Object-Centric Generative Models for Robot Perception and Action

Yizhe Wu
Kellogg College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity 2023

Object-Centric Generative Models for Robot Perception and Action
Candidate: Yizhe Wu
Supervisor: Prof. Ingmar Posner
Examiners: Dr. Christian Rupprecht, Prof. Sungjin Ahn
Date of examination: 21st November, 2023
Date of revision: 8th December, 2023

University of Oxford
Applied Artificial Intelligence Lab (A2I)
Oxford Robotics Institute (ORI)
Department of Engineering Science

Statement of Authorship

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Yizhe Wu
Kellogg College
September 2023

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Ingmar Posner. During my whole study in Oxford, he has been a paternal Batman supporting me with patience and power. I learned how to become an independent researcher step by step from him and I think now I'm ready to contribute on my own to the community. I also want to thank my second "semi-supervisor" Dr. Oiwi Parker Jones for his companionship and constructive feedback to all my works. He is the nicest man I have ever met in my whole life.

I am also very grateful for my collaborators and labmates at the Oxford Robotics Institute. Especially, I would like to thank Martin Engelcke, Oliver Groth, Sudhanshu Kasewa, Chia-Man Hung, Kevin Li Sun, Sasha Salter, Jack Collins, Walter Goodwin, Alexander Mitchell, Rob Weston and Fabian Fuchs. I learned a lot from working together with Martin and Oliver on how to make sure everything is one hundred-percent correct and reproducible, which will remain very important for conducting my future research.

I would like to also express my gratitude to my friends and collegemates in Oxford and at Kellogg college, you contributed to the remaining part of my happy life here.

To my parents, I know our family doesn't like to express love explicitly in words, but your unfailing support of me throughout the past four and half years has not gone unappreciated. Thank you.

Lastly, I would like to thank my wife, Jiaxin Yun(kuer), Although we have been separated more than we would like for eight years as I pursue my Master and PhD degrees abroad, finally, we can start our new life living together forever now.

Abstract

The system of robot manipulation involves a pipeline consisting of the perception of objects in the environment and the planning of actions in 3D space. Deep learning approaches are employed to segment scenes into components of objects and then learn object-centric features to predict actions for downstream tasks. Despite having achieved promising performance in several manipulation tasks, supervised approaches lack inductive biases related to general properties of objects. Recent advances show that by encoding and reconstructing scenes in an object-centric fashion, the model can discover object-like entities from raw data without human supervision. Moreover, by reconstructing the discovered objects, the model can learn a variational latent space that captures the various shapes and textures of the objects, regularised by a chosen prior distribution. In this thesis, we investigate the properties of this learned object-centric latent space and develop novel object-centric generative models (OCGMs) that can be applied to real-world robotics scenarios.

In the first part of this thesis, we investigate a tool-synthesis task which leverages a learned latent space to optimise a wide range of tools applied to a reaching task. Given an image that illustrates the obstacles and the reaching target in the scene, an affordance predictor is trained to predict the feasibility of the tool for the given task. To imitate human tool-use experiences, feasibility labels are acquired from simulated trial-and-errors of the reaching task. We found that by employing an activation maximisation step, the model can synthesis proper tools for the given tasks with high accuracy. Moreover, the tool-synthesis process indicates the existence of a task-relevant trajectory in the learned latent space that can be found by a trained affordance predictor.

The second part of this thesis focuses on the development of novel OCGMs and their applications to robotic tasks. We first introduce a 2D OCGM that is deployed to robot manipulation datasets in both simulation and real-world scenarios. Despite the intensive interactions between robot arm and objects, we find the model discovers meaningful object entities from the raw observations without any human supervision. We next upgrade the 2D OCGM to 3D by leveraging NeRFs as decoders to explicitly model the 3D geometry of objects and the background. To disentangle the object spatial information from its appearance information, we propose a minimum volume principle for unsupervised 6D pose estimation of the

objects. Considering the occlusion in the scene, we further improve the pose estimation by introducing a shape completion module to imagine the unobserved parts of the objects before the pose estimation step. In the end, we successfully apply the model in real-world robotics scenarios and compare its performance in several tasks including the 3D reconstruction, object-centric latent representation learning, 6D pose estimation for object rearrangement, against several baselines. We find that despite being an unsupervised approach, our model achieves improved performance across a range of different real-world tasks.

Contents

List of Abbreviations **xv**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Guiding Questions . . . 4
1.3 Thesis Outline . . . 5
1.3.1 Tool Synthesis Using Affordance Predictor Learned from Tool-use Experiences . . . 7
1.3.2 Unsupervised 2D OCGM for Robotic Scenes . . . 8
1.3.3 Unsupervised OCGM with Pose Estimation in 3D . . . 10
1.3.4 3D OCGMs in Real-world Robotic Applications . . . 11
1.4 Publications and Contributions . . . 12

2 Background **15**
2.1 Latent Variable Models . . . 15
2.2 Object-centric Generative Models . . . 17
2.2.1 Scene Encoder . . . 17
2.2.2 Attentions in the OCGMs . . . 18
2.2.3 Component VAE . . . 22
2.3 Related works . . . 23
2.3.1 Affordance Learning in Robotic Tasks . . . 23
2.3.2 Object-centric Generative Models . . . 24

3 Learning Affordances in Object-Centric Generative Models **27**
3.1 Introduction . . . 29
3.2 Related Work . . . 31
3.3 Method . . . 33
3.3.1 Representing Tasks and Tools . . . 34
3.3.2 Tool Imagination . . . 35
3.4 Experiments . . . 37
3.4.1 Model Training . . . 38
3.4.2 Quantitative Results . . . 38

3.4.3 Qualitative Results . . . 40
3.5 Conclusion . . . 42
3.6 Dataset for the Reaching Tasks . . . 43
3.7 Dataset Construction Minutiae . . . 43
3.8 Architecture and Training Details . . . 44

4 APEX: Unsupervised, Object-Centric Scene Segmentation and Tracking for Robot Manipulation 47
4.1 Introduction . . . 49
4.2 Related Work . . . 51
4.3 APEX: Amortised Parallel Inference with Mixture Models . . . 53
4.3.1 Generative Model . . . 54
4.3.2 Inference Model . . . 56
4.3.3 Inference - Implementation . . . 57
4.3.4 Learning . . . 58
4.4 Experiments . . . 60
4.4.1 Unsupervised Segmentation and Tracking . . . 62
4.4.2 Ablation Study . . . 64
4.4.3 Object Arrangement Task . . . 65
4.5 Conclusions . . . 68

5 ObPose: Leveraging Pose for Object-Centric Scene Inference and Generation in 3D 69
5.1 Introduction . . . 71
5.2 Related Work . . . 74
5.3 Methods . . . 75
5.3.1 Encoder . . . 76
5.3.2 Decoder . . . 79
5.4 Training . . . 80
5.5 Experiments . . . 82
5.5.1 Metrics . . . 82
5.5.2 Unsupervised 3D Object Segmentation . . . 83
5.5.3 Scene Generation and Scene Editing . . . 84
5.6 Conclusion . . . 86

6 DreamUp3D: Object-Centric Generative Models for Single-View 3D Scene Understanding and Real-to-Sim Transfer 87
6.1 Introduction . . . 89
6.2 Related Work . . . 92
6.3 DreamUp3D . . . 94

6.3.1 Data Pre-processing 94
6.3.2 Scene Segmentation 95
6.3.3 Pose Estimation with Shape Completion 95
6.3.4 Scene Reconstruction 97
6.3.5 Training 97
6.4 Experiments 100
6.4.1 Experimental Setup 100
6.4.2 Scene Reconstruction 100
6.4.3 Object-centric Representation 103
6.4.4 Unsupervised Pose Estimation 106
6.4.5 Limitations 108
6.5 Conclusion 110

7 Discussion **111**
7.1 Key Contributions 111
7.2 Limitations 114
7.3 Future Work 117
7.4 Closing Remarks 118

References **121**

List of Abbreviations

2D . . . . . two-dimensional

3D . . . . . three-dimensional

OCGMs . . . . Object-centric generative models

NeRFs . . . . . Nerual radiance fields

VAE . . . . . Variational auto-encoder

CNN . . . . . Convolutional neural network

RPN . . . . . Region proposal network

MPC . . . . . Model predictive control

ELBO . . . . . Evidence lower bound

KL . . . . . Kullback-Leibler

GRAF . . . . . Generative radiance field

IC-SBP . . . . Instance colouring stick-breaking process

MLP . . . . . Multi-layer perceptron

Conv-LSTM . Convolutional long short term memory

1 Introduction

1.1 Motivation

Manipulating the objects in a scene using an actuated robot arm necessitates the coordination of perception and action systems. The perception system aims to interpret the raw observations, such as pixels or point clouds, into meaningful representations, which include explicit representations, such as the object poses [1, 2], the object bounding boxes [3], the object meshes for geometry-based manipulation [4, 5], or latent representations that encode object properties such as the object affordance into latent embedding [6]. Taking these representations, a high level planner can then bridge the gap between the user interface and the low-level robot controller and manipulate the selected objects in a task-oriented way. The perception system thus plays a vital role for the scene understanding that facilitates the object manipulation. With the advance of deep learning, learning-based approaches offer advantages such as real-time inference or generalisation ability on unseen instances. The investigation of these approaches thus delves into the feature learning [7, 8] and collecting larger dataset [9] for improved generalisation ability, which despite a straightforward solution, lacks the inductive bias related to the objectness itself. In contrast to supervised approaches, recent work has found that it is possible to discover object-like entities from raw data in an unsupervised fashion[10–12].

These approaches belong to a VAE framework that learns structured object-centric latent representations by introducing different attention mechanisms to encourage object-level disentanglement. Due to the generative essence of these models, we denote these approaches as object-centric generative models (OCGMs). In contrast to scene segmentation models, OCGMs perform instance segmentation, object pose or position estimation, object-centric latent representation learning, and observation reconstruction in a single model. Each output can then be leveraged by different downstream robotic tasks as high-level visual inputs. Despite the diverse functionalities that a single OCGM model can offer, early OCGMs works only conducted experiments on simple synthetic dataset such as the scenes of moving MNIST digits [11–13]. Thus in this thesis, we investigate whether unsupervised OCGMs can be applied to robotics scenarios that involve objects of diverse and complex shapes, and moreover, we also conduct how the learned object-centric latent representations could potentially facilitate robotic applications.

The first part of this thesis investigates the learning and the application of object-centric latent representations in robotic tasks. By reconstructing the raw observations, the learned latent space of objects can encode a manifold of variations of object shapes and textures. Nevertheless, besides the visual information, we are also interested in whether the object latent space induces sub-manifolds that encode task-relevant properties of the objects, such as the object affordance for a given action in specific environments. In cognitive science, dual-system theory [14] suggests that humans can develop intuition that quickly and automatically maps visual observations to intuitive conclusions based on past experiences. Furthermore, prior research [15] has also found that New Caledonian crows are able to create reaching tools by combining two short combinable parts, which requires anticipating properties of the unseen objects. Such anticipation, or planning on objects, is usually interpreted as involving creative mental modelling. For manipulation tasks, especially tool-use tasks, we posit that the affordance of the tools can be developed by the learning signal established from the trial-and-errors of using the tools in specific tasks. To simulate this process with neural networks, we conduct experiments with

a reaching task, where a robot uses different tools to avoid obstacles in the scene to reach the target. Like how humans or animals are able to infer object properties, a classifier can be trained to replicate this ability by learning the tool affordance from the established dataset of tool-use. This motivates us to unlock the ability of task-oriented tool synthesis that explicitly generates 3D meshes of the synthesised tools by leveraging a classifier that captures a high-level understanding of tool affordance.

An object-centric latent representation is a compact representation that encodes useful object properties which can be used to facilitate downstream robotic tasks. Nevertheless, naturally occurring scenes, especially in robotic scenarios, usually consist of several objects and the background. To discover the objects in the scene in an supervised fashion, the OCGMs can be employed to first decompose the scene into object-like entities and then predict the object-centric latent representations using the encoder module of the model. Therefore, in the second part of this thesis, we propose three OCGMs and apply them to classic robotics tasks. In contrast to previous OCGMs that evaluate models mainly on simple synthetic datasets, we are interested in the unsupervised learning of OCGMs in robotic scenarios. These experiments are more challenging as the objects are real-world objects that have diverse shapes and textures including the robot arm itself. Moreover, in the manipulation tasks, there are interactions between the objects and the robot, which further increases the difficulty of the disentanglement of learned scene components. To this ends, we first explore whether the fully unsupervised approach can still learn meaningful scene components from the raw observations. Leveraging the learned object latent representations and the segmentation masks, we then investigate how these features assist in facilitating manipulation within an actual robotic setting.

As a 2D OCGM only learns to reconstruct object pixels of 2D images, which can provide limited information for robotic applications such as the geometry-based object grasping or collision-aware path planning. Moreover, instead of capturing the 3D geometry of the discovered objects, the learned object-centric latent representations also only model the variations of the objects in the projected 2D image space. To address this limitation, the model can be upgraded to 3D,

leveraging the recent implicit 3D representations, neural radiance fields(NeRFs) [16], as the object- and background-decoder. In 2D OCGM [17], the object-centric latent representation can be factorised into the *where* and *what* components that disentangles the object spatial information from its appearance information. This disentanglement can facilitate scene editing by offering more flexibility and interpretability of the object latent representations. However, recent 3D OCGMs [18, 19] represent each object only using a single latent embedding and thus cannot provide such benefits. To solve this problem, it is conceivable that in contrast to 2D OCGMs, the object shape is not only perceivable in observations such as the point clouds, but can also be predicted as outputs of NeRFs. We therefore propose to leverage a minimum volume principle to estimate the object poses based on the object shapes without using any human supervision or known object templates. Besides the design choices of the models, it is also important to deploy the 3D OCGM in real-world robotic scenes and evaluate it's performance in several downstream perception tasks. These experimental investigations constitute the last contribution of this thesis.

1.2 Guiding Questions

In this thesis, we focus on four guiding questions that investigate the application of OCGMs in robotic tasks. The guiding questions that we propose are the following:

1. Given that generative models naturally capture factors of variation, could they also be used to expose these factors such that they can be modified in a task-driven way?

2. How well does an OCGM learn disentangled components from robot interaction datasets in a fully unsupervised fashion?

3. Can an OCGM explicitly model the 3D geometry of objects and the background while still being able to disentangle the object spatial information from its appearance information, as in 2D OCGMs, without using any known CAD models or templates of the objects?

4. Can the 3D OCGM be applied to real-world robotics scenarios and achieve various functionalities for scene understanding, such as the object detection, 6D pose estimation, object-centric latent representation learning and 3D reconstruction, to facilitate downstream robotic tasks without using any human supervision?

More specifically, we aim to show that (a) by using an affordance predictor (a classifier) trained on a learned latent space of various tools, the model can learn from the simulated experiences of trial-and-errors of tool-use in different reaching tasks to identify the task-relevant trajectory in the latent space. This affordance predictor can then be leveraged to perform task-driven tool synthesis together with the learned tool decoder; (b) by deploying the OCGM to robotics scenarios that involve intensive interactions between the robot and the objects, the model can still learn meaningful object components in an unsupervised fashion and leverage the learned object-centric latent representations to facilitate the object rearrangement tasks; (c) a 3D OCGM that leverages the heuristic of minimum volume principle for unsupervised object pose estimation and object-centric scene generation and scene editing. (d) object pose estimation is improved by a 3D OCGM that is eventually applied to real-world robotics scenarios with a shape completion module. We then demonstrate that a fully unsupervised OCGM can provide several functionalities for scene understanding, such as the object detection, 3D reconstruction, 6D pose estimation and latent representation learning, which can be potentially leveraged by downstream robotic applications.

1.3 Thesis Outline

We begin the thesis by providing the background and an overview of the existing literature on affordance learning and OCGMs in Chapter 2. The Chapters 3 to 6 aims at answering the guiding question and provide the concrete theory, method and the experiment to address the questions raised above. In the remainder of this section, we will first provide a brief overview of individual chapters from Chapter 3 to 6 and how they address each guiding question. Each of the subsections summarises

the motivation, contribution and conclusion of Chapters 3 to 6. An overview of the key contributions in this thesis is illustrated in fig. 1.1.

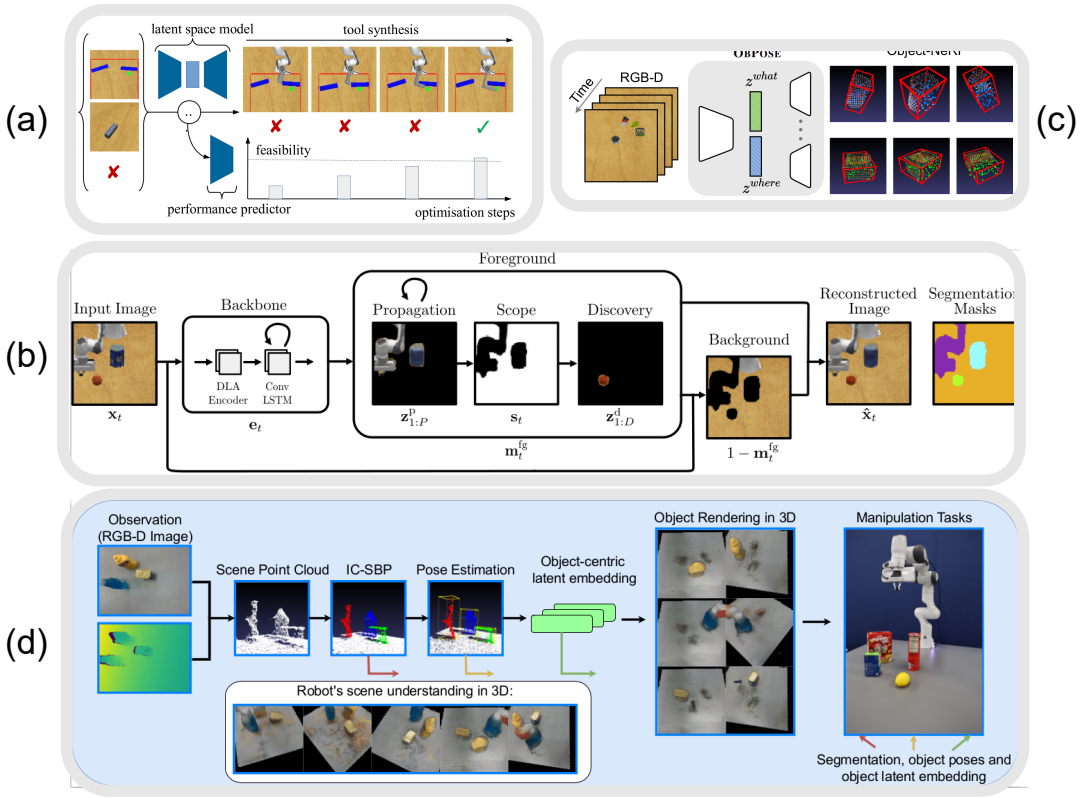


Figure 1.1: An overview over the key contributions of this thesis. (a) In Chapter 3, we train an affordance predictor of tool feasibility in a reaching task and utilise it to synthesis tools in a task-driven way. (b) In Chapter 4, we propose an unsupervised OCGM in 2D for robotics scenarios and evaluated its performance in two common robotic tasks: robot pushing and pick-and-place. (c) In Chapter 5, we upgrade the 2D OCGM to 3D by employing NeRFs as the decoder to explicitly model the 3D geometry of the objects and the scene. We proposes minimum volume principle as heuristic to allow the unsupervised 6D pose estimation so that the appearance and spatial information of the objects is disentangled, which also facilitates scene editing. (d) In Chapter 6, we deploy the 3D OCGM in real-world robotics scenarios. We improve the pose estimation by introducing a shape-completion module for handling the occlusions. We also demonstrate the advantages of using OCGM as the scene understanding module for robotic tasks compared to the vanilla NeRFs by evaluate the OCGM performance in several perception tasks in real-world.

Finally, Chapter 7 revisits our guiding questions, summarises the conclusions and contributions. We also analysis the limitations of the proposed approaches, and provides an outlook for future research directions building upon the contributions of this thesis.

1.3.1 Tool Synthesis Using Affordance Predictor Learned from Tool-use Experiences

Research in cognitive science [14] highlights that, human can develop intuition that automatically and efficiently maps the visual observation to complex patterns of ideas, which includes the approximation of the outcome of a physical process. Gibson [20] defines the concept of *affordance* to describe the intuition of whether an object in the environment is affordable for specific actions. For the tool-use tasks, we posit that an intuition of the affordance of the tools can naturally emerge as the experiences of tool-use accumulate. To these ends, we are interested in whether this process can be replicated using neural networks and then applied to tool synthesis tasks.

To simulate the experiences of tool-use, we collect a dataset that consists of paired data of tool image, task image and a label of the feasibility of the given tool for the task. Concretely, we investigate a reaching task where the robot should use either a stick or a hook to reach a target in the scene while avoiding collision with the obstacles. We train a generative model on the tool images using a differentiable mesh decoder [21] that directly models the 3D geometry of the tools. An affordance predictor is trained together with the learned latent space of the tools to predict the feasibility of the tools for the given task image that describes the obstacles and the reaching target in the scene. Once trained, we leverage the activation maximisation [22] to traverse through the learned latent space of the tools to reach a point that has a highest affordance for the given task. The found latent vector can subsequently be decoded into explicit 3D meshs, which enables the visualisation of the entire process of the tool synthesis.

The experiments presented in section 3.4 indicate that proper tools can be synthesised with high accuracy given the corresponding task image using the learned affordance predictor and the mesh decoder of the tools. Using the activation maximisation the model can traverse from a start point of low affordance in the latent space to a point that corresponds to high affordance for the given tasks. Moreover, the visualisation depicted in fig. 3.4 shows that the morphing of the tools

is a continues process, which indicates the existence of a trajectory in the latent space along which the task-relevant affordance varies continuously. In contrast to previous works in latent representation learning [23, 24] that focus on axis-aligned disentanglement driven by the Gaussian prior, our work sheds light on another possibility on the latent space manipulation, i.e. using a classifier to perform task-oriented, high-level traversal in the latent space. Following our work, [25] also demonstrates that a real quadruped robot is able to find the stable walking pose by guiding the pose interpolation in latent space using a trained high-level stability classifier.

Another interesting finding of this work is that, we found an unseen T-shape tool (see the right-middle example of fig. 3.4) is created when the hook points to left side but the target is on the right. This T-shape tool can be viewed as the interpolation between the hook pointing to the left and the hook pointing to the right. The creation of the T-shape tool indicates that the model is able to first identify which part of the tool is important for the success of the task and only modify that part intentionally. We hope this finding could provide insights on future works on novel tool creation.

1.3.2 Unsupervised 2D OCGM for Robotic Scenes

Following section 1.3.1, we have demonstrated the ability of tool synthesis using the learned latent space of the tools and a high-level affordance predictor. Since this section, we begin with developing OCGMs that can discover object-like entities and learn the object-centric latent representations from raw observations without using human labels. Previous works[10, 11, 26] have found that by leveraging the reconstruction bottleneck in the VAE, the model can segment the scene into object-like entities. This disentanglement is motivated by different attention mechanisms proposed in these works. One of the choices is the spatial attention [12, 13, 27] that allows the model to encode the object latent representations separately into the *where* component that captures the location of the objects and the *what* component that encodes the shape and appearance of the objects. However, these models

conduct experiments mainly on simple synthetic dataset such the moving MNIST digits. We are thus interested in whether the unsupervised OCGMs can also be applied to robotics scenarios, where the scenes have objects that vary widely in size and texture, including the complex articulated object, the robot arm, as well.

To conduct this experiment, we create a pushing dataset consists of trajectories of a Panda arm pushing the YCB objects [28] randomly on the table-top. We also test our model using the real-world Sketchy dataset [29], where videos of a robot gripper performing pick-and-place tasks are recorded. We propose the model APEX that introduces a principled mask normalisation algorithm and a feature-map scene encoder. For the OCGMs, the scenes are explained by multiple components including the foreground and background component. The proposed mask normalisation algorithm first models whether a pixel is jointly occupied by all foreground components and then distribute the occupancy to a single component using a *softmax* operation. In the OCGMs, the VAE operation runs multiple times in each training iteration and is the most computationally expensive operation in the model. To mitigate computation complexity, we propose to perform the encoding not on the input images but on a down-sampled feature map that reuses the outputs of the scene encoder. The benefit of encoding the feature map is that, for a CNN-based encoder, increasing the input image size can result in a quadratic growth of computation, but increasing the channel size only leads to a linear growth of the computational cost. A feature map has a down-sampled size than the original image with a larger channel size. By directly encoding the feature map, we could thus save computations in the VAE-operations. Results in Chapter 4 show that, the model can still successfully segment and track the objects using the proposed encoding approach. Moreover, given the learned object latent representation, we also demonstrate that it can facilitate the robot skill execution on an object arrangement task. Another interesting finding of this work is that, for the most complex object in the dataset, i.e. the robot arm, the model learns to segment it differently in different dataset. For the Panda pushing dataset, the model segments the robot arm into a single entity, whereas it segments the

robot gripper into two separate components in the pick-and-place dataset(Sketchy dataset). We posit that this is because there is relative motion between the grippers in the pick-and-place tasks. This different segmentation results for the same object suggest that there is no unique correct segmentation if we do not provide additional inductive bias to specify the concrete level of abstraction.

1.3.3 Unsupervised OCGM with Pose Estimation in 3D

In section 1.3.2, we showed that an OCGM can learn meaningful object representations in a fully unsupervised fashion in robotics scenarios. However, the robot manipulation is not only object-centric but also in 3D. The 2D decoder in the model can identify the object pixels in the scene image using the decoded object masks, but this projected geometry information is not sufficient for robotics manipulation tasks, such as the geometry-based grasping and the collision-aware path planning. Moreover, the object spatial information in 3D, i.e. the object 6D pose, is also more informative than the 2D bounding boxes inferred from 2D images. For example, the object 6D pose has been shown to facilitate the object grasping [30]. Estimating the 6D poses of the objects usually requires a known CAD Library or templates of the objects that pre-define the canonical poses. To preserve the self-supervised fashion of learning, we assume no such inputs are available both in the training phase and the testing phase. Moreover, the pose estimation under this condition can be ill-posed if there is no unique *correct* pose for the objects that exhibit symmetric shapes.

To address these problems, we propose ObPose that employs NeRFs as the object decoder to explicitly model the object geometry in 3D. We choose RGB-D images as inputs, which is easily available using the depth camera in robotics experiments. Given the camera extrinsic- and the intrinsic parameters, the point clouds can be computed from the depth image and it can be viewed as a discrete sampling of the surface of the observed scene. Therefore, the point cloud observation partially preserves the geometry of the scene in 3D. We use the instance colouring stick-breaking process [31] to perform clustering on the input point clouds for the scene segmentation. The segmented object point clouds thus can be viewed as an

approximation of the original object shape. ObPose defines the pose of the object using a bounding box containing the object that has a minimum volume. Due to the symmetry of the objects, the volume of the bounding box remains unchanged if a pair of the coordinate axes is swapped. We thus choose the bounding box that has the closest distance to the world coordinate as the final bounding box representing the object pose. This heuristic allows the model to predict unique object pose without using any pre-defined labels or object templates.

The experimental results show that ObPose is able to discover meaningful object entities in 3D scenes and to the best of our knowledge, it is also the first NeRF-based OCGM that can estimate 6D poses of the discovered objects without using human supervision. Therefore, ObPose is able to learn object-centric latent representations that only capture the object appearance information. The spatial information is disentangled from the latent representations as in the 2D OCGM works. In Chapter 5.5.3 we further demonstrate that this property facilitates the scene editing. The training of the model uses single view RGB-D images, which is different from the baseline [18]. We observe that despite without having access to multi-view supervision, the model is still able to reconstruct the 3D shapes of the objects.

1.3.4 3D OCGMs in Real-world Robotic Applications

In the last part of this thesis, we develop an OCGM that is applied to real-world robotics scenarios. The current NeRF-based approaches for the robotic applications [32] necessitates retraining the NeRF from scratch before each action, which severely hinders the flexibility of deploying those models in robotic tasks because the environment states need to be frequently updated. Nevertheless, the NeRF-based OCGMs can recognise the objects using the learned object encoder and thus alleviate the burden of retraining the model once it is trained in a self-supervised fashion. This property of the NeRF-based OCGMs is thus of paramount importance for the robotic applications. In ObPose, the minimum volume principle is used for estimating the object poses, but we found that due to the occlusions in the scene, the object shape observed from single-view images can not always be sufficient for

accurate pose prediction. On this basis, we propose DreamUp3D that introduces a shape completion module to inpaint the unobserved parts of the objects before the pose estimation. The shape completion module is supervised via self-distillation that allows it to learn the shape information of the objects by reusing the predictions of the object decoder, which drastically saves the computation during the training.

Beside the model itself, another contribution of this work is to evaluate the performance of the proposed 3D OCGM in real-world scenarios. The real-world robotic data is more expensive to collect than simulation data. However, pretraining the models with simulation data can have performance drop stems from the sim-to-real gap of the data distribution. The OCGM is a self-supervised learning model and thus does not require labelling for training. This property makes it possible to directly collect real-world data for training. The experiment in Chapter 6 demonstrates that our final model, DreamUp3D, can successfully segment the scene into object-like entities in the real-world. Moreover, comparing to NeRFs approaches, the DreamUp3D takes only one RGB-D image as input while the NeRFs necessitate a dense, multi-view images as inputs for the scene reconstruction. The inference time of DreamUp3D is also reduced to only a fraction of a second for each update of the current scene states, which is much faster compared to retraining the NeRF from scratch. Moreover, we demonstrate that the learned object-centric latent representation can be leveraged for the object matching, and the inferred object poses can also facilitate the object rearrangement task in real-world.

1.4 Publications and Contributions

The following publications, listed in their order of appearance, form the main body of chapter 3 to 6.

1. Y. Wu, S. Kasewa, O. Groth, S. Salter, L. Sun, O. Parker Jones, and I. Posner. "Learning Affordances in Object-Centric Generative Models". In: Workshop on Object-Oriented Learning at ICML 2020 (July 2020)

The work [6] won the best paper award in Workshop on Object-Oriented Learning at ICML 2020. The proposed approach has been applied to several works across a

wide range of tasks including locomotion of a real quadruped robot:
A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, S. Gangapurwala, O. Melon, I. Havoutis, and I. Posner. “First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion”. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).
and path planning for reaching tasks:
C.-M. Hung, S. Zhong, W. Goodwin, M. Engelcke, O. Parker Jones, I. Havoutis, I. Posner. Reaching Through Latent Space: From Joint Statistics to Path Planning in Manipulation. IEEE Robotics and Automation Letters (RA-L) with ICRA (IEEE International Conference on Robotics and Automation) 2022.

2. Y. Wu, O. Parker Jones, M. Engelcke, and I. Posner. "APEX: Unsupervised, object-centric scene segmentation and tracking for robot manipulation." In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

3. Y. Wu, O. Parker Jones, and I. Posner. "ObPose: Leveraging Pose for Object-Centric Scene Inference and Generation in 3D." In: *arXiv preprint arXiv:2206.03591.*

4. Y. Wu, H. Sáez de Ocáriz Borde, J. Collins, O. Parker Jones, I. Posner. "DreamUp3D: Object-Centric Generative Models for Single-View 3D Scene Understanding and Real-to-Sim Transfer." Submitted to *IEEE Robotics and Automation Letters (RA-L)*

2
Background

In this chapter, we provide some context for deep generative models and an overview of topics related to the central themes of the thesis: affordance learning and object-centric generative models. We first introduce the latent variable models and the object-centric generative models from section 2.1 to section 2.2.2. Section 2.3 then covers related work; 2.3.1, background knowledge about affordance learning; and section 2.3.2, the recent evolution of object-centric generative models.

2.1 Latent Variable Models

The latent variable model is one type of generative model that attaches an unobserved latent variable $\mathbf{z}_i$ to each observation x_i, where the latent variable $\mathbf{z}_i$ is expected to capture meaningful variations, e.g. textures of objects, ages of human faces or view angles of the scenes, etc. A typical generative process follows:

$$\mathbf{z}_i \sim \mathcal{N}(\mathbf{z}|0, \mathbf{I}), \mathbf{x}_i \sim \mathcal{N}(\mu_\theta(\mathbf{z}_i)|\text{diag}(\sigma_\theta(\mathbf{z_i}))) \tag{2.1}$$

Here μ_θ and σ_θ are usually parameterised by deep neural networks. Given a large scale dataset that could involve millions of latent variables, the inference of the posterior $p(\mathbf{z}|\mathbf{x})$ becomes tractable by introducing amortized variational inference [33, 34]. This learns the shared patterns across the approximate posterior $q(\mathbf{z}|\mathbf{x})$ of

each pair of $\{\mathbf{z}_i, \mathbf{x}_i\}$. To learn disentangled latent representations, [23] rewrites the variational free energy objective function [35] (also known as evidence lower bound(ELBO)) to:

$$\mathcal{L}(\theta, \phi, \beta; \mathbf{x}, \mathbf{z}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}))] - \beta \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})), \tag{2.2}$$

where β is a hyperparameter that balances the regularisation strength and the reconstruction accuracy. $\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$ is the *Kullback-Leibler divergence* [36](*KL divergence*) which measures the distance between the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z})$. θ and ϕ are the parameters utilised for modelling the likelihood and the approximate posterior distribution. The expectation of the second term in Equation 2.2 under the data distribution $p(\mathbf{x})$ can be further broken down into two terms:

$$\mathbb{E}_{p(\mathbf{x})}[\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))] = \mathrm{I}(\mathbf{x}; \mathbf{z}) + \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z})) \tag{2.3}$$

where $\mathrm{I}(\mathbf{x}; \mathbf{z})$ is the mutual information between $\mathbf{x}$ and $\mathbf{z}$ under the distribution of $p(\mathbf{x})q(\mathbf{z}|\mathbf{x})$. We refer readers to [23] for detail derivation. Penalising the $\mathrm{I}(\mathbf{x}; \mathbf{z})$ will reduce the amount of information about $\mathbf{x}$ captured by $\mathbf{z}$ whereas penalizing $\mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}) \parallel p(\mathbf{z}))$ pushes the marginalised approximate posterior $q(\mathbf{z})$ towards the multivariate Gaussian prior $p(\mathbf{z})$. This helps to explain why the latent representation learned by the variational autoencoders (VAEs) has the effects of disentanglement along each latent dimensions and how this disentanglement strength might be adjusted by the hyperparameter β. Given a latent space disentangled along each dimension, we can manipulate the learned latent representation by e.g. traversing each dimension or perform arithmetic operations.

Despite these promising properties, prior work has mostly conducted the experiments using single objects, e.g. faces, chairs, MNIST numbers, etc. However, in real world a scene is usually comprised of several objects and the background. When humans analyse a complex scenario we naturally attend to only the part of the image that is most relevant to the current task. We suggest it is intuitive to first decompose a scene into several components, and then learn the corresponding latent representation of each segmented component and its reconstruction. This

is especially benificial for robotic tasks that involves object-centric operations. Fortunately, this behaviour can still be implemented by learning to reconstruct the observed scene with several properly designed attention mechanisms introduced in the encoding step.

2.2 Object-centric Generative Models

Compared to VAE that encodes the scene into a single latent representation, the OCGMs first learn to decompose the scene into several scene components and the background. Next, a component VAE is employed to encode and reconstruct each segmented scene component, which is later composed to reconstruct the entire scene. In this way, the OCGMs can learn object-centric latent representations and the decoder in the component VAE also models the geometry and texture of each object independently. This object-level disentanglement offers more flexibility for downstream robotic tasks as a scene understanding module. Next, we will explain each module in the OCGMs in detail.

2.2.1 Scene Encoder

The scene encoder in OCGMs encodes the raw RGB image or point cloud inputs into pixel-wise or point-wise embeddings. For the static scene, a U-Net-like [37] structure can be employed, whereas for the video data a ConvLSTM [38] can be used to capture the spatial-temporal information. The predicted scene embedding can be of the same size of the inputs or be down-sampled for the improvement of the computational efficiency. For the image inputs, the CNN layers can be used and for the point cloud inputs we can use the KPConv [39] layer for the encoding. The encoded scene embeddings will then be clustered into scene components and then be further encoded into object-centric representations. As these are two different types of tasks, different features are needed for the learning. The learning of these two types of features can be achieved using two additional MLPs following the scene encoder.

2.2.2 Attentions in the OCGMs

The key designs that cause the unsupervised scene decomposition and also differentiate each OCGM from each other is the special attention mechanisms used in the OCGMs. Next, we will explain different attention mechanisms that can motivate object-centric disentanglement in detail.

Sequential Slot-Based Attention

Sequential slots-based attention [10] employs an attention net that recurrently generates masks over a sequence of steps to condition a component VAE. The component VAE is a VAE that encodes the masked observations into the latent representations and then decodes it to reconstruct the masked area of the scene. The recurrent attention network predicts the component masks and tracks which parts of the image have yet to be explained using an autoregressive process:

$$\mathbf{m}_1 = \alpha_1, \mathbf{m}_k = \mathbf{s}_{k-1} \odot \alpha_k, \mathbf{m}_K = \mathbf{s}_K, \tag{2.4}$$

Here the component mask $\mathbf{m}$ is computed as the multiplication of the attention mask α and the *scope* $\mathbf{s}$, which is initialised and updated as follows:

$$\mathbf{s}_0 = \mathbb{1}^{N\times 1}, \mathbf{s}_k = \mathbf{s}_{k-1} \odot (1 - \alpha_k), \tag{2.5}$$

The *scope* $\mathbf{s}$ keeps track of which pixels in an image have been accounted for. The ones in the *scope* $\mathbf{s}$ indicate unexplained pixels, and zeros represent the already explained part of the image. Without introducing any further inductive bias, [10] demonstrates that the component masks $\mathbf{m}$ can converge to object-like entities by only reconstructing the observed scene composed by the decoded scene components using the component VAE. The learned object-centric latent representations can encode meaningful variations along each dimension of the latent embedding, e.g. the location of the object or the size of the object, which can be visualised by traversing a single dimension of the learned latent embedding and then decoding the modified latent embedding.

Another approach focused on sequential slots-based attention [11] further factorises the object-centric latent representation into a separate triplet of continuous latent variables $\mathbf{z} = \{\mathbf{z}^{what}, \mathbf{z}^{where}, \mathbf{z}^{pres}\}$. Each triplet of latent variables explicitly encodes the position, appearance and presence of each object. Such a disentanglement provides more flexibility for scene editing and is more intepretable compared to encoding the location and appearance jointly using a single latent embedding. For each scene component, the $\mathbf{z}^{what}$ is decoded using a glimpse decoder and then scaled and shifted by a spatial transformer [40] according to $\mathbf{z}^{where}$. As the inference is sequential, the encoder can be implemented utilising a recurrent neural network(RNN) to approximate the dependency between the latent embeddings:

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(z^{pres,n+1} = 0|\mathbf{z}^{1:n}, \mathbf{x}) \prod_{i=1}^{n} q_\phi(\mathbf{z}^{where,i}, \mathbf{z}^{what,i}, z^{pres,i} = 1|\mathbf{z}^{1:i-1}, \mathbf{x}) \quad (2.6)$$

This dependency in the inference step is introduced to avoid encoding the same object twice. However, sequential inference can impose an unnatural ordering on objects [41], which is not desirable as the order of the slots should be permutation-equivariant in the general case where no additional assumptions are imposed on the objects in the scene. To address these limitations, another group of works [41, 42] introduce the instance slots-based attentions to prevent the model from learning an ordered set of slots.

Instance Slot-Based Attention

The first instance slots-based attention we introduce in this section is the slot attention [42]. The slot attention module maps from a set of input feature vectors to a set of output vectors that we refer to as slots. Each slot can then be decoded into a scene component and captures object information such as object shape, object texture etc. Slot attention uses an iterative attention mechanism to map from its inputs to the slots. To prevent the model from learning an unnatural ordering of the slots, the slots are initialised at random and then refined for several iterations to bind to a particular part of the input features. Concretely, given the input feature vectors $\mathbf{e}$ and the randomly initialised slot $\mathbf{o}$, the slot attention module first uses learnable

linear transformations k, q, and v to map inputs and slots to a common dimension D, and then computes the dot-product attention with attention coefficients that are normalised over the slots to introduce competition between the slots:

$$\alpha_{att} = \pi_{\mathrm{SM}}(\frac{1}{\sqrt{D}} k(\mathbf{e}) q(\pi_{\mathrm{LN}}(\mathbf{o}))^T) \tag{2.7}$$

The α_{att}, π_{SM} and π_{LN} denote the computed attention, the softmax operation and the LayerNorm operation respectively. To aggregate the input values to their assigned slots, the update $\mathbf{u}$ to the slots can be computed using a weighted mean as follows:

$$\mathbf{u} = (\alpha_{att} + \epsilon)(v(\mathbf{e})) \tag{2.8}$$

A Gated Recurrent Unit (GRU) [43], $\mathbf{f}_{GRU}$, is employed to update the slots utilising the aggregated input information:

$$\mathbf{o} = \mathbf{f}_{\mathrm{GRU}}(\mathbf{o}', \mathbf{u}) \tag{2.9}$$

with $\mathbf{o}'$ being the slot from refined in last iteration (for the first iteration, the $\mathbf{o}'$ is then the randomly initialised slot). An optional residual connection can also be implemented to facilitate learning:

$$\mathbf{o} = \mathbf{o} + \mathbf{h}(\pi_{\mathrm{LN}}(\mathbf{o})) \tag{2.10}$$

with $\mathbf{h}$ being a mapping function parameterised by a MLP. After T iterations of the slot attention on the input features, the slots can converge to object-like entities after training by reconstructing the observed scenes. In contrast to sequential slot-based attention, the slot attention exhibits two key properties [42]: (1) permutation invariance with respect to the input and (2) permutation equivariance with respect to the order of the slots. These properties allow the predicted slots not to specialize to one particular type of object, which facilitates generalisation.

Another instance slots-based attention is the Instance Colouring Stick-Breaking Process (IC-SBP) [41]. The IC-SBP is a non-parametric, differentiable algorithm that clusters pixel embeddings into a variable number of soft attention masks $\mathbf{m}$.

The soft attention masks are computed following the same stick-breaking process as in sequential slots-based attention:

$$\mathbf{m}_1 = \alpha_1, \mathbf{m}_k = \mathbf{s}_{k-1} \odot \alpha_k, \mathbf{m}_K = \mathbf{s}_K, \tag{2.11}$$

The *scope* $\mathbf{s}_k \in [0,1]^{N\times 1}$ tracks which pixels have not yet been explained. $\mathbf{s}_k$ is initialised and updated as follows:

$$\mathbf{s}_0 = \mathbb{1}^{N\times 1}, \mathbf{s}_k = \mathbf{s}_{k-1} \odot (1 - \alpha_k) \tag{2.12}$$

α_k is computed as the distance between a sampled cluster seed and all individual pixel embeddings (we assume RGB here) according to a kernel ψ. To prevent the model from learning a fixed order of segmented masks, the seed is sampled by first computing a matrix of seed scores, which is initialised by sampling from a uniform distribution $\mathbf{c} \sim U(0,1)$. The seed scores are then multiplied by the scope matrix to mask out the explained pixels. The pixel embedding that has the highest score is then selected as the cluster seed. The sampling from the uniform distribution guarantees the order of the components to be stochastic, therefore there will be no fixed order of the scene components to be learned during training. Different from the slot-attention, the IC-SBP assigns each scene component to an actual pixel, which allows additional inductive bias for the clustering to be introduced. For example, the pixels surrounding the sampled cluster seed could have a higher probability to be clustered together.

Spatial Slot-Based Attention

The two types of attention discussed above employ either sequential inference or iterative refinement to learn a set of order-free object-centric representations. These approaches, however, can be computationally expensive when scaled to many objects in a scene. Inspired by the classic object detection framework [3], the spatial slots-based attention assigns each slot to a feature in a feature map predicted by a CNN-based encoder in parallel. Similar to [11], each feature is further decoded to a triplet of continuous latent variables $\mathbf{z} = \{\mathbf{z}^{what}, \mathbf{z}^{where}, \mathbf{z}^{pres}\}$ to encode the

appearance, location and the presence in a disentangled way. Consider a feature map of size HxW: individual features of the feature map are spatially separated from each other. This is an inherent inductive bias that prevents two slots from explaining the same area, i.e. different objects should be in different locations. A prior distribution over the $\mathbf{z}^{where}$ can also be chosen to specify the average size of the objects in the scene. However, this either requires a priori knowledge about the object sizes in the dataset or is infeasible, because in a general setting the object sizes can be arbitrary, e.g. a huge object occupies the full image. In this sense, the key inductive bias of the spatial slots-based attention might not be valid.

2.2.3 Component VAE

After segmenting the scene into scene components, the component VAE will then encode each segmented scene component into object-centric latent representation and decode them to reconstruct the scene. The most of the computational cost of the OCGMs is spent on this operation. Consider a video of N frames, if the model performs D times of object proposals and tracks P detected objects from previous time steps, then $N \times (D + P)$ times of VAE operations are needed to discover new objects and update the states of objects that are detected and tracked. Therefore, any improvements on the computational efficiency of the component VAE can drastically reduce the computational cost of the OCGMs. For the encoder in the component VAE we can use 2D CNN-based encoder for the images and KPConv-based encoder for the point clouds. The decoder can be a vanilla CNN-based decoder or a Spatial Broadcast Decoder [44]. The latter is shown to be able to help VAEs learn disentangled representations and is therefore commonly chosen in 2D OCGMs [10, 45]. For robotic applications, nevertheless, a 2D decoder can not provide sufficient 3D information for downstream tasks such as the robot manipulation. Therefore, a 3D decoder that directly models the geometry and the texture of the objects can be introduced into the OCGMs. Currently, the 3D decoder can be a generative radiance field (GRAF) [46]. The GRAF is a continuous function which maps the object-centric latent representation, the coordinate and

the viewing direction of a query point to a volume density and an RGB color value. We can learn the GRAFs by parameterising this continuous function with a MLP.

2.3 Related works

In this section, we present works related to the various aspects of the central investigation of this thesis: affordance learning and object-centric generative models.

2.3.1 Affordance Learning in Robotic Tasks

In cognitive psychology, the concept of *Affordance* [20] defines the action possibilities that are readily perceivable by an actor, e.g. a door handle can be turned. According to [20], the world is perceived not only in terms of object shapes and spatial relationships but also in terms of object possibilities for action. The affordance therefore serves as the bridge between the perception and the action of the agent. As in computer vision tasks, visual affordance recognition also aims to classify, detect and segment the objects' affordances in a scene.

A affordance classification models are similar to image classification models that use an encoder to learn visual features and predict an affordance label as the output. The encoder can be a CNN encoder [47] that learns a global feature for the whole image. However, classification accuracy can be influenced by the noisy background features in the observations. To alleviate the influence of the background features, [48] proposes to employ RPN[3, 49] to first extract the object region and then classify the object affordance within the detected object area in the image. Similar to object detection tasks, the interference of object occlusion in complex scenes is still challenging to address. Therefore, datasets that richly contain these situations are of paramount importance for the final performance of the model. To this end, [50] and [51] proposed large-scale object affordance detection datasets to handle scenes that contain intensive occlusion of the objects.

Affordance segmentation is a more precise way to predict affordances. Affordance segmentation is critical for robots interaction because the robot usually only manipulates a part of the object, e.g. the handle of the mug instead of the

whole mug. Similar to instance segmentation tasks, affordance segmentation models [48, 52] also employ a detect and segment pipeline. For these approaches, the segmentation accuracy therefore depends on the accuracy of the object detection. Another group of works propose detection-free [53, 54] approaches that improve the robustness of the affordance segmentation by proposing a context information-based affordance segmentation model that combines the global information and the local information for the affordance prediction. To train the segmentation model, several datasets [55–57] are constructed and conducted as well.

In contrast to the discriminative approaches discussed above that learn affordance prediction directly from human supervision, in this thesis we investigate a weakly supervised, generative approach for affordance learning (see Chapter 3). Our approach first learns a latent space of different tools used for a reaching task via unsupervised reconstruction of the multi-view RGB-D images of the tools. The latent space therefore encodes the shape information of the tools applied to the reaching task. To expose the affordance manifold induced by the learned latent space, we train a classifier (affordance predictor) to learn to predict the feasibility of the tools in the reaching tasks. The trained classifier can then be leveraged to find trajectories traversing the latent space from spaces of low feasibility of the tools to spaces of high feasibility of the tools. Our approach demonstrates that such trajectories exist and can be found by the affordance predictor in a latent space of the objects learned in an unsupervised way, which can also facilitate tool-synthesis using the 3D decoder of the tools.

2.3.2 Object-centric Generative Models

Scene understanding based on object-centric representations has advantages of interpretability, compositional generalisation and data efficiency. Object-centric learning favors using minimum supervision for training and focus on inductive biases that encourage the model to factorise the scene observation into meaningful, object-level entities. The inductive biases used for object-centric learning can be mainly divided into three groups: sequential slots-based, spatial slots-based and instance

slots-based. Early OCGM works [10–12, 58] learn ordered sequences of vectors via applying sequential attention to images. In each step, an attention mask is computed, an auto-encoder network then encodes and decodes the masked observations and learn the object-centric latent representation through the reconstruction bottleneck [26]. A *scope* is usually used to track which pixel has not been explained yet. As the inference is a sequential process, the model has to run N calls of a VAE for N Objects in the scene. Moreover, due to the dependency of the sequential inference, the model is computationally expensive to scale up to large amounts of objects in the scene. Inspired by the Region Proposal Network(RPN)[3, 49] in the classic object detection framework, the spatial slots-based OCGMs [13, 27, 59, 60] associate object slots to different spatial locations and learn a grid of object-centric latent vectors in a fully parallelised fashion. This reduces the time complexity of processing and improves the scalability in terms of object density. Another limitation of the sequential-based OCGMs is that, the learned object-centric representations has an inherent order due to the dependency in the inference process. This is counter-intuitive as the object slots should be symmetric without further inductive biases being introduced. To address this problem, instance-based OCGMs [31, 42, 45, 61–65] propose attention based grouping to learn a permutation-invariant set of vectors. A random sampling step is usually employed to ensure the object-centric vectors are order-free. Besides static scenes, video-based OCGMs [13, 17, 66, 67] are proposed to additionally leverage spatial temporal consistency for learning object entities. With video-based models, OCGMs can also perform object tracking throughout the video sequence [13, 17].

With the advent of differentiable rendering [68], recently proposed OCGMs [18, 46, 69–71] have replaced the 2D CNN decoders the 3D decoders such as NeRFs[16] to explicitly model the 3D geometry and the texture of the objects and the scene. The training of the models are supervised by posed multi-view RGB images, and it can be further accelerated by using the RGB-D images to allow training the model without explicit ray marching [18]. These OCGMs allow novel-view synthesis for downstream tasks. As being unsupervised approaches, the OCGMs will fail in

complex real-world scenarios. Recent works use the semi-supervised training for the real-world datasets. DINOSAUR [72] proposes that rather than training on the raw image pixels, training on the pre-trained DINO [73] features can lead to obvious improvements on instance segmentation accuracy in real-world scenarios. To handle more complex data, more recent models [74, 75] have also employed diffusion model [76] to improve the modeling capacity.

In this thesis, we focus on the investigation on OCGMs applied to robotic scenes. We investigate the unsupervised segmentation performance of OCGMs in robotic tasks that involve complex, articulated robot arms and find the limitation of the current inductive bias for scene segmentation in OCGMs, i.e. for complex objects such as the robot arms, the segmentation results can be not unique in robotic tasks where the robot arms can have different task-dependent motions. Additional inductive biases are thus necessary to further motivate the desired segmentation results of the robot arms. We also emphasis the limitations of 2D OCGMs for robotic tasks caused by not directly modelling the 3D geometry of the objects which is of pivotal importance for robot manipulation. Therefore, we propose two 3D OCGMs (see Chapter 5 and Chapter 6) that, to the best of our knowledge, are the first OCGMs that can disentangle the spatial information of the objects from its appearance information as in the 2D OCGMs. We also demonstrate how this disentanglement facilitates the scene editing by providing more flexibility for scene inference. In addtion to the proposed models, we also, to the best of our knowledge, for the first time apply the 3D OCGMs and emphasis the benefits of using 3D OCGMs for scene understanding in robotic tasks compared to the commonly used NeRF-based approaches. We demonstrate these advantages by comparing its performance in several perception tasks in real-world robotic scenes.

3

Learning Affordances in Object-Centric Generative Models

In this chapter we explore the richness of information captured by the latent space of a vision-based generative model. The model combines unsupervised generative learning with a task-based performance predictor to learn and to exploit task-relevant object *affordances* given visual observations from a reaching task, involving a scenario and a stick-like tool. While the learned embedding of the generative model captures factors of variation in 3D tool geometry (e.g. length, width, and shape), the performance predictor identifies sub-manifolds of the embedding that correlate with task success. Within a variety of scenarios, we demonstrate that traversing the latent space via backpropagation from the performance predictor allows us to *imagine* tools appropriate for the task at hand. Our results indicate that affordances – like the utility for reaching – are encoded along smooth trajectories in latent space. Accessing these emergent affordances by considering only *high-level* performance criteria (such as task success) enables an agent to manipulate tool geometries in a targeted and deliberate way. This work was presented orally and received an outstanding paper award as:

Y. Wu, S. Kasewa, O. Groth, S. Salter, L. Sun, O. Parker Jones, and I. Posner. "Learning Affordances in Object-Centric Generative Models". In: Workshop on

Object-Oriented Learning at ICML 2020 (July 2020)

3.1 Introduction

The advent of deep generative models [e.g. 10, 61, 77] with their aptitude for unsupervised representation learning casts a new light on learning *affordances* [20]. This kind of representation learning raises a tantalising question: Given that generative models naturally capture factors of variation, could they also be used to expose these factors such that they can be modified in a task-driven way? We posit that a task-driven traversal of a structured latent space leads to *affordances* emerging naturally along trajectories in this space. This is in stark contrast to more common approaches to affordance learning where it is achieved via direct supervision or implicitly via imitation [e.g. 78–82]. The setting we choose for our investigation is that of tool synthesis for reaching tasks as commonly investigated in the cognitive sciences [83, 84].

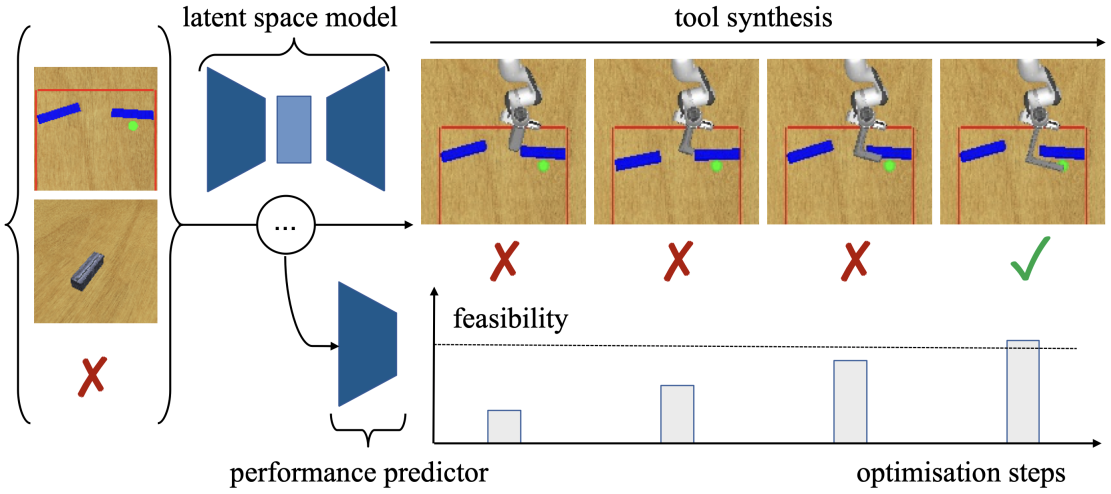


Figure 3.1: Tool synthesis for a reaching task. Our model is trained on data-triplets {task observation, tool observation, success indicator}. Within a scenario, the goal is to determine if a given tool can reach the goal (green) while avoiding barriers (blue) and remaining behind the boundary (red). If a tool cannot satisfy these constraints, our approach (via the performance predictor) *imagines* how one may augment it in order to solve the task. Our interest is in what these augmentations, imagined during *tool synthesis*, imply about the learned object representations.

In order to demonstrate that a task-aware latent space encodes useful affordance information we require a mechanism to train such a model as well as to purposefully explore the space. To this end we propose an architecture in which a task-based

performance predictor (a classifier) operates on the latent space of a generative model (see fig. 3.1). During training the classifier is used to provide an auxiliary objective, aiding in shaping the latent space. Importantly, however, during test time the performance predictor is used to guide exploration of the latent space via activation maximisation [85–87], thus explicitly exploiting the structure of the space. While our desire to affect factors of influence is similar in spirit to the notion of disentanglement, it contrasts significantly with models such as β-VAE [88], where the factors of influence are effectively encouraged to be axis-aligned. Our approach instead relies on a high-level auxiliary loss to discover the direction in latent space to explore.

Our experiments demonstrate that artificial agents are able to *imagine* an appropriate tool for a variety of reaching tasks by manipulating the tool's task-relevant affordances. To the best of our knowledge, this makes us the first to demonstrate an artificial agent's ability to imagine, or synthesise, 3D meshes of tools appropriate for a given task via optimisation in a structured latent embedding. Similarly, while activation maximisation has been used to visualise modified input images before [e.g. 89], we believe this work to be the first to effect deliberate manipulation of factors of influence by chaining the outcome of a task predictor to the latent space, and then decoding the latent representation back into a 3D mesh. Beyond the application of tool synthesis, we believe our work to provide novel perspectives on affordance learning and disentanglement in demonstrating that object affordances can be viewed as *trajectories* in a structured latent space as well as by providing a novel architecture adept at deliberately manipulating interpretable factors of influence.

3.2 Related Work

The concept of an *affordance*, which describes a potential action to be performed on an object (e.g. a doorknob *affords* being turned), goes back to [20]. Because of their importance in cognitive vision, affordances are extensively studied in computer vision and robotics. Commonly, affordances are learned in a supervised fashion where models discriminate between discrete affordance classes or predict masks for image regions which afford certain types of human interaction [e.g. 78, 79, 82, 90–92]. Interestingly, most works in this domain learn from object shapes which have been given an affordance label a priori. However, the affordance of a shape is only properly defined in the context of a task. Hence, we employ a task-driven traversal of a latent space to optimise the shape of a tool by exploiting factors of variation which are conducive to task success.

Recent advances in 3D shape generation employ variational models [93, 94] to capture complex manifolds of 3D objects. Besides their expressive capabilities, the latent spaces of such models also enable smooth interpolation between shapes. Remarkable results have been demonstrated including 'shape algebra' [94] and the preservation of object part semantics [95] and fine-grained shape styles [96] during interpolation. This shows the potential of disentangling meaningful factors of variation in the latent representation of 3D shapes. Inspired by this, we investigate whether these factors can be exposed in a task-driven way. In particular, we propose an architecture in which a generative model for 3D object reconstruction [21] is paired with activation maximisation [e.g. 85–87, 97] of a task-driven performance predictor. Guided by its loss signal, activation maximisation traverses the generative model's latent representations and drives an imagination process yielding a shape suitable for the task at hand.

A key application of affordance-driven shape imagination is tool use. Robotics boasts a mature body of literature studying how robots can utilise tools to improve their performance across a wide range of tasks like reaching [98], grasping [99], pushing [90] and hammering [100]. The pipeline executing tool-based tasks typically starts with models for tool recognition and selection [e.g. 78, 100–103] before

tool properties and affordances are leveraged to compute higher-order plans [104]. Our proposed model lends itself to robotics applications like these, as the learned latent space encodes a rich object-centric representation of tools that are biased for specific tasks.

3.3 Method

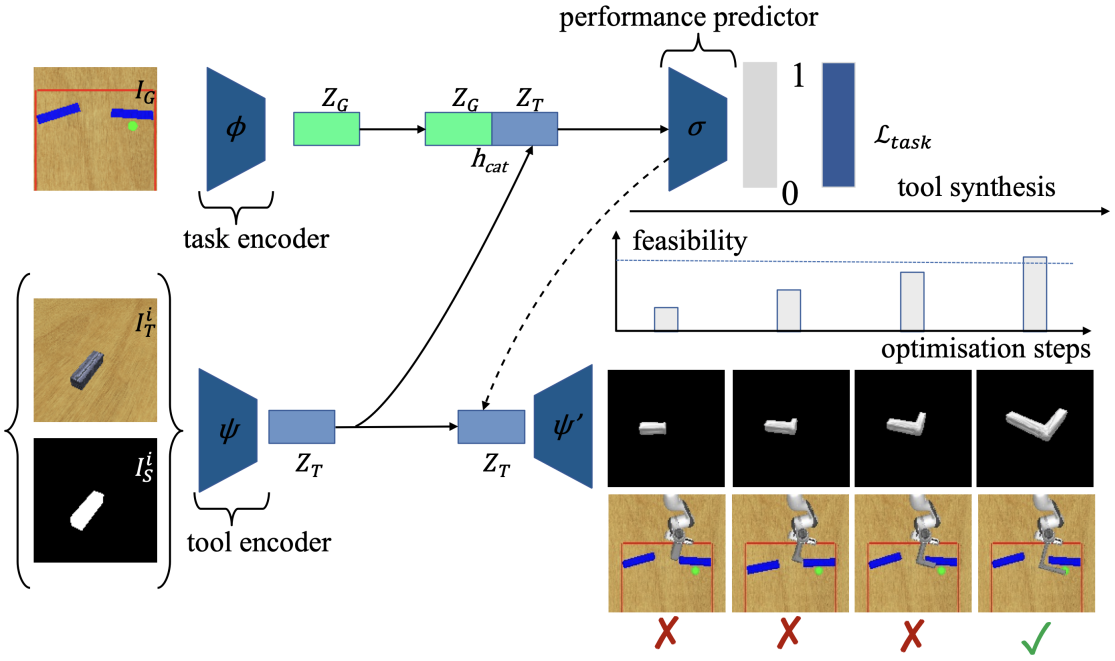


Figure 3.2: The model architecture. A convolutional encoder ϕ represents the task image I_G as a latent vector $\mathbf{z}_G$. In parallel, the 3D tool encoder ψ takes an input image I_T^i and its silhouette I_S^i and produces a latent representation $\mathbf{z}_T$. The concatenated task–tool representation $\mathbf{h}_{cat}$ is used by a classifier σ to estimate the success of the tool at solving the task (i.e. reaching the goal). Given the gradient signal from this performance predictor for success, the latent tool representation $\mathbf{z}_T$ gets updated to render an increasingly suitable tool (via the 3D tool decoder ψ'). We pretrained the encoding and decoding models (ψ, ψ') together as in prior work [105, 106].

Our overarching goal is to perform task-specific tool synthesis for 3D reaching tasks. We frame the challenge of tool imagination as an optimisation problem in a structured latent space obtained using a generative model. The optimisation is driven by a high-level, task-specific performance predictor, which assesses whether a target specified by a goal image I_G is reachable given a particular tool and in the presence of obstacles. To map from tool images into manipulable 3D tools, we first train an off-the-shelf 3D single-view reconstruction model taking as input tool images I_T^i, I_T^j and corresponding tool silhouettes I_S^i, I_S^j as rendered from two different vantage points i and j. After training, the encoder can infer the tool representation that contains the 3D structure information given a single-view RGB image and its silhouette as input. This representation is implicitly used to optimise

the tool configuration to make it suitable for the task at hand. An overview of our model is given in fig. 3.2.

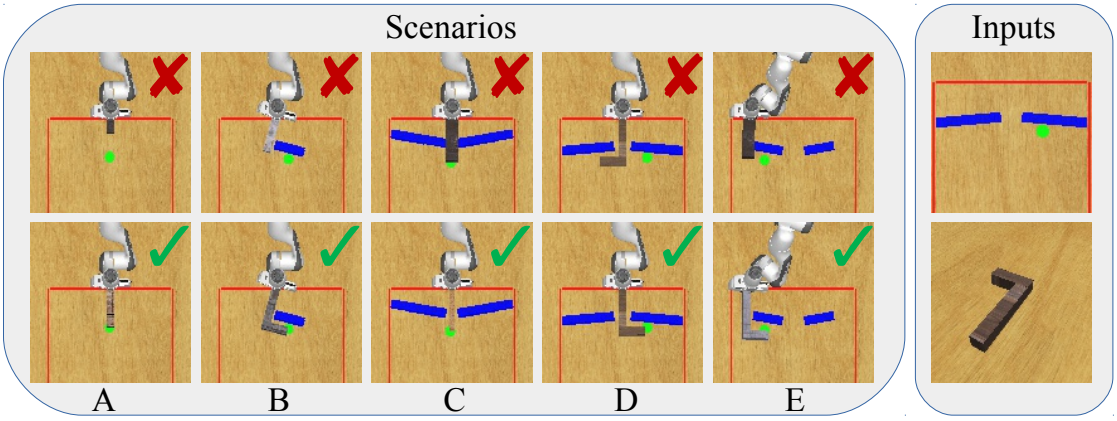


Figure 3.3: (Left) Task examples from our dataset. Top and bottom rows correspond to unsuccessful and successful tool examples respectively. Columns `A` - `E` represent five different task scenario types each imposing different tool constraints including width, length, orientation and shape. Note that the robot is fixed at its base on the table and constrained to remain outside the red boundary. Hence, it can only reach the green target with a tool while avoiding collisions with the blue obstacles. (Right) Model inputs {task observation, tool observation} during training and test time.

More formally, we consider N data instances: $\{(I_G^n, I_T^{n,i}, I_T^{n,j}, I_S^{n,i}, I_S^{n,j}, \rho^n)\}_{n=1}^N$, where each example features a task image I_G, tool images I_T in two randomly selected views i and j, and their corresponding silhouettes I_S, as well as a binary label ρ indicating the feasibility of reaching the target with the given tool. Examples of task images and model inputs are shown in fig. 3.3. In all our experiments, we restrict the training input to such sparse high-level instances. For additional details on the dataset, we refer the reader to the supplementary material.

3.3.1 Representing Tasks and Tools

Given that our tools are presented in tool images I_T, it is necessary for the first processing step to perform a 3D reconstruction of I_T, from pixels into meshes. To achieve this single view 3D reconstruction of images into their meshes, we employ the same architecture as proposed by [105, 106]. The 3D reconstruction model consists of two parts: an encoder network and a mesh decoder. Given the tool

image and its silhouette in view i, i.e I_T^i and I_S^i, we denote the latent variable encoding the tool computed by the encoder, ψ, as

$$\psi(I_T^i, I_S^i) = \mathbf{z}_T. \tag{3.1}$$

The mesh decoder takes $\mathbf{z}_T$ as input and synthesises the mesh by deforming a template. A differentiable renderer [21] predicts the tool's silhoutte $\hat{I}_S^j$ in another view j, which is compared to the ground-truth silhouette I_S^j to compute the silhouette loss $\mathcal{L}_s$. This silhouette loss $\mathcal{L}_s$ together with an auxiliary geometry loss $\mathcal{L}_g$ formulates the total 3D reconstruction loss:

$$\mathcal{L}_{recon} = \mathcal{L}_s + \mu\mathcal{L}_g, \tag{3.2}$$

where μ is the weight of the geometry loss. We refer the reader to [21] regarding the exact hyper-parameter and training setup of the 3D reconstruction model.

Task images I_G are similarly represented in an abstract latent space. For this we employ a task encoder, ϕ, which consists of a stack of convolutional layers.[1] ϕ takes the task image I_G as input and maps it into the task embedding $\mathbf{z}_G$.

3.3.2 Tool Imagination

Task-driven learning The tool representation $\mathbf{z}_T$ contains task-relevant information such as tool length, width, and shape. In order to perform tool imagination, the sub-manifold of the latent space that corresponds to the task-relevant features needs to be accessed and traversed. This is achieved by adding a three-layer MLP as a classifier σ. The classifier σ takes as input a concatenation $\mathbf{h}_{cat}$ of the task embedding $\mathbf{z}_G$ and the tool representation $\mathbf{z}_T$, and predicts the softmax over the binary task success. The classifier learns to identify the task-relevant sub-manifold of the latent space by using the sparse success signal ρ and optimising the binary-cross entropy loss, such that

$$\mathcal{L}_{task}\left(\sigma\left(\mathbf{h}_{cat}\right), \rho\right) = -\left(\rho \log\left(\sigma(\mathbf{h}_{cat})\right) + (1-\rho)\log\left(1-\sigma(\mathbf{h}_{cat})\right)\right), \tag{3.3}$$

[1] Architecture details are provided in the supplementary material.

where $\rho \in \{0, 1\}$ is a binary signal indicating whether or not it is feasible to solve the task with the given tool. The whole system is trained end-to-end with a loss given by

$$\mathcal{L}\left(I_G, I_T^i, I_S^i, I_T^j, I_S^j, \rho\right) = \mathcal{L}_{recon} + \mathcal{L}_{task}. \tag{3.4}$$

Note that the gradient from the task classifier σ propagates through both the task encoder ϕ and the toolkit encoder ψ, and therefore helps to shape the latent representations of the toolkit with respect to the requirements for task success.

Tool imagination Once trained, our model can synthesise new tools by traversing the latent manifold of individual tools following the trajectories that maximise classification success given a tool image and its silhouette (fig. 3.2). To do this, we first pick a tool candidate and concatenate its representation $\mathbf{z}_T$ with the task embedding $\mathbf{z}_G$. This warm-starts the imagination process. The concatenated embedding $\mathbf{h}_{cat}$ is then fed into the performance predictor σ to compute the gradient with respect to the tool embedding $\mathbf{z}_T$. We then use activation maximisation [85–87] to optimise $\mathbf{z}_T$ with regard to $\mathcal{L}_{task}$ of the success estimation $\sigma\left(\mathbf{h}_{cat}\right)$ and a feasibility target $\rho_s = 1$, such that

$$\mathbf{z}_T = \mathbf{z}_T + \eta \frac{\partial \mathcal{L}_{task}\left(\sigma\left(\mathbf{z}_T\right), \rho_s\right)}{\partial \mathbf{z}_T}, \tag{3.5}$$

where η denotes the learning rate for the update. Finally, we apply this gradient update for S steps or until the success estimation $\sigma\left(\mathbf{z}_T\right)$ reaches a threshold γ, and use $\psi'(\mathbf{z}_T)$ to generate the imagined 3D tool mesh represented by $\mathbf{z}_T$.

3.4 Experiments

In this section we investigate our model's abilities in two experiments. First, we verify the functionality of the task performance predictor σ in a *tool selection* experiment where only one out of three tools is successfully applicable. Second, we examine our core hypothesis about task-driven tool synthesis in a *tool imagination* experiment where the model has to modify a tool shape to be successfully applicable in a given task. In both experiments, we compare our full *task-driven* model, in which the tool latent space was trained jointly with the task performance predictor, with a *task-unaware* baseline, in which the 3D tool representation was trained first and the task performance predictor was fitted to the fixed tool latent space. We report our results in table 3.1 as mean success performances within a 95% confidence interval around the estimated mean.

Tool Selection We verify that the classifier σ correctly predicts whether or not a given tool can succeed at a chosen task. For each task, we create a toolkit containing three tool candidates where exactly one satisfies the scenario constraints. The toolkits are sampled in the same way as the remaining dataset and we refer the reader to fig. 3.3 again for illustrations of suitable and unsuitable tools. We check whether the classifier outputs the highest success probability for the suitable tool. Achieved accuracies for tool selection are reported in the left column of table 3.1.

Tool Imagination We evaluate whether our model can generate tools to succeed in the reaching tasks. For each instance the target signal for feasibility is set to $\rho_s = 1$, i.e. *success*. Then, the latent vector of the tool is modified via backpropagation using a learning rate of 0.01 for $10,000$ steps or until $\sigma(\mathbf{h}_{cat})$ reaches the threshold of $\gamma = 0.997$. The imagined tool mesh is generated via the mesh decoder ψ'. This is then rendered into a top-down view and evaluated using a feasibility test which checks whether all geometric constraints are satisfied, i.e. successful reaching from behind the workspace boundary while not colliding with any obstacle. We report the percentage of imagined tools that successfully pass this test in table 3.1.

3.4.1 Model Training

In order to gauge the influence of the task feasibility signal on the latent space of the tools, we train the model in two different setups. A *task-driven* model is trained with a curriculum: First, the 3D reconstruction module is trained on tool images alone. Then, the performance predictor is trained jointly with this backbone, i.e. the gradient from the predictor is allowed to back-propagate into the encoder of the 3D reconstruction network. In a *task-unaware* ablation, we keep the pre-trained 3D reconstruction weights fixed during the predictor training removing any influence of the task performance on the latent space. All models are trained for 15,000 steps in total. The first 10,000 steps are spent on pre-training the 3D reconstruction part in isolation and the remaining 5,000 steps are spent training the task performance predictor. We select checkpoints that have the lowest task loss $\mathcal{L}_{task}$ on the validation split.

3.4.2 Quantitative Results

We start our evaluation by examining the task performance predictor in the tool selection experiment (TS). For each scenario type (`A` - `E`) we present 250 tasks from the test set, each paired with three tool images and their respective silhouettes. We encode each tool with the tool encoder ψ and concatenate its representation $\mathbf{z}_T$ to $\mathbf{z}_G$ obtained from the task image encoder ϕ. Each concatenated pair h_{cat} is passed through the feasibility predictor σ and the tool which scores highest is selected. We

		Tool Selection Success [%]		Tool Imagination Success [%]		
Scn	N	Task-Unaware	Task-Driven	Random Walk	Task-Unaware	Task-Driven
`A`	250	88.8 ± 3.9	90.8 ± 3.6	3.6 ± 2.3	55.6 ± 6.2	**96.4 ± 2.3**
`B`	250	96.4 ± 2.3	97.6 ± 1.9	5.6 ± 2.9	42.0 ± 6.1	**78.8 ± 5.1**
`C`	250	96.4 ± 2.3	97.2 ± 2.1	23.6 ± 5.3	56.8 ± 6.1	**76.4 ± 5.3**
`D`	250	96.8 ± 2.2	98.4 ± 1.6	2.4 ± 1.9	**75.2 ± 5.4**	**81.2 ± 4.8**
`E`	250	87.2 ± 4.1	87.6 ± 4.1	13.6 ± 4.3	**88.4 ± 4.0**	**86.4 ± 4.3**
Tot	1250	93.1 ± 1.4	94.3 ± 1.3	9.8 ± 1.7	63.6 ± 2.7	**83.8 ± 2.0**

Table 3.1: (Left) Tool Selection: Mean accuracy when predicting most useful tool among three possible tools. (Right) Tool Imagination: Comparison of imagination processes when artificially warmstarting from the same unsuitable tools in each instance. Best results are highlighted in bold.

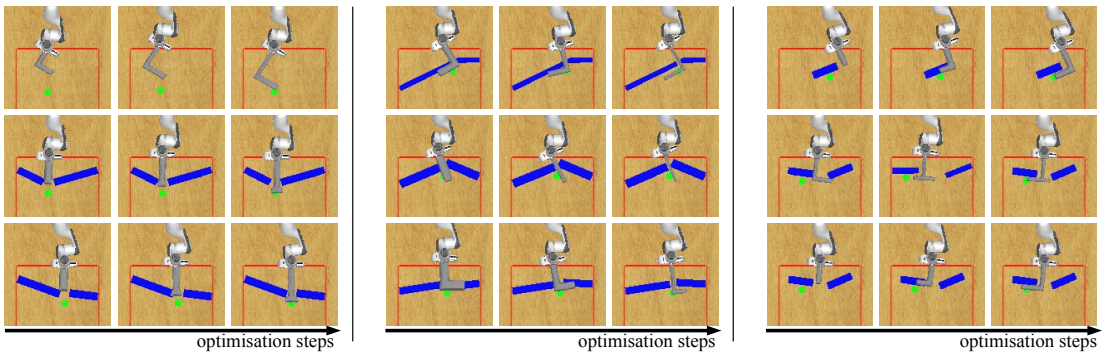


Figure 3.4: Qualitative results of tool evolution during the imagination process. Each row illustrates an example of how the imagination procedure can succeed at constructing tools that solve the task by: (left) increasing tool length, (middle) decreasing tool width, and (right) altering tool shape (creating an appropriately oriented hook). Each row in each grid represents a different imagination experiment.

report the results for this experiment in the left section of table 3.1. The results confirm that σ is able to identify suitable tools with almost perfect accuracy. Tool selection accuracy does not differ significantly between the task-driven and the task-unaware variations of the model. This suggests that the factors of tool shape variation are captured successfully in both cases and the feature vectors produced by ψ are discriminative enough to be separated efficiently via the MLP σ.

After verifying the task performance predictor's functionality in the tool selection experiment, we investigate its ability to drive a generative process in the next experiment. This is done to test our hypothesis about the nature and exploitability of the latent space. Given that the latent space captures factors of variation in 3D tool geometry, we hypothesise that these factors can be actively leveraged to synthesise a new tool by focusing on the performance predictor. Specifically, we present our model with a task image I_G and an unsuitable tool geometry. We then encode the tool via ψ and modify its latent representation $\mathbf{z}_T$ by performing activation maximisation through the performance predictor σ. As the feasibility prediction is pushed towards 1, the tool geometry gradually evolves into one that is applicable to the given task image I_G. In addition to comparing the imagination outcomes for the task-driven and the task-unaware model, we also include a *random walk* baseline, where, in place of taking steps in the direction of the steepest gradient,

we move in a *random direction* in the task-driven latent space for $10,000$ steps. In this baseline the latent vector of the selected tool is updated by a sample drawn from an isotropic Gaussian with mean 0, and, to match the step size of our approach, the absolute value of the ground-truth gradient derived by back-propagating from the predictor as the variance.

For 250 instances per scenario type, we warmstart each imagination attempt with the same infeasible tool across random walk, task-driven, and task-unaware models to enable a like-for-like comparison, with the results presented in table 3.1. The performance of the random walk baseline reveals that a simple stochastic exploration of the latent space is not sufficient to find suitable tool geometries. However, following the gradient derived from the performance predictor leads to successful shaping of tool geometries in a much more reliable way. While the task-unaware ablation provides a strong baseline, transforming tools successfully in 63.6% of the cases, the task-driven model significantly outperforms it, achieving a global success rate of 83.8% on the test cases. This implies that jointly training the 3D latent representation and task performance predictor significantly shapes the latent space in a 'task-aware' way, encoding properties which are conducive to task success (e.g. length, width, and configuration of a tool) along smooth trajectories. Moreover, each of these trajectories leads to higher *reachability* suggesting that these affordances can be seen as a set trajectories in a task-aware latent space.

3.4.3 Qualitative Results

Qualitative examples of the tool imagination process are provided in fig. 3.4 and fig. 3.5. In the right-middle example of fig. 3.4, a novel T-shape tool is created, suggesting that the model encodes the vertical stick-part and horizontal hook-part as distinct elements. The model also learns to interpolate the direction of the hook part between pointing left and right, which leads to a novel tool. As shown in fig. 3.5, tools are modified in a smooth manner, leading us to hypothesise that tools are embedded in a continuous manifold of changing length, width and configuration. Optimising the latent embedding for the highest performance predictor score often

drives the tools to evolve along these properties. This suggests that these geometric variables are encoded as *trajectories* in the structured latent space learnt by our model and deliberately traversed via a high-level task objective in the form of the performance predictor.

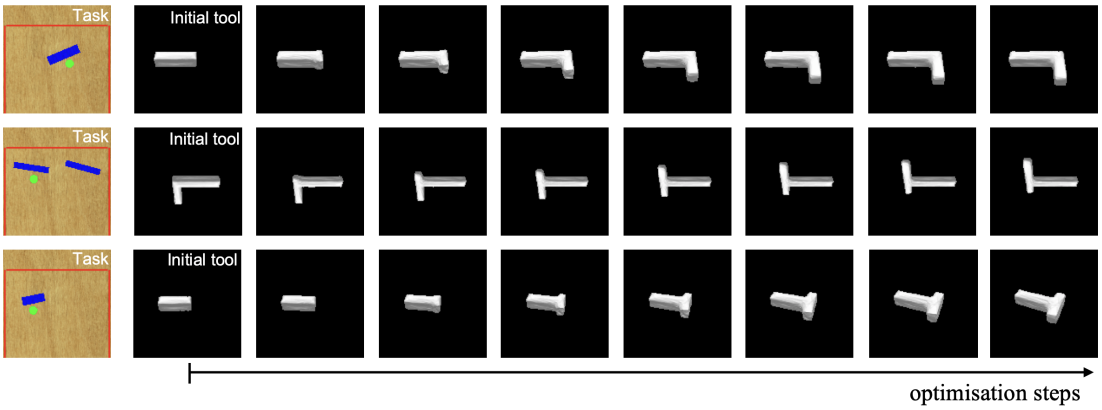


Figure 3.5: Examples of tool synthesis progression during the imagination process. In the top row, a stick tool morphs into a hook. The middle row shows a left-facing hook transforming into a right-facing hook. In the bottom row, the tool changes into a novel T-shape. Constraints on these optimisations are specified via task embeddings corresponding to the task images on the far left.

3.5 Conclusion

In this chapter we investigated the ability of an agent to synthesise tools via task-driven imagination within a set of simulated reaching tasks. Our approach explores a hybrid architecture in which a high-level performance predictor drives an optimisation process in a structured latent space. The resulting model successfully generates tools for unseen scenario types not in the training regime; it also learns to modify interpretable properties of tools such as length, width, and shape. Our experimental results suggest that these object affordances are encoded as *trajectories* in a learnt latent space, which we can navigate through in a deliberate way using a task predictor and activation maximisation, and interpret by decoding the updated latent representations. Ultimately, this may aid in our understanding of object affordances while offering a novel way to disentangle interpretable factors of variation – not only for 3D tool synthesis. To facilitate further work in this area, we plan to release both the reaching dataset and trained model to the community.

3.6 Dataset for the Reaching Tasks

To investigate tool imagination, we designed a set of simulated reaching tasks with clear and controllable factors of influence. Each task image is comprised of a green target button, three red lines delineating the workspace area, and, optionally, a set of blue obstacles. We also vary the goal location and the sizes and positions of the obstacles. For each task image, we provide a second image depicting a tool, i.e. a straight stick or an L-shaped hook, with varying dimensions, shapes, and textures. Given a pair of images (i.e. the task image and the tool image), the goal is to predict whether the tool for a given task scene can reach the target (the green dot) whilst avoiding obstacles (blue areas) and remaining on the exterior of the workspace (i.e. behind the red line). Depending on the task image, the applicability of a tool is determined by different subsets of its attributes. For example, if the target button is unobstructed, then any tool of sufficient length will satisfy the constraints (regardless of its width or shape). However, when the target is hidden behind a corner, or only accessible through a narrow gap, an appropriate tool also needs to feature a long-enough hook, or a thin-enough handle, respectively. As depicted in Figure 3.3, we have designed five scenario types to study these factors of influence in isolation and in combination. The task setting and tool are first rendered in a top-down view for the geometric applicability check. Then the tool is rendered in 12 different views for the 3D reconstruction. The geometric applicability check verifies whether or not any of the tools can reach the target, while satisfying the task constraints.

3.7 Dataset Construction Minutiae

Our synthetic dataset consists of pairs of task image (in top-down view) and tool image as well as the corresponding silhouettes in 12 different views. We also record the tool image from top-down view for the tool-applicability check. All images are 128×128 pixels. The applicability of a tool to a task is determined by sampling 100 interior points of the tool polygon, overlaying the sampled point with the target and rotating the tool polygon between -60 degrees and +60 degrees from

the y-axis. If any such pose of the tool satisfies all constraints (i.e. touching the space behind the red line while not colliding with any obstacle), we consider the tool applicable for the given task. All train- and test-cases are constrained to have only sticks and hooks as tools.

The dataset contains a total of 18,500 scenarios. Table 3.2 shows a breakdown of each by scenario type and split. Each split has an equal number of feasible and infeasible instances, for each scenario type.

Type	Training	Validation
`A`	4,000	500
`B`	4,000	500
`C`	4,000	500
`D`	4,000	500
`E`	-	500
Total	16,000	2,500

Table 3.2: Number of instances by scenario type.

3.8 Architecture and Training Details

As described in the main text, our model comprises two main parts (as depicted in fig. 3.2). On the one hand, we used a task-based classifier, while on the other we used a encoder-decoder model as a single-view 3D reconstruction model. We briefly describe each component in turn. As a 3D reconstruction model, we faithfully re-implement the encoder-decoder architecture identical to [21, 105]. A high-level model schematic is shown in fig. 3.6. The task-based classifier consists of two sub-parts: a task encoder and a classifier. The task encoder stacks five convolutional blocks followed by a single convolutional layer. First, the five consecutive convolutional blocks make use of 16, 32, 64, 64, 128 output channels respectively. Each convolutional block contains two consecutive sub-convolutional blocks, each with a kernel size of 5 and padding of 2 (and with stride of 1 and stride of 2, respectively). Second, the additional convolutional layer has kernel size 4, stride 1, padding 0, and 256 output channels. All convolutional layers use an ELU nonlinearity. The classifier is a three layer MLP with input dimension 768, and

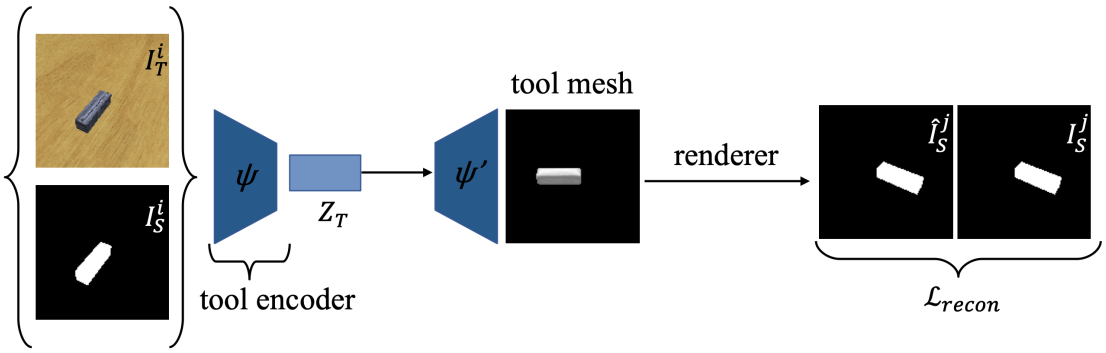


Figure 3.6: Model for 3D reconstruction.

successive hidden layers of 1024 and 512 neurons respectively. Each of these layers use eLU activation functions. Given that the classifier outputs logits for a binary classifier, the output dimension is 2. All experiments are performed in PyTorch, using the ADAM optimiser with a learning rate of 0.0001 and a batch size of 16.

4

APEX: Unsupervised, Object-Centric Scene Segmentation and Tracking for Robot Manipulation

Recent advances in unsupervised learning for object detection, segmentation, and tracking hold significant promise for applications in robotics. A common approach is to frame these tasks as inference in probabilistic latent-variable models. In this chapter, we investigate whether the unsupervised OCGM can learn meaningful object-like entities in robotic scenarios where frequent interactions between the objects and the robot exist in the scene. We propose APEX, a new latent-variable model which is able to segment and track objects in more realistic scenes featuring objects that vary widely in size and texture, including the robot arm itself. This is achieved by a principled mask normalisation algorithm and a high-resolution scene encoder.To evaluate our approach, we present results on the real-world Sketchy dataset. This dataset, however, does not contain ground truth masks and object IDs for a quantitative evaluation. We thus introduce the Panda Pushing Dataset (P2D) which shows a Panda arm interacting with objects on a table in simulation and which includes ground-truth segmentation masks and object IDs for tracking. In both cases, APEX comprehensively outperforms the current state-of-the-art in unsupervised object segmentation and tracking. We demonstrate the efficacy of our segmentations

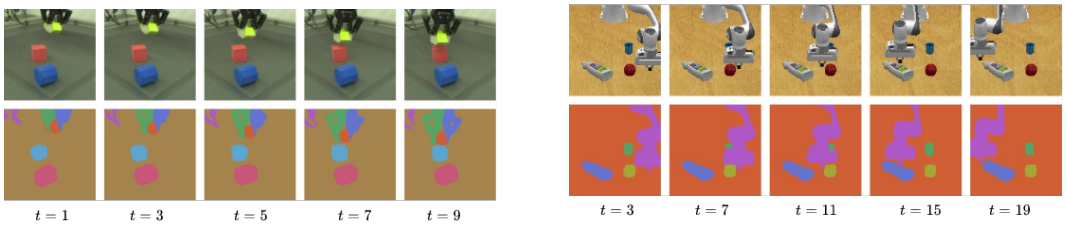
for robot skill execution on an object arrangement task, where we also achieve the best or comparable performance among all the baselines. This work is published as:

Y. Wu, O. Parker Jones, M. Engelcke, and I. Posner. "APEX: Unsupervised, object-centric scene segmentation and tracking for robot manipulation." In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

4.1 Introduction

Scene segmentation and tracking are cornerstones of robotics (e.g. [107, 108]). A principal motivation is the ability to ground sensory observations in state-representations suitable for executing downstream tasks. While considerable advances have been made in using supervised methods for detecting and segmenting objects (e.g. [3, 109]), labelling data is resource intensive and quickly becomes intractable when trying to consider every possible object category for every deployment eventuality. Unsupervised learning – the discovery of representations suitable for task execution without the need for training labels – is therefore emerging as a promising alternative. In particular, recently developed *object-centric* scene representations (e.g. [10, 12, 59, 61, 110, 111]) have the potential of vastly improving data efficiency in robotics and other applications (e.g. [66, 112, 113]). Here we show, however, that current state-of-the-art methods fail on visually complex datasets that are representative of scenarios commonly encountered in robot manipulation.

Within the class of unsupervised, object-centric models, variational autoencoders (VAEs) ([33, 114]) are emerging as a popular choice. Object detection and segmentation are performed by using spatial transformer networks (STNs) [40] and spatial Gaussian mixture models (SGMMs) ([115–117]) to separate objects, respectively. Some works do not impose any further structure on these latent representations (e.g. [10, 42, 66, 110]), while others further factorise the representations to explicitly disentangle foreground objects from the background, object location, appearance, and whether an object slot is used or not (e.g. [11, 59, 111, 118]).In contrast to, e.g., OP3 [66] which was also developed in the context of robotics, we argue the latter is of particular utility in robotics applications where such highly structured representations can be directly used as inputs to a planning module. A recently proposed model of this type called SCALOR [111] achieves particularly good results on segmenting and tracking objects in videos that contain a possibly large number of objects. We demonstrate however, that SCALOR struggles to learn object-centric representations on datasets with objects of widely varying sizes and textures as encountered in robot manipulation.



(a) Sketchy **(b)** Panda Pushing Dataset (P2D)

Figure 4.1: APEX learns to segment and track objects in videos *without supervision* from (a) the established Sketchy dataset [29] and (b) our Panda Pushing Dataset (P2D). Segment colour indicates object ID. APEX accurately segments the diverse objects in the scenes and tracks the blue cup in (b) despite severe occlusion at $t = \{7, 11\}$.

We therefore develop APEX (Amortised Parallel infErence with miXture models), a novel object-centric generative model trained on videos. In contrast to prior work, APEX uses a principled mask normalisation procedure to parameterise an SGMM, which allows the explicit tuning of foreground and background standard deviations. APEX also features an improved scene encoder that outputs a high-resolution feature map that is particularly suitable for tracking [119]. To showcase the efficacy of APEX, we evaluate APEX qualitatively on Sketchy [29], an existing real-world robot manipulation dataset. We also introduce the Panda Pushing Dataset (P2D) as a quantitative benchmark against prior art. This contains videos of a Panda arm interacting with objects on a table in simulation and includes ground truth labels for segmentation masks and tracking objects between frames. It can be observed that APEX comprehensively outperforms recent state-of-the-art methods ([27, 42, 66, 111, 120]) by a large margin in terms of unsupervised segmentation and object tracking. Finally, inspired by the *Open Cloud Robot Table Organization Challenge* [121], we demonstrate the utility of APEX specifically in the context of a robot object re-arrangement task. Here, the superior segmentation performance of APEX, as illustrated in fig. 4.1, as well as the quality of the learned object representations lead to significantly better results compared to prior art.

4.2 Related Work

This work builds on recent literature on object-centric generative models (OCGMs), which are typically formulated as VAEs (e.g. [11, 118]) or generative adversarial networks (GANs) (e.g. [122–124]).Unlike object-centric GANs, VAE-based methods directly provide an amortised inference mechanism for extracting object-centric representations from input images. Early works that use STNs for separating objects do so by sequentially attending to different regions in an image, leading to a computational complexity that increases linearly with the number of objects ([11, 118]). More recent works parallelise the inference of object representations, which has been shown to be particularly useful for images with a large number of objects ([59, 111]). Segmentation-based models that parameterise an SGMM (e.g. [10, 42, 61, 110]) tend to be computationally more expensive than STN-based approaches where objects can be generated as smaller crops rather than image-sized components. A more informative pixel-level labelling, however, is required in applications such as the object arrangement task considered in this work. SLOT-ATTENTION [42] is a prominent recent model belonging to this category and is therefore selected as one of our baselines. The majority of related works learn object representations from individual images (e.g. [10, 42, 110]), but some also exploit temporal information in video sequences (e.g. [12, 66, 111]) to improve object separation. APEX also leverages a VAE, STNs, and an SGMM for the parallel inference of structured, object-centric latent representations from videos. The structure of the model is most closely related to SCALOR [111]. Instead of parameterising a Gaussian image likelihood, however, APEX parameterises an SGMM which allows direct tuning of loss magnitudes from background and foreground modules. Moreover, APEX uses a scene encoder that is particularly suitable for object tracking [125].

A small number of works also explore the use of object-centric representations in robotics ([66, 112]). OP3 [66] uses an object-centric model to predict goal images that contain a set of blocks in a desired configuration and the authors in [112] show that explicit object representations can accelerate the acquisition of robotic manipulation skills. Similarly, COBRA [113] leverages object-centric representations

to improve data efficiency and policy robustness in several RL tasks in visually simple simulated environments. Inspired by [121], we benchmark a number of models on an object manipulation task, where we observe that segmentations and object representations learned by APEX lead to significantly better results compared to the recent state-of-the-art.

4.3 APEX: Amortised Parallel Inference with Mixture Models

Let $\mathbf{x} \in [0,1]^{H \times W \times C}$ be a frame from a video sequence $\mathbf{x}_{1:T}$, where H and W denote the image height and width, C represents the number of image channels (e.g. RGB), and T denotes the number of frames in the video sequence. Consider a scene $\mathcal{S}$ to be formed of K object hypotheses – or *components* – which are each encoded by a set of latent variables such that $\mathcal{S} = \{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$. In particular, we consider two disjoint sets of object hypotheses comprised of *foreground* objects and a *background* component, such that $\mathcal{S} = \mathcal{O}^{\text{fg}} \cup \mathcal{O}^{\text{bg}}$, where $\mathcal{O}^{\text{fg}} = \{\mathbf{z}_1^{\text{fg}}, \ldots, \mathbf{z}_{K-1}^{\text{fg}}\}$ and $\mathcal{O}^{\text{bg}} = \{\mathbf{z}^{\text{bg}}\}$, i.e., $\mathbf{z}^{\text{bg}} = \mathbf{z}_K$. Each foreground component is described by a set of latent variables $\mathbf{z}_k^{\text{fg}} = \{\mathbf{z}_k^{\text{where}}, \mathbf{z}_k^{\text{what}}, z_k^{\text{pres}}\}$ encoding its location, appearance, and existence in the scene (see [11]). $\mathbf{z}^{\text{bg}}$ directly encodes the appearance of the background. For each frame $\mathbf{x}_t$ at time t, foreground components can either be *propagated* from the previous time step or *discovered* in the current image (see [12, 111]).

APEX defines a generative model with learnable parameters θ that is formulated as an SGMM (e.g. [10, 61, 110]) via the image likelihood

$$p_\theta(\mathbf{x}_t|\mathbf{z}_{1:K,t}) = \sum_{k=1}^{K} \mathbf{m}_k(\mathbf{z}_{1:K,t}) \odot \mathcal{N}(\mu(\mathbf{z}_{k,t}), \sigma_k), \tag{4.1}$$

where $\mathbf{m}_{k,t}$ are the segmentation masks, $\mu_{k,t}$ are the means of the Gaussian components, and σ_k is a fixed component standard deviation. Separate standard deviations are used for the foreground and background, i.e. $\sigma_{1:K-1} = \sigma^{\text{fg}}$ and $\sigma_K = \sigma^{\text{bg}}$. A probabilistic prior $p_\theta(\mathbf{z}_{1:K,1:T})$ is defined to regularise the model.

To achieve segmentation and tracking, APEX provides an approximate inference model [35] with learnable parameters ϕ for an object-centric latent representation of how a scene evolves in a sequence of images, i.e.,

$$q_\phi(\mathbf{z}_{1:K,1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_{1:K,t}|\mathbf{z}_{1:K,<t}, \mathbf{x}_{\leq t})\,. \tag{4.2}$$

The inference and generative models are learned jointly as a VAE ([33, 114]). An overview of APEX is illustrated in fig. 4.2. This section proceeds by first defining the

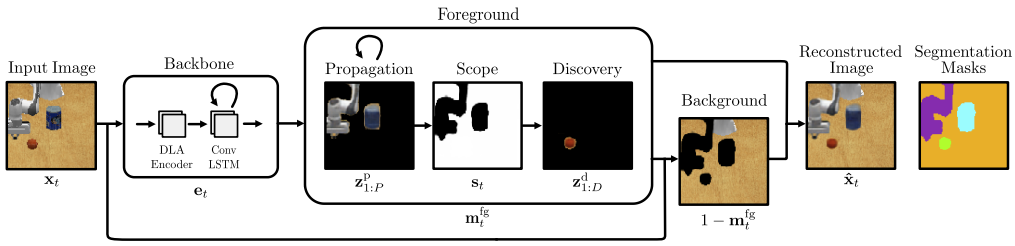


Figure 4.2: Illustration of APEX. A backbone first extracts an encoding from an input image. This is used to propagate objects from the previous time step and to discover additional objects in the current image. The remainder of the image is treated as background. The final image is composed from both the foreground objects and the background.

underlying structure of the generative model, before defining the associated inference model that facilitates the segmentation and tracking of objects in video sequences.

4.3.1 Generative Model

To capture correlations between frames, we assume the latents $\mathbf{z}_{1:K,t}$ depend on the latents of the previous time steps

$$p_\theta(\mathbf{z}_{1:K,1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_{1:K,t}|\mathbf{z}_{1:K,<t}) \,. \tag{4.3}$$

$p_\theta(\mathbf{z}_{1:K,t})$ is further factorised into three terms to represent *propagated* objects O_t^p, *discovered* objects O_t^d, and the background O_t^{bg}. Assuming there are P propagated objects and D newly discovered objects, we assume propagated objects depend on the previous latent variables, newly discovered objects in turn depend on objects that have already been propagated to avoid the *rediscovery* of propagated objects

$$p_\theta(\mathbf{z}_{1:K,t}|\mathbf{z}_{1:K,<t}) = p_\theta(\mathbf{z}_t^{\text{bg}}|\mathbf{z}_{<t}^{\text{bg}})\, p_\theta(\mathbf{z}_{1:D,t}^{\text{d}}|\mathbf{z}_{1:P,t}^{\text{p}})\, p_\theta(\mathbf{z}_{1:P,t}^{\text{p}}|\mathbf{z}_{1:K,<t}) \,. \tag{4.4}$$

To facilitate application in robotics and similar to prior works (e.g. [12, 111]), the latents describing foreground objects in O^{fg} are factorised into $\{\mathbf{z}_k^{\text{where}}, \mathbf{z}_k^{\text{what}}, z_k^{\text{pres}}\}$ describing component location, appearance, and presence in the scene, respectively, such that

$$p_\theta(\mathbf{z}_{k,t}^{\text{p}}|\mathbf{z}_{k,<t}^{\text{fg}}) = p_\theta(z_{k,t}^{\text{pres}}|\mathbf{z}_{k,<t}^{\text{fg}})\, p_\theta(\mathbf{z}_{k,t}^{\text{what}}|\mathbf{z}_{k,<t}^{\text{fg}})^{z_{k,t}^{\text{pres}}}\, p_\theta(\mathbf{z}_{k,t}^{\text{where}}|\mathbf{z}_{k,<t}^{\text{fg}})^{z_{k,t}^{\text{pres}}} \,, \tag{4.5}$$

$$p_\theta(\mathbf{z}_{k,t}^{\text{d}}|\mathbf{z}_{1:P,t}^{\text{p}}) = p_\theta(z_{k,t}^{\text{pres}}|\mathbf{z}_{1:P,t}^{\text{p}})\, p_\theta(\mathbf{z}_{k,t}^{\text{what}})^{z_{k,t}^{\text{pres}}}\, p_\theta(\mathbf{z}_{k,t}^{\text{where}})^{z_{k,t}^{\text{pres}}} \,. \tag{4.6}$$

The segmentation masks $\mathbf{m}_k$ and the means $\boldsymbol{\mu}$ of the Gaussian components in eq. (4.1) are decoded from $\{\mathbf{z}^{\text{where}}_{k,t}, \mathbf{z}^{\text{what}}_{k,t}, z^{\text{pres}}_{k,t}\}$ and $\mathbf{z}^{\text{bg}}_t$. We consider $z^{\text{pres}}_{k,t}$ to be Bernoulli distributed and the remainder of the latents being Gaussian distributed (see [27, 111]). $\mathbf{z}^{\text{what}}_{k,t}$ encodes the appearance of a component and the logits $\alpha_{k,t}$ from which the mask $\mathbf{m}_{k,t}$ is later obtained. $\mathbf{z}^{\text{bg}}_t$ only encodes the appearance of the background component. $\mathbf{z}^{\text{where}}_{k,t}$ encodes a transformation in an STN [40], describing the size and location of a bounding box that contains an object. A separate variable for encoding background location is not needed as every pixel that is not assigned to a foreground object is treated as background.

In contrast to SCALOR, where the foreground mask needs to be explicitly clamped to $[0, 1]$, we employ a principled approach to mask normalisation. In particular, we first introduce a foreground mask $\mathbf{m}^{\text{fg}}_t$ to model the occupancy of the foreground as a whole, and then compute intermediate object masks $\hat{\mathbf{m}}_{1:K-1,t}$ which attribute specific occupancy responsibility to each individual object. Specifically, the mask logits $\alpha_{1:K-1,t}$ for the $K-1$ foreground objects are computed as

$$\alpha_{k,t} = c \, \tanh\left(\text{CNN}(\mathbf{z}^{\text{what}}_{k,t})\right) , \tag{4.7}$$

where c is a fixed constant that constrains the mask logits to $[-c\,, c]$. The foreground mask $\mathbf{m}^{\text{fg}}_t$ is computed as

$$\mathbf{m}^{\text{fg}}_t = \tanh\left(\sum_{k=1}^{K-1} \text{STN}\left(\text{softplus}(\alpha_{k,t})\, z^{\text{pres}}_{k,t}, \mathbf{z}^{\text{where}}_{k,t}\right)\right) , \tag{4.8}$$

where the softplus operation ensures that a possible foreground component makes a non-negative contribution to the foreground mask when $z^{\text{pres}}_{k,t} = 1$. The intermediate object masks $\hat{\mathbf{m}}_{1:K-1,t}$ are obtained such that

$$\hat{\alpha}_{k,t} = \text{STN}\left(\alpha_{k,t}, \mathbf{z}^{\text{where}}_{k,t}\right) + 2c z^{\text{pres}}_{k,t} , \tag{4.9}$$

$$\hat{\mathbf{m}}_{1:K-1,t} = \text{softmax}\left(\hat{\alpha}_{1,t}, \ldots, \hat{\alpha}_{K-1,t}\right) . \tag{4.10}$$

eq. (4.9) maps components with $z^{\text{pres}}_{k,t} = 1$ to an interval that does not overlap with components where $z^{\text{pres}}_{k,t} = 0$. This formalises the intuition that objects which are *not present* will not occupy any physical space. The masks for the $K-1$ foreground

objects are then obtained by the element-wise multiplication of the foreground mask $\mathbf{m}_t^{\text{fg}}$ with intermediate object masks $\hat{\mathbf{m}}_{1:K-1,t}$,

$$\mathbf{m}_{1:K-1,t} = \hat{\mathbf{m}}_{1:K-1,t} \odot \mathbf{m}_t^{\text{fg}} . \tag{4.11}$$

Finally, inspired by the stick-breaking process formulations in prior work ([10, 110]), the background mask $\mathbf{m}_t^{\text{bg}}$ is computed as

$$\mathbf{m}_t^{\text{bg}} = 1 - \mathbf{m}_t^{\text{fg}} . \tag{4.12}$$

4.3.2 Inference Model

The true posterior over latent variables is generally intractable, so a variational approximation $q_\phi(\mathbf{z}_{1:K,1:T}|\mathbf{x}_{1:T})$ is introduced. Mirroring eq. (4.3) in the generative model, the approximate posterior is factorised as

$$q_\phi(\mathbf{z}_{1:K,1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_{1:K,t}|\mathbf{z}_{1:K,<t}, \mathbf{x}_{\leq t}) . \tag{4.13}$$

The posterior at time step t exhibits the same dependencies as were assumed in eq. (4.4) and can thus be written as

$$\begin{aligned} q_\phi(\mathbf{z}_{1:K,t}|\mathbf{z}_{1:K,<t}, \mathbf{x}_{\leq t}) = q_\phi(\mathbf{z}_t^{\text{bg}}|\mathbf{z}_{1:D,t}^{\text{d}}, \mathbf{z}_{1:P,t}^{\text{p}}, \mathbf{x}_t) \\ q_\phi(\mathbf{z}_{1:D,t}^{\text{d}}|\mathbf{z}_{1:P,t}^{\text{p}}, \mathbf{x}_{\leq t})\, q_\phi(\mathbf{z}_{1:P,t}^{\text{p}}|\mathbf{z}_{1:K,<t}, \mathbf{x}_{\leq t}) . \end{aligned} \tag{4.14}$$

We assume that the approximate posterior for the k^{th} propagated object in O_t^p factorises such that

$$\begin{aligned} q_\phi(\mathbf{z}_{k,t}^{\text{p}}|\mathbf{z}_{k,<t}^{\text{fg}}, \mathbf{x}_{\leq t}) = \\ q_\phi(z_{k,t}^{\text{pres}}|\mathbf{z}_{k,\leq t}^{\text{what}}, z_{k,<t}^{\text{pres}}, \mathbf{z}_{k,t-1}^{\text{where}}, \mathbf{x}_{\leq t}) \\ q_\phi(\mathbf{z}_{k,t}^{\text{what}}|\mathbf{z}_{k,t}^{\text{where}}, \mathbf{z}_{k,<t}^{\text{what}}, \mathbf{x}_{\leq t})^{z_{k,t}^{\text{pres}}} \\ q_\phi(\mathbf{z}_{k,t}^{\text{where}}|\mathbf{z}_{k,<t}^{\text{where}}, \mathbf{z}_{k,<t}^{\text{what}}, \mathbf{x}_{\leq t})^{z_{k,t}^{\text{pres}}} \end{aligned} \tag{4.15}$$

and that the posterior for a discovered object factorises as

$$\begin{aligned} q_\phi(\mathbf{z}_{k,t}^{\text{d}}|\mathbf{z}_{1:P,t}^{\text{p}}, \mathbf{x}_{\leq t}) = q_\phi(z_{k,t}^{\text{pres}}|\mathbf{z}_{k,t}^{\text{what}}, \mathbf{z}_{k,t}^{\text{where}}, \mathbf{z}_{1:P,t}^{\text{p}}, \mathbf{x}_{\leq t}) \\ q_\phi(\mathbf{z}_{k,t}^{\text{what}}|\mathbf{z}_{k,t}^{\text{where}}, \mathbf{x}_{\leq t})^{z_{k,t}^{\text{pres}}}\, q_\phi(\mathbf{z}_{k,t}^{\text{where}}|\mathbf{x}_{\leq t})^{z_{k,t}^{\text{pres}}} . \end{aligned} \tag{4.16}$$

Intuitively, $\mathbf{z}_{k,t}^{\text{where}}$ encodes where to look in an image in order to infer $\mathbf{z}_{k,t}^{\text{what}}$ which is used for determining the appearance of an object.

4.3.3 Inference - Implementation

We now proceed with describing the implementation of how these posterior distributions are inferred for object *propagation* and *discovery* as well as the *background*.

Feature Extraction Features are extracted with a shared encoder. Observations $\mathbf{x}_{\leq t}$ are stacked and encoded into a feature map $\mathbf{e}_t \in \mathbb{R}^{H/4 \times W/4 \times F}$ with a deep layer aggregation (DLA) encoder [126] and a ConvLSTM [38] to capture spatio-temporal correlations. F is the number of the feature channels. The encoder outputs a high-resolution feature map that preserves spatial correspondences between the features and input images. This is beneficial for object tracking [119] and in contrast to SCALOR, where a low resolution feature map is used instead. *Propagation:* Using the previous object bounding box at $t-1$, a square feature map $\mathbf{f}^{\mathrm{p}}_{k,t}$ is extracted from $\mathbf{e}_t$ using an STN, whereby the square dimensions are set equal to the larger side of the previous bounding box. A CNN then maps $\mathbf{f}^{\mathrm{p}}_{k,t}$ to a feature vector $\mathbf{c}^{\mathrm{p}}_{k,t}$. *Discovery:* A grid of equally spaced features $\mathbf{c}^{\mathrm{d}}_{1:D,t}$ is extracted from the feature map $\mathbf{e}_t$ to discover new objects.

Object Location *Propagation:* The posterior over $\mathbf{z}^{where}_{k,t}$ is computed with an RNN whose inputs are $[\mathbf{c}^{\mathrm{p}}_{k,t}, \mathbf{z}^{\mathrm{what}}_{k,t-1}, \mathbf{z}^{\mathrm{where}}_{k,t-1}]$. *Discovery:* The posterior over $\mathbf{z}^{\mathrm{where}}_{k,t}$ is computed from $\mathbf{c}^{\mathrm{d}}_{k,t}$ with a 1×1 convolution.

Object Appearance *Propagation:* A *glimpse* $\mathbf{G}_{k,t}$ is cropped from the feature map $\mathbf{e}_t$ using a STN according to $\mathbf{z}^{\mathrm{where}}_{k,t}$. A RNN takes the encoding of the glimpse $\mathbf{G}_{k,t}$ encoded by a CNN and $\mathbf{z}^{\mathrm{what}}_{k,t-1}$ as inputs to infer the posterior over $\mathbf{z}^{\mathrm{what}}_{k,t}$. *Discovery:* The posterior over $\mathbf{z}^{\mathrm{what}}_{k,t}$ is inferred in same fashion as for propagated objects with weights being shared and $\mathbf{z}^{\mathrm{what}}_{0}$ being initialised to a vector of zeros. *Background:* The posterior over $\mathbf{z}^{\mathrm{bg}}_{t}$ is obtained with a CNN using the background mask $\mathbf{m}^{\mathrm{bg}}_{t}$ and the current image $\mathbf{x}_t$.

Object Presence *Propagation:* The posterior over $z^{\mathrm{pres}}_{k,t}$ is computed with an RNN whose inputs are $[\mathbf{c}^{\mathrm{p}}_{k,t}, \mathbf{z}^{\mathrm{what}}_{k,t}, z^{\mathrm{pres}}_{k,t-1}]$. *Discovery:* The presence of objects in the *discovery* phase is determined by: 1. whether a new object is detected and 2. whether that object has been explained by the propagated objects. We use the scope $\mathbf{s}_t$ to indicate which pixels have not been explained yet. This scope is

defined as $\mathbf{s}_t = 1 - \mathbf{m}_t^{\mathrm{p}}$ whereby $\mathbf{m}_t^{\mathrm{p}}$ is computed as in eq. (4.8) but only using propagated objects. We thus decompose the posterior over $z_{k,t}^{\mathrm{pres}}$ into two terms $\{p_{k,t}^{\mathrm{proposal}}, p_{k,t}^{\mathrm{context}}\} \in [0,1]$ so that

$$q_\phi(z_{k,t}^{\mathrm{pres}} | \mathbf{z}_{k,t}^{\mathrm{what}}, \mathbf{z}_{k,t}^{\mathrm{where}}, \mathbf{z}_{1:P,t}^{\mathrm{p}}, \mathbf{x}_{\leq t}) = p_{k,t}^{\mathrm{proposal}}\, p_{k,t}^{\mathrm{context}} . \tag{4.17}$$

$p_{k,t}^{\mathrm{proposal}}$ is obtained with a fully-connected layer from $[\mathbf{c}_{k,t}^{\mathrm{d}}, \mathbf{z}_{k,t}^{\mathrm{what}}, \mathbf{z}_{k,t}^{\mathrm{where}}]$ and describes whether a cell might contain a new object or not. The purpose of $p_{k,t}^{\mathrm{context}}$ is to avoid the rediscovery of objects via the scope $\mathbf{s}_t$. This removes discovered objects that overlap with propagated objects and it is computed as

$$\mathbf{p}_{k,t}^{\mathrm{context}} = \frac{\sum_{i,j} \mathbf{s}_{t,i,j} \ \tanh(\mathrm{softplus}(\alpha_{k,t,i,j}))}{\sum_{i,j} \tanh(\mathrm{softplus}(\alpha_{k,t,i,j}))} , \tag{4.18}$$

where (i, j) are all pixel coordinate tuples in an image. Intuitively, $\mathbf{p}_{k,t}^{\mathrm{context}}$ corresponds to the fraction of the proposed object mask that has not been explained by the propagated objects. *Object Filtering:* To reduce memory requirements, foreground objects are discarded at every time step when $z_{k,t}^{\mathrm{pres}}$ is below a fixed, manually set threshold.

4.3.4 Learning

The inference and generative models can be jointly trained by maximising the evidence lower bound (ELBO). Omitting object subscripts, this is given by

$$\mathcal{L}(\theta, \phi) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_t | \mathbf{z}_{<t}, \mathbf{x}_{\leq t})} \left[\log p_\theta(\mathbf{x}_t | \mathbf{z}_t) \right] \tag{4.19}$$

$$+ \mathbb{KL} \left[q_\phi(\mathbf{z}_t | \mathbf{z}_{<t}, \mathbf{x}_{\leq t}) \parallel p_\theta(\mathbf{z}_t | \mathbf{z}_{<t}) \right] . \tag{4.20}$$

Prior distributions for continuous and discrete variables are assumed to be Gaussian and Bernoulli, respectively. Continuous variables are reparameterised (see [33, 114]) and discrete variables are obtained using the Gumbel-Softmax trick [127]. In practice, we minimise the inclusive KL divergence [128] for the $\mathbf{z}^{\mathrm{what}}$ and $\mathbf{z}^{\mathrm{where}}$ as we empirically find this leads to better performance. We also include an entropy loss on the $K-1$ foreground object masks in the training objective,

$$\mathcal{L}_H = \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{t=1}^{T} \sum_{k=1}^{K-1} -m_{t,i,j}^{\mathrm{fg}}\, m_{k,t,i,j} \ \log m_{k,t,i,j} . \tag{4.21}$$

This penalises pixels being explained by multiple components. The full training objective is the sum of the ELBO and the mask entropy loss:

$$\mathcal{L} = \mathcal{L}(\theta, \phi) + \mathcal{L}_H \,. \tag{4.22}$$

4.4 Experiments

This section presents experiments on unsupervised scene segmentation, object tracking, and a simulated object manipulation task to showcase the capabilities of APEX. Our recent, state-of-the-art baselines consist of SCALOR [111], OP3 [66], SLOT-ATTENTION [42], SPACE [27], and G-SWM [120].

Datasets We perform a qualitative evaluation using the real-world Sketchy dataset [29], which contains demonstration trajectories of a robot arm performing different tasks involving a set of objects. The images are pre-processed as in [41], using a 128×128 resolution and sequences of length 10. Sketchy, however, does not contain object annotations, which prohibits the quantitative evaluation of object segmentation and tracking methods. We therefore introduce the Panda Pushing Dataset (P2D), which shows a Panda arm interacting with objects in simulation and includes pixel-level ground truth segmentations as well as object tracking IDs. Up to three objects are spawned in each episode and the robot-arm moves along a randomly selected straight line in the horizontal direction. Objects in the dataset are sampled from a set of 14 common objects (e.g. mugs, coffee cans, apples) with varying shapes, colours, and textures (see [129]). We collect a total of 2,400 trajectories (2,000 for training, 200 for validation, and 200 for testing) with each trajectory having a length of 20 frames with a resolution of 128×128.

Metrics Similar to prior work (e.g. [61, 110]), the quality of object segmentations is evaluated using the Adjusted Rand Index (ARI) [130] and the Mean Segmentation Covering (MSC) on a held-out test set. To enable rigorous benchmarking, we report results on two variants of these metrics: one which only considers foreground objects (see [110]) as well as one where all pixels and ground truth masks are considered, including those belonging to the background. The latter is relevant in the context of this work as the background needs to be explicitly separated from the foreground for the manipulation task in section 4.4.3. Multi-object tracking metrics are evaluated following the procedure in Weis et al. [131], which is based on the protocol from the established MOT16 tracking benchmark [132].

Implementation Details APEX is trained with ADAM [133] and a learning rate of 10^{-4}. The baselines are trained with the default learning rates and optimisers from the released implementations. The number of components in OP3 and SLOT-ATTENTION is set to $K = 5$ for P2D and to $K = 8$ for Sketchy. For SCALOR, the hard constraint on object size as found in the original implementation is removed as P2D and Sketchy contain objects of various sizes. APEX, SCALOR, and OP3 are trained with a batch size of 4 for 4×10^4 iterations. An additional 10^4 warm-up iterations are used for G-SWM. With the exception of OP3, the models are trained on the full image sequences. OP3 is trained on sub-sequences of length five due to the model capacity limitations on images that are resized to 64×64 as used in the original model. Training APEX, SCALOR, G-SWM and OP3 on a single NVIDIA Titan RTX GPU takes about 20 hours each. For SPACE, the batch size is increased to 16 and the number of training iterations are increased to 2×10^5 and 1×10^5 iterations for P2D and Sketchy, respectively, to account for the fact that SPACE is trained on individual images rather than image sequences. For SLOT-ATTENTION the number of training iterations is further increased to 4×10^5 and 2.5×10^5 iterations for P2D and Sketchy, respectively, as we found that the models take longer to learn reasonable segmentation masks. The resulting wall-clock times for SPACE and SLOT-ATTENTION are about 5 hours and 20 hours, respectively.

	Object tracking	ARI-FG	MSC-FG	ARI	MSC
OP3	✓	0.30 ± 0.03	0.30 ± 0.01	0.15 ± 0.05	0.39 ± 0.03
SCALOR	✓	0.31 ± 0.03	0.37 ± 0.01	0.43 ± 0.04	0.52 ± 0.01
G-SWM *	✓	0.38 ± 0.17	0.42 ± 0.05	0.33 ± 0.13	0.55 ± 0.04
SPACE	✗	0.33 ± 0.17	0.49 ± 0.12	0.72 ± 0.21	0.62 ± 0.09
SLOT-ATT.	✗	0.77 ± 0.24	0.46 ± 0.16	0.25 ± 0.22	0.50 ± 0.14
APEX	✓	**0.89 ± 0.02**	**0.73 ± 0.01**	**0.93 ± 0.00**	**0.80 ± 0.01**

* The G-SWM results are computed with one failed random seed being excluded.

Table 4.1: Mean and standard deviation of the segmentation metrics on P2D from four random seeds.

	MOTA ↑	MOTP ↑	Match ↑	ID S. ↓	FPs ↓	Miss ↓	MD ↑	MT ↑
OP3	−48.6 ± 17.0	66.6 ± 0.9	25.7 ± 4.4	0.3 ± 0.1	74.4 ± 13.4	73.9 ± 4.5	15.2 ± 4.7	15.6 ± 4.8
SCALOR	−125.6 ± 6.3	73.5 ± 0.5	31.6 ± 1.1	2.6 ± 0.0	157.2 ± 6.5	65.8 ± 1.1	19.0 ± 1.7	22.3 ± 2.3
G-SWM *	−41.8 ± 2.0	73.1 ± 0.5	36.7 ± 4.0	2.0 ± 1.0	78.4 ± 2.5	61.3 ± 3.5	25.3 ± 4.6	28.6 ± 3.2
APEX	**50.5 ± 4.0**	**83.2 ± 0.6**	**79.7 ± 0.5**	**0.2 ± 0.1**	**29.2 ± 3.8**	**20.0 ± 0.5**	**70.8 ± 1.0**	**71.2 ± 0.7**

* The G-SWM results are computed with one failed random seed being excluded.

Table 4.2: Mean and standard deviation of the tracking metrics on P2D from four random seeds.

4.4.1 Unsupervised Segmentation and Tracking

Quantitative results for unsupervised segmentation and tracking on P2D are summarised in table 4.1 and table 4.2, respectively.

For both tasks, it can be seen that APEX outperforms the baselines by significant margins. We also observe a smaller standard deviation of the scores for APEX, indicating that APEX is more stable than the baselines. We attribute these improvements to the principled mask normalisation, the high-resolution feature maps, and the mask entropy loss. Qualitative tracking results for APEX are shown in fig. 4.1. It can be seen that APEX is clearly able to track individual objects even through occlusions, thus corroborating the quantitative results in table 4.2.

Qualitative segmentation results for APEX as well as for the baselines are shown in figs. 4.3 and 4.4 on P2D and Sketchy, respectively. In fig. 4.3, a fairly cluttered scene from P2D can be seen where the robot arm is interacting with two objects in close proximity to each other. While APEX clearly segments the foreground objects and the arm itself – a key prerequisite for task planning and control – the baselines struggle to do so. SCALOR is unable to accurately segment the

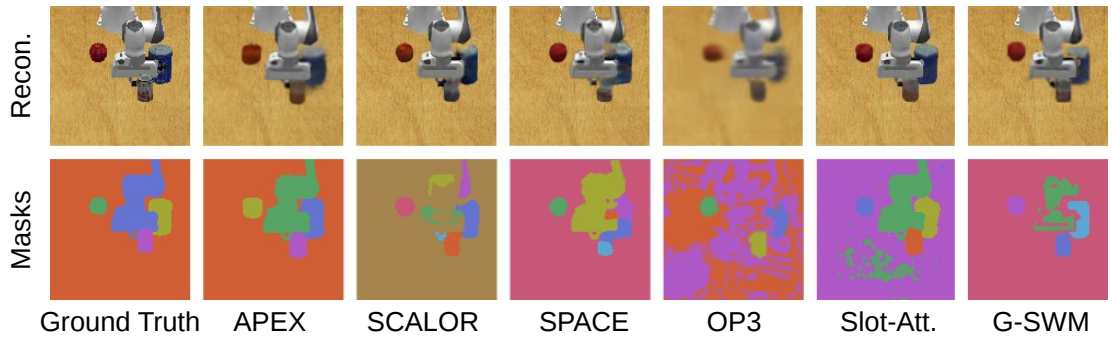


Figure 4.3: Scene segmentation results on P2D. APEX achieves the qualitatively best results, cleanly segmenting all foreground objects as well as the Panda arm.

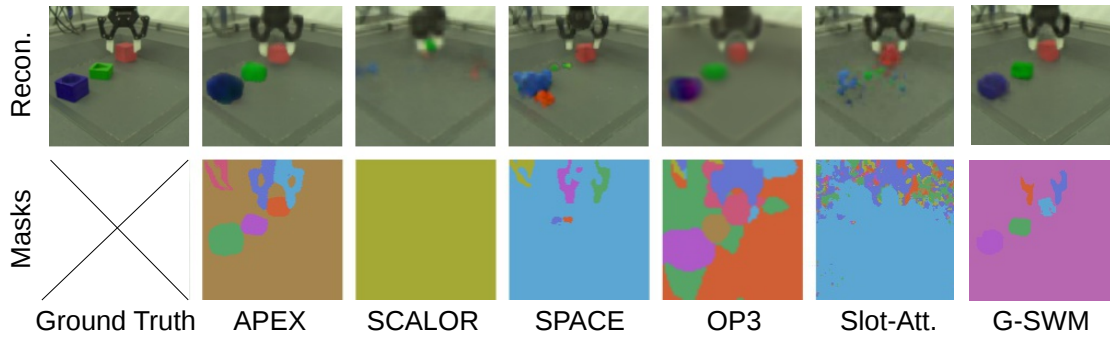


Figure 4.4: Scene segmentation results on Sketchy. In contrast to the baselines, APEX manages to cleanly segment all objects even on this more challenging real-world dataset.

arm with part of it being captured by the background module. We conjecture that this is due to the Gaussian image likelihood used for training SCALOR, as this does not untie the standard deviation of the foreground and the background likelihood as in APEX. A similar problem is observed with G-SWM which uses the same image likelihood modelling as SCALOR. For APEX and SPACE, in contrast, a smaller standard deviation is used for the background likelihood (0.04) than that of the foreground (0.1) to prevent the background module from also capturing the foreground objects. A Gaussian likelihood with a smaller standard deviation leads to a sharper distribution and thus results in a very low likelihood when the reconstructed colour deviates even slightly from the target. This will force the background module to focus on the more uniform background pixels which are easier to reconstruct compared to the foreground pixels. This is further examined in section 4.4.2. While SPACE manages to segment the robot arm from the background,

	ARI-FG	MSC-FG	ARI	MSC
APEX	**0.89 ± 0.02**	**0.73 ± 0.01**	**0.93 ± 0.00**	**0.80 ± 0.01**
Image space STN	0.71 ± 0.27	0.62 ± 0.10	0.91 ± 0.01	0.71 ± 0.08
No entropy loss	0.63 ± 0.18	0.63 ± 0.07	0.92 ± 0.01	0.73 ± 0.05
SCALOR-norm	0.62 ± 0.30	0.60 ± 0.13	0.90 ± 0.03	0.70 ± 0.09
Gaussian likelihood *	0.10 ± 0.01	0.06 ± 0.00	0.00 ± 0.00	0.28 ± 0.00
Gaussian likelihood & SCALOR-norm. *	0.02 ± 0.00	0.06 ± 0.00	0.00 ± 0.00	0.28 ± 0.00

* These models consistently fail to meaningfully segment the images.

Table 4.3: Mean and standard deviation of the segmentation metrics on P2D from four random seeds.

it fails to segment the small objects accurately. Unlike APEX, SPACE operates on static images and can thus not leverage temporal information. We argue that this helps APEX to learn to distinguish objects even when they are physically very close to each other in one frame, as they might move relative to each other in other frames. Both OP3 and SLOT-ATTENTION are able to roughly segment the objects but the segmentation results are noisy and inaccurate.

In fig. 4.4, APEX accurately segments all the objects in the scene – even the cables of the manipulator. We attribute the segmentation of the robot manipulator into left and right grippers to the relative motion (open/close) of the two gripper handles. In contrast, SCALOR fails to segment the objects and while SPACE is able to segment the arm, it omits the other foreground objects. This might be caused by the fact that the objects have a uniform colour and are therefore easily reconstructed by the background module. G-SWM successfully segments the objects, but part of the arm is treated as background. OP3 and SLOT-ATTENTION struggle to predict accurate masks.

4.4.2 Ablation Study

A set of ablation experiments is performed to validate the efficacy of the key design choices that set apart APEX from prior art: better scene encoding, principled mask normalisation, and a mask entropy loss. The results are summarised in table 4.3. Rather than re-using features extracted by the backbone with STNs when inferring object latents, it can be observed that using STNs to extract

information from the input images, such as in SCALOR, performs consistently worse. Re-using the backbone features also facilitates a reduction in model parameters and computation. The foreground scores when training without the additional mask entropy loss are also smaller, indicating that the entropy loss is indeed beneficial for learning to disambiguate foreground objects. It can be seen that mimicking the normalisation scheme from SCALOR results in lower scores compared to the mask normalisation introduced in section 4.3. For the former, the STN outputs a mask $\alpha_k^s \in [0, 1]$ for each component and the masks are normalised according to $(\alpha_k^s)^2 / \sum_k \alpha_k^s$, which is numerically less stable than a softmax. This might also explain the increased standard deviation of the scores, especially for the foreground ARI. Finally, we compare the use of APEX's SGMM image likelihood formulation to using a standard Gaussian likelihood. Both variants of APEX with a standard Gaussian likelihood fail to learn object-centric scene decompositions, highlighting the benefit of the SGMM formulation with separate standard deviations for foreground and background modules.

4.4.3 Object Arrangement Task

Generative models can be used to learn concise and informative representations that can be used in downstream tasks. In contrast to methods where a single latent vector encapsulates all object-relevant information (e.g. [42, 66]), further factorising the information into $[z_{k,t}^{\text{pres}}, \mathbf{z}_{k,t}^{\text{where}}, \mathbf{z}_{k,t}^{\text{what}}]$ as well as foreground and background latents allows us to directly use these representations as inputs to a controller.

We demonstrate this in an object arrangement task where a robot is required to pick and place objects on a table according to a *goal image*. This image is first parsed into foreground objects and background. Assuming depth information is available, e.g. via a stereo camera, the 3D location and shape of an object are obtained by filtering a depth image according to the object mask. The current scene is processed in the same fashion, which allows the current objects to be matched to the associated objects in the goal image according to the smallest $\mathcal{L}_2$ distance in terms of the objects' appearances encoded by $\mathbf{z}_k^{\text{what}}$. This is facilitated by discarding

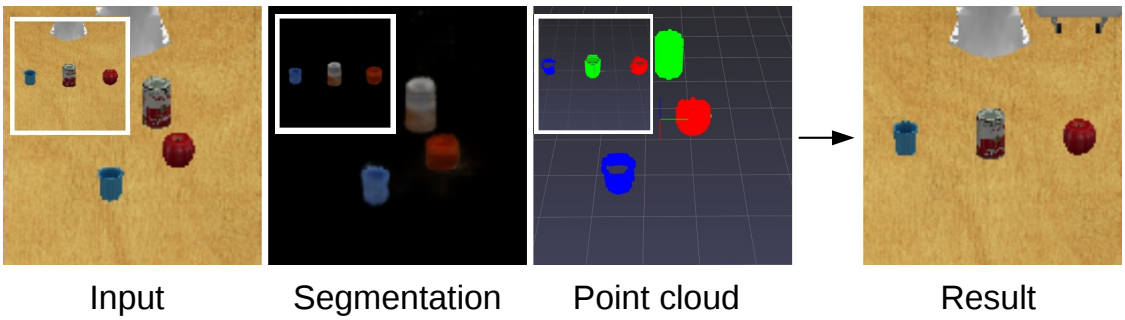


Figure 4.5: Object arrangement task illustration. Given an input of the current scene and a goal image (a), objects are segmented (b) and matched (c). Depth information is used to obtain the 3D locations and shapes of all objects, which are used as inputs to a heuristic control policy that tries to move the objects to the desired locations specified by the target image.

empty detections according to z_k^{pres} and the explicit separation of foreground and background components. The current and desired location and shape of each object serve as inputs to a heuristic control policy that executes a sequence of sub-tasks to move the objects. This is illustrated in fig. 4.5. To avoid collisions, a check for whether the target location is occupied by other objects is conducted before executing a sub-task. If all target locations are occupied, one of the unsorted objects is moved to an edge of the table to make space for the other objects. This object will be moved to its target space at the end of the re-arrangement process.

Three object arrangement tasks are created involving two to four objects on a table. For each task, we create 150 test scenarios with the same group of objects spawned in different locations. The performance of APEX is compared to SCALOR, SPACE and G-SWM. Comparisons against OP3 and SLOT-ATTENTION are omitted here as neither explicitly models foreground and background components which are required for task execution. Performance is quantified using the mean distance of the final object locations relative to the desired object locations. We set a fixed penalty of 1m for outliers where an object falls off the table.

The results are summarised in table 4.4. APEX performs better than SPACE and SCALOR on all three tasks. Some failures are caused by missing detections or incorrect object matches. G-SWM performs better than APEX on tasks with two or three objects but fails to segment the goal image properly on task with

	Two objects	Three objects	Four objects
SPACE	0.07 ± 0.16	0.23 ± 0.25	0.23 ± 0.20
SCALOR	0.22 ± 0.17	0.15 ± 0.17	0.18 ± 0.21
G-SWM	$\mathbf{0.02 \pm 0.09}$	$\mathbf{0.04 \pm 0.12}$	0.25 ± 0.21
APEX	0.04 ± 0.10	0.06 ± 0.13	$\mathbf{0.09 \pm 0.18}$

Table 4.4: Mean and standard deviation of the object distance to the desired goal positions in metres.

four objects. Although G-SWM outperforms APEX when there are fewer objects, APEX appears to pull ahead as the number of objects increases. APEX also performs better on segmenting the robot arm (fig. 4.3, fig. 4.4), which is excluded from this task. SPACE performs worst on tasks with three objects, but better than SCALOR for two objects. We find that SPACE tends to oversegment the objects, i.e., a single object is divided into several components, which reduces the efficacy of both object matching and location estimation. We hypothesise that the improvement of SCALOR compared to SPACE is facilitated by the propagation module which is also incorporated into APEX and aids the learning of higher quality segmentations.

4.5 Conclusions

This chapter proposes APEX , a novel, object-centric generative model designed to provide state-of-the-art unsupervised object segmentation and tracking on datasets commonly encountered in robotics. APEX is evaluated on the established Sketchy dataset [29] for qualitative results and on a custom Panda Pushing Dataset (P2D) for both quantitative and qualitative results. We show that APEX comprehensively outperforms prior art in terms of segmentation and tracking by leveraging improved feature encoding modules as well as a principled normalisation scheme for object and background masks. Finally, we demonstrate the efficacy of the unsupervised object representations learned by APEX on a robot manipulation task that involves the rearrangement of several objects on a table. APEX outperforms most of the baselines due to consistently providing segmentations of significantly higher quality, leading to improvements in object matching and 3D shape extraction.

5

ObPose: Leveraging Pose for Object-Centric Scene Inference and Generation in 3D

In last chapter, we have proposed an unsupervised OCGM for the robotic tasks. Nevertheless, the 2D OCGM can only capture the projected 2D shapes of the objects in the image space, which can provide limited information in robotic applications. In this chapter, we thus present ObPose, an unsupervised object-centric inference and generation model which learns 3D-structured latent representations from RGB-D scenes. Inspired by prior art in 2D representation learning, ObPose considers a factorised latent space, separately encoding object location (*where*) and appearance (*what*). ObPose further leverages an object's *pose* (i.e. location and orientation), defined via a minimum volume principle, as a novel inductive bias for learning the *where* component. To achieve this, we propose an efficient, voxelised approximation approach to recover the object shape directly from a neural radiance field (NeRF). As a consequence, ObPose models each scene as a composition of NeRFs, richly representing individual objects. To evaluate the quality of the learned representations, ObPose is evaluated quantitatively on the YCB, MultiShapeNet, and CLEVR datatasets for unsupervised scene segmentation, outperforming the current state-of-the-art in 3D scene inference (ObSuRF) by a significant margin. Generative

results provide qualitative demonstration that the same ObPose model can both generate novel scenes and flexibly edit the objects in them. These capacities again reflect the quality of the learned latents and the benefits of disentangling the *where* and *what* components of a scene. Key design choices made in the ObPose encoder are validated with ablations. This work is published as pre-print:

Y. Wu, O. Parker Jones, and I. Posner. "ObPose: Leveraging Pose for Object-Centric Scene Inference and Generation in 3D." In: *arXiv preprint arXiv:2206.03591*.

5.1 Introduction

In recent years, object-centric representations have emerged as a paradigm shift in machine perception. Intuitively, inference or prediction tasks in down-stream applications are significantly simplified by reducing the dimensionality of the hypothesis space from raw perceptual inputs, such as pixels or point-clouds, to something more akin to a traditional state-space representation. While reasoning over objects rather than pixels has long been the aspiration of machine vision research, it is the ability to learn such a representation in an unsupervised, generative way that unlocks the use of large-scale, unlabelled data for this purpose. As consequence, research into object-centric generative models (OCGMs) is rapidly gathering pace.

Central to the success of an OCGM are the inductive biases used to encourage the decomposition of a scene into its constituent components. With the field still largely in its infancy, much of the work to date has confined itself to 2D scene observations to achieve both scene inference [e.g. 10, 11, 31, 42, 45, 115, 116] and, in some cases, generation [e.g. 31, 45]. In contrast, unsupervised methods for object-centric scene decomposition operating directly on 3D inputs remain comparatively unexplored [18, 134] – despite the benefits due to the added information contained in the input. As a case in point, [18] recently established that access to 3D information significantly speeds up learning. Another benefit is that, for the parts of an object visible to a 3D sensor, object *shape* is readily accessible and does not have to be inferred, either from a single view [e.g. 21, 105] or from multiple views [e.g. 70, 135]. We conjecture that object shape can serve as a highly informative inductive bias for object-centric learning. As we elaborate below, we reason that the asymmetry of a shape can be used to discover an object's pose, and pose can help to identify and locate an object in space. Here we present ObPose, an unsupervised OCGM that takes RGB-D images (or video) as input and learns to segment the underlying scene into its constituent 3D objects, as well as into an explicit background representation. As we will show, ObPose can also be used for scene generation and editing. Inspired by prior art in 2D settings [11–13, 17, 27, 59], ObPose factorises its latent embedding

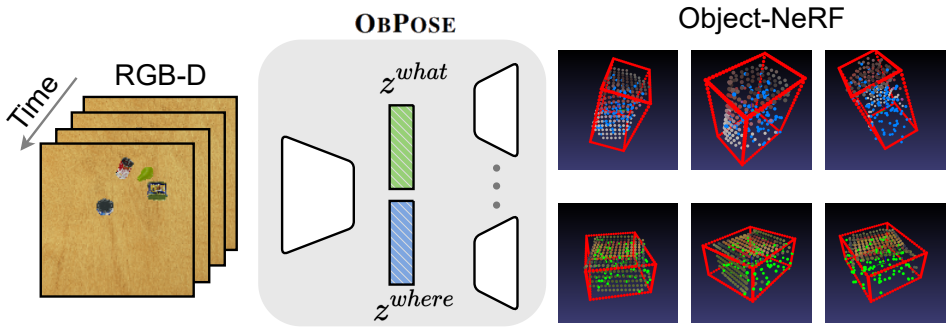


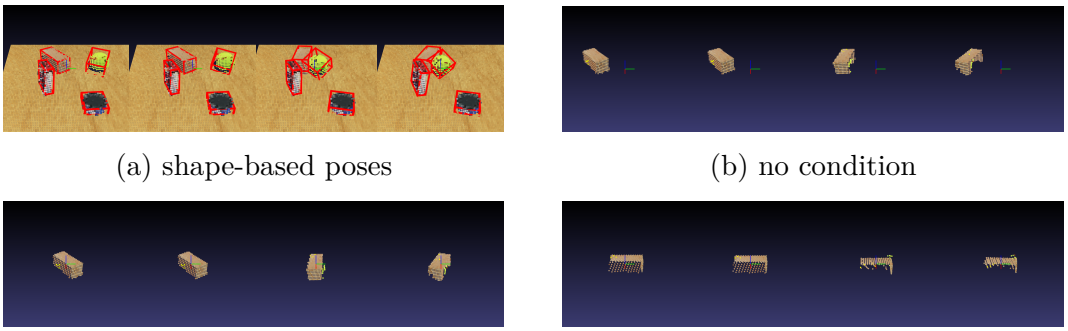
Figure 5.1: Overview. Front-view observations can be fed into ObPose (e.g. video of objects on a table, top left). ObPose then reconstructs the sparse voxelised point cloud and normalised *pose* of each object (two objects shown, top right). Each point within a slot is coloured to represent a specific object.

into a component capturing an object's location and appearance (*where* and *what* components, respectively). This factorisation provides a strong inductive bias, helping the model to disentangle its input into meaningful concepts for downstream use. A key contribution of ObPose is the introduction of *pose* (i.e. location and orientation) as a novel inductive bias.

ObPose is not a pose-estimation model. Rather, ObPose infers pose information from an object's shape to reduce apparent variance and to simplify the learning of the model's *what* component in 3D (see fig. 5.2) – ultimately for use in downstream tasks like segmentation and scene editing. ObPose infers pose information without supervision, using a minimum volume principle defined using the tightest bounding box that constrains the object. Effectively, the tightest bounding box will reveal asymmetries in an object's shape, if there are any, which can be used to constrain the object's orientation. We further propose a voxelised approximation approach that recovers an object's shape in a computationally tractable way from a neural radiance field (NeRF) [136]. Although the recovery of an object's shape from a NeRF can be prohibitively expensive, our approach allows this to be integrated efficiently into the training loop.

In a series of experiments, ObPose outperforms the current state-of-the-art in 3D scene inference, ObSuRF [18], by significant margins. Evaluations are performed on the CLEVR dataset [137], the MultiShapeNet dataset [18, 138], and the YCB dataset for unsupervised scene segmentation [28], in the latter case using both

RGB-D moving-objects (video) and multi-view static scenes. An ablation study on the ObPose encoder serves to validate the design decisions that distinguish its use of attention from alternative attention mechanisms represented by Slot Attention [42] and GENESIS-v2 [31]. In summary, the key contributions of this chapter are: (1) a new state-of-the-art unsupervised scene segmentation model for 3D, ObPose, together with insights into its design decisions; (2) a novel inductive bias for 3D OCGMs, *pose*, together with its motivation; and (3) a general method for fast shape evaluation from NeRFs.



(a) shape-based poses (b) no condition

(c) conditioned on location only (d) conditioned on location and orientation

Figure 5.2: Visualisation of the recovered bounding boxes estimated from the object shape. We additionally illustrate the object (wooden box) appearance viewed by (b) being not conditioned on any location information (c) being conditioned on the shape-based estimated object location and (d) being conditioned on the shape-based estimated object location and orientation. We show that the variance of the object appearance is reduced leveraging the pose information.

5.2 Related Work

ObPose builds upon prior OCGM work on unsupervised segmentation in both 2D and 3D. Most OCGMs for 2D scene segmentation are formulated as variational autoencoders (VAEs) [33, 114], where different likelihood models serve to explain observations. One set of VAE-OCGMs use bounding boxes, derived from spatial transformer networks (STNs) to represent (*glimpse*) individual objects [11–13, 27, 59, 139]. Another set represents objects via unsupervised instance segmentation, using pixel-wise mixture models [10, 26, 42, 45, 61, 66, 115–117]. This latter set relaxes the spatial-consistency requirements imposed by bounding boxes [40], permitting more flexible modelling of objects with complex shapes and textures. However, relaxing spatial consistency has the side-effect that performance can sometimes be biased by features such as the colour of the object [140], which has motivated the search for additional inductive biases. A promising candidate is temporal information. To this end, some works [66, 122, 141] operate on video data and model the correlations between objects explicitly using graph neural networks (GNNs) [142, 143] or Transformers [144].

The idea of a reference pose for an object, has precedent in the context of 6D pose estimation, which aims to find the translation and rotation of an object with respect to some frame of reference. In the supervised setting, labels are defined with respect to a given reference frame [145–150]. Recently, it has been shown that pose between views of an object, or objects from a common category, can be inferred without labels. Such relative poses have been found for point clouds [151] and RGB-D images [152]. To the best of our knowledge, we are the first to propose a minimum volume approach for discovering pose without supervision and to include pose information, to reduce its variance in the latent code, as an inductive bias.

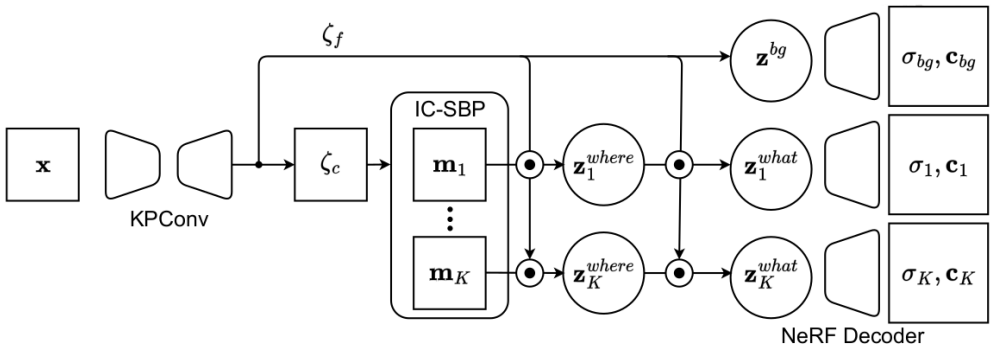


Figure 5.3: Model architecture. Given an input RGB-D image $\mathbf{x}$, the KPConv backbone extracts point embeddings ζ_c and a background component $\mathbf{z}^{bg}$. The point embeddings are clustered by the IC-SBP into soft attention masks, $\mathbf{m}_1 \dots \mathbf{m}_K$, for each object slot. Given these object masks, a *where* module infers the location of each object encoded by $\mathbf{z}^{where}$. Conditioned on object poses, which are recovered via the minimum volume principle, the *what* latents, $\mathbf{z}^{what}$, are encoded and then decoded into object NeRFs. Object NeRFs are finally composed with the background component to reconstruct the observed scene.

5.3 Methods

ObPose takes RGB-D videos (or images) as input and learns to segment scenes into a set of foreground objects, with a single background component. The location and orientation of objects can be estimated from the respective shapes which we reconstruct using NeRFs [136]. In contrast to previous works [18] where object NeRFs have to model object appearance and location jointly, each object NeRF in ObPose only models the 3D geometry and the texture of the objects by conditioning on the predicted object locations and orientations. To perform location and orientation conditioned inference, ObPose clusters pointwise embeddings that are encoded from a standard (KPConv-based) backbone [39] into a soft attention mask for each object using an instance colouring stick-breaking process (IC-SBP). Then a *where*-inference step predicts the object locations and orientations given the object-wise attention masks from the IC-SBP. Finally, a *what*-inference step encodes the appearance and shape information (conditioned on the object locations and orientations). This information is then decoded into NeRFs and all of the NeRFs are composed with the background component to reconstruct the original scene. A schematic overview of the ObPose architecture is provided in Figure 5.3.

5.3.1 Encoder

The input observation $\mathbf{x}_t$ is a RGB-D image of height H and width W for each time-step $t \in \{1 \dots T\}$. RGB-D depth images are converted into the point clouds $\mathbf{p}_t$ using the known camera parameters (extrinsic and intrinsic). The point clouds $\mathbf{p}_t$ and the RGB channels of $\mathbf{x}_t$ are then concatenated and encoded into a point embedding $\zeta_t^{H_1 \times W_1 \times D}$. This is achieved using a U-Net-like backbone module [37] consisting of several KPConv layers [39]. A KPConv layer is an extension of a 2D convolutional neural network (CNN) layer for point clouds, which preserves the translation invariance properties of CNNs. For computational efficiency, output embeddings are upsampled to the resolution of the first downsampled embedding using trilinear interpolation. Following prior work [31], we additionally build two heads (point-wise MLPs) upon the output embedding ζ_t, with one predicting the colour embedding $\zeta_{c,t}^{H_1 \times W_1 \times D_c}$ for the IC-SBP, and the other the feature embedding $\zeta_{f,t}^{H_1 \times W_1 \times D_f}$ for other encoding tasks.

Instance Colouring Stick-Breaking Process for Video

An instance colouring stick-breaking process (IC-SBP) is a clustering algorithm that takes point embeddings $\zeta_{c,t}^{H_1 \times W_1 \times D_c}$ as inputs and outputs K predicted soft attention masks $\mathbf{m}_k \in [0,1]^{H_1 \times W_1}$. The stick-breaking process [10] guarantees that the masks are normalised:

$$\mathbf{m}_1 = \alpha_1, \quad \mathbf{m}_k = \mathbf{s}_{k-1} \odot \alpha_k, \quad \mathbf{m}_K = \mathbf{s}_K, \tag{5.1}$$

where the *scope* $\mathbf{s}_k \in [0,1]^{H_1 \times W_1}$ tracks which pixels have not yet been explained. $\mathbf{s}_k$ is initialised and updated as follows:

$$\mathbf{s}_0 = \mathbb{1}^{H_1 \times W_1}, \quad \mathbf{s}_k = \mathbf{s}_{k-1} \odot (1 - \alpha_k) \tag{5.2}$$

The alpha mask α_k is computed as the distance between a cluster seed $\zeta_{i,j}$ and all individual point embeddings according to a kernel ψ. Readers are referred

to [31] for details on kernel selection and seed sampling. We compute the background component by passing the pixel embeddings $\zeta_{c,t}^{H_1 \times W_1 \times D_c}$ through a multilayer perceptron (MLP) ρ to compute a pre-scope $\mathbf{s}^p$:

$$\mathbf{s}^{\mathbf{p}} = \text{softmax}(\rho(\zeta) \in \mathbb{R}^{H_1 \times W_1 \times 2}) \tag{5.3}$$

By convention, we take the first channel of $\mathbf{s}^p$ to be the background mask and the last channel to be the *scope* $\mathbf{s}_0$ in the IC-SBP.

For the video input, we extend the original IC-SBP to include an additional propagation step. The cluster seed $\zeta_{i,j}$ can be used as an ID for each slot throughout the video. The cluster seed sampled at $t = 1$ is stored and used to compute the alpha masks for frames of $t \geq 2$. To ensure the normalisation of the masks $\mathbf{m}_k$, the SBP operation takes the remaining scope $\mathbf{s}_K$ as the last component mask $\mathbf{m}_K$. This does not have an associated seed for propagation. We run one more step of the SBP for the last component and flag the remaining scope $\mathbf{s}_K$ as unused. Concretely, we add an additional penalty loss to encourage the final remaining scope $\mathbf{s}_K$ to be zero everywhere, thereby motivating the model to explain the whole observation using only the previous slots.

Object Location and Orientation Conditioned Encoding

To facilitate a disentangled encoding of *what* and *where*, we firstly introduce shape-based object location and orientation estimation. We note that a point cloud $\mathbf{P}_{t,k} = \{\mathbf{p}_j | j \in \{1, ..., N_k\}\}$ of object k at time step t can be viewed as a discrete sampling from the object surface and thus preserves the shape information of the objects. Intuitively, the location of the object $\mathbf{T}_{t,k}$ can be initialised at the centre of mass of $\mathbf{P}_{t,k}$:

$$\mathbf{T}_{t,k} = \frac{1}{N_k} \sum_{j \in \{1,...,N_k\}} \mathbf{p}_j \tag{5.4}$$

with N_k being the number of points of object k. For the orientation of the object, which can be represented by a rotation matrix $\mathbf{R}_{t,k} \in \text{SO}(3)$, we propose the following *minimum volume principle* to define a unique shape-based object orientation: given

a set of points, we set the orientation of the object represented by those points to the orientation of the tightest bounding box that contains those points. Concretely, we find this bounding box by first transforming the points under several selection rotation matrices $\mathbf{R}^s \in \mathrm{SO}(3)$ and then computing the volume of the axis-aligned bounding boxes (AABBs) that contain these points. Selection rotation matrices are generated as equivolumetric grids on the SO(3) manifold [153, 154]. This method is based on the HEALPix method of generating equal area grids on the 2-sphere [155]. Due to the symmetry property of the bounding boxes, the smallest bounding box can have multiple solutions, e.g. swapping the x-axis and the y-axis of the coordinates of the bounding box will result in another bounding box that has the same volume. Therefore, we select all the bounding boxes whose volumes are in the range $[V_{\min}, (1+\beta)\, V_{\min}]$ to account for this issue, where $V_{\min}$ is the minimum volume of all the AABBs and β is a small tolerance factor. We pick the AABB whose orientation is closest to the world coordinates and whose orientation is represented by the identity matrix $\mathbf{I} \in \mathbb{R}^{3\times3}$ for the first time step, and otherwise to the orientation in the previous time step. The distance $d_{\mathrm{SO}(3)}$ is measured by the geodesic distance on the SO(3) manifold:

$$d_{\mathrm{SO}(3)} = \arccos\left(\frac{\mathrm{tr}(\mathbf{R}_{t,k}\mathbf{R}_{t-1,k}^{-1}) - 1}{2}\right) \tag{5.5}$$

The points $\mathbf{p}_j \in \mathbf{P}_{t,k}$ are then transformed to their object coordinates given a pair of pose parameters $\{\mathbf{T}, \mathbf{R}\}$ as follows:

$$k\left(\mathbf{p}_j, \mathbf{R}, \mathbf{T}\right) = \frac{2}{s}(\mathbf{R})^{-1}\left(\mathbf{p}_j - \mathbf{T}\right) \tag{5.6}$$

The bounding box size $s \in \mathbb{R}$ is initialised to a sufficiently large number for all objects.

For each object slot at time step t, we find its associated point cloud $\mathbf{P}_{t,k}$ by computing the argmax over the attention masks $\mathbf{m}_{t,k}$ predicted from the IC-SBP. The point features $\zeta_{f,t}$ at time step t are also masked by the attention masks, i.e. $\zeta_{t,k} = \zeta_{f,t} \odot \mathrm{no\_grad}(\mathbf{m}_{t,k})$. The object location $\mathbf{T}_{t,k}$ estimated from the observed points $\mathbf{P}_{t,k}$ is, however, biased toward object surface as the object is only partially observed. We thus use a *where* module that takes $\mathbf{P}_{t,k}$ and $\zeta_{t,k}$ as input and predicts a

$\Delta\mathbf{T}_{t,k}$ to correct the bias. To learn a relative translation we first transform $\mathbf{P}_{t,k}$ using the pose parameters $\{\mathbf{T}_{t,k}, \mathbf{I}\}$ as in eq. (5.6). Concretely, we use a KPConv-based encoder followed by a recurrent neural network (RNN) as the *where* module to predict the mean and the variance of the posterior distribution $q(\mathbf{z}_{t,k}^{where}|x_{\leq t})$ parameterised as a Gaussian. The $\mathbf{z}_{t,k}^{where}$ is decoded to the $\Delta\mathbf{T}_{t,k}$ through an MLP f^{where} such that:

$$\Delta\mathbf{T}_{t,k} = T_{\max}\tanh(f^{where}(\mathbf{z}_{t,k}^{where})) \tag{5.7}$$

with $T_{\max} \in \mathbb{R}$ being the maximum delta translation. Given the updated object location $\hat{\mathbf{T}}_{t,k} = \mathbf{T}_{t,k} + \Delta\mathbf{T}_{t,k}$, we encode the shape and the appearance of the object from the observations transformed in the object pose $\{\hat{\mathbf{T}}_{t,k}, \mathbf{R}_{t,k}\}$. Similar to the *where* module, the posterior distribution $q(\mathbf{z}_{t,k}^{what}|\mathbf{z}_{\leq t,k}, x_{\leq t})$ is also parameterised by a KPConv encoder appended by an RNN. We use two RNN networks to model the prior distributions $p(\mathbf{z}_{t,k}^{where}|\mathbf{z}_{<t,k}^{where})$ and $p(\mathbf{z}_{t,k}^{what}|\mathbf{z}_{<t,k}^{what})$, whose hidden states are decoded by an MLP to the mean and standard deviation. The prior distribution at $t = 0$ are Gaussian distributions of zero mean for both the $\mathbf{z}^{where}$ and the $\mathbf{z}^{what}$.

5.3.2 Decoder

To explicitly model the 3D geometry of scenes, we represent each object and the background as a generative Neural Radiance Field (NeRF) [46, 71, 136]. Each generative NeRF is a function parameterised by an MLP that maps the world coordinates $\mathbf{p}$, the viewing direction $\mathbf{d}$ and the latent encoding $\mathbf{z}$ to a color value $\mathbf{c}$ and a density value σ: $f : (\mathbf{p}, \mathbf{d}, \mathbf{z}) \rightarrow (\mathbf{c}, \sigma)$. To compose the individual NeRFs into a single scene function, we propose to model the scene function as:

$$\sigma = \sigma_{\max}\tanh\left(\sum_{k=1}^{K}\text{softplus}\left(\sigma_k\right)\right), \hat{\sigma}_k = \sigma\underset{K}{\text{softmax}}\left(\sigma_k\right) \tag{5.8}$$

In our case, σ_k can be interpreted as the logits of the probability, indicating whether this voxel is occupied or not, and $\hat{\sigma}_k$ is the normalised density with a range of $[0, \sigma_{\max}]$. For the colours, we can compute the weighted mean: $\mathbf{c} = \frac{1}{\sigma_{\max}}\sum_1^K \hat{\sigma}_k\mathbf{c}_k$. We use the softmax function to account for the fact that the objects should not overlap, without needing to introduce extra hyper-parameters as in [18]. To estimate

the shape of the objects reconstructed from the NeRFs, we would like a fast approximation approach, as the full evaluation of the volumetric rendering is computationally expensive. Inspired by prior work [156], we divide each object bounding box into **S** sparse voxels along each dimension. The occupancy at these voxel centres $\mathbf{p}_v$ can then be evaluated by $\sigma_v = \tanh(\text{softplus}(f(\mathbf{p}_v, \mathbf{z}_k)))$. We denote the voxels as occupied if $\sigma_v > \sigma_T$, with the σ_T being a threshold. Given the set of the occupied voxels and their centre positions $\mathbf{p}_v$, we can recover the object centre $\mathbf{T}_{t,k}^{\text{shape}}$ and the object orientation $\mathbf{R}_{t,k}^{\text{shape}}$ as discussed in 5.3.1.

5.4 Training

Training a NeRF with known depth requires relatively few evaluations. For example, [18] propose to use only two evaluations of the NeRF for each training iteration, i.e. one evaluation at the surface and one evaluation at points between the camera and the surface. The observation loss can be divided into two terms, i.e. a texture loss term and a depth loss term[18]:

$$\mathcal{L}_{\text{obs}} = -\log\left(\sum_{k=1}^{K} \mathcal{N}(\mathbf{x}|\mathbf{c}_k, \sigma_{\text{std}}^2) \odot \hat{\sigma}_k^{\text{surface}}/\sigma_{\max}\right) + (-\log(\sigma(\mathbf{p}^{\text{surface}})) + \sigma(\mathbf{p}^{\text{air}})/\rho^{\text{air}}) \tag{5.9}$$

with σ_{std} denoting a fixed standard deviation, and ρ^{air} is a probability density of the point $\mathbf{p}^{\text{air}}$ being sampled. For the attention mask of the IC-SBP, the learning of the attention masks can be either supervised via a mixtures of Gaussian loss:

$$\mathcal{L}_{\text{att}} = -\left(\log\left(\sum_{k=1}^{K} \mathbf{m}_k \odot (\mathcal{N}(\mathbf{x}|\mathbf{c}_k, \sigma_{\text{std}}^2) \odot \hat{\sigma}_k^{\text{surface}}/\sigma_{\max})\right) + \log\left(\sum_{k=1}^{K} \mathbf{m}_k \odot \hat{\sigma}_k^{\text{surface}}/\sigma_{\max}\right)\right) \tag{5.10}$$

or a L2 loss:

$$\mathcal{L}_{\text{att}} = \sum_{k=1}^{K}((\mathcal{N}(\mathbf{x}|\mathbf{c}_k, \sigma_{\text{std}}^2)/\mathcal{N}(0|0, \sigma_{\text{std}}^2) - \mathbf{m}_k)^2 \odot \hat{\sigma}_k^{\text{surface}}/\sigma_{\max}) + \sum_{k=1}^{K}(\mathbf{m}_k - \hat{\sigma}_k^{\text{surface}}/\sigma_{\max})^2 \tag{5.11}$$

Empirically, we find that using a L2 loss can be beneficial for complex 3D scenes. In the end, we supervise the *where* module, penalise the remaining scope $\mathbf{s}_K$ and

regularize the latent embedding as follows:

$$\mathcal{L}_{\text{others}} = \sum_{k}(\hat{\mathbf{T}}_{t,k} - \mathbf{T}_{t,k}^{\text{shape}})^2 + \sum_{i=1}^{H_1}\sum_{j=1}^{W_1} s_{K,i,j} + \mathbb{KL}(q_\phi(\mathbf{z}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{\leq t})||p_\theta(\mathbf{z}_t|\mathbf{z}_{<t})) \quad (5.12)$$

Taken together, these losses contribute straightforwardly to the overall loss: $\mathcal{L} = \mathcal{L}_{\text{obs}} + \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{others}}$. In training, we use the Adam optimizer with a fixed learning rate of $4e^{-4}$ without any learning rate warm-up or decay strategy.

5.5 Experiments

To evaluate ObPose's performance on unsupervised object-centric inference and generation of 3D scenes we conduct experiments on the CLEVR-3D dataset [137] and the MultiShapeNet dataset used by ObSuRF [18], and on two YCB object datasets [28]. One of the YCB object datasets is a moving-object (video) dataset of RGB-D images captured from a fixed front view. The other YCB dataset contains static images captured from three different points of view, where the three views are obtained by rotating the camera by 120°/240° around the z-axis of the world frame. Performance on all datasets is compared against the recent baseline of ObSuRF [18], which, to the best of our knowledge, is the only unsupervised scene inference and generation model that operates on RGB-D images of 3D scenes. To validate our model design decisions, we compare performance using two alternative attention mechanisms in the encoder, one from slot attention [42] and another from an IC-SBP that does not explicitly model object location and orientation [31]. We further compare two ablations of the ObPose encoder: one conditioned only on the locations of the objects, and one conditioned on the full 6D poses.

5.5.1 Metrics

In the evaluations, we quantify segmentation quality using the Adjusted Rand Index (ARI) [130, 157] and Mean Segmentation Covering (MSC) as metrics. ARI measures the clustering similarity between the predicted segmentation masks and the ground-truth segmentation masks in a permutation-invariant fashion. This is appropriate for unsupervised segmentation approaches where there are no fixed associations between slots and objects. We evaluate segmentation accuracy on foreground objects using the foreground-only ARI and MSC (denoted ARI-FG and MSC-FG, respectively), and on the background using mean Intersection over Union (mIoU). All metrics are normalised between 0 and 1 where a score of 1 indicates perfect segmentation.

	CLEVR-3D		MultiShapeNet	
	ARI-FG↑	ARI↑	ARI-FG↑	ARI↑
ObSuRF	0.96	**0.95**	0.81	0.64
ObSuRF w/o overlap	0.86	0.06	0.94	0.16
slot att.	0.46	0.01	0.52	0.08
ObPose (ours)	**0.99**	0.84	**0.99**	**0.81**

Table 5.1: The segmentation results on the CLEVR-3D dataset and MultiShapeNet. The results are rounded to two decimal places.

	YCB Moving-Object			YCB Static		
	mIoU-BG↑	ARI-FG↑	MSC-FG↑	mIoU-BG↑	ARI-FG↑	MSC-FG↑
ObSuRF w/o overlap with depth	0.89 ± 0.09	0.18 ± 0.21	0.21 ± 0.02	0.92 ± 0.04	0.31 ± 0.15	0.37 ± 0.07
ObSuRF w/o overlap	0.97 ± 0.02	0.28 ± 0.36	0.28 ± 0.26	0.98 ± 0.00	0.22 ± 0.12	0.36 ± 0.05
slot att. *	0.98 ± 0.02	0.13 ± 0.05	0.19 ± 0.02	$\mathbf{1.00 \pm 0.00}$	0.80 ± 0.01	0.84 ± 0.00
ic-sbp	0.96 ± 0.05	0.87 ± 0.02	0.90 ± 0.02	0.97 ± 0.03	0.83 ± 0.09	0.80 ± 0.15
ObPose loc. only (ours)	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.96 \pm 0.00}$	$\mathbf{0.97 \pm 0.00}$	0.99 ± 0.00	$\mathbf{0.89 \pm 0.01}$	$\mathbf{0.87 \pm 0.03}$
ObPose loc.& rot. (ours)	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.96 \pm 0.00}$	$\mathbf{0.97 \pm 0.00}$	0.98 ± 0.00	0.88 ± 0.01	0.84 ± 0.03

* The slot att results are computed with one failed random seed being excluded for the YCB static dataset.

Table 5.2: Mean and standard deviation of the segmentation metrics on the YCB moving-object dataset and the YCB static dataset from three random seeds. The results are rounded to two decimal places. ObPose outperforms the baseline and the ablations as it explicitly estimates the locations and the orientations of the objects, disentangling object location and appearance.

5.5.2 Unsupervised 3D Object Segmentation

Quantitative results for the CLEVR dataset and the MultiShapeNet dataset are shown in section 5.5.2. We first observe that ObPose achieves better foreground segmentation performance (ARI-FG) compared to ObSuRF, which indicates that ObPose can perform better scene inference by conditioning on the location and the orientation. The lower full ARI score is caused by the fact that ObPose learns to include shadows caused by the existence of the objects in each object slot, whereas object shadows are labelled as background in the ground-truth masks. Quantitative segmentation results on the YCB dataset are additionally summarised in section 5.5.2. Here we only report ObSuRF results without using the overlap loss as we first observe that the ObSuRF baseline fails to segment the scenes properly using the default setting for 3D data from the open-sourced code. This might be attributed to the weight of a overlap loss used in ObSuRF, to discourage the objects from overlapping. In [18] the same failure mode is reported. In ObPose, we instead

account for the overlap using a hyper-parameter-free function (*softmax*) in eq. (5.8). This alleviates the computationally expensive hyper-parameter searching process. Interestingly, using the full 6D pose including the orientation and the location of the objects does not strongly affect the segmentation performance compared to using only the object positions as a way to condition the encoding. This suggests that for object segmentation tasks, the object location itself already provides a strong inductive bias for successful decomposition. Similar results are observed elsewhere in the literature [141], where simple ground-truth position information for objects is used in the first video frame, allowing the model to perform scene segmentation of 2D video data in a weakly supervised fashion. In our approach, we explicitly leverage the 3D reconstruction of the objects whose shape is estimated by the proposed voxelised shape approximation approach. This allows the model to infer the locations and the orientations of objects in a computationally efficient way without using any ground-truth labels. ObPose also achieves lower variation on metrics for both datasets, suggesting more stable training compared to the original IC-SBP.

5.5.3 Scene Generation and Scene Editing

Leveraging the disentangled *where* and *what* object-centric representation, ObPose can perform more flexible scene generation and scene editing than previous works[18, 31]. We demonstrate this benefit with the CLEVR-3D dataset. In fig. 5.4(a), we show scenes that are generated from ObPose by first sampling from the learned object and the background latent space and then rendering from the composition of object and background NeRFs. The object locations and orientations can be arbitrarily set to user-defined values. fig. 5.4(b–d) demonstrates that ObPose can further be used for flexible scene editing (i.e. adding, removing, or manipulating objects in a generated or inferred scene). The object-level scene manipulation fig. 5.4(d) as an important function of OCGMs has raised people's attention in generative models [71], ObPose thus for the first time implements this in an inference model.

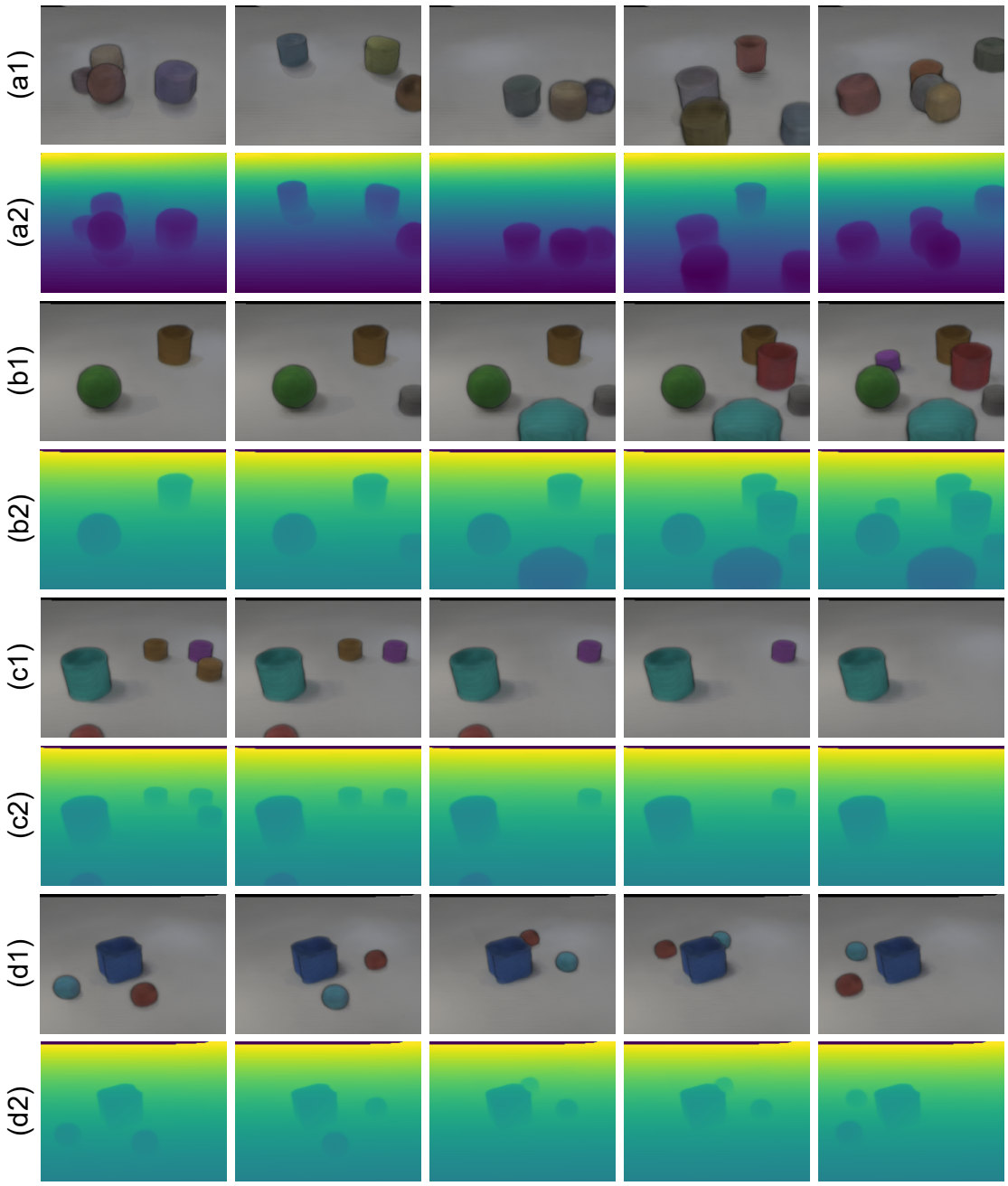


Figure 5.4: We demonstrate the rendered RGB and depth results of the scenes generated by sampling from the learned latent space (a). We also show the scene editing functions of object addition (b), object removal (c) and object-level scene manipulation (d).

5.6 Conclusion

We present ObPose, an object-centric inference and generation model that learns 3D-structured latent representations. The model extends IC-SBP for video input, and is noteworthy for introducing *poses* as an inductive bias for scene inference. The model's ability to infer object pose is facilitated by several recent innovations, including the use of NeRFs and the fast voxelised shape approximation proposed in this chapter. Our experimental results are validated on the CLEVR dataset, the MultiShapeNet dataset, and two synthetic YCB objects datasets. Given its empirical success, outperforming the prior state-of-the-art for static 3D scenes [18] and establishing a baseline for video, we plan to apply ObPose as a vision backbone for robot applications.

6

DreamUp3D: Object-Centric Generative Models for Single-View 3D Scene Understanding and Real-to-Sim Transfer

In this chapter, we propose an OCGM for the 3D scene understanding task and deploy it in real-world robotic scenarios. 3D scene understanding for robotic applications exhibits a unique set of requirements including real-time inference, object-centric latent representation learning, accurate 6D pose estimation and 3D reconstruction of objects. Current methods for scene understanding typically rely on a combination of trained models paired with either an explicit or learnt volumetric representation, all of which have their own drawbacks and limitations. We introduce *DreamUp3D*, a novel Object-Centric Generative Model (OCGM) designed explicitly to perform inference on a 3D scene informed only by a single RGB-D image. *DreamUp3D* is a self-supervised model, trained end-to-end, and is capable of segmenting objects, providing 3D object reconstructions, generating object-centric latent representations and accurate per-object 6D pose estimates. We compare *DreamUp3D* to baselines including NeRFs, pre-trained CLIP-features, ObSurf, and ObPose, in a range of tasks including 3D scene reconstruction, object-matching and object pose estimation. Our experiments show that our model outperforms all baselines by a significant margin in real-world scenarios displaying its applicability

for 3D scene understanding tasks while meeting the strict demands exhibited in robotics applications. This work is published as:

Y. Wu, H. Sáez de Ocáriz Borde, J. Collins, O. Parker Jones, I. Posner. "DreamUp3D: Object-Centric Generative Models for Single-View 3D Scene Understanding and Real-to-Sim Transfer." In: *IEEE Robotics and Automation Letters (RA-L)*

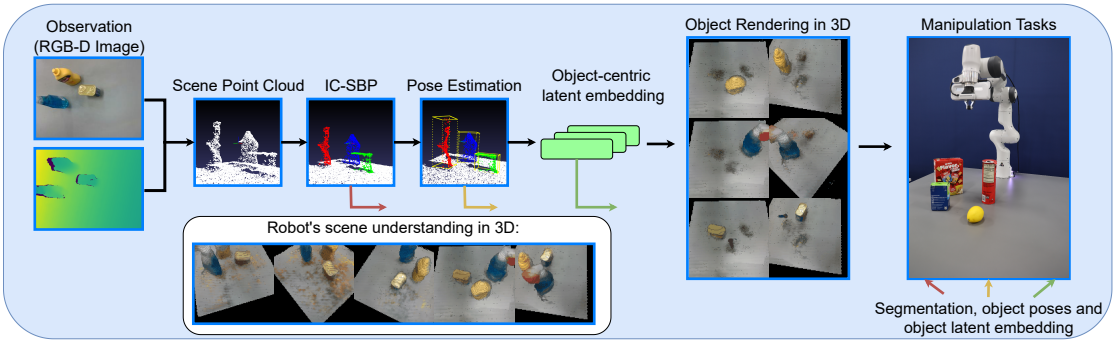


Figure 6.1: DreamUp3D interprets a single view RGB-D image in a 3D object-centric manner. It uses the IC-SBP algorithm to cluster the input point cloud into object masks. Then, it infers the object pose and the object-centric latent representation for each detected object. Each object is reconstructed in 3D and then together with the other objects and background component merged to form the entire 3D scene reconstruction. Predictions can then be leveraged for downstream manipulation tasks.

6.1 Introduction

Robots deployed in the real world face unique challenges as agents operating in unstructured 3D environments with only partial observability. For this reason, 3D scene understanding from limited observations is of paramount importance in facilitating tasks such as real-to-sim transfer and object manipulation. In online applications, observations need to be processed frequently through the 3D perception system to react to changes in the scene. Additionally, task-level planning requires a representation of the scene that is typically object-centric. To this end, the *desiderata* for robots operating in these settings are real-time operation, 3D reconstruction based on single-viewpoint observations, object-centric latent representations, and accurate per-object 6D pose.

Currently, NeRFs [136, 158] are employed as an implicit representation for 3D scene perception and understanding, and they have been widely explored in robotics. Recent studies have tested their capabilities in robotics for grasping [159], localisation [160] and tactile sensing [161]. However, NeRFs face significant limitations as they are formulated to represent the environment as a unified entity [159, 162]. In particular, for each new scene a new NeRF needs to be trained adding significant overhead in time and compute. Extensions of the original NeRF formulation

have reduced the impact of these limitations by leveraging multiresolution hash encoding to reduce the training time of NeRFs [163] and by extending NeRFs to reason about individual objects in a scene [164]. However, training NeRFs requires collecting images from multiple views with maximally varied viewing angles. In addition to this, identifying objects within a scene also relies on known object masks. Therefore, the NeRF formulation fails to stand up to the *desiderata* for real time robot operations in unstructured environments.

In comparison to NeRFs, OCGMs are inherently object-centric and operate in real time at inference. OCGMs leverage different attention mechanisms [27, 165, 166] that promote disentanglement for scene decomposition. The individual components are subsequently encoded into object-centric latent representations and decoded to reconstruct the scene. OCGMs can also be combined with NeRFs to model 3D scenes [19]. Recent advancements have enhanced the learning process by incorporating depth supervision [167], enabling the training of NeRFs using RGB-D inputs without the need for explicit ray marching. However, this approach encodes both object appearance and spatial information into a single latent representation, and thus object poses cannot be inferred independently. On the contrary, ObPose [168] proposes finding the minimum volume bounding box containing the object and uses the pose of the bounding box as the object pose. This enables pose estimation without labelled supervision. However, this shape-based pose estimation can be inaccurate under occlusions. Moreover, these previously proposed 3D OCGMs have only been evaluated in simulated 3D scenes, leaving the investigation of their applicability to real-world scenes to future work.

In this chapter, we propose DreamUp3D, an OCGM that integrates generative radiance fields (GRAFs) [46, 71, 136] to enhance 3D scene comprehension by overcoming the limitations of past models. DreamUp3D demonstrates transfer across real-world environments and overcomes occlusions via a shape completion module which is trained by a shape distillation mechanism that reuses the GRAF predictions as a training signal. This significantly reduces computational requirements by minimising the need for repetitive evaluations of the NeRF model to retrieve object

shapes, which traditionally involves thousands of evaluations using ray marching. Experimentally, we evaluate DreamUp3D against the *desiderata* for 3D scene understanding in robotics tasks. Concretely, we evaluate DreamUp3D in scene reconstruction, object-centric representation learning, and pose estimation.

6.2 Related Work

Recently, NeRFs have shown their potential as a compact 3D representation of the environment and are currently being applied to many robotics tasks, such as reinforcement learning [169], SLAM [170], and for grasping transparent objects [159, 171]. Among NeRF implementations, instant-NGP [163] is commonly chosen due to its fast reconstruction speed. However, these approaches often require retraining the NeRF model before each grasp to update the environment states. GraspNeRF [172] addresses this constraint by proposing a generalisable NeRF that is free from per-scene optimisation. Nevertheless, GraspNeRF is not object-centric and thus cannot interpret the scene at the object level. In contrast, [173] proposes to decode grasping from a learned object-centric latent representation, but the method lacks the ability to decompose a scene and requires a priori knowledge of object poses and the number of objects in the scene. Considering the desiderata for scene understanding for robots operating in the real-world, NeRFs, despite their ability to provide high-fidelity 3D reconstructions given multi-view observations, fundamentally face limitations in terms of object-level inference and real-time operation.

OCGMs are another type of scene understanding model that offer an unsupervised method for learning latent representations at the object level, enabling reasoning at a higher level of abstraction instead of at the pixel or point cloud level. In particular, they rely on inductive biases to promote the decomposition of a scene into its individual components. This enables the models to better understand the underlying structure of a scene and capture the relationships between its constituent objects. Early works have conducted unsupervised scene inference and generation in 2D (MONet [174], Slot Attention [165], GENESIS [175], GENESIS-V2 [166]), and for robotics applications using APEX [17, 176]. In both [17, 176] the 2D OCGM, APEX, is utilised for object matching using the learned object-centric latent representation in an object rearrangement task in simulation and a peg-in-hole task in the real world respectively. Nevertheless, the 2D reconstruction and 2D bounding boxes predicted by such OCGM are of limited use in a 3D world. Recent research [19] has thus focused on combining the 3D representation

power of NeRFs with the versatility of OCGMs. ObSuRF [167] proposes to further accelerate training by leveraging the depth channel of RGB-D images to guide the sampling of the NeRF queries. However, ObSuRF encodes the object appearance and location jointly into a single latent representation, and thus cannot infer per-object pose. ObPose [168] addresses this problem by introducing a minimum volume principle for shape-based 6D pose estimation without using human labels. ObPose is thus the first 3D OCGM that fulfils the desiderata for scene understanding by providing real-time inference, object segmentation and 3D reconstruction, object-centric latent representation learning and unsupervised 6D pose estimation, as DreamUp3D does. ObPose also demonstrates superior performance in terms of segmentation accuracy compared to ObSuRF on the YCB [28], MultiShapeNet [167], and CLEVR datasets [137]. We therefore choose ObPose as the main baseline for the unsupervised pose estimation task.

6.3 DreamUp3D

The following subsections divide the model into modules with distinct functions, starting with data preprocessing, scene segmentation for extracting object masks, pose estimation with shape completion for improved estimates, GRAF representation of objects, and concluding with an overview of model training. The overall model pipeline is depicted in Fig. 6.1. Additionally, an architectural diagram of our model can be found in Fig. 6.2.

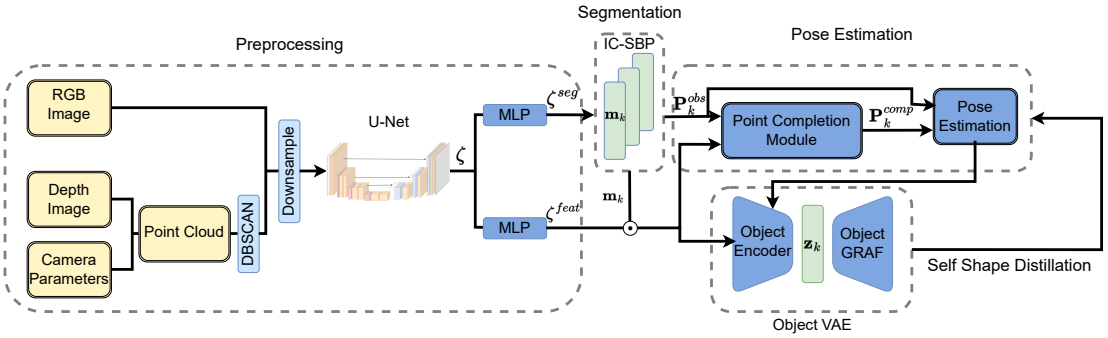


Figure 6.2: Architectural diagram of DreamUp3D. The model is composed of several distinct modules for data preprocessing, scene segmentation, pose estimation and object encoding. See Section 6.3 for details.

6.3.1 Data Pre-processing

Given a single RGB-D image input, $\mathbf{x} \in \mathbb{R}^{\mathbf{H}\times\mathbf{W}\times 4}$, we utilize the depth information to convert the input into a point cloud. Depth images are known to be noisy, especially at object edges or when there is reflectance from metal surfaces. Therefore, we employ a clustering algorithm, DBSCAN [177], to filter out noisy point observations from the raw input. The point cloud is then downsampled to a fixed number of points, N, and used, along with the RGB color, as input to a U-Net-like backbone module [37], which consists of several KPConv layers [39]. The KPConv layer extends the standard 2D convolutional layer to preserve the translation invariance of point cloud inputs. The output encoding produced by the U-Net, denoted as $\zeta \in \mathbb{R}^{N\times D}$, is then passed through two MLP heads for scene segmentation and feature encoding, as detailed in [166], where D represents the dimension of the

feature map. Specifically, based on $\zeta \in \mathbb{R}^{N \times D}$ and the two respective MLPs, we obtain the encodings $\zeta^{seg} \in \mathbb{R}^{N \times D}$ and $\zeta^{feat} \in \mathbb{R}^{N \times D}$. The encoding $\zeta^{seg} \in \mathbb{R}^{N \times D}$ is used for scene segmentation (Section 6.3.2), while $\zeta^{feat} \in \mathbb{R}^{N \times D}$ is directly used for scene reconstruction (Section 6.4.2). This preprocessing step adeptly transforms the input into these two embeddings in a noise-robust manner, which are subsequently utilised by the rest of the model architecture.

6.3.2 Scene Segmentation

Given the point-wise embedding ζ^{seg}, we apply an instance colouring stick-breaking process (IC-SBP) [166] for scene segmentation. IC-SBP is a clustering algorithm that predicts K soft attention masks $\{\mathbf{m}_k\}_{k=1}^{K} \in [0,1]^{N \times 1}$ that follow a stick-breaking process. The predicted attention masks are randomly ordered, which is achieved by stochastically sampling cluster seeds from the point-wise embeddings. We denote the first attention mask, $\mathbf{m}_0$, as the background mask and the last attention mask, $\mathbf{m}_K$, as the redundant scope that tracks the unexplained pixels. We refer readers to [166, 168] for the implementation details. The observed scene point cloud can then be segmented into several object point clouds denoted by $\mathbf{P}_k^{obs}$ where k is the object index.

6.3.3 Pose Estimation with Shape Completion

A 6D pose for each segmented object can be estimated by computing a minimum volume bounding box over the object's masked point cloud [168]. However, estimating a full 6D pose for a partially observed object is challenging and prone to inaccuracies and high variance when estimated from only the information available from a single view. To overcome this challenge, we propose a shape completion module that estimates the shape of occluded parts of objects.

First, each observed object point cloud, $\mathbf{P}_k^{obs}$, is transformed into a canonical pose, $\mathbf{P}_k^{can}$, by transforming each point in the point cloud as follows (in the same manner as [168]):

$$\mathbf{P}_k^{can} = \frac{2}{b}(\mathbf{R}_k)^{-1}\left(\mathbf{P}_k^{obs} - \mathbf{T}_k\right) \tag{6.1}$$

where $\mathbf{T}_k$ is the location of the sampled seed of the object mask $\mathbf{m}_k$ in the IC-SBP algorithm and $\mathbf{R}_k$ is the rotation matrix. For the first transformation, $\mathbf{R}_k$ is set to the identity matrix. The bounding box size $b \in \mathbb{R}^+$ is initialised to a sufficiently large number for all objects.

For each object, the shape completion module encodes the transformed point cloud $\mathbf{P}_k^{can}$ and the masked scene embedding, $\zeta_k^{feat} = \zeta^{feat} \odot \mathbf{m}_k$, using a KPConv-based autoencoder into a shape embedding $\mathbf{e}_k$. Using a tri-plane-based GRAF [46, 71, 136, 178], which we denote as π^{shape}, each shape embedding, $\mathbf{e}_k$, is decoded into a voxelised representation which is used for shape completion. A GRAF is a generative NeRF that models the 3D geometry and texture of an object by predicting the occupancy and the colour of any given query point $\mathbf{p}$. Specifically, π^{shape} learns to map the latent encoding $\mathbf{e}_k$ to the occupancy logits,

$$\sigma_k(\mathbf{p}) = \pi^{shape}(\mathbf{p}, \mathbf{e}_k(\zeta_k^{feat})), \tag{6.2}$$

at the given query point $\mathbf{p}$. The colour prediction from the original GRAF formulation is discarded here as only occupancy is needed for shape completion, resulting in the shape-GRAF, π^{shape}.

To reduce the computational overhead of approximating the 3D shape of the object, we divide each object bounding box into $\mathbf{S}$ voxels along each dimension. The centre position of each voxel, $\mathbf{p}_{v,k}$, is evaluated with an occupancy probability scalar value computed as in [168]:

$$\phi_k(\mathbf{p}_{v,k}) = \tanh\left(\text{softplus}\left(\sigma_k(\mathbf{p}_{v,k})\right)\right). \tag{6.3}$$

Voxels are considered occupied if $\phi(\mathbf{p}_{v,k}) > \phi_T$, with ϕ_T acting as a threshold. Occupied voxels are used as the shape completion points, $\mathbf{P}_k^{comp}$. A minimum volume bounding box that contains both $\mathbf{P}_k^{comp}$ and $\mathbf{P}_k^{obs}$ is used to represent the object pose. Again, the object point cloud, $\mathbf{P}_k^{obs}$, is transformed to a canonical pose, $\mathbf{P}_k^{can}$, using the improved object pose estimate and eq. (6.1), where the improved estimate of the object's position is $\mathbf{T}_k$ and the rotation is $\mathbf{R}_k$.

6.3.4 Scene Reconstruction

A latent embedding $\mathbf{z}_k$ is created by encoding the updated $\mathbf{P}_k^{can}$ and ζ_k^{feat} using a KPConv-based encoder. $\mathbf{z}_k$ is parameterised as a Gaussian and then used by a tri-plane-based GRAF to decode the 3D shape and colour of each object. We denote the GRAF that decodes the object embedding $\mathbf{z}_k$ as the object-GRAF. We follow the method of [178] to predict colour and occupancy logits for each object given the object embedding.

To reconstruct the entire scene each object must be reconstructed individually along with the background component. The occupancy and colour of the background are reconstructed by first encoding the observed point cloud together with the masked scene embedding, $\zeta_0^{feat} = \zeta^{feat} \odot \mathbf{m}_0$, using a KPConv encoder to predict a background latent embedding $\mathbf{z}_{bg}$ before being decoded by a third tri-plane-based background-GRAF.

Computing the occupancy probability of the entire scene, $\phi_{\text{scene}}(\mathbf{p})$, at the queried points $\mathbf{p}$ given the viewing direction, $\mathbf{d}$, using the outputs of the object and background GRAFs can be completed as follows [168]:

$$\phi_{\text{scene}}(\mathbf{p}) = \tanh\left(\sum_{k=0}^{K-1} \text{softplus}\left(\sigma_k(\mathbf{p})\right)\right) \tag{6.4}$$

For the colours of the scene $\mathbf{c}_{\text{scene}}(\mathbf{p}, \mathbf{d})$, we can compute the weighted mean [167, 168]:

$$\mathbf{c}_{\text{scene}}(\mathbf{p}, \mathbf{d}) = \sum_0^{K-1} \hat{\phi}_k(\mathbf{p}) \mathbf{c}_k(\mathbf{p}, \mathbf{d}), \tag{6.5}$$

with $\hat{\phi}_k(\mathbf{p}) = \phi_{\text{scene}}(\mathbf{p}) \underset{K}{\text{softmax}}\left(\sigma_k(\mathbf{p})\right)$.

6.3.5 Training

With known depth, the object-GRAF, shape-GRAF and background-GRAF only require two evaluations in each training iteration [167], i.e. one evaluation at the surface and one evaluation at points sampled between the camera and the surface. We denote these two points as $\mathbf{p}_{surf}$ and $\mathbf{p}_{empty}$ respectively. We refer readers to [167] for how $\mathbf{p}_{surf}$ and $\mathbf{p}_{empty}$ are sampled with the known depth.

The object-GRAF and the background-GRAF are queried at $\mathbf{p}_{surf}$ to learn the reconstruction of the geometry and the colour of the scene by maximising the likelihood of the observations:

$$\mathcal{L}_{\text{obs}} = -\log\left(\phi_{\text{scene}}(\mathbf{p}_{surf})\left(\sum_{k=0}^{K-1}\left(\mathcal{N}(\mathbf{x}|\mathbf{c}_k(\mathbf{p}_{surf},\mathbf{d}_{surf}),\sigma_{\text{std}}^2)\odot\hat{\phi}_k(\mathbf{p}_{surf})\right)\right)^{\eta}\right) \tag{6.6}$$

with $\eta \sim Ber(\phi_{\text{scene}}(\mathbf{p}_{surf}))$ and $\mathbf{d}_{surf}$ being the view directions of points $\mathbf{p}_{surf}$. To prevent the background module from overfitting to the scene, we use RANSAC [179] to detect the largest plane, e.g. the table, in the scene and constraint the background module to only learn to reconstruct the detected plane as the background. The GRAFs are queried at $\mathbf{p}_{empty}$ to learn to fit the empty space of the scene. However, $\mathbf{p}_{empty}$ can be sampled at places that are known to be empty, e.g. the points outside of the object bounding boxes. Instead, for the object-GRAFs, we choose the voxel centres of the object bounding boxes $\mathbf{p}_{v,k,\text{obj}}$, and find those in the empty space as the sampled points denoted as $\mathbf{p}_{e,k,obj}$. The points $\mathbf{p}_{e,k,obj}$ are determined using the known camera parameters and the depth observations. We thus only evaluate the background-GRAF at $\mathbf{p}_{empty}$ and minimise the occupancy in the empty space by computing the $\mathcal{L}_{\text{empty}}$ as follows:

$$\mathcal{L}_{\text{empty}} = \phi_0(\mathbf{p}_{empty})/\rho_{empty} - \sum_{k=1}^{K-1}\log(1-\phi_k(\mathbf{p}_{e,k,obj})) \tag{6.7}$$

where ρ_{empty} is the probability density of the point $\mathbf{p}_{empty}$ being sampled for the background-GRAF. To further improve the reconstruction accuracy, we sample N_r points of $\mathbf{p}_{empty}$ and $\mathbf{p}_{surf}$ from multi-view observations in each training iteration. Note that this is only used for training, and at test time, the inference only uses single-view input. To supervise the shape-GRAF, π^{shape}, we use a self-distillation loss to save computation by reusing the evaluations of the object-GRAF. Concretely, we compute the trilinear interpolation $T_{tr}(\mathbf{p}_{v,k},\phi(\mathbf{p}_{v,k,\text{obj}}))$ of the occupancy predictions of the object-GRAF at $\mathbf{p}_{v,k}$. We then distil the shape information from the object-GRAF to the shape-GRAF by minimising the KL divergence between the Bernoulli distribution of $T_{tr}(\mathbf{p}_{v,k},\phi(\mathbf{p}_{v,k,\text{obj}}))$ and $\phi(\mathbf{p}_{v,k})$:

$$\mathcal{L}_{\text{shape}} = \sum_{k=0}^{K-1}\mathbb{KL}(Ber(T_{tr}(\mathbf{p}_{v,k},\phi(\mathbf{p}_{v,k,\text{obj}})))||Ber(\phi(\mathbf{p}_{v,k}))) \tag{6.8}$$

The IC-SBP attention masks are supervised via a L2 Loss:

$$\mathcal{L}_{\text{att}} = \sum_{k=0}^{K-1}(\mathbf{m}_k - \hat{\phi}_k(\mathbf{p}_{surf}))^2 + (\mathbf{m}_K - (1 - \phi_{\text{scene}}(\mathbf{p}_{surf})))^2 \tag{6.9}$$

Here, $\mathbf{m}_K$ is the redundant scope. The last term in eq. (6.9) encourages the redundant scope to treat points that have a low observation likelihood as not explained points. We also apply the sparsity loss $\mathcal{L}_{\text{sparsity}}$ introduced in [180] to encourage the model to choose empty space when no observations are available, e.g. the space beneath the table in the table-top scene. The total loss is computed as an aggregation of all the aforementioned losses: $\mathcal{L} = \mathcal{L}_{\text{empty}} + \mathcal{L}_{\text{obs}} + \mathcal{L}_{\text{shape}} + \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{sparsity}}$.

6.4 Experiments

In this section, we describe the experimental setup and hardware employed in our experiments, the performance metrics used to evaluate the effectiveness of our model alongside baselines, and present the primary outcomes of our study. Specifically, we investigate the following questions: (i) Does DreamUp3D achieve improved performance for scene understanding compared to NeRFs, especially in terms of inference time and reconstruction accuracy? (ii) Compared to pre-trained object-centric features, can the object-centric latent representation improve performance on downstream tasks such as object matching? and, (iii) Does DreamUp3D with shape completion have better pose estimation compared to other unsupervised, object-centric pose estimation methods in real-world scenarios?

6.4.1 Experimental Setup

Data collection and real-world testing of DreamUp3D utilised a 7-DoF Franka Panda manipulator equipped with an Intel RealSense camera. The system also included a computer with an NVIDIA GeForce RTX 3090 GPU. A subset of YCB objects, including the mustard bottle, fish can, banana, apple, lemon, orange, potted meat, cracker box, pringles, chocolate pudding box, soup can, sponge, gelatin box and spray bottle, were employed for experiments. Each scene collected in the training set consisted of two to four objects randomly arranged on the table (see Figure 6.3). The objects were arranged in graspable poses with inter-objects occlusions. For model training, 200 scenes were collected with each scene captured from four viewpoints, providing camera extrinsics and RGB-D images as model inputs. For assessing 3D reconstruction performance from a single viewpoint, a test set of scenes following the same configuration as the training set were collected. Training of the model took approximately 3 days on a single NVIDIA RTX A6000 GPU.

6.4.2 Scene Reconstruction

In this section, we assess the scene reconstruction capabilities of DreamUp3D compared to a recent state-of-the-art baseline. Additionally, we demonstrate DreamUp3D's

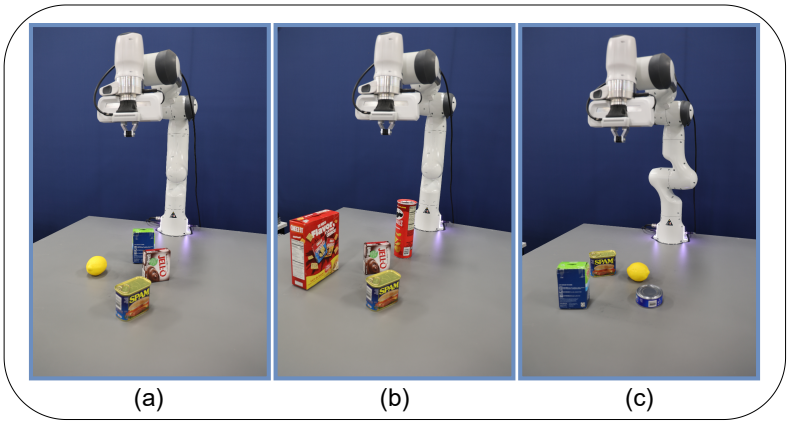


Figure 6.3: Example configurations used in the experiments. Panels depict the 7-DoF Franka Panda robot together with YCB objects randomly selected and configured on the tabletop.

capacity to *imagine* missing parts in the scene in an object-centric fashion based on only a single-view.

Metric. To evaluate the 3D reconstruction capabilities of our model, we use the directed Hausdorff distance (DHD) as our metric.

$$d_H(A, B) = \sum_{a \in A} \min_{b \in B} \|a - b\|. \tag{6.10}$$

In our case, A is the observed ground truth point set, B is the reconstructed point set, and the $\|\cdot\|$ represents the Euclidean distance. Note that, the directed Hausdorff distance only performs the calculation based on the points from the original ground truth point set as reference. This is different from the Chamfer distance, a metric commonly used to measure the dissimilarity between two sets of points. We use the directed Hausdorff distance instead of the Chamfer distance as the point set B generated by DreamUp3D can contain *imagined* points that do not exist in set A (see Fig. 6.4). The absence of reference points in the original set prevents the comparison of minimum distances for these generated points using the Chamfer distance.

Baselines. We use *depth-nerfacto* from *nerfstudio* [181] as one of our baselines. According to the official documentation, the model is not based on any single existing published work. Instead, it integrates numerous published methods that have been identified as highly effective for reconstructing real data when used jointly. In particular, the model combines camera pose refinement, per-image appearance

conditioning, proposal sampling, scene contraction, and hash encoding to accelerate training. In terms of 3D reconstruction, we compare our model to *depth-nerfacto* using several different configurations. We provide the NeRF with 8, 16, and 32 views for reconstruction and optimize it for between 5000 to 15000 training iterations. We also compare our model to another OCGM, ObSuRF [167]. ObSuRF also employs generalizable NeRFs as the object decoder but encodes the spatial information and object appearance into a single latent embedding, preventing the estimation of a 6D pose.We compare the directed Hausdorff distance between ground truth and reconstructions across 12 test scenes, assessing the time taken by each model to reconstruct a given scene. The 12 test scenes are collected in the same way as the training scenes, although, with unseen, random configurations.

Reconstruction Results. DreamUp3D only needs to be trained once and does not necessitate retraining when the tabletop configuration is altered. In contrast, *depth-nerfacto* lacks the ability to generalise and requires retraining for each modified scene configuration. Additionally, DreamUp3D can reconstruct the scene from a single-view RGB-D image and *imagine* the 3D shapes, while *depth-nerfacto* requires multiple views as inputs for each new scene. The results are presented in Table 6.1[1]. It is evident that DreamUp3D achieves significantly faster test time inference, taking only a fraction of a second, whilst also exhibiting superior reconstruction performance, surpassing the performance of the baseline method by an order of magnitude. This improved test-time efficiency of DreamUp3D satisfies the outlined desiderata for real-time robot operations not exhibited by NeRFs. We observe that ObSuRF, another OCGM, also elicits fast test time inference, however, it does not perform as well as DreamUp3D in reconstruction accuracy.

Example RGB and depth reconstructions are depicted in Figure 6.4. Note that in the case of Figure 6.4 (a), it can be observed that part of the objects in the input image are out of frame. DreamUp3D demonstrates its ability to *imagine* the

[1]The NeRFs are off-the-shelf models without any modifications, we observe that providing more views or additional training time does not necessarily improve performance. This indicates that training for 5000 iterations is sufficient.

Model	Views ↓	Fitting steps	Test Time (s) ↓	DHD ↓
DreamUp3D	1	0	**0.24 ± 0.01**	**0.0075 ± 0.0010**
DreamUp3D (FG only)	1	0	**0.23 ± 0.01**	0.0081 ± 0.0023
ObSuRF	1	0	**0.01 ± 0.01**	0.0130 ± 0.0060
NeRF	8	5000	127.41 ± 1.11	0.0351 ± 0.0086
NeRF	16	5000	129.12 ± 1.31	0.0384 ± 0.0077
NeRF	32	5000	128.02 ± 1.34	0.0423 ± 0.0057
NeRF	32	10000	245.73 ± 2.11	0.0463 ± 0.0079
NeRF	32	15000	364.83 ± 1.33	0.0501 ± 0.0109

Table 6.1: 3D reconstruction results for DreamUp3D vs NeRF and ObSuRF baselines. DreamUp3D outperforms the NeRF baselines with up to 32 views and 15000 fitting steps both in terms of speed and reconstruction error (distance to ground-truth). The results for foreground reconstruction is summarised in DreamUp3D (FG only). DreamUp3D also outperforms ObSuRF in terms of reconstruction accuracy.

missing parts of the objects, this characteristic can be attributed to the object-centric scene factorisation of the model. Concretely, DreamUp3D first recognises the partially observed objects in the scene using the learned object encoder and reconstructs the full shape via the learned object decoder. Also, in Figure 6.5, we compare the ground truth point cloud to those reconstructed by *depth-nerfacto* and DreamUp3D. All images are taken from the same camera view. It is evident that DreamUp3D captures the underlying point cloud geometry and objects with greater precision (see Table 6.1). ObSuRF encodes spatial information and appearance information of the object into a single latent embedding, which poses a higher requirement on the learning capacity. We thus find that despite ObSuRF's ability to reconstruct the scene, it fails to segment the scene into meaningful objects, leading to strong reconstruction accuracy although without sufficient scene understanding.

6.4.3 Object-centric Representation

We consider an object matching task to evaluate the real-world applicability of the learned object-centric representations.

Metric. For the object matching task, we report the matching accuracy as our metric.

Baselines. We first conduct an evaluation of our model by comparing it with recent visual-semantic matching techniques, namely CLIPSEMFEAT-N [182], which utilises the pre-trained CLIP model [183] for object matching. Following [183], an

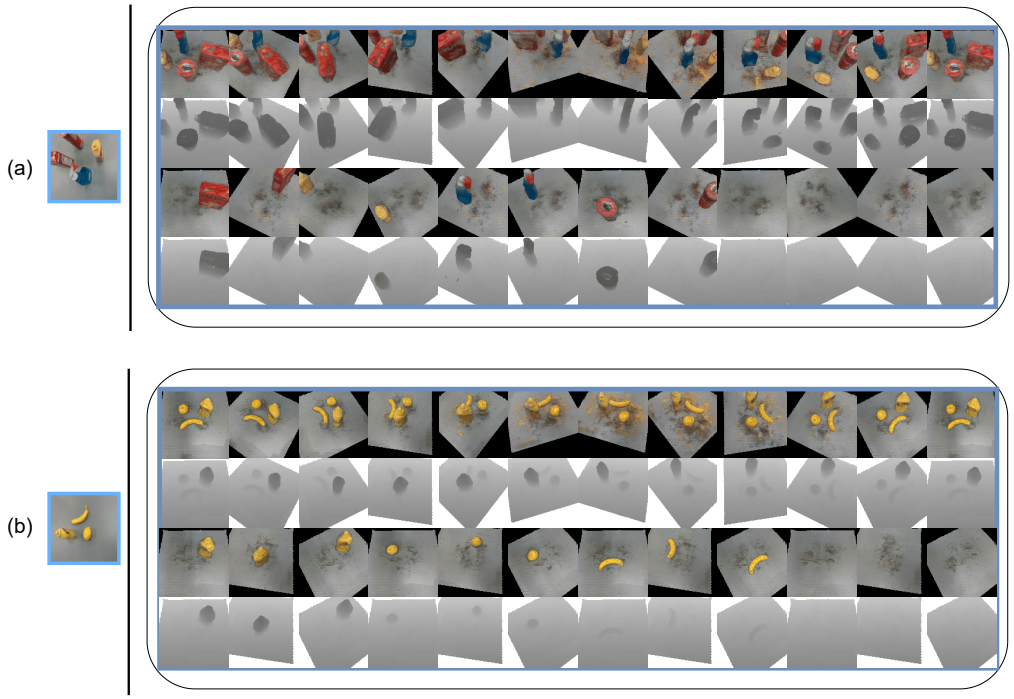


Figure 6.4: Given a single view of the scene, DreamUp3D produces full scene reconstructions from arbitrary vantage points. For two examples, (a) and (b), the top two rows show the scene reconstructions for RGB and depth respectively from various viewpoints. The following two rows show RGB and depth reconstructions for individual objects and the background components. Example (a) demonstrates the ability of DreamUp3D to reconstruct the shapes of the cracker box and the the chips despite being partially out of view in the input image.

instance segmentation network [184] first extracts crops of all the objects present in both the current scene and the given goal image. Cropped objects are then encoded into visual features using the CLIP model. A classification matrix is constructed as the dot product between the normalised visual features and the semantic features acquired by encoding the known class names into a text embedding using the text encoder of the CLIP model. For DreamUp3D, we directly compute the L2-distance and the cosine distance (dot product) of the normalised object-centric latent representations for object matching. We evaluate our model and baselines on a tabletop scene involving multiple objects. For each experiment, we randomly select two to four objects from the YCB dataset, with evaluations conducted across 20 unique scenes in the real-world.

Object Matching Results. The quantitative results for our object-matching

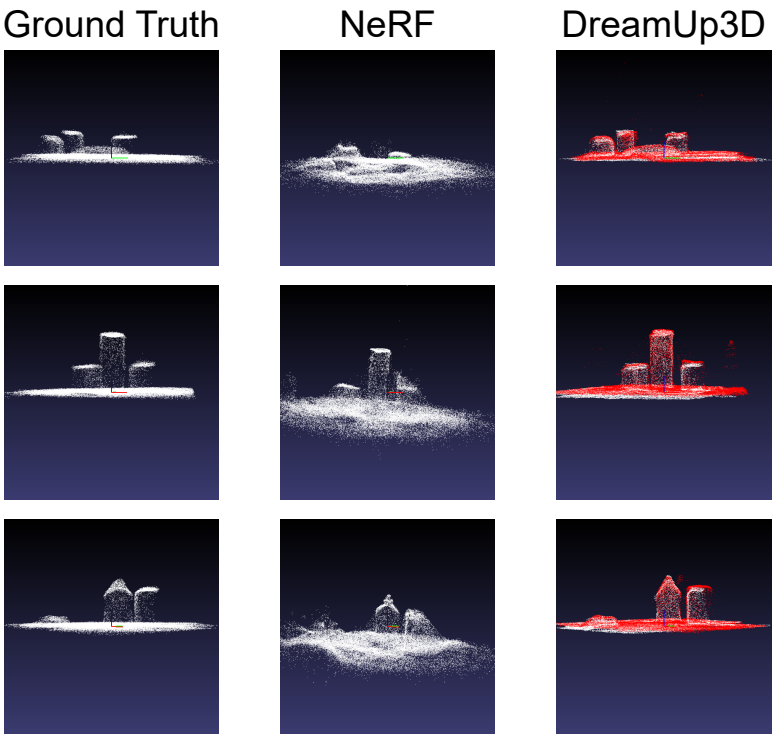


Figure 6.5: Comparison of point cloud ground truth, NeRF reconstruction using 32 views, and DreamUp3D reconstruction using a single view image as input. In the case of DreamUp3D we superimpose the ground truth (white) to the reconstruction (red). All images are taken from the same view point with the size of the point clouds increased for visual clarity.

experiment are summarised in Table 6.2. We observed that the CLIPSEMFEAT-N approach fails to perform object matching well in our experiment as the CLIPSEMFEAT-N approach requires semantically distinguishable visual inputs. However, objects such as the potted meat can only expose a metal surface from the fixed top-down view, which is required by the GG-CNN grasping planner [185] used in [182]. This partial observation leads to the low accuracy of the CLIPSEMFEAT-N approach. DreamUp3D, on the other hand, performs object matching using the object-centric latent representation learned from the scene reconstruction. Our approach thus does not require known object classes and allows for fully self-supervised learning. Using the learned object latent representation DreamUp3D achieves a significantly improved matching accuracy compared to CLIPSEMFEAT-N. However, we also find that matching among objects that share similar shapes, e.g. the chocolate pudding box and the strawberry gelatin box, can be erroneous, which indicates

the limitation of performing object matching using latent representations learned only from scene reconstruction. As an ablation, we include the accuracy of the L2-distance versus the cosine distance (dot product) for the object matching and see a notable improvement when using the cosine distance.

Model	Accuracy [%]
DreamUp3D (cosine distance)	57.6
DreamUp3D (L2-distance)	55.9
CLIPSEMFEAT-N	27.1

Table 6.2: Object matching results DreamUp3D vs CLIPSEMFEAT-N baseline.

6.4.4 Unsupervised Pose Estimation

In this section, we assess the unsupervised pose estimation capabilities of DreamUp3D compared to a recent state-of-the-art baseline, ObPose [168]. We also analyse the common failure modes of shape-based pose estimation caused by the partial observation of the scene.

Metric. Due to the symmetry of some objects, it is possible that there is no unique correct orientation, we thus evaluate the pose using the mean intersection over union (mIoU) accuracy of the predicted bounding boxes and the ground-truth bounding boxes with various thresholds [186, 187] as the metric.

Baseline. The pose estimation performance of DreamUp3D is evaluated compared to the ObPose model [168]. ObPose is the first OCGM that performs unsupervised 6D pose estimation but without the use of a shape completion module to predict the full object shape.

Pose Estimation Results. The quantitative results for pose estimation performance comparing ObPose [168] to DreamUp3D are summarised in Table 6.3. We observe two main failure modes of ObPose, occlusions and noisy scene observations, and demonstrate these in Figure 6.6. With occlusion, the observed object point clouds are not sufficient to accurately approximate the full object shape, which thus leads to an erroneously oriented minimum bounding box (see Figure 6.6 (a)). The noisy scene point clouds are caused by the accuracy-drop

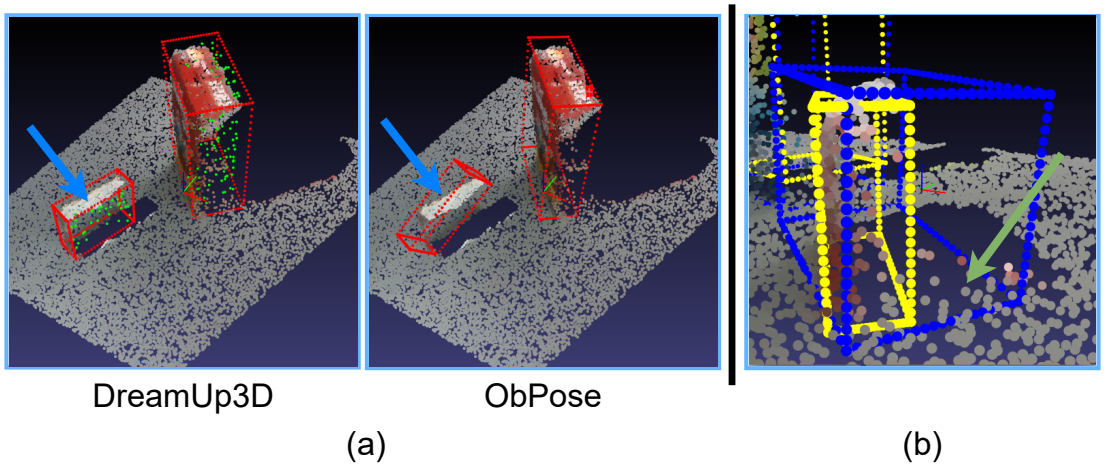


Figure 6.6: Object pose estimation in the real world. (a): The pose estimation with shape completion (DreamUp3D) and without shape completion (ObPose). The predicted object point clouds are visualised using green points. (b) The noisy point clouds projected from the depth observation at the edges of the objects, which become outliers for the minimum bounding box.

of the depth sensor at the object edges. Part of the object point clouds can be mistakenly projected to the table surface and therefore become outliers for the minimum bounding box (see Figure 6.6 (b)). To alleviate this problem, we randomly drop part of the observed point clouds to reduce the outliers and preserve the inpainted point clouds generated from the shape completion module. We observe a clear improvement as the observed object point clouds reduce (see Table 6.3). This indicates the proposed shape completion module can provide more robust shape information compared to the ObPose baseline.

Model	mIoU (0.3) ↑	mIoU (0.5) ↑	mIoU (0.7) ↑
DreamUp3D (SC only)	**0.97**	**0.75**	0.15
DreamUp3D (SC + 10% obs. pcds)	0.95	0.75	**0.17**
DreamUp3D (SC + 30% obs. pcds)	0.90	0.69	**0.17**
DreamUp3D (SC + 50% obs. pcds)	0.92	0.68	0.14
DreamUp3D (SC + 70% obs. pcds)	0.88	0.68	0.08
ObPose (obs. pcds only)	0.90	0.61	0.03

Table 6.3: Pose estimation results DreamUp3D vs ObPose baseline. The mean IoU (mIoU) accuracy with thresholds of $0.3, 0.5$ and 0.7 are reported. The pose estimation using less observed (obs.) point clouds (pcds) with shape completion (SC) achieves improved performance.

Leveraging the estimated object pose, we additionally demonstrate DreamUp3D's ability to flexibly rearrange real-world objects, with the use of object position and

orientation, in Figure 6.7.

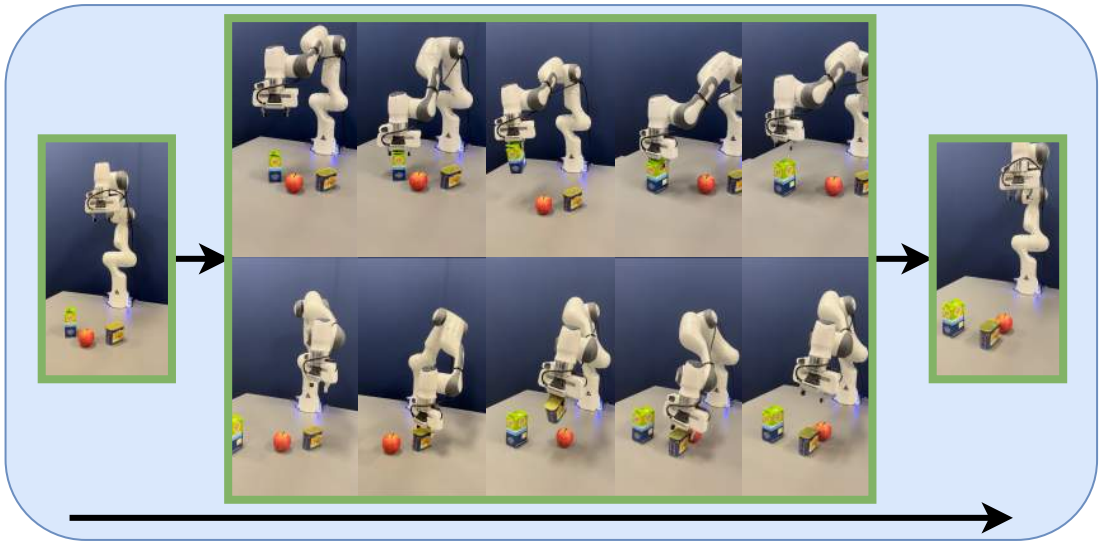


Figure 6.7: Object manipulation in the real world. The agent detects and moves the objects to their canonical poses using the inferred object poses. The robot arm moves the blue and green box away from the apple (first row, left to right). The robot arm moves the can from behind the apple to the front (second row, left to right).

6.4.5 Limitations

Although our method has the advantage of providing strong scene reconstructions, latent object embeddings and 6D pose estimation, the reconstruction could see significant improvements using an RGB-D sensor with greater accuracy. It is important to note that due to sensor limitations, point clouds captured from different angles do not perfectly overlap. This lack of alignment can partly be attributed to errors in camera pose estimation, which are influenced by the robot's forward kinematic accuracy and camera calibration accuracy. Additionally, depth errors can significantly impact the accuracy of our reconstructions, this typically occurs when scenes include shiny surfaces like metals and reflective plastics. These factors collectively contribute to the reconstruction error.

There is potential for DreamUp3D to generalise to out-of-distribution cases where scenes feature a greater number of objects than those seen in training thanks to the the IC-SBP algorithm and the object-centric learning paradigm. We test this hypothesis by collecting additional highly clustered scenes with 4-6 objects

including scenes with unseen objects not included in our training set. As expected, DreamUp3D does not generalise to scenes with unseen objects, achieving a DHD of 0.0105 ± 0.0039. This is a known limitation of OCGMs, as the encoder-based approach of single-view 3D reconstruction is essentially a recognition model [188]. Note that reconstructing unseen objects from single-view observation is a highly ill-posed task. Generalisation to scenes with a greater number of objects is also limited (DHD of 0.0090 ± 0.0021) likely caused by the closeness of objects within the test scenes. Both of these limitations could potentially be addressed by scaling up both data and computational resources to the model.

Finally, our model and experiments focus specifically on table-top scenes due to the RANSAC plan detection. Extending our model to more complex scenes in the wild requires the adaptation of the model and specifically the background modelling component. The hyperparameters such as the kernel sizes of the KPConv encoder and the GRAF network sizes are environment-dependent. They need to be chosen according to the sizes and the complexity of the scene.

6.5 Conclusion

We propose DreamUp3D as an efficient and robust approach for performing 3D object-centric scene inference, object-level representation learning, and 6D pose estimation. In contrast to related work, DreamUp3D achieves these tasks without the need for retraining on new scenes at test time and without requiring multiple views of a static scene. This makes DreamUp3D more readily suited to robotics tasks. Compared to recent baselines, we demonstrate DreamUp3D's improved reconstruction quality and its ability to *imagine* occluded or missing parts of objects in the input image.

Further work is needed to pursue reconstructions in challenging scenarios involving reflective surfaces, although acquiring accurate depth estimates in these conditions using RGB-D cameras is a well-known problem. In the context of object manipulation, incorporating 3D reconstruction for improved grasping presents a promising avenue for future research.

7 Discussion

OCGMs discover the structure in data by reconstructing observations and learning to represent data in a compact fashion. The structured representations of the scenes then lay the foundation for downstream tasks including object manipulation and task planning. In the previous chapters of this thesis, we have introduced several OCGMs and investigated their applications in various manipulation scenarios. In this final chapter we summarise the key contributions of the thesis and revisit the guiding questions discussed in Chapter 1. We also reflect on the limitations of the models proposed and point out future research directions to address the identified shortcomings.

7.1 Key Contributions

In Chapter 1, we have layed out the guiding questions this thesis investigates:

1. Given that generative models naturally capture factors of variation, could they also be used to expose these factors such that they can be modified in a task-driven way?

2. How well does the OCGM learn disentangled components from robot interaction datasets in a fully unsupervised fashion?

3. Can OCGMs explicitly model the 3D geometry of objects and the background while still being able to disentangle object spatial information from its appearance information as in the 2D OCGMs without using any known CAD models or templates of the objects?

4. Can the 3D OCGM be eventually applied to real-world robotics scenarios and achieve a suitably broad range of functionalities for scene understanding, such as the object detection, 6D pose estimation, object-centric latent representation learning and 3D reconstruction, in order to facilitate downstream robotic tasks still without using any human supervision?

In this thesis, we first demonstrate how to leverage a latent space learned from the tool-use experiences for the tool synthesis task. To simulate the trial-and-errors of the robot using the tool for the reaching task, a dataset of labelled feasibility of the given tools with respect to a task scene specified by a RGB image is collected and used to train an affordance predictor. Given a tool that is originally infeasible for the task, the model can leverage the trained affordance predictor to generate gradients to modify the latent embedding of the tool via activation maximisation. By decoding the modified tool latent embedding, we observed that the tool can morph from being infeasible to being feasible for the given task (see figure 3.5). This process endows the robot with the ability of task-driven tool synthesis. From the visualisation of the tool synthesis process, we found that morphing of tool shape is a continuous process. This indicates that there exists a trajectory in the latent space that corresponds to the high-level concept of object affordance. We also found an unseen T-shape tool can be created in some tasks (see the right-middle example of 3.4). This result suggests the model is able to identify the task-relevant part of the tool and modify that part of the tool intentionally.

Next, we explored the unsupervised OCGM performance on robotic tasks, where the scene contains objects of various shapes and textures including the robot arm itself. Experiments demonstrate that the unsupervised OCGM can still discover meaningful entities including the objects and robot arm, despite there being intensive interactions between the objects and the robot. The learned object-centric

latent representations can also be leveraged for the object matching in the object rearrangement task. Another interesting finding is that the segmentation results of the robot arm are different in the two dataset we conducted in the experiment. In the Panda pushing dataset, the robot arm is segmented as a single entity (see fig. 4.3), whereas the robot gripper is segmented into two separate components in the Sketchy dataset due to the relative motion between the two grippers for the pick-and-place action (see fig. 4.4). This result demonstrates that the correct segmentation is not unique if the optimisation target is only the reconstruction loss of the observations. This indicates the limitation of unsupervised scene segmentation learned from self-supervised reconstruction. To these ends, additional inductive bias is necessary to concretely define how the complex objects such as the robot arm should be properly segmented.

The applicability of 2D OCGMs to robotic tasks is limited by the 2D object decoder employed in the model as the robot manipulation is not only object-centric but is also in 3D. Moreover, the learned latent space in 2D OCGMs only capture the variations of the discovered objects in the projected image space, and therefore provide limited information compared to the latent space that directly models the object shapes in 3D. For 3D OCGMs, it is also beneficial to learn a disentangled spatial and appearance representation of the object as in the 2D OCGMs, but there is no available object template to pre-define the canonical 6D pose of the object. To address these problems, we propose ObPose, a model that leverages NeRFs as the object decoder that directly models the 3D geometry of the objects. To estimate the object pose without labels and object templates, ObPose uses a minimum volume heuristic to compute the object poses. Concretely, the model uses the smallest bounding box to contain the object and represents the object pose using the pose of that bounding box. To the best of our knowledge, ObPose is the first NeRF-based OCGM that is able to learn disentangled object pose and appearance representations. The proposed model thus allows the user to sample from the learned latent space to generate objects that always in the canonical poses. Consequently, the user can translate and rotate the generated objects into arbitrary targeted poses, which

provides additional flexibility for the scene editing. Moreover, we also found that despite trained on single-view RGB-D images, the NeRF decoder can still learn to approximately reconstruct the shapes of the objects, although having multi-view supervision can further improve the reconstruction accuracy if available.

In the last part of this thesis, we apply a 3D OCGM in real-world robotic tasks. We found that the heuristic-based pose estimation can be inaccurate when the object is occluded. We thus propose the model DreamUp3D, that introduces a shape completion module before the pose estimation step to inpaint the unobserved part of the objects. The shape completion module is trained in a computationally efficient way. Concretely, we employ a self distillation step to reuse the outputs of the NeRFs to provide the learning signal for the shape completion module. In the experiment, we evaluate the performance of DreamUp3D on several tasks, including the reconstruction accuracy and efficiency, the representation learning for object matching and the pose estimation. We find that DreamUp3D is more suitable as scene understanding module for robotic tasks compared to the NeRF baselines as it does not require retraining before every action. We also demonstrate that the estimated object poses can facilitate the object rearrangement by moving the object to not only targeted locations but also to targeted orientations. We hope that the proposed DreamUp3D can be a convenient vision backbone for the scene understanding integrated into the robot's perception pipeline.

7.2 Limitations

In the previous section, we summarised the key contributions made in this thesis and explained how each of them are evaluated with respect to the hypothesis postulated in Chapter 1. The experiments conducted in Chapter 3-6 demonstrate that OCGM can discover the structure in data in a unsupervised fashion and then be leveraged by robotic tasks. Nevertheless, it is also pivotal to point out the limitations of the current approaches to specify the scope of its application and to identify future research directions.

Firstly, although the proposed OCGMs employ different attention mechanisms to motivate the disentanglement of the scene components, the unsupervised instance segmentation still struggles to decompose scenes into meaningful components in scenes that has tightly-packed objects. The optimisation target used for the training of the OCGMs is the ELBO, which consists of a log-likelihood term for learning the reconstruction of the observations and a regularization term, such as KL divergence, to regularise the learned object-centric latent representations. The multivariate Gaussian is commonly chosen as the prior distribution, which encourages the latent representations to be disentangled along each dimension. However, this regularisation loss can not directly motivate the segmentation behavior of the model. The instance segmentation functionality is therefore a side-effect of maximising the ELBO. In [26], the authors point out that the reconstruction bottlenecks induced by this regularisation term appear to be a sufficient mechanism for discovering the object-like entities in the scene. Nevertheless, the size of the bottlenecks should adjust adaptively according to the visual complexity of the data to balance the trade off between satisfying reconstruction quality and segmentation accuracy. This leads to the difficulty of the hyper-parameter selection of the model. The above discussion indicates the necessity of introducing additional inductive biases that directly defines what is an object, i.e. the objectness.

Secondly, the OCGMs can not handle real-world data that has complex and dynamic background such as the dataset in the autonomous driving tasks. For other vision models, such as the models for the instance segmentation tasks, the models can learn to ignore the background pixels by learning object features from human supervision. However, learning to reconstruct is a more challenging task than learning to classify. Especially in outdoor scenarios, the street views can be totally different from the scenes in the training dataset, e.g. a street in another city that is not in the training dataset. The unexplained background pixels will then be taken as inputs to the downstream object modules and thus lead to erroneous learning signal for the learning of the object representations, including both the object spatial representations such as the object locations and orientations and

the object appearance representations. Learning to reconstruct scenes in the wild poses an extremely hard challenge to the modelling capacity of the decoder and the generalisation ability of the encoder. This thus makes us rethink the necessity of reconstructing all observations from the inputs despite the key learning signal it provides for the unsupervised learning framework.

Finally, in Chapter 6 we have demonstrated how a 3D OCGM can be applied to real-world robotic applications. The learning of a 3D decoder endows the robot with the ability to infer the object 3D shapes in the scenes within a second. The explicit 3D shape of the objects can then be leveraged by several downstream tasks such as path planing with collision avoidance, object grasping and 6D pose estimation. However, learning the 3D reconstruction of the objects and the backgrounds poses two main challenges. First, it is expensive to collect data for learning the 3D reconstruction of the scene. Although in Chapter 5 we showed that it is possible to learn the 3D shape given only single view inputs of each scene, the reconstruction quality can be drastically improved if multi-view observations of the same scene are available. But in Chapter 6, we found that collecting multi-view observations in the real-world takes a longer time for the robot arm to move from one observation pose to another. Moreover, the calibration error can accumulate from the camera calibration error and the forward kinematics error, which further increases the cost of data collection. Second, the recently proposed GRAF[46] as a generative NeRF can learn to map the object-centric latent representations to an implicit representation of the objects in 3D. But there are not many works that have conducted investigations into the model capacity of the GRAF. To improve the decoder capacity for modelling more complex dataset, recent work [189] proposes to first generate randomly posed 2D images and then train a NeRF or diffusion model [190] from scratch to reconstruct the 3D geometry of the generated objects. However, this is computationally expensive and cannot be integrated into the training loop of the OCGMs. A more powerful generative 3D decoder and a thorough evaluation of its capacity for learning from a large-scale 3D dataset is thus important for developing stronger OCGMs.

7.3 Future Work

Regarding the bottleneck of unsupervised segmentation performance of the OCGMs, using semi-supervised learning could be an approach that drastically broadens the applicability of the OCGMs. Recent works [191] have proposed foundation models, e.g. the Segment Anything Model (SAM), for instance segmentation trained on large amount of data. The SAM can be prompted with various inputs, including boxes, or text, making it flexible to be integrated into other applications. The foundation models can perform zero-shot generalisation to unseen objects, without the need for additional training. The segmentation performance of the OCGM can thus be improved by using the segmentation predictions from the foundation model as labels to train the attention module in the model. Moreover, the pre-trained encoder of SAM can also be integrated into the OCGMs as the SAM has learned a general notion of what objects are. The learning of the object segmentation in OCGMs can thus be facilitated by directly training on the predicted embedding from the SAM encoder. When the segmentation performance of the OCGMs is drastically improved, the learning tasks of the OCGMs can thus focus on the object pose estimation, the object-centric latent representation learning, and 3D reconstruction. Moreover, the accurate object masks also contribute to the data collection for training the object-VAE in the OCGMs. This could motivate the investigation on the generalisation ability of the object encoder and the decoder, which has not been thoroughly conducted in previous works. Another approach to improve the segmentation performance is to train the attention module not directly using the raw observation (e.g. image pixels or point clouds) but with the pre-trained features such as DINO [73] or CLIP [192] features. Recent work [72] has investigated the performance of an OCGM trained on DINO features for complex real-world scenarios.

For robotic tasks, in addition to the functionalities such as the object detection, pose estimation, representation learning and 3D reconstruction, that the current OCGM can provide, the modelling of the interactions between the objects is also a valuable puzzle. Given the segmented objects in the scene, the agent can simulate the outcome of a given sequence of actions based on the Newtonian physics. This

prediction ability thus facilitates the control-level planning such as MPC. One way to endow the robot with the ability of physical-aware predictions is to reuse the physics engine from existing simulation software. On this basis, the inputs to the simulation include the detected objects and their 3D meshes, which can be inferred from the scene observation using our proposed 3D OCGM. We thus hope our model might ultimately unlock the control-level planning ability by integrating the existing physics engine into the current pipeline.

7.4 Closing Remarks

In this thesis, we have proposed several 2D and 3D OCGMs for robotic tasks and investigated tool synthesis tasks utilising a learned latent space of various tools and an affordance predictor trained from tool-use experiences. We demonstrated that there exists a task-relevant trajectory in latent space that corresponds to a high-level concept of affordance. For the OCGMs, we find that the unsupervised OCGMs can be applied to robotic tasks despite there existing intensive interactions between the objects and the robot arm. The inconsistent segmentation results of the robot arms in the two different manipulation tasks indicates that the model needs additional inductive bias to define the desirable segmentation performance within the unsupervised learning framework. We next upgrade the 2D OCGM to 3D and achieve disentanglement of the spatial information from the appearance information of the objects via introducing the minimum volume principle for self-supervised object pose estimation in 3D. In the last part of this thesis, we deploy the 3D OCGM in a real-world robotics scenario and evaluate its performance in terms of 3D reconstruction, pose estimation and representation learning.

In conclusion, this thesis contributes to self-supervised, object-centric scene understanding models in 2D and 3D and demonstrated the existence of task-relevant manifold in the object latent space. It is our hope that the OCGMs proposed in this thesis can serve as the perception module that facilitates the scene understanding for the robotic applications. Finally, we believe that our findings

could potentially inspire future research with respect to real-to-sim and unsupervised scene understanding in more challenging real-world scenarios.

References

[1] Yu Xiang et al. “PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).

[2] Chen Wang et al. “Densefusion: 6d object pose estimation by iterative dense fusion”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3343–3352.

[3] Kaiming He et al. “Mask R-CNN”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[4] Van-Duc Nguyen. “Constructing force-closure grasps”. In: *The International Journal of Robotics Research* 7.3 (1988), pp. 3–16.

[5] Andreas ten Pas and Robert Platt. “Using geometry to detect grasps in 3d point clouds”. In: *arXiv preprint arXiv:1501.03100* (2015).

[6] Yizhe Wu et al. “Imagine That! Leveraging Emergent Affordances for Tool Synthesis in Reaching Tasks”. In: *arXiv preprint arXiv:1909.13561* (2019).

[7] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[8] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[9] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[10] Christopher P Burgess et al. “MONet: Unsupervised Scene Decomposition and Representation”. In: *arXiv preprint arXiv:1901.11390* (2019).

[11] SM Eslami et al. “Attend, infer, repeat: Fast scene understanding with generative models”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016).

[12] Adam Kosiorek et al. “Sequential attend, infer, repeat: Generative modelling of moving objects”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[13] Jindong Jiang et al. “SCALOR: Generative world models with scalable object representations”. In: *arXiv preprint arXiv:1910.02384* (2019).

[14] Kahneman Daniel. *Thinking, fast and slow*. 2017.

[15] AMP von Bayern et al. “Compound tool construction by New Caledonian crows”. In: *Scientific reports* 8.1 (2018), p. 15676.

[16] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[17] Yizhe Wu et al. “APEX: Unsupervised, object-centric scene segmentation and tracking for robot manipulation”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 3375–3382.

[18] Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. “Decomposing 3d scenes into objects via unsupervised volume segmentation”. In: *arXiv preprint arXiv:2104.01148* (2021).

[19] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. “Unsupervised Discovery of Object Radiance Fields”. In: *ArXiv* abs/2107.07905 (2021).

[20] James J Gibson. “The theory of affordances”. In: *Hilldale, USA* 1.2 (1977), pp. 67–82.

[21] Shichen Liu et al. “Soft rasterizer: A differentiable renderer for image-based 3d reasoning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7708–7717.

[22] Dumitru Erhan et al. “Visualizing higher-layer features of a deep network”. In: *University of Montreal* 1341.3 (2009), p. 1.

[23] Irina Higgins et al. “beta-vae: Learning basic visual concepts with a constrained variational framework”. In: *International conference on learning representations*. 2016.

[24] Christopher P Burgess et al. “Understanding disentangling in beta-VAE”. In: *arXiv preprint arXiv:1804.03599* (2018).

[25] Alexander L Mitchell et al. “First steps: Latent-space control with semantic constraints for quadruped locomotion”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5343–5350.

[26] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. “Reconstruction bottlenecks in Object-Centric generative models”. In: *arXiv preprint arXiv:2007.06245* (2020).

[27] Zhixuan Lin et al. “Space: Unsupervised object-oriented scene representation via spatial attention and decomposition”. In: *arXiv preprint arXiv:2001.02407* (2020).

[28] Berk Calli et al. “The ycb object and model set: Towards common benchmarks for manipulation research”. In: *2015 international conference on advanced robotics (ICAR)*. IEEE. 2015, pp. 510–517.

[29] Serkan Cabi et al. “Scaling Data-Driven Robotics with Reward Sketching and Batch Reinforcement Learning”. In: *arXiv preprint arXiv:1909.12200* (2019).

[30] Menglong Zhu et al. “Single image 3D object detection and pose estimation for grasping”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3936–3943.

[31] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. “GENESIS-v2: Inferring unordered object representations without iterative refinement”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021).

[32] Justin Kerr et al. “Evo-NeRF: Evolving nerf for sequential robot grasping of transparent objects”. In: *6th Annual Conference on Robot Learning*. 2022.

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

[34] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. 2014.

[35] Michael I Jordan et al. "An Introduction to Variational Methods for Graphical Models". In: *Machine learning* 37.2 (1999), pp. 183–233.

[36] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[38] SHI Xingjian et al. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 802–810.

[39] Hugues Thomas et al. "Kpconv: Flexible and deformable convolution for point clouds". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6411–6420.

[40] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.

[41] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. "GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement". In: *arXiv preprint arXiv:2104.09958* (2021).

[42] Francesco Locatello et al. "Object-Centric Learning with Slot Attention". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[43] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Computer Science* (2014).

[44] Nicholas Watters et al. "Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes". In: *arXiv preprint arXiv:1901.07017* (2019).

[45] Martin Engelcke et al. "GENESIS: Generative scene inference and sampling with object-centric latent representations". In: *arXiv preprint arXiv:1907.13052* (2019).

[46] Katja Schwarz et al. "Graf: Generative radiance fields for 3d-aware image synthesis". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20154–20166.

[47] Anh Nguyen et al. "Detecting object affordances with convolutional neural networks". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 2765–2770.

[48] Thanh-Toan Do, Anh Nguyen, and Ian Reid. "AffordanceNet: An end-to-end deep learning approach for object affordance detection". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 5882–5889.

[49] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[50] Anh Nguyen et al. “Object-based affordances detection with convolutional neural networks and dense conditional random fields”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 5908–5915.

[51] Spyridon Thermos et al. “Deep affordance-grounded sensorimotor object recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6167–6175.

[52] Sachin Mehta et al. “ESPNetv2: A light-weight, power efficient, and general purpose convolutional neural network”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9190–9200.

[53] Timo Lueddecke, Tomas Kulvicius, and Florentin Woergoetter. “Context-based affordance segmentation from 2D images for robot actions”. In: *Robotics and Autonomous Systems* 119 (2019), pp. 92–107.

[54] Xue Zhao, Yang Cao, and Yu Kang. “Object affordance detection with relationship-aware network”. In: *Neural Computing and Applications* 32.18 (2020), pp. 14321–14333.

[55] Shengheng Deng et al. “3d AffordanceNet: A benchmark for visual object affordance understanding”. In: *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1778–1787.

[56] David Inkyu Kim and Gaurav S Sukhatme. “Semantic labeling of 3d point clouds with object affordance for robot manipulation”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 5578–5584.

[57] Jiaqi Jiang et al. “A4T: Hierarchical affordance detection for transparent objects depth reconstruction and manipulation”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9826–9833.

[58] Karl Stelzner, Robert Peharz, and Kristian Kersting. “Faster attend-infer-repeat with tractable probabilistic models”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5966–5975.

[59] Eric Crawford and Joelle Pineau. “Spatially invariant unsupervised object detection with convolutional neural networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 3412–3420.

[60] Rishabh Kabra et al. “Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20146–20159.

[61] Klaus Greff et al. “Multi-Object Representation Learning with Iterative Variational Inference”. In: *International Conference on Machine Learning (ICML)*. 2019.

[62] Mehdi SM Sajjadi et al. “Object scene representation transformer”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9512–9524.

[63] Allan Jabri et al. “DORSal: Diffusion for Object-centric Representations of Scenes *et al.*” In: *arXiv preprint arXiv:2306.08068* (2023).

[64] Ziyi Wu et al. “Slotformer: Unsupervised visual dynamics simulation with object-centric models”. In: *arXiv preprint arXiv:2210.05861* (2022).

[65] Gamaleldin Elsayed et al. "Savi++: Towards end-to-end object-centric learning from real-world videos". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 28940–28954.

[66] Rishi Veerapaneni et al. "Entity Abstraction in Visual Model-Based Reinforcement Learning". In: *Conference on Robot Learning (CoRL)* (2020).

[67] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. "Simple unsupervised object-centric learning for complex and naturalistic videos". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 18181–18196.

[68] Hiroharu Kato et al. "Differentiable rendering: A survey". In: *arXiv preprint arXiv:2006.12057* (2020).

[69] Yizhe Wu, Oiwi Parker Jones, and Ingmar Posner. "Obpose: Leveraging canonical pose for object-centric scene inference in 3d". In: *arXiv preprint arXiv:2206.03591* (2022).

[70] Hong-Xing Yu, Leonidas J Guibas, and Jiajun Wu. "Unsupervised discovery of object radiance fields". In: *arXiv preprint arXiv:2107.07905* (2021).

[71] Michael Niemeyer and Andreas Geiger. "Giraffe: Representing scenes as compositional generative neural feature fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11453–11464.

[72] Maximilian Seitzer et al. "Bridging the gap to real-world object-centric learning". In: *arXiv preprint arXiv:2209.14860* (2022).

[73] Mathilde Caron et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[74] Ziyi Wu et al. "SlotDiffusion: Object-Centric Generative Modeling with Diffusion Models". In: *arXiv preprint arXiv:2305.11281* (2023).

[75] Jindong Jiang et al. "Object-centric slot diffusion". In: *arXiv preprint arXiv:2303.10834* (2023).

[76] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[77] Martin Engelcke et al. "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *arXiv preprint arXiv:1907.13052* (2019).

[78] Vadim Tikhanoff et al. "Exploring affordances and tool use on the iCub". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013, pp. 130–137.

[79] A. Myers et al. "Affordance detection of tool parts from geometric features". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1374–1381.

[80] Y. Liu et al. "Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1118–1125.

[81] H. Grabner, J. Gall, and L. Van Gool. "What makes a chair a chair?" In: *Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 1529–1536.

[82] Thanh-Toan Do et al. "AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[83] Stanley H Ambrose. "Paleolithic technology and human evolution". In: *Science* 291.5509 (2001), pp. 1748–1753.

[84] Nathan J Emery and Nicola S Clayton. "Tool use and physical cognition in birds and mammals". In: *Current Opinion in Neurobiology* 19.1 (2009), pp. 27–33.

[85] Dumitru Erhan et al. *Visualizing higher-layer features of a deep network*. Tech. rep. University of Montreal, 2009.

[86] Matthew D Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Proceedings of the IEEE European Conference on Computer Vision*. 2014, pp. 818–833.

[87] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *Workshop at International Conference on Learning Representations*. 2014.

[88] Irina Higgins et al. "β-VAE: Learning basic visual concepts with a constraied variational framework". In: *ICLR*. 2017.

[89] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. *Inceptionism: Going Deeper into Neural Networks*. 2015. URL: `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`.

[90] Alexander Stoytchev. "Behavior-Grounded Representation of Tool Affordances". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2005, pp. 3071–3076.

[91] H. Kjellström, J. Romero, and D. Kragic. "Visual Object-Action Recognition: Inferring Object Affordances from Human Demonstration". In: *Computer Vision and Image Understanding* (2010), pp. 81–90.

[92] Tanis Mar et al. "Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3200–3206.

[93] R. Girdhar et al. "Learning a Predictable and Generative Vector Representation for Objects". In: *ECCV*. 2016.

[94] Jiajun Wu et al. "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 82–90.

[95] Amit Kohli, Vincent Sitzmann, and Gordon Wetzstein. "Inferring Semantic Information with 3D Neural Scene Representations". In: *arXiv preprint arXiv:2003.12673* (2020).

[96] Wang Yifan et al. "Neural Cages for Detail-Preserving 3D Deformations". In: *arXiv preprint arXiv:1912.06395* (2019).

[97] Fabian B Fuchs et al. “Scrutinizing and De-Biasing Intuitive Physics with Neural Stethoscopes”. In: *arXiv preprint arXiv:1806.05502* (2018).

[98] Lorenzo Jamone et al. “Learning Object Affordances for Tool Use and Problem Solving in Cognitive Robots”. In: *Proceedings of the 2nd Italian Workshop on Artificial Intelligence and Robotics*. 2015, pp. 68–82.

[99] Kuniyuki Takahashi et al. “Tool-body assimilation model considering grasping motion through deep learning”. In: *Robotics and Autonomous Systems* 91 (2017), pp. 115–127.

[100] Kuan Fang et al. “Learning task-oriented grasping for tool manipulation from simulated self-supervision”. In: *arXiv preprint arXiv:1806.09266* (2018).

[101] Yixin Zhu, Yibiao Zhao, and Song Chun Zhu. “Understanding Tools: Task-Oriented Object Modeling, Learning and Recognition”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[102] Namiko Saito et al. “Tool-Use Model Considering Tool Selection by a Robot Using Deep Learning”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 270–276.

[103] Annie Xie et al. “Improvisation through Physical Understanding: Using Novel Objects as Tools with Visual Foresight”. In: *arXiv preprint arXiv:1904.05538* (2019).

[104] Marc Toussaint et al. “Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning”. In: *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, 2018.

[105] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. “Neural 3D mesh renderer”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3907–3916.

[106] Nanyang Wang et al. “Pixel2mesh: Generating 3D mesh models from single RGB images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–67.

[107] 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).

[108] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[109] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 28 (2015).

[110] Martin Engelcke et al. “GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations”. In: *International Conference on Learning Representations (ICLR)* (2020).

[111] Jindong Jiang et al. “SCALOR: Generative World Models with Scalable Object Representations”. In: *International Conference on Learning Representations (ICLR)* (2020).

[112] Markus Wulfmeier et al. “Representation Matters: Improving Perception and Exploration for Robotics”. In: *arXiv preprint arXiv:2011.01758* (2020).

[113] Nicholas Watters et al. “COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration”. In: *arXiv preprint arXiv:1905.09275* (2019).

[114] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *International Conference on Machine Learning (ICML)* (2014).

[115] Klaus Greff et al. “Tagger: Deep unsupervised perceptual grouping”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016).

[116] Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. “Neural expectation maximization”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[117] Sjoerd Van Steenkiste et al. “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions”. In: *arXiv preprint arXiv:1802.10353* (2018).

[118] Jonathan Huang and Kevin Murphy. “Efficient Inference in Occlusion-Aware Generative Models of Images”. In: *arXiv preprint arXiv:1511.06362* (2015).

[119] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. “Objects as Points”. In: *arXiv preprint arXiv:1904.07850* (2019).

[120] Zhixuan Lin et al. “Improving Generative Imagination in Object-Centric World Models”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6140–6149.

[121] *IROS 2020: Open Cloud Robot Table Organization Challenge (OCRTOC)*. 2020. URL: `http://www.ocrtoc.org/%5C#/`.

[122] Sébastien Ehrhardt et al. “RELATE: Physically plausible multi-object scene synthesis using structured latent spaces”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 11202–11213.

[123] Thu H Nguyen-Phuoc et al. “BlockGAN: Learning 3D object-aware scene representations from unlabelled images”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 6767–6778.

[124] Michael Niemeyer and Andreas Geiger. “GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields”. In: *arXiv preprint arXiv:2011.12100* (2020).

[125] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. “Tracking Objects as Points”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[126] Fisher Yu et al. “Deep Layer Aggregation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[127] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: *International Conference on Learning Representations (ICLR)* (2017).

[128] Yingzhen Li and Yarin Gal. "Dropout inference in Bayesian neural networks with alpha-divergences". In: *International conference on machine learning*. PMLR. 2017, pp. 2052–2061.

[129] Berk Calli et al. "Yale-CMU-Berkeley Dataset for Robotic Manipulation Research". In: *International Journal of Robotics Research (IJRR)* 36.3 (2017), pp. 261–268.

[130] William M Rand. "Objective Criteria for the Evaluation of Clustering Methods". In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850.

[131] Marissa A. Weis et al. "Unmasking the Inductive Biases of Unsupervised Object Representations for Video Sequences". In: *arXiv preprint arXiv:2006.07034* (2020).

[132] Anton Milan et al. "MOT16: A Benchmark for Multi-Object Tracking". In: *arXiv preprint arXiv:1603.00831* (2016).

[133] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)* (2015).

[134] Cathrin Elich et al. "Weakly supervised learning of multi-object 3D scene decompositions using deep shape priors". In: *Computer Vision and Image Understanding* 220 (2022), p. 103440.

[135] Haozhe Xie et al. "Pix2vox: Context-aware 3D reconstruction from single and multi-view images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2690–2698.

[136] Ben Mildenhall et al. "NeRF: Representing scenes as neural radiance fields for view synthesis". In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–421.

[137] Justin Johnson et al. "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2901–2910.

[138] Angel X. Chang et al. "ShapeNet: An Information-Rich 3D Model Repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[139] Taufik Xu et al. "Multi-objects generation with amortized structural regularization". In: *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019).

[140] Marissa A Weis et al. "Unmasking the inductive biases of unsupervised object representations for video sequences". In: *arXiv preprint arXiv:2006.07034* 2 (2020).

[141] Thomas Kipf et al. "Conditional Object-Centric Learning from Video". In: *arXiv preprint arXiv:2111.12594* (2021).

[142] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.

[143] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* (2021). https://distill.pub/2021/gnn-intro.

[144] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[145] He Wang et al. “Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation”. In: *arXiv preprint arXiv:1901.02970* (2019).

[146] Dengsheng Chen et al. “Learning canonical shape space for category-level 6D object pose and size estimation”. In: *Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 11973–11982.

[147] Xiaolong Li et al. “Category level articulated object pose estimation”. In: *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 3706–3715.

[148] Chen Wang et al. “6-pack: Category-level 6d pose tracker with anchor-based keypoints”. In: *International Conference on Robotics and Automation (ICRA)* (2020), pp. 10059–10066.

[149] Xu Chen et al. “Category level object pose estimation via neural analysis-by-synthesis”. In: *European Conference on Computer Vision* (2020), pp. 139–156.

[150] Meng Tian, Marcelo H. Ang, and Gim Hee Lee. “Shape prior deformation for categorical 6D object pose and size estimation”. In: *European Conference on Computer Vision* (2020), pp. 530–546.

[151] Xiaolong Li et al. “Leveraging SE(3) Equivariance for Self-Supervised Category-Level Object Pose Estimation”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[152] Walter Goodwin et al. “Zero-Shot Category-Level Object Pose Estimation”. In: *arXiv preprint arXiv:2204.03635* (2022).

[153] Kieran Murphy et al. “Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold”. In: *arXiv preprint arXiv:2106.05965* (2021).

[154] Anna Yershova et al. “Generating uniform incremental grids on SO(3) using the Hopf fibration”. In: *The International Journal of Robotics Research* 29.7 (2010), pp. 801–812.

[155] Krzysztof M Gorski et al. “HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere”. In: *The Astrophysical Journal* 622.2 (2005), p. 759.

[156] Lingjie Liu et al. “Neural sparse voxel fields”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 15651–15663.

[157] Lawrence Hubert and Phipps Arabie. “Comparing partitions”. In: *Journal of Classification* 2.1 (1985), pp. 193–218.

[158] Kyle Gao et al. “NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review”. In: *ArXiv* abs/2210.00379 (2022).

[159] Justin Kerr et al. “Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects”. In: *6th Annual Conference on Robot Learning*.

[160] Jianlin Liu et al. “NeRF-Loc: Visual Localization with Conditional Neural Radiance Field”. In: *arXiv preprint arXiv:2304.07979* (2023).

[161] S Zhong et al. “Touching a NeRF: Leveraging Neural Radiance Fields for Tactile Sensory Data Generation”. In: 2023, pp. 1618–1628.

[162] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. “Implicit object mapping with noisy data”. In: *arXiv preprint arXiv:2204.10516* (2022).

[163] Thomas Müller et al. “Instant neural graphics primitives with a multiresolution hash encoding”. In: *ACM Transactions on Graphics (ToG)* 41.4 (2022), pp. 1–15.

[164] Bangbang Yang et al. “Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering”. In: *International Conference on Computer Vision (ICCV)*. Oct. 2021.

[165] Francesco Locatello et al. “Object-Centric Learning with Slot Attention”. In: *ArXiv* abs/2006.15055 (2020).

[166] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. “GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement”. In: *Neural Information Processing Systems*. 2021.

[167] Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. “Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation”. In: *ArXiv* abs/2104.01148 (2021).

[168] Yizhe Wu, Oiwi Parker Jones, and Ingmar Posner. “ObPose: Leveraging Canonical Pose for Object-Centric Scene Inference in 3D”. In: *ArXiv* abs/2206.03591 (2022).

[169] Allan Zhou et al. “NeRF in the Palm of Your Hand: Corrective Augmentation for Robotics via Novel-View Synthesis”. In: *arXiv preprint arXiv:2301.08556* (2023).

[170] Edgar Sucar et al. “iMAP: Implicit mapping and positioning in real-time”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6229–6238.

[171] Jeffrey Ichnowski et al. “Dex-NeRF: Using a neural radiance field to grasp transparent objects”. In: *arXiv preprint arXiv:2110.14217* (2021).

[172] Qiyu Dai et al. “GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF”. In: *arXiv preprint arXiv:2210.06575* (2022).

[173] Valts Blukis et al. “Neural Fields for Robotic Object Manipulation from a Single Image”. In: *arXiv preprint arXiv:2210.12126* (2022).

[174] Christopher P. Burgess et al. “MONet: Unsupervised Scene Decomposition and Representation”. In: *ArXiv* abs/1901.11390 (2019).

[175] Martin Engelcke et al. “GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations”. In: *ArXiv* abs/1907.13052 (2019).

[176] Jun Yamada, Jack Collins, and Ingmar Posner. “Efficient Skill Acquisition for Complex Manipulation Tasks in Obstructed Environments”. In: *arXiv preprint arXiv:2303.03365* (2023).

[177] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[178] Eric R Chan et al. “Efficient geometry-aware 3D generative adversarial networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16123–16133.

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

[180] Alex Yu et al. “Plenoctrees for real-time rendering of neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5752–5761.

[181] Matthew Tancik et al. “Nerfstudio: A Modular Framework for Neural Radiance Field Development”. In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH ’23. 2023.

[182] Walter Goodwin et al. “Semantically Grounded Object Matching for Robust Robotic Scene Rearrangement”. In: *2022 International Conference on Robotics and Automation (ICRA)* (2021), pp. 11138–11144.

[183] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *International Conference on Machine Learning*. 2021.

[184] Yu Xiang et al. “Learning rgb-d feature embeddings for unseen object instance segmentation”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 461–470.

[185] Douglas Morrison, Peter Corke, and J. Leitner. “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach”. In: *ArXiv* abs/1804.05172 (2018).

[186] Fabio Tosi et al. “Distilled semantics for comprehensive scene understanding from videos”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4654–4665.

[187] Yang You et al. “Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13647–13656.

[188] Maxim Tatarchenko et al. “What do single-view 3d reconstruction networks learn?” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3405–3414.

[189] Ben Poole et al. “Dreamfusion: Text-to-3d using 2d diffusion”. In: *arXiv preprint arXiv:2209.14988* (2022).

[190] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[191] Alexander Kirillov et al. “Segment anything”. In: *arXiv preprint arXiv:2304.02643* (2023).

[192] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.