Representation of SO(3) for Deep Rotation Learning with Uncertainty”. In: *Robotics: Science and Systems (RSS)*. 2020.

[69] Charles R Qi et al. “Frustum PointNets for 3D Object Detection from RGB-D Data”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[70] Charles R Qi et al. “Imvotenet: Boosting 3d object detection in point clouds with image votes”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[71] Charles R Qi et al. “PointNet: Deep learning on point sets for 3d classification and segmentation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[72] Charles R. Qi et al. "Deep Hough Voting for 3D Object Detection in Point Clouds." In: *International Conference on Computer Vision (ICCV)*. 2019.

[73] Kanishka Rao et al. "RL-CycleGAN: Reinforcement learning aware simulation-to-real". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[74] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[75] Jeremy Reizenstein et al. "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction". In: *International Conference on Computer Vision (ICCV)*. 2021.

[76] Shaoqing Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Neural Information Processing Systems (NeurIPS)*. 2014.

[77] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[78] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. "FCAF3D: Fully Convolutional Anchor-Free 3D Object Detection". In: *European Conference on Computer Vision (ECCV)*. 2022.

[79] Lorenz Rumberger, Lisa Mais, and Dagmar Kainmueller. "Probabilistic Deep Learning for Instance Segmentation". In: *ECCV BioImage Computing Workshop*. 2020.

[80] Christian Rupprecht et al. "Learning in an uncertain world: Representing ambiguity through multiple hypotheses". In: *International Conference on Computer Vision (ICCV)*. 2017.

[81] Vishal Satish, Jeffrey Mahler, and Ken Goldberg. "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks". In: *IEEE Robotics and Automation Letters (RA-L)* (2019).

[82] S Shi, X Wang, H PointRCNN Li, et al. "3d object proposal generation and detection from point cloud". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[83] Shaoshuai Shi et al. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[84] Yawar Siddiqui et al. "Panoptic lifting for 3d scene understanding with neural fields". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[85] Maximilian Sieb and Katerina Fragkiadaki. "Data dreaming for object detection: Learning object-centric state representations for visual imitation". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2018.

[86] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[87] Xuan Su et al. “Dual Diffusion Implicit Bridges for Image-to-Image Translation”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[88] Zhi Tian et al. “FCOS: Fully convolutional one-stage object detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[89] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[90] Emanuel Todorov, Tom Erez, and Yuval Tassa. “Mujoco: A physics engine for model-based control”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[91] Fabio Tosi et al. “Nerf-supervised deep stereo”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[92] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel recurrent neural networks”. In: *International Conference on Machine Learning (ICML)*. 2016.

[93] Jacob Varley et al. “Shape completion enabled robotic grasping”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[94] Fan Wang and Kris Hauser. “Dense robotic packing of irregular and novel 3D objects”. In: *IEEE Transactions on Robotics*. 2021.

[95] Fan Wang and Kris Hauser. “Stable bin packing of non-convex 3D objects with a robot manipulator”. In: *International Conference on Robotics and Automation (ICRA)*. 2019.

[96] Lei Wang et al. “Two natural heuristics for 3D packing with practical loading constraints”. In: *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*. 2011.

[97] Enze Xie et al. “PolarMask: Single Shot Instance Segmentation With Polar Representation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[98] Qian Xie et al. “MLCVNet: Multi-Level Context VoteNet for 3D Object Detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[99] Bangbang Yang et al. “Learning object-compositional neural radiance field for editable scene rendering”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[100] Shuo Yang et al. “Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing”. In: *IEEE Transactions on Automation Science and Engineering*. 2023.

[101] Alex Yu et al. “pixelnerf: Neural radiance fields from one or few images”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[102] Hong-Xing Yu et al. “Learning object-centric neural scattering functions for free-viewpoint relighting and scene composition”. In: 2023.

[103] Tianhe Yu et al. “Scaling robot learning with semantically imagined experience”. In: *arXiv preprint arXiv:2302.11550*. 2023.

[104] Xumin Yu et al. “PoinTr: Diverse point cloud completion with geometry-aware transformers”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[105] Wentao Yuan et al. “Pcn: Point completion network”. In: *International Conference on 3D Vision (3DV)*. 2018.

[106] Hao Zhang et al. “A simple framework for open-vocabulary segmentation and detection”. In: *International Conference on Computer Vision (ICCV)*. 2023.

[107] Hao Zhang et al. “DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[108] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

Appendix A

COB-3D Dataset Overview

COB-3D is a simulated dataset for a variety of 2D and 3D perception tasks that we developed throughout the work presented in Chapters 3 and 4. We designed it to be representative of the challenges encountered in real-world robotic applications, while still exhibiting state-of-the-art diversity and realism. The most recent version of the dataset is COB-3D-v2, and it can be downloaded from our project page. We hope that it will be of benefit to the broader robotics and vision communities.

Dataset Format

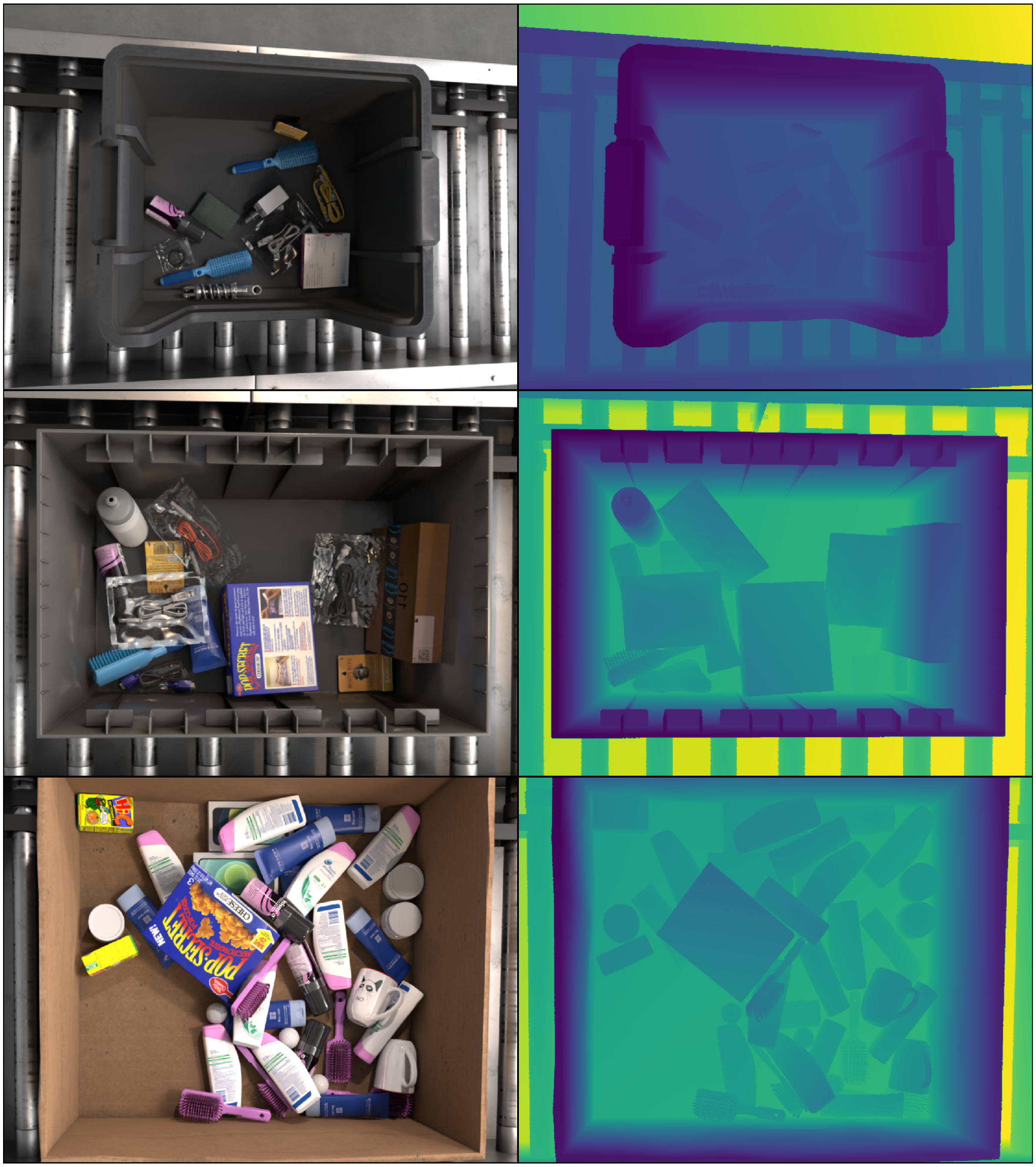
COB-3D-v2 contains 6955 scenes, split into 6259 train, 696 validation. For each scene, we provide the following:

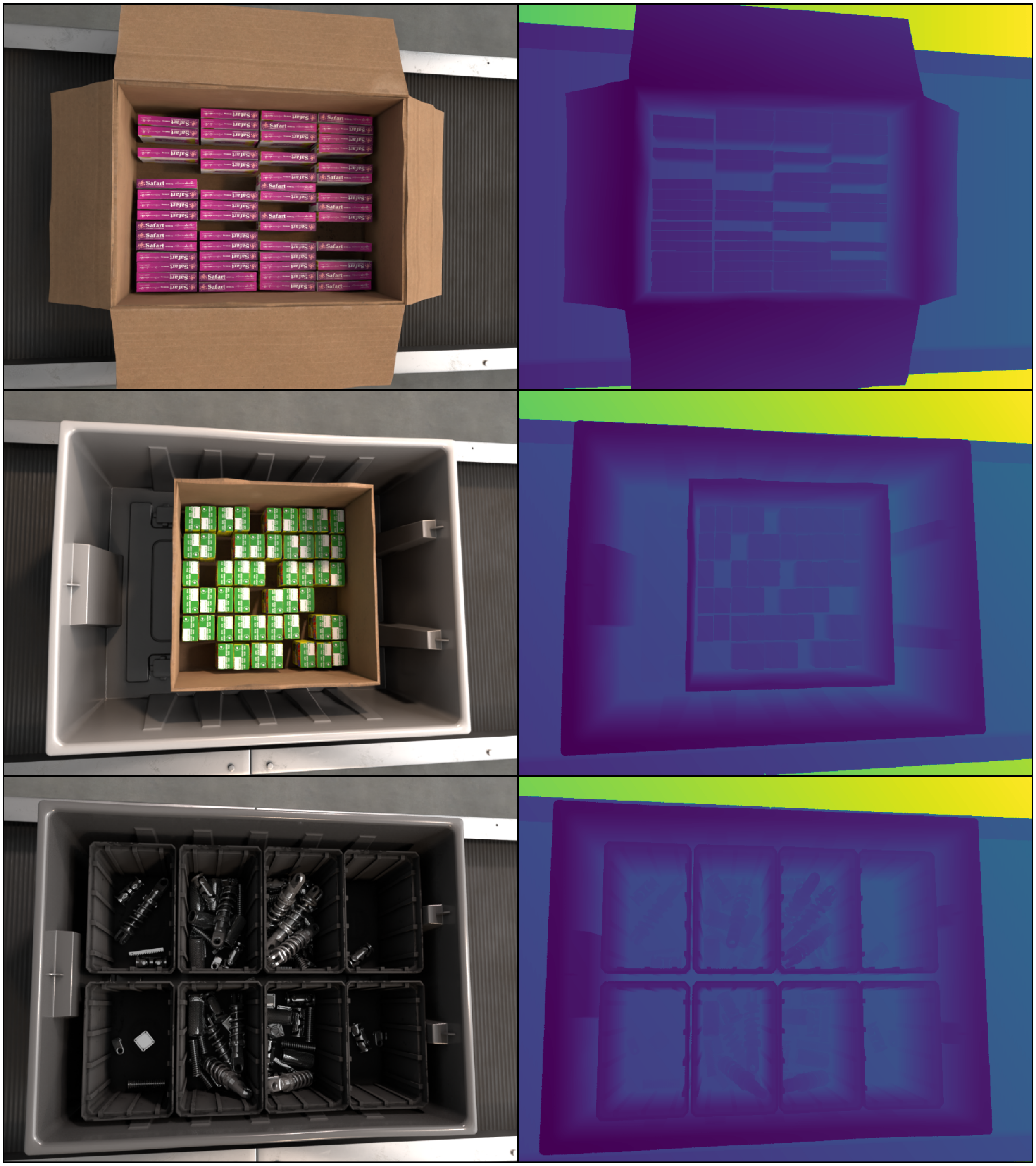
```
|-- rgb:                The rendered RGB image.
                        Shape (3, H, W), dtype float32. Values scaled to [0, 1].

|-- intrinsic:          The camera intrinsics.
                        Shape (3, 3), dtype float32.

|-- depth_map:          The rendered depth map corresponding to ‘rgb‘.
                        Shape (H, W), dtype float32.

|-- normal_map:         The rendered normal map corresponding to rgb‘.
                        Shape (3, H, W), dtype float32.

|-- near_plane:         The minimum depth value of the scene’s working volume.
                        Scalar, float32.

|-- far_plane:          The maximum depth value of the scene’s working volume.
                        Scalar, float32.
```

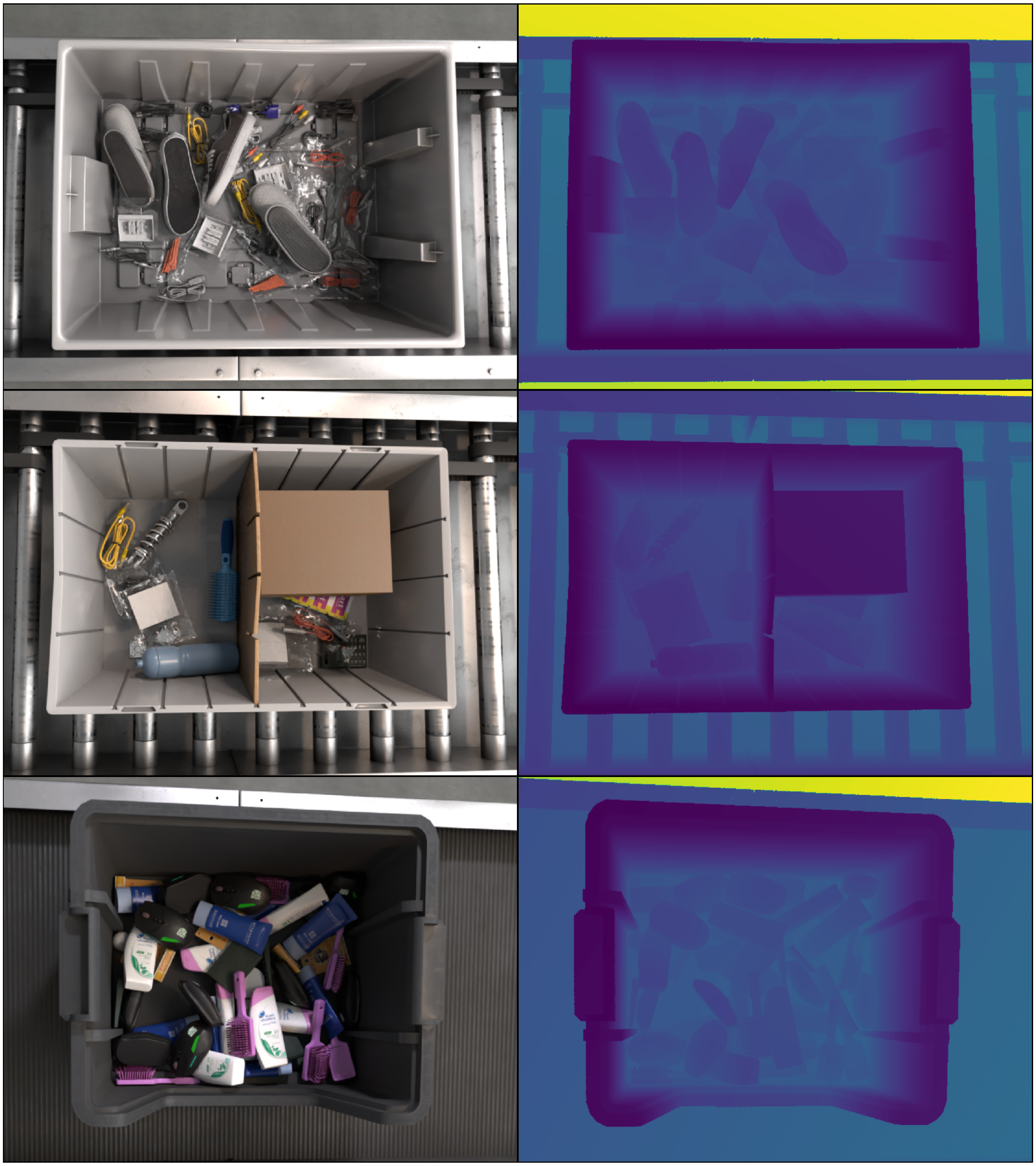
```
|-- segm/
    |-- boxes:          2D bounding boxes for each object in the scene.
                        Shape (N_objects, 4), dtype float32.
                        These are pixel coordinates relative to ‘rgb‘.
                        The box format is ‘[x_low, y_low, x_high, y_high]‘.

    |-- masks:          Binary masks for each object in the scene.
                        Shape (N_objects, H, W), dtype bool.

    |-- amodal_masks:   Amodal instance masks for each object.
                        Shape (N_objects, H, W), dtype bool.

|-- bbox3d/
    |-- poses:          The pose (4x4 matrix) of each object’s 3D bounding box.
                        It transforms from the bbox frame to the camera frame.
                        Shape (N_objects, 4, 4), dtype float32.

    |-- dimensions:     The dimensions of each 3D bounding box.
                        Shape (N_objects, 3), dtype float32.

    |-- corners:        The corner points of each 3D bounding box.
                        The points are specified in the camera frame.
                        Shape (N_objects, 8, 3), dtype float32.

|-- mesh_ids:           The mesh_id of each object. List[str], length N_objects.

|-- obj_poses/
    |-- poses:          The pose (4x4 matrix) of each mesh.
                        It transforms from the mesh frame to the camera frame.
                        Note that the mesh frame may differ from the bbox frame!
                        Shape (N_objects, 4, 4), dtype float32.

    |-- scales:         The scale of each mesh.
                        Shape (N_objects, 3), dtype float32.

|-- voxel_grid/
   |-- voxels:          The surface of each mesh, extracted into a voxel grid.
                        Shape (N_objects, n_voxels, n_voxels, n_voxels),
                        dtype bool.

   |-- extents:         The extents of each object’s voxel grid.
                        ‘voxels[i]‘ span the cuboid ‘[-extents[i], extents[i]]‘,
                        in the object frame ‘obj_poses/poses[i]‘.
                        Shape (N_objects, 3), dtype float32.
```

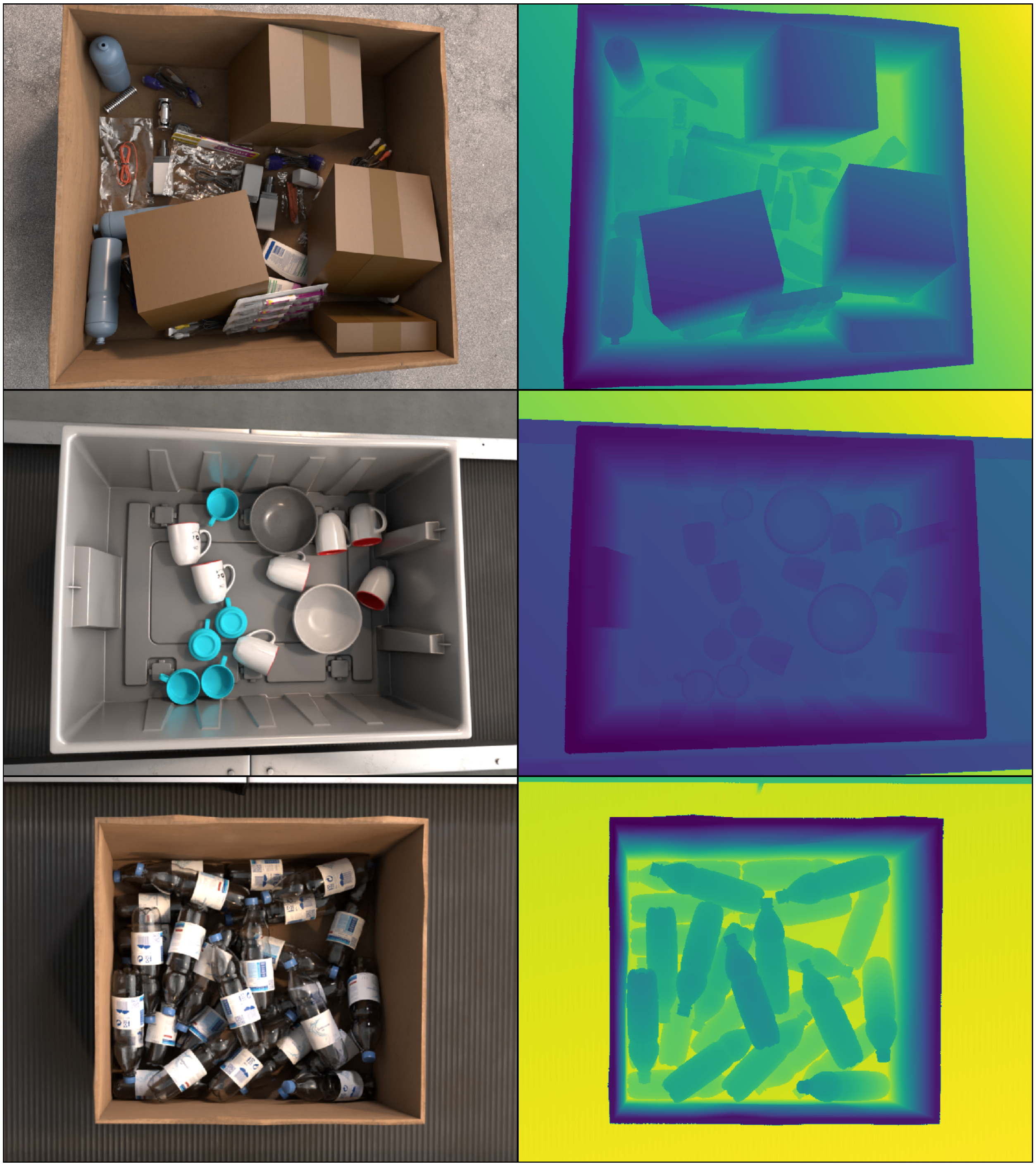
Example Scenes

The following pages exhibit some representative scenes from COB-3D-v2, showcasing the visual quality and diversity of the dataset. In each row, the left column is the rendered RGB, and the right column is the rendered depth map.









Appendix B

Proof of Quantile-Confidence Box Equivalence

In this section, we present the proof of Theorem 1 from Section 3.4:

Theorem 1

A quantile box with quantile q is a confidence box with confidence $p = 1 - q$ when $p(b|h)$ is an ordered object distribution.

Proof Sketch:

Let $P(b)$ be a distribution over an ordered set of boxes where for any two distinct boxes b_1, b_2 in the sample space, one must be contained in the other, $b_1 \subset b_2$ or $b_2 \subset b_1$. We'll show that a quantile box b_q is a confidence box with $p = 1 - q$ by 1) constructing a confidence box b_p for any given q, 2) showing that any $x \in b_p$ must have $O(x) > q$, and 3) therefore $b_p \subseteq Q(q) \subseteq b_q$ so the quantile box is a confidence box.

1) Confidence Box:

For any $p = 1 - q$, we'll show how to construct a confidence box b_p. Using the ordered object distribution property of $P(b)$, we can define ordering as containment $b_1 < b_2 \equiv b_1 \subset b_2$. This ordering defines an inverse cdf:

$$F^{-1}(p) = \inf\{x : P(b \leq x) \geq p\} \tag{B.1}$$

Let $b_p = F^{-1}(1-q)$ be the inverse cdf of p; by definition b_p is a confidence box with confidence p since $P(b \leq b_p) = P(b \subseteq b_p) \geq p$

2) Occupancy of b_p:

We'll show that any $x \in b_p$ satisfies $O(x) > 1-p$. First we'll prove that that $P(b \geq b_p) > 1-p$. Let $b_0 = \inf\{b : b < b_p\}$, the smallest box that is strictly contained in b_p. (If no such b_0 exists, then b_p must be the smallest box in the distribution order such that $P(b \geq b_p) = 1$ and $P(b \geq b_p) > 1-p$ for $p \neq 0$)

Since b_p is the inverse cdf of p, we know that $P(b \leq b_0) < p$, otherwise b_0 would be the inverse cdf of p (i.e. $b_0 = b_p$ a contradiction). It follows that

$$\begin{aligned} P(b \geq b_p) &= P(b > b_0) && \text{(B.2)} \\ &= 1 - P(b \leq b_0) && \text{(B.3)} \\ &> 1 - p && \text{(B.4)} \end{aligned}$$

Now consider any point $x \in b_p$:

$$\begin{aligned} O(x) &= P(x \in b) && \text{(B.5)} \\ &= \int_b \mathbb{1}\{x \in b\} p(b) db && \text{(B.6)} \\ &\geq \int_{b \geq b_p} \mathbb{1}\{x \in b\} p(b) db && \text{(B.7)} \\ &= \int_{b \geq b_p} p(b) db && \text{(B.8)} \\ &= P(b \geq b_p) && \text{(B.9)} \\ &> 1 - p && \text{(B.10)} \end{aligned}$$

Where (B.7) follows from the nonegativity of $\mathbb{1}\{x \in b\}p(b)$. (B.8) follows from $x \in b_p$, $b_p \subseteq b$ which implies $x \in b$.

3) Quantile-Confidence Box:

Since any $x \in b_p$ satisfies $O(x) > 1-p$, it follows that $b_p \subseteq Q(1-p)$, where $Q(q) = \{x : O(x) > q\}$ is the occupancy quantile with quantile q. The quantile box by construction must contain the occupancy quantile $Q(q) \subseteq b_q$, therefore we have $b_p \subseteq Q(1-p) \subseteq b_q$, and

$$\begin{aligned} P(b \subseteq b_q) &\geq P(b \subseteq b_p) && \text{(B.11)} \\ &\geq p && \text{(B.12)} \end{aligned}$$

So b_q is a confidence box with confidence requirement p.