Affordance-grounded Robot Perception and Manipulation in Adversarial, Translucent, and Cluttered Environments

by

Xiaotong Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2023

Xiaotong Chen

cxt@umich.edu

ORCID iD: 0000-0003-1988-059X

Acknowledgments

I am sincerely grateful to my advisor, Professor Odest Chadwicke Jenkins, the leader of Laboratory4Progress, for his unwavering support, kindness, and encouragement throughout my Ph.D. journey. His mentorship has been instrumental in shaping my research and personal growth. From providing project ideas and refining presentation skills to instilling confidence in my work, Chad has been an exceptional guide. My Ph.D. journey is not a complete straight way, and I have felt disappointed and perplexed during the time. Fortunately, Chad knows how to deal with that and he is there to help.

I extend my heartfelt thanks to Professor Iris Bahar. I was collaborating with her lab at Brown University in my first project. Her boundless support and insightful feedback ensured a smooth progress of my Ph.D. research.

I would sincerely express my appreciation to my senior Ph.D. colleagues when I joined the Robotics program and Laboratory4Progress: Zhiqiang Sui, Zheming Zhou, Zhen Zeng, Karthik Desingh, and Kevin French. Thank you all for your dedication to making such an open, collaborative environments for our lab, and invaluable demonstration and feedback on how to be a good researcher. I was so fortunate to have a smooth start with your supervision. Besides, it is also my pleasure to work together with other collaborators and members in our lab: Jasmine Berry, Jana Pavlasek, Anthony Opipari, Stanley Lewis, Alphosus Adu Bredu, Emily Sheetz, Cameron Kisailus, Liz Olson, Jorge Vilchis, Rui Chen, Shiyang Lu, Kaizhi Zheng, Huijie Zhang, Zeren Yu, Thomas Cohn, Haonan Chang, Jinfan Zhou, Yanqi Liu, and Zhefan Ye.

My deepest appreciation goes to my thesis committee members, professors Sonia Chernova, Jason Corso, Nima Fazeli, Jessy Grizzle, and my advisor Chad Jenkins. Their constructive feedback and encouragement were vital in shaping this dissertation.

Lastly, I owe a profound debt of gratitude to my family. To my parents, who have endlessly supported me throughout this international study, I am forever grateful. I would thank from the bottom of my heart to my partner, Jialin Huang. Her understanding and love motivated me to be a better person in every aspect. And to my beloved cat, Coco, for being a delightful and comforting companion during this journey.

TABLE OF CONTENTS

Acknowledgments . . . . **ii**

List of Figures . . . . **viii**

List of Tables . . . . **xiii**

Abstract . . . . **xv**

Chapter

1 Introduction **1**

1.1 Motivation 1

1.2 Challenges 3

1.3 Dissertation Contribution 5

2 Related Works **7**

2.1 Robotic Perception for Manipulation 7

2.2 Object 6D Pose Estimation 9

2.3 Object Affordance Representation and Learning 10

2.4 Datasets and Annotation Pipelines 10

3 GRIP: Generative Robust Inference and Perception for Semantic Robot Manipulation in Adversarial Environments **12**

3.1 Motivation 12

3.2 Related Works 15

3.2.1 Perception for Manipulation 16

3.2.2 Object Detection and Pose Estimation 16

3.3 Problem Formulation 17

3.4 Approach 18

3.4.1 Object Detection 19

3.4.2 Pose Estimation 20

3.5 Experiments 23

3.5.1 Implementation 23

3.5.2 Dataset and Baselines 25

3.5.3 Evaluation 25

3.5.4 Robotic Manipulation 29

3.6 Summary 29

4 ACF: Manipulation-Oriented Object Perception in Clutter through Affordance Coordinate Frames 31

4.1 Motivation 31

4.2 Related Works 33

4.2.1 Affordances for Robotic Manipulation 34

4.2.2 Object Pose Estimation for Manipulation 35

4.2.3 Task-Oriented Grasping 35

4.3 Affordance Coordinate Frames 36

4.3.1 Representation 36

4.3.2 Estimation Pipeline 37

4.4 Experiments 39

4.4.1 Ablation Studies on Axis Estimation 41

4.4.2 Comparison with NOCS 42

4.4.3 Object-Level Detection Accuracy 45

4.4.4 Application to Robotic Manipulation Task 45

4.5 Summary 48

5 ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception 50

5.1 Motivation 50

5.2 Related Work 53

5.2.1 Direct & Human-in-the-loop labelling 53

5.2.2 End-to-End Labellers 54

5.3 ProgressLabeller 55

5.3.1 Camera pose estimation 55

5.3.2 Object alignment by multi-view silhouette matching 55

5.3.3 Semi-automatic labelling pipeline 57

5.3.4 Annotation accuracy estimation from simulation 58

5.4 Experiments 60

5.4.1 Evaluation on label generation 61

5.4.2 Multi-camera dataset creation 63

5.4.3 6D object pose estimation Fine-tuning 64

5.4.4 Other applications 68

5.5 Summary 70

6 ClearPose: Large-scale Transparent Object Dataset and Benchmark 71

6.1 Motivation 71

6.2 Related Work 74

6.2.1 Transparent Dataset and Annotation 74

6.2.2 Transparent Depth Completion and Object Pose Estimation 75

6.3 Dataset 76

6.3.1 Dataset Objects and Statistics 76

6.3.2 Pose Annotation 78

6.4 Benchmark Experiments 80
6.4.1 Depth Completion 80
6.4.2 Instance-Level Object Pose Estimation 81
6.5 Discussions 85
6.6 Summary 86
7 TransNet: Transparent Object Manipulation Through Category-Level Pose Estimation 87
7.1 Motivation 87
7.2 Related Work 88
7.2.1 Transparent Object Visual Perception for Manipulation 88
7.2.2 Opaque Object Category-level Pose Estimation 89
7.3 TransNet 90
7.3.1 Architecture Overview 90
7.3.2 Object Instance Detection and Segmentation 91
7.3.3 Cross-task Consistency for Depth-Normal Pairs 91
7.3.4 Generalized Point Cloud 92
7.3.5 Transformer Feature embedding 93
7.3.6 Pose and Scale Estimation 93
7.4 Experiments 95
7.4.1 Comparison with Baseline 97
7.4.2 Ablation Study 98
7.4.3 Robot Experiment 100
7.5 Summary 101
8 Conclusion and Discussion 104
8.1 Conclusion 104
8.2 Limitations 105

8.3 Future Directions 106
8.3.1 Novel Object Pose Estimation and Manipulation 106
8.3.2 Dense Reconstruction for Transparent Shapes 106
8.3.3 Affordance-grounded Robotic Task Execution 107
8.3.4 Human-factor Research in Robotic Data Annotation 107
8.4 Summary 107
Bibliography **109**

LIST OF FIGURES

FIGURE

1.1 Robots in various applications. (Top Left) The first industrial robot, Unimate [141], is pouring coffee for a woman. (Top Middle) Industrial robot arms are assmbling metal parts [4]. (Top Right) A PR-2 robot is cooking in household environment [5]. (Bottom Left) A Boston Dynamics Spot mobile robot [2] is collecting the helmet on the snowy ground. (Bottom Middle) A self-driving vehicle equipped with various sensors like LiDAR and camera [3]. (Bottom Right) An Agility Digit humanoid robot [1] is stepping upstairs while carrying a box. 2

1.2 Challenges faced in robot visual perception system. (a) Two rooms with normal and dark lighting conditions. (b) Two clusters of different liquid containers that all share ‘grasp’, ‘contain’, and ‘pour’ affordances. (c) A scene with common transparent furniture and another scene with transparency, reflective metal, and occlusions between objects. 3

1.3 Demonstrations of robots and tools working against various challenges. (Top Left) The Fetch robot picks up an object under the dark environment. (Bottom Left) The Fetch robot uses the novel bottle and spatula to execute pouring and stirring affordances. (Top Right) RGB, segmentation mask, surface normal, and ground truth depth annotations of the collected large-scale transparent dataset. (Bottom Right) The Fetch robot uses novel transparent glasses to execute pick-place and pouring affordances. 5

2.1 Current state-of-the-art robotic visual perception systems on object state estimation. . 8

3.1 Our *GRIP* system perceiving and grasping an object in adversarially darkened lighting. *GRIP* uses two stages of (a) PyramidCNN object detection bounding boxes with confidence score greater than 0.1 (green boxes), shown along with the ground truth (red box), and (b) sample-based generative inference. The (c) resulting estimate and (d) its localized pose (highlighted in cyan) enables (e) the Michigan Progress Fetch robot to accurately grasp the potted meat can object. 13

3.2 Overview of *GRIP*. The robot operating in a dark and cluttered environment is to grasp the meat can from its RGBD observation. Stage 1 takes the RGB image and generates object bounding boxes with confidence scores. Stage 2 takes the depth image and performs sample-based generative inference to estimate the pose for each object in the scene. The samples in Stage 2 are initialized according to bounding boxes from Stage 1. From this estimate, the robot performs manipulation on the meat can object. 15

3.3 Plot of pose accuracy vs. iteration numbers of all converged trials. The point-wise distance is calculated using ADD and ADD-S metrics [192] and not used in pose estimation. After about 150 iterations, the point-wise distance reaches below 0.005m. . 24
3.4 Testing dataset with YCB objects under adversarial settings. The base-setting data is collected with regular lighting without occlusions. The dark-setting data is collected with lights off in the room. The single lighting data is collected with a flash light. Object poses are the same in previous three settings. Data in three occlusion scenes is collected with the same objects randomly stacked. 26
3.5 The comparison between DOPE, PoseCNN+ICP and our *GRIP* two-stage method on pose estimation accuracy of 4 objects mentioned in Sec. 3.5.3. 27
3.6 Comparison of *GRIP*, DOPE and PoseCNN under adversarial scenarios. In varied lighting condition, DOPE only detects 006_mustard_bottle correctly while PoseCNN makes inaccurate depth estimation (marked yellow). In occlusion condition, DOPE misses half of the objects while PoseCNN fails to detect object poses in clutter. *GRIP* correctly detects all objects under both scenes except 051_large_clamp under occlusion setting (cyan arrow) where the sampling converges to object 052_extra_large_clamp because of their geometric similarity. 28
3.7 Overall pose estimation accuracy of 15 YCB objects using PoseCNN and our *GRIP* method. 28
3.8 A Fetch robot is packing grocery based on pose estimates from the *GRIP* pipeline. . . 30

4.1 Affordance Coordinate Frames (ACF) provide keypoint and axis constraints for each manipulation action afforded by an object. In the above example, we estimate a "grasp" ACF on the handle of a mug, as well as a "pour" ACF on the body of a mug (bottom left). The robot uses the estimated ACFs to execute a task involving pouring liquid from the mug (right). The ACF is generalizable across object instances. 32
4.2 Objects can be divided into functional parts which represent their affordances. The Affordance Coordinate Frame (ACF) is defined as a 3D keypoint and a directed axis, which are consistent across affordance classes. The ACF can be used by a robot to plan manipulation actions. The object parts can be re-assembled to generate new instances with novel shapes. 33
4.3 The perception pipeline for estimating the affordance coordinate frames of a mug with two affordance parts: a container part and a handle part. (a) The Faster R-CNN module detects bounding boxes from features extracted through feature pyramid network out of a 4-channel fused RGB-depth image. (b) The corresponding region of interest (ROI) features are further fused with global features using self-attention module in [17]. 3D keypoint and directed axis endpoint offsets are learned in two parallel network heads (C is the number of part classes). The voters are formed by adding the learned offsets to uniformly sampled seeds within a bounding box and estimated keypoints and axes are computed by clustering the voter points. (c) Another network head learns 2D part affinity fields, which connect parts of an object. 36
4.4 The models in the train set and the validation / test set. For the synthetic dataset and robot simulation experiments, we add random texture to each object, as shown in Figure 4.8 and Figure 4.9. 40

4.5 Comparison between each method for different part estimation with respect to mAP curves. The top row shows rotation accuracy and the bottom row shows translation accuracy. 43
4.6 Qualitative results on our dataset with the ACF-Endpoints method. The red points show the keypoint of each part, orange arrows show the estimated axes and green arrows show the ground truth axes. Then, the white line shows the connection of two parts. . . 43
4.7 Grasp poses and pour, stir manipulation waypoints composed from detected ACFs. . . 45
4.8 An example of ACF estimation based grasping pipeline. From left to right, 3 pairs of estimation result images (detected body, handle, stir, head are with blue, green, red and purple masks) and grasping snapshots are shown for 3 objects (highlighted in red bounding boxes) in object clutter. The grasp poses are decided as shown in Figure 4.7. 46
4.9 An example of ACF estimation based drink serving pipeline. From left to right, we show the ACF estimation result (left most) and snapshots of grasping two containers and pouring liquid (in cyan and yellow), and grasping a spoon to stir the liquid inside the container. 46
4.10 The mask, ACF keypoint and axis estimations are overlaid on the left images. On top-right, Fetch robot is grasping cups, pouring to another container based on ACF estimation in top-left image. On bottom-right, Fetch robot is picking up a spoon to stir inside a bowl based on bottom-left image. 48

5.1 The ProgressLabeller offers an interactive GUI for aligning all kinds of objects in the 3D scene to generate large-scale datasets with ground truth pose labels. The left image shows the rough 6D pose estimates from one state-of-the-art RGB-D deep models trained on public YCB dataset, and the right images shows fine-tuned pose estimates from the same model after retraining using generated data from ProgressLabeller. The pose estimates are then used for robotic grasping experiments. 51
5.2 Capture of ProgressLabeller user interface for multi-view re-projection checking. The top-left view shows the aligned object models with rendered texture at labelled poses. The bottom-left view and top-right view show the silhouettes and boundaries respectively at the same camera view, and the bottom-right view shows the boundaries from another view for validation. 56
5.3 Diagram for an object shown under a camera. c denote the location of camera and c_x, c_y, c_z are its x, y, z axis. p is a plane parallel to camera plane and passing through object's center v_o. ω the rotation direction parallel to c_x and passing through v_o. θ_1 is the magnitude of rotation radius. v is the translation direction within the plane p and θ_2 is the translation magnitude. On the right hand side is the projection image, the object in the transparent color is the object with ground truth pose. 59
5.4 From left to right, we show the 3D labeller view of LabelFusion, ProgressLabeller and cropped RGB image with masks from LabelFusion, ground truth and ProgressLabeller. The top and bottom row show a sample from HOPE and YCB dataset respectively. We find even ground truth pose labels have easily observable errors, shown as the spacing between black masks and object boundaries. 60
5.5 The top and bottom row shows RGB images in training and testing set respectively. From left to right, the images are taken from Primesense, RealSense D435i and L515. 63

5.6 A synthesized RGB image using Blenderproc is shown on the left. The original image is shown on the right. The YCB objects are rendered and overlaid to original image according to labelled poses. Depth synthesized images are generated in the same way. 65
5.7 The Fetch robot is grasping objects on tabletop based on estimated poses. The pose estimates are shown as projected point clouds and coordinate frames in top-right smaller images. The higher and lower frames correspond to pre-grasp and grasp poses. 67
5.8 Comparison of best (left) and worst (right) validation pose renderings from the NSVF model. The smaller images with black background on top corners are the ground truth masks, and those on bottom corners are rendered output from NSVF, and the middle larger images are the renderings imposed onto the original captured image in greyscale. The best image (left) had an average pixelwise L2 error of 0.0067, and the worst (right) had an average pixelwise L2 error of 0.1867. 68
5.9 The labelling on transparent champagne cups. When there is no reliable depth (top-left), ProgressLabeller enables checking the pose validity by re-projection object models (bottom-left) to RGB images from different views (top-right and bottom-right). 69

6.1 Sample images from existing transparent datasets. TOD and StereOBJ-1M are collected using stereo RGB cameras, and other datasets used RGB-D cameras (except for Omniverse which is completely synthetic). 72
6.2 Sample images from ClearPose dataset. On the left, we show RGB images taken for different object subsets under various lighting conditions and backgrounds. On the right, we show examples of different types of testing scenes, such as covered by a translucent box (top-left), novel background with opaque distractor objects (top-right), non-planar cases (middle-left), filled with liquid (middle-right), and heavy clutters of transparent objects (bottom). 73
6.3 Objects included in the ClearPose dataset. On the left, we show the rendered CAD models for each object. From top to bottom there are bowls and plates (1st row), containers and bottles (2nd row), wine cups, two mugs and a pitcher (3rd row), water cups (4th row), and spoons, a fork, knives as well as chemical supplies (bottom row). On the right, we show real images of household objects on the top right, and chemical supplies on the bottom right. 77
6.4 Distribution of instance annotations and viewpoint orientation coverage statistics for every object in the ClearPose dataset. The 'kitchen' category includes fork, spoon, knife, mug, plate and pitcher objects. 78
6.5 Transparent object pose annotation pipeline using ProgressLabeller. On top-left we show one sample of aligned RGB-D images from continuous streams captured by the camera. The frames are fed into visual SLAM to estimate camera trajectory, then the objects' poses are annotated in a multi-view silhouette alignment interface shown at the bottom-right. Finally, the object poses, surface normals, and fixed depths are calculated and rendered as output. 79
6.6 Qualitative depth completion results. From left to right, there are RGB image, raw depth, ground truth depth rendered using object CAD models, completed depth by TransCG and ImplicitDepth. 82

6.7 Visualization of pose estimation in ClearPose dataset. From left to right, they are raw image, ground truth object poses, pose estimation results of method from Xu et al., FFB6D$_{g/g}$, FFB6D$_{g/c}$, FFB6D$_{r/r}$. From top to bottom, results are shown in different test scenes in Table 6.3. Objects are projected to color masks based on their pose estimates, with their 6DoF poses marked as the red-green-blue coordinate frame. . . . 84
6.8 Examples of multi-layer transparent object appearance. In the left image, the annotated bounding boxes show large overlap between object pairs, where the objects behind are still perceivable in some cases with less light distortion. In the right image, objects behind the translucent surface are still detectable as well. 86

7.1 The two-stage architecture of TransNet. Image patches of RGB, ray direction, surface normal, and completed depth are concatenated and embedded to learn the translation, x-axis, z-axis, and scale separately of the estimated category-level pose. 90
7.2 Transparent objects used in experiments. The image on the left show the 25 objects in the training set. The image on the right shows 8 novel objects in the test set for vision evaluation and robotic experiments. 96
7.3 Qualitative results of category-level pose estimates from TransNet. White bounding boxes depict ground truth pose annotations, while colored boxes depict TransNet's estimates. Different colors represent different categories. The top two rows show accurate estimates, while the bottom row highlights inaccurate estimates when TransNet is confronted with heavyily cluttered scenes. Objects with no bounding box correspond to categories not used during training and evaluation. For axial symmetric objects, we use the ground truth x-axis to produce a 3D bounding box. 102
7.4 Robotic experiments. The top row is the demo of pouring using a water cup and a bowl. The bottom row is the demo of pick-and-place using two water cups. The left column is the initial state, the middle column is picking, and the right column is pouring or placing. 103

LIST OF TABLES

TABLE

3.1 Overall Performance (Area Under accuracy-threshold Curve) of 15 YCB Objects on DOPE, PoseCNN with ICP and our *GRIP* method. Symmetric objects are marked with stars and evaluated using ADD-S; asymmetric objects are evaluated using ADD. . . . 29

4.1 Object-Part-Action Relations. The top table indicates the parts contained in an object with an ‘x’. The bottom table indicates the actions which correspond to a part. The ‘container’ and ‘handle’ parts are separated in the way shown in Figure 4.2. ‘stir’ refers to the straight handle part of spoons, spatulas and hammers, and ‘scoop’ refers to the head part that supports scoop of spoons and spatulas. 41

4.2 Comparison of proposed ACF method with different axis representation and NOCS on object part keypoint position and axis estimation. The numbers inside the table are mean Average Precision (mAP) within different error tolerances. 44

4.3 Task evaluation on grasp, pour, stir and drink serving which includes two pour and a stir action in sequence. 47

5.1 Results on feature matching distance between rendered object RGB image and original image. ’Pos (cm)’, ’Rot (quat)’ refers to positional error in centimeters and rotational error calculated as norm of difference in quaternions as in [80]. LabelFusion cannot work on T-LESS data streams at all, and it cannot reconstruct one of HOPE samples scene entirely, so we trim the sequence to 1/4 length. Object models in T-LESS and NOCS datasets do not have enough features to measure the feasure-based pixel distances. 61

5.2 The pose estimation accuracy of FFB6D cross-validated on datasets collected using three cameras. 64

5.3 Grasp record comparison on part of YCB objects using grasp poses generated based on pose estimates from pretrained and fine-tuned FFB6D on the collected dataset, with Blenderproc data synthesis, LabelFusion annotations and ProgressLabeller annotations. 66

5.4 Average pixelwise errors for the train, validation, and test sets from the NSVF model trained on ProgressLabeller outputs. 70

6.1 Comparison between existing transparent object datasets and ClearPose. *Trans10K is a transparent segmentation dataset and has no object-centric pose labels. *StereObj1M is not publicly available at the time of submission so the #frame and #pose annotation are estimated based on the published ratio of transparent objects in the entire object set [111] 72

6.2 Depth completion benchmark results of ImplicitDepth and TransCG on 6 different test scenarios of the ClearPose dataset. 81
6.3 Pose estimation accuracy comparison on different test sets of the ClearPose dataset. FFB6D$_{r/r}$ refers to train and test FFB6D model both on raw depth. Similarly, FFB6D$_{g/c}$, FFB6D$_{g/g}$ refer to train on ground truth, test on completed depth from TransCG, and train and test both on ground truth, respectively. The values are averaged across all objects in the dataset. 83

7.1 Comparison with the baseline on the Clearpose Dataset. 97
7.2 Ablation study for TransNet 98
7.3 Comparison between different embedding methods 99
7.4 Accuracy for depth-normal pair estimation 99
7.5 Robot experiment success rate 101

ABSTRACT

Robots in the future need to work in natural scenarios and finish a variety of tasks without much supervision from humans. To achieve this goal, we want the robots to perform perception and action robustly and adaptively in unstructured environments. For example, robots are expected to correctly perceive objects in unseen cases, such as dark environments, heavy clutter or transparent materials. Besides, they should learn skills that are transferable across novel objects in categories rather than fixed on known instances. In this dissertation, we focus on the problem of perceiving and manipulating various objects in complex adversarial environments. Specifically, we explore on three aspects including *robustness* to adversarial environments, *synergistic* perception and action, and *adaptable* data-driven perception pipelines for customized settings.

First, we explore the possibility to achieve robustness for object pose estimation algorithms against environmental changes, like object occlusions and lighting changes. We contribute a two-stage approach *GRIP* that combines both the discriminative power of deep convolutional neural networks (CNNs) and robustness in probabilistic generative inference. Our results indicate that *GRIP* has better accuracy through comparison with end-to-end pose estimation baselines, and efficacy in a grocery packing task in the dark scene.

Second, we focus on how to generalize object representation to category-level with grounded affordance for task execution. We propose the Affordance Coordinate Frame (ACF) representation that enables direct connection between perception and executable action. Along with that, an object part category-level scene perception pipeline is contributed to estimate *ACFs* in cluttered environments on novel objects. Our pipeline outperforms state-of-the-art methods for object detection, as well as category-level pose estimation for object parts. We further demonstrate the applicability of ACF to robot manipulation tasks like grasping, pouring and stirring.

Third, we contribute an annotation pipeline that enables large-scale dataset creation and benchmarking on transparent objects. The proposed *ProgressLabeller* pipeline has a multi-view annotation interface that allows fast and accurate pose annotation on RGB-D video streams. *ProgressLabeller* is proved to generate more accurate annotations in object pose estimation and grasping experiments. Using *ProgressLabeller*, we contributed *ClearPose* as the first large-scale RGB-D transparent object dataset with various adversarial conditions such as lighting changes, object clutters. *ClearPose* is made to support benchmarking of data-driven approaches on depth completion, object pose estimation and robotic manipulation. Then, we built an object pose estimation based manipulation framework, *TransNet*, for daily transparent objects. The system aims to generalize the pose estimation to unseen novel objects defined in several categories like wine cups and bottles. We finally demonstrate the efficacy of the system with robotic pick-and-place and pouring tasks, paving the way for more complex manipulations such as table-setting and drink serving.

CHAPTER 1

Introduction

1.1 Motivation

The era of robotics and automation began with the invention of the Unimate, the first industrial robot by George Devol in 1954 [46]. Since then, robots have taken on various forms, including but not limited to industrial robot arms, domestic service mobile manipulators, legged mobile bots, and humanoids, to cater to diverse application scenarios (Figure 1.1). Advancements in robotic hardware systems have driven the aspiration for robots to become intelligent, which involves having a general-purpose understanding of the physical world, learning from humans to acquire specific skills, and being aware of potential failures or safety issues. However, most robots in the past few decades have been designed to perform repetitive tasks in highly structured environments, such as welding and screw tightening in factory production lines [120]. These robots are efficient for specific tasks but lack flexibility to handle the diversity and uncertainty present in open, unstructured environments like homes. For example, objects will have various appearances under different lighting conditions and occlusions, and robots could fail to sense accurate depths on black surfaces or transparent material. Achieving a general-purpose intelligent robot poses several challenges in perception, planning, and control, as well as integrating individual systems into the real world.

In this dissertation, we focus mainly on the perception aspect of robotics. The rise of deep learning suggests a new way for robotic visual perception. Since AlexNet [97] demonstrated that deep learning could achieve better accuracy than traditional machine learning on image classification, vision tasks have been approached in a new way to achieve better performance, like

Figure 1.1: Robots in various applications. (Top Left) The first industrial robot, Unimate [141], is pouring coffee for a woman. (Top Middle) Industrial robot arms are assmbling metal parts [4]. (Top Right) A PR-2 robot is cooking in household environment [5]. (Bottom Left) A Boston Dynamics Spot mobile robot [2] is collecting the helmet on the snowy ground. (Bottom Middle) A self-driving vehicle equipped with various sensors like LiDAR and camera [3]. (Bottom Right) An Agility Digit humanoid robot [1] is stepping upstairs while carrying a box.

human [63] and object [75] pose estimation, 3D scene understanding [168], visual navigation [9], neural rendering and reconstruction [131], etc. These methods could provide essential 3D information of the environment around the robot, leading to vision-based systems for navigation [9], manipulation [213], and robot-human interaction [11], and learning from demonstration [207]. However, the deployment of vision-based robot manipulation systems will raise new problems and challenges, such as the imperfect sensor readings, and the learning gap between simulation and the real world environment. We expect generalized object representation that supports robot manipulation, and current 2D detection/segmentation works are still not enough as they cannot directly provide actionable outputs such as grasp poses or motion trajectories. In this work, our aim is to provide actionable representations for robot manipulation, while having the visual perception system robust to uncertainty in unstructured environments.

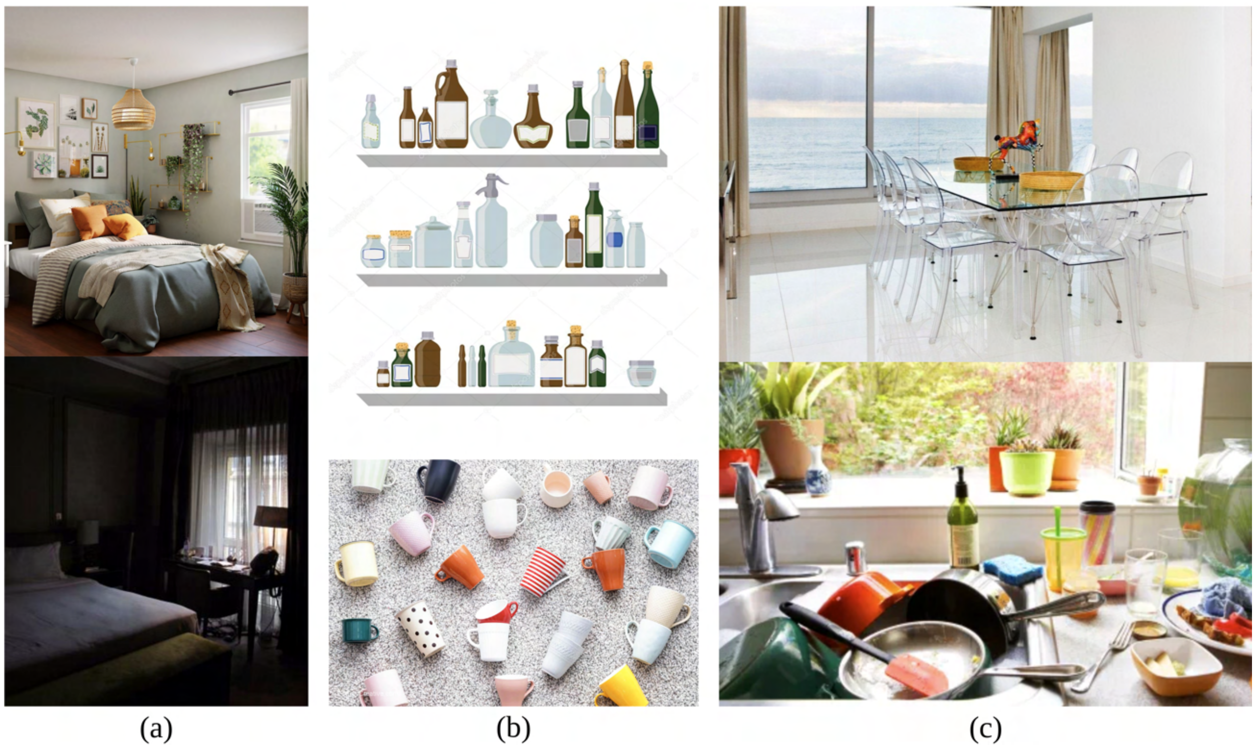


Figure 1.2: Challenges faced in robot visual perception system. (a) Two rooms with normal and dark lighting conditions. (b) Two clusters of different liquid containers that all share ‘grasp’, ‘contain’, and ‘pour’ affordances. (c) A scene with common transparent furniture and another scene with transparency, reflective metal, and occlusions between objects.

1.2 Challenges

In this dissertation, we address problems in robotic perception that will enable deployment of real robots to accomplish manipulation tasks in common human environments. Specifically, the difficulty of learning generalized knowledge of objects, and the surrounding environments comes from at least three basic aspects.

The first is the partial observability and openness to adversarial conditions that lead to inefficient data. Range sensors like RGB, RGB-D, stereo cameras or LiDAR have limited working ranges, resolutions, and measurement noises. The sensor readings can be affected by environments or the structure of the individual objects themselves. For example, the RGB pixel values of objects will vary under different lighting conditions, as shown in Figure 1.2 (a). Human environments also have objects with black surfaces, and metal or transparent materials that could cause invalid measurement on depth sensors. Since the training data is mostly limited and hard to cover all these variations, the testing environment is considered ‘adversarial’ and could cause failure in perception.

The second is the diversity of object forms and functionalities, as shown in Figure 1.2 (b). There are massive object categories, various articulated objects like pliers and drawers, non-rigid deformable objects like ropes, cloth, plastic bags, etc. Objects with the same ‘affordance’, which can be defined as the ability to support certain types of actions, could have different appearances. However, for instance-level 6 degree-of-freedom (DOF) pose estimation systems [182, 187], they assume known object geometry models, which constrains the methods to a finite set of rigid objects.

The third is the complexity of scene context, as shown in Figure 1.2 (c). The object appearances could change under different lighting conditions, or occluded by the other objects around. The spatial and affordance relations between objects are also important for completing a task. For example, the robot needs to consider the supporting relations when it works with object clutter, open a drawer before retrieving the target object inside, and be careful when it is picking up a full cup of water.

Our aim in this dissertation is to provide robustness, generalized representation, and scalability for robot perception systems to tackle the aforementioned three challenges. First, for the *robustness* to the imperfect sensor readings and limited data, we want to marry the robustness from probabilistic inference with the speed and accuracy advantage of deep learning algorithms to reach a balance on precision and recall, especially we expect a combined system to work robustly in adversarial environments. An example of object manipulation in the dark environment is shown in the top left image of Figure 1.3. Second, for the *generalizable representation* for various objects that share the same affordances, we want to generalize the object-specific learned representation, e.g. 6D pose, to category-level, so that the system trained on a small set of object instances could generalize to other unseen instances and relax the dependency on known object 3D geometry models. A robot demonstration of using novel object to execute pouring and stirring affordance is shown in the bottom left image of Figure 1.3. Third, for the *adaptability* for the complicated combination of different objects and their contexts, we want to have the ability to create enough data for deep learning based system to adapt quickly to any customized environment. Given the

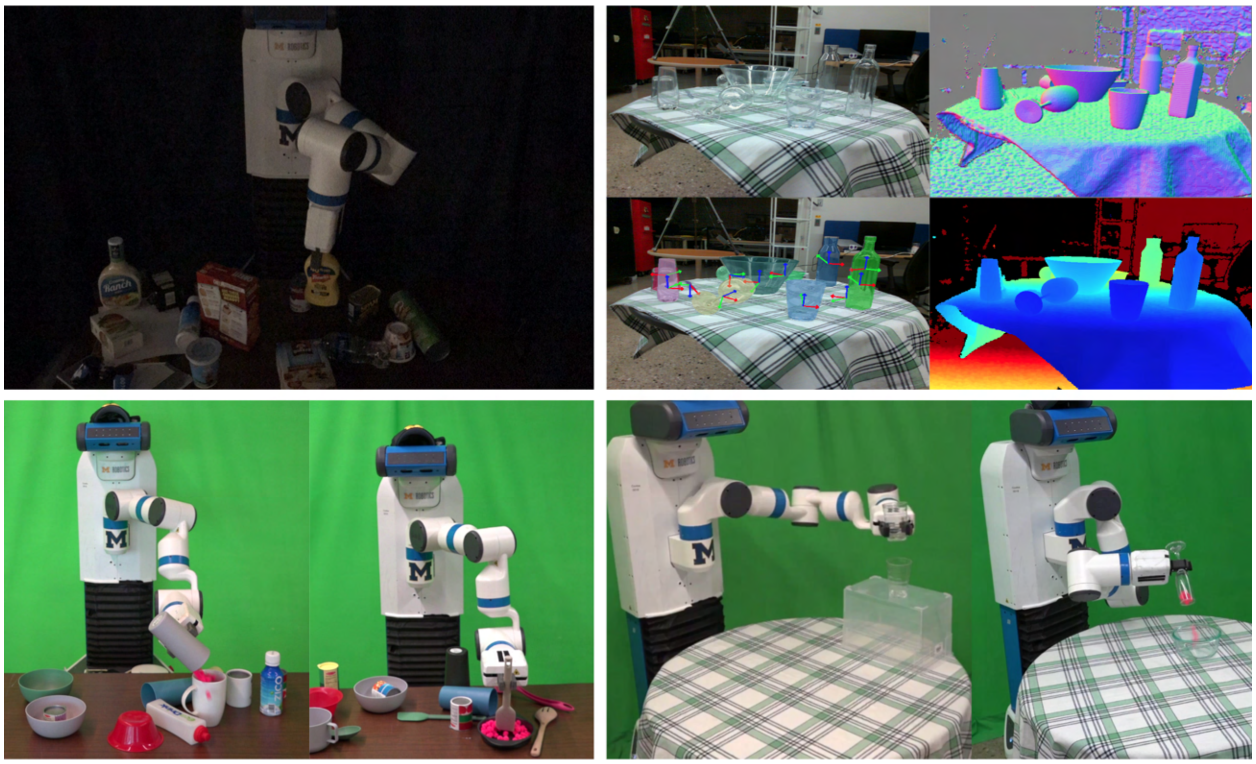


Figure 1.3: Demonstrations of robots and tools working against various challenges. (Top Left) The Fetch robot picks up an object under the dark environment. (Bottom Left) The Fetch robot uses the novel bottle and spatula to execute pouring and stirring affordances. (Top Right) RGB, segmentation mask, surface normal, and ground truth depth annotations of the collected large-scale transparent dataset. (Bottom Right) The Fetch robot uses novel transparent glasses to execute pick-place and pouring affordances.

current computation power, we could build pipelines that create data for pretrained models to quickly adapt to specific robotic deployment scenarios. Besides, such a data generation system paves the way for research on challenging objects or environments, such as transparent objects that lack distinguishable RGB features and cannot be detected by most depth sensors due to lack of light reflection. The dataset sample and robot manipulation demonstration using the trained perception system are shown in the right two images of Figure 1.3.

1.3 Dissertation Contribution

In this dissertation, our objective is to develop data-driven robotic perception systems on object state estimation for manipulation that are robust, generalizable, and scalable to novel environments or objects. In particular, we make the following contributions:

- In GRIP (Chapter 3), we improve the *robustness* of perception systems by proposing a combined discrimitive-generative method for robust pose estimation and robot manipulation in adversarial environments. Through comparison with end-to-end pipelines, we demonstrate the probabilistic inference could enhance robustness when CNNs fail to localize objects correctly in challenging unseen environments in the training set.

- In ACF (Chapter 4), we generalize the system to *category-level* by exploring synergistic visual perception and robot action through the concept of affordance with a deep network that estimates object part-level executable representation for robot manipulation. The proposed system frees the assumption of known object 3D model and demonstrates generalized manipulation across object instances within each category.

- In ProgressLabeller (Chapter 5), we contribute a rapid data generation pipeline to improve the *adaptability and scale* of data-driven perception systems. Our pipeline could create enough data to feed a deep network towards its best performance with little effort in data annotation, and also open possibility of creating dataset for transparent objects.

- In ClearPose (Chapter 6), we contribute a large-scale RGB-D transparent object dataset using ProgressLabeller. The dataset could open the door for data-driven perception research on transparent objects, including tasks like detection, depth completion, pose estimation, and robot manipulation etc.

- In TransNet (Chapter 7), we build a category-level transparent object pose estimator for robot manipulation. TransNet could generalize the estimated pose to novel object instances and perform pick-and-place as well as pouring tasks in the new environment settings.

CHAPTER 2

Related Works

This dissertation is aimed to explore robust, adaptive, and generalizable robotic object perception in complex, adversarial environments. In relation to this goal, we provide an overview of several sub-areas in the following sections in this paper: object pose estimation, affordance representation and learning, and dataset and annotation pipelines. We list current state-of-the-art robotic visual perception systems, focusing on their inherent affordance and functionality, as well as the degree of uncertainty they encounter within the surrounding environment. The unique contribution of this work is to integrating the level of function generalization concerning object affordance and sequential task, while adapting to adversarial environments including different object materials, lighting conditions, and occlusion between objects. Through this comprehensive approach, this paper seeks to fill the gap of existing works and makes a step forward towards general object perception and task execution. The overall contribution of this dissertation relative to concurrent research works is illustrated in Figure 2.1 below.

2.1 Robotic Perception for Manipulation

Robots need sensors to receive feedback from their surroundings, and vision sensors, like RGB and depth cameras, are a common choice. Robotic perception and manipulation systems are often constructed as a sequential two-stage pipeline. First, the states of objects of interest are estimated using computer vision approaches from the sensor images or 3D point clouds. Then, the grasping pose or manipulation trajectory can be calculated or estimated based on the object state

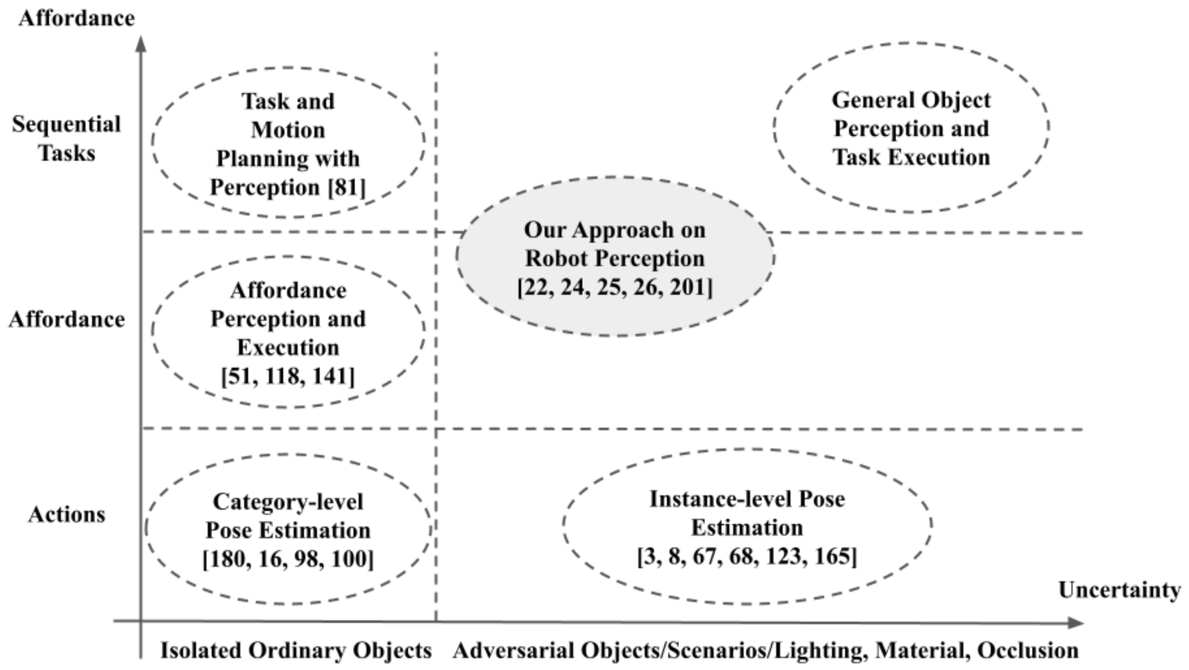

Affordance
Sequential Tasks
Affordance
Actions
Task and Motion Planning with Perception [81]
General Object Perception and Task Execution
Our Approach on Robot Perception [22, 24, 25, 26, 201]
Affordance Perception and Execution [51, 118, 141]
Category-level Pose Estimation [180, 16, 98, 100]
Instance-level Pose Estimation [3, 8, 67, 68, 123, 165]
Uncertainty
Isolated Ordinary Objects
Adversarial Objects/Scenarios/Lighting, Material, Occlusion

Figure 2.1: Current state-of-the-art robotic visual perception systems on object state estimation.

estimates and carried out by the robot to accomplish the tasks. Methods like model registration using local [84, 158] or global descriptors [159, 8, 191], or analysis-by-synthesis [169, 99], were proposed to estimate object 6D poses from 2D image features or object renderings, often with assumption of known object 3D models. Local refinement methods like Iterative Closest Point (ICP) [156], RANSAC [56], importance sampling [173], etc., are used to further correct the estimations. The pioneer works [34, 35, 137, 138] in this area integrated these methods and proposed pose estimation pipelines and did simple object grasping and pick-place manipulation. More recently, robotic grasping detection and object-level 6D pose estimation emerged with the application of deep learning. Grasp Pose Detection (GPD) [177], DexNet [122] introduced end-to-end grasp pose estimation from RGB(D) images directly that are object-model agnostic. On the other hand, PoseCNN [193] and DenseFusion [187] proposed end-to-end deep pose estimation based on RGB and RGB-D image input, respectively. The estimated object poses can be directly applied for grasping and manipulation.

2.2 Object 6D Pose Estimation

Object 6D pose is a great representation that can be directly used in generating robotic grasping and manipulation trajectories. Prior works take the assumption of known object 3D models in a database, and the problem is converted to a classification plus 3D rigid shape registration. Before the prevalence of deep learning, traditional methods using Random Forest [13, 130], template matching [75], descriptor matching [84, 159], point-pair features [76], analysis-by-synthesis [169, 99] have been proposed to detect objects and estimate their poses. Then there comes pure end-to-end deep-learning pipelines from PoseCNN [192], BB8 [151], and SSD-6D [92] started the trend, followed by a large set of methods using RGB images [182, 14, 105, 147, 100, 70], or RGB-depth images [187, 73, 74] as input. Besides, there are two-stage methods that take the neural network output as a prior, and search the object pose through probabilistic inference [173], point-set matching [132] or Monte-Carlo Tree Search [133].

In order to generalize pose estimation approach to category-level and free of object 3D models, Wang et al. [188] proposed Normalized Coordinate Space (NOCS) that learns a canonical 3D shape for all instances within a category and generalizes to novel unseen instances. Later works [179, 7, 102, 28, 21, 106, 82] explored ideas including geometry shape priors, canonical shape encoding, convolutional layers that fuse RGB and depth input, etc. and improve the pose estimate accuracy with respect to 3D Intersection-over-Union (IoU), and percentage within certain distance/angle threshold, e.g. 50% within 5° and 5cm. Recent works have achieved much higher accuracy. 6D-ViT [218] directly proposed a transformer-based end-to-end pipeline. FS-Net [26] that learns explicitly two orthogonal axes that represent the 3D rotation. GPV-Pose [47] reduce errors further by cross-checking different types of losses for reconstruction, bounding box estimation, and axis estimation. Very recently, there are works exploring completely 3D model-free pose estimation without even a category prior. Gen6D [116] implements a image patch matching pipeline to retrieve an unseen object pose relative to a few images with provided poses.

2.3 Object Affordance Representation and Learning

The concept of ‘Affordance’ was introduced by J.J. Gibson back in 1977 [58] in Psychology to describe *specific combination of the properties of its substance and its surfaces taken with reference to an animal* This definition basically tells what the environment could offer the animal for good or ill. The application of this term in Computer Vision and Robotics is specified to the functionality of everyday objects to certain goal-directed tasks, for example, we say a cup has *contain* and *pour* affordance because it could contain or pour water. How to build a knowledge-base, reason, sense, and act over object affordances become a popular research direction [68, 205, 160]. AFNet [185, 186] proposed a knowledge network and its ontology for robotics, which classifies common affordances into 3 categories: structural, material, and semantic affordances. Structure affordance corresponds to the 2D/3D shape of objects that mainly affect the visual recognition and afforded motions, while material and semantic affordances define fine-grained classifications and the usages. Recently, more works have been proposed in visual recognition like object detection and segmentation [48, 198, 33], motion generation [67, 66, 128], model-based synergistic perception and action [125, 57, 31, 95], and robot task learning [65, 150, 174, 213].

2.4 Datasets and Annotation Pipelines

As deep-learning based approaches outperformed traditional vision-based robotic systems in perception accuracy, the acquirement on large-scale, high-quality datasets become a new bottleneck. Different from pure RGB images for vision tasks like image classification and object detection [40], robotic tasks based on visual perception often require RGB-D or 3D point cloud data to support manipulation reasoning. Recent general-purpose object grasping datasets include Jacquard [42], GraspNet-1billion [54], while most of robotic grasping works experiment with smaller scale datasets [122, 177, 32]. For object pose estimation, LINEMOD [75], T-LESS [77], YCB-Video [192], Rutgers-APC [153], IC-BIN [49], HOPE [183], etc., were contributed each with a small set of object instances with 6D pose annotations. More recently, category-level object pose estimation became a new research topic based on the NOCS dataset [188] with 6 categories: *bottle*, *bowl*, *camera*, *can*, *laptop*, and *mug*. The annotation approach of object poses mostly depends on the estimated camera trajectories estimated with external hardware like fiducial tags [143], turntable [91] or motion capture systems, or without hardware using RGB structure-from-motion or depth-based fusion systems like KinectFusion [139], ElasticFusion [190], etc.. Then the object poses are manually annotated once in 3D workspace, and calculated to each image frame based on the camera poses. LabelFusion [126] was proposed as a tool in such way to create customized large-scale dataset which uses ICP to assist object model registration to the reconstructed 3D point clouds. Besides real-world datasets, synthetic data generation is another import source to create large-scale training data for deep networks. Platforms like Unreal NDDS [181], BlenderProc [41], Unity Perception [12], etc., support photorealistic rendering and domain randomization that could create a large amount of data with object shape models from public model sets [19, 142]. Besides, some robotic simulation software like NVIDIA Isaac Sim [38], AI2THOR [96], Habitat 2.0 [175], etc., could also support ray-tracing in rendering and generate pose data. Despite the smaller difference between synthetic and real data, the sim2real gap still remains an active research question. A recent work [197] developed a crowdsource UI to augment open-source object CAD models onto scanned real-world scenes for 3D tabletop scene understanding.

CHAPTER 3

GRIP: Generative Robust Inference and Perception for Semantic Robot Manipulation in Adversarial Environments

3.1 Motivation

Taking advantage of the renaissance in deep neural networks, machine learning has achieved great progress in object detection and segmentation and image recognition. These deep learning methods are also prevalent in robotics for problems, including manipulation in clutter [62] and learning of manipulation actions [103]. For 6D object pose estimation, learning-based Convolutional Neural Networks (CNNs) have achieved promising accuracy and real-time inference speed [192, 182, 187]. Notably, these successes rely on well-designed models and adequate training resources. The robustness and generalization capability of CNNs heavily depend on the training data, which represents a certain range of conditions that could be faced by robots. However, due to the complex and dynamic nature of the real world, robots are subject to unforeseen environmental conditions, which are not present in the training data.

More specifically, CNNs recognition systems introduce vulnerability to errors (both benign and malicious) due to the effects of overfitting during the training process. Distorted objects and/or objects captured under poor lighting conditions could be enough to defeat the recognition abilities of a CNN [51]. Such perception errors can lead to (potentially disastrous) outcomes for embodied systems acting in the real world. These challenges for **robust perception** become that much more challenging when an adversary can modify the environment to exploit the vulnerabilities of a CNN.

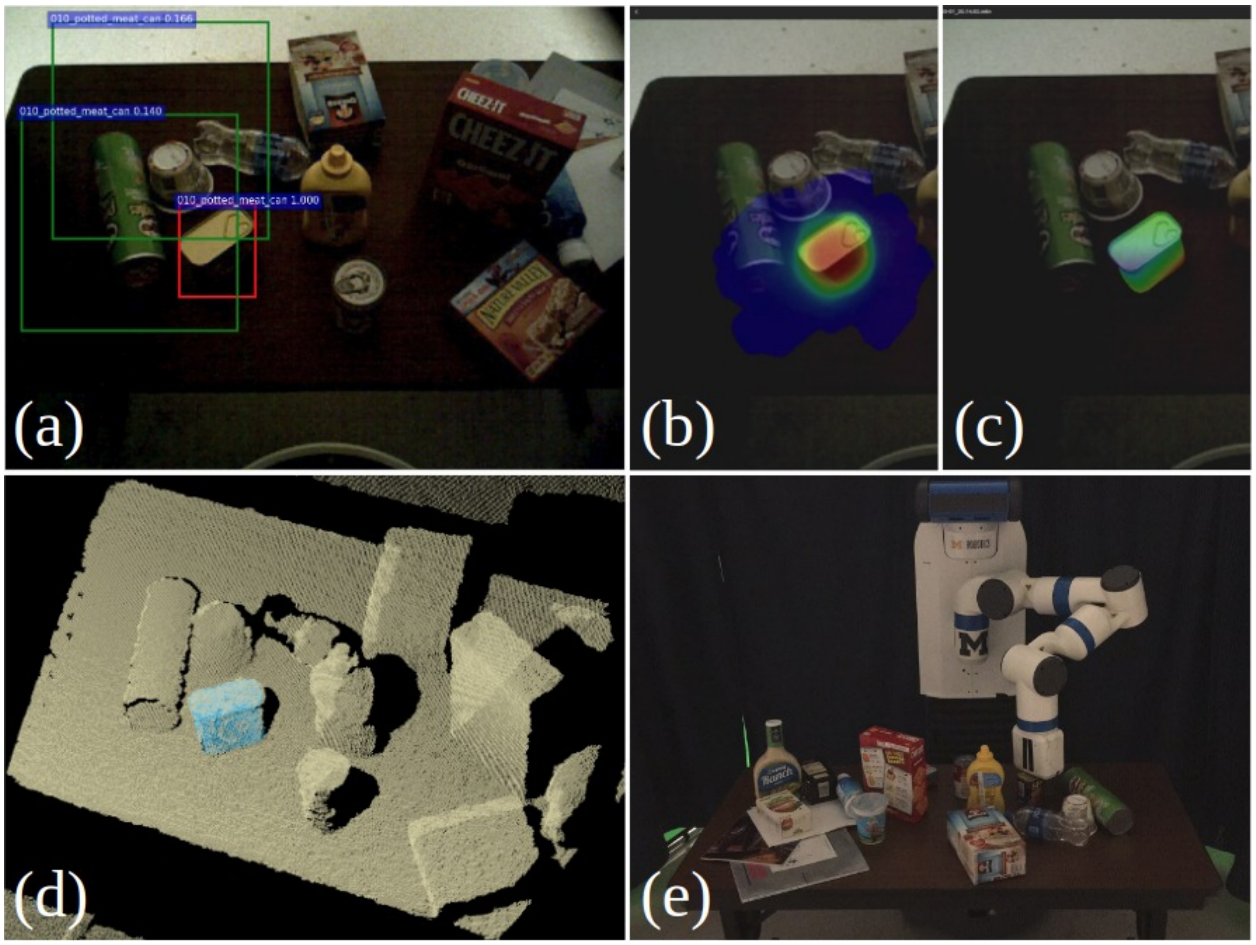

010_potted_meat_can 0.166
010_potted_meat_can 0.140
010_potted_meat_can 1.000
(a)
(b)
(c)
(d)
(e)

Figure 3.1: Our *GRIP* system perceiving and grasping an object in adversarially darkened lighting. *GRIP* uses two stages of (a) PyramidCNN object detection bounding boxes with confidence score greater than 0.1 (green boxes), shown along with the ground truth (red box), and (b) sample-based generative inference. The (c) resulting estimate and (d) its localized pose (highlighted in cyan) enables (e) the Michigan Progress Fetch robot to accurately grasp the potted meat can object.

For instance, in the context of object recognition for a robotic system, a possible malicious attack (through simple modifications of an environment) has the potential to drastically alter and even manipulate a robot's final behavior. Fig. 3.1 shows such a robot manipulation task under dark scene.

Generative-discriminative algorithms [173, 113] offer a promising avenue for robust perception. Such methods combine inference by deep learning (or other discriminative techniques) with sampling and probabilistic inference models to achieve robust and adaptive perception in adversarial environments. The value proposition for generative-discriminiative inference is to get the best out of existing approaches to computational perception and robotic manipulation while avoiding their shortcomings. We want the robustness of belief space planning [86, 87] without its computational intractability. The recall power of neural networks without excessive overfitting [103]. The efficiency of deterministic inference without its fragility to uncertainty [55, 134]. Generative-discriminative algorithms may be especially advantageous when exposed to adversarial attack, building on foundational ideas in this space [137, 138, 117, 85, 35]. Furthermore, we expect our approach will be more generally applicable to guard against broad categories of attack with a clear pathway for explanability of the resulting perceptual estimates.

In this paper, we present *Generative Robust Inference and Perception (GRIP)* as a two-stage method to explore generative-discriminiative inference for object recognition and pose estimation in adversarial environments. Within *GRIP*, we represent the first stage of inference as a CNN-based recognition distribution. The CNN recognition distribution is used within a second stage of generative multi-hypothesis optimization. This optimization is implemented as a particle filter with a static state process. We show that our *GRIP* method produces comparable and improved performance with respect to state-of-the-art pose estimation systems (PoseCNN [192] and DOPE [182]) under *adversarial scenarios* with varied lighting and cluttered occlusion. Moreover, we demonstrate the compatibility of *GRIP* with goal-directed sequential manipulation in object pick-and-place tasks with a Michigan Progress Fetch robot.

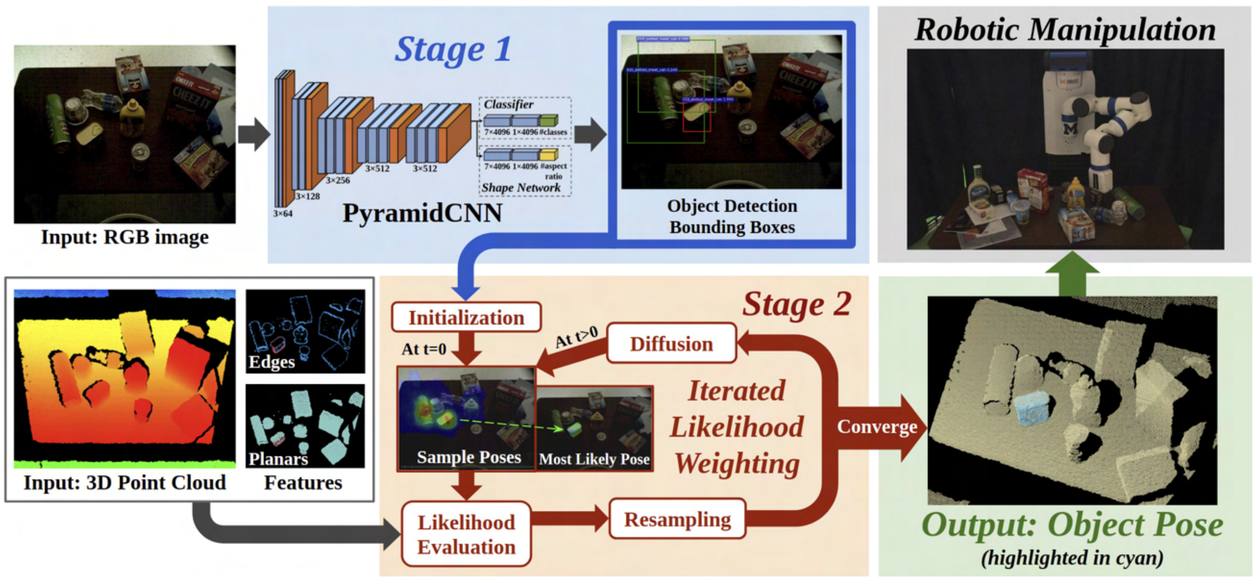


Figure 3.2: Overview of *GRIP*. The robot operating in a dark and cluttered environment is to grasp the meat can from its RGBD observation. Stage 1 takes the RGB image and generates object bounding boxes with confidence scores. Stage 2 takes the depth image and performs sample-based generative inference to estimate the pose for each object in the scene. The samples in Stage 2 are initialized according to bounding boxes from Stage 1. From this estimate, the robot performs manipulation on the meat can object.

3.2 Related Works

To get the best of both worlds, we consider the state-of-the-art as the relative strengths and weaknesses of deep learning and generative inference for robust perception. We are particularly interested in complementary properties of these methods for making perceptual decisions, where the weaknesses of one can be addressed by the strengths of the other. Despite the strengths of CNNs, they have several shortcomings that leave them vulnerable to adversarial action, such as their *opacity* in understanding how its decisions are made, *fragility* for generalizing beyond overfit training examples, and *inflexibility* for recovering when false decisions are produced. For these methods, Goodfellow *et al.* [60] demonstrated that adversarial examples are misclassified both in the case of different architectures or different subsets of the training data. These weaknesses for CNNs play to the strengths of robustness for generative probabilistic inference, which are inherently: *explainable, general, and resilient* through the process of generating, evaluating, and maintaining a distribution of many hypotheses representing possible decisions. However, this robustness comes at the cost of computational efficiency. Probabilistic inference, in contrast to CNNs, is often computationally intractable with complexity that grows exponentially with the number of variables. *GRIP* aims to overcomes these limitations by combining the strengths of deep learning and probabilistic inference through a two-stage algorithm, illustrated in Fig. 3.2 and discussed later in Section 3.4. The remainder of this background section will provide a broader overview of related existing works.

3.2.1 Perception for Manipulation

Perception is a critical step for robotic manipulation in unstructured environments. Ciocarlie *et al.* [34] proposed an architecture for reliable grasping and manipulation, where non-touching, isolated objects are estimated by clustering the surface normal of RGBD sensor data. The MOPED framework [35] has been proposed for object detection and pose estimation using iterative clustering estimation from multi-view features. A bottom-up approach is taken in [144] using RANSAC and Iterative Closest Point registration (ICP), relying solely on geometric information. Narayanan *et al.* [137] integrated global search with discriminatively trained algorithms to balance robustness and efficiency, which works on multi-object identification, assuming known objects.

For manipulation in dense cluttered environments, ten Pas and Platt [178] showed success in detecting grasp affordances from 3D point clouds. In [177], they sample grasp pose candidates based on their geometric plausibility, from which feasible grasp poses are selected by a CNN. Regarding manipulation with known object geometry models, [172, 43, 208] proposed generative sampling approaches to scene estimation for object poses and physical support relations. However, these methods used object detection bounding boxes with hard thresholding as the prior for generative sampling, which might cause false negatives.

3.2.2 Object Detection and Pose Estimation

Learning-based approaches have been used as modules in object pose estimation systems, or directly built end-to-end approaches. Sui *et al.* [173] proposed a sample-based two-stage framework to sequential manipulation tasks, where object detection results are used as prior of sample initialization. Mitash *et al.* [132] developed a two-stage approach, which ran stochastic sampling of congruent sets [129] to get object poses based on the semantic map from a segmentation network. Regarding end-to-end systems, PoseCNN [192] was proposed by constructing a neural-network that learned segmentation, object 3D translation, and 3D rotation separately. This work also contributed a large object dataset, called YCB-Video-Dataset, for benchmarking robotics pose estimation and manipulation approaches. DOPE [182] outperformed PoseCNN in estimation accuracy and robustness in dark and occluded scenes by training the network on a generated synthetic dataset from domain-randomization and photo-realistic simulation. Another recent work, DenseFusion [187], utilized two networks to extract RGB and depth features separately. 6D poses are learned from the combined feature and refined the pose by another residual network module.

In this paper, we focus on the pose estimation problem in *adversarial scenarios*. Liu *et al.* [113] provided insight into handling adversarial clutter, yet provided limited evaluations of its approach or comparisons with state-of-the-art methods. We believe that the performance of CNNs relies highly on the consistency of the testing environment to the training set, and that the same is true for the two-stage methods in [173] and [132] since they rely on high-quality CNN output from their first stages. Our main contribution is the development of a two-stage pose estimation system that is robust under adversarial scenarios and able to recover from false detections from its own first stage.

3.3 Problem Formulation

Given an RGB-D observation (Z_r, Z_d) from the robot sensor and 3D geometry models of a known object set, our aim is to estimate the conditional joint distribution $P(q, b | o, Z_r, Z_d)$ for each object class o, where q is the six DoF object pose and b is the object bounding box in the RGB image. The problem can be formulated as:

$$P(q, b|o, Z_r, Z_d) \tag{3.1}$$

$$= P(q|b, o, Z_r, Z_d)P(b|o, Z_r, Z_d) \tag{3.2}$$

$$= \underbrace{P(q|b, o, Z_d)}_{\text{pose estimation}} \underbrace{P(b|o, Z_r)}_{\text{detection}} \tag{3.3}$$

Equations (3.1) and (3.2) are derived using chain rule statistics and Equation (3.3) represents the factoring of object detection and pose estimation. Here, we assume that pose estimation is conditionally independent of RGB observation, while object detection is conditionally independent of depth observation.

Ideally, we could use Markov Chain Monte Carlo (MCMC) [69] to estimate the distribution of Equation (3.1). However, the state space of the entire states is so large that it is intractable to directly compute. End-to-end neural network methods can also be used to calculate the distribution [192, 182, 187]. These results place a heavy reliance on proper coverage of the input space in the training set. This data reliance makes such methods vulnerable to unforeseen environment changes. SUM [173] implements a combination of Equation (3.1) to filter over hard detections provided by a CNN, thereby enabling it to filter out false positive CNN detections. The limitation of SUM is its inability to recover from false negatives that are eliminated from consideration in object proposal and detection stages. On the other hand, our *GRIP* paradigm is able to compensate for data deficiency by employing a generative sampling method in the second stage.

3.4 Approach

We propose a two-stage paradigm to combine object detection and pose estimation, as shown in Fig. 3.2. In the first stage of inference, PyramidCNN performs object detection and generates a prior distribution $P(b|o, Z_r)$ of 2D bounding boxes for each object label o. In the second stage, we perform generative multi-hypothesis optimization to estimate the joint distribution $P(q, b|o, Z_{(r,d)})$

for each object label o using the first stage output as prior. The second stage is implemented as an iterated likelihood weighting filter [127]:

$$\underbrace{P(q_0, b_0|o, Z_{(r,d)})}_{\text{Sample Initialization}} = P(q_0|b_0)P(b_0|o, Z_r) \tag{3.4}$$

$$P(q_t, b_t|o, Z_{(r,d)}) = \eta \underbrace{P(Z_{(r,d)}|q_t, b_t, o)}_{\text{Likelihood}} \underbrace{\overline{P}(q_t, b_t|o, Z_{(r,d)})}_{\text{Proposal}} \tag{3.5}$$

$$\overline{P}(q_t, b_t|o, Z_{(r,d)}) = \int\int \underbrace{P(q_t, b_t|q_{t-1}, b_{t-1})}_{\text{Diffusion}} \cdot P(q_{t-1}, b_{t-1}|o, Z_{(r,d)})dq_{t-1}db_{t-1} \tag{3.6}$$

where η is the normalizing factor. In Equation (3.4), initial pose q_0 is generated from bounding boxes b_0, which are sampled from the prior distribution generated by the first stage. After the second stage, we get a probability distribution of pose estimation as shown in Equation (3.1). We consider the best estimate as the one with highest probability. Equivalently, best pose q^* satisfies,

$$q^*, b^* = \underset{q,b}{\operatorname{argmax}}\, P(q, b|o, Z_r, Z_d) \tag{3.7}$$

3.4.1 Object Detection

The goal of the first stage is to provide a probability distribution map for an object class o in a given input image. To achieve this, we exploit the discriminative power of CNNs. Inspired by region proposal networks (RPN) in [152], our PyramidCNN serves as a proposal method for the second stage. We choose VGG-16 networks [165] to extract features, which are directed to two fully convolutional networks (FCN) [118]: a classifier learning the object labels and a shape network learning the bounding box aspect ratios. The structure of PyramidCNN is detailed in Fig. 3.2.

The input to our networks is a pyramid of images at different scales. This enables the networks to detect objects with different sizes and appearing at various distances. Thus, the output contains a pyramid of heatmaps representing bounding boxes associated with confidence scores, positions, aspect ratios, and sizes for each object class. Different from end-to-end learning systems, we do not apply any threshold to the confidence scores in order to avoid any false negatives generated by the first stage.

3.4.2 Pose Estimation

The purpose of the second stage is to estimate the object pose by performing iterated likelihood weighting, which offers us robustness and versatility over the search space. This is critical in our context since the manipulation task heavily depends on the accuracy of pose estimations. We expect the second stage to perform robustly even with inaccurate detection from the first stage.

3.4.2.1 Initial Samples

We use a set of weighted samples $\{q^{(i)}, w^{(i)}\}_{i=1}^{M}$ to represent the belief of object pose, where each 6D sample pose $q^{(i)}$ corresponds to a weight $w^{(i)}$. Given an object class o, its pose q, and the corresponding geometry model, we can render a 3D point cloud observation $\boldsymbol{r}$ using the z-buffer of a 3D graphics engine. Essentially, these rendered point clouds are what would be observed if the object had the hypothesized poses, which we refer to as *rendered samples* hereafter. The samples are initialized according to the first stage output. As mentioned in Section 3.4.1, our CNN produces a density pyramid that is essentially a list of bounding boxes with confidence scores. We perform importance sampling over the confidence scores and initialize our samples uniformly within the 3D workspaces indicated by sampled bounding boxes as shown in Equation (3.4). More samples are spawned within bounding boxes with higher confidence scores.

3.4.2.2 Likelihood Function

The weight of each sample is calculated by the likelihood function, which evaluates the compatibility of a sample with observations as shown in Equation (3.5). The likelihood function consists of several parts, including bounding boxes weight, raw pixel-wise inlier ratio, and feature-based inlier ratio. We first define the raw pixel-wise inlier function as:

$$\text{Inlier}(p, p^{'}) = \mathbf{I}\left(||p - p^{'}||_2 < \epsilon\right), \tag{3.8}$$

where $p, p^{'} \in \mathbb{R}^3$ refer to a point in observation point cloud $\boldsymbol{z}$ and a point in rendered point cloud from sample pose respectively. $\mathbf{I}$ is the indicator function. A rendered point is considered as an inlier if it is within a certain sensor resolution range ϵ from an observed point. The point-wise inlier ratio of a rendered sample is then defined as:

$$I = \frac{1}{|\boldsymbol{r}|} \sum_{(u,v) \in z} \text{Inlier}(\boldsymbol{r}_{(u,v)}, \boldsymbol{z}_{(u,v)}), \tag{3.9}$$

where (u, v) refers to 2D image indices in the rendered sample point cloud $\boldsymbol{r}$ and observation point cloud $\boldsymbol{z}$. $| \cdot |$ refers to point cloud size.

Besides raw point-wise inliers, we extract geometry feature point clouds from both rendered samples and observation point clouds and compute feature inlier ratios. Hereby, we enhance the robustness of the likelihood function by considering contextual geometric information from 3D point clouds. This term prunes wrong poses that agree with the observation only in individual points but neglect higher-level geometric information such as depth discontinuity and sharp object surfaces. We apply feature point extraction introduced by Zhang *et al.* [210] based on local surface smoothness:

$$c_{(u,v)} = \frac{|| \sum_{(u',v') \in \mathrm{N}(u,v)} \left(\boldsymbol{p}_{(u',v')} - \boldsymbol{p}_{(u,v)}\right) ||}{|\mathrm{N}(u, v)| \cdot ||\boldsymbol{p}_{(u,v)}||} \tag{3.10}$$

$c_{(u,v)}$ is calculated by adding all displacement vectors from $\boldsymbol{p}_{(u,v)}$ to each of its neighbor points $\mathrm{N}(u, v)$. The point cloud $\boldsymbol{p}$ here can be either rendered sample $\boldsymbol{r}$ or observation $\boldsymbol{z}$. The value is then normalized by the size of $\mathrm{N}(u, v)$ and the length of vector $\boldsymbol{p}_{(u,v)}$. Intuitively, c describes the depth changing rate within a certain local range, which has larger values in areas with acute depth changes and smaller values where object surfaces are consistent. We extract two features, edge points and planar points, by selecting point sets with largest and smallest c values respectively. To balance feature point density in areas with different observation quality, we set a maximum number of edge points and planar points to be extracted from a certain local area. Essentially, a point at (u, v) can be selected as an edge or a planar point only if its c value is larger or smaller than a threshold and if the number of selected points has not exceeded the limit. We find that the algorithm is insensitive to our feature extraction parameters. Finally, we apply feature extraction on both rendered sample and scene observation point cloud to get sample features and observation features. We use the same inlier calculation in Equations (3.8) and (3.9) to calculate feature inlier ratios.

The weight w of each hypothesis q is defined as

$$W(q) = \underbrace{\alpha_{box} w_{box} + \alpha_b I_b}_{\text{network terms}} + \underbrace{\alpha_r I_r + \alpha_e I_e + \alpha_p I_p}_{\text{geometric terms}} \tag{3.11}$$

where w_{box} is the confidence score of the bounding box. I_r is the ratio of pixel-wise inliers in the whole rendered sample point cloud. I_b is the inlier ratio in the portion of rendered sample that is within the bounding box (I_b is 0 if no rendered sample point falls into the bounding box). I_e and I_p are inlier ratios in sample edges and sample planars with respect to observation features. The coefficients α_* represent the importance of each likelihood term and sum up to 1. The first two terms, w_{box} and I_b, *network terms*, are heavily determined by the bounding boxes and describe the consistency between pose sample and detection result. The last three terms, *geometric terms*, weigh how much the current hypothesis explains itself in the scene geometry.

3.4.2.3 Update Process

To produce object pose estimations, we follow the procedure of iterated likelihood weighting by first assigning a new weight to each sample. Resampling is done with replacement according to sample weights. During the diffusion process shown in Equation (3.6), each pose $q_t^{(i)}$ is diffused in the space subject to zero-mean Gaussian noises $\mathcal{N}_{T,t}(0, \sigma_{T,t}^2)$ and $\mathcal{N}_{R,t}(0, \sigma_{R,t}^2)$ with time-varying variances for translation and rotation respectively. The standard deviations $\sigma_{T,t}$ and $\sigma_{R,t}$ at iteration t are decayed according to $W(q_t^*)$, the weight of best pose estimation q_t^* at that iteration. Bounding boxes $b_t^{(i)}$ are diffused uniformly within the image. The algorithm terminates when $W(q_t^*)$ reaches a threshold $\bar{w}$, or the iteration limit is reached. Finally, we assume the pose weights of objects in the scene will be much higher than those for non-existing objects.

3.5 Experiments

3.5.1 Implementation

We use PyTorch for our CNN implementation based on a VGG16 model pre-trained on ImageNet [40]. (more architectures are tested in [113]). The shape network branch of our CNN predicts 7 different aspect ratios. The size of a training image is 224×224 and contains a single object. The aspect ratio of an object in the training image can be inferred from the width and height of the object. We apply a softmax at the end of the network to generate probability distribution of object classification and aspect ratio. We use cross entropy as the loss function in training.

Our second stage pose estimation relies on the OpenGL graphics engine to render depth images with 3D geometry models and hypothesis poses on Nvidia GTX1080/RTX2070 graphics cards. During the iterated likelihood weighting process, we allocate 625 samples for each iteration and run the algorithm for 400 iterations in total, with ϵ set to 0.005m. The sample size is limited by the buffer size of our rendering engine, while the iteration limit was set since our pose estimation converges after approximately 150 iterations (see, e.g., Fig. 3.3) in less than 10s ($\sim$60ms per iteration). Point distance threshold ϵ was set to approximated distance between adjacent points in 3D point clouds.

In the feature extraction mentioned in Section 3.4.2.2, we extract up to 5 edge points and 2 planar points from each 5×5 pixel non-overlap sliding window. Given a 3D point cloud $\boldsymbol{p}$, we consider $\boldsymbol{p}_{(u,v)}$ as an *edge point* if $ln(c_{(u,v)}) \geq -5.5$ (see Equation (3.10)), or a *planar point* otherwise. These hyper-parameters are determined experimentally for clear indication of object boundaries as well as surfaces. The likelihood coefficients are set to $\alpha_{box} = 0.1, \alpha_b = 0.1, \alpha_r = 0.3, \alpha_e = 0.25, \alpha_p = 0.25$. Through experiments, we find that the system performance is sensitive to the total category weight allocated to network terms and geometric terms, rather than the allocation within each

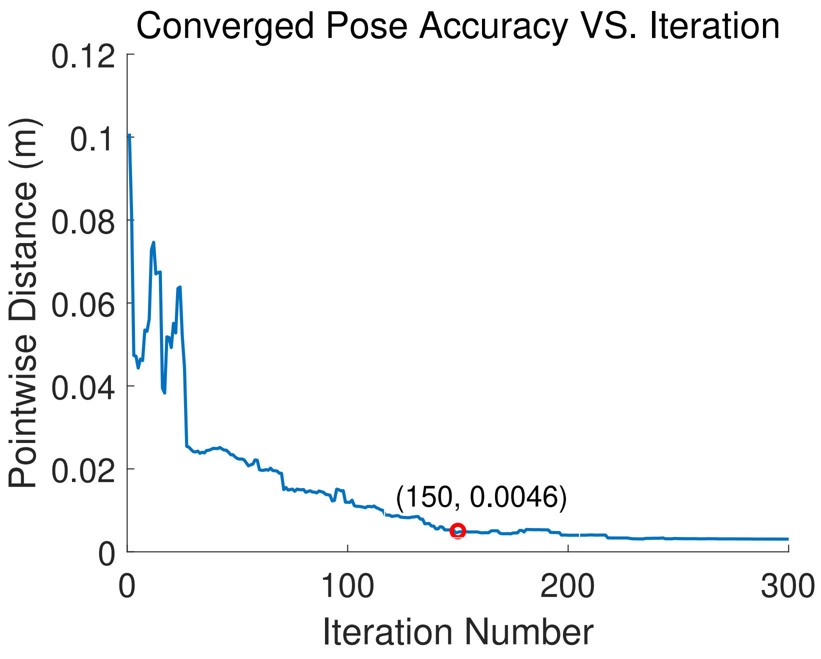


Figure 3.3: Plot of pose accuracy vs. iteration numbers of all converged trials. The point-wise distance is calculated using ADD and ADD-S metrics [192] and not used in pose estimation. After about 150 iterations, the point-wise distance reaches below 0.005m.

category. If the first stage produces accurate detection evaluated by mean average precision (mAP), one can take advantage of it by allocating more weight to network terms. Otherwise, one should reduce the weight of network terms to attenuate the negative impact of underperforming first stage. Since our first stage produces low-mAP detection, we allocate only 20% of the weight on network terms. We allocate the remaining 80% to geometric terms since these terms are robust to adversarial scenarios and unreliable first stage detection. Further weight allocation within each category is done approximately evenly.

During diffusion, standard deviations of the Gaussian noises are decayed by a common factor λ_t, which drops exponentially from 1.0 to 0.0 as $W(q_t^*)$ increases from 0.6 to 1.0. In other words, the standard deviations at iteration t are given by $\sigma_{*,t} = \lambda_t \sigma_{*,0}$, where

$$\lambda_t = \mathbf{I}_{W(q_t^*) \geq 0.6} \cdot \left(\frac{1 - W(q_t^*)}{1 - 0.6} \right)^5 + \mathbf{I}_{W(q_t^*) < 0.6} \cdot 1 \tag{3.12}$$

Initial standard deviations are $\sigma_{T,0} = 0.07m$ and $\sigma_{R,0} = 0.3rad$ for translation and rotation respectively. The threshold $\bar{w}$ for convergence is set to 0.9.

3.5.2 Dataset and Baselines

We use the YCB video dataset [192] as the training data for our first stage PyramidCNN. The YCB video dataset consists of 133,827 frames of 21 objects under normal conditions with balanced and adequate lighting but no occlusion. To test the performance of our two-stage method with baseline methods, PoseCNN [192] and DOPE [182], we collect a testing dataset (i.e., adversarial YCB dataset) from 40 scenes with 15 out of 21 objects from YCB video dataset under adversarial scenarios. In each scene, we place 5-7 different objects on a table and collect seven frames: one in normal lighting, one in darkness, two with different single light sources, and three with different cluttered object placements (see Fig. 3.4). The dark setting and two single-lighting settings cause bias in image pixels values from the training set and thus undermine network prediction. We refer to these settings as *varied lighting* for simplicity. In addition, object clutter causes occlusions as well as natural information loss and challenges the robustness of pose estimation algorithms. All the scene images and 3D point clouds are gathered by the RGB-D sensor on our Fetch robot. Ground truth bounding boxes and 6D poses are manually labeled.

3.5.3 Evaluation

3.5.3.1 Comparing accuracy with PoseCNN and DOPE with 4 YCB objects

We compare our pose estimation accuracy with PoseCNN (with ICP refinement) and DOPE on the YCB dataset under different adversarial settings. We use pre-trained models from the authors' Github page for PoseCNN[1] and DOPE[2] and train our first stage PyramidCNN using 2500 frames from the original YCB video dataset. Since DOPE is trained with 5 of 21 objects from the YCB Video Dataset, we first compare all three methods on 4 of them: 003_cracker_box, 005_tomato_soup_can, 006_mustard_bottle and 010_potted_meat_can. The fifth object, 004_sugar_box,

[1]https://rse-lab.cs.washington.edu/projects/posecnn/

[2]https://github.com/NVlabs/Deep_Object_Pose

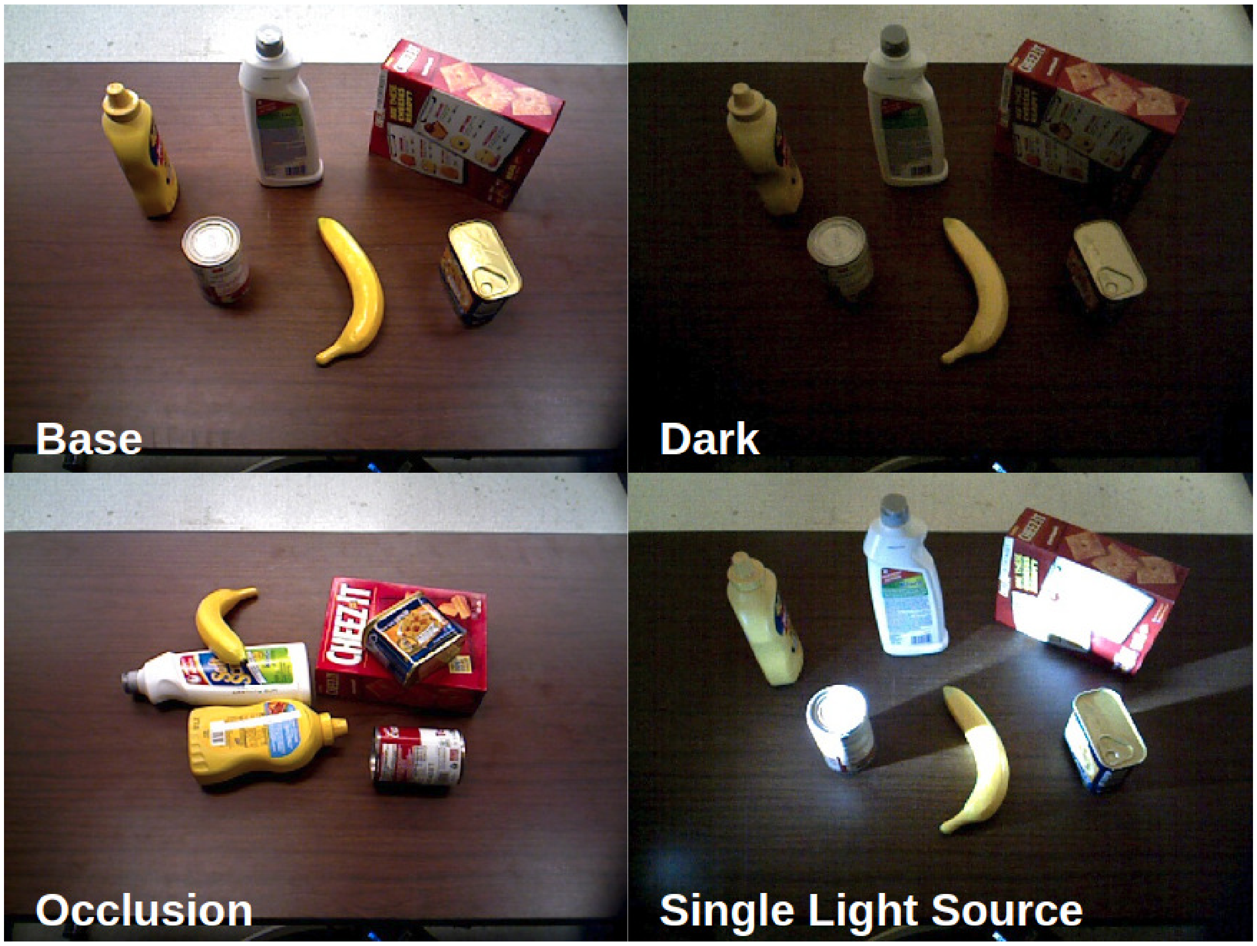
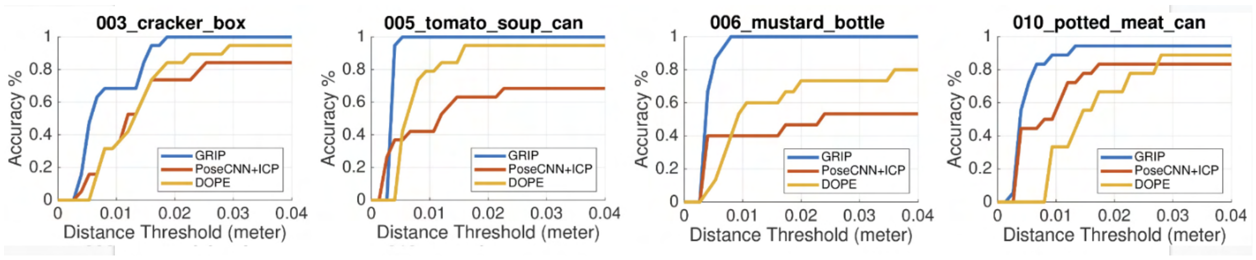
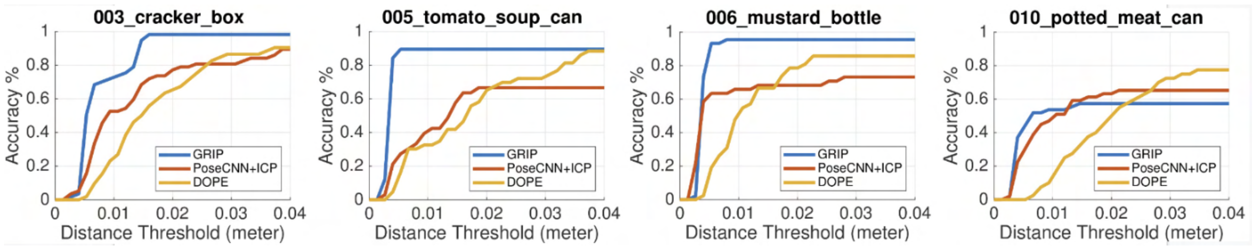


Figure 3.4: Testing dataset with YCB objects under adversarial settings. The base-setting data is collected with regular lighting without occlusions. The dark-setting data is collected with lights off in the room. The single lighting data is collected with a flash light. Object poses are the same in previous three settings. Data in three occlusion scenes is collected with the same objects randomly stacked.

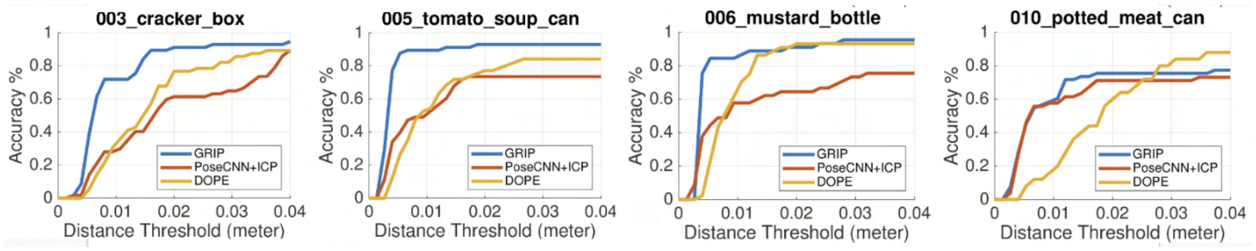
was unavailable from the market when this experiment was set up. We use ADD and ADD-S metrics [192] to calculate pose error for asymmetric and symmetric objects (marked with asterisks in Table. 3.1) respectively. In manipulation tasks, the bearable pose estimation error is bounded by the clearance that objects have when placed in the robot end effector. Based on the sizes of Fetch robot gripper and target objects, we choose 0.04m as the maximum error tolerance. We then plot accuracy-threshold curves within a range of [0.00m, 0.04m] in Fig. 3.5 and calculate AUC (Area Under accuracy-threshold Curve) as the evaluation metric. *GRIP* outperforms the other two methods under most error thresholds, especially lower ones, and thereby facilitates robotic manipulation tasks. See Fig. 3.6 for a qualitative comparison of all three methods with different adversarial



(a) Base settings.



(b) Varied Light settings.



(c) Occlusion settings

Figure 3.5: The comparison between DOPE, PoseCNN+ICP and our *GRIP* two-stage method on pose estimation accuracy of 4 objects mentioned in Sec. 3.5.3.

settings.

3.5.3.2 Comparing accuracy with PoseCNN with 15 YCB objects

Next, we perform an extensive comparison of our method with PoseCNN (with ICP) on 15 of the 21 YCB objects. Table 3.1 and Fig. 3.7 show our overall results and detailed accuracy evaluations for each object.

GRIP outperforms PoseCNN+ICP for most objects under all 3 settings. All methods have worse performances under varied lighting and occlusions as opposed to the basic setting. We can infer the strengths and weaknesses of each method from its performance variance among different objects. For example, PoseCNN with ICP performs better on symmetric objects such as 003_cracker_box and 061_foam_brick as opposed to others such as 021_bleach_cleanser. Symmetric

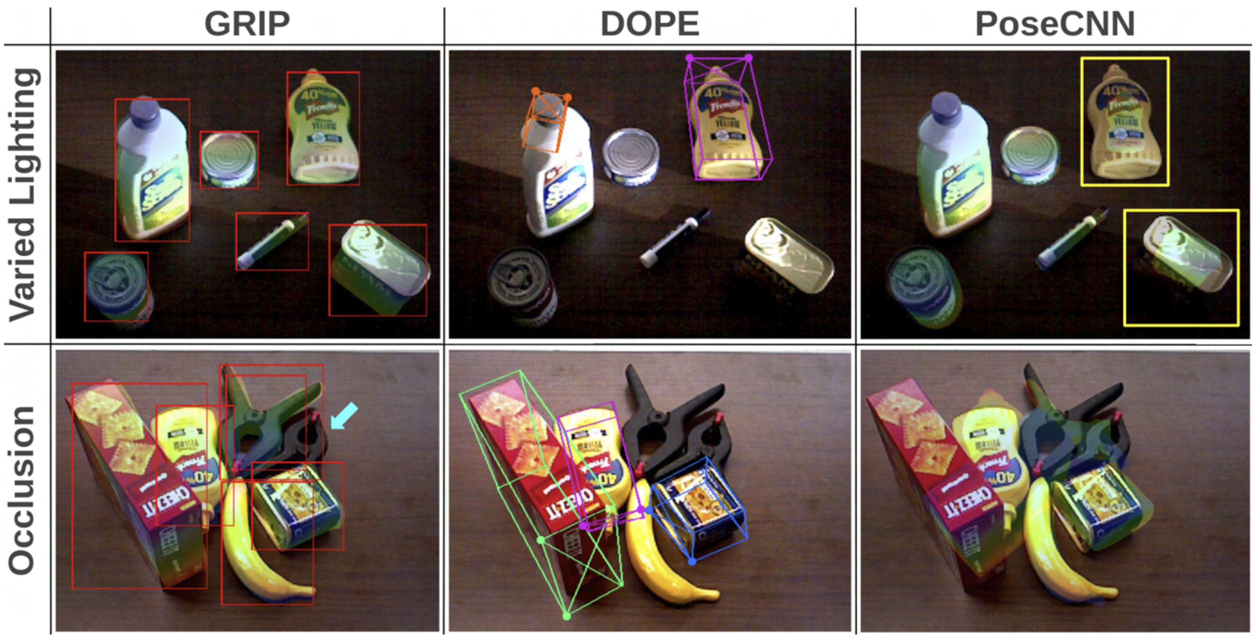


Figure 3.6: Comparison of *GRIP*, DOPE and PoseCNN under adversarial scenarios. In varied lighting condition, DOPE only detects 006_mustard_bottle correctly while PoseCNN makes inaccurate depth estimation (marked yellow). In occlusion condition, DOPE misses half of the objects while PoseCNN fails to detect object poses in clutter. *GRIP* correctly detects all objects under both scenes except 051_large_clamp under occlusion setting (cyan arrow) where the sampling converges to object 052_extra_large_clamp because of their geometric similarity.

objects contain repetitive features which are more likely to be captured by learning-based systems. *GRIP* performs better on objects that are well recognizable under a depth camera. Large and compact objects such as 006_mustard_bottle and 024_bowl naturally generate dense and continuous 3D point cloud observations that effectively capture their geometry. Objects with thin or articulated parts, such as 037_scissors, 052_extra_large_clamp, and 025_mug, produce sparse point clouds around their handle-like parts that do not effectively reveal the scene geometry, especially object

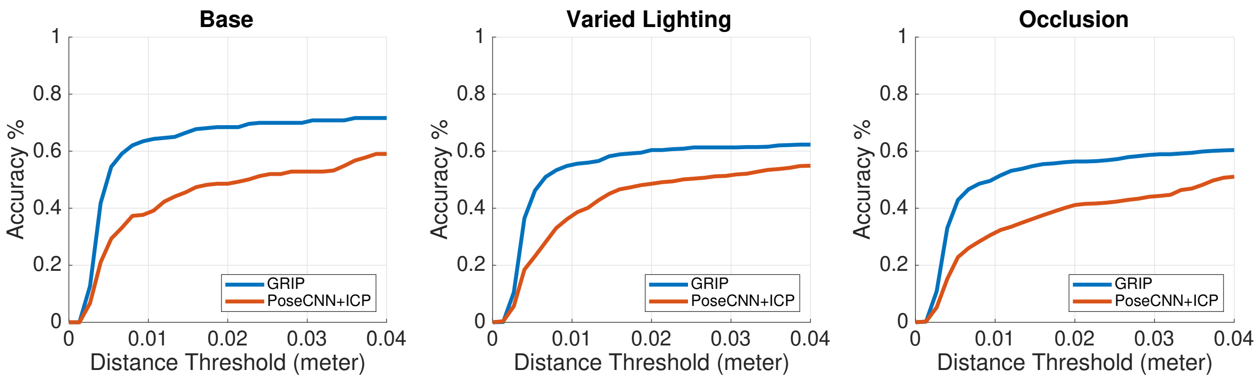


Figure 3.7: Overall pose estimation accuracy of 15 YCB objects using PoseCNN and our *GRIP* method.

Area Under Accuracy-	Base			Varied Lighting			Occlusions		
Threshold Curve	DOPE	PoseCNN*	**GRIP**	DOPE	PoseCNN*	**GRIP**	DOPE	PoseCNN*	**GRIP**
003_cracker_box*	0.6384	0.5925	**0.7878**	0.5509	0.6225	**0.7923**	0.5703	0.4850	**0.7442**
005_tomato_soup_can*	0.7691	0.5535	**0.9015**	0.5326	0.5181	**0.8104**	0.6372	0.6013	**0.8347**
006_mustard_bottle*	0.5720	0.4280	**0.8860**	0.6290	0.6310	**0.8552**	0.7295	0.5864	**0.8208**
007_tuna_fish_can*	\	0.3763	**0.7670**	\	0.3616	**0.7849**	\	0.3915	**0.6220**
010_potted_meat_can*	0.5556	0.6756	**0.8226**	0.4347	**0.5273**	0.5006	0.5045	0.5962	**0.6342**
011_banana	\	0.3922	**0.5467**	\	0.3449	**0.5591**	\	0.2137	**0.2750**
019_pitcher_base	\	**0.2442**	0.1774	\	**0.2490**	0.1659	\	**0.3341**	0.1874
021_bleach_cleanser	\	0.3302	**0.5671**	\	0.3111	**0.5523**	\	0.2204	**0.4635**
024_bowl*	\	0.3190	**0.8674**	\	0.3397	**0.7109**	\	0.2345	**0.6185**
025_mug	\	**0.3491**	0.2201	\	**0.3176**	0.2170	\	**0.2216**	0.2094
037_scissors	\	**0.5450**	0.3192	\	**0.5812**	0.1647	\	**0.3548**	0.1783
040_large_marker*	\	0.2071	**0.7537**	\	0.4094	**0.6750**	\	0.2736	**0.6711**
051_large_clamp	\	0.2405	**0.4927**	\	**0.2061**	0.1642	\	0.0645	**0.2551**
052_extra_large_clamp	\	**0.4000**	0.1742	\	0.1460	**0.1742**	\	**0.2147**	0.1441
061_foam_brick*	\	0.8094	**0.8333**	\	0.6380	**0.8011**	\	0.5419	**0.8297**
Overall	\	0.4308	**0.6078**	\	0.4136	**0.5285**	\	0.3556	**0.4992**

Table 3.1: Overall Performance (Area Under accuracy-threshold Curve) of 15 YCB Objects on DOPE, PoseCNN with ICP and our *GRIP* method. Symmetric objects are marked with stars and evaluated using ADD-S; asymmetric objects are evaluated using ADD.

orientations. Hence, our *GRIP* algorithm best suits scenarios where rich depth sensory data are available due to detectable object dimensions and surface materials or high-definition depth sensors. Finally, distinguishing near-identical objects remains challenging. For instance, 051_large_clamp and 052_extra_large_clamp have identical colors and shapes and differ only insignificantly in sizes. This results in poor estimation accuracy by all methods.

3.5.4 Robotic Manipulation

GRIP has been successfully used as the perception module in real robot manipulation tasks, such as a grocery packing task shown in the video[3]. Figure 3.8 shows snapshots from the sequence of the task.

3.6 Summary

We have introduced *GRIP* as a two-stage method for robust 6D object pose estimation suited to adversarial settings. *GRIP* demonstrated similar and improved performance with respect to

[3]https://www.youtube.com/watch?v=W0KvzMhIJxo

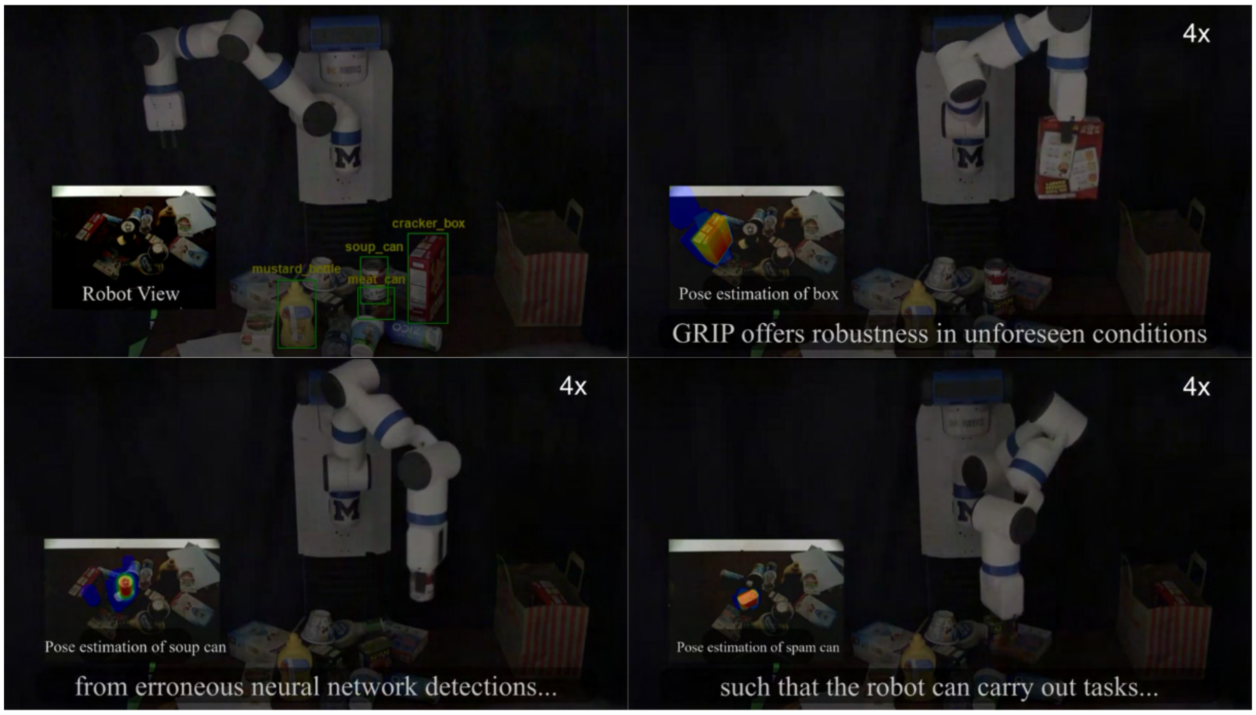

Robot View
cracker_box
soup_can
mustard_bottle
meat_can
4x
Pose estimation of box
GRIP offers robustness in unforeseen conditions
4x
Pose estimation of soup can
from erroneous neural network detections...
4x
Pose estimation of spam can
such that the robot can carry out tasks...

Figure 3.8: A Fetch robot is packing grocery based on pose estimates from the *GRIP* pipeline.

state-of-the-art neural network pose estimators considering the adversarial YCB dataset. The key insight of *GRIP* is to avoid hard thresholding, which introduces false positives and false negatives, until a final pose estimate is required. Avoiding hard thresholds increases the possibility of finding the real pose in adversarial environments. In addition, a generative second stage inherently provides an avenue for explainable perception, without requiring deciphering network weights. Also, this generative process readily extends to tracking over multiple instances of time through the inclusion of a proper process model. The results presented are also amenable to improvement due to the limited types of features considered. These benefits come at the cost of assuming only one instance of each object is present in the scene. For future work, we aim to investigate these limitations through exploring features amenable to robust inference with multiple object instances in greater clutter and sampling techniques for faster convergence.

CHAPTER 4

ACF: Manipulation-Oriented Object Perception in Clutter through Affordance Coordinate Frames

4.1 Motivation

In order for robots to assist people in unstructured environments such as homes and hospitals, they must be capable of performing diverse manipulation actions. Simple pick and place actions are not sufficient for robots to exploit the affordances [59] of objects when executing common tasks, such as grasping a container to pour as shown in Figure 4.1. Object affordances [186] provide an agent-centric way of describing available actions that an object offers given the capability of the agent. To interact with common household objects, a robot must make use of the actions afforded by the component parts of the object. In this work, we exploit the notion of affordance and represent an object class as a composition of functional parts, which we call *affordance parts*. As shown in Figure 4.2, the concept of a mug can be generalized to the composition of two affordance parts, namely the container and the handle parts.

Recently, the notion of affordance for manipulation has garnered increasing interest in the robotics community. Previous work on object affordances for robotic manipulation focuses on recognizing the affordance labels of novel objects [48], which does not directly translate to specific manipulation poses for use by a planner towards task completion. Recent advances have enabled pose estimation of novel objects at the category level [188, 21]. However, robots often need fine-grained pose information about specific object parts (e.g. a container handle) for manipulation

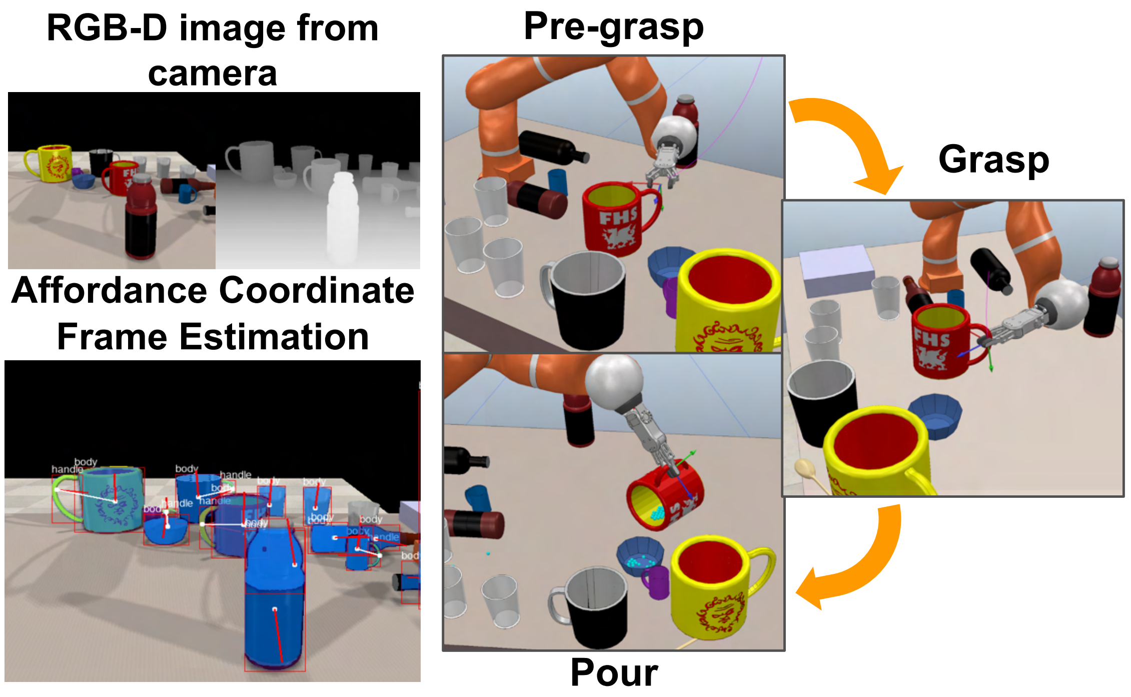


Figure 4.1: Affordance Coordinate Frames (ACF) provide keypoint and axis constraints for each manipulation action afforded by an object. In the above example, we estimate a "grasp" ACF on the handle of a mug, as well as a "pour" ACF on the body of a mug (bottom left). The robot uses the estimated ACFs to execute a task involving pouring liquid from the mug (right). The ACF is generalizable across object instances.

actions. For such tasks, the category-level object pose is not sufficient to perform manipulation, due to large intra-category variance. Generalizable representations for object localization using keypoints have been proposed [125, 149], but they rely heavily on user-specified constraints in order to define manipulation actions over the keypoints.

In this paper, we aim to develop a generalized object representation, the Affordance Coordinate Frame (ACF), that seamlessly links object part affordances and robot manipulation poses. Our key insight lies in that each object class is composed of a collection of functional parts, where each part is strongly associated with its affordances (see Figure 4.2). For robots to exploit the affordances of object parts, we learn a category-level pose for each object part, similarly to the notion of category-level pose for objects [188]. In particular, the category-level pose for each object part serves as a coordinate frame (hence the name Affordance Coordinate Frame) in which to pre-define manipulation poses, such as grasp and pour actions. Building on the insights from Zeng et al. [206], our work proposes a deep learning based pipeline for estimating the ACF of

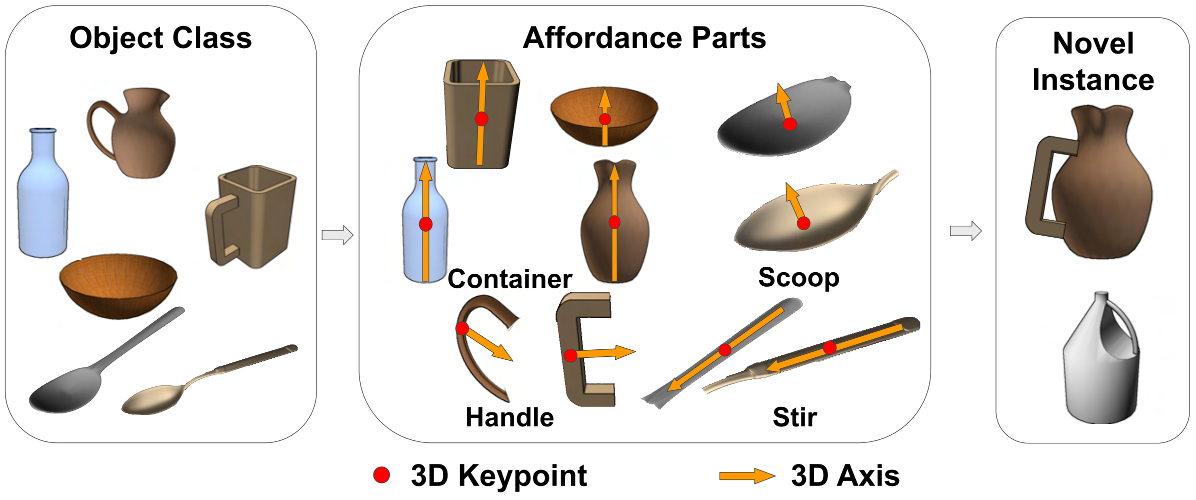


Figure 4.2: Objects can be divided into functional parts which represent their affordances. The Affordance Coordinate Frame (ACF) is defined as a 3D keypoint and a directed axis, which are consistent across affordance classes. The ACF can be used by a robot to plan manipulation actions. The object parts can be re-assembled to generate new instances with novel shapes.

object parts from RGB-D observations. A robot can attach pre-defined manipulation poses to the estimated ACF when executing tasks. By defining the ACF with respect to object parts, our method allows for better generalization and robustness under occlusion, because the robot can see and act on observable parts directly. For example, the handle of a mug can be grasped when its container part is not visible. In contrast to works aimed at pose estimation for robotic manipulation [182, 146], we do not assume known 3D object geometry models or 6D object poses. We show the accuracy and generalization of our method in detecting novel object instances, as well as estimating ACF for novel object parts in our experiments. We demonstrate that the proposed method outperforms the state-of-the-art methods for object detection, as well as category-level pose estimation for object parts. We then demonstrate manipulation tasks through ACF estimation in cluttered environments in both simulation and on a Fetch manipulation platform.

4.2 Related Works

In the following subsections, we discuss related works in three aspects: the *affordance* concept initiates the proposition of our work that connects visual perception and robotic manipulation through our proposed representation; the *object pose estimation* basically shows a pipeline of manipulation based on poses of entire objects, with which we compare the estimation accuracy in experiments; the *task-oriented grasping* includes related application to robot grasping, with which we both interact with objects considering part semantics while our pipeline also fits other actions than grasp.

4.2.1 Affordances for Robotic Manipulation

The concept of affordance has been explored in many aspects of robot manipulation. For visual perception, Nguyen et al. [48] proposed AffordanceNet to achieve state-of-the-art accuracy on affordance detection and segmentation on affordance datasets [140, 136]. The affordance detection together with segmentation can reduce search space of grasp pose for manipulation as explored in [45], and grasp center point and axis are fitted from the shape of masks in [33]. Our main contribution is to propose a model that jointly learns part-based 3D keypoints and axes as well as association between parts. On the other hand, various affordance representations have been proposed for task execution. Affordance templates [66] were developed for highly constrained manipulation tasks but require user inputs to manually register it with robot observations. Affordance wayfield [128] extended the similar idea to a gradient field. In [88, 89], affordances are modeled as an existence possibility using certainty or belief functions, based on the relation of environmental shape primitives and end-effector poses. The formulation supports hierarchical relations from basic grasps to bi-manual operations. Our work differs from these works as we define affordances from object part level to be visually perceivable and generalizable to objects with similar functionality.

Recent works that connect visual perception of object affordances with robot manipulations are most relevant to our work. kPAM [125] uses 3D semantic keypoints as the object representation for category-level object manipulations. The core advantage of ACF over kPAM is: when multiple instances are presented in the cluttered environment, ACF has the notion of compatibility to determine which ACFs compose to the same object, so it would be robust to cluttered scenes. Specific to tool manipulation, Qin et al. [149] also proposed a keypoint-based representation, which specifies the grasp point and function point (hammer head for example) on the tool and effect point on the target object. The robot motion was solved through optimization over the keypoint-based constraints. Compared to 3D keypoints registered on whole objects, our ACF representation decomposes object into object parts based on their functionality, and include explicit orientation for manipulation. Similarly, kPAM 2.0 [57] and AffKp [198] also introduces the orientation from/with keypoints to define manipulation constraints.

4.2.2 Object Pose Estimation for Manipulation

Object pose estimation offers a formal way to perform manipulation with known object geometry models. Instance-level pose estimators have been extensively studied in the vision community [182, 74]. In the robotics community, works have focused on the problem of object localization in clutter for the purpose of robotic pick and place tasks [207, 172, 27]. The success of parts-based methods have been demonstrated in highly cluttered scenes [44, 146]. However, they rely on known mesh models and do not extensively validate the representations for manipulation.

Recently, several works aim to extend that to category-level pose estimation, which relaxes the assumption of known object geometry models. Wang et al. [188] proposed Normalized Object Coordinate Space (NOCS) as a dense reconstruction of object category in canonical space, under the assumption that the intra-category shape variance can be well approximated by a representative 3D shape. Chen et al. [21] gave another category-level representation modeled by a latent space vector generated from a variational autoencoder, which does not assume dense point correspondence. A recent work [104] extended [188] to articulated objects, which solves the joint axes and parameters after estimating poses of individual parts. Compared to these representations of whole objects, the ACF focuses on functional parts of objects rather than entire object-level poses.

4.2.3 Task-Oriented Grasping

Task-oriented grasping incorporates semantics into classic grasping to provide richer affordance-based manipulation. Our part-based affordance representation shares the same idea as in [45, 95,

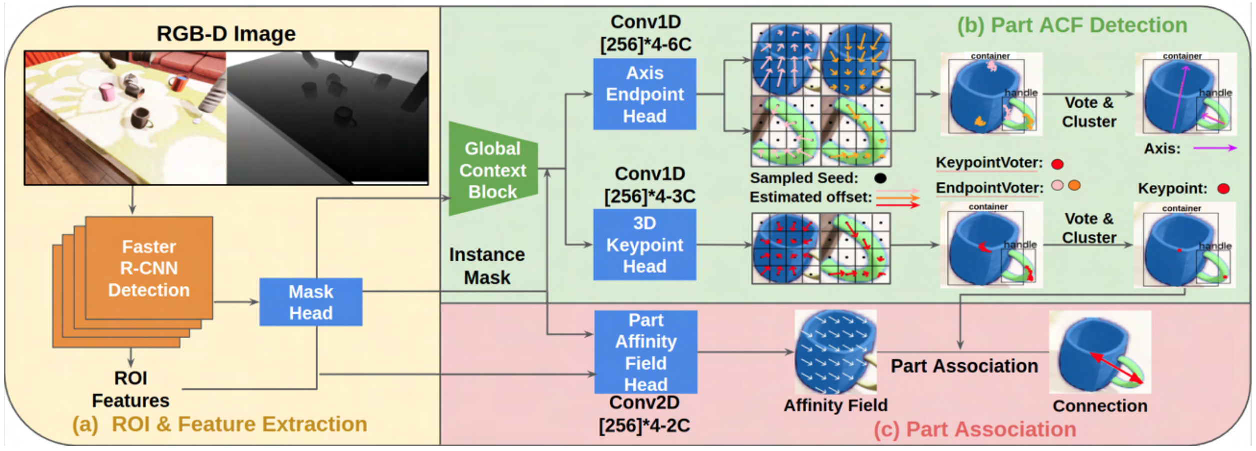


Figure 4.3: The perception pipeline for estimating the affordance coordinate frames of a mug with two affordance parts: a container part and a handle part. (a) The Faster R-CNN module detects bounding boxes from features extracted through feature pyramid network out of a 4-channel fused RGB-depth image. (b) The corresponding region of interest (ROI) features are further fused with global features using self-attention module in [17]. 3D keypoint and directed axis endpoint offsets are learned in two parallel network heads (C is the number of part classes). The voters are formed by adding the learned offsets to uniformly sampled seeds within a bounding box and estimated keypoints and axes are computed by clustering the voter points. (c) Another network head learns 2D part affinity fields, which connect parts of an object.

110] that use task semantics to prioritize different regions of objects for task-oriented grasping detection. Similarly in [167], task-based external wrench is considered through grasp analysis to extend DexNet [123] to a task-oriented grasping service. Compared to [10], our representation also divides the whole objects into functional parts but also adds keypoints and axes for ease of manipulation. In general, we share the common part-based representation for robot manipulation, while ACFs can afford guidance on the motion trajectory after grasping by modeling the whole objects into connected parts.

4.3 Affordance Coordinate Frames

4.3.1 Representation

Given a predefined task, our goal is to detect and localize objects in a scene in terms of their parts such that manipulation actions can be executed towards task completion. The problem is divided into two subtasks: the first is to perceive affordances from RGB-D observations from a robot, and the second is to generate manipulation actions from the perceived affordances. We define affordances as the functional interaction between an object and an agent. For common household objects, such affordances are associated with object parts, rather than the whole object, and we therefore associate ACFs with object parts. An ACF is formally defined as a 3D keypoint with a directed axis with its origin at the keypoint location. The ACF allows the affordance to be perceived from visual data while also acting as a concrete and generalization parametrization of a robot action. Figure 4.2 illustrates the geometric relation of the defined axes. The advantages of the ACF are two-fold:

1. Better generalization to novel objects with similar functionality, since parts in one category have smaller visual difference compared to full objects.
2. They provide a sparse representation while providing enough constraints for robotic manipulation by exploiting the symmetric nature of object parts.

4.3.2 Estimation Pipeline

We estimate instance-level object part ACFs using a network similar to Mask R-CNN [71]. The overall framework is illustrated in Figure 4.3. To incorporate depth information, we add an additional channel to the first backbone layer to accept 4-channel RGB-D input. We add three network head architectures for 3D keypoint, axis and part affinity field estimation. Similar to the mask head, estimates are made in every proposed region of interest (ROI) with a detected bounding box and corresponding ROI feature. We concatenate each ROI feature with a global feature computed using Global Context Block (a self-attention module) from [17] to fuse spatial and channel context. In this pipeline, all three heads use convolutional layers and we employ deep Hough voting [74] to estimate the 3D keypoint and axis.

4.3.2.1 3D Keypoint Estimation

We sample 3D seed points and estimate the keypoint position by learning the position offset between all the seeds and the target keypoint. Each ROI consists of $N \times N$ superpixels. The image pixel coordinates and depth values of the seeds are computed through bilinear interpolation similar to RoIAlign [71]. Their 3D positions are restored through an inverted camera intrinsic transformation.

The keypoint loss is defined as L1 loss of estimated 3D offset against ground truth offset filtered by ground truth mask:

$$f_{\text{vote}}(\text{loss}(i)) = \frac{\sum_{i=1}^{N\times N} \text{loss}(i) \cdot M_i^*}{\sum_{i=1}^{N\times N} M_i^*} \tag{4.1}$$

$$L_{\text{keypoint}} = f_{\text{vote}}\left(\sum_{k\in\{x,y,z\}} \left\| t_i^k - t_i^{k*} \right\|_1 \right) \tag{4.2}$$

where Equation (4.1) describes the voting loss function f_{vote} which averages the loss value $\text{loss}(i)$ at seed i region with the confidence score M_i^*. t_i^k denotes the offset of seed i along direction k, t_i^{k*} denotes ground truth offset.

4.3.2.2 Directed Axis Estimation

We represent the directed axis using two 3D endpoints. We learn two separate 3D keypoints to form the axes using a network structure similar to the 3D keypoint estimation head.

The loss function is composed of three parts. The first loss L_{endpoint} is similar to L_{keypoint}, where the loss is the sum of losses for both endpoints. The second loss L_{axis} encourages the voter points to be near to the axis by calculating the distance from the voter points to the ground truth axis:

$$L_{\text{axis}} = f_{\text{vote}}\left(\left\| (t_{i,m} - t_{i,m}^*) \times n^* \right\|_2 \right) \tag{4.3}$$

where $t_{i,m}^*$ is the ground truth translation from the seed i to the m_{th} axis endpoint, and n^* is the ground truth normalized 3D axis. The third loss $L_{\text{direction}}$ corrects the direction of the estimated axis.

Since the connection of two 3D keypoints determines the axis direction, the directions of connection between corresponding voters of the two keypoints should be close to the ground truth axis:

$$L_{\text{direction}} = f_{\text{vote}} \left(1 - (t_{i,2} - t_{i,1}) \cdot n^{*}\right) \tag{4.4}$$

The final loss for axis estimation is the sum of L_{endpoint}, L_{axis} and $L_{\text{direction}}$. During inference, two endpoints are clustered with the same method as for 3D keypoint estimation. Then, the final output axis is a directed link between endpoints.

4.3.2.3 Part Affinity Field Estimation

In multi-instance environment, we also need to determine which parts can be associated to the whole objects. In human pose estimation community, to represent association property between joints, Part Affinity Field (PAF) was proposed in [18]. PAF is a set of 2D unit vectors that indicate the connecting directions between neighbor limbs. Inspired by this, the network is designed to predict a set of 2D unit vectors, which determine the potential associated part direction for each ROI. The network structure is similar to the mask head, estimating 2D vectors p_i for seed i that point from the corresponding part keypoint to its associated part keypoint. The loss function is defined as:

$$L_{\text{paf}} = f_{\text{vote}}(\|p_i - p^{*}\|) \tag{4.5}$$

where p^{*} is the ground truth 2D unit vector pointing toward the target part keypoint. During inference, the mean direction of affinity fields within estimated mask will be used as final field direction.

4.4 Experiments

We evaluate our method from the following perspectives: part-level pose estimation accuracy; and robustness of applying ACF to robot manipulation.

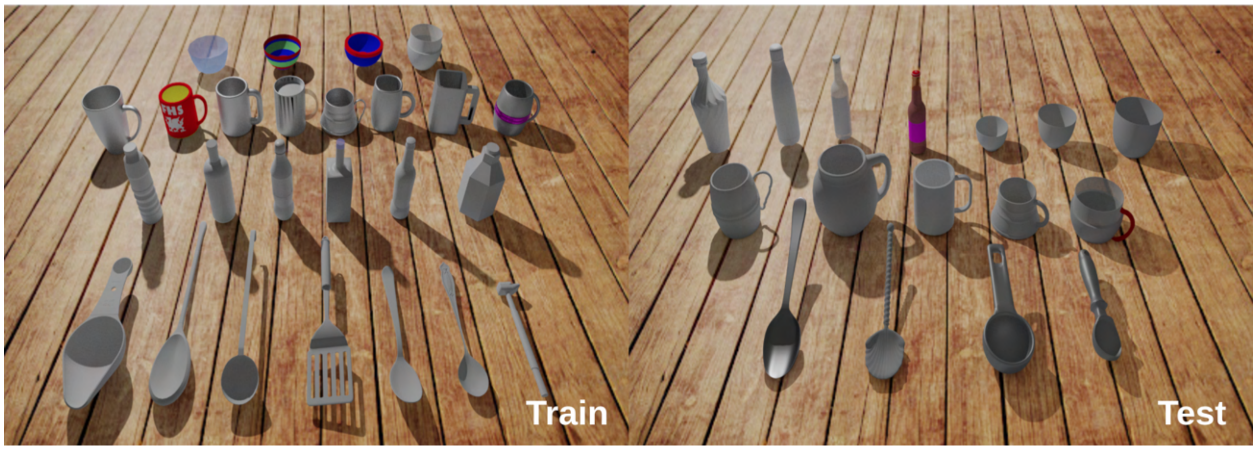


Figure 4.4: The models in the train set and the validation / test set. For the synthetic dataset and robot simulation experiments, we add random texture to each object, as shown in Figure 4.8 and Figure 4.9.

Dataset: We create a synthetic dataset of images which includes RGB, depth, instance segmentation and object part pose in cluttered indoor environments using the NDDS data generation plugin [181] in Unreal Engine 4. We select several hand-scale objects that support robot grasping and manipulation for the drink-serving task. The object models include the bottle, mug and bowl categories from the ShapeNet dataset [19], and the spoon, spatula, hammer categories from public available object model cites like TurboSquid [142], as shown in Figure 4.4. We divide objects into four predefined parts: *container*, *handle*, *stir*, *scoop*. The object-part-action relation is detailed in Table. 4.1. The keypoints for *container* and *scoop* are defined as the center of their geometry shapes. The keypoints for *scoop* and *handle* are the tangent points on the line parallel to the upright orientation. The axes are labeled as follows: *container*, from bottom to top; *handle*, from external tangent point to container direction; *handle*, from the tail endpoint to scoop endpoint; *scoop*, the upwards normal at the center point of concave surface. Examples are shown in Fig. 4.2. During the rendering, all object models are randomly translated and rotated. Scene backgrounds and object textures are randomized. In total, we render 20k images, with 1k images each for validation and test.

Implementation: We use Pytorch to implement the deep network modules. We use ResNet50 [72] with feature pyramid network as a backbone with a pretrained model on the COCO dataset. The initial weight for the input depth channel is set as the average of the 3 RGB channels at the first

Table 4.1: Object-Part-Action Relations. The top table indicates the parts contained in an object with an 'x'. The bottom table indicates the actions which correspond to a part. The 'container' and 'handle' parts are separated in the way shown in Figure 4.2. 'stir' refers to the straight handle part of spoons, spatulas and hammers, and 'scoop' refers to the head part that supports scoop of spoons and spatulas.

	Part			
Object	container	handle	stir	scoop
bottle	x			
mug	x	x		
bowl	x			
spoon			x	x
spatula			x	x
hammer			x	
Action	container	handle	stir	scoop
grasp	x	x	x	
stir			x	
scoop			x	x
contain	x			
pour	x			

layer of the backbone network. The axis endpoint head and the 3D keypoint head consist of shared multi-layered perceptrons. During inference, from each ROI, we select seed samples whose projected coordinates fall into masks, and get 3D position voters by adding estimated offsets. Then, the Mean Shift algorithm [36] is applied to cluster voters, and the center of clusters is regarded as the final estimate. The implementation of the mean shift algorithm for deep Hough voting has the same structure as in He et al. [74]. We trained our network on a Nvidia 2080 Super GPU with a batch size of 2 for 60 epochs, using the Adam optimizer with 0.001 learning rate and 0.001 weight decay.

4.4.1 Ablation Studies on Axis Estimation

Apart from using two 3D keypoints as representation (we call the method **ACF-Endpoints**), we tested another two ways to represent a 3D axis. **ACF-Vector** represents the ACF as a 3D unit vector. During training, the loss L_{vector} is defined to be the difference between estimated vector n and ground truth direction vector n^*. During inference the estimated vector is directly regarded as the axis. **ACF-ScatterLine** uses a 3D point set along the axis, with a binary label indicating the closer endpoint for each point in the set. During training, the loss is defined as the sum of L_{axis}, L_{inner} and L_{label}. L_{inner} pushes points to make offsets t perpendicular to the axis direction n^* so that points spread out along the axis direction which benefits the linear regression during inference. L_{label} corrects the two endpoints order by a binary cross entropy with logits loss (BCELoss in Equation (8)) between estimated closer endpoint index l and ground truth closer endpoint index l^*. During inference, the axis is solved through linear regression with RANSAC from estimated point set and the axis direction is determined by point labels.

$$L_{\text{vector}} = f_{\text{vote}}(\|n_i - n^*\|) \tag{4.6}$$

$$L_{\text{inner}} = f_{\text{vote}}(t_i \cdot n^*) \tag{4.7}$$

$$L_{\text{label}} = f_{\text{vote}}(\text{BCELoss}(l_i, l_i^*)) \tag{4.8}$$

We compare the three variants of learning ACF axis. The metric of calculating angular distance between estimated and ground truth axis n_1, n_2 is defined as $180/\pi \cdot \arccos{(n_1 \cdot n_2)}$. From Table 4.2 and the first row in Figure 4.5, we find ACF-Endpoints achieves the highest mean average precision (mAP) on *container*, *handle* parts and slightly worse as ACF-Vector on *scoop* parts. The reason might be that the *scoop* parts do not span a long distance along the direction of defined ACF axis, so the two endpoints cannot be learned to be as distinct as other parts. ACF-ScatterLine performs much better in *stir* and *container* parts compared to *scoop* and *handle*. We believe that is because the line-shape scatter suits better to straight line or cylinder shapes with a clear center axis or a distinct primary direction. Overall, ACF-Endpoints performs robustly among parts with different shapes and achieves 40.9 mAP within 15°, 5cm threshold.

4.4.2 Comparison with NOCS

We compare our method with a state-of-the-art category-level 6D object pose estimation method **NOCS** [188]. We trained NOCS network to do part category-level pose estimation and compared

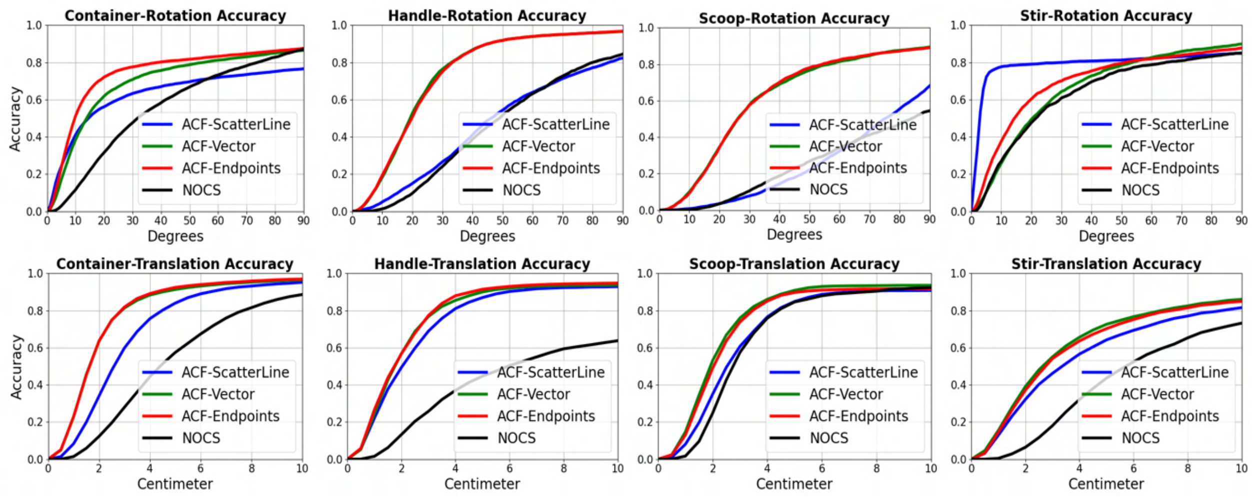


Figure 4.5: Comparison between each method for different part estimation with respect to mAP curves. The top row shows rotation accuracy and the bottom row shows translation accuracy.

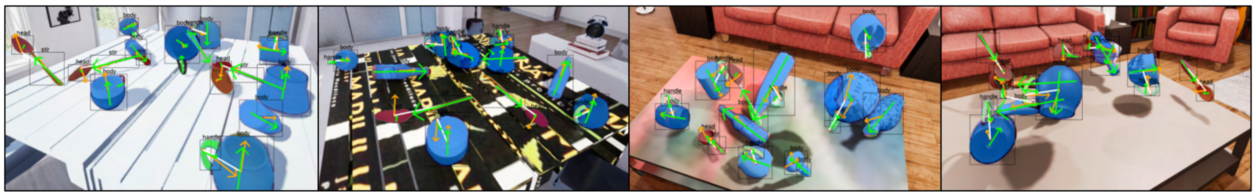


Figure 4.6: Qualitative results on our dataset with the ACF-Endpoints method. The red points show the keypoint of each part, orange arrows show the estimated axes and green arrows show the ground truth axes. Then, the white line shows the connection of two parts.

Table 4.2: Comparison of proposed ACF method with different axis representation and NOCS on object part keypoint position and axis estimation. The numbers inside the table are mean Average Precision (mAP) within different error tolerances.

Part	**Method**	10°|2cm	15°|2cm	10°|5cm	15°|5cm
container	NOCS	4.9	7.2	8.9	15.9
	ACF-Endpoints	**37.7**	**46.3**	**49.7**	**63.1**
	ACF-Vector	28.4	37.8	37.4	51.0
	ACF-ScatterLine	17.0	20.8	36.0	44.7
handle	NOCS	0.4	1.7	0.7	2.6
	ACF-Endpoints	**14.4**	**24.9**	**19.2**	**35.0**
	ACF-Vector	12.1	22.4	17.7	33.6
	ACF-ScatterLine	3.0	6.1	1.4	4.7
stir	NOCS	3.1	4.1	17.9	24.1
	ACF-Endpoints	20.4	26.4	33.3	45.0
	ACF-Vector	15.9	23.2	23.9	35.8
	ACF-ScatterLine	**30.2**	**30.5**	**58.3**	**58.8**
scoop	NOCS	0.0	0.1	0.1	0.3
	ACF-Endpoints	7.1	14.9	9.2	20.4
	ACF-Vector	**7.5**	**15.9**	**10.0**	**20.9**
	ACF-ScatterLine	0.3	0.7	0.7	1.8
mean	NOCS	2.1	3.2	6.9	10.8
	ACF-Endpoints	**19.9**	**28.1**	**27.9**	**40.9**
	ACF-Vector	16.0	24.8	22.2	35.3
	ACF-ScatterLine	12.6	14.5	24.9	28.6

the two methods with respect to part keypoint position and axis direction. The results are shown in Table 4.2 and Figure 4.5. We can figure out that the NOCS does not perform well in our tasks. We think there are two main reasons: (1) The intra-class variance. In the NOCS original paper, they considered object-level classes. For instance, they separate objects into mug, bottle, and bowl classes based on their geometry. However, we regard all of these containers as the container and handle classes based on their functionality. The container parts of mug, bottle and bowl could vary more than that within one object class. (2) The amount of data versus information needed to learn. Compared to the training dataset used in [188], our dataset is about 20X smaller and our training iteration is 4X fewer.

4.4.3 Object-Level Detection Accuracy

Besides part level keypoint and axis estimation for manipulation, we also aim to evaluate object level detection. The intuition is that compared to whole object detection, part-based representation might be more robust in highly cluttered scenes where objects are partly occluded. We choose mugs as test objects and compare our method with a Mask R-CNN baseline. Mask R-CNN achieves 93.2 and 70.6 mAP on uncluttered and cluttered scenes respectively, while ACF-Endpoints achieves 93.3 and 90.2 mAP on uncluttered and cluttered scenes. Both methods perform similarly in uncluttered scenes, while ACF-Endpoints performs better in heavily clutters. The result supports reliability for robotic manipulation in cluttered environments based on the proposed part-level perception method.

4.4.4 Application to Robotic Manipulation Task

To test the perceived ACF applicability, we created a scenario for robotic manipulation to complete a drink-serving task in CoppeliaSim simulation platform [154] as well as using a Fetch robot in real world. During the task execution, we assume individual step sequence (grasp, pour, stir) is given as shown in Figure 4.7 so that after successful grasps, the pour, stir action can be finished without need of re-grasping.

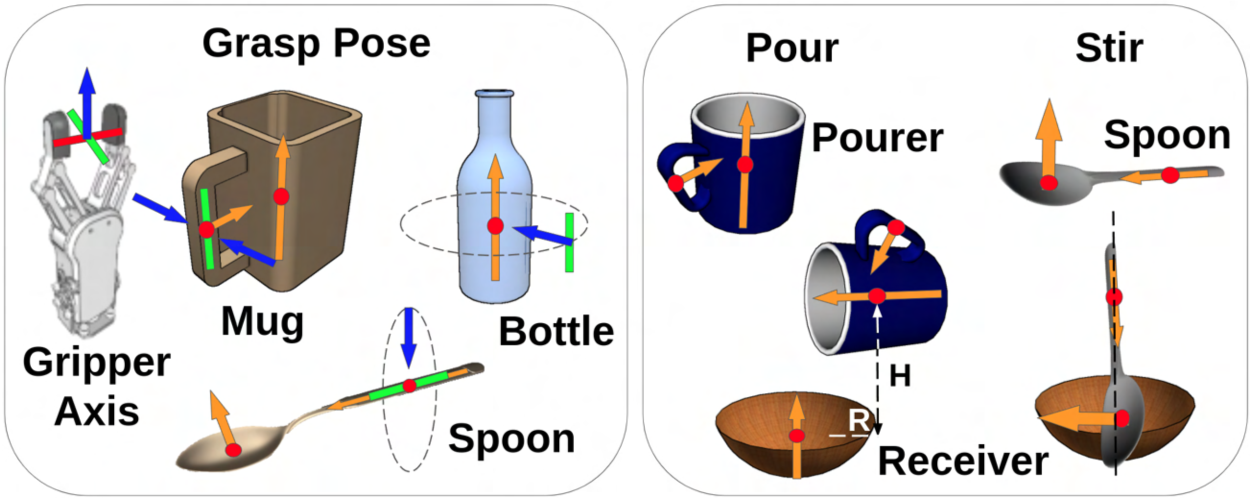


Figure 4.7: Grasp poses and pour, stir manipulation waypoints composed from detected ACFs.

For mugs, the grasp pose is located at handle keypoint, with blue axis orthogonal to both the handle and container axes, and green axis parallel to the container axis; For bottles, the grasp pose is located at container keypoint, with green axis parallel to container axis, and blue axis towards the container keypoint; For spoons, the grasp pose is located at handle keypoint, with green axis parallel to its stir axis, and blue axis towards the handle keypoint. We don't have grasp poses for bowls yet. For pouring, the trajectory of pourer is first constrained to avoid liquid spilling by holding its container part axis to be upright. The pouring action rotates the axis in vertical plane, with keypoint located at a fixed height H=15cm, and a rotation radius R=5cm relative to the receiver's keypoint. For stirring, the spoon is supposed to move downwards with its stir axis aligned with container axis,

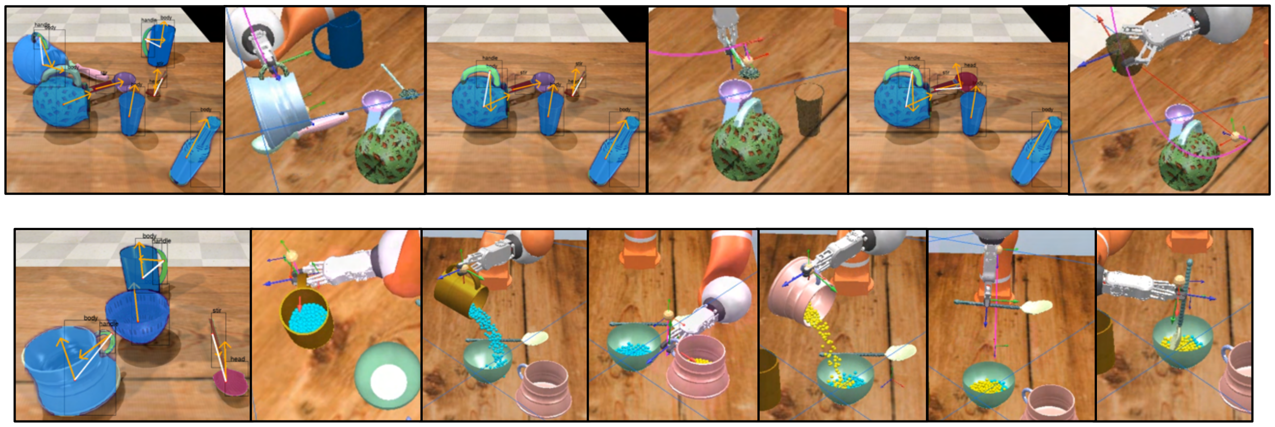


Figure 4.9: An example of ACF estimation based drink serving pipeline. From left to right, we show the ACF estimation result (left most) and snapshots of grasping two containers and pouring liquid (in cyan and yellow), and grasping a spoon to stir the liquid inside the container.

As shown in Figure 4.8, 4.9, random object clutters including containers and spoons are generated on tabletop, and a KUKA arm equipped with a parallel gripper is performing actions step by step to finish the task. Two RGB-D cameras are set on the two sides of the table and provide input to the ACF perception system, which estimates ACF keypoints and axes. The manipulator IK and motion planning (SBL [162]) algorithms are provided by CoppeliaSim and OMPL library.

Table 4.3: Task evaluation on grasp, pour, stir and drink serving which includes two pour and a stir action in sequence.

Task	**# Trials**	**Success Rate**
Grasp Mug	340	62.65%
Grasp Bottle	162	70.99%
Grasp Spoon	152	63.81%
Pour	130	78.46%
Stir	133	71.43%
Drink serve	143	37.06%

We evaluate the accuracy of ACF estimation with respect to the task success rate of three individual actions: grasping, pouring liquid (simulated as a set of particles) from one container into another, and stirring liquid in a container. We also evaluate the overall drink serving task, which includes pouring two kinds of liquid into the same container and stir the liquid afterwards.The grasping is considered successful when the gripper is able to grasp the object and lift it up without dropping. For evaluations on pouring actions, we calculate the ratio of the change in liquid volume of pourer and receiver, and consider it successful if the ratio is greater than 0.7. For stirring, it is considered successful when the position error is less than 2cm. The overall drink serving task is considered successful if all single actions succeed. Number of trials and success rates for all tasks are listed in Table 4.3. We find mugs and spoons are more challenging to grasp than bottles, because spoons have larger position errors in ACF estimation and mug handles have very small areas for valid grasps as shown in Figure 4.7. We find similar success rate for pour and stir actions, while pour often fails during the pouring action and stir often fails during motion. The main failure of the pour task is that much of water is poured outside of container because the pourer doesn't reach close to above the center of receiver. On the other hand, the stir task requires the robot to insert the spoon into the container from above, which will lead to collision with container inner surface. For drink serving, many trials fail on one of two pour actions which leads to a lower success rate. We expect taking masked object part point cloud into consideration could help design more adaptive trajectories.

We further evaluated the efficacy of ACF-based manipulation on a real Fetch robot. We realized the network purely trained from synthetic data could not generalize well to real objects, so we fine-tuned the network with around 10K images collected and labelled through ProgressLabeller [29]. We tested pour and stir tasks and got both 60% success rate out of 10 trials over novel instances. In failed trials, the errors were mostly due to the MoveIt! motion planner. Figure 4.10 shows snapshot of one grasp-pour and one grasp-stir task. The entire tasks recording will be attached in the supplementary video.

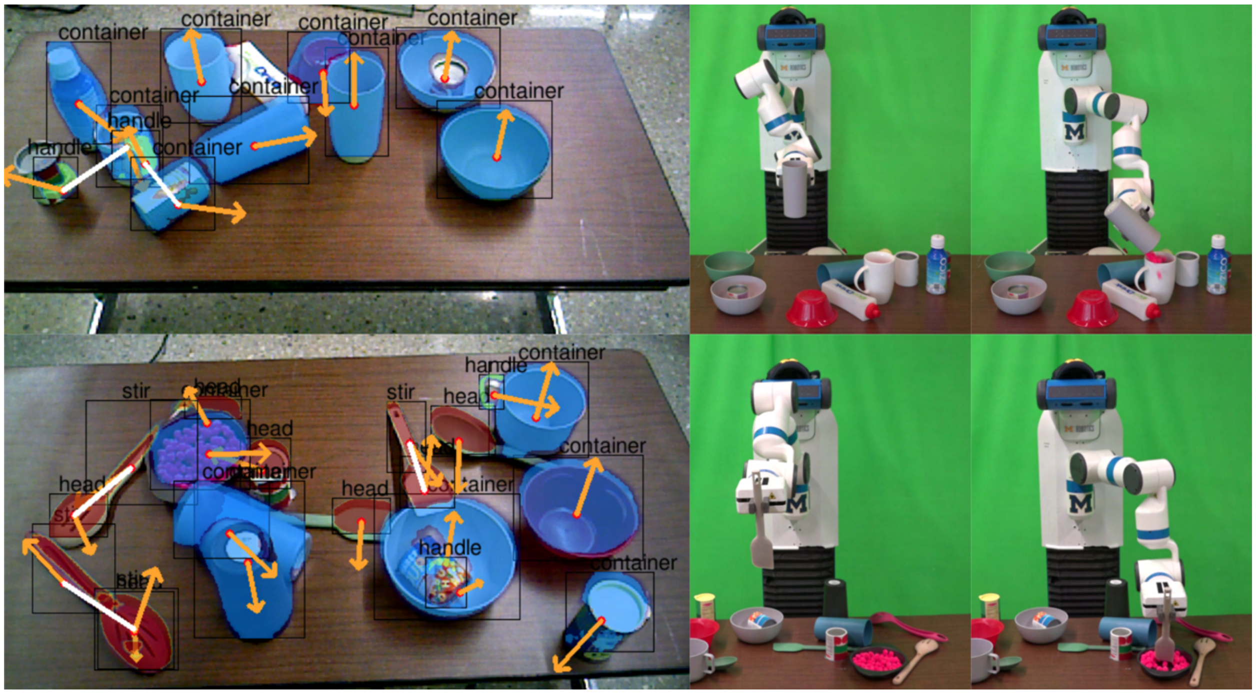


Figure 4.10: The mask, ACF keypoint and axis estimations are overlaid on the left images. On top-right, Fetch robot is grasping cups, pouring to another container based on ACF estimation in top-left image. On bottom-right, Fetch robot is picking up a spoon to stir inside a bowl based on bottom-left image.

4.5 Summary

We present a novel representation of object affordances called Affordance Coordinate Frame (ACF), and we propose a deep learning based perception method for estimating ACF given a RGB-D image. We demonstrate that our proposed perception method outperforms state-of-the-art method (NOCS [188]) in estimating ACF of novel objects. Our method also outperforms Mask R-CNN in detecting novel objects especially under cluttered environment. We further demonstrate the applicability of ACF for a drink serving task. In future, we will extend the work to incorporate a larger variety of object categories and compare with other analytical geometry-based manipulation systems.

CHAPTER 5

ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception

5.1 Motivation

Visual perception tasks often require vast amounts of labelled data due to their use of deep neural networks. Such deep neural networks have outperformed traditional methods in object pose estimation [100, 73, 78] when trained on public large-scale datasets [192, 77, 75]. However, considering the practice of deploying such systems in real-world robotics applications, such as semantic grasping and manipulation, current pose estimation systems can prove the difficulty of adaptation to different objects and settings without retraining with a customized large-scale dataset.

In particular, our need for training data is a result of object labels for pose estimation being defined to specific 3D object models (both geometry shape and texture). Learned models cannot be fine-tuned to transfer to similar object instances without additional training data. Recent work has made advances in category-level or unseen pose estimation [104, 145]. However, the objects included only cover a small set and there is no evidence showing the estimated pose is reliable enough for robotic manipulation. Further, the estimation results of deep neural networks are often vulnerable to environmental changes [27], including different lighting conditions, occlusions and object's special appearance like transparent or reflective surfaces. Synthetic data generation with domain randomization and photo-realistic rendering [181, 41] could improve generalizability, but it is still challenging to simulate real-world lighting as well as the noise inherent in the sensor

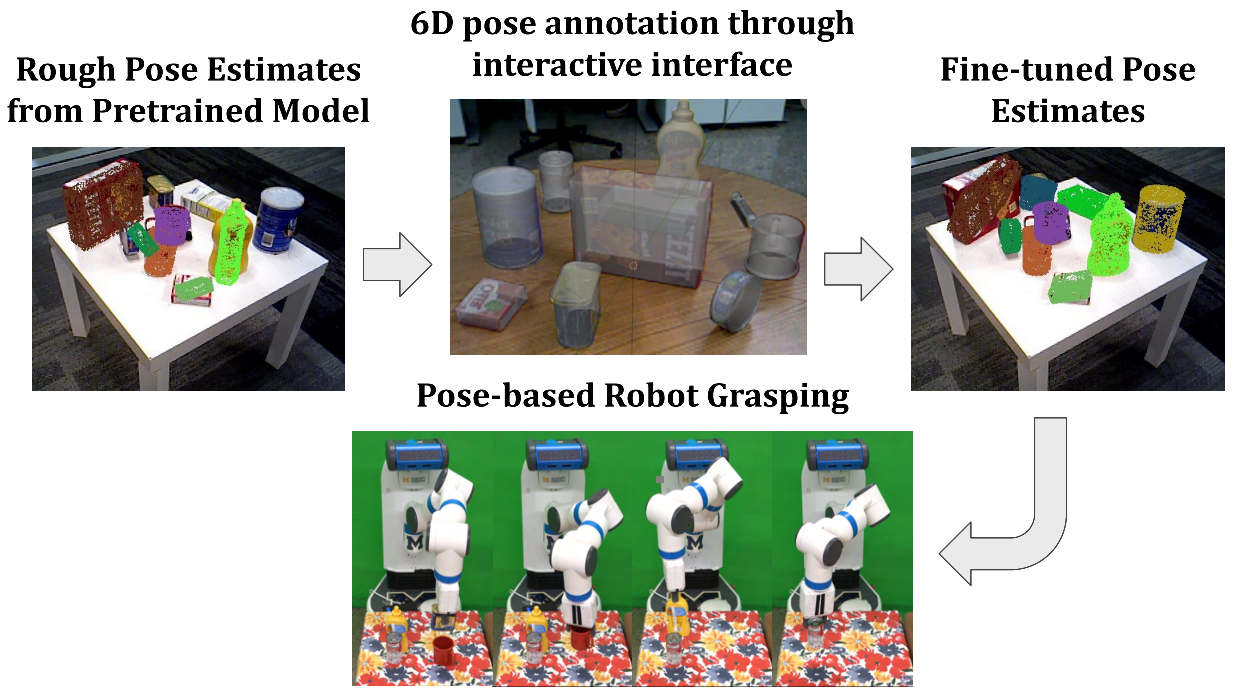

6D pose annotation through interactive interface
Rough Pose Estimates from Pretrained Model
Fine-tuned Pose Estimates
Pose-based Robot Grasping

Figure 5.1: The ProgressLabeller offers an interactive GUI for aligning all kinds of objects in the 3D scene to generate large-scale datasets with ground truth pose labels. The left image shows the rough 6D pose estimates from one state-of-the-art RGB-D deep models trained on public YCB dataset, and the right images shows fine-tuned pose estimates from the same model after retraining using generated data from ProgressLabeller. The pose estimates are then used for robotic grasping experiments.

modality. We show in the experiment that the network trained using real data is still over-performing synthetic data.

To address the problem of adaptation for deep pose estimation systems and their application to robotic manipulation, we propose **ProgressLabeller** (illustrated in Figure 5.1) as a method and implementation for creating large customized datasets more efficiently. Inspired by LabelFusion [126], ProgressLabeller collects training data of objects *in situ* in a mixed-initiative manner, similar in spirit to work by Gouravajhala et al. [61]. It takes visual streams of color images that observe objects in a physical environment as input. Objects in this stream only need to be labelled once by a human user through visual annotation. ProgressLabeller builds on recent advances in Structure-from-Motion [163] and visual SLAM [135] to produce both a 3D reconstruction and camera pose along the trajectory of the collected visual stream, where the annotated object labels can be propagated to all frames.

Compared to depth-based fusion methods, the color feature-based pipeline of ProgressLabeller suffers less noisy or invalid readings than that from depth sensing. Further, the use of color by ProgressLabeller allows it to include objects that are transparent and reflective [111] into the pose estimation process, as long as there exists textures from other objects or background. From an interface perspective, our implementation of ProgressLabeller aims to provide a more interactive design geared for users performing labeling tasks. This interface design enables seamless labelled pose validation in different views by checking and correcting the discrepancy between the masks of re-projected object models and original RGB images (an example is shown in Figure 5.2). The intuition is that, if then the labelled pose is close to ground truth, the re-projected object mask should align with the object's true area in RGB images from multiple views.

In this paper, we introduce ProgressLabeller as a semi-automatic approach to object 6D pose labelling on RGB(D) image sequence/video frames that works for transparent objects. Our aim is to release ProgressLabeller as an open-source tool for more effective dataset generation for object and pose recognition by robots. We evaluate the labelling accuracy of ProgressLabeller against LabelFusion on data stream samples from 4 public datasets with respect to segmentation mask and object pose accuracy. With the proposed system, we created a dataset of YCB objects [15] (about 1.2M object instance labels) within 2 days of data collection and labelling. The dataset presents more challenges than the public YCB-Video dataset [192] with more occlusions and better coverage of camera view directions, and collected using three different RGB-D cameras to evaluate the generalization across sensors. We fine-tuned a state-of-the-art RGB-D deep pose estimator [73] on our dataset and observed a large improvement on pose estimation accuracy and robotic grasping success rate, compared with the pretrained model on public dataset, as well as on the same amount of data from image-synthesis or LabelFusion annotation.

5.2 Related Work

With ProgressLabeller, a user can scalably label new datasets with camera world pose, scene object poses and scene object segmentations. This process is enabled by fusing streaming RGB (or RGB-D) inputs into a single scene-wide representation, and then allowing a human user to input relevant 6-DoF information via 3D modelling interfaces (such as those provided by Blender [37]). This process demonstrates label stability even over long input video streams, and due to its functionality with direct RGB inputs, can label even difficult objects such as transparent cups. We discuss below methods related to ProgressLabeller.

5.2.1 Direct & Human-in-the-loop labelling

The creation of 2D segmentation data is analogous to the object detection, keypoint detection, or semantic segmentation tasks (depending on desired output labels). Tools such as LabelMe [157] required users to directly interact with the underlying data to be labelled. This manual process was improved by model-assisted approaches such as Deep Extreme Cut [124] which decreases the amount of user effort necessary to label images. Shared autonomy and mixed-initiative methods have also been used in this approach, in which the user provides coarse pose or other estimations which are fine-tuned via a model-informed approach [200].

5.2.2 End-to-End Labellers

Previous tools have been created to enable this style of learning process. LabelFusion [126] is perhaps the most commonly utilized example. LabelFusion utilizes streaming RGB-D inputs to create a dense reconstruction of the scene, which is then labelled semi-manually by aligning 3D object models to the 3D reconstruction. While this approach is typically robust, it relies on RGB-D input for reconstruction, and experiences difficulties under certain regimes. In particular, transparent objects cause problems for commonly employed depth sensor technologies, and long-running input streams typically result in large amounts of 'drift'.

Some methods have been introduced to eliminate the need for CAD models in the labelling process. Singh et al. [166] proposed a method which utilizes user labelled keypoints and bounding boxes to generate pose and segmentation labels. This frees the system from dependency on CAD models, but requires user interaction directly with the images. SALT [170] proposed utilizing GrabCut to generate 3D bounding boxes and image segmentation labels for relevant scenes. This allows removing the dependency on object masks while also allows the labelling of dynamic scenes such as human gait videos.

Other works sought to improve the labelling procedure itself. EasyLabel [171] allows for semi-automatic labelling of scenes via sequentially added objects. This process is scalable, and generates high quality labels. However, it requires tight physical control over the scene to be labelled, which is not always feasible to obtain. Objectron [6] utilized modern smartphone's AR capabilities combined with human-labelled 3D bounding boxes to scalably create a large scale dataset. This method however is susceptible to label drift during long-duration input videos. KeyPose [112] specifically sought to generate labelled datasets for transparent objects. This method utilized stereoscopic images taken from a robot armature in order to avoid the problems of typical depth cameras have with transparent objects.

5.3 ProgressLabeller

In ProgressLabeller, we provide an interactive GUI for users to label object poses for a large amount of data in several steps. We incorporate camera pose estimation systems that reconstructed a 3D scene for aligning objects with, and calculate pose transforms that can be use to propagate the labelled object pose to all image frames. We provide interface to switch views among every RGB images to enable verification of labelled object pose by checking the alignment of re-projected mask with RGB images in multiple views. In this work, we assume the 3D mesh models of objects are provided and the objects are static in the scene during data collection (in special cases like T-LESS dataset [77], the objects are put onto a turntable and moving altogether while the cameras are static, the algorithm can still work because no environment background is captured by the cameras).

5.3.1 Camera pose estimation

We incorporated ORB-SLAM3 [16], KinectFusion [139] and COLMAP [163, 164] to do RGB, depth or RGB-D camera pose estimation as well as reconstruction. ORB-SLAM3 is selected as the default method for its balanced speed and accuracy. COLMAP was tested and proved to be more accurate but the time cost of reconstruction over more than 1000 images is unaffordable.

5.3.2 Object alignment by multi-view silhouette matching

Different from methods that align object 3D models with reconstructed point clouds, our system created a multi-view graphical user interface that overlays the object model's projection onto the original RGB images, so that the object pose errors could be easily detected from areas with misalignment of object texture, silhouette and boundary, as shown in Figure 5.2. Compared to depth reconstructed based methods, the pose label accuracy is improved based on higher accuracy of RGB than depth sensing. Besides, the system can also be used to label scenes with unreliable depth from transparent objects and backgrounds (see Section 5.4.4).

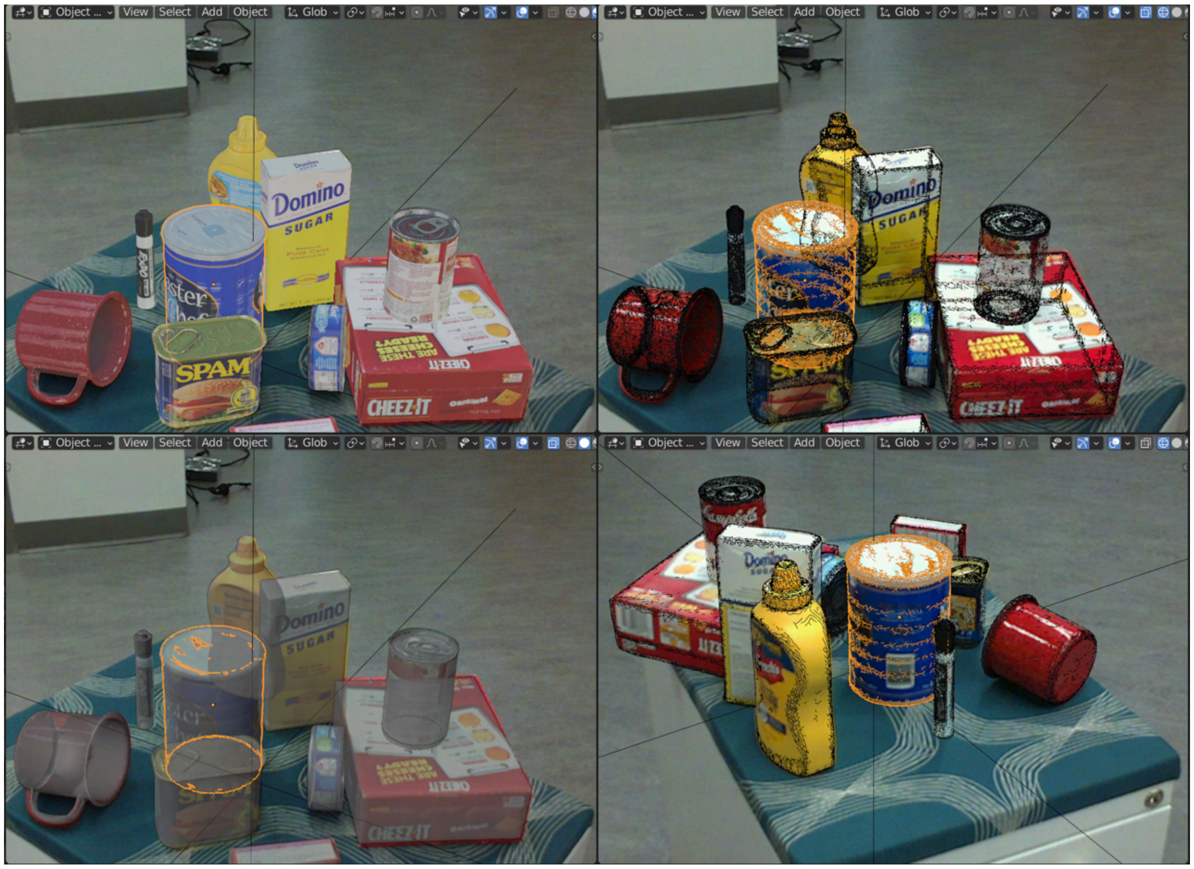


Figure 5.2: Capture of ProgressLabeller user interface for multi-view re-projection checking. The top-left view shows the aligned object models with rendered texture at labelled poses. The bottom-left view and top-right view show the silhouettes and boundaries respectively at the same camera view, and the bottom-right view shows the boundaries from another view for validation.

5.3.3 Semi-automatic labelling pipeline

We build ProgressLabeller as a plugin on Blender [37], which provides a good multiple view interface of overlaid RGB image, 3D reconstructed scene and objects for labelling and verifying poses. The overall procedure of labelling object poses includes several steps within the Blender graphical interface:

1. Import RGB(D) images and object 3D models into the 3D interactive workspace. Set parameters including camera intrinsics, camera pose estimation parameters and display settings, etc.

2. Do camera pose estimation, then the estimated camera poses and a reconstructed 3D point cloud will be added to the workspace. Depth images are used to solve the scale of reconstruction.

3. (optional) When depth input is available, another point cloud fusing depth input based on estimated camera poses can be generated to provide a denser view of the entire scene and help find objects' rough positions.

4. (optional) Do plane alignment for a better viewpoint and ease of correcting object pose

5. Align the object so that its re-projection matches the ground truth area in multiple views of RGB images

6. Export labelled object poses and render segmentation masks, bounding box labels, etc.

In step 2, when there is only RGB image available, the scene scale (3D point cloud and camera's trajectory position) is unknown, to solve this scale problem, we take advantage of the known object sizes by dragging them to align with the 3D reconstruction and verify that in multiple different views. In step 3, we implemented a depth fusion module based on the estimated camera poses from RGB reconstruction, based on the Signed Distance Functions as in KinectFusion [139]. The fused point cloud could give a rough reference of object locations. In step 4, Iterative Closest Point (ICP) is used for aligning RGB feature or depth point cloud to X-Y plane in Blender. In step 5, the object pose is labelled with visually ensured accuracy, which is the essential design that enables labelling of objects with only RGB images based on multi-view geometry. By re-projecting object's 3D model and check whether it aligns perfectly with the object's true textures or silhouettes in the RGB images from multiple views, users can fine-tune the object's pose and verify the error seamlessly until the object matches the images.

The entire labelling pipeline typically takes around 30 minutes for one data stream about 5K images. Data import and export takes about 20-30% of time, with a rendering speed around 4-20 images per second depending on number of objects in the scene. The rendering could be done in parallel on a GPU if a vast amount of data is required. ORB-SLAM3 camera pose estimation takes about 10-20% of time. Manual labelling and verification take the rest of time.

5.3.4 Annotation accuracy estimation from simulation

We verify the accuracy of annotations throughout this multi-view silhouette matching process by simulating an iterative object pose update process. In each iteration, given a certain camera frame, we assume the object will be translated in a plane parallel with its x-y plane, or rotated about z-axis (for better control), towards a pose that maximizes the Intersection-over-Union (IoU) between the rendered silhouette at current pose and ground truth.

5.3.4.1 Problem Definition

Given set of N images $I^{\{i\}}$ with their corresponding camera pose $T^{\{i\}}$ in the world frame, $i \in \{1, 2, 3, \dots, N\}$. Our goal is to find ground truth object pose $T_{gt}^{\text{obj}\{j\}}$ in the world frame for all the objects $j \in \{1, 2, 3, \dots, M\}$ in the scene. We define the projection operator as $S^{\{i,j\}} = \text{Proj}(T^{\{i\}}, T^{\text{obj}\{j\}})$, which render object j given its CAD model, camera pose $T^{\{i\}}$ and object pose $T^{\text{obj}\{j\}}$ into an object texture/silouette $S^{\{i,j\}}$. Also defined the IoU operator as $\text{IoU}_{\text{obj}\{j\}}(I^{\{i\}}, \text{Proj}(T^{\{i\}}, T^{\text{obj}\{j\}}))$ to calculate the IoU for pixels in object j between real image $I^{\{i\}}$ and synthetic texture/silouette $S^{\{i,j\}}$.

The multi-view texture/silhouette matching iterative update is proceeded with a goal to maximize the IoU. Given the pose for object j in the kth iteration $T^{\text{obj}\{j\}}_{(k)}$, in $(k+1)$th iteration:

$$T^{\text{obj}\{j\}}_{(k+1)} = \text{argmax}_{f[T^{\text{obj}\{j\}}_{(k)}]} \text{IoU}_{\text{obj}\{j\}}(I^{\{i\}}, \text{Proj}(T^{\{i\}}, f[T^{\text{obj}\{j\}}_{(k)}])) \tag{5.1}$$

where $f[T^{\text{obj}\{j\}}_{(k)}]$ describes all possible translation start from the $T^{\text{obj}\{j\}}_{(k)}$ that is within the plane p or the rotation along the axis ω as shown in Figure 5.3. So:

$$f[T^{\text{obj}\{j\}}_{(k)}] = \exp^{\widehat{\xi}_1 \theta_1} \exp^{\widehat{\xi}_2 \theta_2} T^{\text{obj}\{j\}}_{(k)} \tag{5.2}$$

where $\xi_1 = \begin{bmatrix} -\omega \times v_o \\ \omega \end{bmatrix}$, $\xi_2 = \begin{bmatrix} v \\ 0 \end{bmatrix}$ are the twist coordinates for twist $\widehat{\xi_1}, \widehat{\xi_2}$.

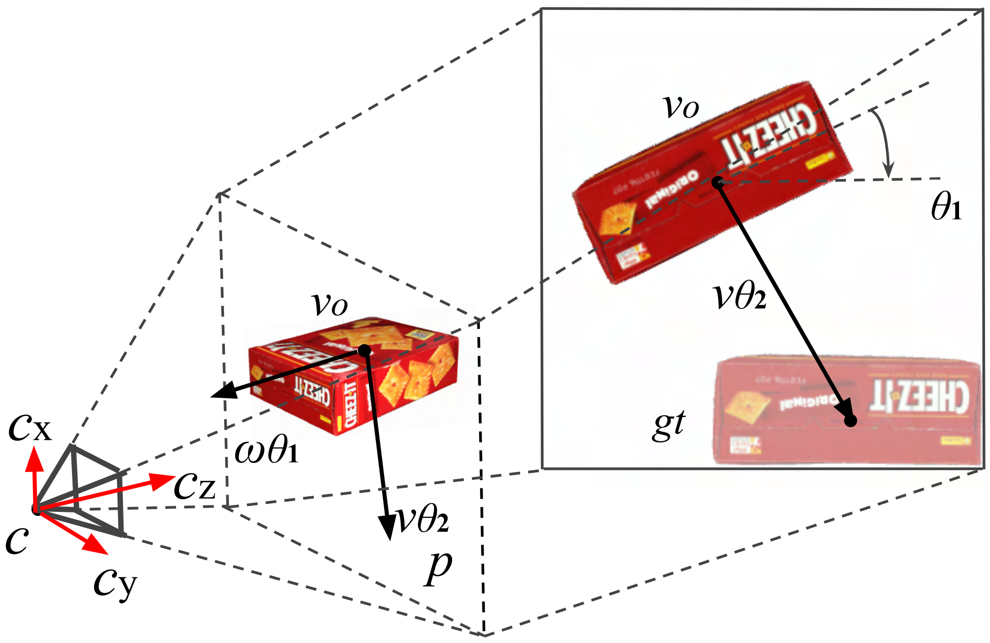


Figure 5.3: Diagram for an object shown under a camera. c denote the location of camera and c_x, c_y, c_z are its x, y, z axis. p is a plane parallel to camera plane and passing through object's center v_o. ω the rotation direction parallel to c_x and passing through v_o. θ_1 is the magnitude of rotation radius. v is the translation direction within the plane p and θ_2 is the translation magnitude. On the right hand side is the projection image, the object in the transparent color is the object with ground truth pose.

5.3.4.2 Simulation results

We generate a CAD model sets with 44 different CAD models. For each run, we generate T_{gt}^{obj} with a random rotation matrix and location at the origin. 40 cameras are created with their z axis pointing towards the origin and a random location at a sphere around the object. The initial pose T_0^{obj} is generated by adding a random position noise from Gaussian distribution with variance of 10cm to origin and with a random 3D orientation. During each iteration, v, θ_1, θ_2 in Equation 5.2 are discretized for simulation. The result shows that it takes around 10.36 iterations for the algorithm to converge within 1mm location error and dot product larger than 0.99 between ground truth and converged rotation axes.

5.4 Experiments

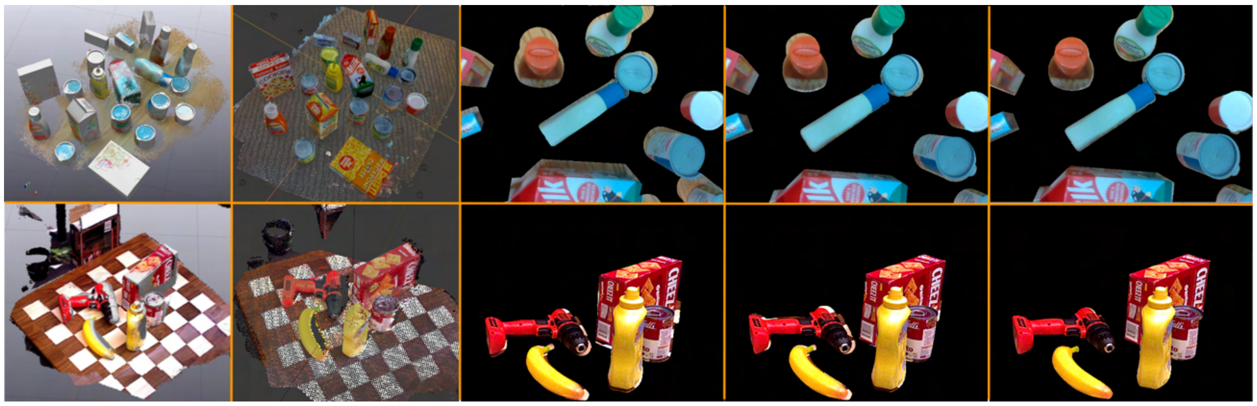


Figure 5.4: From left to right, we show the 3D labeller view of LabelFusion, ProgressLabeller and cropped RGB image with masks from LabelFusion, ground truth and ProgressLabeller. The top and bottom row show a sample from HOPE and YCB dataset respectively. We find even ground truth pose labels have easily observable errors, shown as the spacing between black masks and object boundaries.

In this section, we report results on several evaluation experiments on the data generated using ProgressLabeller. The output label quality is firstly evaluated on public datasets against ground truth. Then we introduce a large scale dataset with object pose annotations, and report the evaluation results of fine-tuning a state-of-the-art deep pose estimation on accuracy. The improved accuracy is further evaluated in robotic grasping experiment. Finally, we show two potential applications:

labelling a complex scene with transparent objects in front of reflective backgrounds, and training a neural rendering model on a single object from collected data and do image synthesis.

Dataset	Method	Mask IoU ↑	Pos (cm) ↓	Rot (quat) ↓	ADD (AUC-0.1d) ↑	ADD-S (AUC-0.1d) ↑	Feature (Pixel) ↓	Time (min)
HOPE	LabelFusion*	0.6729	1.5825	0.2561	0.7095	0.7101	4.1248	31
	ProgressLabeller	0.8600	1.0948	0.0801	0.9566	0.9566	1.9479	48
	Ground Truth	-	-	-	-	-	2.2752	-
YCB-Video	LabelFusion	0.9089	0.5165	0.0741	0.9977	0.9989	4.7365	29
	ProgressLabeller	0.8849	1.7537	0.0636	0.7642	0.7662	3.08	16
	Ground Truth	-	-	-	-	-	3.3868	-
NOCS*	LabelFusion	0.7684	2.2825	2.9092	0.6436	0.6438	-	12
	ProgressLabeller	0.8785	1.4949	2.6153	0.8674	0.8679	-	15
T-LESS*	LabelFusion*	-	-	-	-	-	-	-
	ProgressLabeller	0.8810	0.6779	0.0856	0.6626	0.6642	-	8

Table 5.1: Results on feature matching distance between rendered object RGB image and original image. 'Pos (cm)', 'Rot (quat)' refers to positional error in centimeters and rotational error calculated as norm of difference in quaternions as in [80]. LabelFusion cannot work on T-LESS data streams at all, and it cannot reconstruct one of HOPE samples scene entirely, so we trim the sequence to 1/4 length. Object models in T-LESS and NOCS datasets do not have enough features to measure the feasure-based pixel distances.

5.4.1 Evaluation on label generation

We evaluate the label quality with respect to pose accuracy against ground truth and time cost on ProgressLabeller versus LabelFusion [126]. Specifically, we use both tools to label on 64 object instances from 8 sample sequences among the 4 public pose estimation datasets including YCB-Video [192], T-LESS [77], NOCS [104], and HOPE [183].

Evaluation metrics. We use mask Intersection-over-Union (IoU) between rendered mask of object at labelled pose and at ground truth to evaluate the segmentation, and select 4 distance metrics to compare a labelled pose with ground truth pose, including 3D positional error E_p, 3D rotational error E_r, average pairwise distance (ADD) E_{ADD} [75] and average pairwise distance in symmetric case (ADD-S) E_{ADDS} [192], defined as follows:

$$E_p = \|t - t^*\|, E_r = \min\{\|q - q^*\|, \|q + q^*\|\} \tag{5.3}$$

$$E_{ADD} = \frac{1}{|M|} \sum_{x \in M} \|(Rx + t) - (R^*x + t^*)\| \tag{5.4}$$

$$E_{ADDS} = \frac{1}{|M|} \sum_{x_1 \in M} \min_{x_2 \in M} \|(Rx_1 + t) - (R^*x_2 + t^*)\| \tag{5.5}$$

where $M = \{x_i \in \mathbb{R}^3\}$ is the object 3D point set, and R, q, t, R^*, q^*, t^* refers to the 3D rotation matrix, rotation represented in quaternion, 3D position vector of labelled pose and ground truth respectively. Specifically for ADD and ADD-S, we report the value of area under the accuracy-threshold curve obtained by varying the distance threshold from 0 to 0.1 diameter of objects (AUC-0.1d) following metrics among pose estimation papers.

Besides, from the observation that even ground truth labels are still not perfectly correct (re-projected object masks align with true area in RGB images from multi-view), we propose an RGB feature based evaluation on object pose without ground truth. Specifically, we render the object at labelled pose to get an image with only an object in front of black background, and extract SIFT features from both the rendered image and original RGB image, then calculate the average of feature matching distance in pixels. In practice, we assume the labelled pose is close to ground truth, and remove incorrect matching with a distance threshold of 10 pixels. Figure 5.4 shows qualitative examples of labelled poses, where we observe more accurate re-projection results from ProgressLabeller than LabelFusion output as well as provided ground truth on masked RGB images. We believe the main error cause in LabelFusion is the accumulated depth sensing error during reconstruction. Another drawback of LabelFusion UI design is that, there is no measurement or display of object poses in the 3D scene, so the users need to switch back-and-forth between the 3D scene and the final generated images that has image masks rendered at labelled poses. Also its 3D model as well as rendering result doesn't display textures, so it's hard to deal with symmetric-shape objects in HOPE and NOCS.

The quantitative results are shown in Table 5.1. From the comparison, we see both labellers took similar amount of time to annotate data streams, while ProgressLabeller is more accurate and robust in most streams. LabelFusion has higher accuracy on YCB-Video dataset, which has a large feature-based pixel distance. For example, we found the pose annotation of object 006_mustard_bottle was flipped 180° in one scene.

5.4.2 Multi-camera dataset creation

From analysis of absolute accuracy result above and to evaluate robustness on existing deep pose estimation networks, we aim to create a more accurate, multi-camera, cluttered dataset on YCB objects [15]. We mounted 3 RGB-D cameras with different sensing technologies, ASUS Xtion Pro Primesense Carmine 1.09 (structured light), Intel RealSense D435i (stereo) and RealSense L515 (LiDAR), on a Fetch robot and collected data streams at ∼15Hz when Fetch is slowly driving around the tabletop object scene with a constant speed. Each scene contains almost a bit more than full round, with 1K∼3K paired RGB-D images from each camera. In this way, the collected data is free of motion blur and covers full round view of objects and occlusions. The dataset includes 16 scenes with 10 objects placed in both isolation and dense clutter, and another 4 scenes each with a single object for unit test. Figure 5.5 gives an example. It took 2 days collected and labelled the dataset, with about 120K images and 1.2M labelled object instances.

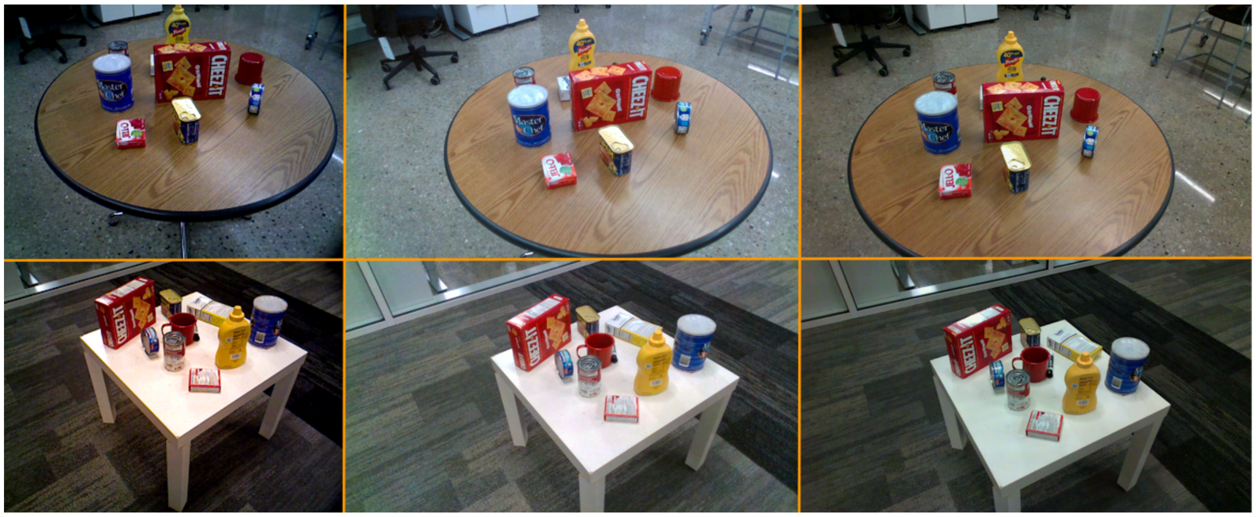


Figure 5.5: The top and bottom row shows RGB images in training and testing set respectively. From left to right, the images are taken from Primesense, RealSense D435i and L515.

Train set/Test set	Primesense		RealSense L515		RealSense D435i	
Metric: AUC (0-10cm)	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
Pretrained on Asus Xtion Pro Live	21.63	41.16	47.43	72.90	44.67	68.00
Fine-tuned on Primesense	**60.02**	**79.21**	55.32	77.85	49.75	73.35
Fine-tuned on RealSense L515	33.45	58.16	**68.84**	**83.69**	57.67	76.26
Fine-tuned on RealSense D435i	26.39	50.38	63.82	82.52	**64.64**	**82.36**

Table 5.2: The pose estimation accuracy of FFB6D cross-validated on datasets collected using three cameras.

5.4.3 6D object pose estimation Fine-tuning

Using the collected dataset, we fine-tuned one of the state-of-the-art deep pose estimation neural networks taking RGB-Depth images as input, FFB6D [73] over its pretrained model on YCB-Video dataset. In particular, we created 3 training sets by downsampling to 1/10 the training set collected by 3 cameras respectively and got 3 fine-tuned models by training on a RTX 3080 with batch size 6 for 20 epochs and default settings in other parameters. Then, we cross-validated the 3 fine-tuned models, along with pretrained model, on 3 test sets taken from 3 cameras.

5.4.3.1 Evaluation on pose accuracy across different cameras

We report the AUC for ADD and ADD-S metrics between 0 and 10cm, to match the original result in [73]. The result is shown in Table 5.2, where we find the fine-tuned dataset recognizes the sensor modality and noise difference between 3 cameras as the test set ADD and ADD-S are mostly the highest on the same training set.

5.4.3.2 Evaluation against LabelFusion and synthetic data on pose accuracy and robotic grasping

We used 3 datasets to train FFB6D with the same setting as above. The first is the collected Primesense subset labelled by ProgressLabeller, the second is the same data labelled by LabelFusion, and the third is synthesized images by rendering YCB objects over both raw RGB and depth images according to the labelled poses using Blenderproc [41], which mostly maintain the same object

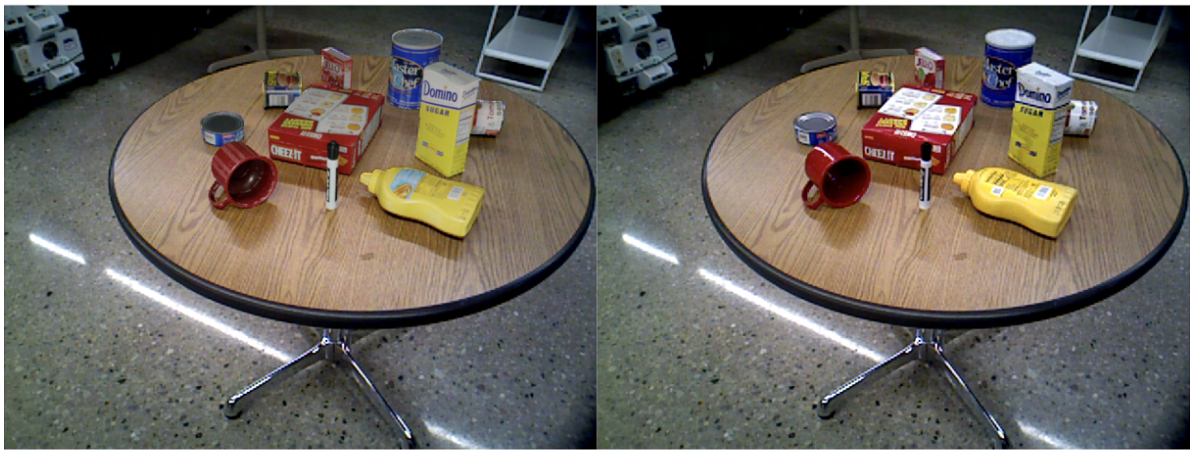


Figure 5.6: A synthesized RGB image using Blenderproc is shown on the left. The original image is shown on the right. The YCB objects are rendered and overlaid to original image according to labelled poses. Depth synthesized images are generated in the same way.

pose distribution and sensor noise as real data, as shown in Figure 5.6.

We tested pose estimation accuracy from models trained on the above 3 datasets, with respect to ADD-AUC and pose-based grasping success rate. For pose-based grasping, we manually defined grasp poses along the symmetrical axes on the object 3D models, and grouped them by the grasping tolerance, defined by the distance between two gripper fingers (10.39 cm for Fetch) minus the object's diameter along the grasping direction. Obviously, a smaller grasping tolerance requires more accurate pose estimates for a successful grasp. For example, 002_master_chef_can is challenging to grasp as its tolerance is only 0.36 cm. We repeated grasping on every tolerance for 5 times based on the pose estimates, and the success/failure statistics is shown in Table 5.3. Overall, the fine-tuning over ProgressLabeller data improves the grasping success rate the most. Among experiments, we found in some cases the grasp is still successful given large rotational error, where the object was aligned to another pose during grasping, such as 002_master_chef_can and 006_mustard_bottle when grasping from the side. Also, sometimes the objects were grasped along another direction, such as 009_gelatin_box and 010_potted_meat_can, in this case, their actual grasp tolerance might change. We expect evaluation on object placement accuracy to reveal these errors. Otherwise, the grasping test results generally matches the pose accuracy presented in ADD. Figure 5.7 shows an example of grasping experiment.

Network model		Pretrained		Blenderproc		LabelFusion		ProgressLabeller	
Object name	Grasp tolerance (cm)	ADD	Grasp	ADD	Grasp	ADD	Grasp	ADD	Grasp
002_master_chef_can	0.36	25.16	0/5	48.67	3/5	44.78	4/5	**55.42**	**5/5**
003_cracker_box	3.90	19.57	1/5	50.76	1/5	60.8	4/5	**71.01**	**5/5**
005_tomato_soup_can	3.65	27.74	1/5	51.12	2/5	65.48	4/5	**66.79**	**5/5**
006_mustard_bottle	6.33 2.50	31.02	1/5 0/5	58.87	**5/5** **5/5**	70.77	**5/5** 4/5	**73.47**	**5/5** **5/5**
007_tuna_fish_can	7.09 2.04	15.14	2/5 2/5	26.41	3/5 4/5	30.02	**5/5** 4/5	**39.87**	**5/5** **5/5**
009_gelatin_box	7.60 3.14 1.62	13.25	2/5 2/5 2/5	32.79	4/5 3/5 4/5	51.45	**5/5** **5/5** **5/5**	**55.24**	**5/5** **5/5** 4/5
010_potted_meat_can	4.75 2.08 0.78	19.68	5/5 1/5 2/5	39.56	3/5 **4/5** 0/5	47.48	4/5 **4/5** **5/5**	**50.64**	**5/5** **4/5** **5/5**
025_mug	1.24	19.10	1/5	60.70	3/5	49.31	1/5	**66.53**	**4/5**
040_large_marker	8.67	20.10	0/5	44.75	2/5	**55.89**	**3/5**	53.73	**3/5**
overall	-	-	22/75	-	46/75	-	62/75	-	**70/75**

Table 5.3: Grasp record comparison on part of YCB objects using grasp poses generated based on pose estimates from pretrained and fine-tuned FFB6D on the collected dataset, with Blenderproc data synthesis, LabelFusion annotations and ProgressLabeller annotations.

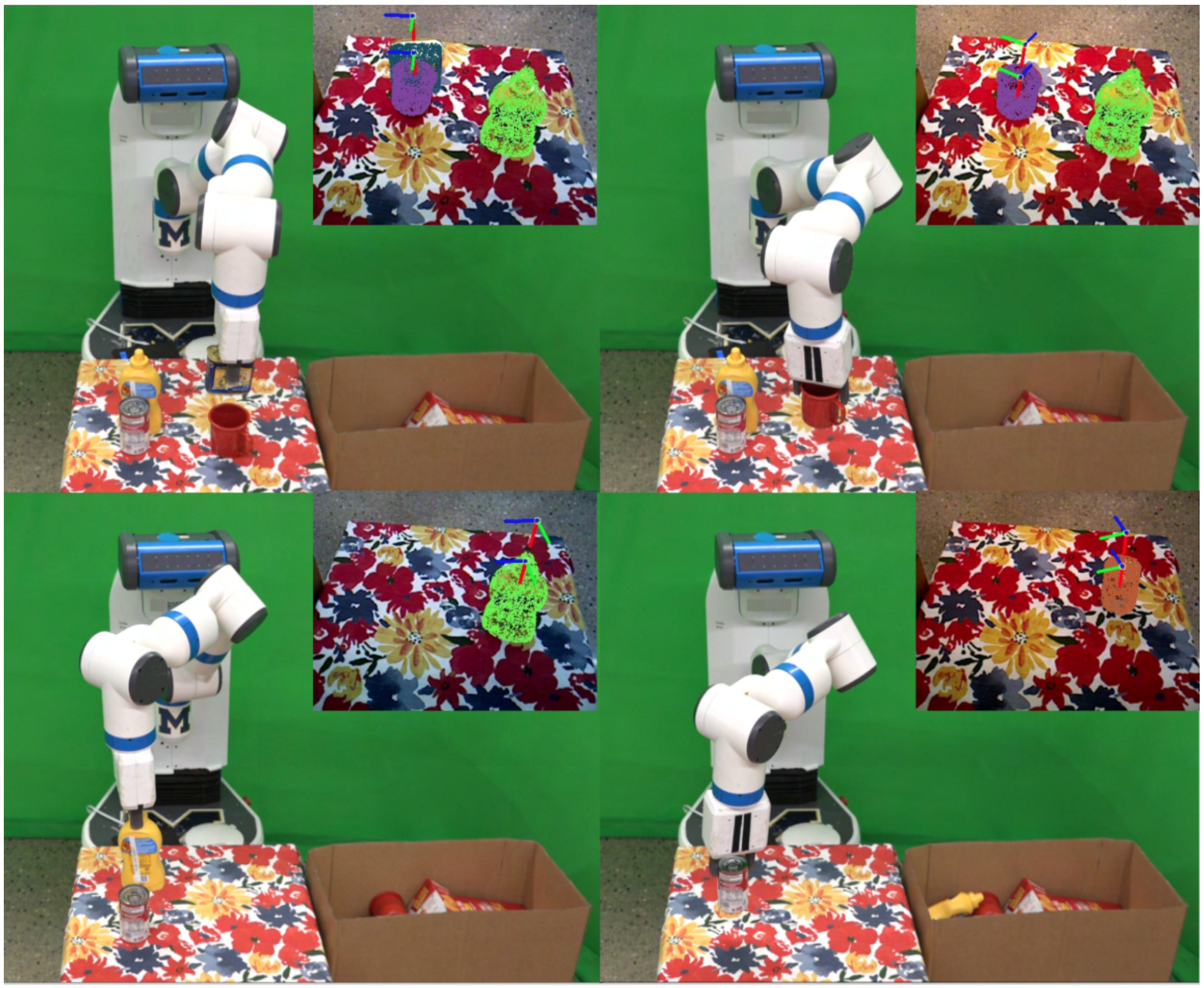


Figure 5.7: The Fetch robot is grasping objects on tabletop based on estimated poses. The pose estimates are shown as projected point clouds and coordinate frames in top-right smaller images. The higher and lower frames correspond to pre-grasp and grasp poses.

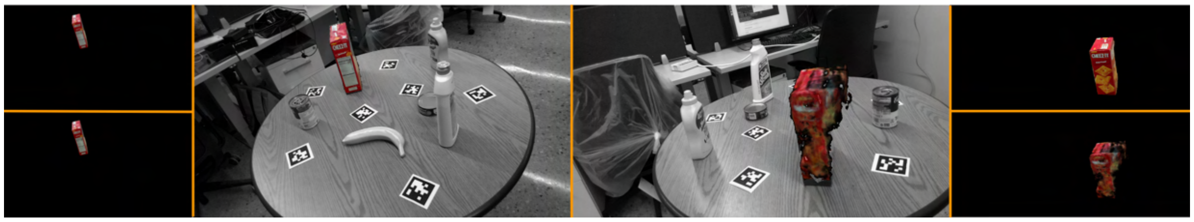


Figure 5.8: Comparison of best (left) and worst (right) validation pose renderings from the NSVF model. The smaller images with black background on top corners are the ground truth masks, and those on bottom corners are rendered output from NSVF, and the middle larger images are the renderings imposed onto the original captured image in greyscale. The best image (left) had an average pixelwise L2 error of 0.0067, and the worst (right) had an average pixelwise L2 error of 0.1867.

5.4.4 Other applications

5.4.4.1 Transparent Dataset Labelling

We collected RGB-D videos with transparent cups and labelled their poses using ProgressLabeller. Figure 5.9 shows in detail that even there is no 3D points around the transparent area when we fused raw depth according to the estimated camera poses, the tool enables accurate pose labelling by matching the object's mask with RGB images. We believe a large-scale 3D dataset of transparent objects can be efficiently created using ProgressLabeller.

5.4.4.2 Training neural radiance fields on objects

As a brief additional experiment, we provide prima facie data to show the effectiveness of our labeller in generating datasets for training a Neural Sparse Voxel Fields (NSVF) model [108]. This model permits rendering objects at novel poses given multiple input views with corresponding camera poses. We use ProgressLabeller to label a scene containing a 003_cracker_box, with 3509 images in total. From this dataset, we uniformly random sample 100 images with object poses for train, test and validation set respectively. We remove the background from the training images using the segmentation from labeller, and an NSVF model was trained to 75k total iterations on a dual GPU machine consisting of an Nvidia RTX 3060 TI and RTX 3070. After training, renderings were made at each camera pose given in the train, test, and validation sets. These renderings were evaluated via a pixel averaged L2 error against the ground truth segmented images produced

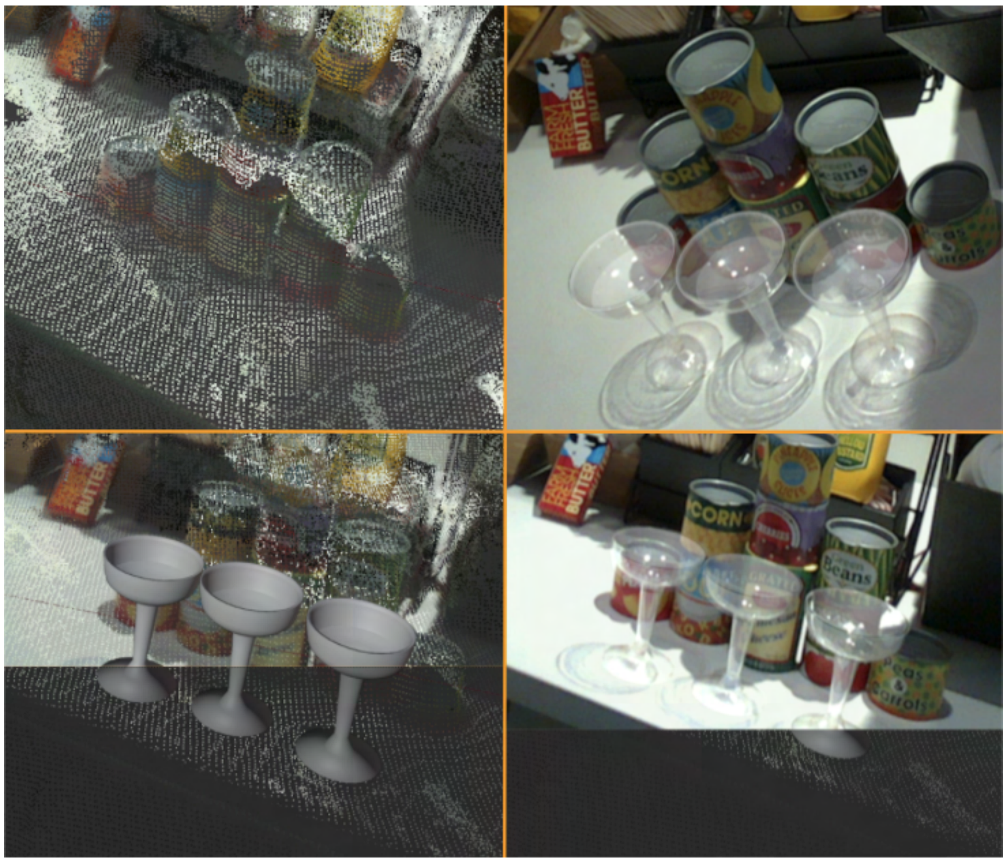


Figure 5.9: The labelling on transparent champagne cups. When there is no reliable depth (top-left), ProgressLabeller enables checking the pose validity by re-projection object models (bottom-left) to RGB images from different views (top-right and bottom-right).

by ProgressLabeller where each pixel channel was normalized to the range $[0, 1]$. Those results presented in Table 5.4. Visual comparisons between the best and worst validation set renderings along with their associated errors are presented in Figure 5.8.

Dataset	Train	Validation	Test
Per-pixel L2 error	0.0108	0.0460	0.0409

Table 5.4: Average pixelwise errors for the train, validation, and test sets from the NSVF model trained on ProgressLabeller outputs.

5.5 Summary

In this work, we have presented the design and application ProgressLabeller for object pose annotation. Through comparison to LabelFusion, we show that its higher accuracy with similar labelling time, efficiency of creating a large-scale customized dataset, and potential in fine-tuning pose estimation deep neural networks for robotic grasping. In future, we can improve ProgressLabeller by adding online object model generation and support for dynamic scenes.

CHAPTER 6

ClearPose: Large-scale Transparent Object Dataset and Benchmark

6.1 Motivation

Transparent and translucent objects are prevalent in daily life and household settings. When compared with opaque and Lambertian objects, they present additional challenges to visual perception systems. The first challenge is that transparent objects do not exhibit consistent RGB color features across varying scenes. Since the visual appearance of these objects depends on a given scene's background, lighting, and organization, their visual features can drastically differ between scenes thereby confounding feature-based perception systems. The second challenge is the inaccurate depth measurements among RGB-Depth (RGB-D) cameras on transparent or translucent materials due to the lack of reliable reflections. This challenge is especially meaningful for state-of-the-art pose estimation approaches that require accurate depth measurements as input. To overcome these challenges, computational perception algorithms have been proposed for a variety of visual tasks, including image segmentation, depth completion, and object pose estimation. In this paper, our aim is to complement recent work in transparent object perception by providing a large-scale, real-world RGB-D transparent object dataset. Furthermore, we use the new large-scale dataset to perform benchmark analysis of state-of-the-art perception algorithms on transparent object depth completion and pose estimation tasks.

There are several existing datasets focusing on transparent object perception with commodity

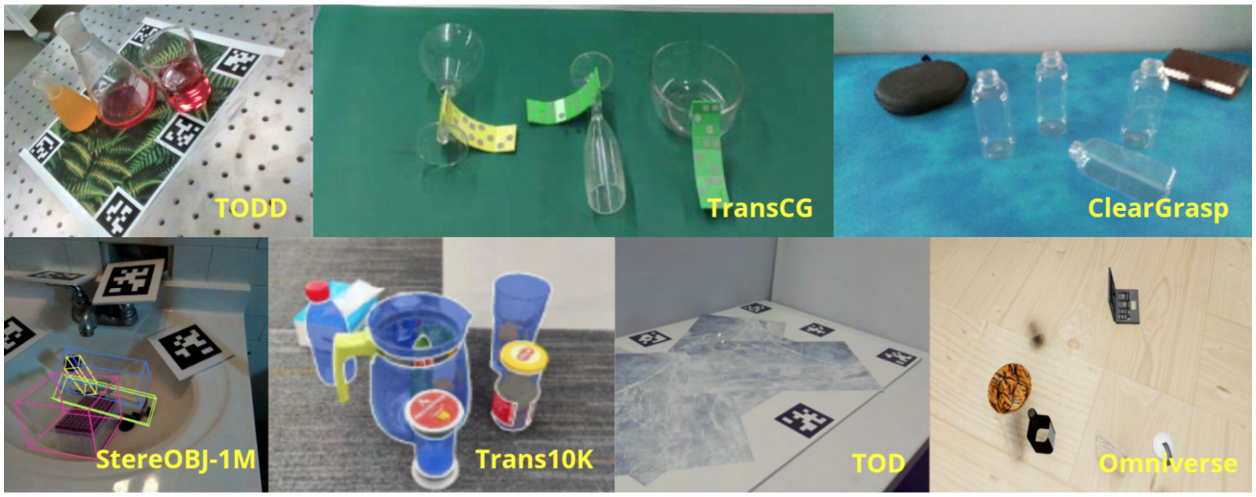


Figure 6.1: Sample images from existing transparent datasets. TOD and StereOBJ-1M are collected using stereo RGB cameras, and other datasets used RGB-D cameras (except for Omniverse which is completely synthetic).

Table 6.1: Comparison between existing transparent object datasets and ClearPose. *Trans10K is a transparent segmentation dataset and has no object-centric pose labels. *StereObj1M is not publicly available at the time of submission so the #frame and #pose annotation are estimated based on the published ratio of transparent objects in the entire object set [111]

dataset	modality	#obj	#frame	#pose annotation
TOD [112]	RGB-D	15	48K(real)	∼0.1M
ClearGrasp [161]	RGB-D	10	50K(syn)+286(real)	∼0.2M
TODD [196]	RGB-D	6	15K(real)	∼0.1M
Omniverse [217]	RGB-D	9	60K(syn)	∼0.2M
TransCG [53]	RGB-D	51	58K(real)	∼0.2M
Trans10K* [194]	RGB	10K	15K(real)	seg only
ProLIT [214]	Light-Field	5	75K(syn)+300(real)	∼0.1M
StereObj1M* [111]	Stereo	7	∼150K(real)	∼0.6M
ClearPose (ours)	RGB-D	63	**360K(real)**	**∼5M**

RGB stereo or RGB-D sensors, as summarized in Figure 6.1. While these datasets target transparent object perception, most are relatively small-scale (no more than 50K real-world frames), include few cluttered scenes (typically with less than 3 objects per image), do not offer diverse categories of commonplace household objects, and record limited lighting changes. These limitations motivate the introduction of **ClearPose**, a large-scale real-world transparent object dataset labeled with ground truth pose, depth, instance-level segmentation masks, surface normals, etc. that (1) has a comparable size with ordinary opaque object pose estimation datasets like YCB-Video [192]; (2)

contains challenging heavy clutter scenes including multiple layers of occlusion between transparent objects; (3) contains a variety of commonplace household object categories (bottle, cup, wine, container, bowl, plate, spoon, knife, fork, and some chemical lab supplies); (4) covers diverse lighting conditions and multiple adversarial testing scenes. Further details of ClearPose in relation to existing datasets is included in Table. 6.1 with sample images from ClearPose shown in Figure 6.2.

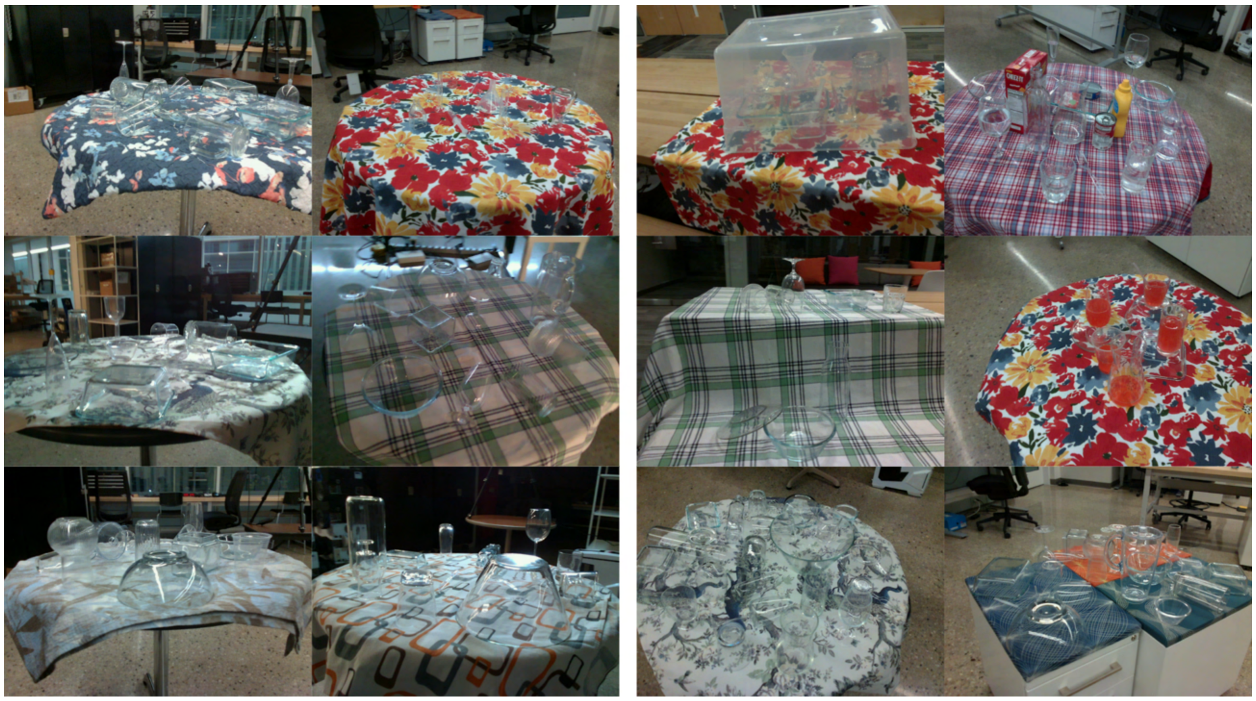


Figure 6.2: Sample images from ClearPose dataset. On the left, we show RGB images taken for different object subsets under various lighting conditions and backgrounds. On the right, we show examples of different types of testing scenes, such as covered by a translucent box (top-left), novel background with opaque distractor objects (top-right), non-planar cases (middle-left), filled with liquid (middle-right), and heavy clutters of transparent objects (bottom).

The labeling of such a large-scale dataset requires both efficiency and accuracy. To achieve these qualities in ClearPose, we take advantage of a recently published pipeline named Progress-Labeller [29]. The ProgressLabeller pipeline combines visual SLAM and an interactive graphical interface with multi-view silhouette matching-based object alignment to enable efficient and accurate labeling of the transparent object poses from RGB-D videos, which realizes rapid data annotation and exempts from the broken depth problem by transparent objects. Given the unique scale and relevance of ClearPose, we envision it will serve beneficial as a benchmark dataset on transparent perception tasks. In this current paper, we include benchmark analysis for a set of state-of-the-art visual perception algorithms on ClearPose. We target our benchmark analysis on the tasks of depth completion and RGB-D pose estimation.

6.2 Related Work

6.2.1 Transparent Dataset and Annotation

As mentioned in Table 6.1, there are several existing datasets and associated annotation pipelines that focus on transparent objects. With the exception of works such as Trans10k [194], TranSCut [199] and TOM-Net [22] that are focused on 2D image segmentation or matting, most recent transparent perception datasets are collected in an object-centric 3D setting using RGB-D, stereo or light-field sensor modalities.

Similar to the case of opaque objects, datasets created in simulation, supporting photo-realistic rendering with ray-tracing, are more readily created and can produce very realistic examples of transparent object appearance. Examples of such simulated datasets include those [161] rendered by Blender, [214] by Unreal Engine and [217] by Nvidia Omniverse platform. While simulated datasets are appealing for their ease of creation, they often lack realistic feature artifacts (e.g. sensor noise, object wear marks, true lighting etc.) which can impact downstream perception systems that rely on the synthetic dataset (i.e. the syn-to-real gap).

Among existing real-world datasets, TOD [112] and StereObj1M [111] use RGB stereo cameras together with AprilTags. First, camera pose transforms are solved from AprilTag detection, and then several 2D keypoints on the objects are manually annotated in keyframes that are farthest to each other in the sequence. The corresponding 3D keypoint positions are solved by multi-view triangulation from those labeled 2D keypoints, and finally, the object 6D poses are solved from 3D keypoints as an Orthogonal Procrustes problem and propagated to all frames. In TOD, the authors also introduced a method to record ground truth depth images: they record the positions of transparent objects in the scene, and put their opaque counterparts that share the same shape at the same pose in separate collects. This pipeline was also used for real-world data collection in ClearGrasp [161], however, it's extremely inefficient to replace transparent objects and their counterparts repetitively. Moreover, it is unclear how or whether this approach could be applied to data collection in complex scenes with cluttered objects as is typical in household settings. In Xu et al. [196], transparent objects are placed in several fixed locations relative to AprilTag arrays, with pose distribution diversity achieved by attaching a camera to a robot end-effector. This method still restricts the relative position between objects and is inefficient for complex scenes. In Fang et al. [53], all objects are attached to a large visual IR marker so that an optical tracking algorithm can estimate the objects' 6D poses. In this way, the collection can support dynamic scenes. On the other hand, all the object instances in collected data are accompanied by visually obscuring external labels which may not exist in natural environments.

Overall, datasets except for StereObj1M are still not large-scale and require external efforts on hardware, such as deploying robotic arms, calibration between multiple sensors, fiducial or optical markers. Instead of using markers or complex robotic apparatuses, we take advantage of an existing labeling system, ProgressLabeller [29], that is based on visual SLAM to produce accurate camera poses efficiently on recorded RGB-D videos. There are two assumptions in our labeling pipeline, both of which can be easily met: our pipeline assumes static scenes during video capturing and that scene backgrounds have adequate RGB features for visual SLAM processing.

6.2.2 Transparent Depth Completion and Object Pose Estimation

Zhang et al. [212] presented early work on the problem of depth completion from inaccurate depth using deep neural networks. Zhang et al. introduced an approach to estimate surface normal and boundaries from RGB images that then solved for the completed depth using optimization. ClearGrasp [161] adapted the method to work for transparent objects and demonstrated robotic grasping experiments on transparent objects from completed depth. Tang et al. [176] integrated the ClearGrasp network structure with adversarial learning to improve depth completion accuracy. Zhu et al. [217] proposed a framework that learns local implicit depth functions from the inspiration of neural radiance fields and performs self-refinement to complete the depth of transparent objects. Xu et al. [196] proposed to first complete the point cloud by projecting the original depth using a 3D encoder-decoder U-Net and then re-project the completed point cloud back to depth, and finally complete this depth using another encoder-decoder network given the ground truth mask. Fang et al. [53] also used a U-Net architecture to perform depth completion and demonstrated robotic grasping capabilities with their approach.

KeyPose [112] was proposed for keypoint-based transparent object pose estimation on stereo images. It outperformed DenseFusion [187], even with ground truth depth, and achieved high accuracy on the TOD dataset. Chang et al. [20] proposed a 3D bounding box representation and reported results comparable to KeyPose in multi-view pose estimation. StereObj1M [111] benchmarked KeyPose and another RGB-based object pose estimator, PVNet [147], on more challenging objects and scenes, where both methods achieved lower accuracy with respect to the ADD-S AUC metric (introduced in [192]) with both monocular and stereo input. Xu et al. [195] proposed a two-stage pose estimation framework that performs image segmentation, surface normal estimation, and plane approximation in the first stage. The second stage then combines output from the first stage with color and depth RoI features for input to an RGB-D pose estimator originally designed for opaque objects [180] to regress 6D object poses. This method also outperformed DenseFusion and [180] fed with ClearGrasp output depth by a large margin. In this paper, we evaluate how well state-of-the-art RGB-D methods [73] designed for opaque object pose estimation can perform on transparent objects compared with [195].

6.3 Dataset

6.3.1 Dataset Objects and Statistics

As shown in Figure 6.3, there are 63 objects included in the ClearPose dataset. There are 49 household objects, including 14 water cups, 9 wine cups (with a thin stem compared with water cups), 5 bottles (with an opening smaller than the cross-section of the cylindrical body), 6 bowls, 5 containers (with 4 corners while bowls are classified with round shapes), and several other objects like pitcher, mug, spoon, etc. Moreover, the dataset contains 14 chemical supply objects, including a syringe, a glass stick, 2 reagent bottles, 3 pans, 2 graduated cylinders, a funnel, a flask, a beaker, and 2 droppers.

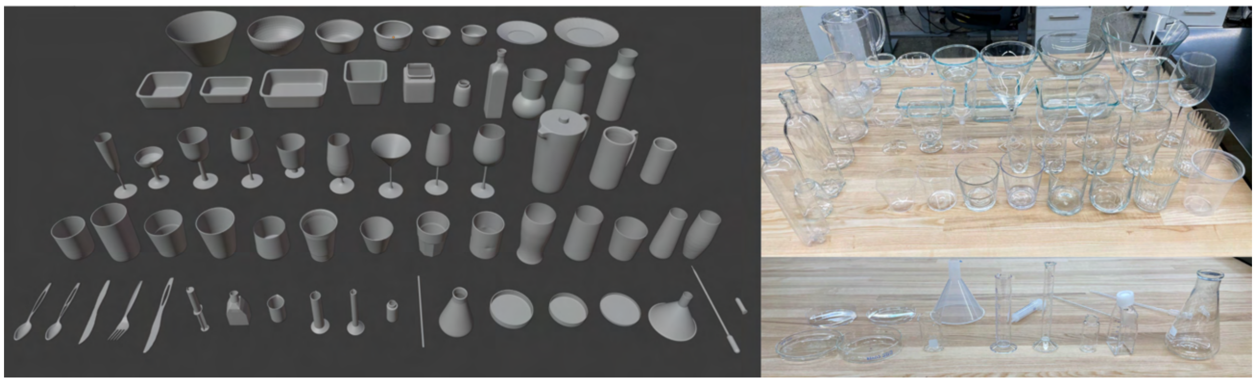


Figure 6.3: Objects included in the ClearPose dataset. On the left, we show the rendered CAD models for each object. From top to bottom there are bowls and plates (1st row), containers and bottles (2nd row), wine cups, two mugs and a pitcher (3rd row), water cups (4th row), and spoons, a fork, knives as well as chemical supplies (bottom row). On the right, we show real images of household objects on the top right, and chemical supplies on the bottom right.

All the images are collected using a RealSense L515 RGB-depth camera, with a raw resolution of 1280×720. After object pose annotation, the central part of each image is cropped and reshaped to 640×480 for reduced storage space and faster CNN training and inference. For the training set, we separate all 63 objects into 5 separate subsets and collected 4-5 scenes with different backgrounds for each subset. Each scene is scanned by the hand-held camera moving around the tabletop scene at 3 different heights with 3 different lighting conditions (bright room light, dim room light, dim room light with sidelight from a photography lighting board). For the testing set, as the appearance of transparent objects depends on their context within a scene, we consider 6 different test cases and collect corresponding scenes as follows: (1) different backgrounds: novel backgrounds that never appeared in the training scenes with each object subset. (2) heavy occlusions: cluttered scenes each with about 25 objects that form multiple layers of occlusion when viewed from the table's side. (3) translucent/transparent covers: scenes with all transparent objects placed inside a translucent box. (4) together with opaque objects: transparent objects placed together with opaque YCB and HOPE objects, which did not appear in the training set. (5) filled with liquid: scenes with transparent objects filled with different colored liquid. (6) non-planar configuration: scenes with objects placed

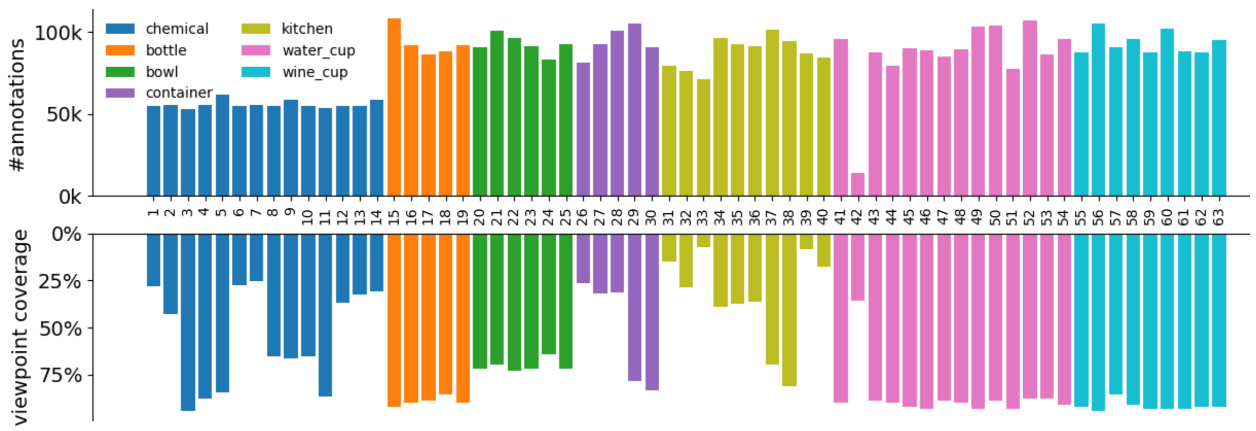


Figure 6.4: Distribution of instance annotations and viewpoint orientation coverage statistics for every object in the ClearPose dataset. The 'kitchen' category includes fork, spoon, knife, mug, plate and pitcher objects.

onto different surfaces with multiple heights. Sample RGB images from both training and test sets are included in Figure 6.2.

We calculate several statistics about the ClearPose dataset. In total there are 362,325 RGB-D frames captured in 56 scenes, with 4,927,806 object instance annotations with 6 DoF poses, segmentation masks, surface normals, and ground truth depth images. The distribution of object instance annotation and camera viewpoint coverage are shown in Figure 6.4 colored by object category, where we see our dataset has roughly even viewpoint coverage for most objects. Viewpoint coverage is calculated by projecting collected object orientations onto a unit sphere and counting the covered region percentage over the sphere's surface. For symmetrical objects, regions with the same object appearance are considered together. Some objects like plates, forks, large bowls can only be placed in certain orientations, so they have reduced viewpoint coverage. The 2nd water_cup was broken during the data collection process so it has fewer instance annotations.

6.3.2 Pose Annotation

We use the ProgressLabeller [29] to annotate the 6D poses of transparent objects and render object-wise segmentation masks, ground truth depths, surface normals, etc. from the labeled poses. As shown in Figure 6.5, the first step of the ProgressLabeller pipeline is to run ORB-SLAM3 [16] on collected RGB-D video frames to produce camera pose estimates. During data collection, we notice

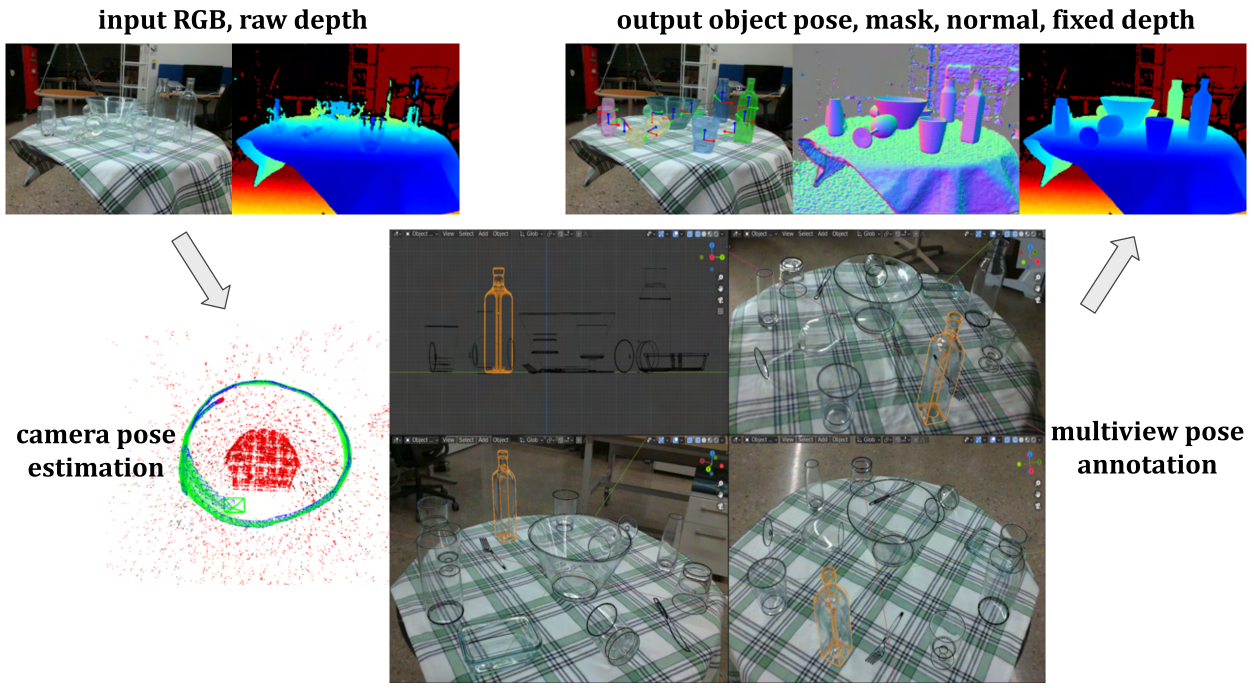

input RGB, raw depth
output object pose, mask, normal, fixed depth
camera pose estimation
multiview pose annotation

Figure 6.5: Transparent object pose annotation pipeline using ProgressLabeller. On top-left we show one sample of aligned RGB-D images from continuous streams captured by the camera. The frames are fed into visual SLAM to estimate camera trajectory, then the objects' poses are annotated in a multi-view silhouette alignment interface shown at the bottom-right. Finally, the object poses, surface normals, and fixed depths are calculated and rendered as output.

ORB-SLAM3 sometimes couldn't estimate camera pose well in case of extreme transparent object clutter, where background RGB features are heavily distorted. In these cases, the camera view needs to capture some background area or other landmark objects that can provide stable features. The next step is to import the reconstructed scene (cameras and point cloud) into the Blender workspace, select several camera views from different orientations, and import the object 3D CAD model into the workspace. Then, the object silhouettes/boundaries can be directly compared and matched with original RGB images from multiple views simultaneously when the user drags the object across the scene to tune its position and orientation. Figure 6.5 shows an example case of a matched transparent bottle. Optionally, the user can select to first locate the object onto the 2D fitting plane of the support table, etc., as shown on top-left of Figure 6.5. After labeling all objects' poses in the 3D workspace, their poses in every camera frame are calculated by dividing them with estimated camera poses. The output ground truth depth images (fixed depth) are generated by overlaying rendered object CAD model depth to the original depth images. Then the surface normals are calculated from the depth images. It takes around 30 minutes to finish labeling one scene using this pipeline, including visual SLAM for camera pose estimation, object pose manual alignment, and output image rendering.

6.4 Benchmark Experiments

In this section, we provide benchmark results of recent research on depth-related transparent object perception using deep neural networks, including scene level depth completion, and both instance-level and category-level RGB-D pose estimation. As mentioned in Section 6.3.1, we test the generalization capability of such systems on 6 aspects of appearance novelty with transparent objects: new background, heavy occlusion, translucent cover, opaque distractor objects, filled with liquid, and non-planar placement. Specifically, around 200K images are selected for training, and for each of 6 test cases, 2K images from corresponding scenes are randomly sampled to compose the testing set for evaluation.

6.4.1 Depth Completion

We selected two recent depth completion works that are publicly available, ImplicitDepth [217] and TransCG [53] as baseline methods for the depth completion task on transparent object scenes. (ClearGrasp [161] was shown to be less accurate than both works, and Transparenet [196] was released around the date of this submission.) ImplicitDepth is a two-stage method that learns local implicit depth functions in the first stage through ray-voxel pairs similar to neural rendering and refines the depth in the second stage. TransCG is built on DFNet [79] which was initially developed for image completion. We trained both networks following their original papers' training iterations and hyper-parameters. Specifically, ImplicitDepth was trained on a 16G RTX3080 GPU with a 0.001 learning rate and iterated around 2M frames for each of the two stages. TransCG was trained on an 8G RTX3070 GPU with a fixed 0.001 learning rate and iterated around 900K frames in total. Both works use Adam as the optimizer. Then we evaluated the two works on 6 test sets mentioned in Section 6.3.1 with metrics defined in [50]. The results are shown in Table 6.2. TransCG surpassed ImplicitDepth in most tests with fewer training iterations, which implies that methods using DFNet can outperform designs using voxel-based PointNet for transparent depth completion. Across different tests, both methods perform poorly in Translucent Cover scenes and achieved the best performance in New Background scenes. Other scene variations such as Filled Liquid, Opaque Distractor, and Non Planar do not substantially impact the methods' accuracy. Figure 6.6 shows examples of qualitative results from both methods compared with the ground truth.

Table 6.2: Depth completion benchmark results of ImplicitDepth and TransCG on 6 different test scenarios of the ClearPose dataset.

Testset	Metric	RMSE↓	REL↓	MAE↓	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25}\uparrow$
New Background	ImplicitDepth	0.07	0.05	0.04	67.00	87.03	97.50
	TransCG	0.03	0.03	0.02	86.50	97.02	99.74
Heavy Occlusion	ImplicitDepth	0.11	0.09	0.08	41.43	66.52	91.96
	TransCG	0.06	0.04	0.04	72.03	90.61	98.73
Translucent Cover	ImplicitDepth	0.16	0.16	0.13	22.85	41.17	73.11
	TransCG	0.16	0.15	0.14	23.44	39.75	67.56
Opaque Distractor	ImplicitDepth	0.14	0.13	0.10	34.41	55.59	83.23
	TransCG	0.08	0.06	0.06	52.43	75.52	97.53
Filled Liquid	ImplicitDepth	0.14	0.13	0.11	32.84	53.44	84.84
	TransCG	0.04	0.04	0.03	77.65	93.81	99.50
Non Planar	ImplicitDepth	0.18	0.16	0.15	20.34	38.57	74.02
	TransCG	0.09	0.07	0.07	55.31	76.47	94.88

6.4.2 Instance-Level Object Pose Estimation

There is one recent work of Xu et al. [195] focusing on transparent object pose estimation using raw RGB and depth images. This work doesn't have source code publicly available, so we re-implemented the proposed method following the original paper for inclusion in our benchmark analysis. This method is implemented as a two-stage pipeline, for which we trained Mask R-CNN [71] for instance-level segmentation and DeepLabv3 [24] for surface normal estimation, and with an XYZ 3D coordinate map of the supporting plane feature in the first stage. The second stage replicates most of the architecture and loss functions described in [180] to ultimately regress dense

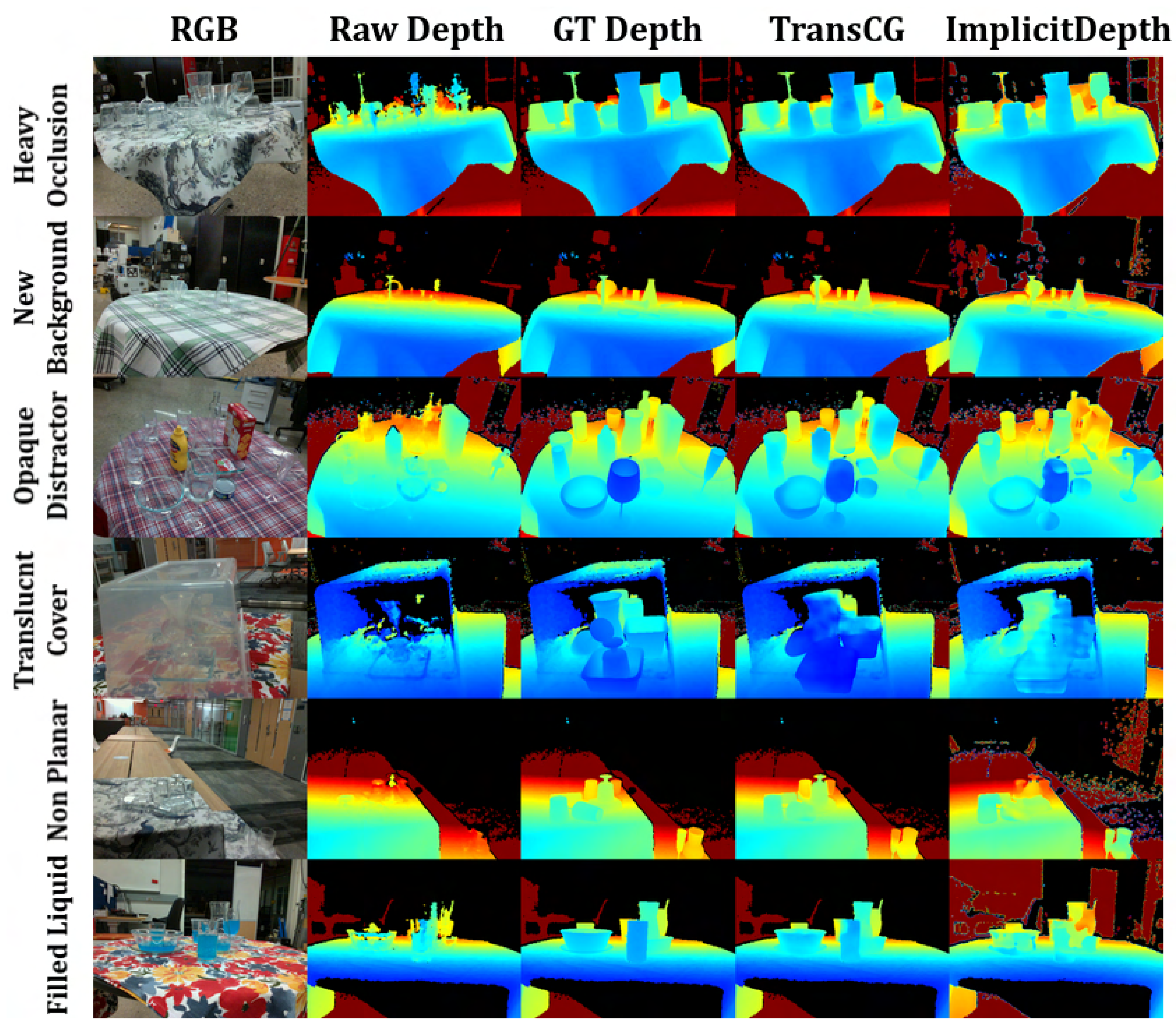


Figure 6.6: Qualitative depth completion results. From left to right, there are RGB image, raw depth, ground truth depth rendered using object CAD models, completed depth by TransCG and ImplicitDepth.

pixel-wise 3D translation, 3D rotation delta from a set of fixed rotation anchors, and confidence scores. In practice, we trained the networks on an RTX2080-SUPER GPU. Mask R-CNN has trained 5 epochs with SGD optimizer, batch size of 5, and learning rate of 0.005. DeepLabv3 was trained 2 epochs with Adam optimizer, batch size of 4 and learning rate of 0.0001, and second stage network was trained around 180K iterations with Adam optimizer, batch size of 4, and learning rate of 0.0005. We compare this method [195] with a state-of-the-art RGB-D pose estimator that was originally designed for opaque objects, FFB6D [73]. The FFB6D estimator follows a two-stream RGB and point cloud encoder-decoder architecture with fusion between blocks. FFB6D is trained on a 16G RTX3080 GPU for 5 epochs with batch size 6. All the hyper-parameters follow the default value from the original implementation. For our analysis of FFB6D pose estimation performance, we run experiments with different depth options in training and testing: with raw, ground truth, and completed depth from TransCG, as detailed in Table 6.3.

Table 6.3: Pose estimation accuracy comparison on different test sets of the ClearPose dataset. FFB6D$_{r/r}$ refers to train and test FFB6D model both on raw depth. Similarly, FFB6D$_{g/c}$, FFB6D$_{g/g}$ refer to train on ground truth, test on completed depth from TransCG, and train and test both on ground truth, respectively. The values are averaged across all objects in the dataset.

Testset	Metric	Xu et al.	FFB6D$_{r/r}$	FFB6D$_{g/c}$	FFB6D$_{g/g}$
New Background	Accuracy	50.958	44.264	49.517	59.694
	ADD(-S)	45.233	43.452	47.691	58.224
Heavy Occlusion	Accuracy	24.193	14.723	15.160	26.331
	ADD(-S)	22.953	17.869	17.862	31.875
Translucent Cover	Accuracy	14.353	5.5617	4.5345	13.433
	ADD(-S)	14.311	7.5983	5.8054	17.620
Opaque Distractor	Accuracy	42.630	0.4618	1.3331	2.3525
	ADD(-S)	39.036	0.7628	1.5516	3.0685
Filled Liquid	Accuracy	34.500	7.6908	9.0584	16.228
	ADD(-S)	32.251	11.153	10.828	18.583
Non Planar	Accuracy	21.024	7.4924	7.5843	15.567
	ADD(-S)	18.411	7.8021	6.7339	16.986

For evaluation, we use two metrics based on Average-point-Distance (ADD and ADD-S) from [192]. ADD is calculated as the average Euclidean distance of corresponding point pairs from two object point clouds separately at the ground truth pose and predicted pose. ADD-S is calculated as the minimum distance of every point from the predicted point cloud to the ground truth point cloud. In Table 6.3, 'Accuracy' is calculated as the percentage of all pose estimates on the test set with ADD error less than 10cm. 'ADD(-S)' is calculated as Accuracy-Under-Curve area integrated from 0-10cm error, which is then scaled from 1 to 100 as the percentage.

As shown in Table 6.3, from the comparison between different training and testing combination within FFB6D, the upper bound performance appears when the network are both trained and tested on ground truth depth. When both the training and testing data come to raw, the metric drops a lot.

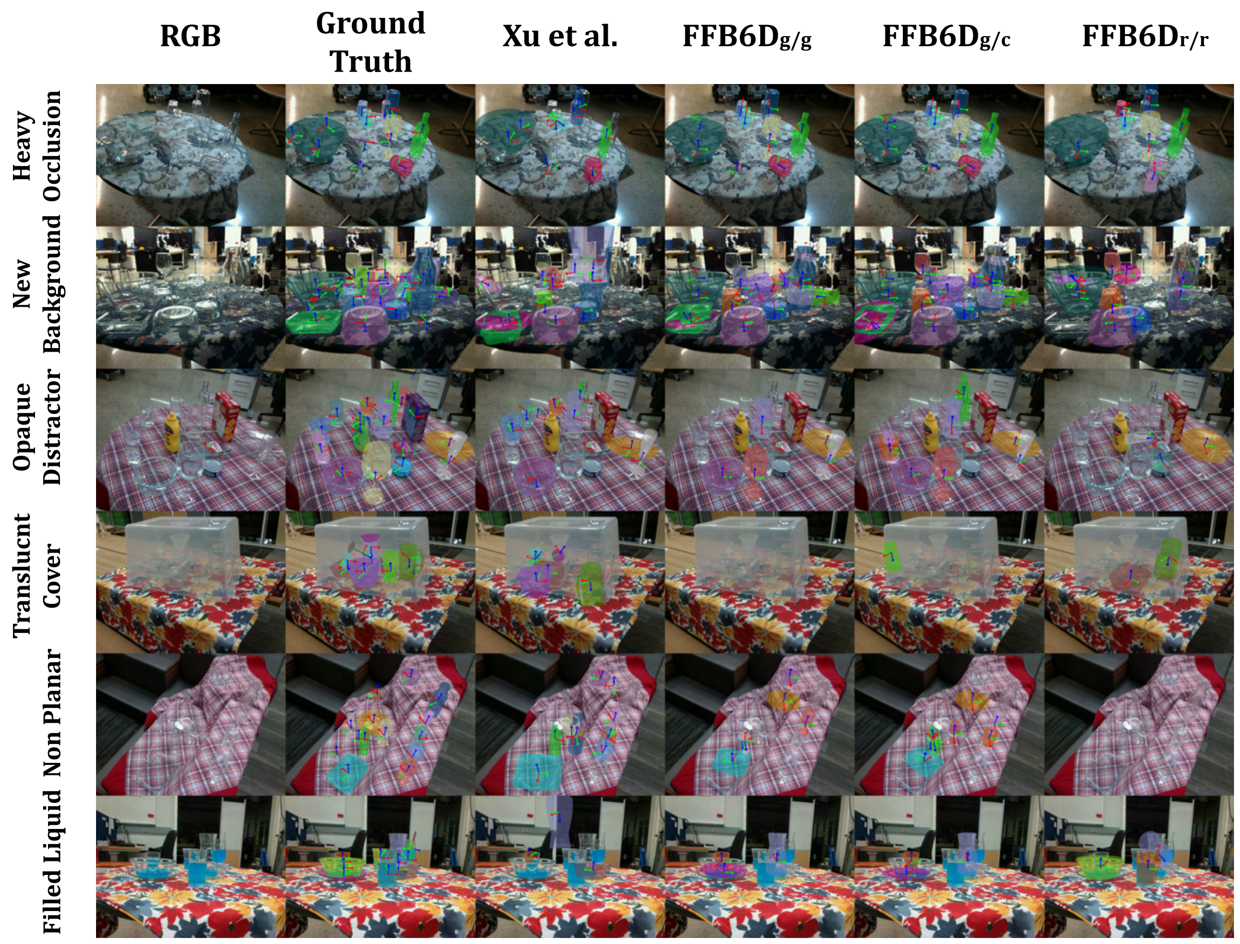


Figure 6.7: Visualization of pose estimation in ClearPose dataset. From left to right, they are raw image, ground truth object poses, pose estimation results of method from Xu et al., FFB6D$_{g/g}$, FFB6D$_{g/c}$, FFB6D$_{r/r}$. From top to bottom, results are shown in different test scenes in Table 6.3. Objects are projected to color masks based on their pose estimates, with their 6DoF poses marked as the red-green-blue coordinate frame.

Obviously, inaccurate depth would be the difficulty for transparent object pose estimation. It should be mentioned that training on ground truth depth, testing on completed depth (from TransCG) almost display the same accuracy. Although TransCG is good at depth completion, the disparity between ground truth and depth completion would make the network in vain. Generally, the easiest test case is New Background, and the accuracy drops a lot in the other 5 scenarios. When we compare the accuracy of Xu et al. with variants of FFB6D, we find they are comparable in New Background, Heavy Occlusion, Translucent Cover, and Non Planar scenes, while Xu et al. is much better in Opaque Distractor and Filled Liquid scenes. One possible reason is that there are some unseen colors mixing in the transparent objects, adding remarkable noise to object keypoint regression during the FFB6D inference process, which is not used by Xu et al. Overall, the pose estimation accuracy of current methods is still much worse than that on opaque objects with RGB features (with ADD-S around 90 on public datasets [78]). Some qualitative examples of pose estimates are shown in Figure 6.7.

6.5 Discussions

There are some common classes of objects with transparency/translucency not included in our dataset, for example, those with colored transparent/translucent materials, with markers or labels, together with opaque parts, etc. Instead, our focus in the ClearPose dataset is to investigate pure transparency that exhibits relatively few features for perception. On the other hand, we anticipate the open-source ProgressLabeller [29] will facilitate more large-scale customized transparent datasets in the future.

As for benchmarking perception models, we didn't include a complete list of recent state-of-the-art approaches due to resource constraints (i.e. compute and time limitations). Based on our current dataset and benchmark results, there are several possible extensions: (1) Comparison of RGB-only pose estimators with RGB-D methods that are free of transparent object broken depth issues. (2) Category-level pose estimation for transparent objects, for which the ClearPose dataset has categories of bowls, bottles, wine_cups, etc. that are with similar 3D shape and topology. (3) Neural rendering on transparent objects considering environment contexts, such as varied lighting and occlusions. (4) Transparent object grasping and manipulation experiment in practical scenes, including the 6 test scenarios mentioned in the benchmark.

Besides, an especially interesting problem emerging from our heavy cluttered, and translucent covered test scenes is the multi-layer appearance of transparent objects. As shown in Figure 6.8, because of transparency/translucency, some image pixels could belong to more than one object. New detection and segmentation annotation rules, such as bounding box non-maximum suppression threshold, or segmentation mask format over the image, could be proposed and explored based on

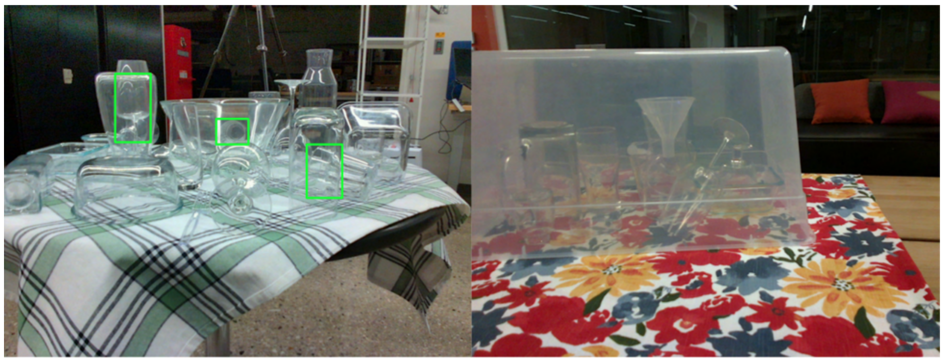


Figure 6.8: Examples of multi-layer transparent object appearance. In the left image, the annotated bounding boxes show large overlap between object pairs, where the objects behind are still perceivable in some cases with less light distortion. In the right image, objects behind the translucent surface are still detectable as well.

our dataset as future work.

6.6 Summary

In this paper, we described the contribution of **ClearPose**, a new large-scale RGB-D transparent object dataset with annotated poses, masks, and associated labels created using a recently proposed pipeline. We performed a set of benchmarking experiments on depth completion and object pose estimation tasks using state-of-the-art methods over 6 different generalization test cases that are common in practical scenarios. Results from our experiments demonstrate that there is still much room for improvement in some cases, such as heavy clutter, transparent objects filled with liquid, or being covered by other translucent surfaces. The dataset and benchmark code implementations will be made public with the intention to support further research progress in transparent RGB-D visual perception.

CHAPTER 7

TransNet: Transparent Object Manipulation Through Category-Level Pose Estimation

7.1 Motivation

From glass doors and windows to kitchenware and all kinds of containers, objects with transparent materials are prevalent throughout daily life. Thus, perceiving the pose (position and orientation) of transparent objects is a crucial capability for autonomous perception systems seeking to interact with their environment. However, transparent objects present unique perception challenges both in the RGB and depth domains.

For RGB, the color appearance of transparent objects is highly dependent on the background, viewing angle, material, and lighting condition, due to light reflection and refraction effects. For depth, common commercially available depth sensors record mostly invalid or inaccurate depth values within the transparent region. Such visual challenges, especially missing detection in the depth domain, pose severe challenges for autonomous object manipulation and obstacle avoidance tasks. This paper addresses these challenges by studying how category-level transparent object pose estimation may be achieved using end-to-end learning.

Recent works have shown promising results on grasping transparent objects by completing the missing depth values followed by the use of a geometry-based grasp engine [161, 81, 53], transfer learning from RGB-based grasping neural networks [189], light-field feature learning [215], or domain-randomized depth noise simulation [39]. For more advanced manipulation tasks such as rigid body pick-and-place or liquid pouring, geometry-based estimations, such as symmetrical axes, edges [148] or object poses [121, 195, 39], are required to model the manipulation trajectories. Instance-level transparent object poses could be estimated from keypoints on stereo RGB images [112, 111], a light-field camera [216, 214], or directly from a single RGB-D image [195] with support plane assumptions. Recently emerged large-scale transparent object datasets [161, 196, 111, 53, 30] pave the way for addressing the problem using deep learning.

In this paper, we set out to extend the frontier of 3D transparent object perception by building upon recent work. We introduce *TransNet* [209], a category-level pose estimation pipeline for transparent objects that outperforms a state-of-the-art baseline. We further explore the effect of input modalities, feature embedding methods, and depth-normal consistency through the learning process to provide insights for future research. Moreover, we build an autonomous transparent object manipulation system using TransNet and demonstrate its efficacy in pick-place and pouring tasks.

7.2 Related Work

7.2.1 Transparent Object Visual Perception for Manipulation

Transparent objects must be perceived before they can be acted on. Lai *et al.* [101] and Khaing *et al.* [93] developed CNN models to detect transparent objects from RGB images. Xie *et al.* [194] proposed a deep segmentation model that achieved state-of-the-art accuracy. ClearGrasp [161] employed depth completion for use with pose estimation on robotic grasping tasks, where they trained three DeepLabv3+ [25] models to learn transparency mask, surface normal, and boundary, respectively. Follow-on studies developed different approaches for depth completion, including implicit functions [217], NeRF reconstruction [81], combined point cloud and depth features [196], adversarial learning [176], multi-view geometry [20], RGB image completion [53], and sim2real transfer [39]. Weng *et al.* [189] used transfer learning from the RGB to the depth sensor domain without completing raw depth. For instance-level pose estimation, Xu *et al.* [195] utilized segmentation, surface normal, and image coordinate UV-map as input to a network similar to [180] that can estimate 6 DOF object pose. Keypose [112] proposed to estimate 2D keypoints and regress object poses from stereo images using triangulation. For other special sensors, Xu *et al.* [199] used light-field images to perform segmentation using a graph-cut-based approach. Kalra *et al.* [90] trained Mask R-CNN [71] using polarization images as input to outperform the baseline that was trained on only RGB images by a large margin. Zhou *et al.* [216, 215, 214] employed light-field images to learn features for robotic grasping and object pose estimation. Along with the proposed methods, massive datasets, across different sensors and both synthetic and real-world domains, have been collected and made public for various related tasks [194, 161, 112, 214, 90, 111, 217, 196, 53, 30]. Compared with these previous works, and to the best of the authors' knowledge, we propose the first category-level pose (6D pose + 3D scale) estimation approach developed specifically for transparent objects.

7.2.2 Opaque Object Category-level Pose Estimation

Category-level object pose estimation aims to estimate unseen objects' 6D pose within seen categories and their scale or canonical shape. Wang *et al.* [188] introduced the Normalized Object Coordinate Space (NOCS) for dense 3D correspondence learning and solved for object pose and scale using the learned NOCS map and masked depth [184]. More recent works explored different aspects to improve pose estimation accuracy. A category-level shape prior is found to be beneficial for pose estimation accuracy in [179] and further improved in [23, 26, 47]. DualPoseNet [106], 6D-ViT [218], ACR-Pose [52], and CPPF [203] proposed to incorporate rotation-invariant embedding, Transformer networks, Generative Adversarial Networks, and deep point-pair-feature, respectively. However, these techniques require high-quality depth input provided by opaque objects with Lambertian light reflectance. One recent work by Dai *et al.* [39] proposed a data generation system that simulates the noise on non-Lambertian surfaces of active stereo depth cameras and demonstrated its usage in category-level pose estimation and robotic grasping.

Compared with these techniques, TransNet takes advantage of recent advances in transparent depth completion approaches and directly learns category-level pose estimation from imperfect depth predictions, carefully considering multi-modal input, feature embedding architecture, and depth-normal consistency.

7.3 TransNet

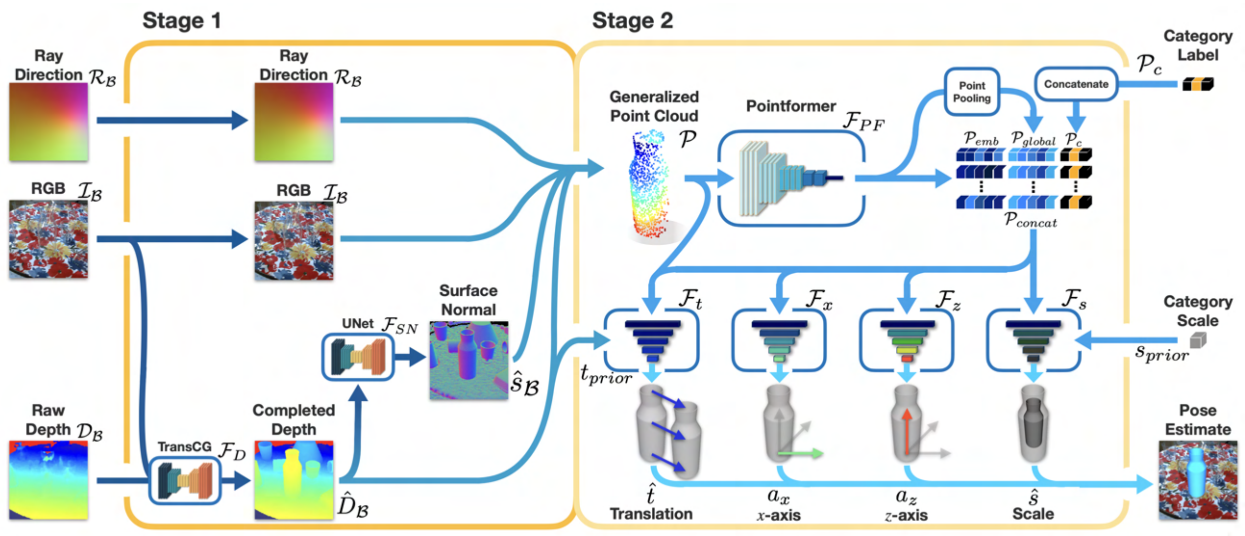


Figure 7.1: The two-stage architecture of TransNet. Image patches of RGB, ray direction, surface normal, and completed depth are concatenated and embedded to learn the translation, x-axis, z-axis, and scale separately of the estimated category-level pose.

Given an input RGB-D pair $(\mathcal{I}, \mathcal{D})$ viewing a scene that includes transparent objects, our goal is to predict the objects' 6D rigid body transformations $[\mathbf{R}|\mathbf{t}]$, 3D scales $\mathbf{s}$ in the camera coordinate frame and object classes within a predefined category set, where $\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3$ and $\mathbf{s} \in \mathbb{R}^3_+$. In this problem, inaccurate or invalid depth readings exist within the image region of transparent objects (represented as a binary mask). To solve the category-level pose estimation problem, we propose a novel two-stage deep neural network pipeline, *TransNet*.

7.3.1 Architecture Overview

TransNet first applies a fine-tuned instance segmentation module to extract individual object patches from the overall image. As shown in Figure 7.1, in the first stage, TransNet takes the patches as input and extracts multi-modal features corresponding to each transparent object instance. Depth completion and surface normal estimation are applied on RGB-D patches to obtain depth-normal pairs through cross-task consistency learning. The estimated depth-normal pairs, RGB, and ray direction patches are concatenated to feature patches, followed by a random sampling strategy within the instance masks to generate features referred to as the *generalized point cloud*. In the second stage, the generalized point cloud is processed through Pointformer [218] to produce embedded feature vectors. The pose is then separately estimated in four decoder modules for object translation, x-axis, z-axis, and scale, respectively.

7.3.2 Object Instance Detection and Segmentation

Similar to other category-level pose estimation work [47], we fine-tune a Mask R-CNN [71] model to obtain the object's bounding box $\mathcal{B}$, segmentation mask $\mathcal{M}$ and category label $\mathcal{P}_c$. Patches of ray direction $\mathcal{R}_\mathcal{B}$, RGB $\mathcal{I}_\mathcal{B}$ and raw depth $\mathcal{D}_\mathcal{B}$ are extracted according to the bounding box $\mathcal{B}$ and serve as input to the first stage of TransNet. The UV mapping itself is an important cue when estimating poses from patches [83], as it provides information about the relative position and size of the patches within the overall image. We use ray direction instead of UV mapping because it also contains camera intrinsic information. Here the ray direction $\mathcal{R}$ represents the direction from camera origin to each pixel in the camera plane under the camera frame. For each pixel (u, v):

$$\begin{aligned} p &= \begin{bmatrix} u & v & 1 \end{bmatrix}^T \\ \mathcal{R} &= \frac{K^{-1}p}{\|K^{-1}p\|^2} \end{aligned} \tag{7.1}$$

where p is the homogeneous UV coordinate in the image plane and K is the camera intrinsic matrix.

7.3.3 Cross-task Consistency for Depth-Normal Pairs

We apply a recent state-of-the-art approach, TransCG [53] for depth completion ($\mathcal{F}_D$) and U-Net [155] for surface normal estimation ($\mathcal{F}_{SN}$).

$$
\begin{aligned}
\hat{\mathcal{D}}_{\mathcal{B}} &= \mathcal{F}_D\left(\mathcal{I}_{\mathcal{B}}, \mathcal{D}_{\mathcal{B}}\right) \\
\hat{\mathcal{S}}_{\mathcal{B}} &= \mathcal{F}_{SN}\left(\hat{\mathcal{D}}_{\mathcal{B}}\right)
\end{aligned}
\tag{7.2}
$$

where the depth estimation $\hat{\mathcal{D}}_{\mathcal{B}}$ is used as input to the surface normal estimation module. The two models are first trained separately with depth and surface normal losses, $\mathcal{L}_d$ and $\mathcal{L}_s$, and then trained together using cross-task consistency. Following [204], we design a consistency loss $\mathcal{L}_{con}$ to train both networks. Both $\mathcal{L}_d$ and $\mathcal{L}_s$ are implemented using L_2 loss for depth completion and surface normal estimation within transparent mask $\mathcal{M}$.

$$
\begin{aligned}
\mathcal{L}_d &= \frac{1}{N_p} \sum_{p \in \mathcal{M}} \left\| \hat{\mathcal{D}}_p - \mathcal{D}_p^* \right\|^2 \\
\mathcal{L}_s &= \frac{1}{N_p} \sum_{p \in \mathcal{M}} \left\| \hat{\mathcal{S}}_p - \mathcal{S}_p^* \right\|^2 \\
\mathcal{L}_{con} &= \frac{1}{N_p} \sum_{p \in \mathcal{M}} \left\| \hat{\mathcal{S}}_p - \mathcal{F}_{SN}(\mathcal{D}_{\mathcal{B}}^*)_p \right\|^2
\end{aligned}
\tag{7.3}
$$

where $\mathcal{D}^*$ and $\mathcal{S}^*$ are the ground truth depth and surface normal images, and N_p refers to the number of masked pixels, respectively. During training, these losses are used according to the perceptual loss training strategy described by Zamir et al. [204]. Following cross-task consistency training, the depth completion and surface normal estimation networks are frozen and used to generate input for the second stage of TransNet.

7.3.4 Generalized Point Cloud

The generalized point cloud is implemented as a concatenation of multi-modal input features: $\mathcal{I}_{\mathcal{B}}$, $\mathcal{R}_{\mathcal{B}}$, $\hat{\mathcal{D}}_{\mathcal{B}}$, and $\hat{\mathcal{S}}_{\mathcal{B}}$. Specifically, we randomly sample N pixels within the transparent mask $\mathcal{M}$ of the feature patch to obtain a generalized point cloud $\mathcal{P} \in \mathbb{R}^{N \times 10}$. The experiment described in Section 7.4.2 explores the best choice of features for use in the generalized point cloud.

7.3.5 Transformer Feature embedding

Given a generalized point cloud $\mathcal{P}$ corresponding to a detected object, TransNet employs Pointformer [218], a multi-stage transformer-based point cloud embedding method to encode object features $\mathcal{P}_{emb} \in \mathbb{R}^{N \times d_{emb}}$.

$$\mathcal{P}_{emb} = \mathcal{F}_{PF}\left(\mathcal{P}\right) \tag{7.4}$$

Experiments described in Section 7.4.2 evaluate the effectiveness of using alternative point cloud embedding methods, such as 3D-GCN [107], for use in the transparent object context with noisy depth data.

Next, a Point Pooling layer (a multilayer perceptron (MLP) with max-pooling) is applied to extract global features $\mathcal{P}_{global}$. The global features are then concatenated with local features $\mathcal{P}_{emb}$ and a one-hot category $\mathcal{P}_c$ label from the instance segmentation network. The concatenations result in a feature vector to be used as input to the decoder:

$$\begin{aligned}
&\mathcal{P}_{global} = \text{MaxPool}\left(\text{MLP}\left(\mathcal{P}_{emb}\right)\right) \\
&\mathcal{P}_{concat} = \left[\mathcal{P}_{emb}, \mathcal{P}_{global}, \mathcal{P}_c\right]
\end{aligned} \tag{7.5}$$

7.3.6 Pose and Scale Estimation

After the feature embeddings are extracted based on multi-modal input, four separate decoders are applied to estimate the object's translation, x-axis, z-axis, and scale.

Translation Residual Estimation The translation decoder $\mathcal{F}_t$ learns a 3D translation residual from the object translation prior t_{prior} calculated as the average of predicted 3D coordinate over the sampled pixels in $\mathcal{P}$.

$$\begin{gathered}
t_{prior} = \frac{1}{N_p} \sum_{p \in N} K^{-1} \left[u_p \ v_p \ 1\right]^T \hat{\mathcal{D}}_p \\
\hat{t} = t_{prior} + \mathcal{F}_t\left(\left[\mathcal{P}_{concat}, \mathcal{P}\right]\right)
\end{gathered} \tag{7.6}$$

where u_p, v_p are the 2D pixel coordinate for the selected pixel. We use the L_1 loss between the ground truth and estimated position:

$$\mathcal{L}_t = \left|\hat{t} - t^*\right| \tag{7.7}$$

Rotation Estimation

We decouple the 3D rotation matrix R into two orthogonal axes, x-axis a_x and z-axis a_z, and estimate them separately. The network learns confidence values to deal with the problem that the regressed two axes are not orthogonal:

$$\begin{aligned}
[\hat{a}_i, c_i] &= \mathcal{F}_i\left(\mathcal{P}_{concat}\right),\ i \in \{x, z\} \\
\theta_z &= \frac{c_x}{c_x + c_z}\left(\theta - \frac{\pi}{2}\right) \\
\theta_x &= \frac{c_z}{c_x + c_z}\left(\theta - \frac{\pi}{2}\right)
\end{aligned} \tag{7.8}$$

where c_x, c_z denotes the confidence for the learned axes. θ represents the angle between a_x and a_z. θ_x, θ_z are obtained by solving an optimization problem and then used to rotate a_x and a_z within their common plane.

For the training loss, we use L_1 plus cosine similarity loss $\mathcal{L}_{r_x}, \mathcal{L}_{r_z}$ for each individual axis. Angular loss, $\mathcal{L}_a$, is also used to constrain a perpendicular relationship between the two axes and confidence loss $\mathcal{L}_{con_x}, \mathcal{L}_{con_z}$ to learn axis confidence:

$$\begin{aligned}
\mathcal{L}_{r_i} &= |\hat{a}_i - a_i^*| + 1 - \langle \hat{a}_i, a_i^* \rangle,\ i \in \{x, z\} \\
\mathcal{L}_a &= \langle \hat{a}_x, \hat{a}_z \rangle \\
\mathcal{L}_{con_i} &= |c_i - \exp\left(\alpha \left\| \hat{a}_i - a_i^* \right\|_2\right)|,\ i \in \{x, z\}
\end{aligned} \tag{7.9}$$

where α is a constant to scale the distance. Thus the overall loss for the second stage is:

$$\begin{aligned}
\mathcal{L} =& \lambda_s \mathcal{L}_s + \lambda_t \mathcal{L}_t + \lambda_{r_x} \mathcal{L}_{r_x} + \lambda_{r_z} \mathcal{L}_{r_z} + \\
& \lambda_{r_a} \mathcal{L}_a + \lambda_{con_x} \mathcal{L}_{con_x} + \lambda_{con_z} \mathcal{L}_{con_z}
\end{aligned} \tag{7.10}$$

where $\lambda_{r_x}, \lambda_{r_z}, \lambda_{r_a}, \lambda_t, \lambda_s, \lambda_{con_x}, \lambda_{con_z}$ are weights for each loss.

Scale Residual Estimation Similar to the translation decoder, we define the scale prior s_{prior} as the average of scales of all object 3D CAD models within each category. Then the scale of a given instance is calculated as follows:

$$\hat{s} = s_{prior} + \mathcal{F}_s \left(\mathcal{P}_{concat} \right) \tag{7.11}$$

The loss function is defined as the L_1 loss between the ground truth scale and estimated scale:

$$\mathcal{L}_s = |\hat{s} - s^*| \tag{7.12}$$

7.4 Experiments

Dataset We evaluated TransNet and baseline models on the Clearpose Dataset [30] for depth completion, surface normal estimation, and categorical transparent object pose estimation tasks. As shown in Figure 7.2, 33 object instances from 3 ClearPose categories (*bowl, water_cup,* and *wine_cup*) were selected for evaluating the relevant models on each task. 25 of the objects are used in the training set, totaling 190K RGB-D images. Hence, 8 objects not seen during training are used for testing, with 60K test images available. In practice, we train the models using the full 190K image training set and uniformly sample images of the test object to form a representative 5K image test set. For the robot manipulation experiments, we used 1 *bowl* object and 1 *wine_cup* object from the test set along with another 1 new *wine_cup* and 2 new *water_cups* purchased from the market that fit the robot parallel gripper.

Implementation Details Our model was trained in several stages. For experiments included in the ablation study and baseline comparison, ground truth instance segmentations are used to generate input patches, while for the robot experiments, a Mask R-CNN [71] model fine-tuned on the Clearpose dataset is used for generating input patches. The image patches were generated from object bounding boxes and re-scaled to a fixed shape of 256×256 pixels. Cross-task consistency

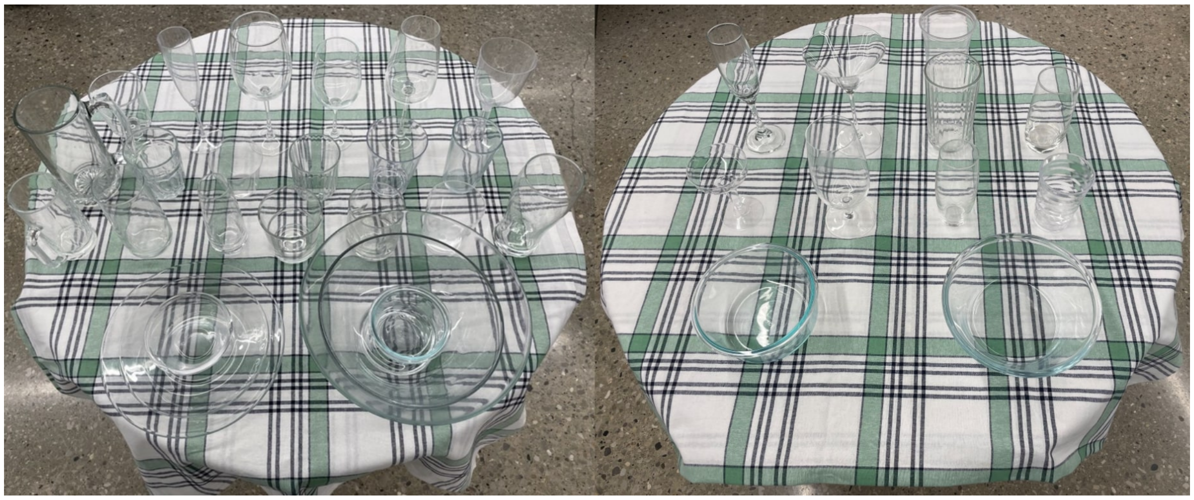


Figure 7.2: Transparent objects used in experiments. The image on the left show the 25 objects in the training set. The image on the right shows 8 novel objects in the test set for vision evaluation and robotic experiments.

training is performed using the AdamW optimizer [119] with learning rate of $1e^{-3}$. The pretrained surface normal estimation model provided by Zamir *et al.* [204] is used for initial network parameters. Cross-task training hyperparameters follow the perceptual loss strategy described in [204] until convergence using a batch size of 8. For TransCG, the AdamW optimizer [119] is used for training with $\lambda_{smooth} = 0.001$ and the overall learning rate is 0.001 to train the model until converge. For U-Net, we used the Adam optimizer [94] with a learning rate of $1e^{-4}$ to train the model until convergence. For both surface normal estimation and depth completion, the batch size was set to 24. For the second stage, the training hyperparameters of Pointformer and pose and scale estimation were selected following [218, 47]. The learning rate for all loss terms were kept the same during training, $\{\lambda_{r_x}, \lambda_{r_z}, \lambda_{r_a}, \lambda_t, \lambda_s, \lambda_{con_x}, \lambda_{con_z}\} = \{8, 8, 4, 8, 8, 1, 1\}\, e^{-4}$. The Ranger optimizer [109, 202, 211] was used with a linear warm-up for the first 1000 iterations, then a cosine annealing method at the 0.72 anneal point was used. All the experiments for pose estimation were trained on a 16G RTX3080 GPU until loss convergence.

Evaluation metrics For category-level pose estimation, this study follows [47, 26] in using 3D intersection over union (IoU) between the ground truth and estimated 3D bounding box at 25%, 50% and 75% thresholds. Additionally, $5°5cm$, $10°5cm$, and $10°10cm$ are used as metrics. The reported measurements for each of these three metrics represent the percentage of a model's pose estimates with error less than the specified metric's degree and distance thresholds.

For depth completion evaluation, the root of mean squared error (RMSE), absolute relative error (REL) and mean absolute error (MAE) are used as metrics along with $\delta_{1.05}$, $\delta_{1.10}$, $\delta_{1.25}$, where δ_n is calculated as:

$$\delta_n = \frac{1}{N_p} \sum_p \mathbf{I} \left(\max \left(\frac{\hat{\mathcal{D}}_p}{\mathcal{D}_p^*}, \frac{\mathcal{D}_p^*}{\hat{\mathcal{D}}_p} \right) < n \right) \tag{7.13}$$

where $\mathbf{I}(\cdot)$ represents the indicator function. $\hat{\mathcal{D}}_p$ and $\mathcal{D}_p^*$ represent estimated and ground truth depth for each pixel p.

For surface normal estimation, RMSE and MAE errors of normalized vectors are used as metrics with thresholds of $11.25°$, $22.5°$, and $30°$. Here, measurements reported for the $11.25°$ metric represent the percentage of estimates with an angular distance less than $11.25°$ from the ground truth surface normals. The mean angular error in degrees (MEAN) is also reported.

7.4.1 Comparison with Baseline

We chose one state-of-the-art categorical opaque object pose estimation model (GPV-Pose [47]) as a baseline, which was trained with estimated depth from TransCG [53] for a fair comparison. From Table 7.1, TransNet outperformed the baseline in most of the metrics on the Clearpose dataset. The $3D_{25}$ metric is relatively easy to achieve strong performance, so there is no substantial difference between the two models' corresponding $3D_{25}$ performance. When compared to the baseline on the remaining metrics, TransNet achieved around a 3 $\times$ improvement on $3D_{75}$, 3.5 $\times$ on $10°5$cm, $10°10$cm and 8 $\times$ on $5°5$cm. Qualitative results are shown in Figure 7.3 for TransNet.

Table 7.1: Comparison with the baseline on the Clearpose Dataset.

Method	$3D_{25}$ ↑	$3D_{50}$ ↑	$3D_{75}$ ↑	$5°5$cm ↑	$10°5$cm ↑	$10°10$cm ↑
GPV-Pose	95.4	65.1	13.2	2.7	12.5	15.5
TransNet	**97.3**	**80.3**	**39.9**	**22.7**	**45.4**	**50.6**

Table 7.2: Ablation study for TransNet

Trial	Embedding		Generalized Point Cloud				Consistency	Separate Category	$3D_{25}\uparrow$	$3D_{50}\uparrow$	$3D_{75}\uparrow$	$5°5cm\uparrow$	$10°5cm\uparrow$	$10°10cm\uparrow$
	PF	3D-GCN	RGB	Depth	Normal	Ray-direction								
1		✓	✓	✓		✓			98.3	72.7	17.6	6.6	22.5	28.0
2	✓		✓	✓		✓			97.9	76.9	28.4	9.4	8.0	29.6
3	✓		✓	✓	✓				95.4	61.6	11.6	3.6	15.3	31.3
4	✓		✓	✓	✓	✓			**98.4**	81.6	34.4	13.0	38.6	45.1
5	✓		✓	✓	✓	✓	✓		98.1	**81.8**	39.2	15.8	41.4	46.1
6	✓		✓	✓	✓	✓	✓	✓	97.3	80.3	**39.9**	**22.7**	**45.4**	**50.6**

7.4.2 Ablation Study

In Table 7.2, we compared the performance of TransNet to that of modified versions of the network architecture. The results of this study informed Trial 6 as the final architecture of TransNet and the one that was used in robot experiments. Results from the ablation study may also be used to guide future work in the area of transparent object pose estimation.

Embedding Method Between trial 1 and trial 2, we compared the effect of the embedding method, 3D-GCN [107] and Pointformer [218], on TransNet accuracy. The use of Pointformer resulted in higher accuracy for TransNet on most metrics compared to when 3D-GCN was used. More details are shown in Table 7.3. Crucially, when the ground truth depth was changed to estimated depth (modeling the change from opaque to transparent setting), Pointformer retained substantially more accuracy than 3D-GCN. A possible explanation for this result is based on how 3D-GCN propagates information between nearest neighbors. While this is an efficient method for propagating information locally, it suffered when given the noisy completed depth as input. Pointformer, on the other hand, shares information across the whole point cloud, possibly contributing to increased robustness when input data is noisy. Therefore, given depth with large uncertainty, the transformer-based embedding method might be more powerful than embedding methods using nearest neighbors.

Modalities for Generalized Point Cloud The comparison of trials 24 was used to identify the optimal combination of feature inputs for the generalized point cloud in TransNet. For trials 2 and 4, we compared the effect of adding the estimated surface normal to the generalized point cloud. Results from this comparison imply that surface normals serve as a good complement when depth estimation is not accurate, as in the transparent setting. For trials 3 and 4, we compared the potential

Table 7.3: Comparison between different embedding methods

Depth type	Embedding	$3D_{25}\uparrow$	$3D_{50}\uparrow$	$3D_{75}\uparrow$	$5°5cm\uparrow$	$10°5cm\uparrow$	$10°10cm\uparrow$
Ground truth	3D-GCN	99.8	96.6	55.4	70.9	86.0	90.3
	Pointformer	99.7	96.6	74.5	61.9	83.7	86.5
Estimation	3D-GCN	98.3	72.7	17.6	6.6	22.5	28.0
	Pointformer	97.9	76.9	28.4	9.4	29.6	35.3

Table 7.4: Accuracy for depth-normal pair estimation

Depth Metrics	RMSE↓	REL↓	MAE↓	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25}\uparrow$
w/o consistency	**0.056**	**0.044**	**0.041**	68.61	89.35	**98.89**
w consistency	0.057	**0.044**	**0.041**	**70.49**	**89.47**	98.57
Surface Normal Metrics	RMSE↓	MAE↓	MEAN↓	11.25°↑	22.5°↑	30°↑
w/o consistency	0.19	0.13	11.43	56.75	88.45	**96.64**
w consistency	**0.12**	**0.08**	**8.96**	**73.62**	**92.56**	96.58

benefit of including ray-direction. Notably, the ray-direction input contains both camera intrinsic and pixel positional information, which we hypothesize is critical for the network to be able to model accurate 3D spatial relationships. All the metrics showed that including both surface normal and ray-direction results in improved pose estimation accuracy.

Consistent Learning For trials 4 and 5 in Table 7.2, TransNet is trained without and with cross-task consistency, respectively. For trial 4, both surface normal estimation and depth completion networks in TransNet's first stage are trained separately and without the $\mathcal{L}_{con}$ loss. Results from this comparison show that adding cross-task consistency training improved TransNet performance on high accuracy metrics: $5°5$cm, $3D_{75}$. From Table 7.4, adding a cross-task consistency loss resulted in improved accuracy for depth-normal pair estimation, particularly for the surface normal estimation network. Therefore, cross-task consistent training improves the performance of category-level transparent object pose estimation and is included in TransNet.

Separate Category As shown in Table 7.2 when comparing results in trial 5 and trial 6, it was found that TransNet's accuracy improves when a single instantiation of the model is trained per category as opposed to being trained on multiple categories. Notably, this result is most apparent for the high accuracy metric $5°5$cm

Why TransNet Outperforms GPV-Pose Comparing GPV-Pose with the trial 1 model leads to an observation that both models use 3D-GCN and a similar network architecture. The only difference is that instead of using the generalized point cloud as input, GPV-Pose directly uses a point cloud (the multiplication of depth and ray direction) as input. Yet, trial 1 outperforms GPV-Pose on pose estimation tasks. We hypothesize that the camera intrinsic and pixel positional information contained in ray-direction degrades in accuracy after being multiplied directly with noisy depth data to form the point cloud input to GPV-Pose. In contrast, TransNet retains ray-direction and depth information as distinct feature channels in the generalized point cloud with learned transformations to combine each feature modality. These learned transformations potentially allow TransNet to be more robust in the face of noisy depth data.

7.4.3 Robot Experiment

We used a Fetch robot for manipulation and an Intel RealSense L515 camera to take RGB-D images as used in the ClearPose dataset. The robot-camera extrinsic calibration is done using AprilTags [98]. Inspired from [31], we designed two robot manipulation tasks: pouring and pick-and-place to evaluate the efficacy of TransNet to support robot manipulation of transparent objects using 3 *water_cups*, 2 *wine_cups*, and 1 *bowl* not appearing in the training set.

For grasping the object, we calculate the approaching direction as orthogonal to the object's symmetrical axis and the grasp location as the object's center point.

Pick-and-place This task is composed of a pick action and a place action. For the pick action, the robot is to grasp a cup (either *water_cup* or *wine_cup*) on the table-top with its parallel gripper based on its pose estimate from TransNet. For the place action, the robot is to place the cup upright in a pre-defined location. The target poses for the robot's gripper are solved by aligning the object's estimated bottom surface parallel to the target location surface with a 2 cm offset.

Pouring This task is composed of a pick action and a pouring action. The pick action follows the same protocol as *pick-and-place*. For the pouring action, the robot moved the cup above the bowl based on the bowl's estimated location while also keeping the cup's z axis upright to prevent

Table 7.5: Robot experiment success rate

Task	Objects	#Pick	Success Rate	#Place(Pour)	Success Rate
Pick & Place	water_cup	16/20	80%	12/16	75%
	wine_cup	13/20	65%	12/13	92.3%
Pouring	water_cup & bowl	15/20	75%	10/15	66.7%
	wine_cup & bowl	14/20	70%	11/14	78.6%

spilling. Then the content in the cup is poured into the bowl by tilting the cup's z axis.

Result The poses and scales used by the robot are output directly from TransNet without additional post-processing. Quantitative results measuring the robot's success rate are included in Table 7.5. TransNet enabled the robot to perform pour and pick-place actions with promising success rates. Visualization examples are included in Figure 7.4.

7.5 Summary

In this paper, we introduced *TransNet*, a two-stage pipeline for category-level transparent object pose estimation that outperformed the baselines. Ablation studies were performed to understand the benefits of using multi-modal input, feature embedding modules, and cross-task consistency. The efficacy of TransNet was demonstrated by enabling a robot system to perform transparent object manipulation tasks with high success rates. In the future, we could further improve the pose estimation accuracy as there is still a large gap between TransNet and state-of-the-art opaque object datasets. One direction is to consider objects' material information, as we observed a performance gap between perceiving glass and plastic objects.

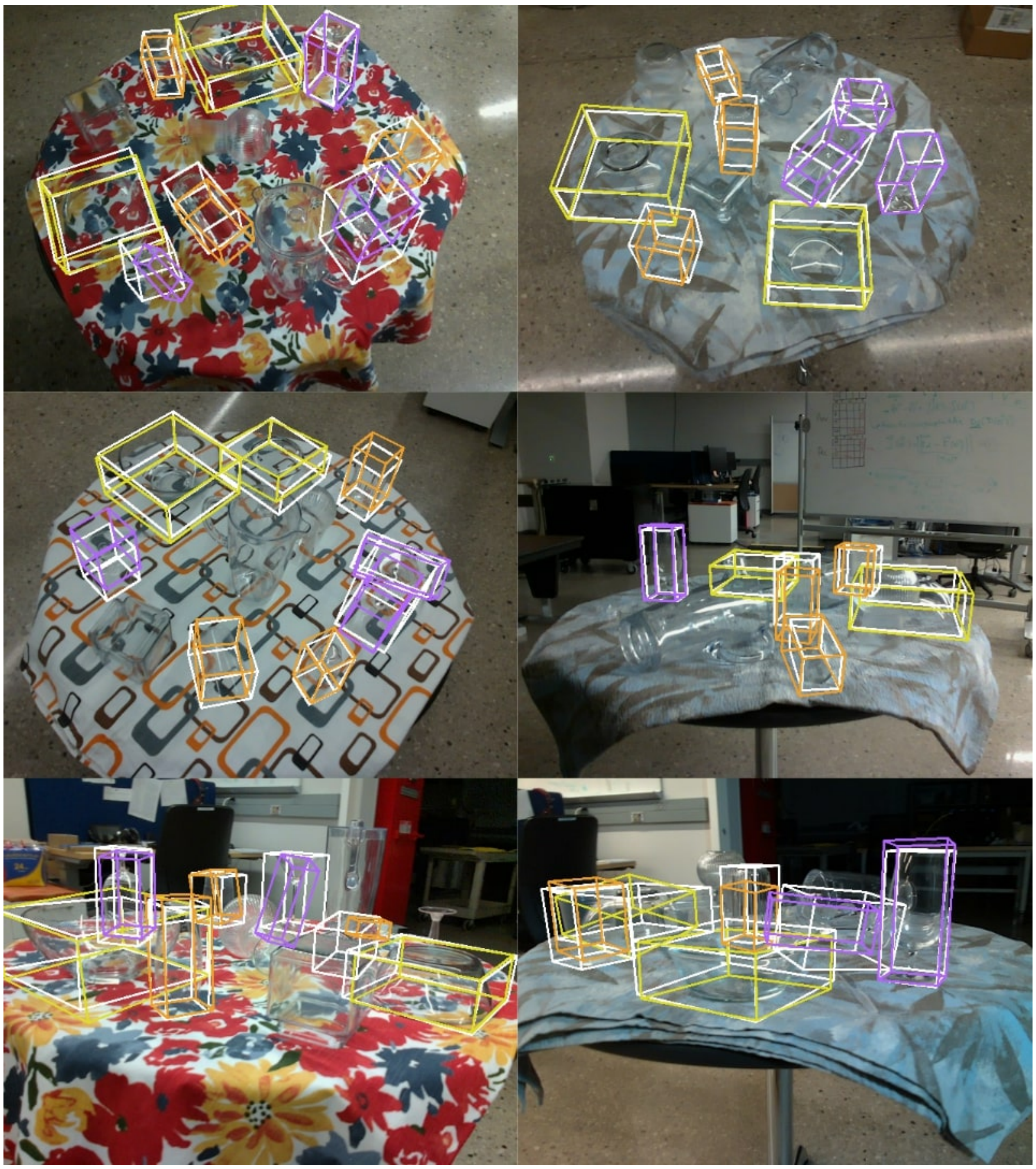


Figure 7.3: Qualitative results of category-level pose estimates from TransNet. White bounding boxes depict ground truth pose annotations, while colored boxes depict TransNet's estimates. Different colors represent different categories. The top two rows show accurate estimates, while the bottom row highlights inaccurate estimates when TransNet is confronted with heavyily cluttered scenes. Objects with no bounding box correspond to categories not used during training and evaluation. For axial symmetric objects, we use the ground truth x-axis to produce a 3D bounding box.

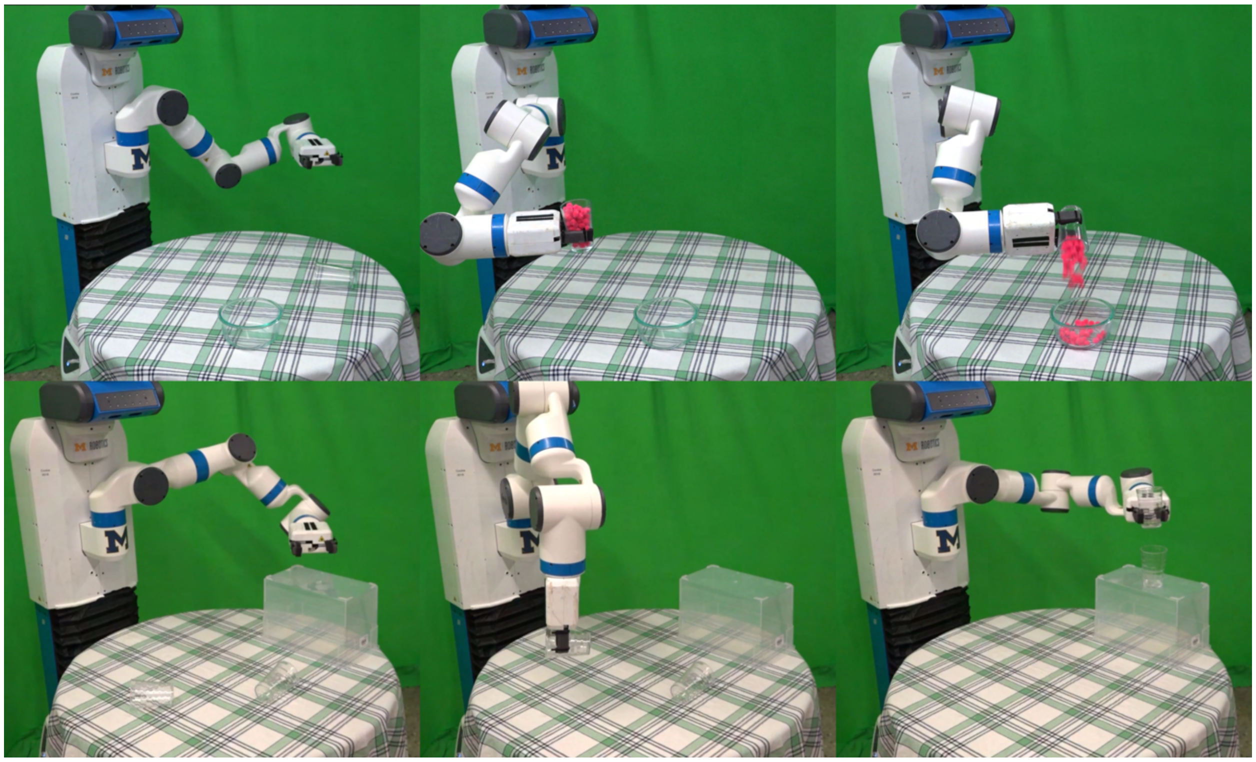


Figure 7.4: Robotic experiments. The top row is the demo of pouring using a water cup and a bowl. The bottom row is the demo of pick-and-place using two water cups. The left column is the initial state, the middle column is picking, and the right column is pouring or placing.

CHAPTER 8

Conclusion and Discussion

8.1 Conclusion

A practical manipulation robot should be able to perceive objects and accomplish tasks based on the generalizable affordance representation in complex environments, such as varied lighting conditions, object occlusions, unseen object instances, and challenging materials like transparency. In this dissertation, we focused on the aforementioned difficulties in robot perception and contribute methods and datasets to enable *robustness* to adversarial environment changes, *generalizable affordance* across object instances, and *scalablity* through large-scale data collection, when facing challenging robot manipulation scenarios.

In *GRIP*, we combined deep neural networks with probabilistic inference, to achieve fast and robust pose estimation in dark environments. Based on the adversarial YCB dataset we collected, we provide quantatitive evidence of better robustness of *GRIP* than pure neural networks. In *ACF*, we proposed generalized object representation that could support affordance-based manipulation tasks like grasp, pour, stir, etc., directly for novel object instances with recognized affordance combinations. The learned 3D keypoint and axis estimation system could generate robot manipulation trajectories and accomplish robotic tasks. In *ProgressLabeller*, we contribute a dataset generation approach using video annotation. Based on visual SLAM camera trajectory estimation and a multiview object pose annotation interface, a large-scale dataset could be annotated in a few days. We show *ProgressLabeller* could provide even better accuracy on several public datasets on object pose estimation. Further, we contribute the largest comprehensive RGB-D dataset of transparent objects, *ClearPose* with annotation from *ProgressLabeller*. *ClearPose* contains various testing scenes, such as transparent containers filled with liquid, heavy occlusion between objects, mixed opaque and transparent objects, etc. We then build a deep-learning based category-level pose estimation approach, *TransNet*, which could learn from the vast data in *ClearPose* and generalize to novel objects. Finally, we demonstrate *TransNet* could support pick-and-place as well as pouring tasks on novel transparent objects based on the affordance representation proposed in *ACF*.

8.2 Limitations

There are several limitations for the proposed dissertation that could be further refined. First, we make an assumption on the objects of interest as rigid objects. In this case, our system cannot deal with deformable objects like towels and clothes [201]. To extend the current approach to deformable object manipulation, proper object representation is necessary. However, this problem is highly illposed and still remains an active research question. Second, in GRIP, we contributed a combined discrimitive-generative perception pipeline, but later we did not further explore more advanced systems, and explored on deep-learning based approaches instead. The main reason on this point is that we cannot find a good way to accelerate current probabilistic inference approaches, for example particle filters. There are concurrent works [114, 115] focusing on hardware acceleration of such methods and they are potential improvements to the proposed perception system. Third, the way of applying estimated object poses to real robotic movements is not carefully explored in this dissertation. We demonstrated object pose-based grasping in GRIP and ProgressLabeller experiments, and pouring and stirring actions in ACF experiments, but the actions are hardcoded manually. The proposed system could be expanded to incorporate learning from human demonstration, or reinforcement learning based on object pose estimates.

8.3 Future Directions

The output of this paper could serve as a basis towards the goal of more generalized robotic sensing and acting. In general, we aim to have robots to estimate the state of arbitrary objects within novel environments, and also be able to accomplish the afforded tasks by the objects.

8.3.1 Novel Object Pose Estimation and Manipulation

From instance-level to category-level, the object pose estimation task is generalized gradually to unseen objects from the training set. Recently there are publications [116] focused on the pose estimation task over completely novel objects without a categorical prior. From the view of robotic perception and manipulation, one first step to examine and utilize the pose estimates is to do pose-based grasping over these unseen objects. Besides, how to transfer robot manipulation policy from known objects to novel objects based on their pose estimates, or design robot learning systems, could also be further explored.

8.3.2 Dense Reconstruction for Transparent Shapes

With the 3D object shape models, 2D depth measurements and camera trajectory provided in ClearPose, dense reconstruction on transparent objects become a feasible research direction. Single-view depth estimation of transparent shapes have been proved to support robotic grasping [161], yet for 3D reconstruction, most existing works require complex optical hardware or setup, e.g., active light source and light-field and polarization cameras, to recover the light rays through reflection and refraction patterns, which is extremely difficult to integrate with a robot platform. A recent work incorporates light reflection model into Neural Radiance Field (NeRF) [64], and it is possible to also consider light refraction to deal with small scale objects for robotic manipulation.

8.3.3 Affordance-grounded Robotic Task Execution

Affordance could serve as a better semantic representation than object in various robotic tasks. Currently, affordance mostly serve as a label manually annotated over objects in datasets, and they haven't covered most of daily manipulation tasks. One direction is to collect a large-scale affordance dataset including both visual representations like object detection/segmentation and keypoints, and robot dynamic motion, like trajectories. Possibly language descriptions could be annotated as well. One of our recent works [213] categorized everyday manipulation tasks by motion constraints, which is an extension of the affordance coordinate frame proposed in this paper. The work also includes a manipulation benchmark that provides language description, multi-view camera observation, and object states during the task execution. Learning approaches on category-level manipulation policy and task execution could be proposed based on this benchmark.

8.3.4 Human-factor Research in Robotic Data Annotation

ProgressLabeller [29] provides a way of annotating videos for 3D visual perception tasks, like 3D detection and 6D object pose estimation. Currently it has only been tested within a small population of researchers on 3D object pose data generation. Based on this work, human-factor research [200] could be explored focusing on improving the efficiency and user experience of this system. We could also add more features to expand this system for adaptive data annotation on dynamic scenes, non-rigid objects, or objects without known 3D CAD models.

8.4 Summary

This dissertation provides experience and insights in object-centric robotic visual perception under complex environments. One insight to deal with potential neural network errors in adversarial environments is to combine that with probabilistic generative methods. We propose GRIP as such an approach robust to lighting changes and object occlusions. Another complimentary way is to enable fast real-world dataset creation. We contribute such a pipeline, ProgressLabeller, and demonstrate its efficacy on improving end-to-end pose estimators. We also expand the applicability of the object state estimates. We propose ACF as a representation that could be used to do tasks like pouring and stirring with novel object instances. We finally integrate these factors into a ‘bartender bot’ that could perceive and manipulation transparent objects by category, and accomplish a series of tasks like pick-and-place and pouring liquid.

BIBLIOGRAPHY

[1] Agility's two-legged robot digit is for sale and ford is the first customer. `https://techcrunch.com/2020/01/05/agilitys-two-legged-robot-digit-is-for-sale-and-ford-is-the-first-cust`

[2] Boston dynamics spot video. `https://www.youtube.com/watch?v=6Zbhvaac68Y`.

[3] Ford unveils new self-driving test vehicle for 2022 launch. `https://www.cnbc.com/2020/10/20/ford-unveils-new-self-driving-test-vehicle-for-2022-launch.html`.

[4] Industrial robotics: An introduction and beginner's guide. `https://www.rg-robotics.com/industrial-robotics-an-introduction-and-beginners-guide`

[5] Pr2. `https://www.razorrobotics.com/robots/pr2/`.

[6] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7822–7831, 2021.

[7] S. Akizuki and M. Hashimoto. Asm-net: Category-level pose and shape estimation using parametric deformation. In *British Machine Vision Conference (BMVC)*, 2021.

[8] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 585–592. IEEE, 2011.

[9] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[10] L. Antanas, P. Moreno, M. Neumann, R. P. de Figueiredo, K. Kersting, J. Santos-Victor, and L. De Raedt. Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach. volume 43(6), pages 1393–1418. Springer, 2019.

[11] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[12] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, et al. Unity perception: Generate synthetic data for computer vision. *arXiv preprint arXiv:2107.04259*, 2021.

[13] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3364–3372, 2016.

[14] Y. Bukschat and M. Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach. *arXiv preprint arXiv:2011.04307*, 2020.

[15] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[16] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[17] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *arXiv preprint arXiv:1904.11492*, 2019.

[18] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, pages 7291–7299, 2017.

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

[20] J. Chang, M. Kim, S. Kang, H. Han, S. Hong, K. Jang, and S. Kang. Ghostpose*: Multi-view pose estimation of transparent objects for robot hand grasping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5749–5755. IEEE, 2021.

[21] D. Chen, J. Li, Z. Wang, and K. Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11973–11982, 2020.

[22] G. Chen, K. Han, and K.-Y. K. Wong. Tom-net: Learning transparent object matting from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9233–9241, 2018.

[23] K. Chen and Q. Dou. Sgpa: Structure-guided prior adaptation for category-level 6d object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2773–2782, 2021.

[24] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[25] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[26] W. Chen, X. Jia, H. J. Chang, J. Duan, L. Shen, and A. Leonardis. Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1581–1590, 2021.

[27] X. Chen, R. Chen, Z. Sui, Z. Ye, Y. Liu, R. I. Bahar, and O. C. Jenkins. Grip: Generative robust inference and perception for semantic robot manipulation in adversarial environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3988–3995. IEEE, 2019.

[28] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *European Conference on Computer Vision*, pages 139–156. Springer, 2020.

[29] X. Chen, H. Zhang, Z. Yu, S. Lewis, and O. C. Jenkins. Progresslabeller: visual data stream annotation for training object-centric 3d perception. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13066–13073. IEEE, 2022.

[30] X. Chen, H. Zhang, Z. Yu, A. Opipari, and O. Chadwicke Jenkins. Clearpose: Large-scale transparent object dataset and benchmark. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*, pages 381–396. Springer, 2022.

[31] X. Chen, K. Zheng, Z. Zeng, C. Kisailus, S. Basu, J. Cooney, J. Pavlasek, and O. C. Jenkins. Manipulation-oriented object perception in clutter through affordance coordinate frames. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 186–193. IEEE, 2022.

[32] H. Cheng, Y. Wang, and M. Q.-H. Meng. Grasp pose detection from a single rgb image. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4686–4691. IEEE, 2021.

[33] F.-J. Chu, R. Xu, and P. A. Vela. Learning affordance segmentation for real-world robotic manipulation via synthetic images. *IEEE Robotics and Automation Letters*, 4(2):1140–1147, 2019.

[34] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Şucan. Towards reliable grasping and manipulation in household environments. In *Experimental Robotics*, pages 241–252. Springer, 2014.

[35] A. Collet, M. Martinez, and S. S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011.

[36] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[37] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[38] N. Corporation. Nvidia isaac sim webpage. `https://developer.nvidia.com/isaac-sim`.

[39] Q. Dai, J. Zhang, Q. Li, T. Wu, H. Dong, Z. Liu, P. Tan, and H. Wang. Domain randomization-enhanced depth simulation and restoration for perceiving and grasping specular and transparent objects. *arXiv preprint arXiv:2208.03792*, 2022.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[41] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[42] A. Depierre, E. Dellandréa, and L. Chen. Jacquard: A large scale dataset for robotic grasp detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3516. IEEE, 2018.

[43] K. Desingh, O. C. Jenkins, L. Reveret, and Z. Sui. Physically plausible scene estimation for manipulation in clutter. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1073–1080. IEEE, 2016.

[44] K. Desingh, S. Lu, A. Opipari, and O. C. Jenkins. Efficient nonparametric belief propagation for pose estimation and manipulation of articulated objects. *Science Robotics*, 4(30), 2019.

[45] R. Detry, J. Papon, and L. Matthies. Task-oriented grasping with semantic and geometric scene understanding. In *IROS*, pages 3266–3273. IEEE, 2017.

[46] J. G. C. Devol. Programmed article transfer, June 13 1961. US Patent 2,988,237.

[47] Y. Di, R. Zhang, Z. Lou, F. Manhardt, X. Ji, N. Navab, and F. Tombari. Gpv-pose: Category-level object pose estimation via geometry-guided point-wise voting. *arXiv preprint arXiv:2203.07918*, 2022.

[48] T.-T. Do, A. Nguyen, and I. Reid. AffordanceNet: An end-to-end deep learning approach for object affordance detection. In *ICRA*, pages 1–5. IEEE, 2018.

[49] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3583–3592, 2016.

[50] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems*, 27, 2014.

[51] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1:1, 2017.

[52] Z. Fan, Z. Song, J. Xu, Z. Wang, K. Wu, H. Liu, and J. He. Acr-pose: Adversarial canonical representation reconstruction network for category level 6d object pose estimation. *arXiv preprint arXiv:2111.10524*, 2021.

[53] H. Fang, H.-S. Fang, S. Xu, and C. Lu. Transcg: A large-scale real-world dataset for transparent object depth completion and grasping. *arXiv preprint arXiv:2202.08471*, 2022.

[54] H.-S. Fang, C. Wang, M. Gou, and C. Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020.

[55] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[56] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[57] W. Gao and R. Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE RA-L*, 2021.

[58] J. J. Gibson. The theory of affordances. *Hilldale, USA*, 1(2):67–82, 1977.

[59] J. J. Gibson and L. Carmichael. *The senses considered as perceptual systems*, volume 2(1). Houghton Mifflin Boston, 1966.

[60] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[61] S. R. Gouravajhala, J. Yim, K. Desingh, Y. Huang, O. C. Jenkins, and W. S. Lasecki. Eureca: Enhanced understanding of real environments via crowd assistance. In *Sixth AAAI conference on human computation and crowdsourcing*, 2018.

[62] M. Gualtieri, A. ten Pas, and R. Platt. Pick and place without geometric object models. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7433–7440. IEEE, 2018.

[63] R. A. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018.

[64] Y.-C. Guo, D. Kang, L. Bao, Y. He, and S.-H. Zhang. Nerfren: Neural radiance fields with reflections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18409–18418, 2022.

[65] A. Hämäläinen, K. Arndt, A. Ghadirzadeh, and V. Kyrki. Affordance learning for end-to-end visuomotor robot control. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1781–1788. IEEE, 2019.

[66] S. Hart, P. Dinh, and K. Hambuchen. The affordance template ROS package for robot task programming. In *ICRA*, pages 6227–6234. IEEE, 2015.

[67] S. Hart, A. H. Quispe, M. W. Lanighan, and S. Gee. Generalized affordance templates for mobile manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6240–6246. IEEE, 2022.

[68] M. Hassanin, S. Khan, and M. Tahtali. Visual affordance and function understanding: A survey. *ACM Computing Surveys (CSUR)*, 54(3):1–35, 2021.

[69] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[70] R. L. Haugaard and A. G. Buch. Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings. *arXiv preprint arXiv:2111.13489*, 2021.

[71] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[72] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[73] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013, 2021.

[74] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. PVN3D: A deep point-wise 3D keypoints voting network for 6dof pose estimation. In *CVPR*, pages 11632–11641, 2020.

[75] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[76] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. Going further with point pair features. In *European conference on computer vision*, pages 834–848. Springer, 2016.

[77] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017.

[78] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas. Bop challenge 2020 on 6d object localization. In *European Conference on Computer Vision*, pages 577–594. Springer, 2020.

[79] X. Hong, P. Xiong, R. Ji, and H. Fan. Deep fusion network for image completion. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2033–2042, 2019.

[80] D. Q. Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[81] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.

[82] M. Z. Irshad, T. Kollar, M. Laskey, K. Stone, and Z. Kira. Centersnap: Single-shot multi-object 3d shape reconstruction and categorical 6d pose and size estimation. *arXiv preprint arXiv:2203.01929*, 2022.

[83] X. Jiang, D. Li, H. Chen, Y. Zheng, R. Zhao, and L. Wu. Uni6d: A unified cnn framework without projection breakdown for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11174–11184, 2022.

[84] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[85] D. Joho, G. D. Tipaldi, N. Engelhard, C. Stachniss, and W. Burgard. Nonparametric bayesian models for unsupervised scene analysis and reconstruction. *Robotics*, page 161, 2013.

[86] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[87] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[88] P. Kaiser, E. E. Aksoy, M. Grotz, and T. Asfour. Towards a hierarchy of loco-manipulation affordances. In *2016 IEEE/RSJ IROS*, pages 2839–2846. IEEE, 2016.

[89] P. Kaiser and T. Asfour. Autonomous detection and experimental validation of affordances. *IEEE RA-L*, 3(3):1949–1956, 2018.

[90] A. Kalra, V. Taamazyan, S. K. Rao, K. Venkataraman, R. Raskar, and A. Kadambi. Deep polarization cues for transparent object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8602–8611, 2020.

[91] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[92] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pages 1521–1529, 2017.

[93] M. P. Khaing and M. Masayuki. Transparent object detection using convolutional neural network. In *International Conference on Big Data Analysis and Deep Learning Applications*, pages 86–93. Springer, 2018.

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

[95] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic. Affordance detection for task-specific grasping using deep learning. In *Humanoids*, pages 91–98. IEEE, November 2017.

[96] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[97] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[98] M. Krogius, A. Haggenmiller, and E. Olson. Flexible layouts for fiducial tags. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2019.

[99] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 954–962, 2015.

[100] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.

[101] P.-J. Lai and C.-S. Fuh. Transparent object detection using regions with convolutional neural network. In *IPPR Conference on Computer Vision, Graphics, and Image Processing*, volume 2, 2015.

[102] T. Lee, B.-U. Lee, M. Kim, and I. S. Kweon. Category-level metric scale object shape and pose estimation. *IEEE Robotics and Automation Letters*, 6(4):8575–8582, 2021.

[103] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[104] X. Li, H. Wang, L. Yi, L. J. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3706–3715, 2020.

[105] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[106] J. Lin, Z. Wei, Z. Li, S. Xu, K. Jia, and Y. Li. Dualposenet: Category-level 6d object pose and size estimation using dual pose network with refined learning of pose consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3560–3569, 2021.

[107] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[108] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. *Neural Information Processing Systems (NeurIPS)*, 2020.

[109] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[110] W. Liu, A. Daruna, and S. Chernova. Cage: Context-aware grasping engine. 2019.

[111] X. Liu, S. Iwase, and K. M. Kitani. Stereobj-1m: Large-scale stereo image dataset for 6d object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10870–10879, 2021.

[112] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige. Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11602–11610, 2020.

[113] Y. Liu, A. Costantini, R. Bahar, Z. Sui, Z. Ye, S. Lu, and O. C. Jenkins. Robust object estimation using generative-discriminative inference for secure robotics applications. In *Proceedings of the International Conference on Computer-Aided Design*, page 75. ACM, 2018.

[114] Y. Liu, C. E. Derman, G. Calderoni, and R. I. Bahar. Hardware acceleration of robot scene perception algorithms. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2020.

[115] Y. Liu, A. Opipari, T. Guerin, and R. I. Bahar. Hardware acceleration of nonparametric belief propagation for efficient robot manipulation. In *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 51–51, 2022.

[116] Y. Liu, Y. Wen, S. Peng, C. Lin, X. Long, T. Komura, and W. Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. *arXiv preprint arXiv:2204.10776*, 2022.

[117] Z. Liu, D. Chen, K. M. Wurm, and G. von Wichert. Table-top scene analysis using knowledge-supervised mcmc. *Robotics and Computer-Integrated Manufacturing*, 33:110–123, 2015.

[118] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[119] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[120] J. Luh. An anatomy of industrial robots and their controls. *IEEE Transactions on Automatic Control*, 28(2):133–153, 1983.

[121] I. Lysenkov, V. Eruhimov, and G. Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. *Robotics*, 273(273-280):2, 2013.

[122] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on robotics and automation (ICRA)*, pages 5620–5627. IEEE, 2018.

[123] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019.

[124] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 616–625, 2018.

[125] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.

[126] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3235–3242. IEEE, 2018.

[127] S. J. Mckenna and H. Nait-Charif. Tracking human motion using auxiliary particle filters and iterated likelihood weighting. *Image Vision Comput.*, 25:852–862, 2007.

[128] T. McMahon, O. C. Jenkins, and N. Amato. Affordance wayfields for task and motion planning. In *IROS*, pages 2955–2962. IEEE, 2018.

[129] N. Mellado, D. Aiger, and N. J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33(5), pages 205–215. Wiley Online Library, 2014.

[130] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global hypothesis generation for 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2017.

[131] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[132] C. Mitash, A. Boularias, and K. Bekris. Robust 6d object pose estimation with stochastic congruent sets. *arXiv preprint arXiv:1805.06324*, 2018.

[133] C. Mitash, A. Boularias, and K. E. Bekris. Improving 6d pose estimation of objects in clutter via physics-aware monte carlo tree search. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3331–3338. IEEE, 2018.

[134] S. Mohan, A. H. Mininger, J. R. Kirk, and J. E. Laird. Acquiring grounded representations of words with situated interactive instruction. In *Advances in Cognitive Systems*. Citeseer, 2012.

[135] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[136] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA*, pages 1374–1381. IEEE, 2015.

[137] V. Narayanan and M. Likhachev. Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances. In *Robotics: Science and Systems*, 2016.

[138] V. Narayanan and M. Likhachev. Perch: Perception via search for multi-object recognition and localization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5052–5059. IEEE, 2016.

[139] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011.

[140] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *IROS*, pages 5908–5915. IEEE, 2017.

[141] S. Y. Nof. *Handbook of industrial robotics*. John Wiley & Sons, 1999.

[142] O. of Louisiana Entertainment. Turbosquid main page. `https://www.turbosquid.com/`.

[143] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011.

[144] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka. Rigid 3d geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research*, 31(4):538–553, 2012.

[145] K. Park, A. Mousavian, Y. Xiang, and D. Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10710–10719, 2020.

[146] J. Pavlasek, S. Lewis, K. Desingh, and O. C. Jenkins. Parts-based articulated object localization in clutter using belief propagation. In *IROS*. IEEE, 2020.

[147] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[148] C. J. Phillips, M. Lecce, and K. Daniilidis. Seeing glassware: from edge detection to pose estimation and shape recovery. In *Robotics: Science and Systems*, volume 3, page 3, 2016.

[149] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. 2019.

[150] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7278–7285. IEEE, 2020.

[151] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017.

[152] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[153] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[154] E. Rohmer, S. P. Singh, and M. Freese. Coppeliasim (formerly V-REP): a versatile and scalable robot simulation framework. In *IROS*, 2013.

[155] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015.

[156] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[157] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[158] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[159] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. IEEE, 2010.

[160] E. Şahin, M. Cakmak, M. R. Doğar, E. Uğur, and G. Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472, 2007.

[161] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3642. IEEE, 2020.

[162] G. Sánchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics research*, pages 403–417. Springer, 2003.

[163] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[164] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016.

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

[166] R. P. Singh, M. Benallegue, Y. Yoshiyasu, and F. Kanehiro. Rapid pose label generation through sparse representation of unknown objects. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10287–10293. IEEE, 2021.

[167] J. Song, A. Tanwani, J. Ichnowski, M. Danielczuk, K. Sanders, J. Chui, J. A. Ojea, and K. Goldberg. Robust task-based grasping as a service. In *CASE*, pages 22–28. IEEE, 2020.

[168] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[169] M. R. Stevens and J. R. Beveridge. Localized scene interpretation from 3d models, range, and optical data. *Computer Vision and Image Understanding*, 80(2):111–129, 2000.

[170] D. Stumpf, S. Krauß, G. Reis, O. Wasenmüller, and D. Stricker. Salt: A semi-automatic labeling tool for rgb-d video sequences. *arXiv preprint arXiv:2102.10820*, 2021.

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

[172] Z. Sui, O. C. Jenkins, and K. Desingh. Axiomatic particle filtering for goal-directed robotic manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4429–4436. IEEE, 2015.

[173] Z. Sui, Z. Zhou, Z. Zeng, and O. C. Jenkins. Sum: Sequential scene understanding and manipulation. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3281–3288. IEEE, 2017.

[174] J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Robotics Research*, pages 701–720. Springer, 2018.

[175] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021.

[176] Y. Tang, J. Chen, Z. Yang, Z. Lin, Q. Li, and W. Liu. Depthgrasp: Depth completion of transparent objects using self-attentive adversarial network with spectral residual for grasping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5710–5716. IEEE, 2021.

[177] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[178] A. Ten Pas and R. Platt. Localizing handle-like grasp affordances in 3d point clouds. In *Experimental Robotics*, pages 623–638. Springer, 2016.

[179] M. Tian, M. H. Ang, and G. H. Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020.

[180] M. Tian, L. Pan, M. H. Ang, and G. H. Lee. Robust 6d object pose estimation by learning rgb-d features. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6224. IEEE, 2020.

[181] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield. NDDS: NVIDIA deep learning dataset synthesizer, 2018. `https://github.com/NVIDIA/Dataset_Synthesizer`.

[182] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[183] S. Tyree, J. Tremblay, T. To, J. Cheng, T. Mosier, J. Smith, and S. Birchfield. Household objects for pose estimation (hope), 2022.

[184] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[185] K. M. Varadarajan and M. Vincze. Afnet: The affordance network. In *Asian Conference on Computer Vision*, pages 512–523. Springer, 2012.

[186] K. M. Varadarajan and M. Vincze. Afrob: The affordance network ontology for robots. In *IROS*, pages 1343–1350. IEEE, 2012.

[187] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[188] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[189] T. Weng, A. Pallankize, Y. Tang, O. Kroemer, and D. Held. Multi-modal transfer learning for grasping transparent and specular objects. *IEEE Robotics and Automation Letters*, 5(3):3791–3798, 2020.

[190] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[191] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE international conference on robotics and biomimetics*, pages 2987–2992. IEEE, 2011.

[192] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics Science and Systems*, 2018.

[193] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *RSS*, 2018.

[194] E. Xie, W. Wang, W. Wang, M. Ding, C. Shen, and P. Luo. Segmenting transparent objects in the wild. In *European Conference on Computer Vision*, pages 696–711. Springer, 2020.

[195] C. Xu, J. Chen, M. Yao, J. Zhou, L. Zhang, and Y. Liu. 6dof pose estimation of transparent object from a single rgb-d image. *Sensors*, 20(23):6790, 2020.

[196] H. Xu, Y. R. Wang, S. Eppel, A. Aspuru-Guzik, F. Shkurti, and A. Garg. Seeing glass: Joint point cloud and depth completion for transparent objects. *arXiv preprint arXiv:2110.00087*, 2021.

[197] M. Xu, P. Chen, H. Liu, and X. Han. To-scene: A large-scale dataset for understanding 3d tabletop scenes. In *ECCV*, 2022.

[198] R. Xu, F.-J. Chu, C. Tang, W. Liu, and P. Vela. An affordance keypoint detection network for robot manipulation. *IEEE RA-L*, 2021.

[199] Y. Xu, H. Nagahara, A. Shimada, and R.-i. Taniguchi. Transcut: Transparent object segmentation from a light-field image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3442–3450, 2015.

[200] Z. Ye, J. Y. Song, Z. Sui, S. Hart, J. Vilchis, W. S. Lasecki, and O. C. Jenkins. Human-in-the-loop pose estimation via shared autonomy. In *26th International Conference on Intelligent User Interfaces*, pages 387–391, 2021.

[201] H. Yin, A. Varava, and D. Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803, 2021.

[202] H. Yong, J. Huang, X. Hua, and L. Zhang. Gradient centralization: A new optimization technique for deep neural networks. In *European Conference on Computer Vision*, pages 635–652. Springer, 2020.

[203] Y. You, R. Shi, W. Wang, and C. Lu. Cppf: Towards robust category-level 9d pose estimation in the wild. *arXiv preprint arXiv:2203.03089*, 2022.

[204] A. R. Zamir, A. Sax, N. Cheerla, R. Suri, Z. Cao, J. Malik, and L. J. Guibas. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206, 2020.

[205] P. Zech, S. Haller, S. R. Lakani, B. Ridge, E. Ugur, and J. Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271, 2017.

[206] Z. Zeng, P. S. Joshi, and O. C. Jenkins. Unsupervised learning of affordance coordinate frame for robotic task generalization. In *ICRA workshop: 2nd International Workshop on Computational Models of Affordance in Robotics*, 2019.

[207] Z. Zeng, Z. Zhou, Z. Sui, and O. C. Jenkins. Semantic robot programming for goal-directed manipulation in cluttered scenes. In *ICRA*, pages 7462–7469. IEEE, 2018.

[208] Z. Zeng, Z. Zhou, Z. Sui, and O. C. Jenkins. Semantic robot programming for goal-directed manipulation in cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7462–7469. IEEE, 2018.

[209] H. Zhang, A. Opipari, X. Chen, J. Zhu, Z. Yu, and O. C. Jenkins. Transnet: Category-level transparent object pose estimation. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*, pages 148–164. Springer, 2023.

[210] J. Zhang and S. Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.

[211] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in Neural Information Processing Systems*, 32, 2019.

[212] Y. Zhang and T. Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018.

[213] K. Zheng, X. Chen, O. C. Jenkins, and X. E. Wang. Vlmbench: A compositional benchmark for vision-and-language manipulation. *Neural Information Processing Systems (NeurIPS)*, 2022.

[214] Z. Zhou, X. Chen, and O. C. Jenkins. Lit: Light-field inference of transparency for refractive object localization. *IEEE Robotics and Automation Letters*, 5(3):4548–4555, 2020.

[215] Z. Zhou, T. Pan, S. Wu, H. Chang, and O. C. Jenkins. Glassloc: Plenoptic grasp pose detection in transparent clutter. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4776–4783. IEEE, 2019.

[216] Z. Zhou, Z. Sui, and O. C. Jenkins. Plenoptic monte carlo object localization for robot grasping under layered translucency. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[217] L. Zhu, A. Mousavian, Y. Xiang, H. Mazhar, J. van Eenbergen, S. Debnath, and D. Fox. Rgb-d local implicit function for depth completion of transparent objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4649–4658, 2021.

[218] L. Zou, Z. Huang, N. Gu, and G. Wang. 6d-vit: Category-level 6d object pose estimation via transformer-based instance representation learning. *arXiv preprint arXiv:2110.04792*, 2021.