Experimental Evaluation of Affordance Detection Applied to 6-DoF Pose Estimation for Intelligent Robotic Grasping of Household Objects

by

Aidan Keaveny

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Recent computer vision research has demonstrated that deep convolutional neural networks can be trained on real images to add context to object parts for simultaneous object detection and affordance segmentation. However, generating such a dataset with expensive hand annotations for pixel-wise labels presents a challenge for training deep convolutional neural networks. In this thesis, a method to automate dataset generation of real and synthetic images with ground truth annotations for affordance detection and object part 6-DoF pose is presented. A variant of Mask R-CNN is implemented and trained to perform affordance detection and integrated within DenseFusion, a two-stage framework for 6-DoF pose estimation. The primary contribution of this work is to experimentally evaluate 6-DoF pose estimation with object segmentation and affordance detection, which was done on the YCB-Video benchmark dataset and the ARL AffPose dataset. It was demonstrated that 6-DoF pose estimation with object segmentation slightly outperforms pose estimation with affordance detection, as the latter operates on a subset of RGB-D data. However, the advantage of pose estimation with affordance detection is realized when the trained model is deployed on a robotic platform to grasp complex objects, such that an 11% improvement in terms of grasp success rate was experimentally demonstrated for a power drill.

Acknowledgements

I would first like to thank my supervisors, Professor Soo Jeon and Professor William Melek, for their patience and encouragement throughout my masters as I began my journey without a background in Robotics.

Many thanks to Professor Bryan Tripp for lending me YCB objects to generate our dataset.

I would also like to praise my lab mates, Somesh Daga and Shiyi Yang, for their support in using our robotic platform and those at the UW Robotics Team for guidance navigating the field of robotics. I am grateful to have shared my time at the University of Waterloo with many of my colleagues, I have learnt so much from all of you.

I would also like to thank my friends and family, who have always loved and supported me throughout my journey in life. I would not be where I am today without you. A special thanks to those I lived with, Daniel Sola, Andrew Downie and Matthew Cann, for many insights, discussions and laughs throughout the COVID-19 pandemic.

Lastly, I acknowledge the support of the Ontario Graduate Scholarship (OGS) Program and the University of Waterloo, which made my academic journey possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

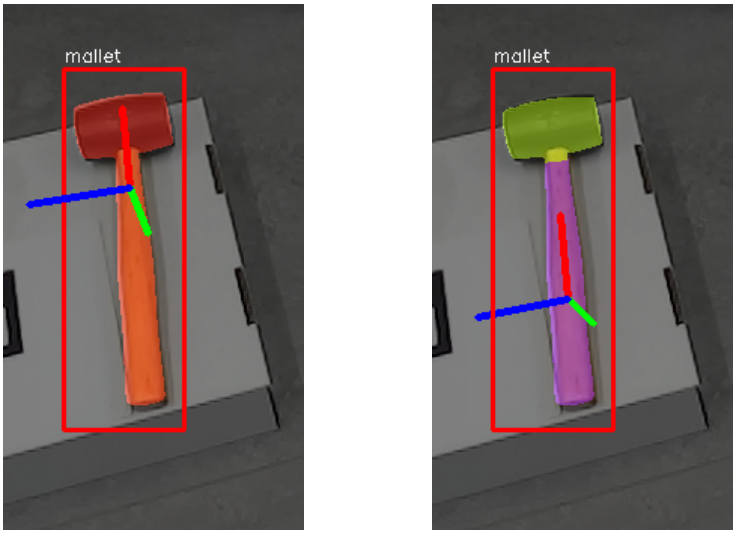
Introduction

Perception and manipulation is trivial for many humans as prior experience can be leveraged to help one understand the contents of a scene (i.e. objects of interest) as well as the functionalities of certain objects (i.e. affordances). However, autonomous agents are currently limited to simplified and structured environments to allow for successful completion of pick and place tasks. To this end, methods for scene understanding applied to 6-DoF pose estimation in the context of robotic grasping are reviewed in this thesis. Fast and robust 6-DoF pose estimation is essential for any autonomous agent to physically interact with its environment, which has been exemplified by the Amazon Picking Challenge [1]. Accurate pose estimation also has widespread applications in domains such as augmented reality [2] and autonomous driving [3, 4].

Affordance detection [5] is highly useful for determining what actions can be performed on which object(s) where in a given scene. Recent computer vision research has demonstrated that deep Convolutional Neural Networks (CNN), such as AffordanceNet [6], can be trained on real images to add context to object parts for simultaneous object detection and affordance segmentation. As such, the different functionalities of an object should be identified and localized through feature learning techniques (CNN-based models), which then provides context to estimate the Six Degrees of Freedom (6-DoF) pose of an object (Fig. 1.1). Note that the right hand coordinate (RHS) system is used throughout this thesis with a consistent colour scheme such that the x-axis is in red, y-axis is in green and z-axis is in blue.

One limitation of current state-of-the-art frameworks for 6-DoF pose estimation, such as DenseFusion [7], is that 6-DoF pose is estimated to the centroid of the 3D bounding volume that contains an object. In the context of robotic grasping, this information may lead to a failed grasp attempt if the centroid of the 3D bounding volume does not lie on

the object itself. Thus, performance for robotic grasping can be improved if 6-DoF pose is estimated to object parts that are well suited for grasping, such as the handle of a mallet (Fig. 1.1).



(a) Object Segmentation (b) Affordance Detection

Figure 1.1: Comparison of Object Segmentation and Affordance Detection. The colour of the mask represents the affordance label. Grasp: purple; pound: light green.

However, deploying Deep Learning (DL) frameworks to applications of interest is challenging as data driven approaches require large datasets with a wide variety of high quality images. Recent work [8] for detecting object affordances has focused efforts at a pixel-wise level, which requires labor-intensive, manual annotations. Such an approach tends to induce bias into a trained model(s) as one overfits to a tiny subset of objects in the real-world. Generalization with DL remains as an issue, but one could supplement real images with recent trends to generate synthetic images [9], which allows for environments that can be easily altered in terms of background scenes, lighting conditions, etc.

In this thesis, real and synthetic images are leveraged to generate a large dataset to evaluate performance for 6-DoF pose estimation with object segmentation and affordance segmentation. The foundation of this work is based on Mask R-CNN [10] for object segmentation, AffordanceNet [6] for affordance detection and DenseFusion [7] for 6-DoF pose estimation. Lastly, a ROS node was developed and deployed to experimentally validate two pipelines for pose estimation on a robotic platform using a 7-DoF Barrett WAM arm and 8-DoF BarrettHand.

1.1 Contributions

The primary contribution of this work is an experimental evaluation of performance for 6-DoF pose estimation with object segmentation and affordance detection. In order to do so, a post-processing technique was developed to generate desired ground truth annotations, namely affordance labels and object part 6-DoF poses, from object 6-DoF poses. The post-processing technique was first applied to the YCB-Video dataset [11], which is a benchmark dataset for 6-DoF pose estimation. Similarly, the post-processing technique was also applied to a dataset generated in this thesis, which contains over 100K real and synthetic images with ground truth annotations for object segmentation and 6-DoF pose estimation. Note that all annotations were auto-generated using 3D object mesh files. To the best of our knowledge [8, 12], this is the first dataset with ground truth affordance labels for robotic manipulation that was not generated from manual, pixel-wise annotations. The contributions can be summarized as follows:

- Experimental evaluation of performance for 6-DoF pose estimation with object segmentation and affordance detection.
- A post-processing technique to generate ground truth annotations for affordance labels and object part 6-DoF poses from object 6-DoF poses.
- A dataset which contains over 100K real and synthetic images with 11 different household objects and 9 different affordance labels.

1.2 Outline

The remainder of this thesis is organized as follows:

- Chapter 2 provides insight to previous works for 6-DoF pose estimation, namely classical methods and modern frameworks that leverage DL. The affordance detection problem is later formulated and contrasted against object segmentation. This chapter ends with a discussion on benchmark datasets for affordance detection or 6-DoF pose estimation that leverage real, synthetic, or a mixture of real and synthetic images.
- Chapter 3 outlines a method for generating desired ground truth annotations, namely affordance labels and object part 6-DoF poses, from object 6-DoF poses. A dataset with real and synthetic images is also auto-generated, which contains ground truth annotations for object segmentation and 6-DoF pose.

- Next, in Chapter 4, performance for 6-DoF pose estimation with object segmentation and affordance detection is evaluated on real, synthetic and a mixture of real and synthetic images.
- Chapter 5 extends an experimental evaluation as trained models are deployed on a robotic platform for grasping experiments.
- Lastly, Chapter 6 provides a summary of this work as well as a discussion of shortcomings and directions for future work.

Chapter 2

Literature Review

This chapter outlines classical techniques for 6-DoF pose estimation, which lays the foundation to better understand more modern techniques that leverage DL. The pinhole camera model is described first as it outlines the mathematical relationship for projecting points in 3D space to the image plane. Generally, classical techniques for 6-DoF pose estimation obtain the estimated pose from a single image by computing a relationship between coordinates in 3D space and the image plane. Approaches that use 3D point cloud data, such as Iterative Closest Point (ICP), take advantage of a 3D model to compute a transform between the scene (target) and the model (source) using registration algorithms. This chapter ends with a discussion on DL frameworks for 6-DoF pose estimation and affordance detection as well as real and synthetic datasets for training such networks.

2.1 Pinhole Camera Model

Image formation for the pinhole camera model requires a projection of 3D points in the camera frame F_C onto the image plane F_i (Fig. 2.1). The focal point is the point at which all rays pass through the camera origin, the principal axis z extends from the camera origin perpendicular to the image plane, and the principal point lies at the intersection of the principal axis and the image plane. Horizon f_u and vertical f_y focal lengths are represented as the horizontal and vertical distance in terms of image plane pixels between the camera center and image plane.

If a ray is followed from the camera origin or focal point, similar triangles can be used to show that a 3D point $P = [X, Y, Z]$ in the camera coordinate frame can be projected

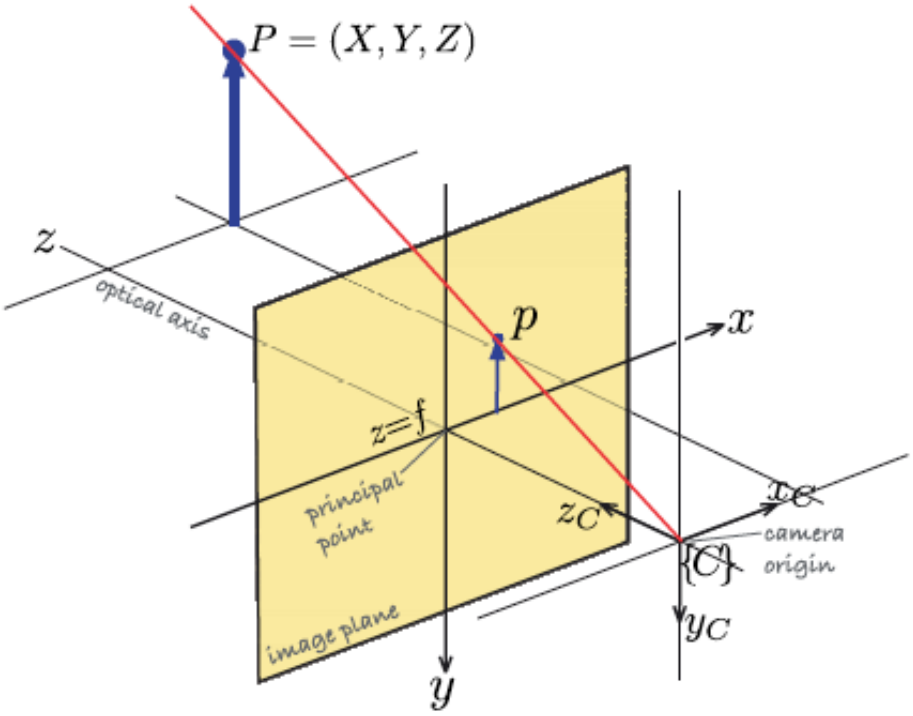


Figure 2.1: The Pinhole Camera Model taken from [13]

onto a 2D point $p = (u, v)$ on the image plane (Fig. 2.1). Horizontal u_0 and vertical v_0 offsets can be added if the image coordinate frame is not at the principal point:

$$p = [u, v] = \left[f_u \frac{X}{Z} + u_0, f_v \frac{Y}{Z} + v_0 \right] \tag{2.1}$$

It is convention to define the origin of non-negative pixel coordinates (u, v) at the top-left hand corner of the image plane (Fig. 2.2). Thus, two translation vectors c_x, c_y can be used to offset points on the image plane, which have their origin defined at the principal point. Equation 2.1 can be rewritten as a projection matrix or camera calibration matrix K using homogeneous coordinates:

$$p_h = \underbrace{\begin{bmatrix} f_u & 0 & c_x \\ 0 & f_v & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} [I|0] P_h \tag{2.2}$$

2.2 6-DoF Pose Estimation

The above discussion on the projection of an object 3D point cloud to the image plane is only valid if the point cloud exists in the camera frame F_C. The camera's extrinsic parameters, or 6-DoF pose $T \in SE(3)$, which is composed of a rotation matrix (roll, pitch and yaw) in $R \in SO(3)$ and a translation vector $t \in \mathbb{R}^3$, is required to transform the object point cloud from the world frame F_W to the camera frame F_C. Thus, given an object point cloud in the world frame F_W, the point cloud in the camera frame F_C projected onto the image plane can be computed as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsics } K} \underbrace{\begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Extrinsics } T = [R|t]} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

This thesis is focused on robust methods to estimate the object pose with respect to the camera frame F_C from a single image. The camera intrinsic matrix K can be computed using existing camera calibration methods such as OpenCV's checkerboard based method. It is assumed that the 3D model of the object is available and that the canonical frame of the object is defined in the world frame F_W.

Issues with robust 6-DoF pose estimation arise from cluttered scenes, occlusion, and varying lighting conditions [7, 11]. Objects that are rotationally symmetric about an axis, such as a bowl, are also challenging as different orientations can generate identical observations.

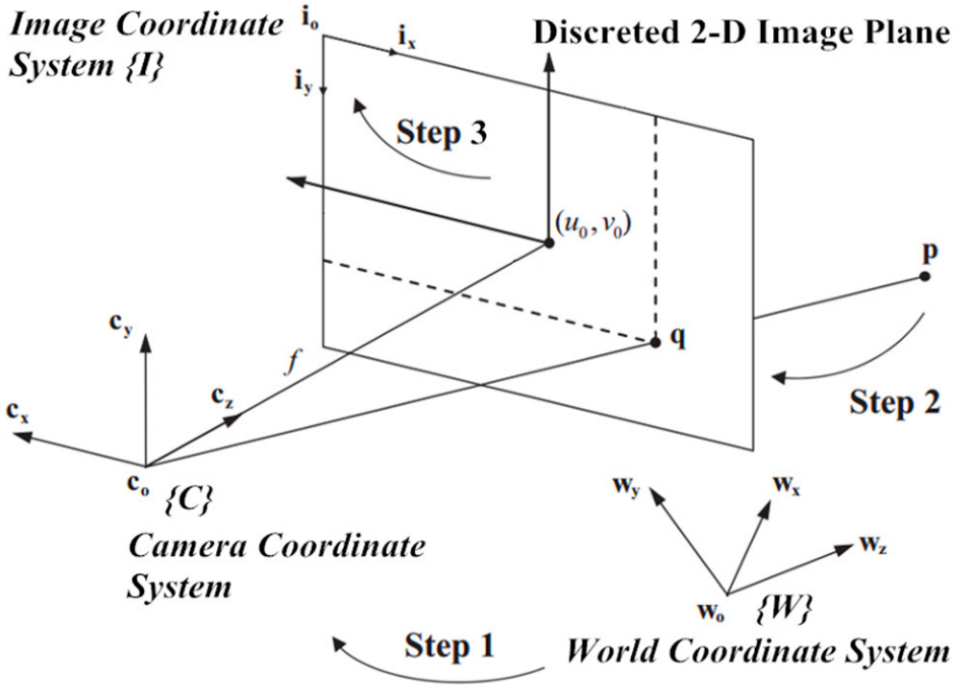


Figure 2.2: Overview of 6-DoF Pose Estimation taken from [14]

2.3 Classical Pose Estimation

Classical methods for 6-DoF pose estimation can be broadly classified as template-based methods [15, 16, 17] or feature-based methods [7, 11, 18, 19, 20].

2.3.1 Template Matching

For template-based methods, a template is first obtained by rendering a 3D model, which is then used to compute correspondences between the scene (target) and the model (source) point clouds. At each location in the image, correspondences are then scored using a similarity function, such as gradient orientation [15] or surface normals [16], and the pose hypothesis with the best score is selected. Template-based methods are useful in detecting texture-less objects but sensitive to large changes in appearance, as the template will have low similarity scores. Lastly, template-based methods are either sampling based or iterative in nature, which hinders real-time applications.

2.3.2 Perspective-n-Point

For feature-based methods, local features are extracted from keypoints. Classical methods for pose estimation, which use feature extractors such as Scale Invariant Feature Transform (SIFT) [21] or Speeded-Up Robust Features (SURF) [22], rely on detecting and matching keypoints with known object models. Thus, the Perspective-n-Point (PnP) problem can be formulated as given a set of 3D object points O_i in the world frame F_W along with their 2D projection o_i onto the image plane, the object pose $[R|t]$ is estimated with respect to a calibrated camera (Fig. 2.3).

PnP is a well studied problem in computer vision and is formulated as a non-linear least squares minimization [14]. The classical solution outlined in [23] requires 3 corresponding points (i.e. P3P), which is sensitive to noise. Accuracy for PnP can be improved by including more than 3 correspondences (i.e. EPnP), which expresses multiple points as a weighted sum of four virtual control points. Random Sample Consensus (RANSAC) [24] is also commonly used for outlier rejection with EPnP, which repeats the algorithm on different correspondences and selects the pose with the smallest square error. That said, implementing EPnP with RANSAC would be too slow for real-time deployment. A comprehensive review of PnP algorithms is outlined in [14].

In practice, a registration step is also required prior to deploying PnP, to determine n correspondences between features on the image plane and points from an object 3D model in the world frame F_W. Registration can be done offline with a known ground truth pose of the object, as outlined in OpenCV's tutorial for real Time pose estimation of a textured object.

PnP is still widely used in many deep learning frameworks for 6-DoF pose estimation today [11, 18, 19] but feature extractors swap classical algorithms, such as SIFT [21] or SURF [22], for more powerful and robust CNNs. PnP often fails to match noisy keypoints, which leads to wrong data associations between 2D and 3D pairs, when the reference image is different to the current frame.

2.3.3 Iterative Closest Point

Iterative Closest Point (ICP) is a registration algorithm often used to align two sets of point clouds in a given sensor frame [25]. As such, ICP find correspondences between a scene point cloud and a 3D model by first computing and later matching distinct features on the two sets of point clouds (Fig. 2.4). Note that an initial pose estimate $[\hat{R}|\hat{t}]$ is required to compute the error distance for each sampled point in (2.4). The iterative process of finding correspondences and computing the error distance is repeated until a criterion is satisfied.

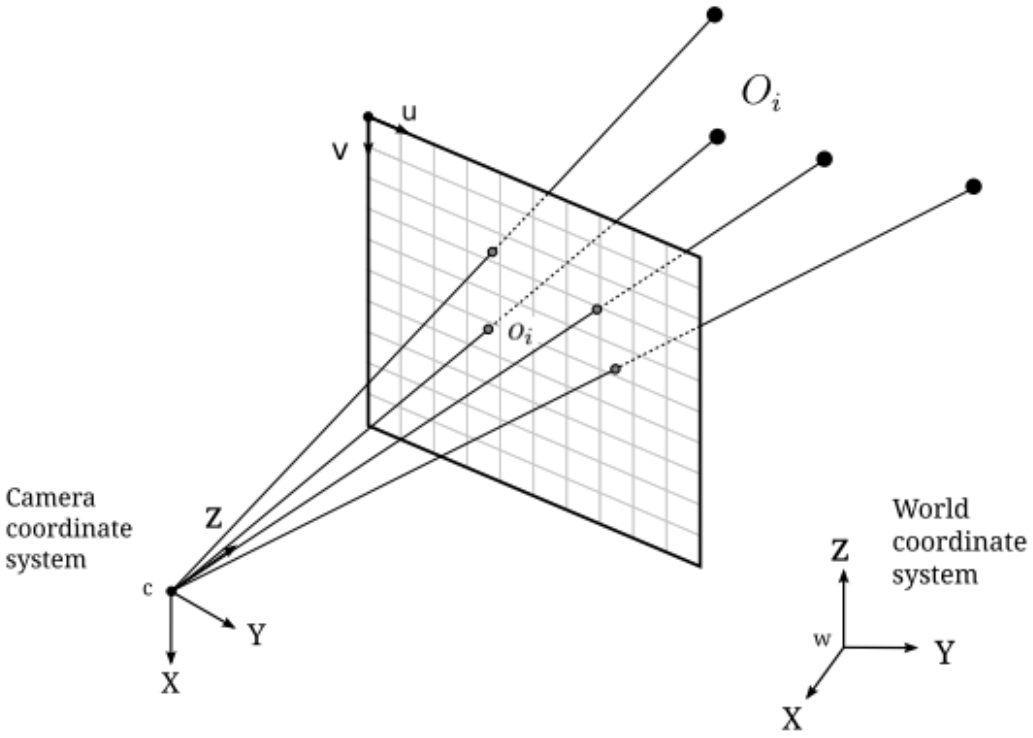


Figure 2.3: Illustration of the Perspective-n-Point Algorithm taken from [14]

$$d = \frac{1}{N} \sum_{i=1}^{N} \|y_i - (\hat{R}x_i + \hat{t})\| \tag{2.4}$$

To this end, ICP is often used with PnP to refine an initial pose estimate by incorporating depth information given by an infrared sensor (e.g. Microsoft Kinect) or stereo-vision camera (e.g. Stereolab ZED) [11, 26]. ICP often fails if the initial pose estimation is poor or if the two point clouds are dissimilar, such as trying to refine an object pose without first segmenting the object from the scene (Fig. 2.4). Note that segmenting an object of interest from the scene would increase the accuracy of feature matching. Thus, ICP is typically used in learning-based methods for which segmentation masks are made available

[11, 26]. However, implementing ICP is too slow for real-time deployment as it is iterative in nature and operates on expensive 3D point clouds.

2.4 Learning-Based Pose Estimation

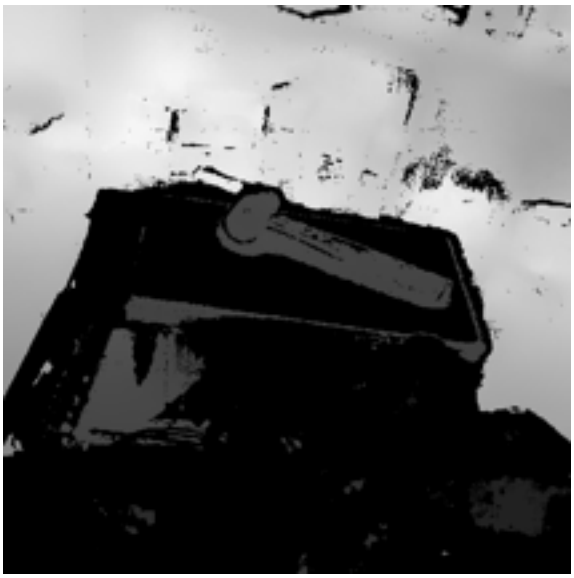
2.4.1 RGB Architectures

As previously mentioned in Section 2.3, classical methods rely on detecting and matching keypoints with known object models. More recently, the trend has been to learn robust keypoints, descriptors, and matching through a large number of sample images [11, 18, 19]. Frameworks such as Deep Object Pose Estimation (DOPE) [18] learn points of interest via 2D heatmaps of where a keypoint is located (i.e. belief maps). An alternative to learning sparse keypoints is offered by dense methods in Pixel-wise Voting Network (PVNet) [19], which predict unit vectors pointing to keypoints for each pixel. The latter has been more effective in estimating pose under heavy occlusion [19]. Thus, the PnP algorithm is still widely used in many deep learning frameworks for 6-DoF pose estimation today and such frameworks can be deployed in real-time [18, 19].

Different to the above approaches, which leverage PnP, frameworks such as PoseCNN [11] directly regress the 6-DoF pose of an object. The first stage of PoseCNN is object segmentation. As such, segmentation masks provide us with rich information about an object relative to spare keypoints, but can introduce additional errors if an object mask contains pixels of another object or background [7, 27]. PoseCNN uses object labels from a segmentation mask to predict a unit vector from each pixel to the center of each object. Hough voting is then used to estimate the 2D pixel coordinates of an object centroid, which is combined with a separate estimate of the depth to an object centroid, to recover an object 3D translation vector t. Lastly, the 3D Rotation R is estimated by regressing convolutional features inside an object bounding box to a quaternion representation of R. However, as the authors of DOPE [18] highlight, directly regressing the pose can lead to additional errors when deploying a trained model with a different camera as the network can overfit to the camera intrinsics.

2.4.2 Heterogeneous RGB-D Architectures

One of the main issues with architectures, such as DOPE [18] or PVNet [19], that utilize only 2D RGB images is that they can perform poorly in terms of accuracy, as errors that are small on the image plane can be large in 3D space. Sources of error stem from the fact



(a) Depth Image

(b) Segmentation Mask

(c) Mallets 3D Mesh

(d) Scene Point Cloud

(e) ICP-fitting of the Mallet

Figure 2.4: Human Assisted ICP-fitting of the Mallets 3D Mesh to a 3D Reconstructed Scene

that obtaining 2D-3D correspondences from local patches can be difficult and that different keypoints in 3D space may overlap after projection onto the image plane [20]. To address this problem, frameworks such as PoseCNN [11] use depth images and ICP to refine an initial pose estimate. However, making full use of 3D input is an expensive post-processing step as runtime for PoseCNN+ICP is reportedly 10.6 s per image [7], which is two orders of magnitudes off from real-time implementation. However, it is clear from the literature [7, 11, 20] that heterogeneous architectures that fuse 2D RGB features with 3D point cloud features outperform architectures that only use 2D RGB features in terms of accuracy.

Figure 2.5 displays an overview of DenseFusion, which is a two-stage framework for pose estimation that operates on 2D and 3D features. The first stage leverages instance segmentation which produces region proposals and segmentation masks for objects of interests. To this end, performance for DenseFusion is highly dependent on the accuracy of the framework used for object segmentation. The authors of [27] highlight that object boundaries are most susceptible to errors with segmentation as pixels can be incorrectly classified as the background. As a result, detailed information about the structure of an object is lost or smoothed. The second stage operates only on regions of interest and depth images are masked, as the entire 3D point cloud of the scene does not need to be processed, to reduce computational costs. Cropped RGB image data (i.e. colour embeddings) and masked point cloud data (i.e. geometric embeddings) are later processed independently by a CNN and a PointNet like architecture.

The authors of DenseFusion [7] highlight that blindly fusing 2D and 3D features could degrade performance under heavy occlusion and/or segmentation errors as features may contain pixels or points on other objects or parts of the background. Color and geometric features are instead fused locally using a pixel-wise fusion network based on unsupervised confidence scoring (Fig. 2.5). As a result, DenseFusion produces a per-pixel pose estimate and the most confident prediction is used to generate the final pose. The authors of DenseFusion [7] investigated the robustness of their model towards occlusion such that performance does not degrade with significantly greater levels of occlusion. Note this is not the case for PoseCNN [11] and PointFusion [28]. PointFusion is similar to DenseFusion as colour and geometric embeddings are fused in a heterogeneous architecture, but DenseFusion significantly outperforms PointFusion's global fusion of 2D RGB and 3D depth features [7]. DenseFusion can also further refine a pose estimate with an iterative refinement module that does not significantly increase computational costs [7]. Note that the authors of DenseFusion report a 200x improvement in terms of runtime (e.g. 0.06 s per image), which is near real-time implementation, as compared to PoseCNN+ICP [7].

Of the different ways to fuse RGB and depth modalities [29, 30], using depth as an additional channel to RGB input (i.e. *early-fusion*) is the most simplistic method. Previous

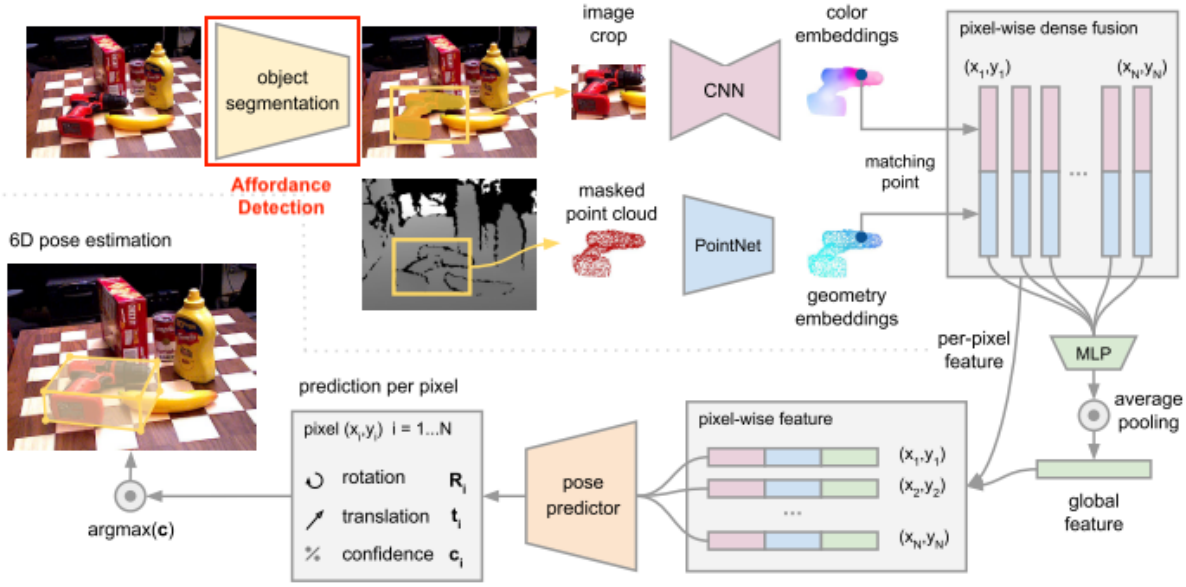


Figure 2.5: Overview of the DenseFusion Framework taken from [7]

works [29] highlight that one encoder cannot simultaneously extract features from RGB and depth images. To solve this, one can use two encoders, one for RGB features and one for depth features, and concatenate features later in the network (i.e. *late-fusion*). This effectively doubles the number of parameters for the backbone feature extractor and fusing heterogeneous features may introduce deviating statistics into the network, which may not lead to an improvement in preformance.

For 3D recognition tasks, such as 6-DoF pose estimation, point clouds are an important geometric data structure [31]. However, different to processing regular input data such as RGB features, effectively representing a point cloud is challenging as it is an unordered set of vectors. PointNet [31] pioneered the use of max pooling to achieve permutation invariance in processing sets of unordered 3D point clouds. PointNet has many desirable properties, such as avoiding lossy operations by discretizing a point cloud into voxels, and it scales linearly with the number of input points [31]. An overview of PointNet is outlined in [31].

2.4.3 Discussion

In selecting a 6-DoF pose estimation framework for robotic grasping, it is important to consider the trade-off between runtime and accuracy. Note that it is not always possible to deploy state-of-the-art computer vision research on constrained hardware. From the literature [7, 11], incorporating depth information generally improves performance for 6-DoF pose estimation but missing pixels on shiny surfaces or translucent objects can also degrade performance. The advantages and disadvantages of learning based methods for 6-DoF pose estimation are summarized in Table 2.1.

Table 2.1: Pros and Cons of Several 6-DoF Pose Estimation Frameworks

	Pros	**Cons**
DOPE (RGB) [18]	• Train only on synthetic images	• Sparse keypoints can lead to larger errors
PoseCNN (RGB-D) [11]	• Dense RGB features	• Slow ICP refinement of an initial pose estimate
DenseFusion (RGB-D) [7]	• High Accuracy • Fast Runtime	• Incorporating depth could degrade performance

Nonetheless, all previously mentioned frameworks for pose estimation predict the 3D translation vector t to the centroid of the 3D bounding volume that contains an object. This may be appropriate to grasp simple objects but not for more complex objects such as a power drill (Fig. 2.5). A key issue is addressed in this thesis to improve pose estimation in the context of robotic grasping with scene understanding. It is proposed that this approach can be applied to any two-staged framework for 6-DoF pose estimation, such as DenseFusion [7].

2.5 Visual Affordance for Robotic Manipulation

The concept of affordances was first introduced by the psychologist James Gibson in 1966 [5]. Affordance detection is highly useful for determining what actions can be performed on which object(s) where in a given scene. Different to visual (e.g. color, texture or shape) and/or physical (e.g. weight) properties, affordance detection allows an autonomous agent to learn intelligent interactions with an object to assist humans in various tasks.

Of the different ways to model visual affordance [5], pixel-wise labels are considered, similar to previous work [6], as state-of-the-art object segmentation networks can be lever-

aged. However, affordance detection is a more challenging problem than object segmentation as a single object can take on multiple labels, such as grasp and pound (Fig. 1.1), which introduces problems such as ranking, correlation and dependency [32]. An object affordances can also be dynamic and an autonomous agent should consider prior knowledge (e.g. a hammer may be first 'graspable' then 'poundable').

Lastly, an affordance detection framework should run in real-time and generalize well to objects in new environments and/or novel objects.

2.5.1 Object Segmentation vs Affordance Detection

The problem of understanding affordances at a pixel-wise level is referred to as "object part labelling" by the computer vision community or "affordance detection" by robotic researchers. Roboticists place emphasis on object parts that an agent can interact with while the concept of affordances is not restricted solely to objects, such as a street, by the computer vision community. A comprehensive review of visual affordance is outlined in [5].

The rise of deep learning has vastly allowed for near real-time affordance detection frameworks with encoder-decoder architectures such as AffordanceNet [6]. To this end, the different functionalities of an object should be identified and localized through feature learning techniques (CNN-based models), which then provides context to estimate a suitable grasp location (Fig. 1.1).

The task of task of object segmentation and affordance detection are formalized in Table 2.2, such that they both leverage object detection frameworks. Mask R-CNN [10] is an extension of prior works, such as Faster R-CNN [33], for region proposal based object detectors. There are two main stages to Mask R-CNN, the first leverages a Region Proposal Network (RPN) to propose candidate object bounding boxes. The latter operates on low resolution feature maps to simultaneously predict three outputs:

1. Classify each object in an image, $L_{\text{object class}}$
2. Regress bounding box offsets to localize 2D positions of each object, $L_{\text{bounding box}}$
3. Classify each pixel within each bounding box to its most probable object label, L_{mask}

The key difference between object segmentation and affordance detection lies within the mask branch, such that the former segments a single object label whereas affordance detection segments one or multiple affordance labels (Fig. 1.1). To classify each pixel

within an object bounding box to its most probable affordance label, affordance detection frameworks, such as AffordanceNet [6], leverage object segmentation networks, such as Mask R-CNN [10].

Table 2.2: Learning Objectives for Object Segmentation and Affordance Detection

	Object Segmentation	**Affordance Detection**
$L_{\text{object class}}$	1. Classify each object in the scene	1. Classify each object in the scene
$L_{\text{bounding box}}$	2. Localize each object in the scene	2. Localize each object in the scene
L_{mask}	3. Label pixels corresponding to each *object*	3. Label pixels corresponding to each *affordance*

2.6 Real and Synthetic Datasets for Learning-based Methods

One key issue addressed in this study is how to generate effective datasets with ground truth annotations for affordance labels and object part 6-DoF poses. Ultimately, trained models should learn inherent features of objects robust to varying backgrounds and lighting conditions. Previous work for generating real datasets investigated the effects of single vs multi- object scenes, number of background environments and frame rate on performance [34]. The authors of [35] also explored the effects of collecting a dataset with multiple cameras, all with different camera intrinsic matrices. In this section, several benchmark datasets are presented, which contain either ground truth annotations for affordance detection [8, 12] or 6-DoF pose estimation [9, 11] but not both.

Affordance Detection

The authors of [6] applied AffordanceNet to the University of Maryland (UMD) dataset [8], which consists of 105 kitchen, workshop and garden objects. The UMD dataset contains approximately 30k RGB-depth-affordance mask trios, whereby each affordance mask contains pixel-wise, hand annotations. However, as the authors of AffordanceNet [6] highlight, the UMD dataset contains only one object in each image without clutter or occlusion, which is a convenient way for researchers to frame the affordance detection problem. Many real-world, robotic applications, such as the Amazon Picking Challenge [1], contain scenes with heavy clutter and occlusion. The authors of [6] also deployed AffordanceNet to the IIT-AFF dataset [12], but this dataset contains less than 10K images.

It is well known that deep frameworks require large datasets with high quality images to generalize well. Hand annotations for pixel-wise affordance labels limits the generation of such datasets, as [36] reported 35 hours of work to hand annotate 800 images. Prior work in affordance detection [5] have tried to overcome the issue of hand annotations through semi-supervised learning or transfer tables to map object parts to affordance labels. The first study required a human in each scene to represent context, while the second crafted a custom lookup table to map object parts to affordance labels. As performance for CNNs is typically tied to how representative the training data is of the true data, practitioners should avoid generating a new dataset for every environment they wish to deploy a trained model in. As trained models will normally be deployed in real settings, which follows a different distribution from synthetic datasets, one typically observes a drop in performance, or the *reality gap* [18], if the model is trained solely on synthetic images. Adversarial learning frameworks such as [37, 38] may decrease the *reality gap*, but not to the same extent as fine-tuning a network on real images [36]. For instance, [36] studied the use of synthetic images on supervised training of a Mask R-CNN framework for category-agnostic instance segmentation whereby the network was trained on 50K synthetic depth images. However, 800 real images were required to fine-tune the network to bridge the drop in performance, or *reality gap*, when exposing real images to models trained on synthetic images.

6-DoF Pose

The authors of DOPE [18] developed NVIDIA's Deep learning Dataset Synthesizer (NDDS) plugin for NVIDIA's Unreal Engine 4 (UE4), which auto-generates photorealistic (PR) or domain randomized (DR) images. NDDS exports metadata throughout scene capturing, namely RGB-depth-segmentation mask trios and 6-DoF pose, for each object in an image. This work led to the creation of the Falling Things (FAT) dataset [9], which is a synthetic dataset with ground truth annotations for object segmentation and 6-DoF pose estimation. The aim of this work was to leverage synthetic images to train DOPE to demonstrate that the *reality gap* can be bridged by a simple combination, such as a 50/50 split, of PR and DR images. PR images were generated across 3 different domains with physical constraints and DR images were generated with random distractors, textures, backgrounds, object poses, and lighting conditions. The authors of DOPE [18] reported that their simple architecture, which was trained solely on synthetic data, achieved similar results to PoseCNN, which was trained on a mixture of real and synthetic images. However, DOPE operates solely on RGB images and depth images can be more difficult to simulate as they can contain missing pixels. It is proposed that similar results to DOPE may not be achieved with a pose estimation framework that operates on RGB-D data, which will be verified later in

Chapter 4.

PoseCNN was trained on the YCB-Video dataset [11], which contains 19,138 real and 80,000 synthetic images. Images were captured or simulated with an Asus Xtion Pro Live at a cropped resolution of 640x480. Note, 92 videos were used to generate real images and synthetic images were generated by randomly placing objects in an image with a black background. Each scene contains 3 to 9 objects, selected from 21 different YCB objects [39], with varying levels of occlusion or clutter. The authors of the YCB-Video dataset auto-generate annotations by manually specifying the pose of each object using 3D mesh files in the first frame of each video. Ground truth annotations suffer from errors such as rolling shutter of the RGB sensor, inaccuracies in the object models, slight asynchrony between RGB and depth sensors, and uncertainties in the intrinsic and extrinsic parameters of the cameras [11]. However, this level of error is accepted in the literature [7, 27].

Leveraging Real and Synthetic Images

Traditionally, previous works have used real datasets for affordance detection [8, 12] and 6-DoF pose estimation [16]. However, real datasets can be limited in the number of high quality images, such as IIT-AFF [12], or can only provide a limited range of object poses, backgrounds and lighting conditions, such as the LINEMOD dataset [16]. Recent work for object segmentation has reduced the annotation effort by pre-training on a large number of synthetic images and later fine-tuning on a small number of real images [36]. More ambitious work for affordance detection has attempted to train a model solely on synthetic images [40], but there still exists a large drop in performance when exposing the model to real images. For 6-DoF pose estimation, previous work for RGB frameworks have trained solely on synthetic images [18]. However, it is more common for RGB-D frameworks to train on a dataset with a mixture of real and synthetic images. The advantages and disadvantages of real and synthetic images are summarized in Table 2.3.

In closing, it is proposed that synthetic datasets can supplement, and not replace, real datasets. The gap between synthetic and real images will continue to decrease as rendering techniques in simulators, such as UE4, improve. However, it remains extremely challenging to evaluate the effectiveness of a synthetic dataset with quantitative measures. Work such as the FAT dataset [9] has provided researchers with intuition and tools to generate more effective synthetic datasets. However, model performance remains to be the best measure for the effectiveness of a synthetic dataset.

Table 2.3: Pros and Cons of Real and Synthetic Images

	Pros	**Cons**
Real	• High utility	• Computationally expensive to manually label
Synthetic	• Computationally inexpensive to generate a wide variety of object poses, background and lighting conditions	• Domain Shift (i.e. *reality gap*)

Chapter 3

Generating Real and Synthetic RGB-D Datasets

From the literature, no existing dataset contains both ground truth annotations for affordance labels and object part 6-DoF poses. It was also observed that no benchmark dataset for affordance detection, such as UMD [8] or IIT-AFF [12], contains a large number of annotated scenes with cutter as pixel-wise, hand annotations is labor-intensive.

The main contributions of this chapter is two-fold. A post-processing technique is presented first, which generates desired ground truth annotations, namely affordance labels and object part 6-DoF poses, from object 6-DoF poses. This method was applied to the YCB-Video dataset [11], which natively contains ground truth annotations for object segmentation and 6-DoF pose. Afterwards, methods to auto-generate real and synthetic images were leveraged to generate a dataset, which contains over 100K real and synthetic images with 11 different household objects and 9 different affordance labels. Both datasets were used throughout Chapter 4 to evaluate performance for 6-DoF pose estimation with object segmentation and affordance detection.

3.1 Overview

Following an approach similar to the YCB-Video dataset [11], a Stereolab ZED camera was used to auto-generate a dataset with real and synthetic images, which contains ground truth annotations for object segmentation and 6-DoF pose. A post-processing technique was later applied to generate desired ground truth annotations, namely affordance labels and object part 6-DoF poses.

In this thesis, it was assumed that 6-DoF pose is estimated to known objects for which 3D mesh files are made available a priori. As such, 5 common household tools were selected, as they can easily be grasped, along with 6 YCB objects from the YCB-Video dataset to generate a dataset (Fig. 3.1). Mesh files for these 11 objects were either generated or publicly available as it will be discussed in this chapter.

The canonical frame of object parts that were well suited for grasping are displayed in Figure 3.1. Physical constraints of a three-finger gripper, such as a BarrettHand, were considered in selecting which object part to grasp. For instance, it would be difficult for a three-finger gripper to grasp the handle of the mug or pitcher.

Stereolab ZED Camera

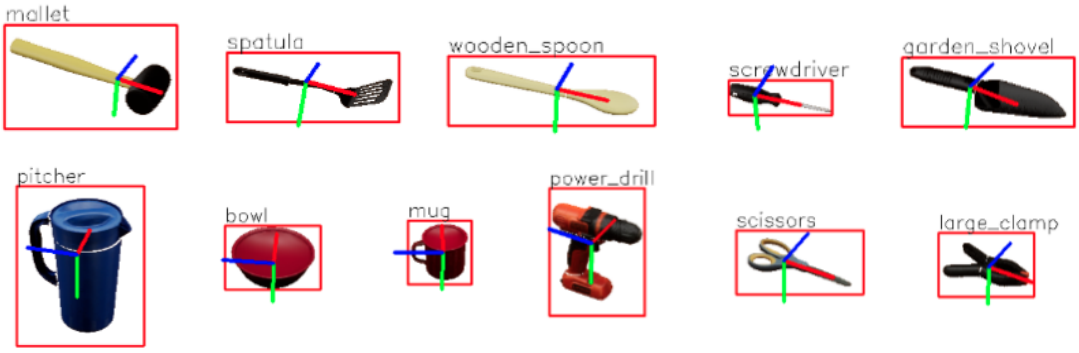
A Stereolab ZED camera, or stereo vision, was used throughout dataset generation and later to provide visual input to the robot. Infrared sensors are also commonly used in the literature, such as an Asus Xtion RGB-D sensor [11], but they are limited to indoor applications, whereas stereo vision can operate in both indoor and outdoor environments.

The ZED camera has a pair of RGB cameras fixed at a baseline distance from one another (Fig. 3.2). Using the two viewpoints, or triangulation, an associated depth map is computed from left and right RGB images. The depth map expresses distance as a 16 bit integer and the effective depth range is approximately 0.5 m to 20 m. In this thesis, rectified RGB-D images are used as combining RGB and depth modalities can improve performance for 6-DoF pose estimation [11]. However, depth images can contain missing pixels from objects that are transparent or glossy and/or objects that are too close or too distant.

Blender

As previously mentioned, the object 3D mesh file was required to auto-generate ground truth annotations for object segmentation and 6-DoF pose estimation. The authors of [18] highlight that the object mesh in a simulator, such as Blender, must be equal to the object dimensions in reality. To this end, objects were limited to one instance in dataset generation. It is possible for one to generate a dataset with 20 different instances of an object, such as a mug, but one would also require the object mesh for each instance of that object.

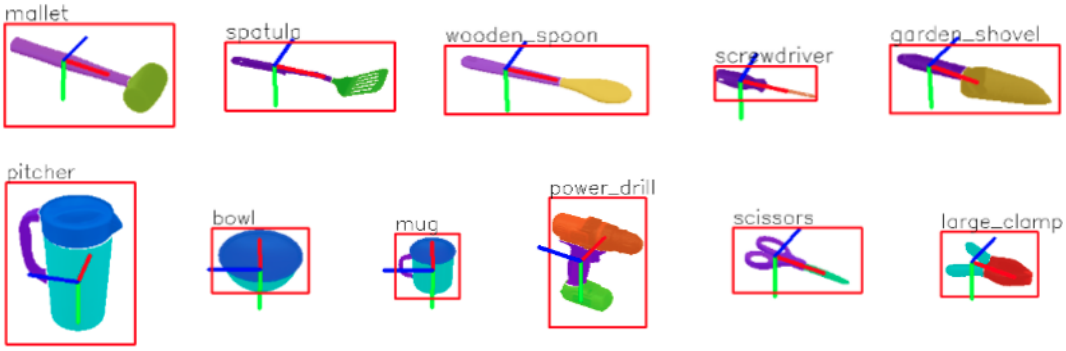
Mesh files for the 5 household objects in dataset generation were scanned based on a miniaturized projection technique. Canonical frames for each object were defined to be



Household

YCB [39]

(a) Centroids located at the Object Center of Mass



Household

YCB [39]

(b) Affordance Masks with Centroids located at Object Part Center of Mass. The colour of the mask represents the affordance label. Grasp: purple; screw: orange; scoop: yellow; pound: light green; support: green; cut: teal; wrap-grasp: light blue; contain: dark blue; clamp: red.

Figure 3.1: Display of the 11 Objects from the ARL AffPose Dataset

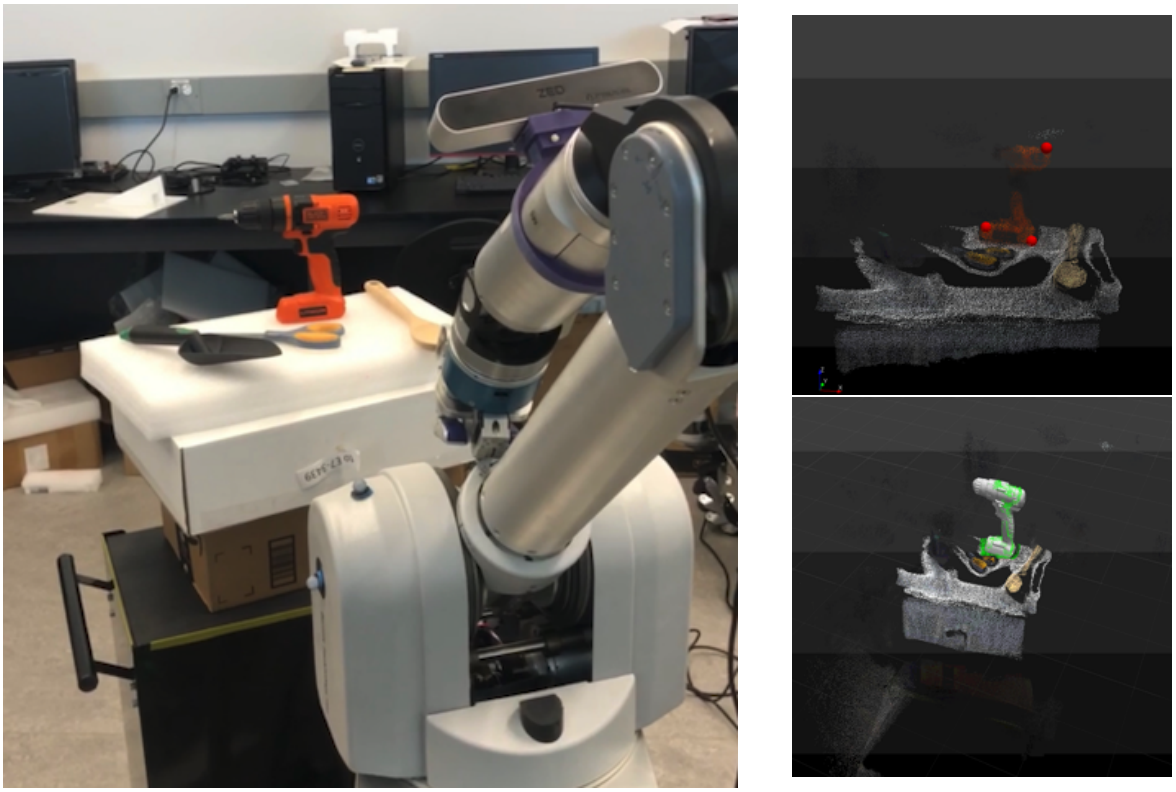


Figure 3.2: Stereolab ZED Camera mounted on a 7-DoF Barrett WAM Arm used throughout Dataset Generation (left) and Human Assisted ICP-fitting of the Power Drill (right)

the object center of mass, which was determined from the object volume in Blender (Fig. 3.1). Similarly, the remaining 6 YCB objects have publicly available mesh files, such that the object centroids are also defined to be the object center of mass.

Object parts were then sectioned in Blender to contextualize different functionalities (i.e. affordances). For instance, the mallet was sectioned into its handle for grasping and head for pounding (Fig. 3.3). The canonical frame for an object part was also aligned to the object part center of mass (Fig. 3.3).

Post-processing for an Object Part 6-DoF Pose

To generate desired ground truth annotations, which were affordance labels and object part 6-DoF poses, from object 6-DoF poses, three mesh files were prepared in Blender:

1. Object mesh with the centroid aligned to the object center of mass (Fig. 3.3a)

2. Object part mesh with the centroid aligned to the object center of mass (Fig. 3.3b)
3. Object part mesh with the centroid aligned to the object part center of mass (Fig. 3.3c)

Blender was then used to determine the exact translation from the object to the object part coordinate frame (Fig. 3.3b and Fig. 3.3c). It is proposed that no additional sources of error was introduced in generating an object part 6-DoF pose as Blender was leveraged to determine the exact transformation.

Once an object part 6-DoF pose is known, an object part point cloud can be projected onto the image plane, with known camera instrinsics, to generate an affordance mask. In this thesis, each object part was assigned a single, static affordance label. It is assumed that it would be easy to re-assign an affordance label based on an action primitive with an object part 6-DoF pose relative to pixel-wise, hand annotations.

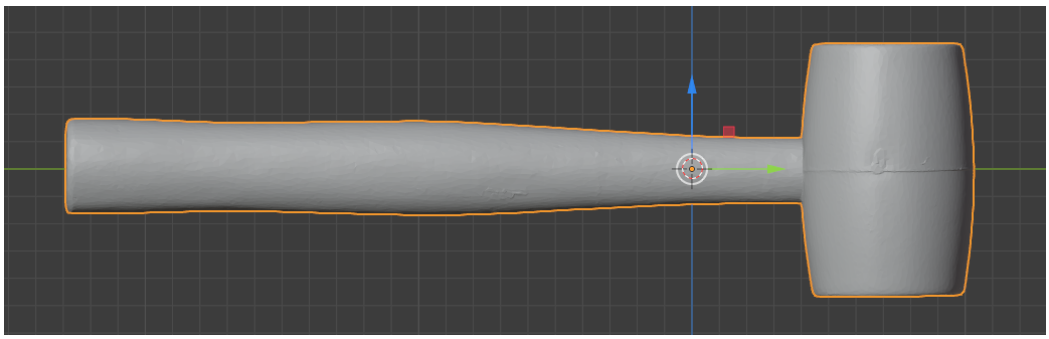
YCB-Video Dataset

Prior to dataset generation, 3D mesh files for the 21 YCB objects in the YCB-Video dataset were post-processed to generate desired ground truth annotations (Fig. 3.4). Note that the YCB-Video dataset, which natively contains ground truth annotations for object segmentation and 6-DoF pose, and 3D mesh files were publicly available online. For the 21 YCB objects in the YCB-video dataset, 14 YCB objects inherit a single affordance label, such as wrap-grasp for the banana or soup can (Fig. 3.4). Thus, these simple objects were not included throughout dataset generation as scene understanding with object segmentation or affordance detection provides the same information to grasp such objects.

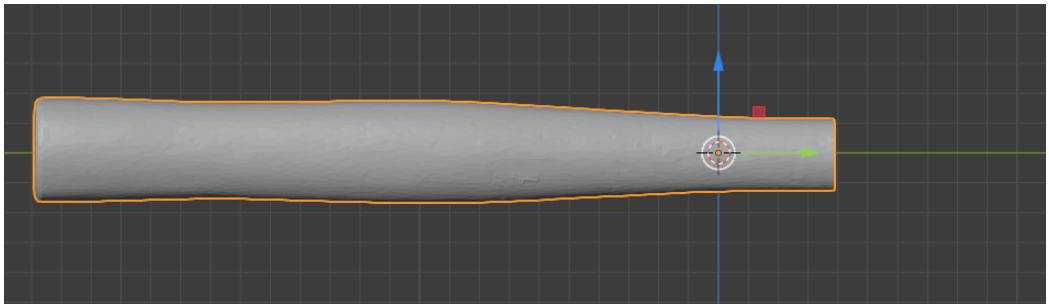
Ultimately, affordance detection provides a more suitable grasp location for complex objects such as the power drill (Fig. 3.5). Note that the distribution of affordance labels, such as the wrap-grasp in teal, is highly unbalanced relative to the distribution of object labels in the YCB-Video dataset (Fig. 3.5).

3.2 Generating Real Images using LabelFusion

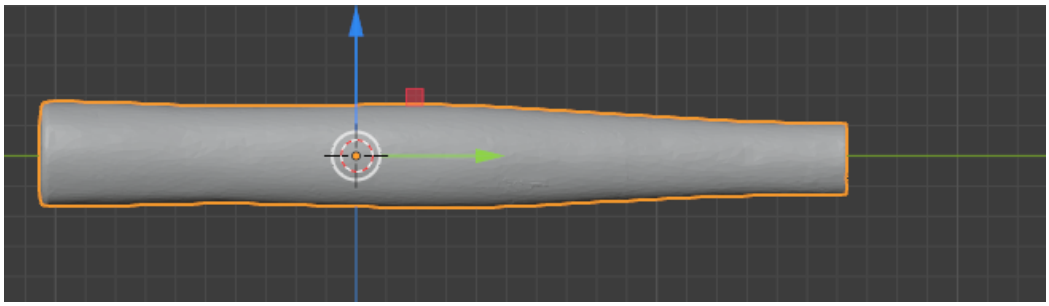
In this work, annotations for real images were auto-generated similar to previous work such as the YCB-video dataset [11]. The authors of [34] developed LabelFusion, which leverages ElasticFusion [41], a dense SLAM method to produce a 3D reconstruction of a scene from a RGB-D video with multiple viewpoints (i.e. structure from motion). Human-assisted



(a) Mallet Object



(b) Handle from Mallet Object in the Object Frame



(c) Handle from Mallet Object in the Object Part Frame

Figure 3.3: Display of the Canonical Frames for the Mallet in the ARL AffPose Dataset

Figure 3.4: Display of the 21 YCB Objects from the YCB-Video Dataset taken from [11]

ICP-fitting was then used to align objects to the reconstructed scene to determine the final 6-DoF pose of the object (Fig. 3.5). With known camera intrinsics, labelled point clouds can be back-projected to the image plane to generate object segmentation masks. The entire annotation process took approximately 30 seconds per object in a simple scene and up to 5 minutes per object in a cluttered scene per video. The authors of [34] reported that it took only a few days to collect over 1,000,000 labelled images.

Objects, and not object parts, were aligned with ICP-fitting in 3D reconstructed scenes and a post-processing technique was applied to generate affordance labels and object part 6-DoF poses (Fig. 3.2). This ultimately reduced the annotation effort in each scene and avoided additional sources of error as ICP-fitting of object parts was extremely difficult. It was observed that aligning object parts, which have smaller point clouds relative to an object, led to larger errors with human assisted ICP-fitting in LabelFusion. For instance, attempting to align object parts for screwdriver led to generated poses that were not physically plausible, such that the x-axis (in red) of the screwdrivers handle and tool bit were not aligned (Fig. 3.1).

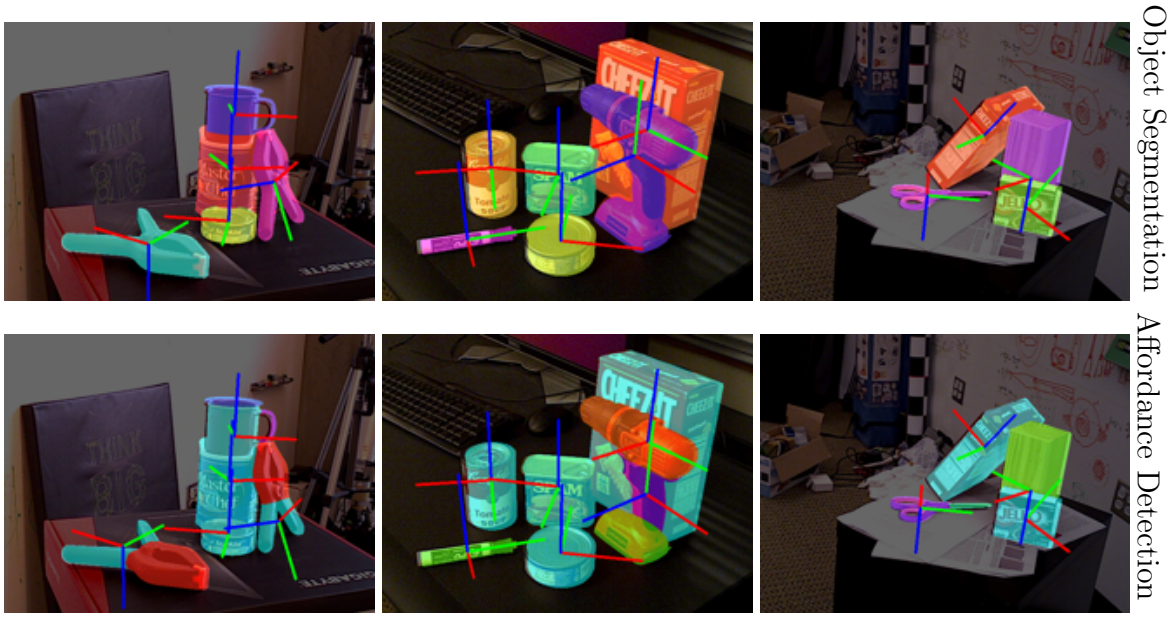


Figure 3.5: Display of Real Images from the YCB-Video dataset with Overlaid Masks and 6-DoF Pose. The colour of the mask represents the affordance label. Grasp: purple; screw: orange; support: green; cut: teal; wrap-grasp: light blue; contain: dark blue; clamp: red.

Ultimately, the goal throughout dataset generation was to deploy a trained model(s) in a lab environment with a ZED camera mounted on a robot. However, the objective was not to produce an ad hoc dataset, which overfits a trained model(s) to a lab environment. Thus, 71 different videos were captured in household and lab environment to generate real images (Fig. 3.6). Each of the 14 different scenes contains randomly selected foreground objects and background clutter. Note that the dataset consists of 10 scenes from a household environment, 3 scenes from a lab environment and only 1 scene in a lab environment with a camera mounted on a manipulator (Fig. 3.6). Objects were also placed in random orientations at random locations in each scene. Note that symmetrical objects, such as the bowl, are difficult to align in 3D reconstructed scenes as an infinite number of poses can lead to identical observation. Such objects are rotationally symmetric about an axis (e.g. y-axis for the bowl) (Fig. 3.1).

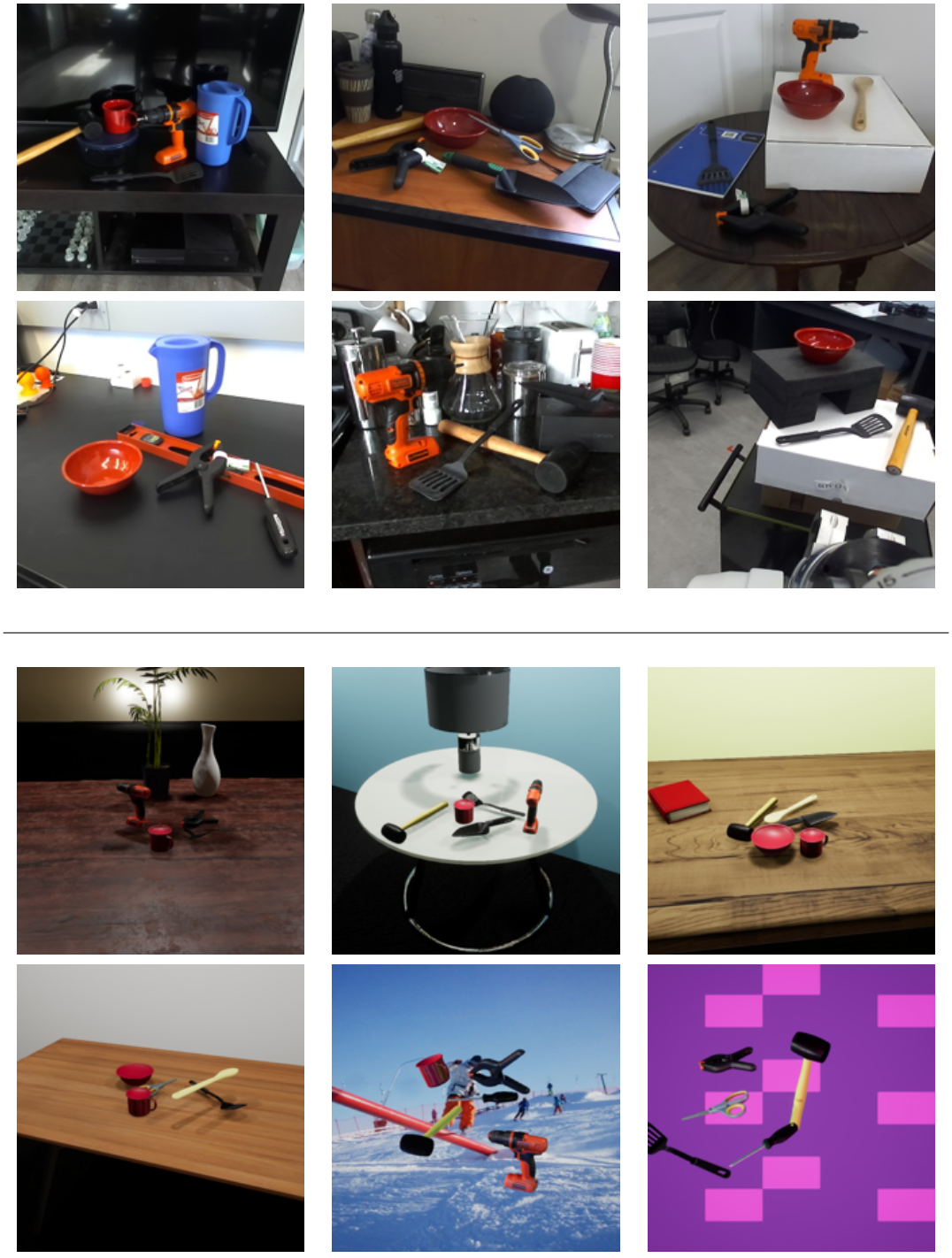


Figure 3.6: Display of Real (top) and Synthetic (bottom) Images from the ARL AffPose dataset

3.3 Generating Synthetic Images using NVIDIA's Unreal Engine

In this thesis, real images were supplemented with synthetic PR and DR images. Procedures from previous work, namely the FAT dataset [9], and resources from the online GitHub community for NDDS and DOPE were used to generate synthetic images. Note that the authors of DOPE [18] investigated performance with different mixtures of synthetic images, for which performance was comparable as long as at least 40% of PR or DR images were present. As such, a simple 50/50 split of PR and DR images was included in this dataset (Fig. 3.6). 80 different videos were simulated throughout scene capturing for synthetic images using a virtual camera with the same resolution and intrinsics as the ZED camera used to capture real images. Objects were again placed randomly within 5 PR or 3 DR scenes with random orientations.

3.3.1 Photorealistic (PR) Images

Photorealistic images were auto-generated by placing randomly selected foreground objects in each scene. The environments were custom-built in UE4 and modelled as standard household environments or workspaces (Fig. 3.6). Lighting conditions from an indoor spot light were constant in each scene but varied across each scene. Throughout scene capturing the virtual camera was moved randomly with respect to a target volume, which user-defined limits were set for the camera's pitch and yaw angles and distance along the camera's z-axis.

3.3.2 Domain Randomized (DR) Images

Domain randomized images were auto-generated by placing randomly selected foreground objects in a scene with a random background, which included solid colours, multi-colored grid patterns or an image from the COCO dataset [42] (Fig. 3.6). Three random spot lights were used to vary the lighting conditions throughout scene capturing. The virtual camera for DR images also moved randomly throughout scene capturing similar to the process outlined for PR images.

3.4 ARL AffPose Dataset

Table 3.1 provides key statistics for the generated dataset, which was named the Advanced Robotics Lab (ARL) Affordance Pose (AffPose) dataset.

The ARL AffPose dataset was generated by capturing 44,527 real RGB-D images across 14 different scenes and 79,661 synthetic RGB-D images across 5 PR scenes and 3 DR scenes. Note, 8 out of 71 real videos were randomly held-out for testing images similar to previous work [11]. All real or synthetic images were captured or simulated with a ZED camera at a resolution of 1280 width $\times$ 720 height. Each scene contains 3 to 6 objects from 11 different objects (Fig. 3.1). Note that there exists a larger domain shift from synthetic to real images, as compared to variation within real images from different scenes. Although virtual environments were modeled to be similar to real scenes, the gap between synthetic and real images will continue to decrease as rendering techniques in simulators, such as UE4, improve.

For the annotation effort, approximately 15 hours were required to generate synthetic images and 53 hours, or 45 minutes for each of the 71 videos, to generate real images. Note, these time estimates do not account for time to become familiar with building custom environments in UE4 or LabelFusion. It is proposed that the ability to generate a large number of real images in a relatively short period of time, relative to pixel-wise, hand annotations [8, 12], is crucial for deploying affordance detection frameworks in novel settings.

Table 3.1: Statistics from the ARL AffPose Dataset

	Real	**Syn**
Objects	11	
Affordance Labels	9	
Camera	Stereolab ZED	
Resolution	1280x720	
Images	44,527	79,661
Scenes	14	8
Total Number of Videos	71	80
Held-out Videos for Testing	8	0
Min Object Count	3	
Max Object Count	6	

3.5 Discussion

Accuracy of LabelFusion's Ground Truth Annotations

As previously mentioned, generating ground truth annotations using methods such as LabelFusion is widely accepted in the literature [11, 27]. For this reason, the accuracy of ground truth annotations for 6-DoF pose was not evaluated. Future work could attempt to do so using a Vicon motion capture system. However, to do so one must mount Vicon markers on each object, which may introduce artificial features on each object (Fig. 3.7). It is proposed that several tests could provide insight on the magnitude of error for LabelFusion, but this error would vary from scene to scene, as the ground truth 6-DoF pose is dependent upon human-assisted ICP fitting of object meshes in 3D reconstructed scenes.

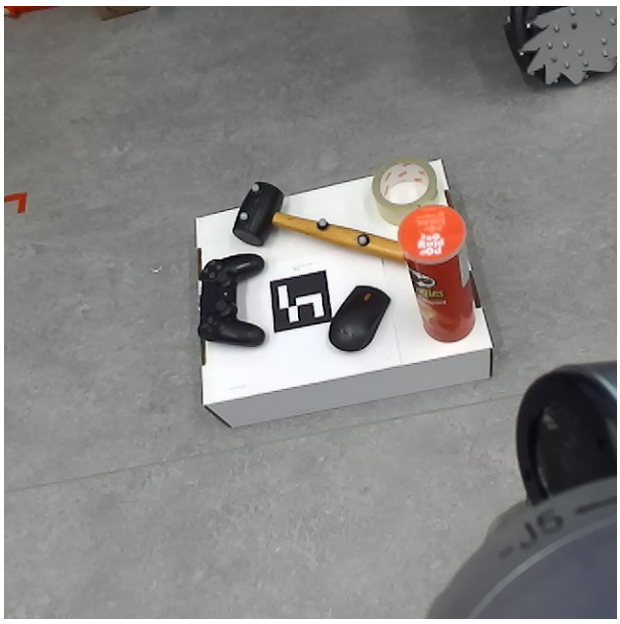


Figure 3.7: Mallet Object with Vicon Markers

Chapter 4

Object Segmentation and Affordance Detection Applied to 6-DoF Pose Estimation

In this chapter, an affordance detection network is integrated within a 6-DoF pose estimation network. That is, open source work for Mask R-CNN [10] written in PyTorch [43] was modified to have a network similar to AffordanceNet [6].

The main contributions of this chapter is an experimental evaluation of object segmentation and affordance detection applied to 6-DoF pose estimation. It is hypothesized that performance for 6-DoF pose estimation with affordance detection may decrease on object parts as one would operate on a subset of RGB-D data. With this in mind, the proposed pipeline is evaluated on the YCB-Video dataset as well as on the ARL AffPose dataset with different combinations of real, synthetic and a mixture of real and synthetic images.

All experiments were conducted on a workstation running Ubuntu 18.04 with an Intel Core i7-9700K CPU @ 3.60G Hz and one Nvidia GeForce RTX 2080 Ti GPU. This work was implemented using Python 3 and PyTorch.

4.1 Scene Understanding with Mask R-CNN

4.1.1 Overview

Figure 4.1 displays an overview of AffordanceNet [6], which outputs an object label, bounding box and multi-class affordance masks. The end-to-end architecture is similar to Mask

R-CNN [6], a widely used framework for object segmentation, such that a CNN backbone first extracts image features from a RGB image. The Region Proposal Network (RPN), which shares the same feature map as the backbone, proposes candidate bounding boxes or regions of interest (RoI). RoIAlign extracts and pools each RoI to a fixed 7x7 feature map as each RoI can vary in size and head branches for object classification and bounding box regression use fully connected (FC) layers. Note that a FC layer requires a fixed input size, whereas input size can vary with convolution layers. Lastly, two FC layers are used to classify the object class and regress the object location and a series of convolution and deconvolution layers are used to predict the multi-class affordance mask.

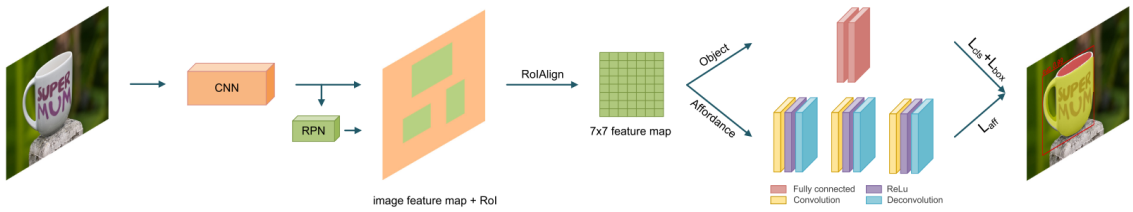


Figure 4.1: Overview of AffordanceNet taken from [6]

AffordanceNet was publicly released in 2017 and was implemented in Caffe with a VGG16 backbone. PyTorch is used in this work for ease of implementation and open source work was modified for `Mask R-CNN`, which was based on Torchvision's implementation of `Mask R-CNN`. Two key differences exist between AffordanceNet and this work. First, a modified mask branch, which is discussed later in following subsection, and second, VGG16 is replaced for ResNet50 with a Feature Pyramid Network (FPN) [44]. Note that ResNet50 is a more modern, powerful feature extractor that has 27 million parameters relative to 138 million parameters for VGG16.

4.1.2 Mask Branch

For multi-class affordance mask predictions, the mask branch in Mask R-CNN was modified such that RoIs are tiled, or duplicated, to equal the number of affordance labels for each object (Fig. 4.2). That is, affordance labels are coupled to an object class as affordance labels are static for each object based on a pre-defined action primitive. For instance, the handle of a mallet can be grasped to pound a nail into a wall. Binary cross entropy (BCE) loss can then be used to classify each pixel within each RoI as each mask contains only one affordance label. A multi-class affordance mask is later constructed by concatenating

each binary mask with its associated affordance label. Note that there may be overlap in the final affordance mask with a poor prediction.

The approach in this work is different to AffordanceNet, which predicts multiple affordance labels within each RoI. This work requires multiple resizing operations, which can be time consuming for each object, but additional complexity is introduced in AffordanceNet with masks that contain more than one affordance label. Note that a similar runtime is achieved in this work as compared to AffordanceNet.

As RoIAlign resizes a feature map to low resolutions, such as 7x7 or 14x14, segmenting multiple affordance labels is challenging, which increases in difficulty with a greater number of affordance labels. AffordanceNet leverages three blocks of a convolution layer, Rectified Linear Unit (ReLU) non-linear activation, and deconvolution layer to learn features in the mask branch (Fig. 4.1). The three deconvolution layers upsample 7x7 low resolution masks to 244x244 high resolution masks. However, deconvolutional layers are memory intensive, which may limit a practitioner's ability to implement AffordanceNet. For instance, it was observed that due to a lack of memory with one Nvidia GeForce RTX 2080, three deconvolutional layers for upsampling could not be used in this work. Instead, four blocks of a convolution layer and ReLU non-linear activation and one deconvolutional layer were implemented to learn features in a mask branch (Fig. 4.2).

4.1.3 Loss Functions

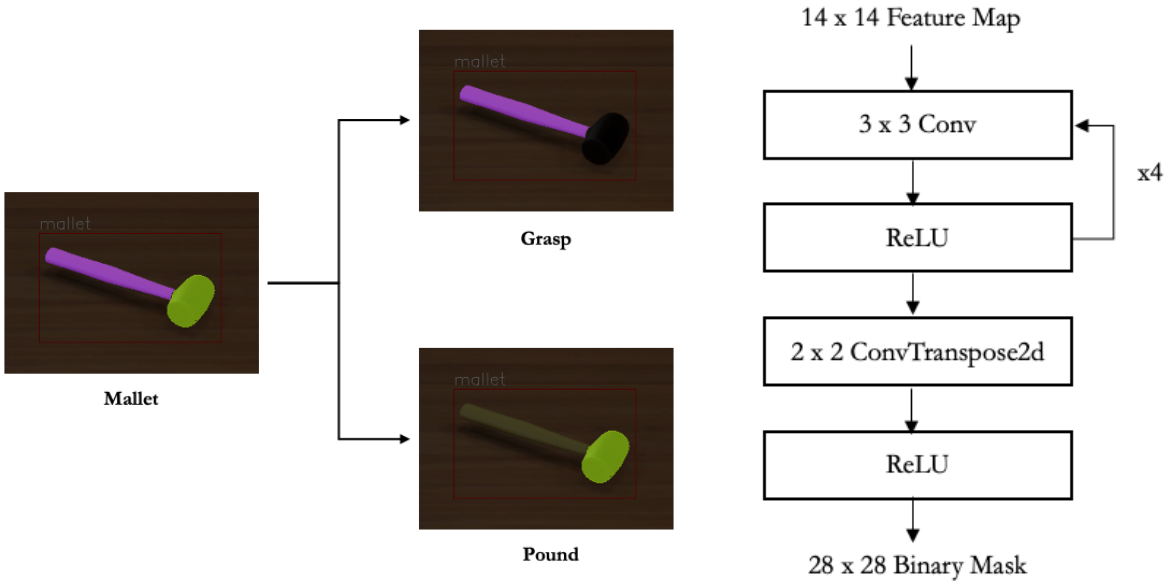
Similar to Mask R-CNN [10] and AffordanceNet [6], a multi-task loss was used to jointly train the object class, object location and mask branch:

$$L = L_{\text{object class}} + L_{\text{bounding box}} + L_{\text{mask}} \tag{4.1}$$

A multinomial cross entropy (CE) loss was used to classify the $C_{obj} + 1$ object classes, $L_{\text{object class}}$, for each RoI. Losses for bounding box regression, $L_{\text{bounding box}}$, and segmentation, L_{mask} were then computed only for positive RoIs, which the network has classified as having a foreground object. The smooth L1 loss was used for the bounding box regression offsets [33]. Lastly, BCE was used for the mask loss with object or affordance labels.

4.1.4 Evaluation Metric

Performance on the UMD dataset was benchmarked using the weighted F_B^w-measure [45], which is a widely used metric for affordance detection [6, 40, 46]. The weighted F_B^w-



(a) Tiling RoIs for Segmenting Affordance Labels (b) Overview of the Mask Branch

Figure 4.2: Overview of the Mask Branch within Mask R-CNN for Object Segmentation and Affordance Detection

measure captures the well-known trade off between precision and recall and applies a weighting function to errors that incorporates a dependency between pixels and location of errors in (4.2). Note that β is typically set to 1, which controls the trade-off between complete-detection (i.e. high precision) and over-detection (i.e. high recall), see [45].

$$F_B^w = (1+\beta^2)\frac{\text{Precision}^w + \text{Recall}^w}{\beta^2 \cdot \text{Precision}^w + \text{Recall}^w} \tag{4.2}$$

4.1.5 Implementation Details

Mask R-CNN for object segmentation and affordance detection was trained using stochastic gradient descent with 0.9 momentum and 1e-4 weight decay for 20 epochs. The learning rate was initially set to 1e-3 and decreased by a factor of 10 at epochs 3 and 5. All input images were resized to a minimum and maximum size of 600 and 1000, respectively. Images were also augmented with the same pipeline of affine flips (left/right and up/down), cropping and Gaussian blurring. Similar to AffordanceNet, 15 anchors in the RPN were

used and the top 2000 RoIs were selected for computing the mutli-task loss. Non-maximum Suppression (NMS) was applied to RoI proposals with a threshold of 0.7 and RoIs must have an Intersection over Union (IoU) of 0.5 to be considered a foreground object. The top 1000 RoIs were selected during inference and object classification scores were thresholded at 0.9 and binary masks at 0.5. Lastly, COCO pre-trained weights provided by Torchvision were leveraged to achieve state-of-the-art results.

4.1.6 UMD Dataset

Before presenting results on the ARL AffPose Dataset, the UMD dataset was first used to validate Mask R-CNN for affordance detection. Results are presented in Table 4.1, such that this work attributes the small gain in performance (0.6%) to a more powerful feature extractor, as VGG16 in AffordanceNet [6] was replaced with ResNet50 with a FPN. Recent work [47] has investigated performance for affordance detection with several powerful feature extractors. It is expected than using an even more powerful backbone feature extractor, such as ResNet101, can further increase performance. That said, oftentimes generalization and false positives on novel objects are of greater concern than accuracy when deploying trained models for object detection in the real world [48]. It can also be difficult for a practitioner to deploy state-of-the-art DL frameworks that are memory intensive on constrained hardware such as a robotic platform.

Runtime for the implementation of Mask R-CNN for affordance detection in this thesis was 124 ms relative to 150 ms for AffordanceNet [6]. As the two networks were tested on different hardware, it is difficult to say which is faster, but the implementation in this thesis consumes less memory as masks are not upsampled to higher resolutions. Note that this work did not observe any noticeable improvement in performance by upsampling binary masks to higher resolutions such as 112x112.

As previously mentioned in Section 2.6, the UMD dataset contains pixel-wise hand annotations. Such a process is not only extremely time consuming, but resulting masks are of low quality relative to predictions using Mask R-CNN (Fig. 4.3). Further, the UMD dataset only contains clutter-free scenes which is a convenient way for researchers to frame the affordance detection problem. Many real-world robotic applications, such as the Amazon Picking Challenge [1], contain scenes with heavy clutter and occlusion. This work aims to address the lack of a large, annotated benchmark dataset with cluttered scenes for affordance detection by generating the ARL AffPose dataset.

Table 4.1: Comparison of Results with the Weighted F_B^w-measure on the UMD Dataset. Two key differences exist between AffordanceNet and Mask R-CNN for affordance detection. First, a modified mask branch, and second, VGG16 is replaced with ResNet50 with a FPN.

	AffordanceNet [6]	**Mask R-CNN**
grasp	73.1	75.0
cut	76.2	74.7
scoop	79.3	76.8
contain	83.3	84.0
pound	83.6	84.7
support	82.1	84.5
wrap-grasp	81.4	83.8
Mean	79.9	80.5

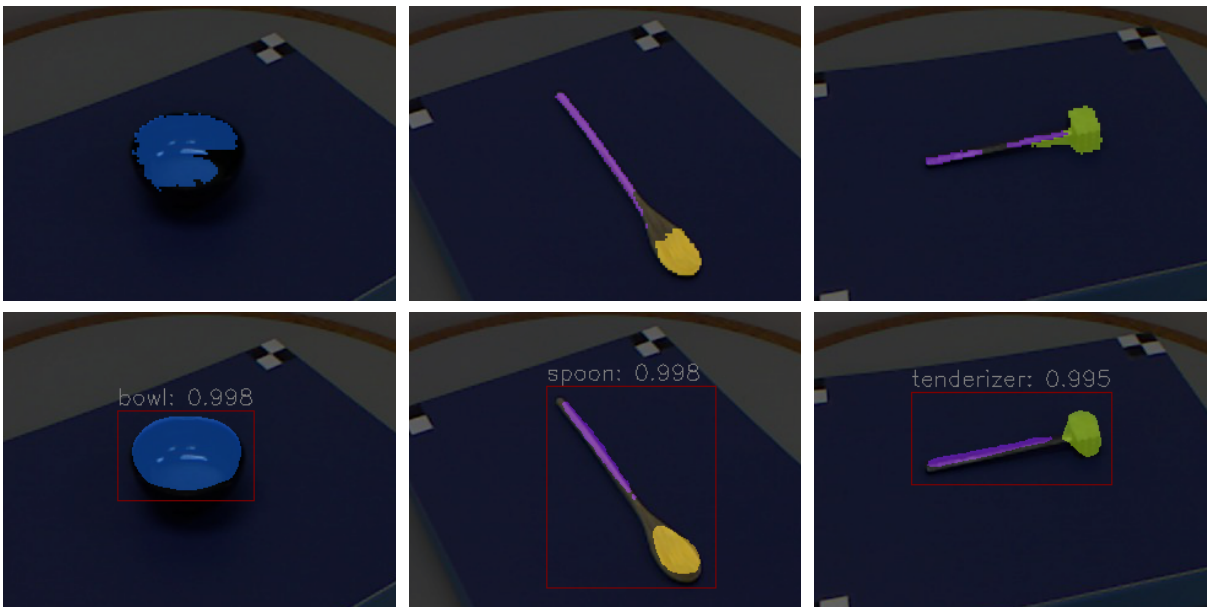


Figure 4.3: Examples of Ground Truth Masks and Predicted Masks using Mask R-CNN for Affordance Detection on the UMD Dataset. The colour of the mask represents the affordance label. Grasp: purple; scoop: yellow; pound: light green; contain: dark blue.

4.1.7 ARL AffPose Dataset

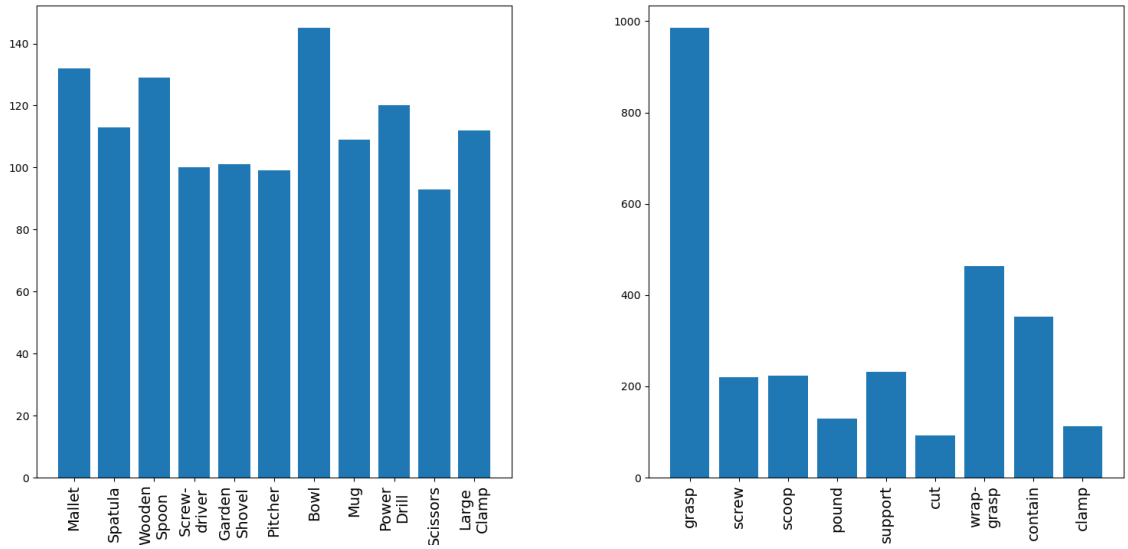
Object Segmentation vs Affordance Detection

Results for object segmentation on the ARL AffPose Dataset are first presented in Table 4.2. Note that in general performance with the 6 YCB objects was greater than performance with the 5 household objects. Mask R-CNN had difficulties detecting darker objects, such as the screwdriver on the TV stand (Fig. 4.5), which has the lowest F_B^w score. It is assumed that darker objects, such as the spatula, screwdriver, and garden shovel, have lower F_B^w scores as they blended into darker backgrounds in the ARL AffPose dataset.

Table 4.2: Comparison of Results with the Weighted F_B^w-measure for Object Segmentation on the ARL AffPose Dataset

	Real	**Synthetic**	**Synthetic + Real**
Mallet	82.5	22.9	81.7
Spatula	74.8	39.9	73.6
Wooden Spoon	81.1	35.6	81.7
Screwdriver	64.3	26.8	66.5
Garden Shovel	75.8	38.6	77.6
Pitcher	81.4	54.9	82.2
Bowl	92.0	76.6	92.4
Mug	92.8	78.1	93.3
Power Drill	92.7	76.0	93.2
Scissors	90.9	80.8	91.1
Large Clamp	87.6	61.8	87.0
Mean	83.3	53.8	83.7

Similarly, results for affordance detection on the ARL AffPose Dataset are presented in Table 4.3. Note that the distribution of object class labels in the ARL AffPose dataset is evenly distributed while the distribution of affordance labels is not (Fig. 4.4). Note that 9 out of 11 objects inherit a grasp affordance while only 2 objects, such as the screwdriver and power drill, inherit a screw and grasp affordance. Further, the screwdriver has poor object segmentation performance, which correlates to a low F_B^w score for the screw affordance in Table 4.3.

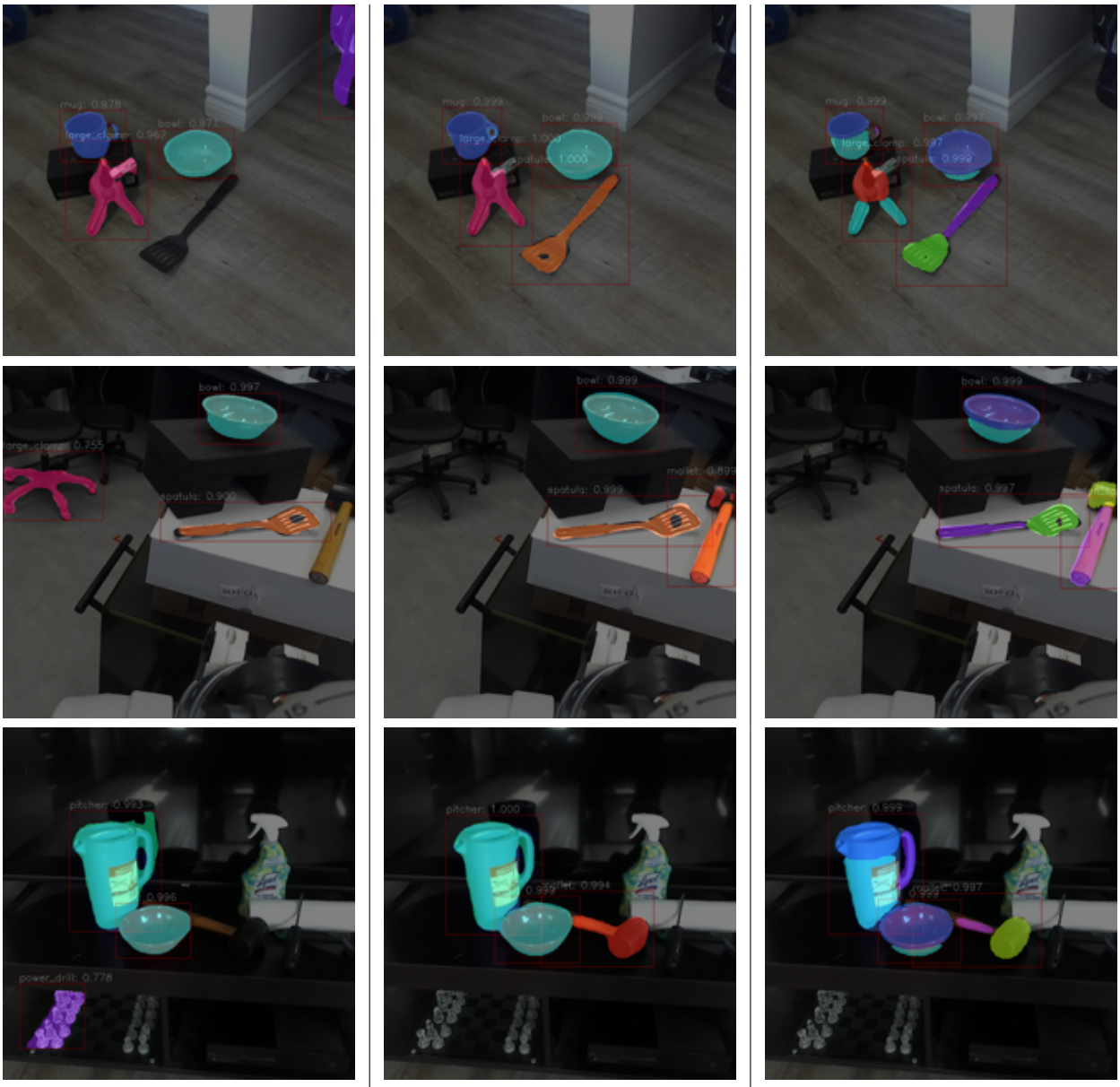


(a) Object Class Labels

(b) Affordance Labels

Figure 4.4: Distribution of Object Class Labels and Affordance Labels on the ARL AffPose Test Set

Note that no noticeable improvement in performance was observed by incorporating weights for the uneven distribution of affordance labels into BCE loss (L_{mask}). It it assumed that low performance for the screw affordance is due to a large number of instances that are hard to detect, such as the screwdriver that blends into the TV stand (Fig. 4.5). Note that Mask R-CNN will not segment the screwdriver if it does not detect a RoI for the object. Work such as RetinaNet [49] has investigated weighting easy and hard examples for training object detectors with Focal Loss, which modulates CE to reduce the loss for well classified examples. However, this mainly applies to single shot object detectors, such as YOLO [50], which have a large number (e.g. 100K) of possible object locations. In two-stage object detectors, such as Faster R-CNN or Mask R-CNN, the RPN filters candidate bounding boxes to a few thousand (e.g. 2K). As such, no noticeable improvement was observed in this work by replacing CE with Focal Loss for object classification $L_{\text{object class}}$. It is proposed that one could improve performance by implementing image processing techniques to improve the brightness of darker pixels in the ARL AffPose Dataset.



Object Seg. (Synthetic) | Object Seg. (Real) | Affordance Detection

Figure 4.5: Examples of Object Segmentation and Affordance Detection Results on the ARL AffPose Dataset. Models were trained on synthetic or real images. The colour of the mask represents the affordance label. Grasp: purple; screw: orange; scoop: yellow; pound: light green; support: green; cut: teal; wrap-grasp: light blue; contain: dark blue; clamp: red.

Table 4.3: Weighted F_B^w-measure for Affordance Detection on the ARL AffPose Dataset

	Synthetic + Real
grasp	76.5
screw	65.0
scoop	82.5
pound	82.9
support	81.7
cut	78.2
wrap-grasp	76.1
contain	85.5
clamp	83.9
Mean	79.1

Real and Synthetic Images for Object Segmentation

An ablation study for object segmentation on the ARL AffPose Dataset with real, synthetic or a mixture of synthetic and real images is first presented in Table 4.2. The backbone feature extractor was frozen when training on synthetic images, similar to previous work [51], as COCO pre-trained weights were used. Note that synthetic images were normalized to the intensity mean and standard deviation of images from the COCO dataset [42].

It was observed that performance drops by 29.5% when training on synthetic images and testing on real images when the backbone feature extractor was frozen. For a fair comparison of performance to the network trained on real images, the entire network was trained on synthetic images, but preformance dropped by an additional 6.1%. Note, that model performance was worse on certain objects, such as the mallet (Δ 59.6%), relative to other objects, such as the scissors (Δ 10.1%). When exposing real images to the model trained on synthetic images, the model failed to detect at least one target object and had one false positive in each frame (Fig. 4.5). Previous work such as [40] have leveraged domain adaptation to reduce the *reality gap* on the UMD dataset [40], but the authors still report a 25.3% drop in performance when training on synthetic UMD images and testing on the real UMD dataset.

With the above in mind, synthetic images can be leveraged for object detection in two ways. First, the network can be pre-trained on synthetic images and later be re-trained on real images, such that a small increase (0.4%) in performance is observed in Table 4.2. However, one could also fine-tune the network pre-trained on synthetic images with significantly fewer real images to reduce the annotation effort while achieving a similar level

of performance, which is similar to previous work [36]. For instance, Table 4.4 contains results with the weighted F_B^w-measure for when only 500 randomly selected real images are available for training. Note that a 19.7% increase in performance is reported when pre-training on synthetic images.

Table 4.4: Ablation Study with Weighted F_B^w-measure for Object Segmentation on the ARL AffPose Dataset with 500 Randomly Selected Real Images.

	Real	**Synthetic + Real**
Mallet	44.5	58.8
Spatula	42.1	68.2
Wooden Spoon	40.1	75.1
Screwdriver	25.5	63.0
Garden Shovel	37.1	69.8
Pitcher	51.7	80.2
Bowl	77.3	92.0
Mug	86.6	93.1
Power Drill	84.0	92.7
Scissors	85.1	91.3
Large Clamp	79.5	86.0
Mean	59.4	79.1

4.1.8 Discussion

Generalization

In real-world, robotic applications, trained models will encounter a wide variety of objects. To this end, the task of generalizing affordance detection to novel or unseen objects is challenging as this work coupled object detection with affordance labelling (Fig. 4.2). Synthetic images can aid object segmentation networks generalize as a broader range of objects, backgrounds, noise and lighting conditions can be incorporated into the training data [36]. However, different to object segmentation, the concept of affordances is abstract in nature as objects that are different in appearance may hold the same affordance labels [5]. For instance, the handles of a mallet and mug can both inherit a grasp affordance (Fig. 3.1). Object affordances are also dynamic such that they may change depending on the task at hand. For instance, the head of a wooden spoon has a primary affordance of scooping but could also inherit a secondary affordance of pounding (Fig. 3.1). The ARL

AffPose dataset maps each object part to a primary affordance label, but it is assumed that relatively little effort is required to assign secondary affordances and rankings to object parts, as compared to pixel-wise, hand annotations [8, 12]. Work such as [32] has begun to investigate affordance detection on novel objects by leveraging category-agnostic affordance segmentation. Similar to the *reality gap* on the UMD dataset [40], there is a noticeable drop in performance (15.9%) on the UMD dataset for category agnostic affordance segmentation.

4.2 6-DoF Pose Estimation with DenseFusion

4.2.1 Overview

The architecture for `DenseFusion` was publicly released in 2019 and implemented in PyTorch, which was left as is, as the objective of this work was to evaluate performance for 6-DoF pose estimation with object segmentation and affordance detection. In the DenseFusion framework, the color embedding network is a ResNet-18 encoder followed by 4 upsampling layers as a decoder (Fig 2.5). The PointNet-like architecture is a MLP followed by an average-pooling reduction function. Both the 2D and 3D feature embeddings are of dimension 128. Similarly, the iterative pose refinement module consists of 4 fully connected layers.

4.2.2 Loss Functions

The learning objective for DenseFusion is to minimize the distance between sampled points from an object mesh transformed by the ground truth $[R|t]$ and predicted $[\hat{R}|\hat{t}]$ pose:

$$\text{ADD} = \frac{1}{N}\sum_{i} \|(Rx_i + t) - (\hat{R}x_i + \hat{t})\| \tag{4.3}$$

where x_i denotes the i^{th} point of the N randomly selected 3D points from the object 3D model. Equation (4.3) is commonly used in the literature for evaluating performance for 6-DoF pose estimation [7, 11, 15, 27], which is referred to as the Average Distance (ADD) metric. However, (4.3) is only suitable for objects with a unique canonical frame, such as a power drill (Fig. 3.2). Symmetric objects, which are rotationally symmetric about an axis, can generate identical observations for more than one or possibly an infinite number of canonical frames, such as a screwdriver (Fig. 3.2). The ADD-S metric is used instead

for symmetric objects, which finds the distance between the closest point from an object mesh transformed by the ground truth $[R|t]$ and predicted $[\hat{R}|\hat{t}]$ pose:

$$\text{ADD-S} = \frac{1}{N}\sum_{i}\min_{0<k<N}\|(Rx_i + t) - (\hat{R}x_k + \hat{t})\| \tag{4.4}$$

As previously mentioned in Section 2.4, DenseFusion produces a pose estimation $[\hat{R}|\hat{t}]$ for each pixel. The per dense-pixel loss L_i^p becomes ADD for asymmetric objects in (4.3) or ADD-S for symmetric objects in (4.4). The authors of DenseFusion also introduce a confidence term c for the network to learn to balance the confidence among the per dense-pixel predictions and add a second regularization term:

$$L = \frac{1}{N}\sum_{i}\|L_i^p + c_i - w\log c_i\| \tag{4.5}$$

where N is the number of randomly selected dense-pixel features from P elements of the segment and w is a balancing hyperparameter. In (4.5), a low confidence will result in low per dense-pixel loss but high regularization, and vice versa. Note that the pose estimate with the highest confidence is used as the final output.

4.2.3 Evaluation Metrics

For all experiments, ADD or ADD-S less than 2 cm (ADD<2 cm) is reported for asymmetric (e.g. power drill) and symmetric (e.g. screwdriver) objects, respectively, as this is the maximum error to grasp an object with a 2-finger gripper as observed by [18]. Note that DenseFusion reports the ADD-S for all objects, which is an ambiguity-invariant pose error metric that fails to provide insight to rotational errors for asymmetric objects. Area Under the Curve with ADD or ADD-S thresholded at 10 cm (AUC<10 cm) is also reported similar to previous work [7, 11]. Note that the authors of PoseCNN [11] plot an accuracy-threshold curve as evaluating a pose at a fixed threshold (e.g. 2 cm) cannot reveal how a method performs for an incorrect pose with respect to that threshold.

4.2.4 Implementation Details

Adam was used as the optimizer with the learning rate set to 1e-4 and the learning rate decay set to 0.3 for 500 epochs, largely following the hyperparameters outlined in the

original work [7]. Throughout training, RGB images were normalized and random noise was added to object and object part point clouds. Lastly, w in (4.5) was selected to be to be 0.017 and 2 refinement iterations were used.

Note that for a fair evaluation of DenseFusion with object segmentation and affordance detection, 6-DoF pose was not estimated for all object parts on each object but rather to one object part that was well suited for grasping. For instance, each of the 11 objects in the ARL AffPose dataset maps to at least two object parts, with 25 object parts in total. Training DenseFusion on each object part would ultimately double the number of classes to 25 relative to the 11 classes for each object.

As previously mentioned, it is hypothesized that pose estimation for an object part is more challenging than pose estimation for an object as the former operates on a subset of RGB-D data. An object part can also be more occluded than the object itself. In this section, performance is evaluated on the YCB-Video and ARL AffPose dataset.

4.2.5 YCB-Video Dataset

Results for 6-DoF pose estimation with object segmentation and affordance detection on the YCB-video dataset are presented in in Table 4.5. Note that ground truth segmentation and affordance masks on the YCB-video dataset are used to strictly evaluate performance for 6-DoF pose estimation. It was observed that predicted masks with object segmentation or affordance detection led to a different number of missed detections, or false negatives, for certain objects. The authors of DenseFusion report results with predicted masks from an object segmentation network to compare performance against PointFusion [11] and PoseCNN [11]. It was observed that performance with AUC<10 cm drops by 1.9% when using these predicted masks relative to ground truth masks.

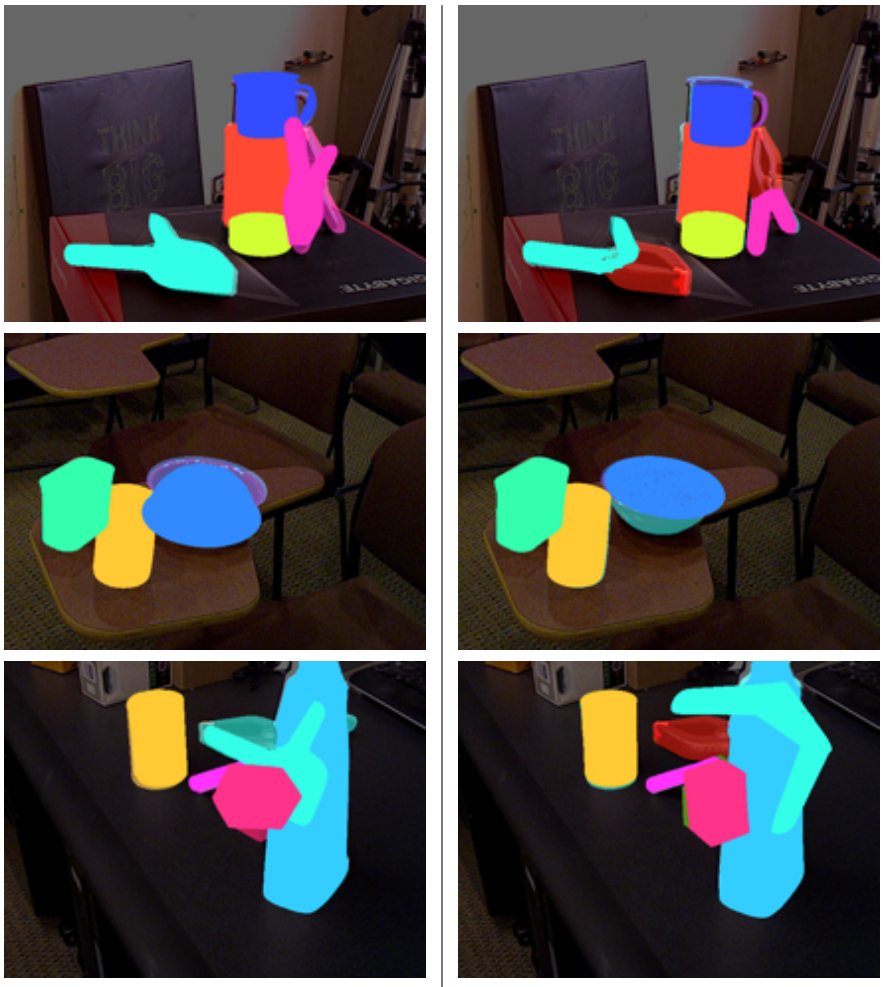
In Table 4.5, the trained model on average has slightly lower performance with affordance detection for objects that inherit more than one affordance label (e.g. pitcher). These results supports a hypothesis that pose estimation with affordance detection is more challenging than pose estimation with object segmentation. Note that the average number of masked point clouds is smaller for object parts than objects in Table 4.5. It is proposed that pose estimation with affordance detection should perform much worse than pose estimation with object segmentation when an object part is more occluded than the object itself, which is illustrated in the last row of Figure 4.6.

It was observed that the trained model can also be more confident in certain objects, such as the bowl and large clamp, which results in higher accuracy for these objects in Table 4.5. Masked point clouds with affordance detection could have greater variability

in terms of shapes and number of points, which allows DenseFusion to be more confident in class predictions (i.e. improved class separability). For instance, pose estimation with affordance detection predicts a more accurate estimate for the large clamp and bowl relative to pose estimation with object segmentation in Figure 4.6.

Table 4.5: Comparison of Results with number of points for masked point clouds (PCD), confidence of per-pixel predictions (c), AUC<10 cm and ADD<2 cm for 6-DoF Pose Estimation with Object Segmentation and Affordance Detection on the YCB-Video Dataset. Object names in bold are symmetric and object names in italic inherit a single affordance label.

	Object Segmentation				**Affordance Detection**			
	PCD	c	AUC	<2 cm	PCD	c	AUC	<2 cm
002_master_chef_can	15402	0.510	70.5	68.5	15384	0.617	71.2	70.4
003_cracker_box	22918	0.532	94.2	96.3	22916	0.453	93.9	98.7
004_sugar_box	18695	0.801	97.2	100.0	18655	0.832	97.7	100.0
005_tomato_soup_can	9857	0.858	88.1	84.7	9855	0.896	90.1	87.8
006_mustard_bottle	18773	0.506	92.2	91.6	18772	0.609	94.3	94.4
007_tuna_fish_can	4700	0.619	84.6	72.2	4698	0.607	81.8	67.9
008_pudding_box_16k	9831	0.629	96.8	100.0	9830	0.430	93.6	92.1
009_gelatin_box	11162	0.993	98.7	100.0	11161	0.997	98.2	100.0
010_potted_meat_can	7909	0.704	86.8	84.7	7908	0.698	85.7	81.6
011_banana	8105	0.376	80.8	82.1	8104	0.449	76.7	79.7
019_pitcher_base	46460	0.554	97.8	99.8	30468	0.523	91.6	88.8
021_bleach_cleanser	21192	0.550	94.1	96.1	21191	0.608	94.7	96.9
024_bowl	15147	0.391	89.9	99.8	13546	0.918	95.3	98.3
025_mug	8425	0.655	95.4	98.9	6274	0.334	91.4	94.2
035_power_drill	18250	0.695	96.7	99.8	3817	0.864	95.3	99.6
036_wood_block	13440	0.380	95.0	100.0	13435	0.460	96.6	100.0
037_scissors	3466	0.310	86.4	86.2	3464	0.556	85.7	82.9
040_large_marker	2561	0.659	93.5	99.9	2561	0.766	94.8	100.0
051_large_clamp	6498	0.597	91.6	99.9	2295	0.933	97.5	100.0
052_extra_large_clamp	7507	0.376	92.1	97.4	2931	0.858	93.9	94.6
061_foam_brick	6604	1.000	95.9	100.0	6603	1.000	92.9	100.0
Mean	13186	0.605	91.3	93.2	11137	0.686	91.1	91.8



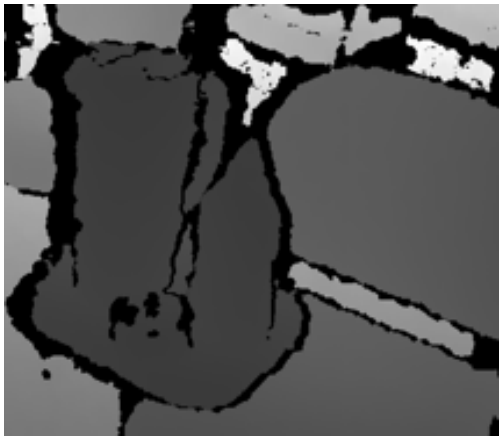
Object Segmentation | Affordance Detection

Figure 4.6: Examples of 6-DoF Pose Estimation with Object Segmentation or Affordance Detection on the YCB-Video Dataset

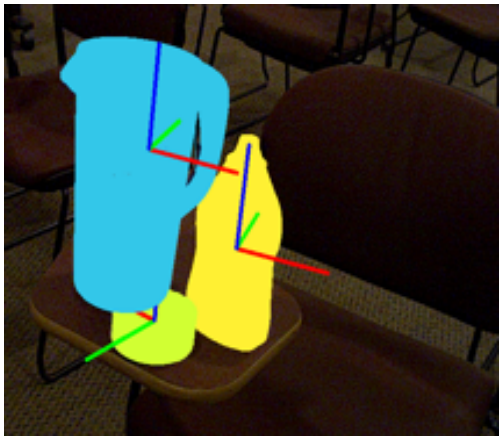
4.2.6 ARL AffPose Dataset

Similarly, for the ARL AffPose dataset it was observed that pose estimation with affordance detection resulted in lower accuracy relative to pose estimation with object segmentation in Table 4.6. Note that ground truth segmentation and affordance masks were used again to evaluate the two methods for pose estimation.

Upon comparing values in Table 4.6 and Table 4.5, it was observed that performance with the same 5 YCB objects was worse on the ARL AffPose dataset relative to results on the YCB-Video dataset. Note that point clouds for these 5 YCB objects with the ARL AffPose dataset contain fewer points in Table 4.6. In the DenseFusion framework, masked point clouds are obtained using a depth image and a corresponding segmentation mask. Note that the YCB-video dataset was generated with an Asus Xtion RGB-D sensor, which is an infrared sensor, whereas a Stereolab ZED camera, or stereo-vision, was used to generate the ARL AffPose dataset. The qualitative differences between depth images with an infrared sensor and stereo-vision is displayed in Figure 4.7. To this end, the drop in performance on the ARL AffPose dataset was attributed to smaller masked point clouds. The mean translation error, which is the l_2 norm of the ground truth and predicted translation vector, was 0.701 cm on the YCB-video dataset (Fig. 4.6) and 2.148 cm on the ARL AffPose dataset (Fig. 4.8) for pose estimation with object segmentation. Note that there also exists a larger difference in the number of points for masked point clouds with affordance detection on the ARL AffPose Dataset (47%) relative to the YCB-Video dataset (16%), which led to a larger drop in performance.

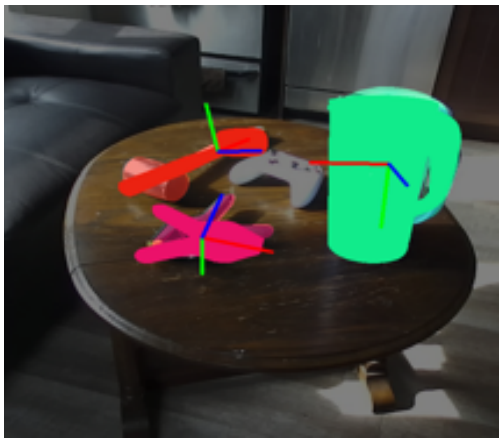


(a) Asus Xtion Depth Image



(b) Predictions for 6-DoF Pose Estimation on the YCB-Video Dataset

(c) Stereolab ZED Depth Image



(d) Predictions for 6-DoF Pose Estimation on the ARL AffPose Dataset

Figure 4.7: Examples of Depth Images and Predictions for 6-DoF Pose Estimation on the YCB-Video and ARL AffPose Dataset

Real and Synthetic Images for 6-DoF Pose Estimation with Object Segmentation

An ablation study for pose estimation with object segmentation is also presented in Table 4.7. However, different to training Mask R-CNN, DenseFusion was trained on either real,

Table 4.6: Comparison of Results with number of points for masked point clouds (PCD), confidence of per-pixel predictions (c), AUC<10 cm and ADD<2 cm for 6-DoF Pose Estimation with Object Segmentation and Affordance Detection on the ARL AffPose Dataset. Object names in bold are symmetric.

	Object Segmentation				**Affordance Detection**			
	PCD	c	AUC	<2 cm	PCD	c	AUC	<2 cm
Mallet	7472	0.470	83.6	66.5	3704	0.544	73.3	61.7
Spatula	3679	0.346	63.6	54.8	1701	0.261	28.5	7.8
Wooden Spoon	9393	0.257	60.6	38.6	4696	0.351	31.2	0.9
Screwdriver	1672	0.419	88.3	92.8	1518	0.327	88.3	90.4
Garden Shovel	12583	0.346	72.2	47.6	4313	0.476	50.1	31.7
Pitcher	21964	0.358	68.3	19.0	14176	0.382	76.4	31.2
Bowl	9044	0.654	91.2	98.3	7191	0.749	89.9	97.7
Mug	5671	0.324	72.2	16.7	2467	0.363	35.0	0.0
Power Drill	7789	0.468	81.5	66.2	1844	0.590	73.5	20.3
Scissors	5026	0.392	77.2	49.1	4313	0.510	74.4	49.1
Large Clamp	5566	0.654	87.8	97.9	2051	0.446	87.2	96.9
Mean	8169	0.426	76.9	58.9	4361	0.454	64.3	44.3

synthetic or a combined dataset of real and synthetic images without pre-training. It was observed that training with a combined dataset of real and synthetic images resulted in the best performance relative to training with either real or synthetic images in Table 4.7. Note that performance for DenseFusion does not degrade to the same extend as Mask R-CNN when training on synthetic images and testing on real images.

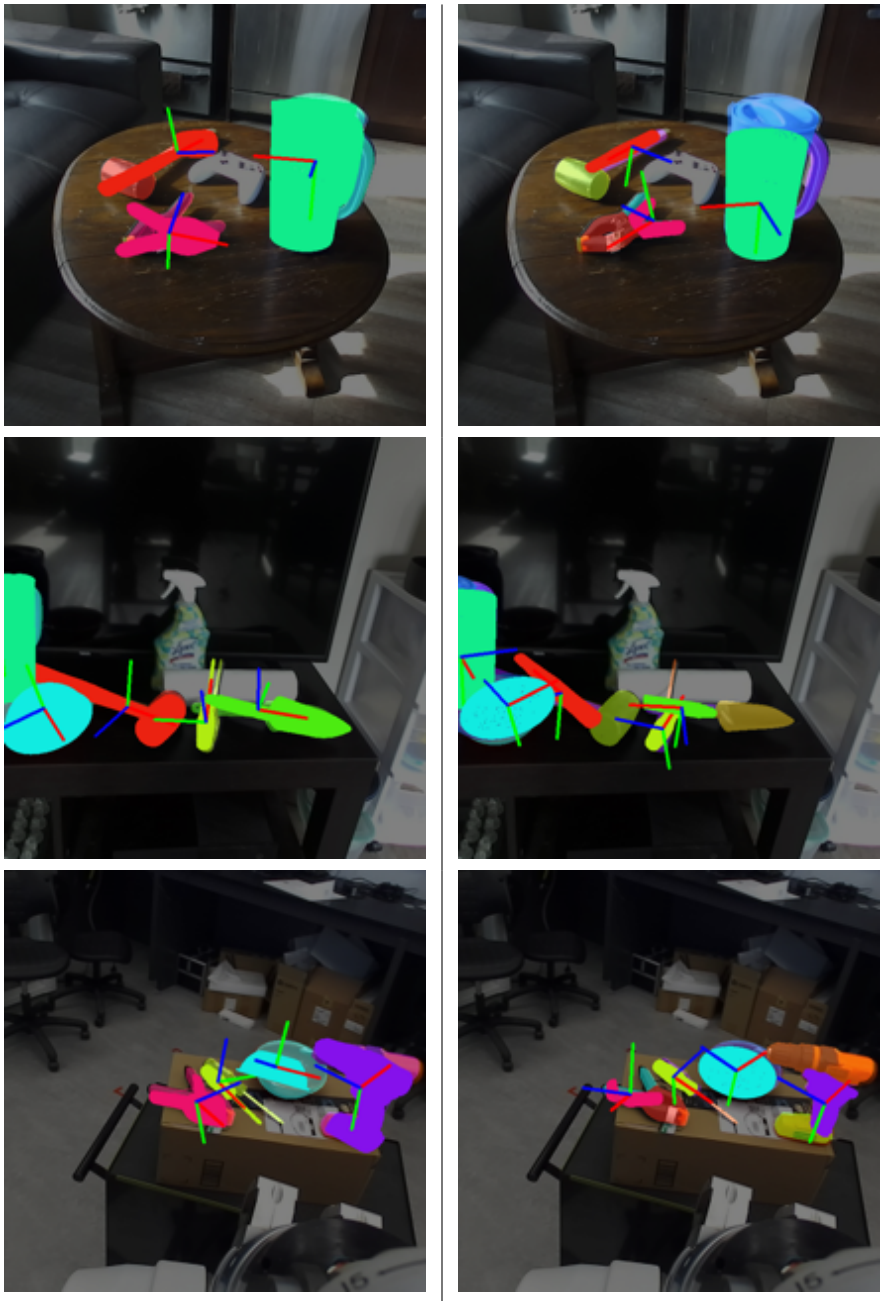
Table 4.7: Ablation study for Pose Estimation with Object Segmentation on the ARL AffPose Dataset with either Real, Synthetic or Real + Synthetic Images

	Real		**Synthetic**		**Real + Synthetic**	
	AUC	<2 cm	AUC	<2 cm	AUC	<2 cm
Mallet	76.2	63.7	75.9	56.3	83.6	66.5
Spatula	42.2	27.7	38.5	3.2	63.6	54.8
Wooden Spoon	45.3	18.4	42.3	18.9	60.6	38.6
Screwdriver	79.2	74.5	84.1	68.9	88.3	92.8
Garden Shovel	38.0	7.7	40.8	0.6	72.2	47.6
Pitcher	41.3	0.0	69.7	28.1	68.3	19.0
Bowl	75.4	8.1	93.2	96.6	91.2	98.3
Mug	77.1	52.9	62.1	3.1	72.2	16.7
Power Drill	56.7	18.9	81.9	62.8	81.5	66.2
Scissors	74.1	46.1	68.4	48.3	77.2	49.1
Large Clamp	85.0	81.5	86.2	97.3	87.8	97.9
Mean	67.5	44.0	62.8	36.3	76.9	58.9

4.2.7 Discussion

Pose Estimation with Object Segmentation and Affordance Detection

From the results presented in Tables 4.5 and 4.6, it is clear that pose estimation with object segmentation outperforms pose estimation with affordance detection. As previously mentioned, this is the expected result as pose estimation with affordance detection operates on a subset of RGB-D data. Performance for DenseFusion with affordance detection can also degrade if an object part is severely occluded (Fig. 4.6). It was also observed that performance for DenseFusion is dependent upon the size of masked point clouds or quality of an input depth image (Fig. 4.7). However, it is proposed that the true advantage of pose estimation with affordance detection is realized when the trained model is deployed on a robotic platform to grasp complex objects, which is outlined in the next chapter.



Object Segmentation | Affordance Detection

Figure 4.8: Examples of 6-DoF Pose Estimation with Object Segmentation or Affordance Detection on the ARL AffPose Dataset

Chapter 5

Experimental Evaluation

In this chapter, DenseFusion with object segmentation and affordance detection is deployed on a robotic platform to experimentally validate performance. As such, experiments in this chapter were designed to test the accuracy of pose estimation and see if it is sufficient for robotic grasping.

Grasp success rate was used to quantitatively evaluate experiments such that a grasp was considered successful if the robot can lift an object up 10 cm without dropping it. The aim of these experiments was to demonstrate a similar grasp rate for pose estimation with affordance detection relative to pose estimation with object segmentation. As previously mentioned, pose estimation with affordance detection is a more challenging task than pose estimation with object segmentation but it may improve performance for grasping of complex objects.

5.1 Object Pose for Grasping

The different coordinate frames and transforms that are required to grasp an object are discussed in this section. First, 6-DoF pose estimation frameworks, such as DenseFusion [7], compute the object pose with respect to the camera frame T_C^O. However, to grasp an object, the 6-DoF pose of the object with respect to the base link frame of a manipulator T_B^O is required.

Figure 5.1 displays a 7-DoF Barrett WAM arm and 8-DoF BarrettHand that was used throughout grasping experiments. Note that a BarrettHand is a three-finger gripper, which allows for greater translation errors as compared to two-finger grippers in the literature

[7, 18]. Note that [18] observed that the estimate of the object pose must be accurate within +/- 2 cm to successfully grasp an object with a two-finger gripper. The distance from the centroid of the Barretthand to the tip of each finger was measured to be 17.75 cm.

5.1.1 Coordinate Transforms

Four main coordinate frames were used to grasp an object (Fig. 5.1): 1) base link of the manipulator F_B, 2) camera frame F_C, 3) object frame F_O, and 4) end effector or target frame F_T. Coordinate frames for the Barrett WAM arm and BarrettHand are defined by the manufacture in the Unified Robot Description Format (URDF) file. The camera frame as defined by the manufacture was also used. However, how the camera was mounted was left as a design exercise.

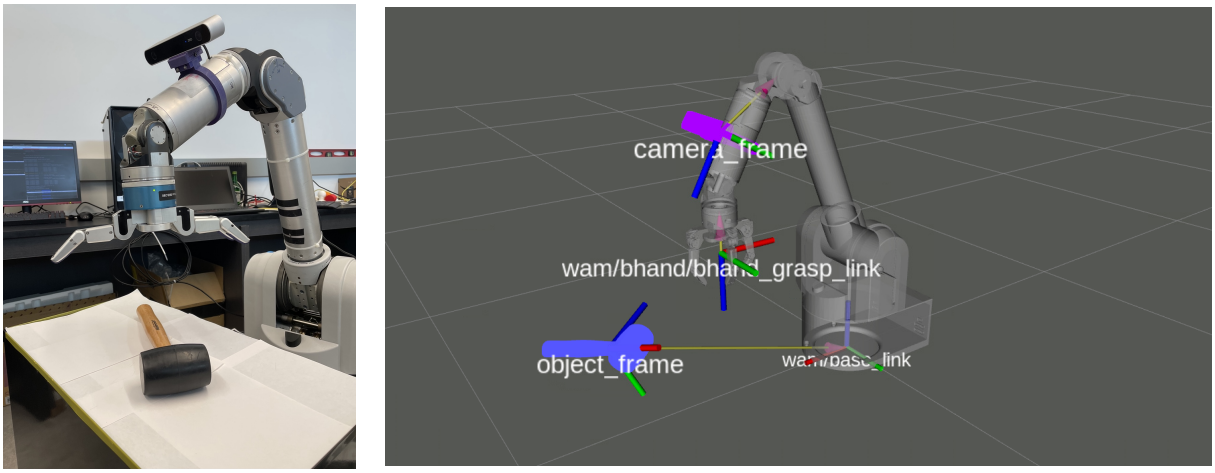


Figure 5.1: Illustration of Different Coordinate Frames for Grasping an Object

Camera Mount

In general, a camera for a robotic platform can be mounted in two ways: 1) fixed on a static mount or 2) directly mounted on the manipulator. For a static mount, the transform from the base link of the manipulator to the camera frame T_B^C is static, but the manipulator can block the camera's line of sight. One can address this issue by mounting the camera directly on the manipulator, but the transform from the base link to the camera frame T_B^C is no longer static. The dynamic transform can be computed by first measuring the

static transform from the camera frame to the nearest joint on the manipulator $T_{J_i}^C$, which was the forearm link in this work. Forward kinematics can then be used to compute the transform from the base link to the forearm link $T_B^{J_i}$ given the current configuration of the manipulator:

$$T_B^O = \underbrace{T_B^{J_i} \cdot T_{J_i}^C}_{T_B^C} \cdot T_C^O = T_B^C \cdot T_C^O \tag{5.1}$$

In this work, the ZED camera was mounted on the forearm of the manipulator to emulate training images during inference in terms of distance to object and angle of attack for the camera (Fig. 5.1). Note that it was assumed that the robot dynamics did not change significantly.

End Effector

Lastly, the end effector frame should be aligned with an object frame to grasp an object in (5.2).

The steps that are required to grasp the mallet are outlined to better understand how this may work (Fig. 5.1). Note that the canonical frame of the mallet was defined such that the x-axis points forward, y-axis down, and z-axis to the left (Fig. 3.1). A top-down grasp strategy would first align the normal vector of the end effector (i.e. z-axis) to the normal vector of the mallet (i.e. y-axis). A rotation about the yaw axis can then be applied to align the fingers of the BarrettHand (i.e. x-axis) with the perpendicular axis of the mallet (i.e. z-axis). Note that all mesh files in this work were modified in Blender such that the x-axis points forward, y-axis down, and z-axis to the left (Fig. 1.1).

Inverse Kinematics (IK) can then be used to determine the joint positions, velocities or torques to achieve the desired state of the end effector once the desired transformation for grasping an object T_B^T has been computed. Note that the intelligent control and path planning of the manipulator lies outside the scope of this work.

$$T_B^T = T_B^C \cdot T_C^O \cdot T_O^T \tag{5.2}$$

5.2 System Overview

The pipeline for pose estimation, which was developed in this work, was deployed using Robot Operating System (ROS) Melodic on Ubuntu 18.04 (Fig. 5.2). Experiments began by first mounting the Barrett WAM arm on the Robotnik Summit XL mobile base, which was kept stationary. The Summit XL's ethernet was used to setup a wireless ROS network and the arm was controlled by connecting the workstation to the WAM via a Controller Area Network (CAN) bus. Note that both the ZED camera and Barrett WAM arm were calibrated using methods provided by each manufacture.

WAM Arm

`barrett-ros-pkg` was used to interface with the Barrett WAM arm, which is a ROS wrapper for `libbarrett`, a real-time controls library written in C++. The WAM node exposes functionality to control the arm via a suite of ROS topics and services. The stack can be ran on the WAM's on-board PC or on an external PC that is connected via CAN.

Transform Publisher

The ZED camera, which was mounted on the forearm of the Barrett WAM arm, was deployed using the `zed-ros-wrapper`. To transform the object pose with respect to the base link of the manipulator in (5.1), the static transform between the camera frame and the forearm link was measured. Offsets in the x and y-axis were known as the camera mount was designed in this work. However, it was difficult to measure the exact distance along the z-axis from the camera frame to the forearm link when the arm was fully extended. Several tests were conducted using an ArUco marker, which is a binary square fiducial marker that is used in robust pose estimation [52], to fine-tune the static transform.

Pose Estimation

The ROS node for pose estimation subscribed to the ZED Camera and operated on RGB and depth images in a synchronized callback. Once an estimate of the object pose T_C^O was known, it was transformed to the object pose with the with respect to the base link of the manipulator T_B^O and later published. Results for runtime performance of the deployed pipeline are presented in Table 5.1. Note that the deployed pipeline with ROS melodic required Python 2.7, which was different to Python 3.5+ used throughout training Mask R-CNN and DenseFusion.

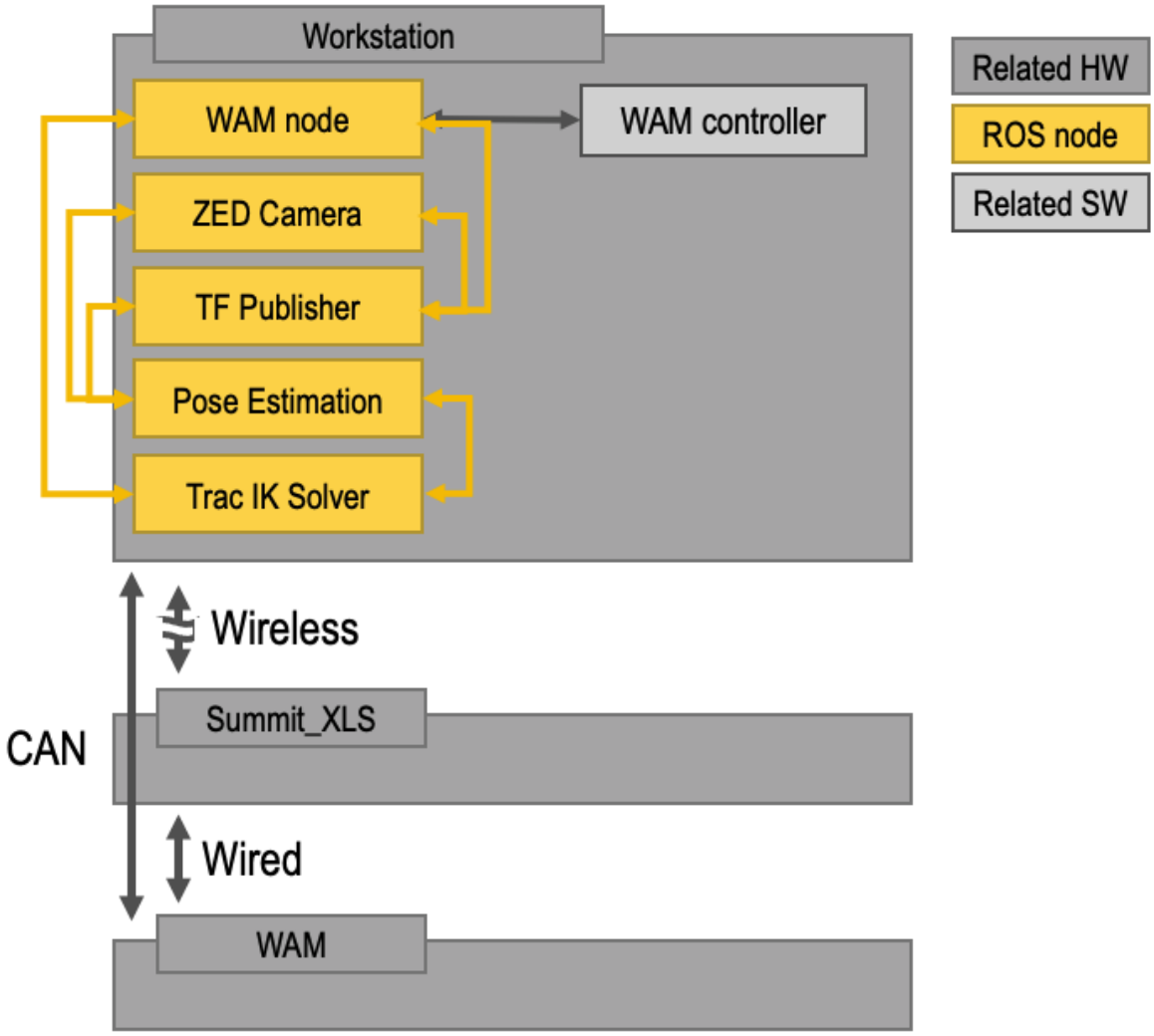


Figure 5.2: Overview of the System Architecture used throughout Grasping Experiments

Table 5.1: Runtime breakdown in ROS

Object Segmentation	DenseFusion	Combined
0.15 [s]	0.08 [s]	0.23 [s]

Trac IK

Lastly, a ROS node was developed to control the 7-DoF Barrett WAM arm using IK. `trac-ik` was selected as a IK solver as it capable of controlling a redundant manipulator such as the 7-DoF Barrett WAM [53]. In a callback subscribed to the Pose Estimation node, an IK solver was used to compute joint positions that transforms the end effector from a given state to a desired state, which are defined in Cartesian coordinates. However, blindly commanding the robot to the desired end state can lead to unexpected errors in cluttered scenes and simple path planning was required.

Two grasp strategies were executed throughout experiments: 1) top-down grasps and 2) forward grasps. A sequence of commands to execute a top-down grasp would be as follows:

1. Move to a pre-defined capture position,
2. Identify the object 6-DoF pose in a single-frame,
3. Align the end effector frame to the object frame,
4. Compute joint positions using IK to achieve desired end state,
5. Move to a position defined to be 10 cm above the object,
6. Move directly downwards and grasp the object,
7. Move 10 cm directly upwards to test for a successful grasp.

5.3 Grasping Experiments

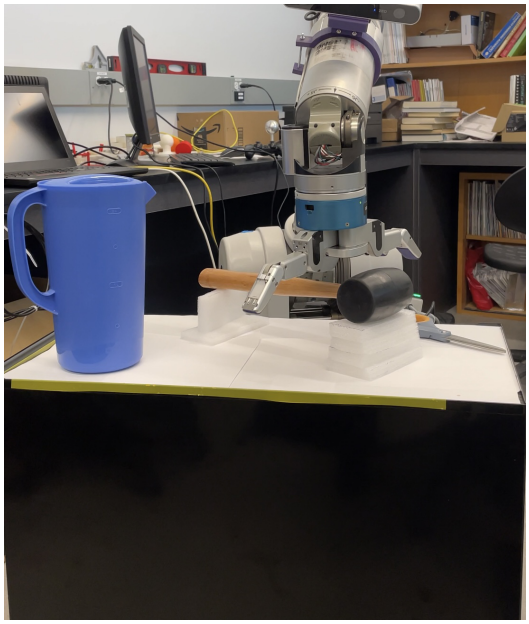
To quantitatively evaluate pose estimation in the context of robotic grasping, a scene was created with 1 target object on a workbench along with 2 distractor objects (Fig. 5.3). Each target object was placed at 4 different locations on the workbench and 3 orientations

were used at each location, similar to previous work [7, 18]. As the chosen framework for pose estimation operates on a single RGB-D frame, each grasp was repeated 3 times, for a total of 36 grasps for each object. Note that a Kalman filter [54] could be implemented to reduce noisy estimates of the object pose across several RGB-D frames as it is assumed that objects remain static in the scene, similar to [27]. However, a Kalman filter was not implemented in this work as the objective was to strictly evaluate pose estimation with object segmentation and affordance detection.

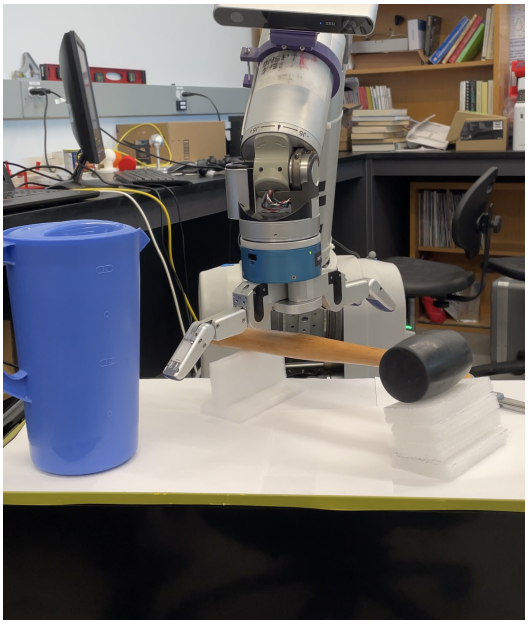
The scene used throughout grasping experiments was similar to a low number (10 of 71) of real scenes in the ARL AffPose Dataset (Fig. 5.3). The workbench was covered with white printer paper in an effort to mask the foam padding used to allow the manipulator to physically grasp objects. The pre-defined capture position of the WAM was set such that the scene during inference was similar to scenes during dataset generation in terms of the distances to objects and the angle of attack for the camera. A qualitative drop in performance was observed with drastic changes to the distance to each object and/or viewpoint, was was expected. Additional scenes can be included throughout dataset generation to improve such results.

A subset of 4 objects were selected from the 11 objects in the ARL AffPose dataset for grasping experiments (Fig. 3.1). The mallet, spatula, garden shovel and power drill all inherit noticeable differences between the centroids of the canonical frames for objects and object parts. Note that the wooden spoon was not included as it has similar shape and size to the mallet. Objects that are rotationally symmetric, such as the screwdriver, were also not included in grasping experiments. Such objects are prone to large orientation errors with pose estimation as they can generate identical observations from multiple viewpoints. Note that with an assumption of planar grasping, the screwdriver can be grasped more reliably by first aligning the normal vector of the end effector to be perpendicular to the workbench. Afterwards, a pose estimate can be used to apply a rotation about the yaw to align the fingers of the gripper to be perpendicular to the longitudinal axis of the screwdriver (Fig. 3.1).

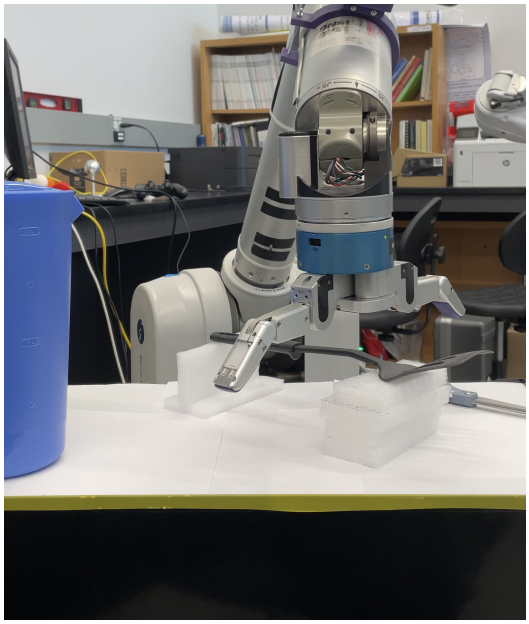
As previously mentioned, two grasping strategies were implemented: 1) top-down grasps for the mallet, spatula and garden shovel and 2) forward grasps for the power drill. For forward grasps, the robot was first commanded to a position defined to be 10 cm in front of the object and later commanded to move directly forward to grasp an object. Note that the IK solver failed to compute joint positions in all but one location for forward grasps of the power drill as the object was positioned too close to the base of the manipulator. It was decided that the position of the workbench should not change, to change the distance from the camera to the power drill, as this could adversely affect accuracy for pose estimation. Forward grasps ultimately require more intelligent path planning in cluttered scene



Mallet Object Centroid



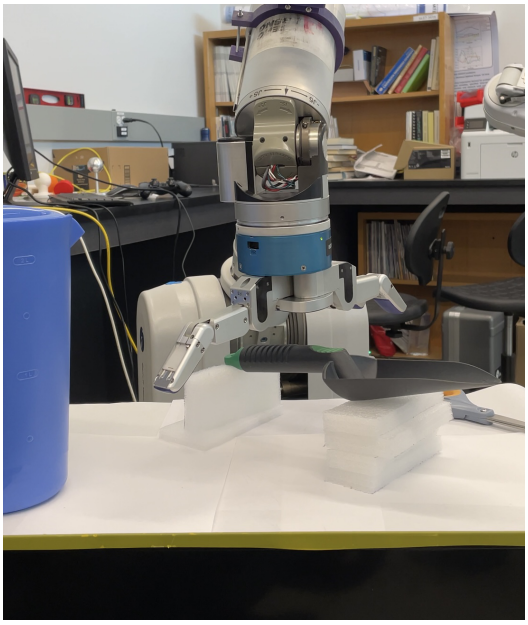
Mallet Object-part Centroid



Spatula Object Centroid

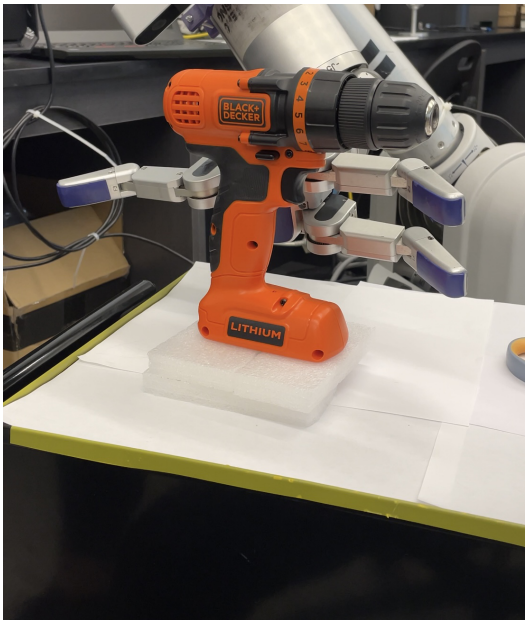
Spatula Object-part Centroid

Garden Shovel Object Centroid



Garden Shovel Object-part Centroid

Power Drill Object Centroid



Power Drill Object-part Centroid

Figure 5.3: Grasping Four Objects from the ARL AffPose Dataset with Object Segmentation (left) and Affordance Detection (right)

relative to top down grasps. Despite this, the power drill was included throughout grasping experiments as it best exemplifies how affordance detection can lead to more intelligent grasping of complex objects (Fig. 5.3).

Results from the grasping experiments are presented in Table 5.2. It is difficult to say which is the exact cause of error for each failed grasp attempt, and sources of error include error from pose estimation, error in the static transform of the camera frame, and miscalibration of the relative encoders of the Barrett WAM arm. Note that home positions for the seven relative encoders were set during calibration but the WAM arm enters an idle phase between calibration and normal operation such that the home positions can easily change. It is assumed that pose estimation was the largest source of error and several failure modes for a failed grasp attempt include translation and/or orientation errors (Fig. 5.4). It was observed that the BarrettHand allows for relatively large errors in the estimated pose of an object and results in Table 5.2 could change if a different end effector were used. As previously mentioned, the authors of DOPE [18] observed that the estimate of the object pose must be accurate within +/- 2 cm to successfully grasp an object with a two-finger gripper.

In closing, throughout grasping experiments a similar grasp success rate was demonstrated for the two methods for pose estimation. However, it was only an experimental validation of the two methods. It was observed that pose estimation with affordance detection tends to be more noisy as it operates on a subset of RGB-D data in a single frame. Future work could implement a Kalman Filter, similar to [27], to improve the robustness of the proposed pipeline.

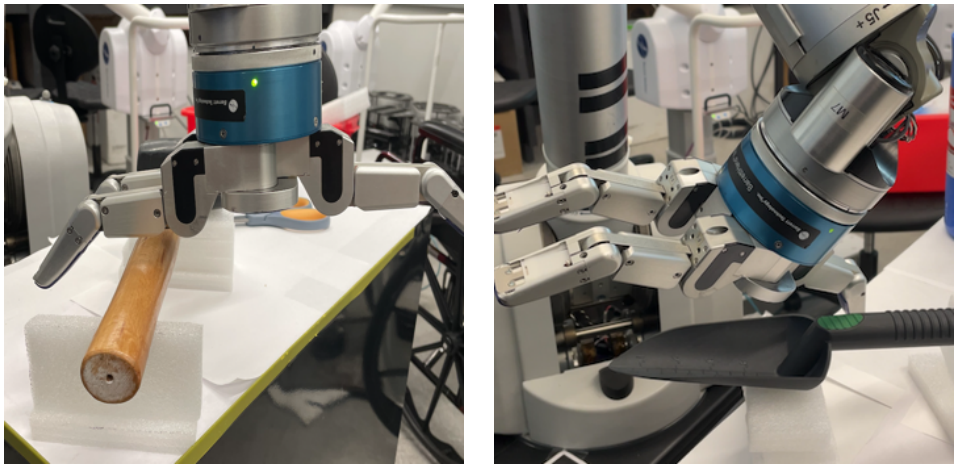
Table 5.2: Comparison of Results for Grasping Four Target Objects from the ARL AffPose Dataset.

	Mallet	**Spatula**	**Garden Shovel**	**Power Drill**
Object	36/36	33/36	32/36	7/9
Affordance	34/36	33/36	31/36	8/9

5.4 Discussion

5.4.1 Scene Understanding for 6-DoF Pose Estimation

In the context of robotic grasping, a similar result for pose estimation with affordance detection can be achieved with existing methods for pose estimation with object segmen-



Translation Error Orientation Error

Figure 5.4: Examples of Different Failure Modes Throughout Grasping Experiments

tation and a post-processing step to transform the object pose to a more optimal grasp location. Such an approach would use the same transformation, or a simple translation of the object centroid to an object part centroid (Fig. 3.3), as outlined in dataset generation in Section 3.1. It is assumed that performance for robotic grasping of complex objects should improve while performance for pose estimation should not degrade, as one would not operate on a subset of RGB-D data. If this is the case, one may question the need for affordance detection.

It is proposed that the value of affordance detection lies in generalizing to novel objects such that one would not need to apply a transformation to determine the most optimal grasp location for each object. For affordance detection on novel objects, the network produces an interesting result for the PlayStation 4 controller, which labels the handles as a grasp affordance (Fig. 5.5). However, this serves as motivation to decouple object detection from affordance segmentation, as the network would assign an object label from the 11 objects in the ARL AffPose dataset to the PlayStation 4 controller. Note that category-agnostic affordance segmentation [32] would be more appropriate in this instance.

That said, generalizing 6-DoF pose estimation frameworks to novel objects is more challenging as it was assumed in this work that the specific instance of the object 3D mesh is made available a priori. One could attempt to grasp novel objects on a planar surface using 2D outputs from a category-agnostic affordance segmentation network and a corresponding depth image, similar to [6, 32, 40]. Note that the centroid of the grasp

frame would be the mean of the grasp pixels projected into 3D space and the orientation of the longitudinal axis of the object could be determined by a line of best fit applied to the grasp pixels. However, such a solution would be limited to top-down grasps of simple objects.

Figure 5.5: Affordance Detection of Novel Objects. The colour of the mask represents the affordance label. Grasp: purple; screw: orange; cut: teal; wrap-grasp: light blue; contain: dark blue; clamp: red.

5.4.2 6-DoF Pose Estimation

With regards to the above discussion for deploying a simple solution for top-down grasps of novel objects, one may also question the need for 6-DoF pose estimation.

The answer to this question depends on how one defines their objectives. If one is only concerned with planar grasping of simple objects, one would only need to deploy an object segmentation or affordance detection network using a infrared sensor or camera with stereo vision. However, in this work, forward grasps of more complex object, such as the power drill, were demonstrated using the outputs of a 6-DoF pose estimation framework (Fig. 5.3). Further, qualitative results are included for estimating 6-DoF pose of objects that do not lie on flat surfaces (Fig. 5.6). Note that more intelligent path planning and control may be required to grasp such objects in cluttered scenes.

Lastly, objects were constrained to lie flat on the workbench throughout grasping experiments (i.e. planar grasping). It was observed that the predicted normal vector for objects were noisy relative to the gravity vector, which led to failed grasp attempts in some instances. Future work could encode a prior for gravity to improve performance for 6-DoF pose estimation.

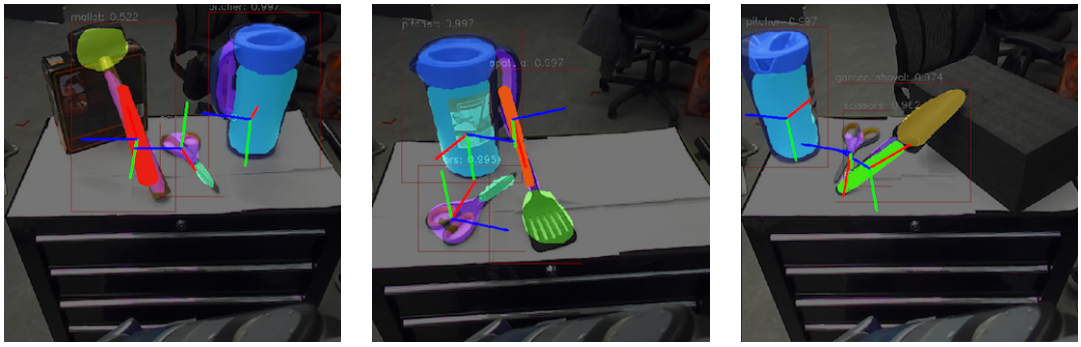


Figure 5.6: 6-DoF Pose Estimation for Non-planar Objects.

Chapter 6

Conclusion

In this thesis, an experimental evaluation of 6-DoF pose estimation with object segmentation and affordance detection was conducted. Motivation stems from the belief that pose estimation affordance detection can improve robotic grasping of complex objects.

Previous work for generating affordance labels was limited by expensive hand annotations for pixel-wise labels, which presents a challenge for training deep CNNs. In this work, real and synthetic images were auto-generated with ground truth annotations for object segmentation and 6-DoF pose. A post-processing method was developed to generate desired ground truth annotations, namely affordance labels and object part 6-DoF poses.

Mask R-CNN was implemented and trained for object segmentation and affordance detection on the UMD dataset and the ARL AffPose dataset. Performance in terms of accuracy and runtime was on par with state-of-the-art results for affordance detection. Afterwards, DenseFusion was trained on the YCB-Video dataset and the ARL AffPose dataset, such that pose estimation with object segmentation slightly outperformed pose estimation with affordance detection. It was observed that synthetic images can supplement real images to boost performance for object segmentation, affordance detection, and pose estimation, as a wide variety of object poses and a broader range of backgrounds, noise and lighting conditions can be incorporated into the training data. However, the drop in performance when training deep CNNs on synthetic images and deploying on real images will continue to decrease as rendering techniques in simulators, such as UE4, improve. Lastly, in experimentally validating this work on a robotic platform, it was demonstrated that pose estimation with affordance detection can improve robotic grasping of complex objects.

6.1 Future Work

In experimentally validating this work, it was observed that trained models are limited to objects within the ARL AffPose dataset. Generalization remains an open issue within deep learning. However, it is proposed that one could attempt to grasp novel objects on a planar surface using 2D outputs from a category-agnostic affordance segmentation network and a corresponding depth image. To this end, future work should decouple object detection and affordance segmentation, similar to [32], as objects that are different in appearance may hold the same affordance label and vice-versa.

As synthetic images are computationally inexpensive to generate, future work could incorporate techniques from adversarial learning frameworks, such as [40], to conduct grasping experiments with trained models on synthetic images. This could aid generalization for trained models, which has widespread applications for robotics in unstructured environments [1].

Additionally, future work could demonstrate how affordance detection could improve the manipubility of an object such as a mallet. For instance, if a manipulator's goal is to pound a nail into a piece of wood, it is crucial that the agent first grasps the handle of the mallet to optimally complete this task. It is assumed that more intelligent path planning would be required for such a demonstration and that affordance detection would play a vital role.

References

[1] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. Analysis and Observations from the First Amazon Picking Challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2018. ISSN 15583783. doi: 10.1109/TASE.2016.2600527.

[2] E. Marchand, H. Uchiyama, and F. Spindler. Pose Estimation for Augmented Reality: A Hands-On Survey. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2633–2651, 2016. ISSN 10772626. doi: 10.1109/TVCG.2015.2513408.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? the KITTI Vision Benchmark Suite. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. ISSN 10636919. doi: 10.1109/CVPR.2012.6248074.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D Object Detection Network for Autonomous Driving. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 6526–6534, 2017. doi: 10.1109/CVPR.2017.691.

[5] M. Hassanin, S. Khan, and M. Tahtali. Visual Affordance and Function Understanding. *ACM Computing Surveys*, 54(3):1–35, 2021. ISSN 15577341. doi: 10.1145/3446370.

[6] T. Do, A. Nguyen, and I. Reid. AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5882–5889, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8460902.

[7] C. Wang, D. Xu, Y. Zhu, R. Martin-Martin, C. Lu, L. Fei-Fei, and S. Savarese. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. *Proceedings of*

the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3338–3347, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00346.

[8] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos. Affordance Detection of Tool Parts from Geometric Features. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1374–1381, 2015. ISSN 10504729. doi: 10.1109/ICRA.2015.7139369.

[9] J. Tremblay, T. To, and S. Birchfield. Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 2119–2122, 2018. ISSN 21607516. doi: 10.1109/CVPRW.2018.00275.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2017. ISSN 19393539. doi: 10.1109/TPAMI.2018.2844175.

[11] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *CoRR*, abs/1711.00199, 2017. URL http://arxiv.org/abs/1711.00199.

[12] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Object-Based Affordances Detection with Convolutional Neural Networks and Dense Conditional Random Fields. *IEEE International Conference on Intelligent Robots and Systems*, pages 5908–5915, 2017. ISSN 21530866. doi: 10.1109/IROS.2017.8206484.

[13] P. Corke. *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Springer, second edition, 2017.

[14] X. Lu. A Review of Solutions for Perspective-n-Point Problem in Camera Pose Estimation. *Journal of Physics: Conference Series*, 1087(5), 2018. ISSN 17426596. doi: 10.1088/1742-6596/1087/5/052009.

[15] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Texture-Less Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012. doi: 10.1109/TPAMI.2011.206.

[16] 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. *12th international conference on Computer Vision*, 3:548–562, 2012. URL http://link.springer.com/10.1007/978-3-642-33868-7.

[17] Z. Cao, Y. Sheikh, and N. K. Banerjee. Real-time scalable 6DOF pose estimation for textureless objects. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2441–2448, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487396.

[18] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. *CoRR*, abs/1809.10790, 2018. URL http://arxiv.org/abs/1809.10790.

[19] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. PVNET: Pixel-wise Voting Network for 6DoF Pose Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4556–4565, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00469.

[20] 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. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 11629–11638, 2020. ISSN 10636919. doi: 10.1109/CVPR42600.2020.01165.

[21] D. G. Low. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, pages 91–110, 2004. URL https://www.cs.ubc.ca/$\sim$lowe/papers/ijcv04.pdf.

[22] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision–ECCV 2006*, pages 404–417, 2006. URL http://link.springer.com/chapter/10.1007/11744023_32.

[23] X. S. Gao, X. R. Hou, J. Tang, and H. F. Cheng. Complete Solution Classification for the Perspective-Three-Point Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003. ISSN 01628828. doi: 10.1109/TPAMI.2003.1217599.

[24] M. A. Fischler and R. C Bolles. Random Sample Paradigm for Model Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Graphics and Image Processing*, 24(6):381–395, 1981. ISSN 00010782.

[25] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

[26] J. M. Wong, V. Kee, T. Le, S. Wagner, G. L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. S. Johnson, J. Wu, B. Zhou, and A. Torralba. SegICP: Integrated deep semantic segmentation and pose estimation. *IEEE International Conference on Intelligent Robots and Systems*, pages 5784–5789, 2017. ISSN 21530866. doi: 10.1109/IROS.2017.8206470.

[27] D. C. Hoang, T. Stoyanov, and A. J. Lilienthal. 6D Object Pose Estimation from Accurate 3D Instance-aware Semantic Reconstructions for Warehouse Robots. *2019 European Conference on Mobile Robots, ECMR 2019 - Proceedings*, 2019. doi: 10.1109/ECMR.2019.8870927.

[28] D. Xu, D. Anguelov, and A. Jain. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00033.

[29] J. Zeng, Y. Tong, Y. Huang, Q. Yan, W. Sun, J. Chen, and Y. Wang. Deep Surface Normal Estimation with Hierarchical RGB-D Fusion. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6146–6155, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00631.

[30] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H. M. Gross. Efficient RGB-D semantic segmentation for indoor scene analysis. *CoRR*, abs/2011.06961, 2020. URL https://arxiv.org/abs/2011.06961.

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 77–85, 2017. doi: 10.1109/CVPR.2017.16.

[32] F. J. Chu, R. Xu, L. Seguin, and P. A. Vela. Toward Affordance Detection and Ranking on Novel Objects for Real-World Robotic Manipulation. *IEEE Robotics and Automation Letters*, 4(4):4070–4077, 2019. ISSN 2377-3766. doi: 10.1109/lra.2019.2930364.

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2577031.

[34] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake. LabelFusion: A Pipeline for Generating Ground Truth Labels for Real RGBD Data of Cluttered Scenes. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3235–3242, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8460950.

[35] T. Grenzdörffer, M. Günther, and J. Hertzberg. YCB-M: A Multi-Camera RGB-D Dataset for Object Recognition and 6DoF Pose Estimation. *CoRR*, abs/2004.11657, 2020. URL https://arxiv.org/abs/2004.11657.

[36] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg. Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 7283–7290, 2019. ISSN 10504729. doi: 10.1109/ICRA.2019.8793744.

[37] Y. H. Tsai, W. C. Hung, S. Schulter, K. Sohn, M. H. Yang, and M. Chandraker. Learning to Adapt Structured Output Space for Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7472–7481, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00780.

[38] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang. Taking a Closer Look at Domain Shift: Category-level Adversaries for Semantics Consistent Domain Adaptation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2502–2511, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00261.

[39] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. YCB: Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set. 22(3):36–52, 2015. ISSN 10709932. doi: 10.1109/MRA.2015.2448951. URL http://arxiv.org/abs/1502.03143%0Ahttp://dx.doi.org/10.1109/MRA.2015.2448951.

[40] 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. ISSN 23773766. doi: 10.1109/LRA.2019.2894439.

[41] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. *Robotics: Science and Systems*, 11, 2015. ISSN 2330765X. doi: 10.15607/RSS.2015.XI.001.

[42] T. Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. H., P. P., D. R., P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[44] Y. Liang, W. Changjian, L. Fangzhao, P. Yuxing, L. Qin, Y. Yuan, and H. Zhen. FPN: Feature pyramid networks for object detection. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, pages 1702–1707, 2019. ISSN 10823409. doi: 10.1109/ICTAI.2019.00251.

[45] R. Margolin, L. Zelnik-Manor, and A. Tal. How to Evaluate Foreground Maps? *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Section 4):248–255, 2014. ISSN 10636919. doi: 10.1109/CVPR.2014.39.

[46] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Detecting Object Affordances with Convolutional Neural Networks. *IEEE International Conference on Intelligent Robots and Systems*, pages 2765–2770, 2016. ISSN 21530866. doi: 10.1109/IROS.2016.7759429.

[47] C. N. D. Minh, S. Z. Gilani, S. M. S. Islam, and D. Suter. Learning Affordance Segmentation: An Investigative Study. *2020 Digital Image Computing: Techniques and Applications, DICTA 2020*, 2020. doi: 10.1109/DICTA51227.2020.9363390.

[48] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11): 3212–3232, 2019. ISSN 21622388. doi: 10.1109/TNNLS.2018.2876865.

[49] T. Yi Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. RetinaNet: Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. ISSN 19393539. doi: 10.1109/TPAMI.2018.2858826.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-time Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.91.

[51] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On Pre-Trained Image Features and Synthetic Images for Deep Learning. *CoRR*, abs/1710.10710, 2017. URL http://arxiv.org/abs/1710.10710.

[52] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. *Image and Vision Computing*. ISSN 02628856. doi: 10.1016/j.imavis.2018.05.004.

[53] P. Beeson and B. Ames. TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics. *IEEE-RAS International Conference on Humanoid Robots*, pages 928–935, 2015. ISSN 21640580. doi: 10.1109/HUMANOIDS.2015.7363472.

[54] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.