DISSERTATION

6D Pose Estimation of Objects Using Limited Training Data

conducted in partial fulfillment of the requirements for the degree of a

Doktor der technischen Wissenschaften (Dr. techn.)

supervised by

Ao.Univ. Prof. Dipl.-Ing. Dr. techn. Markus Vincze
E376 Automation and Control Institute

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology

by

Kiru Park, MSc
DOB 04.02.1987
Matr. Nr.: 11704353

Vienna, November 2020 Kiru Park

Acknowledgment

When I started my PhD, I doubted whether doing a PhD is the right decision. I knew that the journey is not easy and a long trip. Fortunately, I have been surrounded by many outstanding colleagues that make the journey fun and successful. I have never regretted the decision during the entire journey because of their helpful and delightful support.

First of all, I would like to thank my supervisor, Prof. Markus Vincze, for granting me to start my PhD and for his unconditional support and trust in me. It is my pleasure to collaborate with Dr. Timothy Patten, who has given me detailed and careful feedback on what I have made out of my research. The priceless guidance and supervision they made improve my research performance beyond my original ability. I sincerely thank the external reviewer of my dissertation, Prof. Vincent Lepetit, for his valuable insights and comments.

I am grateful to all my current and former colleagues at the Vision for Robotics lab, who make me stimulating and fun; Dominik Bauer, Nikola Djukic, Matthias Hirschmanner, Michael Koller, Edith Langer, Bernhard Neuberger, Simon Schreiberhuber, Markus Suchi, Stefan Thalhammer, Jean-Baptiste Nicolas Weibel, Georg Jäggle, Sergey Alexandrov, Mohammad Reza Loghmani, Georg Halmetschlager. Thanks to all of them for the insightful scientific discussions that initiate and develop my research. The social events with them always made me refresh and smoothly get used to a new culture and atmosphere.

Nobody has been more important to me in the pursuit of my research than the members of my family. I would like to thank my parents and my parents-in-law, who give me credit with continuous support from the physically separated place. A huge thank you goes to my son, Chancel, who always makes me smile and refresh at home and grows up healthy and bright.

Finally, a warm thank you to my wife, Hyeyoung, who is my eternal companion for supporting me and making me intensely focus on research by caring for everything else in the new atmosphere. It must have been impossible to complete my doctoral research without her. I would love to dedicate this work to her.

The research leading to these results has partially funded by the Austrian Science Fund (FWF) under grant agreement No. I3969-N30 (InDex), the Austrian Research Promotion Agency (FFG) under grant agreement No. 879878 (K4R), the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement No. 600623 (STRANDS) and No. 610532 (SQUIRREL), and the OMRON Corporation.

Abstract

Pose estimation of objects is an important task to understand the surrounding environment for interacting with the objects in robot manipulation and augmented reality applications. Major computer vision tasks, such as object detection and classification, have significantly improved using Convolutions Neural Networks (CNN). Likewise, recent pose estimation methods using CNN have achieved high performance using a large amount of training data, which is, however, difficult to obtain from real environments. This thesis presents multiple methods that overcome the limited source of training in practical scenarios while solving common challenges in object pose estimation.

Symmetry and occlusion of objects are the most common challenges that make estimations inaccurate. This thesis introduces a method that regresses pixel-wise coordinates of an object while resolving ambiguous views from symmetric poses with a novel loss function in the training process. Coordinates of occluded regions are also predicted regardless of visibility, which makes the method robust to occlusion. The method shows state-of-the-art performance in the evaluations using only a limited number of real images. Nevertheless, annotating object poses in images is a difficult and time-consuming task, which prevents pose estimation methods from learning a new object from real scenes that are clutter. This thesis introduces an approach that leverages a few cluttered images of an object to learn its appearances in arbitrary poses. The novel refinement step updates pose annotations of input images to reduce pose errors that are common if poses are self-annotated by camera tracking or manually annotated by humans. Evaluations present the generated images from the method lead to state-of-the-art performance compared to methods using 13 times the number of real training images.

Domains such as retail shops face new objects very often. Thus, it is inefficient to train pose estimators for new objects every time. Furthermore, it is difficult to build precise 3D models of all instances in real-world environments. A template-based method in this thesis tackles these practical challenges by estimating poses of a new object using previous observations of the same or similar objects. The nearest observations are used to determine the object's locations, segmentation masks, and poses. The method is further extended to predict dense correspondences between the nearest observation and a target object for transferring grasp poses from similar experiences. Evaluations using public datasets show the template-based method performs better than baseline methods for segmentation and pose estimation tasks. Grasp experiments using a robot show the benefit of leveraging successful grasp experiences that significantly improve the grasp performance for familiar objects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, robots are working at home to clean floors and in public spaces to interact with people while displaying necessary information. Figure 1.1 shows typical functions of practical robots in real environments. The advances in sensor technologies have improved the accuracy and robustness of indoor localization tasks, which enables mobile robots to navigate to desired locations in complex outdoor environments for delivering packages. However, what people have expected from robots includes more advanced tasks as presented in Figure 1.2. For example, robots are expected to detect and manipulate objects [1], [2], learn a new object [1], [3] by themselves, and clean up a room by moving objects to places where the objects are supposed to belong [4]. Due to the uncertainty of real environments at home and offices, manipulating objects has been dominantly performed in industrial domains where environments can be strongly constrained without uncertainty, e.g., bin-picking of parts. Thus, it is important to detect and estimate objects' locations and orientations to manipulate them in uncertain environments such as tables and shelves where the objects are randomly placed. Pose estimation of objects is essential for understanding the surrounding environment and performing physical interactions with the objects.

Recent advances in artificial intelligence and computer vision researches based on massive amounts of data have greatly enhanced the robustness of object recognition pipelines using 2D images [5]–[8]. The recognition pipelines are applicable to different domains if the dataset used for training includes the target classes. For example, since people's appearances or their poses are usually similar across different domains, a detection pipeline trained on the MS-COCO dataset [9] is applicable to any domain that needs to detect people from images. However, it is challenging to train object recognizers if no dataset provides training samples of target objects. Since real environments consist of articulated objects, no prior knowledge, such as 3D models and training images, is available for robots to predict their poses. Thus, the lack of sufficient training data has prevented robots from applying advanced computer vision methods. This motivates the methods introduced in this thesis to face limited training resources for pose estimation pipelines in practical scenarios.

This chapter introduces why pose estimation of objects is essential, areas of applications using pose estimation, and challenges that have to be addressed for estimating precise poses in practical scenarios.

Figure 1.1: Examples of robots that provide services in real environments. The robots have robust navigation skills to provide services such as guiding a person, displaying information, cleaning a floor, delivering packages. Left: Werner from EU STRANDS[1] project, Middle: Roborock5[2] from Roborock Technology Co., Ltd, Right: Scout[3] from Amazon.

Figure 1.2: Manipulation of objects is one of the useful tasks that people expect for robots. Due to the uncertainty of real environments, the manipulation task has been applied to industrial environments with strong constraints. Left: Kenny from EU SQUIRREL[1] project picking up a toy from the ground. Right: Human Support Robot (HSR) [10] from Toyota picking up a bottle and handing over to a person.

1.1 Applications of Pose Estimation

A pose of an object is the most detailed information that describes the status of the object in 3D space. When a pose of the object is known in an image, the location (bounding box) of the object in the image is known, and the pixel-wise segmentation can be derived by comparing a rendered depth image with the input image. Therefore, it is possible to perform high-level tasks beyond simply detecting the existence of objects. There are two main applications that require precise poses of objects using images:

[1] http://strands.acin.tuwien.ac.at
[2] https://us.roborock.com/pages/roborock-s5
[3] https://blog.aboutamazon.com/transportation/meet-scout

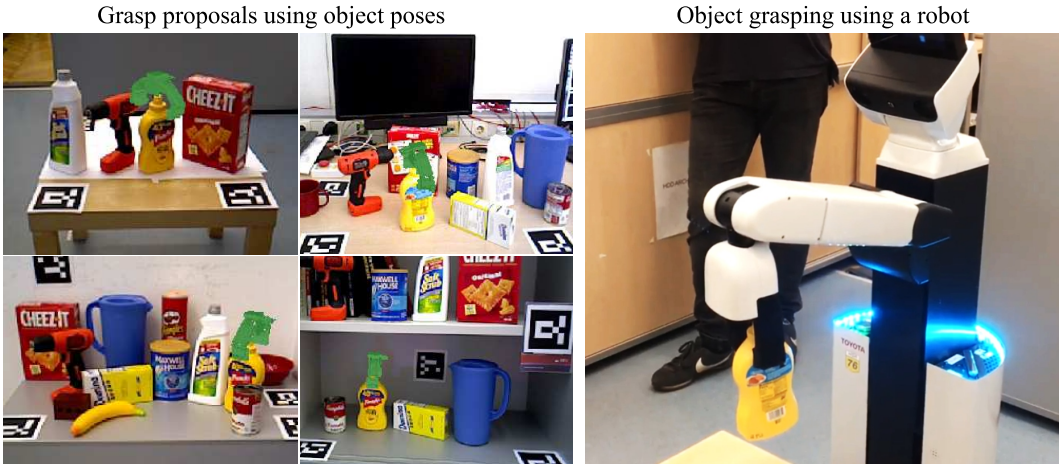

Figure 1.3: Examples of using poses of an object in Robot Manipulation. The grasp poses (green grippers in the left side) are proposed using estimated poses of the yellow bottle. The stable grasp poses are annotated by a physics simulation or demonstrations.

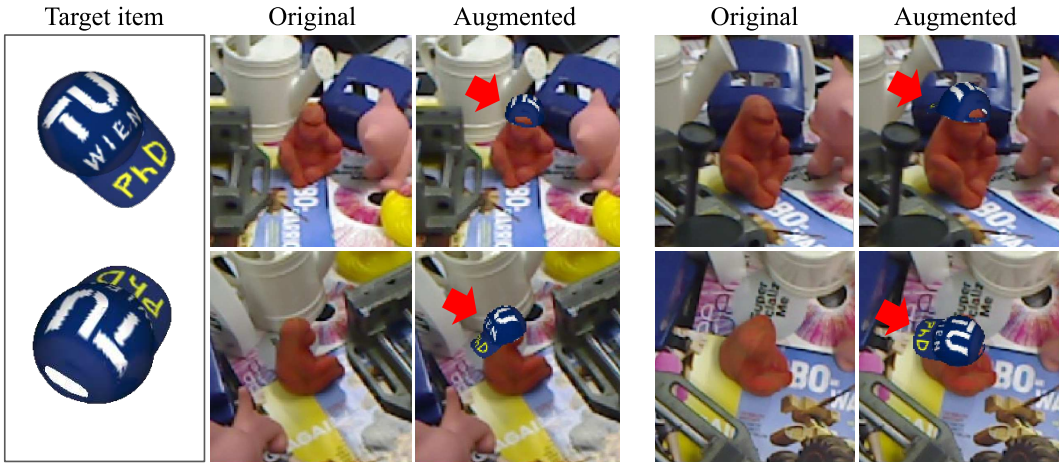

Figure 1.4: Examples of using poses of an object in Augmented Reality. The blue cap is rendered in a consistent orientation and location with respect to the ape by estimating poses of it. The method introduced in Chapter 3 is used to estimate poses.

robot manipulation and augmented reality applications.

In robot manipulation tasks, the object pose is used not only for picking the object robustly but also for placing it in the correct orientation (Figure 1.3). The knowledge of objects surrounding the target also helps the planning of safe trajectories without collision. When the poses of target objects are known, a robot computes grasp poses in the robot's reference frame and corresponding joint values. Thus, it is important to estimate accurate 3D locations of an object in the robot's coordinate frame. The employment of depth cameras has significantly improved the accuracy of pose estimation. Pixel-wise depth measurements from the cameras are easily converted to 3D point

clouds and aligned to the robot's coordinate frame with an actual environment scale.

For augmented reality applications, the pose of an object is used to display corresponding 3D information or images directly onto the object (Figure 1.4). It is important to compute poses in real-time for displaying information at correct locations while moving cameras or objects. In contrast to the robot manipulation tasks, a small translation error with respect to the principal camera axis, z-axis, is not crucial as the location of 3D information after projecting them onto the 2D image plane is not sensitive to the error. This property has enabled the applications to use common color cameras in smartphones and tablet PCs without depth measurements.

For both scenarios, it is common to have a new target object. When the new object is an articulated object, the object's geometry and texture often differ entirely from what has been observed from other objects. In other words, it is difficult to transfer prior knowledge of known objects to the new object. In this case, either training images or a 3D model of the object should be created. In general, 3D models are easily available in industries where objects are manufactured by 3D CAD models. This is one of the reasons why poses of objects have been widely used for bin-picking of industrial parts [11], [12]. On the other hand, it has not been easy to have a 3D model of an object in good quality at home and in offices using commercially available cameras in robots or tablets. It is also difficult to collect training images of an object with precise pose annotations. These difficulties have prevented recent pose estimation methods from applying to real applications. On the other hand, people recognize a new object by observing it from few viewpoints and predicting appearances of the object in unseen viewpoints using the knowledge of partially observed geometries and similar object classes. Thus, a key stepping stone towards practical applications is reducing efforts for training object recognizers (detectors and pose estimators) for new objects while keeping high-end performance, which is the purpose of this thesis.

1.2 Challenges

There are a number of challenges that need to be addressed to create robust pose estimation pipelines for practical scenarios. Figure 1.5 shows examples of challenges in pose estimation. Firstly, occlusion and symmetric objects have been mainly considered as common challenges. When a part of an object is occluded, appearances of local parts have to be leveraged to predict the poses of the object. When an object is symmetric, the same appearance is observed from different poses. In other words, multiple correct answers exist for the same input. If a recognition pipeline predicts a single pose for an input, the pipeline is easily confused by different answers for the similar inputs without prior knowledge of the symmetry. As many articulated objects and industrial objects are symmetric, both challenges should be carefully handled in pose estimation pipelines.

It is difficult to obtain training resources for pose estimation if 3D models do not exist for new objects. When color images are mainly used for recognition without depth images (e.g., augmented reality applications using mobile devices), it is even more difficult since 3D models should include high-quality textures for training. Furthermore,

[1] http://www.squirrel-project.eu/

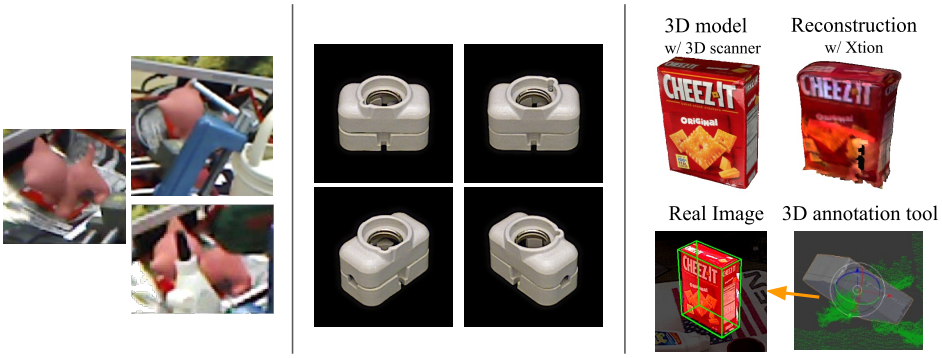

Figure 1.5: Challenges in object pose estimation.

annotating 6D poses of objects in images is a very difficult task for people who are unfamiliar with controlling 3D models in a virtual 3D world, and time-consuming compared to annotating 2D bounding boxes of objects. The types of available training resources vary with different scenarios. For example, when it is possible to reconstruct a 3D model of an object by moving a camera or the object, the geometry of the reconstructed 3D model can be directly used for training and matched to input depth images [12]. On the other hand, color images perform a crucial role in determining poses when depth images are not available as input (e.g., a monocular camera on a tablet), or texture information is necessary to distinguish different objects with similar geometries (e.g., cereal boxes). In this case, the reconstructed 3D models should include high-quality texture information to render synthetic images. Otherwise, real images with pose annotations are necessary for training. It is often difficult to obtain precise 3D models when constraints of an environment limit available viewpoints (e.g., a fixed camera on top) or geometries of objects are slightly different (e.g., vegetables and fruit). Therefore, types of training resources should be determined based on objects classes, input modalities during training and test, and environmental constraints where training data is collected.

Since this thesis's main focus is using limited training resources, practical scenarios are introduced to define the best way to obtain training resources while minimizing humans' efforts. In other words, instead of assuming all training resources are available, each chapter introduces a method and specifies what kinds of training resources and test modalities are available based on practical use cases.

1.3 Contributions and Outline

An overview of contributions made in this thesis is summarized in Figure 1.6. Methods proposed in Chapter 3 and 4 use RGB images for pose estimation, which is useful when objects have similar geometries but different textures or depth sensors are not available. In both chapters, 3D models are assumed to be given without texture information. The pose estimator, *Pix2Pose*, introduced in Chapter 3 performs pixel-wise regression of

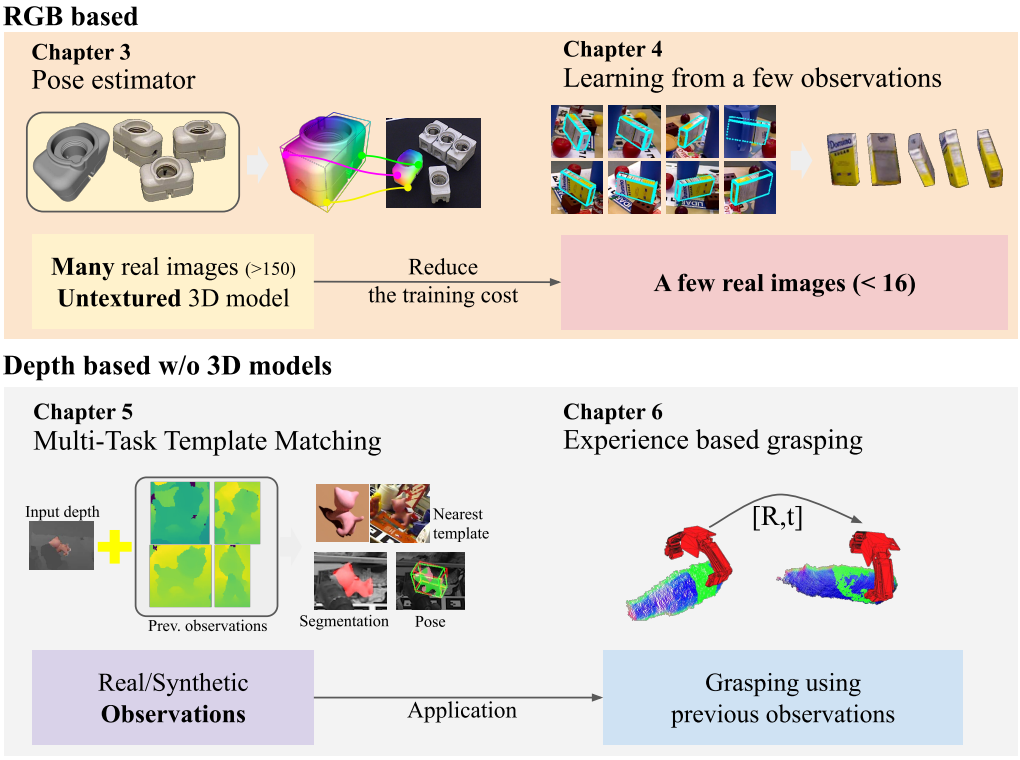

Figure 1.6: An overview of methods introduced in each chapter.

objects coordinates using RGB images. For training, non-textured 3D models and real images (200 to 3000) with pose labels are used. The object learning method, *Neural Object Learning (NOL)*, introduced in Chapter 4 is designed to minimize the number of real images required for training pose estimators. Instead of a large number of real images of an object without occlusion, the method synthesizes images of the object from arbitrary views by leveraging a few observations of the object. In Chapter 5 and 6, robot manipulation tasks are mainly considered without having 3D models of target objects. In both chapters, previous observations are directly used to transfer knowledge (e.g., successful grasp poses) of the previous one to the new input. *Multi-Task Template Matching (MTTM)* is designed to match the nearest templates of an object to detect and segment the object from the input image. Relative pose transformation from the nearest template to the detected target is also computed. The method, *DGCM-Net*, introduced in Chapter 6 employs basic ideas of MTTM for robot manipulation, which transfers successful grasp experience with similar geometries and classes to new inputs.

1.3.1 Pixel-Wise Regression of Object Coordinates

Although recent work using CNN for pose estimation has shown precise results with real-time performance using RGB images without depth information, estimations are often unstable and imprecise with occluded objects when a network directly regresses a representation of the pose (e.g., quaternions and axis-angle) or a few key points

of objects (e.g., 2D projected points of a 3D bounding box) that are used to match 2D-3D correspondences. Furthermore, these methods suffer from symmetric objects as a network is forced to predict completely different values for similar images. We introduce a novel method, Pix2Pose, that can supplement any 2D detection pipeline for additional pose estimation. Pix2Pose predicts pixel-wise 3D coordinates of an object using RGB images instead of a few key points to make estimations more robust. The network is trained to predict entire object points regardless of the visibility of points to enhance the performance for occluded objects. A specialized loss function, the *transformer loss*, is introduced to guide the network correctly for symmetric objects. As a result of the prediction, each pixel forms a 2D-3D correspondence that is used to compute poses by the Perspective-n-Point algorithm (PnP) [13]. The experiments with Pix2Pose using public datasets show outperforming results against previous work even if objects are occluded or symmetric.

Training resources Practically, the geometry of a reconstructed 3D model using a commercial RGB-D camera is reasonably good in different setups such as rotating or manipulating an object in front of a camera or moving a camera around an object. On the other hand, the quality of textures is often ignored in 3D reconstruction approaches. Furthermore, in industries, textures of CAD models are often undefined since real textures can be determined by materials used for manufacturing. Thus, the method is optimized and designed to use a 3D model of an object with real images that have pose annotations while ignoring texture information in 3D models. The symmetric poses of an object are assumed to be known before the training process. The experiments performed for a pose benchmark challenge show Pix2Pose is suitable for different domains that use no real images for training.
Detailed descriptions and discussions of the contributions above are addressed in Chapter 3 and have been published in the scientific paper [Park, ICCV 2019].

1.3.2 Neural Object Learning

Even though the method described above is successfully trained with a limited number of real images, it has been observed that the method suffers from unseen poses that are not covered by real training images. Thus, the real training images should cover the entire range of target poses, which is difficult to satisfy in real environments due to the difficulty of pose annotation and limited movements of a camera. This is why textured 3D models are used to render synthetic training images from uniformly sampled viewpoints. As explained above, however, the quality of textures of 3D models is not guaranteed to be sufficient to render realistic images. Furthermore, the texture quality becomes worse if images of a target object are captured from cluttered scenes, which is a common configuration in practical scenarios. To overcome these challenging issues, we propose a method, NOL, to synthesize images of an object in arbitrary poses using a few cluttered images with pose annotations and a non-textured 3D model of the object. For new objects, NOL requires 3D models and cluttered color images (less than 16 images in our evaluations) with pose annotations to map color information to vertices. A novel refinement step is introduced to adjust poses of objects in the source images, which overcomes pose annotation errors of source images. Evaluation results show that

images created by NOL are sufficient to train CNN-based pose estimation methods and achieves state-of-the-art performance.

Training resources The knowledge of 3D representation and a few observations of objects are usually sufficient for humans to recognize objects in a new environment. Likewise, NOL composes appearances of objects in arbitrary poses using 3D models and a few (a maximum of 16) cluttered and unconstrained images without using depth images, which is sufficient to lead a pose estimator to achieve state-of-the-art performance.
Detailed descriptions and discussions of the contributions above are addressed in Chapter 4 and have been published in the scientific paper [Park, ECCV 2020].

1.3.3 Multi-Task Template Matching

For picking and grasping an object from tables and boxes, the geometry of the object is more important than its texture. When geometries of objects are similar, a successful way of grasping them should be similar. Furthermore, CNN-based pose estimation methods that directly regress poses of objects have to be re-trained or fine-tuned every time when a new object is added. This motivates us to develop a novel framework, MTTM, for 6D pose estimation and segmentation of objects using a template set of depth images, which does not require further training of the CNN for a new object. The outputs of MTTM are the NN template with the closest pose, pixel-wise segmentation masks and the pose transformation from the pose of the NN template to that of the object in the test image. The experimental results show that MTTM successfully retrieves the nearest template for detecting a target object and uses it to predict a segmentation mask and pose of the object.

Training resources The network is trained with synthetically rendered depth images. Realistic and noisy depth images are rendered by simulating a stereo camera. For a target object, a set of observations of the object is used to build a set of templates. The observations can be either synthetic depth images if a 3D model is available, or real depth images with segmentation masks. The pose annotation is not necessary when a relative pose between a template and the input is sufficient for a target task (e.g., transfer a grasp pose from a template to the input). Otherwise, pose labels should be defined to compute a pose with respect to a predefined object coordinate frame.
Detailed descriptions and discussions of the contributions above are addressed in Chapter 5 and have been published in the scientific paper [Park, ICRA 2019].

1.3.4 Dense Correspondence Matching for Experience-based Grasping

The basic idea of MTTM is extended to grasp new objects using successful grasp experiences. As a grasp success is relevant for the association between a grasp pose and the local geometry of an object, a new pipeline is proposed to predict dense geometrical correspondences between two observations. The objects with similar geometries are identified through encoding global geometries, and successful grasp experiences of similar

objects are transferred through local correspondence matching. DGCM-Net encodes global geometrical feature vectors using depth images such that similar geometries are represented nearby in feature space to retrieve the nearest experience. DGCM-Net additionally predicts dense geometrical correspondences between pairs of depth images that are used to compute transformations between the local region around the grasp of a stored experience and the corresponding region on an object in a new scene. A novel representation, View-Dependent Normalized Object Coordinates (VD-NOC), is proposed to represent geometrical correspondences of similar objects regardless of their poses in a camera frame. Experimental results using a new dataset show that DGCM-Net precisely recovers grasps from experiences with the same object or objects with similar geometries from the same or different classes. Experiments using a real robot show that the inclusion of grasp proposals from DGCM-net using the closest experience significantly improves the performance of the baselines that do not employ information from experience. Additional experiments present the capability of collecting successful grasps for a novel object without supervision. The experiments show that it is often sufficient to have one or two successful grasp experiences to reliably grasp the same object. Experiments using a mug and a drill show DGCM-Net can be used to transfer meaningful grasp positions of an object for specific functionality such as handles of the mug and drill.

Training resources The network is trained with synthetically rendered depth images using representative 3D models of classes. The shape variations are simulated during the rendering process by randomly scaling models via each axis. All experiments are performed without additional training for a new object and a new class. Grasp attempts are stored together with input images with segmentation masks to build a database of successful experiences.
Detailed descriptions and discussions of the contributions above are addressed in Chapter 6 and have been published in the scientific paper [Patten & Park, Frontiers in Robotics and AI 2020].

1.4 List of Publications

Parts of the content presented in this dissertation have been previously published in the following papers:

- ***Kiru Park**, Timothy Patten, and Markus Vincze.* Neural Object Learning for 6D Pose Estimation Using a Few Cluttered Images. In Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2020.

- *Mohammad Reza Loghmani, Luca Robbiano, Mirco Planamente, **Kiru Park**, Barbara Caputo, and Markus Vincze.* Unsupervised Domain Adaptation through Inter-modal Rotation for RGB-D Object Recognition. IEEE Robotics and Automation Letters (RA-L), 2020.

- *Bernhard Neuberger, Timothy Patten, **Kiru Park**, and Markus Vincze.* Self-initialized Visual Servoing for Accurate End-effector Positioning. In 2020 6th

International Conference on Control, Automation and Robotics (ICCAR), pp. 676-682. IEEE, 2020.

- *Timothy Patten, **Kiru Park**, and Markus Vincze.* DGCM-Net: Dense Geometrical Correspondence Matching Network for Incremental Experience-based Robotic Grasping. Frontiers in Robotics and AI, 2020.
- ***Kiru Park**, Timothy Patten, and Markus Vincze.* Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 7668-7677. IEEE, 2019.
- ***Kiru Park**, Timothy Patten, Johann Prankl, and Markus Vincze.* Multi-task template matching for object detection, segmentation and pose estimation using depth images. In 2019 International Conference on Robotics and Automation (ICRA), pp. 7207-7213. IEEE, 2019.
- *Stefan Thalhammer, **Kiru Park**, Timothy Patten, Markus Vincze, and Walter Kropatsch.* Sydd: Synthetic depth data randomization for object detection using domain-relevant background. 24th Computer Vision Winter Workshop (CVWW), 2019.
- ***Kiru Park**, Johann Prankl, and Markus Vincze.* Mutual hypothesis verification for 6D pose estimation of natural objects. In Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 2192-2199. IEEE, 2017.

Chapter 2

Background

This chapter summarizes the basic knowledge that helps to understand this thesis. The definition of the 6D pose estimation task and different metrics that have been used to measure pose errors are introduced. Previous work related to object pose estimation and robot manipulation is reviewed.

2.1 Task Definition

The task of the methods introduced in this thesis is to estimate 6D poses of an object using RGB, RGB-D, and depth images. For a given coordinate frame of a known object and an input image, the pose of the object is a transformation that locates the object from the object's local frame to the camera's frame so that the appearance of the transformed object is similar to that of the input image (see Figure 2.1). For rigid body objects, a pose is defined by six parameters that represent special Euclidean group SE(3), which consists of three components for translations with respect to the x, y, and z axes and the other three components for rotations that satisfy the special orthogonal group constraint SO(3). 3D models of target objects are assumed to be given except for the methods in Chapter 5 and in Chapter 6 where partial observations of objects are directly used to match templates and estimate poses. Thus, relative transformations between templates (previous experience in Chapter 6) and target scenes are estimated.

2.2 Metrics

Different ways of measuring pose errors have been used for evaluation [14]. A standard metric, AD{D|I}, has been used to evaluate results for LineMOD [15] and LineMOD Occlusion [16]. This measures the average distance of vertices between a ground truth pose and an estimated pose as formulated by,

$$e_{ADD}(P_e,P_{\mathrm{GT}},\mathcal{M}) = \frac{1}{M}\sum_{\mathrm{x}\in\mathcal{M}} |P_e\mathrm{x} - P_{\mathrm{GT}}\mathrm{x}|, \tag{2.1}$$

where P_e and P_{GT} are estimated and ground-truth poses respectively. $\mathcal{M}$ is a set of vertices of a 3D model and M is a number of vertices. For symmetric objects, the

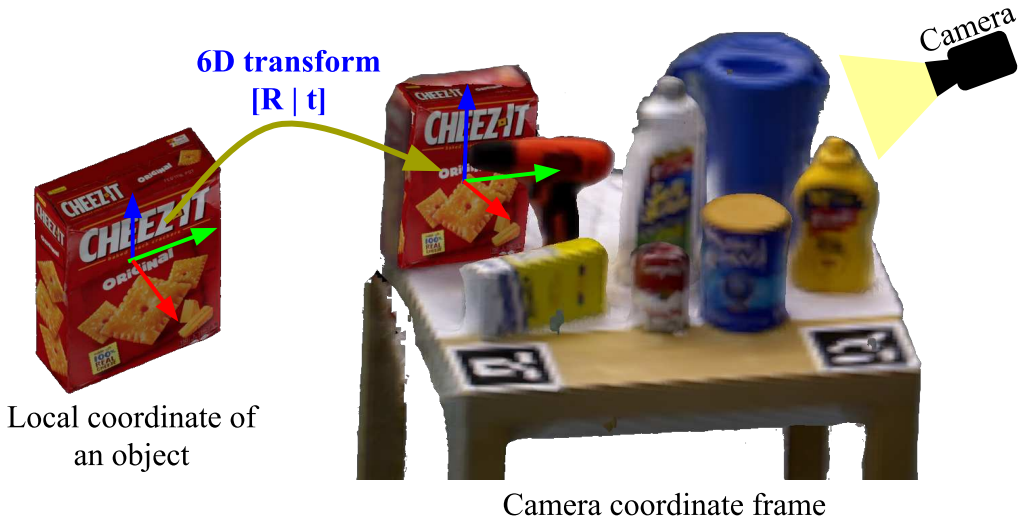

Figure 2.1: The definition of the 6D pose estimation task. The output of the pose estimation is transforming the local coordinate frame of an object to the camera coordinate frame. When the pose is accurate, the appearance of the object in the estimated pose should be similar to that of the input image.

average distance to the nearest vertices is used instead,

$$e_{ADI}(P_e,P_{\text{GT}},\mathcal{M}) = \frac{1}{M}\sum_{\text{x}_1\in\mathcal{M}}\min_{\text{x}_2\in\mathcal{M}}|P_e\text{x}_1 - P_{\text{GT}}\text{x}_2|. \tag{2.2}$$

The pose is considered correct when the error is less than 10% of the maximum 3D diameter of an object.

However, the AD{D|I} metric does not suitably handle view ambiguities that are caused by occlusion and symmetry. There should be multiple right answers for an object if a part of the object is occluded by other objects or itself, and the same appearance of the visible part can be observed from different poses. This motivates a new metric, Visible Surface Discrepancy (VSD). The VSD metric measures distance errors of visible parts only, which makes the metric invariant to ambiguities caused by symmetries and occlusion as formulated by,

$$e_{VSD}(P_e,P_{\text{GT}},\mathcal{M},\delta,\tau) = \frac{1}{V}\sum_{p\in V_e\cup V_{\text{GT}}} c(p, D_e, D_{\text{GT}},\tau), \tag{2.3}$$

where V_e and V_{GT} are 2D masks of the visible surface of models in the estimated and ground-truth poses. D_e and D_{GT} are rendered depth images of models in each pose. The function c counts how many pixels have larger distances than a threshold τ or are not visible in one of the rendered depth images. Thus, the lower number means the better pose estimation with more inlier pixels. In a pose benchmark of different pose estimation methods [17], the pose is regarded as correct when the error is less than 0.3 with τ=20mm and δ=15mm. This metric is mainly used for the T-Less dataset where all objects are symmetric and often occluded.

The recent challenge for pose benchmark [18] proposes a new metric, BOP score, that consists of three different pose errors (VSD, MSSD, and MSPD) [14] and their average recall values over different thresholds. The errors on each metric are computed while measuring recalls for each threshold value. For more details about pose metrics, it is recommended to refer to Hodaň *et al.* [14].

2.3 Related work

This section summarizes previous work related to the methods and approaches introduced in this thesis. Different approaches have been used for pose estimation are reviewed. The types of training data that have been used in previous work are discussed. Previous methods that learn and propose grasp poses are introduced to provide an overview and highlight the difference of a new method using successful grasp experience.

2.3.1 Pose Estimation Approaches

The traditional way of estimating poses of objects in an image is to match features extracted from the test image and training images that have pose annotations of the objects. As the performance of visual recognition tasks such as object detection and segmentation is significantly improved by data-driven approaches using CNNs, the hand-crafted features are replaced with feature maps encoded by CNNs. This section briefly introduces hand-crafted features and different approaches using CNNs that have been used for pose estimation.

Local Feature Matching Local features have been widely used to find point-to-point correspondences between multiple images such as Multi-View Stereo [19], [20], Structure From Motion (SfM) [21], and Visual SLAM [22]. For input images, keypoints are detected by a method to localize distinguishable points such as corners and edges. Feature descriptors are computed for each detected keypoint. Finally, point-to-point correspondences between images are derived by matching similar features from two images. The 2D locations of keypoints of matched points form 2D-2D correspondences that are used to compute transformations of cameras. Depending on the type of images, different types of descriptors have been used: RGB [23]–[25] and Pointclouds [26], [27]. In contrast to the two-view based methods that estimate movements of cameras, keypoints are extracted only from points of target objects for pose estimation. Generally, 3D locations of keypoints in the training images are known using pose annotations, which are used to form 2D-3D correspondences for RGB images and 3D-3D correspondences for point clouds. The local feature-based methods suffer from objects that produce an insufficient number of unique features such as texture-less objects for RGB inputs and symmetric objects for pointcloud inputs. As three or four correspondences produces a hypothesis, a large number of hypotheses is usually created by local feature matching, which requires additional techniques such as RANdom SAmple Consensus (RANSAC) [28] iterations for removing outlier correspondences and hypothesis verification steps for removing false positives [29].

Global Feature Matching Global features represent appearances of objects in different poses. Thus, local textures and shapes are not necessarily unique, which makes the methods using global features outperform local feature-based methods for texture-less objects [15], [30]. In contrast to the local features that have corresponding keypoint detection methods, global features should be computed on specific regions of interest (ROI) that should be given by uniformly sampling windows (sliding-window) or additional segmentation methods. Thus, the performance of a method relies on the quality of given ROIs. Like local features, different types of global features have been used depending on input modalities: RGB [15], [30], RGB-D [15], [31], and pointcloud [12], [32]–[34]. Point Pair Features [12] have been dominantly used for object pose estimation when a 3D model of an object is available, and pointclouds are used for recognition [12], [35]–[37] as the method produces accurate and reliable results for different types of datasets [17]. The features are extracted from local pairs of points while knowing the global association of those points, which combines the benefit of local and global feature matching. Even though the method relies only on geometries of objects, it outperforms other methods that use both RGB and depth images [35]. As the method relies on surface normals of points, it suffers from noisy depth inputs. The method needs additional verification steps, including multiple ICP refinements for hypotheses as a single matched point-pair can produce a hypothesis, which is computationally expensive.

Learned Feature Matching As CNNs trained on large scale datasets such as ImageNet [38] and MS-COCO [9] have shown great performance for object classification [39] and detection [5], features designed by researchers are replaced with trainable features. For example, Kehl et al. [40] train local descriptors using RGB-D images and convolutional auto-encoders. Each local correspondence of an object votes for the center of the object using the relative position to where the local patch is collected from. Oberweger *et al.* [41] predict multiple heat-maps to localize the 2D projections of 3D points of objects using local patches. These methods are robust to occlusion because they focus on local information only. However, additional computation is required to derive the best result among pose hypotheses, which makes these methods slow.

Global features are also encoded by CNNs after cropping a patch of an object and learn a manifold that can distinguish different objects and poses in feature spaces using RGB-D images [42]–[44]. A triplet loss guides the feature to have a closer distance for the same class and similar poses. For a given ROI from an image, the network encodes the feature vector of the ROI. The nearest feature in a codebook that is derived by rendering objects at uniform viewpoints is retrieved to propose a pose and a class label. As same as the global feature matching, these methods require additional detection pipelines to provide ROIs that contains target objects. The metric learning needs pairing of positive and negative samples, which potentially suffer from symmetric objects by assigning symmetric poses of an object as negative pairs. Balntas *et al.* [43] proposes a method to overcome this issue by measuring the similarity of objects poses to make positive pairs when renderings of different poses are similar.

More recently, Sundermeyer *et al.* [45] propose an auto-encoder network to train implicit representations of poses without supervision using RGB images only. Manual handling of symmetric objects is not necessary for this work since the implicit repre-

sentation can be close to any symmetric view. However, it is difficult to specify 3D translations using rendered templates that only give a good estimation of rotations. The size of the 2D bounding box is used to compute the z-component of 3D translation, which is too sensitive to small errors of 2D bounding boxes that are given from a 2D detection method.

Direct Pose Regression One of the benefits of CNN-based methods is the network can be trained end-to-end so that it is possible to train the network to produce values representing object poses using images directly without performing an intermediate process. Different types of representations have been used to predict poses of objects directly from an image such as the 2D locations of projected points of 3D bounding boxes [46], [47], classified view points [48], unit quaternions and translations [49], or the Lie algebra representation with the translation of z-axis [50]. Except for methods that predict projected points of the 3D bounding box, which requires further computations for the Perspective-n-Point (PnP) algorithm [13], the direct regression is computationally efficient since it does not require additional computation such as correspondence matching and RANSAC iterations. The drawback of these methods, however, is the lack of correspondences that can be useful to generate multiple pose hypotheses for the robust estimation of occluded objects. Furthermore, symmetric objects are usually handled by limiting the range of viewpoints, which sometimes requires additional treatments, e.g., training a CNN for classifying view ranges [46]. Xiang *et al.* [49] propose a loss function that computes the average distance to the nearest points of transformed models in an estimated pose and an annotated pose. However, searching for the nearest 3D points is time consuming and makes the training process inefficient.

Dense Correspondence Matching Instead of predicting a few correspondences by feature matching or a pose representation by the direct regression, Brachmann *et al.* [16], [51] have proposed methods that predict pixel-wise 3D coordinates of an object defined in the local frame of the object using the auto-context random forest. As the output of the model represents 3D coordinates of every pixel on the object, a large number of 2D-3D correspondences is created, which makes the pipeline robust to occlusion, and have a possibility to produce multiple hypotheses.

The encoder and decoder architecture have been used not only for computing latent features that represent necessary information of inputs to reconstruct output images but also for predicting pixel-wise predictions such as semantic segmentation, image de-noising [52], and image in-painting [53]. Recently, using Generative Adversarial Network (GAN) [54] improves the quality of generated images that are less blurry and more realistic, which have been used for the image-to-image translation [55], image in-painting and de-noising [56], [57] tasks. The high-quality results of these generative models show a clue that the pixel-wise predictions of 3D coordinates are significantly improved by employing CNN-based generative models, which initially motivate the method introduced in Chapter 3.

2.3.2 Training Samples for 6D Pose Estimation and Grasping

Previous work using CNNs requires a large number of training images of an object that covers a range of poses in test scenes sufficiently. Since it is difficult to annotate 6D poses of objects manually, synthetic training images are created using a textured 3D model of an object [48]. However, it is difficult to obtain a 3D model with high-quality texture from the real world without special devices such as the BigBIRD Object Scanning Rig [58] that provides precise pose information of cameras and objects to precisely align textures from different views. Using synthetic data introduces the domain gap between synthetic and real images, which should be specially treated with domain adaptation techniques [59]. It is possible to use only approximately 200 real images and apply various augmentation methods to successfully train pose estimation pipelines [46], [51]. However, the performance highly depends on the range of poses in the real training samples as the limited coverage of poses in training images causes inaccurate results for novel poses. This limitation has been tackled by using both real and synthetic images for training [59]–[62], which currently achieves state-of-the-art performance. The advantage of using both sources is that synthetic images supplement images for novel poses that are not observed in real images while real images regularize the network from over-fitting to synthetic images.

However, both textured 3D models and more than 200 real images with pose annotations are difficult to obtain from the real world. Furthermore, textured 3D models in public datasets are captured separately from constrained environments [15], [58], [63], [64] such as single objects with a simple background and precise camera pose localization tools. However, this well-constrained setup is difficult to replicate in real scenarios, e.g., a target object on a table is often occluded by other objects, camera poses are noisy without manual adjustments, and lighting conditions are not consistent. Thus, it is challenging to derive training images of new objects from cluttered scenes.

3D Object Modeling and Multi-View Texturing RGB-D images have been used to build 3D models by presenting an object in front of a fixed camera while rotating a turn table [65], manipulating the object using a robot end-effector [66] or human hands [67]. Alternatively, a mobile robot is used to actively move a camera to build a model of a fixed object in [3]. Even though these methods produce good 3D models in terms of geometry, textures are not optimized or even explicitly considered. Depth images are also required to align different views.

On the other hand, it is possible to map multiple images from different views to 3D mesh models using camera pose information [68], [69]. These approaches produce 3D models with high-quality textures since their optimization tries to assign continuous source images to neighboring pixels. However, these methods require depth images for correcting pose errors, which causes misalignment of color values and disconnected boundaries when different source images are not correctly aligned. Image-based rendering (IBR) has been used to complete a large scene by in-painting occluded area using multiple images from different view points [70]–[73]. These methods re-project source images to a target image using the relative poses of viewpoints. Then, projected images from different views are integrated with a weighted summation or an optimization based on different objective functions. However, IBR methods are designed to complete

large-scale scenes and suffer from noisy estimations of the camera and object poses, which causes blurry or misaligned images.

Differentiable Rendering The recent development of differentiable rendering enables the rendering process to be included during network training [74]–[76]. Therefore, the relationship between the 3D locations of each vertex and UV coordinates for textures are directly associated with pixel values of 2D rendered images, which have been used to create 3D meshes from a single 2D image [74], [75]. Furthermore, it is possible to render trainable features of projected vertices, which are optimized to minimize loss functions defined in a 2D image space.

Grasp Estimation The significant amount of attention given to robotic grasping has resulted in a large number and high diversity of techniques. A common strategy uses known object instances, which are provided as CAD models or are captured by a modeling process, e.g., [65], [67]. Given a known grasp configuration for an object in its local coordinate system, the task of grasping is simplified to estimating the pose of the object such that the grasp pose is transformed into the new scene. Traditional methods identify hand-crafted features to localize an object model within a scene [77]–[79], but more recent advances for pose estimation have been made using CNN-based pose estimation [49] and grasping pipelines achieve high success rate [80], [81]. The main limitation of this direction of research, however, is the closed-world assumption. The approach is restricted to only the objects for which a model is provided and thus cannot generalize to unknown objects.

To address the problem of grasping unknown objects, local geometry can serve as a strong cue. For example, fitting primitives and estimating grasps based on the geometrical structure of the primitives [82] or fitting superquadratics and synthesizing grasp poses at the points of minimum curvature [83] have been shown to work in certain cases. More often though, unknown object grasping is addressed by learning from data [84]. Along this line, methods predict the success of a proposed grasp by training a traditional classifier [85], [86] or deep neural network [87]–[92]. Alternatively, grasp simulation or analytical grasp metrics are computed for objects in model databases to generate training data [93]–[99]. The task is then to learn a model that predicts the value of the grasp metric given a proposal and then select the grasp that is most likely to succeed. There is also work that avoids the sampling and scoring procedure by directly predicting a grasp pose with a quality measure [100]. The generative method has proven to be computationally superior and sufficiently fast to be integrated in a closed-loop system. While the work for unknown object grasping has made considerable achievements, they are limited by the diversity of the training data. Out of distribution objects may not receive accurate grasp quality predictions and may fail. Thus, it is necessary to continuously learn and add new examples to the training set. Unfortunately, the deep neural networks that are applied do not have the capacity to be updated online.

Another approach to grasping is to leverage real robot experience and learn end-to-end strategies. One direction is to employ reinforcement learning [101]–[104]. The advantage of an end-to-end approach is that complete grasping policies are learned directly from visual input, which removes the need for a dedicated perception pipeline

with an additional motion planner for execution. A disadvantage, however, is that the policies are only be applied to scenarios that are perceptually similar, and thus generalization to novel environments is limited. Unsupervised methods, such as [105], better generalize to unseen scenarios and objects. They are more general to the task and less sensitive to the training scenes. These methods learn an embedding that can used to retrieve manipulation policies for online execution. Despite these advances, the major drawback of both self- and unsupervised learning is that many attempts are needed for training.

Experience-based grasping is much more efficient than reinforcement learning methods since far fewer examples are needed to learn grasps. The common approach is to accumulate samples of past success or failures to guide the grasp selection in new scenarios, under the assumption that objects with similar shape (or appearance) can be grasped in a similar way. Some work defines global shape descriptors and train a discriminative classifier to identify the similarity between object shapes to transfer grasps to familiar objects [106]–[108]. Other work leverage local feature descriptors to identify the relevant local regions associated with contact points to transfer grasps between objects within the same class [109]. Another approach is to analyze object regions and to maintain a library of prototypical grasps for recurring object parts. This is accomplished by measuring the similarity between regions on the surface of objects such as with height maps [110] or by surface distributions or densities [111]–[114]. A major assumption is that the observed parts are equivalent, which means grasp transfer is the application of a transformation from the prototype to the scene. They do not deal with the possibility of scale change or deformation. Such geometry variation would have to be stored as a new experience. Another drawback of prior work is that they use hand-crafted features to encode shape information.

Chapter 3

Pixel-Wise Coordinate Regression

3.1 Motivation

Although the inclusion of depth images has induced significant improvements by providing precise 3D pixel coordinates [17], [37], depth images are not always easily available. For example, mobile phones and tablets are typical devices for augmented reality applications and are offering no depth data. The quality of depth images differs from surface materials, lighting conditions, and environments, e.g., shiny and transparent objects that produce wrong depth measurements. As such, substantial research has been dedicated to estimating poses of known objects using RGB images only. The recognition using RGB images is also potentially useful for robots with multiple cameras, e.g., single RGB-D camera and multiple RGB cameras, as the same model can be used to recognize objects using RGB images from different cameras while producing more accurate results with RGB-D images.

Even though recent studies have shown great performance on pose estimation of objects in RGB images using Convolutional Neural Networks (CNN) [46], [47], [50], [51], the poses are not accurate as previous methods using depth images. Among the previous methods, it has been observed that estimated poses are more accurate when a network predicts 2D locations of projected 3D object coordinates [46], [47] instead of predicting 3D representation directly [49], [50]. Thus, the network does not need to learn the complex association between 2D and 3D correspondences that can be solved by PnP algorithms [13] without any training process. In this case, the network focuses more on detecting and locating keypoints of the objects in 2D space, which is also intuitive for humans as well (see Figure 3.1). However, since previous work predicts eight corners of 3D bounding boxes, small errors on a few points cause wrong pose estimations. Sundermeyer *et al.* [45] show that the encoder-decoder architecture produces ideal renderings of objects in good quality as outputs of the network for randomly augmented input images of the objects. This shows the possibility of predicting pixel-wise values relevant to object poses using the encoder-decoder architecture, e.g., 3D coordinates in the object frame [51]. This initiates the basic ideas of the method introduced in this chapter.

Pose estimation is more challenging when objects are occluded or symmetric. Training CNNs are often distracted by symmetric poses that have similar appearance inducing

very large errors in a naïve loss function. In previous work, a strategy to deal with symmetric objects is to limit the range of poses while rendering images for training [41], [48] or simply to apply a transformation from the pose outside of the limited range to a symmetric pose within the range [46] for real images with pose annotations. This approach is sufficient for objects that have infinite and continuous symmetric poses on a single axis, such as cylinders, by simply ignoring the rotation about the axis. However, as pointed in [46], when an object has a finite number of symmetric poses, it is difficult to determine poses around the boundaries of view limits. For example, if a box has an angle of symmetry, π, with respect to an axis and a view limit between 0 and π, the pose at $\pi + \alpha(\alpha \approx 0, \alpha > 0)$ has to be transformed to a symmetric pose at α even if the detailed appearance is closer to a pose at π. Thus, a loss function has to be investigated to avoid penalizing pose predictions of correct symmetric poses.

This chapter proposes solutions to the following research questions.

Research Questions

(1) What is the best way of estimating object poses using CNNs?
(2) How to guide the method to be more robust to occlusion?
(3) How to handle symmetric objects?

A novel method, *Pix2Pose*, is proposed to predict pixel-wise 3D coordinates of an object using RGB images with the encoder-decoder architecture. As a result of the prediction, each pixel forms a 2D-3D correspondence that is used to compute poses by the Perspective-n-Point algorithm (PnP) [13]. Since a number of correspondences are significantly larger than a few keypoints (e.g., eight corners for 3D bounding boxes), the method is more accurate while rejecting outlier predictions efficiently. To be robust to occlusion, the network predicts not only visible areas but also occluded parts of the object. For symmetric objects, a specialized loss function, the *transformer loss*, is proposed to robustly train the network for known symmetric poses of objects.

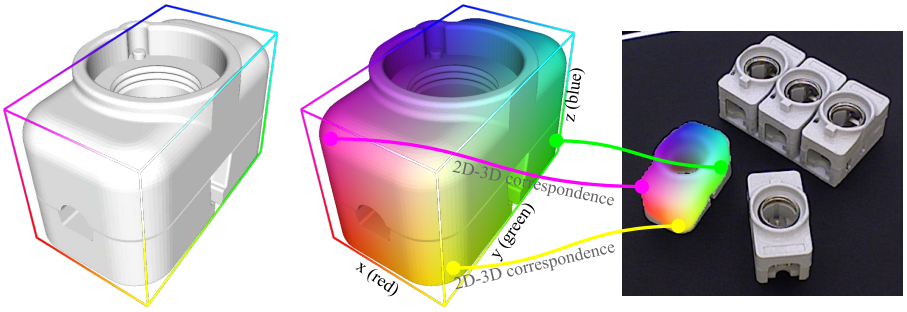

Figure 3.1: An example of converting a 3D model to a colored coordinate model. Normalized coordinates of each vertex are directly mapped to red, green and blue values in the color space. Pix2Pose predicts these colored images to build a 2D-3D correspondence per pixel directly without any feature matching operation.

3.2 Pix2Pose Network

This section provides a detailed description of the network architecture of Pix2Pose and loss functions for training. As shown in Figure 3.2, Pix2Pose predicts 3D coordinates of individual pixels using a cropped region containing an object. The robust estimation is established by recovering 3D coordinates of occluded parts and using all pixels of an object for pose prediction. A single network is trained and used for each object class. The texture of a 3D model is not necessary for training and inference.

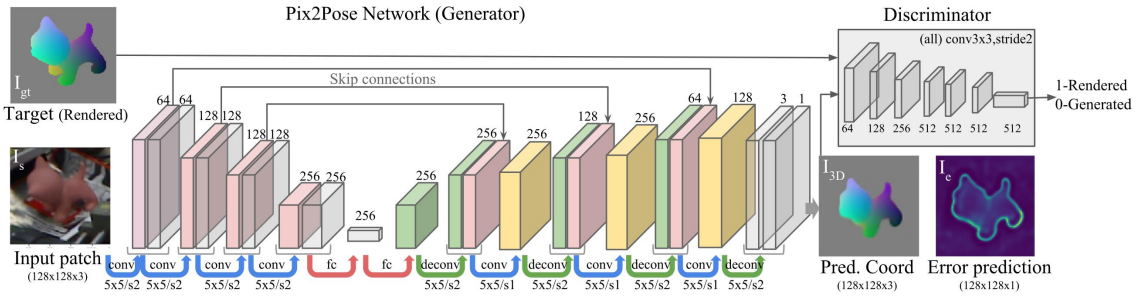

Figure 3.2: An overview of the architecture of Pix2Pose and the training pipeline.

3.2.1 Network Architecture

The architecture of the Pix2Pose network is described in Figure 3.2. The input of the network is a cropped image I_s using a bounding box of a detected object class. The outputs of the network are normalized 3D coordinates of each pixel I_{3D} in the object coordinate and estimated errors I_e of each prediction, I_{3D},$I_e = G(I_s)$, where G denotes the Pix2Pose network. The target output includes coordinate predictions of occluded parts, which makes the prediction more robust to partial occlusion. Since a coordinate consists of three values similar to RGB values in an image, the output I_{3D} can be regarded as a color image. Therefore, the ground truth output is easily derived by rendering the colored coordinate model in the ground truth pose. An example of 3D coordinate values in a color image is visualized in Figure 3.1. The error prediction I_e is regarded as a confidence score of each pixel, which is directly used to determine outlier and inlier pixels before the pose computation.

The cropped image patch is resized to 128×128px with three channels for RGB values. The sizes of filters and channels in the first four convolutional layers, the encoder, are the same as in [45]. To maintain details of low-level feature maps, skip connections [115] are added by copying the half channels of outputs from the first three layers to the corresponding symmetric layers in the decoder, which results in the more precise estimation of pixels around geometrical boundaries. The filter size of every convolution and deconvolution layer is fixed to 5×5 with stride 1 or 2 denoted as *s1* or *s2* in Figure 3.2. Two fully connected layers are applied for the bottleneck with 256 dimensions between the encoder and the decoder. The batch normalization [116] and the *LeakyReLU* activation are applied to every output of the intermediate layers except

the last layer. In the last layer, an output with three channels and the tanh activation produces a 3D coordinate image I_{3D}, and another output with one channel and the sigmoid activation estimates the expected errors I_e.

3.2.2 Network training

The main objective of training is to predict an output that minimizes errors between a target coordinate image and a predicted image while estimating expected errors of each pixel.

3.2.3 Transformer loss for 3D coordinate regression

To reconstruct the desired target image, the average L1 distance of each pixel is used. Since pixels belonging to an object are more important than the background, the errors under the object mask are multiplied by a factor of β (≥ 1) to weight errors in the object mask. The basic reconstruction loss $\mathcal{L}_r$ is defined as,

$$\mathcal{L}_r = \frac{1}{n}\Big[\beta \sum_{i \in M} ||I^i_{3D} - I^i_{gt}||_1 + \sum_{i \notin M} ||I^i_{3D} - I^i_{gt}||_1\Big], \tag{3.1}$$

where n is the number of pixels, I^i_{gt} is the i^{th} pixel of the target image, and M denotes an object mask of the target image, which includes pixels belonging to the object when it is fully visible. Therefore, this mask also contains the occluded parts to predict the values of invisible parts for robust estimation of occluded objects.

The loss above cannot handle symmetric objects since it penalizes pixels that have larger distances in the 3D space without any knowledge of the symmetry. While having the advantage of predicting pixel-wise coordinates, the 3D coordinate of each pixel is easily transformed to a symmetric pose by multiplying a 3D transformation matrix to the target image directly. Hence, the loss can be calculated for a pose that has the smallest error among symmetric pose candidates as formulated by,

$$\mathcal{L}_{3D} = \min_{p \in \text{sym}} \mathcal{L}_r(I_{3D}, R_p I_{gt}), \tag{3.2}$$

where $R_p \in \mathbb{R}^{3x3}$ is a transformation from a pose to a symmetric pose in a pool of symmetric poses, sym, including an identity matrix for the given pose. The pool sym is assumed to be defined before the training of an object. This novel loss, the transformer loss, is applicable to any symmetric object that has a finite number of symmetric poses. This loss adds only a tiny effort for computation since a small number of matrix multiplications is required. The transformer loss in Equation (3.2) is applied instead of the basic reconstruction loss in Equation (3.1).

To analyze the effect of transformer loss, the `obj-05` in the T-Less dataset [63] is used. To see the variation of loss values, 3D coordinate images are rendered while rotating the object around the z-axis. Loss values are computed using the coordinate image of a reference pose as a target output I_{gt} and images of other poses as predicted outputs I_{3D} in Equation (3.1) and Equation (3.2). As shown in Figure 3.3, the L1 loss in Equation (3.1) produces large errors for symmetric poses around π, which is the

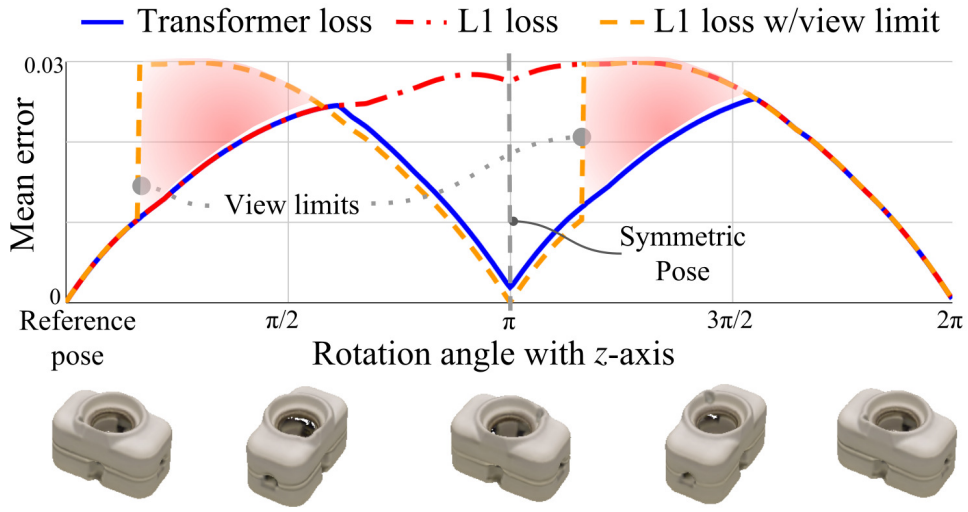

Figure 3.3: Variation of the reconstruction loss for a symmetric object with respect to z-axis rotation. Obj-05 in the T-Less [63] dataset is used.

Transformer loss	L1 w/view limits	L1
55.2	47.2	33.4

Table 3.1: Recall ($e_{\text{vsd}} < 0.3$) of obj-05 in T-Less using different reconstruction losses for training.

reason why the handling of symmetric objects is required. On the other hand, the value of the transformer loss produces minimum values on 0 and π, which is expected for `obj-05` with an angle of symmetry of π. The result denoted by *view limits* shows the value of the L1 loss while limiting the z-component of rotations between 0 and π. The pose that exceeds this limit is rotated to a symmetric pose. As shown in Figure 3.3, values are significantly changed at the angles of view limits and over-penalize poses under areas with red in Figure 3.3, which causes noisy predictions of poses around these angles. The results in Table 3.1 show the transformer loss significantly improves the performance compared to the L1 loss with the view limiting strategy and the L1 loss without handling symmetries. Qualitative results of the different strategies are discussed in Section 3.4.6.

3.2.4 Loss for error prediction

The error prediction I_e estimates the difference between the predicted image I_{3D} and the target image I_{gt}. This is identical to the reconstruction loss $\mathcal{L}_{\text{r}}$ with $\beta = 1$ such that pixels under the object mask are not penalized. Thus, the error prediction loss $\mathcal{L}_{\text{e}}$ is written as,

$$\mathcal{L}_{\text{e}} = \frac{1}{n} \sum_{i} ||I_{\text{e}}^{i} - \min\left[\mathcal{L}_{\text{r}}^{i}, 1\right]||_2^2, \beta = 1. \tag{3.3}$$

The error is bounded to the maximum value of the sigmoid function.

3.2.5 Training with Generative Adversarial Loss

As we reviewed in Section 2.3.1, using Generative Adversarial Network (GAN) [54] improves the quality of generated images with less blurry and more precise outputs. In the method, the discriminator network is added to determine whether the input is ground-truth or predicted values. In particular, with GAN, it is possible to penalize the network when a prediction has missing parts of the object due to occlusion while ground-truth images always provide entire pixels of objects regardless of visibility, which is similar to the image in-painting task. The adversarial loss term from GAN [54], $\mathcal{L}_{\text{GAN}}$, is employed to train the network. The loss is defined as,

$$\mathcal{L}_{\text{GAN}} = \log D(I_{gt}) + \log(1 - D(G(I_{\text{src}}))), \tag{3.4}$$

where D denotes the discriminator network. Finally, the objective of the training with GAN is formulated as,

$$G^* = \arg\min_G \max_D \mathcal{L}_{\text{GAN}}(G,D) + \lambda_1 \mathcal{L}_{\text{3D}}(G) + \lambda_2 \mathcal{L}_{\text{e}}(G), \tag{3.5}$$

where λ_1 and λ_2 denote weights to balance different tasks.

3.3 Two-Staged Pose prediction

This section gives a description of the process that computes a pose using the output of the Pix2Pose network. The overview of the process is shown in Figure 3.4. Before the estimation, the center, width, and height of each bounding box are used to crop the region of interest and resize it to the input size, $128 \times 128px$. The width and height of the region are set to the same size to keep the aspect ratio by taking the larger value. Then, they are multiplied by a factor of 1.5 so that the cropped region potentially includes occluded parts. The pose prediction is performed in two stages and the identical network is used in both stages. The first stage aligns the input bounding box to the center of the object, which could be shifted due to different 2D detection methods. It also removes unnecessary pixels (background and uncertain pixels) that are not preferred by the network. The second stage predicts a final estimation using the refined input from the first stage and computes the final pose.

3.3.1 Stage 1: Mask prediction and Bbox Adjustment

In this stage, the predicted coordinate image I_{3D} is used for specifying pixels that belong to the object including the occluded parts by taking pixels with non-zero values. The error prediction is used to remove the uncertain pixels if an error for a pixel is larger than the outlier threshold θ_o. The valid object mask is computed by taking the union of pixels that have non-zero values and pixels that have lower errors than θ_o. The new center of the bounding box is determined with the centroid of the valid mask. As a

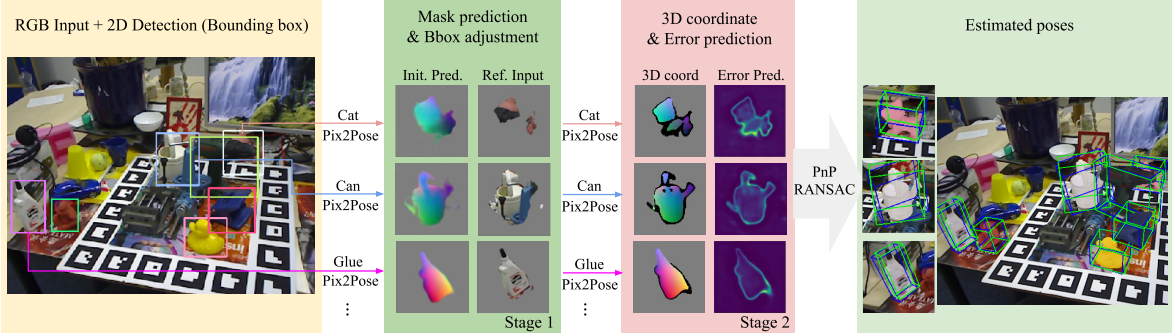

Figure 3.4: Examples of the pose estimation process. An image and 2D detection results are the input. In the first stage, the predicted results are used to specify important pixels and adjust bounding boxes while removing backgrounds and uncertain pixels. In the second stage, pixels with valid coordinate values and small error predictions are used to estimate poses using the PnP algorithm with RANSAC. Green and blue lines in the result represent 3D bounding boxes of objects in ground truth poses and estimated poses.

result, the output of the first stage is a refined input that only contains pixels in the valid mask cropped from a new bounding box. Examples of outputs of the first stage are shown in Figure 3.4. The refined input possibly contains the occluded parts when the error prediction is below the outlier threshold θ_o, which means the coordinates of these pixels are easy to predict despite occlusions.

3.3.2 Stage 2: Pixel-wise 3D coordinate regression with errors

The second estimation with the network is performed to predict a coordinate image and expected error values using the refined input as depicted in Figure 3.4. Black pixels in the 3D coordinate samples denote points that are removed when the error prediction is larger than the inlier threshold θ_i even though points have non-zero coordinate values. In other words, pixels that have non-zero coordinate values with smaller error predictions than θ_i are used to build 2D-3D correspondences. Since each pixel already has a value for a 3D point in the object coordinate, the 2D image coordinates and predicted 3D coordinates directly form correspondences. Then, applying the PnP algorithm [13] with RANdom SAmple Consensus (RANSAC) [28] iteration computes the final pose by maximizing the number of inliers that have lower re-projection errors than a threshold θ_{re}. It is worth mentioning that there is no rendering involved during the pose estimation since Pix2Pose does not assume textured 3D models. This also makes the estimation process fast.

3.4 Evaluation

In this section, experiments on three different datasets are performed to compare the performance of Pix2Pose to state-of-the-art methods. The evaluation using the LineMOD [15] dataset shows the performance for objects without occlusion in the single object scenario. For the multiple object scenario with occlusions, LineMOD Occlusion [16] and T-Less [63] are used. The evaluation on T-Less shows the most significant benefit of Pix2Pose since T-Less provides texture-less CAD models and most of the objects are symmetric, which is more challenging and common in industrial domains. Results of the Benchmark of Object Pose (BOP) challenge present how Pix2Pose generally performs with different datasets from various domains in comparison to state-of-the-art methods for the same amount of training sources.

3.4.1 Implementation details

For training, the batch size of each iteration is set to 50, the Adam optimizer [117] is used with initial learning rate of 0.0001 for 25K iterations. The learning rate is multiplied by a factor of 0.1 for every 12K iterations. Weights of loss functions in Equation (3.1) and Equation (3.5) are: β=3, λ_1=100 and λ_2=50. For evaluation, a 2D detection network and Pix2Pose networks of all object candidates in test sequences are loaded to the GPU memory, which requires approximately 2.2GB for the LineMOD Occlusion experiment with eight objects. The standard parameters for the inference are: θ_i=0.1, θ_o=[0.1, 0.2, 0.3], and θ_{re}=3. Since the values of error predictions are biased by the level of occlusion in the online augmentation and the shape and size of each object, the outlier threshold θ_o in the first stage is determined among three values to include more numbers of visible pixels while excluding noisy pixels using samples of training images with artificial occlusions. Figure 3.5 shows the examples of refined inputs using different outlier threshold values. All parameters used for evaluation are introduced in Appendix A. The training and evaluations are performed with an Nvidia GTX 1080 GPU and i7-6700K CPU.

Augmentation of training data A small number of real images are used for training with various augmentations. Image pixels of objects are extracted from real images and pasted to background images that are randomly picked from the MS-COCO dataset [9]. After applying the color augmentations on the image, the borderlines between the object and the background are blurred to make smooth boundaries. A part of the object area is replaced by the background image to simulate occlusion. Lastly, a random rotation is applied to both the augmented color image and the target coordinate image. The same augmentation is applied to all evaluations except sizes of occluded areas that need to be larger for datasets with occlusions, LineMOD Occlusion and T-Less. Sample augmented images are shown in Figure 3.6. Table 3.2 and 3.3 summarize the parameters used for the augmentation. As explained in Section 3.3, the network recognizes two types of inputs, with background pixels in the first stage and without background pixels in the second stage. Thus, a mini-batch is altered for every iteration as shown in Figure 3.6. Target coordinate images are rendered before training by placing the object in the

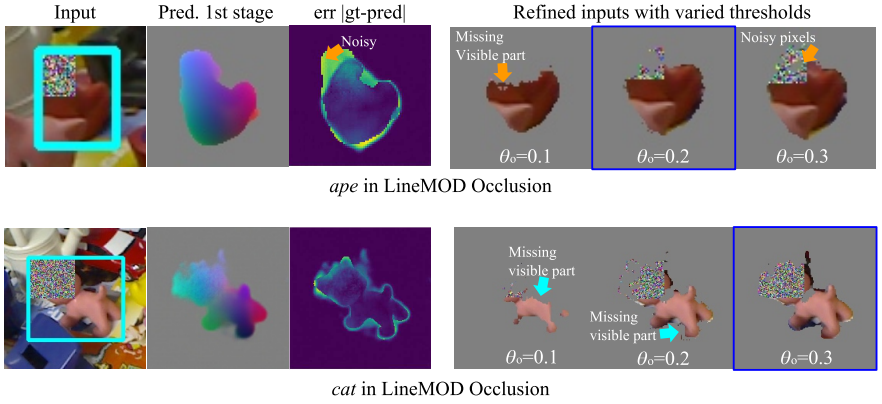

Figure 3.5: Examples of refined inputs in the first stage with varied values for the outlier threshold. Values are determined to maximize the number of visible pixels while excluding noisy predictions in refined inputs. The brighter pixel in images of the third column represents the larger error.

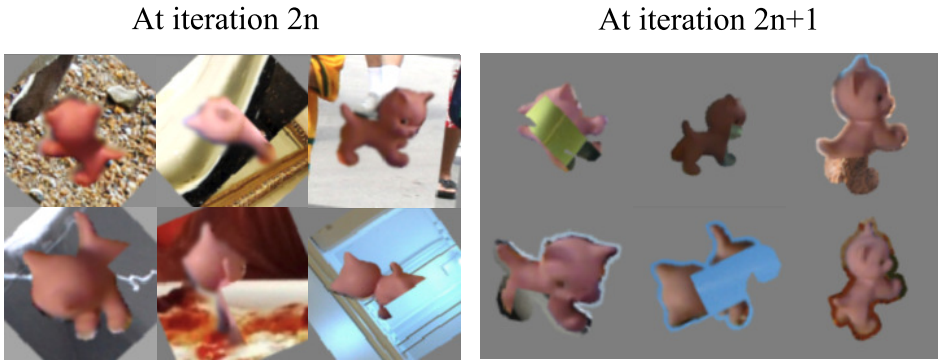

Figure 3.6: Examples of mini-batches. A mini-batch is altered for every training iteration. Left and right images are for the first and the second stage, respectively.

Add (each channel)	Contrast normalization	Multiply	Gaussian Blur
$\mathcal{U}$(-15, 15)	$\mathcal{U}$(0.8, 1.3)	$\mathcal{U}$(0.8, 1.2) per channel chance=0.3	$\mathcal{U}$(0.0, 0.5)

Table 3.2: Color augmentations applied to all evaluations.

Type	Random rotation	Fraction of occluded area	
Dataset	All	LineMOD	LineMOD Occlusion, T-Less
Range	$\mathcal{U}$(-45°, -45°)	$\mathcal{U}$(0, 0.1)	$\mathcal{U}$(0.04, 0.5)

Table 3.3: Image manipulation applied to each dataset.

ground truth poses using the colored coordinate model as in Figure 3.1.

2D detection network An improved Faster R-CNN [5], [6] with Resnet-101 [118] and Retinanet [7] with Resnet-50 are employed to provide classes of detected objects with 2D bounding boxes for all target objects of each evaluation. The networks are initialized with pre-trained weights using the MS-COCO dataset [9]. The same set of real training images is used to generate training images. Cropped patches of objects in real images are pasted to random background images to generate training images that contain multiple classes in each image.

3.4.2 LineMOD

For training, test sequences are separated into a training and test set. The divided set of each sequence is identical to the work of [47], [51], which uses 15% of test scenes, approximately less than 200 images per object, for training. A detection result, using Faster R-CNN, of an object with the highest score in each scene is used for pose estimation since the detection network produces multiple results for all 13 objects. For the symmetric objects, marked with (*) in Table 3.4, the pool of symmetric poses sym is defined as, $sym = [I, R_z^{\pi}]$, where R_z^{π} represents a transformation matrix of rotation with π about the z-axis.

Table 3.4 shows Pix2Pose significantly outperforms state-of-the-art methods that use the same amount of real training images without textured 3D models. Even though methods on the last two columns of Table 3.4 use textured 3D models for pose refinement, our method shows competitive results against these methods. The results on symmetric objects show the best performance among methods that do not perform pose refinement. This verifies the benefit of the transformer loss, which improves the robustness of initial pose predictions for symmetric objects. Figure 3.7 visualizes result examples.

Train src	Real images only (approx. 150) w/o ref.				synthetic	w/ refinement	
Objects	Pix2Pose	YOLO-6D [47]	Brachmann [51]	BB8 [46]	AAE [45]	BB8 [46]	SSD-6D [48]
Ape	58.1	21.6	33.2	27.9	4.0	40.4	65
Bench vise	91.0	81.8	64.8	62.0	20.9	91.8	80
Camera	60.9	36.6	38.4	40.1	30.5	55.7	78
Can	84.4	68.8	62.9	48.1	35.9	64.1	86
Cat	65.0	41.8	42.7	45.2	17.9	62.6	70
Driller	76.3	63.5	61.9	58.6	24.0	74.4	73
Duck	43.8	27.2	30.2	32.8	4.9	44.3	66
Eggbox*	96.8	69.6	49.9	40.0	81.0	57.8	100
Glue*	79.4	80.0	31.2	27.0	45.5	41.2	100
Hole puncher	74.8	42.6	52.8	42.4	17.6	67.2	49
Iron	83.4	75.0	80.0	67.0	32.0	84.7	78
Lamp	82.0	71.1	67.0	39.9	60.5	76.5	73
Phone	45.0	47.7	38.1	35.2	33.8	54.0	79
Average	**72.4**	56.0	50.2	43.6	31.4	62.7	76.7

Table 3.4: LineMOD: Percentages of correctly estimated poses (AD{D|I}-10%).

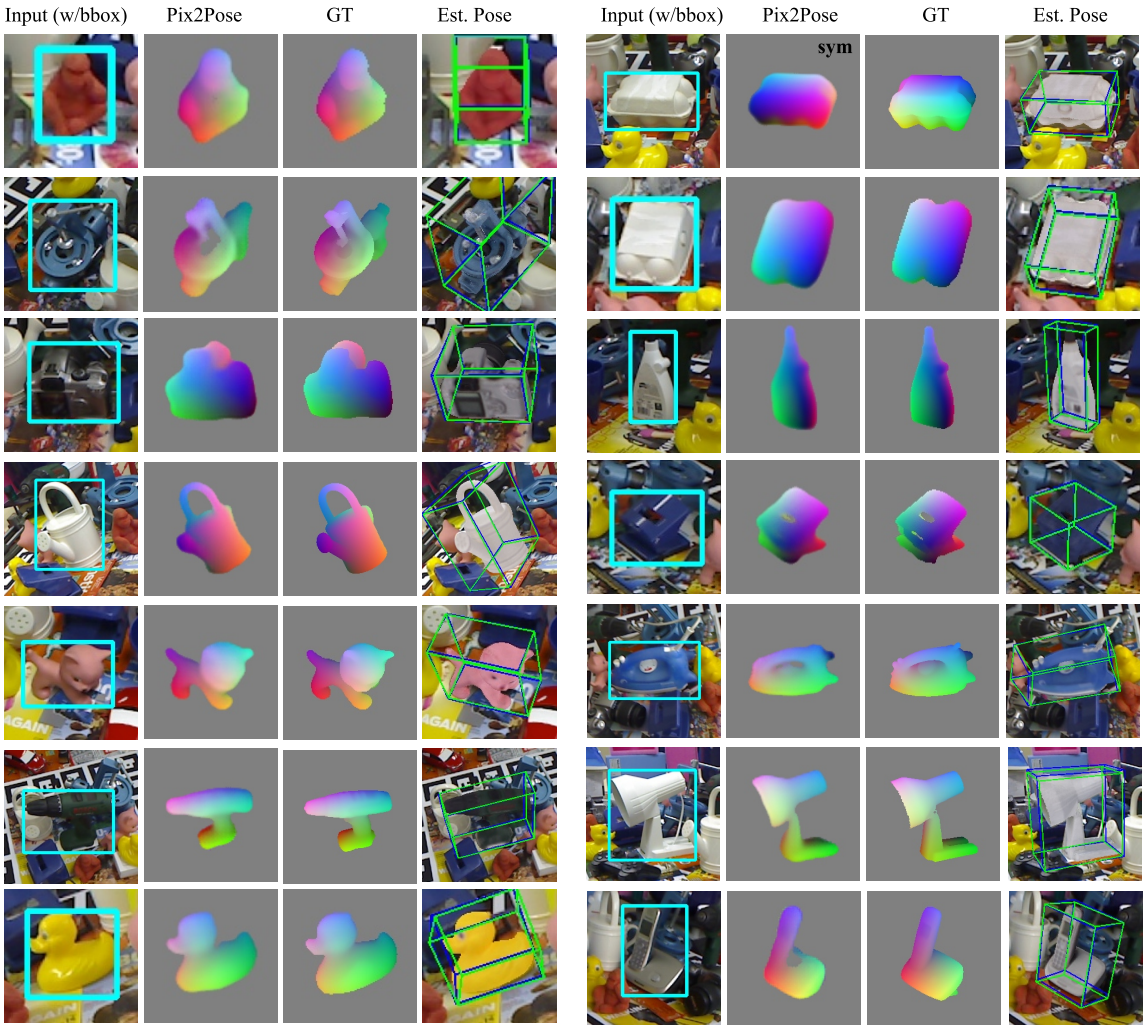

Figure 3.7: Example results on LineMOD. The result marked with *sym* represents that the prediction is the symmetric pose of the ground truth pose. 3D boxes in green and blue are ground-truth and estimated poses respectively.

3.4.3 LineMOD Occlusion

LineMOD Occlusion (LineMOD-Occ) is created by annotating eight objects in a test sequence of LineMOD. Thus, the test sequences of eight objects in LineMOD are used for training without overlapping with test images. Faster R-CNN is used as a 2D detection pipeline.

As shown in Table 3.5, Pix2Pose significantly outperforms the method of [47] using only real images for training. Furthermore, Pix2Pose outperforms the state of the art on three out of eight objects. On average it performs best even though methods of [41] and [49] use more images that are synthetically rendered by using textured 3D models of objects. Although these methods cover more various poses than the given small number of images, Pix2Pose robustly estimates poses with less coverage of training poses.

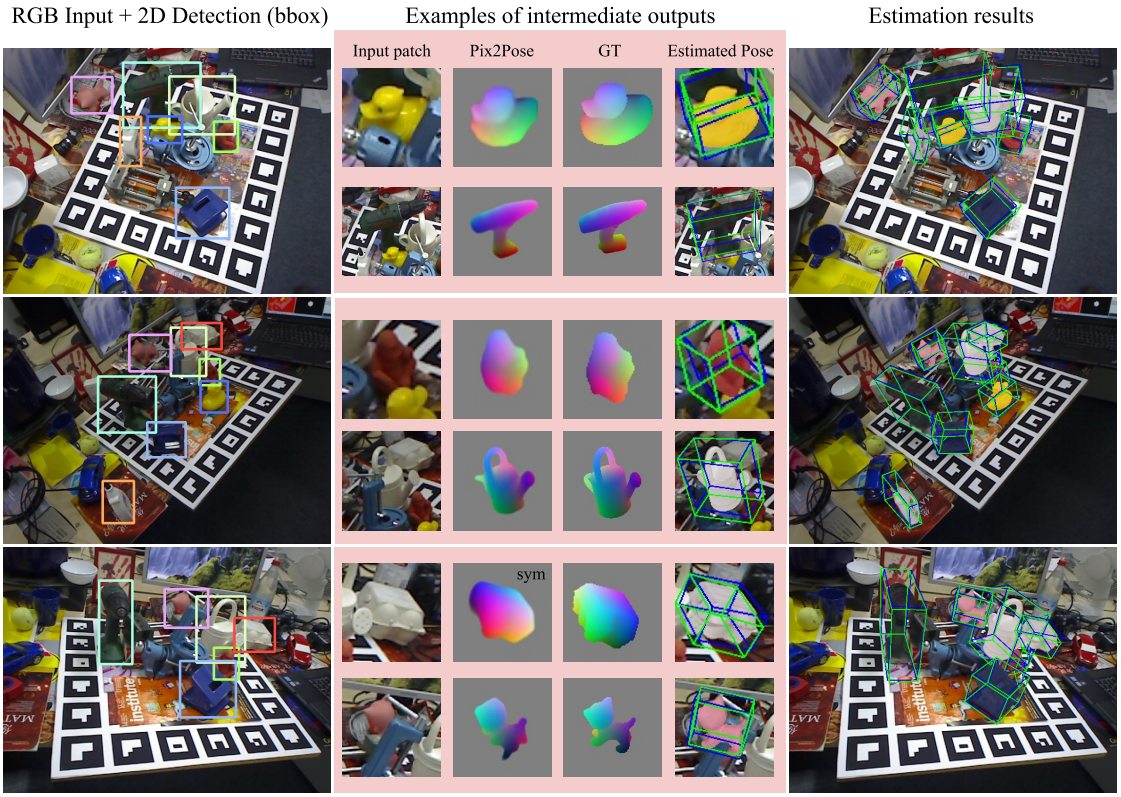

Figure 3.8: Example results on LineMOD Occlusion. The precise prediction of occluded parts enhances robustness.

Method	**Pix2Pose**	Oberweger† [41]	PoseCNN† [49]	Tekin [47]
Ape	**22.0**	17.6	9.6	2.48
Can	44.7	**53.9**	45.2	17.48
Cat	**22.7**	3.31	0.93	0.67
Driller	44.7	**62.4**	41.4	7.66
Duck	15.0	19.2	**19.6**	1.14
Eggbox*	25.2	**25.9**	22.0	-
Glue*	32.4	**39.6**	38.5	10.08
Hole puncher	**49.5**	21.3	22.1	5.45
Average	**32.0**	30.4	24.9	6.42

Table 3.5: LineMOD Occlusion: object recall (AD{D|I}-10%). (†) indicates the method uses synthetically rendered images and real images for training, which has better coverage of viewpoints.

3.4.4 T-Less

In this dataset, a CAD model without textures and a reconstructed 3D model with textures are given for each object. Even though previous work uses reconstructed models for training, to show the advantage of our method, CAD models are used for training (as shown in Figure 3.1) with real training images provided by the dataset. To

minimize the gap of object masks between a real image and a rendered scene using a CAD model, the object mask of the real image is used to remove pixels outside of the mask in the rendered coordinate images. The pool of symmetric poses *sym* of objects is defined manually similar to the `eggbox` in the LineMOD evaluation for box-like objects such as `obj-05`. For cylindrical objects such as `obj-01`, the rotation component of the z-axis is simply ignored and regarded as a non-symmetric object. The experiment is performed based on the protocol of [17]. Instead of a subset of the test sequences in [17], full test images are used to compare with the state of the art [45]. Retinanet is used as a 2D detection method and objects visible more than 10% are considered as estimation targets [17], [45].

The result in Table 3.6 shows Pix2Pose outperforms the-state-of-the-art method that uses RGB images only by a significant margin. The performance is also better than the best learning-based methods [40], [51] in the benchmark [17]. Although these methods use color and depth images to refine poses or to derive the best pose among multiple hypotheses, our method, that predicts a single pose per detected object, performs better than these methods without refinement using depth images.

Input	RGB Only		RGB-D	
Method	Pix2Pose	AAE [45]	Kehl [40]	Brachmann [51]
obj-1	38.4	8.9	7	8
obj-2	35.3	13.2	10	10
obj-3	40.9	12.5	18	21
obj-4	26.3	6.6	24	4
obj-5	55.2	34.8	23	46
obj-6	31.5	20.2	10	19
obj-7	1.1	16.2	0	52
obj-8	13.1	19.7	2	22
obj-9	33.9	36.2	11	12
obj-10	45.8	11.6	17	7
obj-11	30.7	6.3	5	3
obj-12	30.4	8.2	1	3
obj-13	31.0	4.9	0	0
obj-14	19.5	4.6	9	0
obj-15	56.1	26.7	12	0
obj-16	66.5	21.7	56	5
obj-17	37.9	64.8	52	3
obj-18	45.3	14.3	22	54
obj-19	21.7	22.5	35	38
obj-20	1.9	5.3	5	1
obj-21	19.4	17.9	26	39
obj-22	9.5	18.6	27	19
obj-23	30.7	18.6	71	61
obj-24	18.3	4.2	36	1
obj-25	9.5	18.8	28	16
obj-26	13.9	12.6	51	27
obj-27	24.4	21.1	34	17
obj-28	43.0	23.1	54	13
obj-29	25.8	26.7	86	6
obj-30	28.8	29.6	69	5
Average	**29.5**	18.4	24.6	17.84

Table 3.6: Object recall ($e_{\text{VSD}} < 0.3$, $\tau = 20$mm) on all test scenes using PrimeSense in T-Less. Results of [40] and [51] are cited from [17].

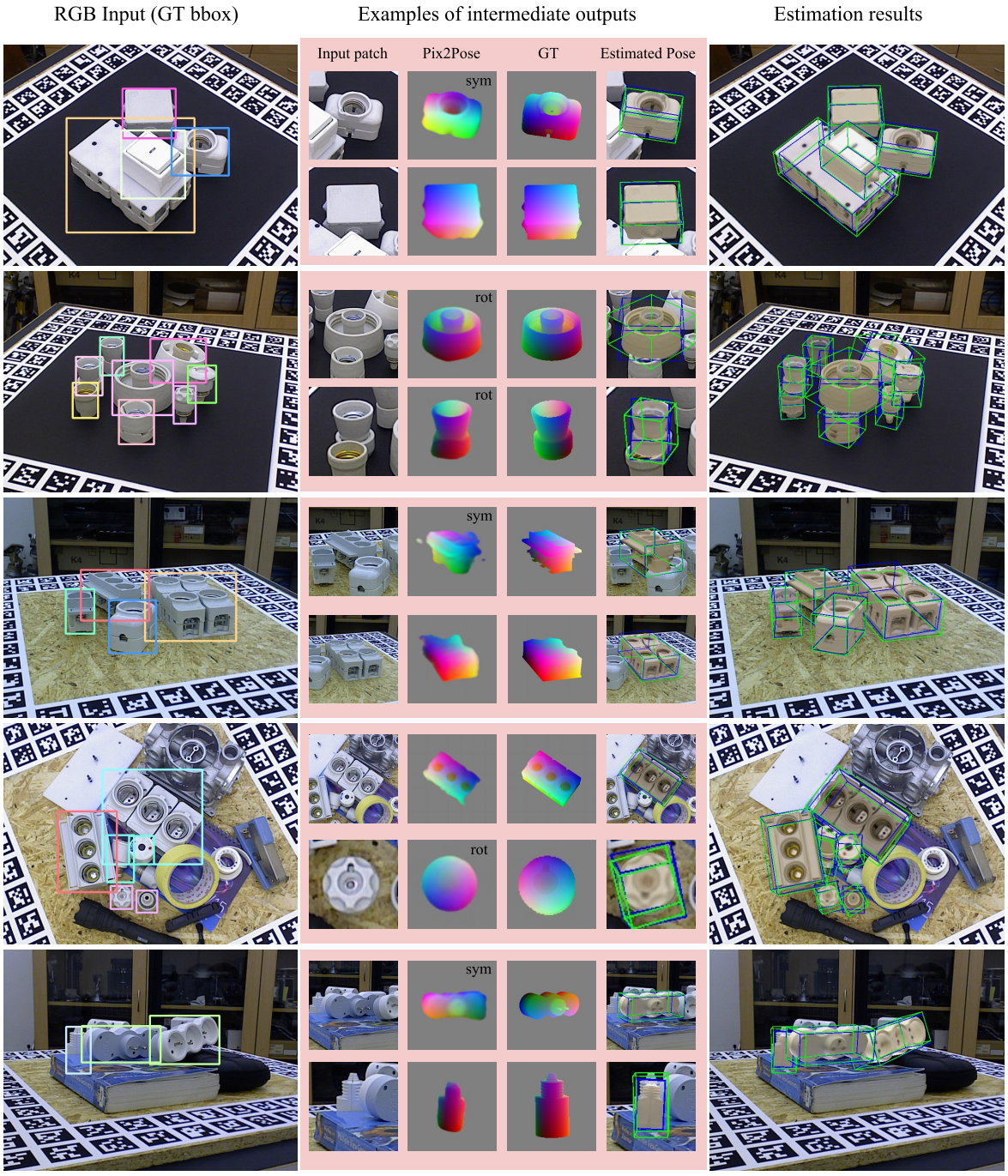

Figure 3.9: Example results on T-Less. For visualization, ground-truth bounding boxes are used to show pose estimation results regardless of the 2D detection performance. Results with *rot* denote estimations of objects with cylindrical shapes.

3.4.5 BOP Challenge

Pix2Pose is further evaluated with different pose estimation methods by participating the BOP Challenge [17] for two years in a row (2019 and 2020). In the challenge, participants are asked to submit pose estimation results for eleven datasets including seven core datasets that are mainly used for measuring overall performance. As all hyper-parameters should be the same for all objects and datasets, three outlier thresholds, $\theta_o = [0.15, 0.25, 0.35]$, are used instead of using a fixed value for each object in the previous evaluations. For each 2D detection, an outlier threshold with a larger number of inlier pixels is selected after PnP-RANSAC iterations. Mask R-CNN is used to supply 2D detection results and mask predictions that are used for specifying target points during the ICP refinement. Table 3.7 report the summary of results in the first year (2019). Thanks to the handling of symmetries with the transformer loss, Pix2Pose produces reliable results for T-Less that consists of symmetric and texture-less industrial objects compared against other methods [59], [61] that directly predict pixel-wise correspondences. After the ICP refinement using depth images, Pix2Pose performs the best on YCB-Video [49] and RU-APC [119] in comparison to all participated methods. However, results are worse than other methods when synthetic images are used for training (e.g., LM-O and HB) because the small range of color augmentations optimized for real images are used for synthetic images. Furthermore, synthetic training images are rendered with a simple OpenGL based pipeline without varied lighting conditions and cluttered backgrounds, which generates appearances of images apart from those of real images. Due to the lack of high-quality synthetic images, the overall performance of RGB-based methods is significantly lower than PPF-based methods. Nevertheless, Pix2Pose outperforms PPF methods for Rutgers-Amazon Picking Challenge dataset [119] where the quality of depth images are significantly worse.

From the results of the first year, participants and organizers have learned that data-driven methods using RGB images rely on the quality of synthetic training images when no real training image is available for dataset. Photo-realistic images are created and provided after rendering photo-realistic images of cluttered scenes in the second year (2020) to encourage the participation of the data-driven methods. The encoder part of the Pix2Pose network is replaced with the first three stages of Resnet-50 with pre-trained weights using Imagenet [38], which provides better initialization of parameters [59]. Table 3.8 summarizes the results of the second year. Pix2Pose outperforms the winner of the first year and is placed at fourth in terms of the overall performance on core datasets. The best method [122] requires textured 3D models for training the pose refiner network. The second best method uses Mask R-CNN [6] and RetinaMask [125] for 2D detection and mask prediction, and PPF for pose estimation. Therefore, Pix2Pose leads CNN-based methods that use the same amount of information for objects [45], [59], [61] after the ICP refinement. Implementation details including parameters are shared in the public repository [1].

[1] `https://github.com/kirumang/Pix2Pose`

	Real training image		-	✓	✓	-	-	-	✓	-
Method	Test img	AVG	LMO	TLESS	TUDL	ICBIN	ITODD	HB	YCBV	RUAPC
PPF(w/edge) [12]	RGB-D	0.550	0.515	0.500	0.851	0.368	0.570	0.671	0.375	0.170
Félix&Neves [120], [121]	RGB-D	0.412	0.394	0.212	0.851	0.323	0.069	0.529	0.510	
AAE(w/ICP) [45]	RGB-D	0.398	0.237	0.487	0.614	0.281	0.158	0.506	0.505	-
CDPN [61]	RGB	0.353	0.374	0.124	0.757	0.257	0.070	0.470	0.422	-
AAE [45]	RGB	0.270	0.146	0.304	0.401	0.217	0.101	0.346	0.377	-
Pix2Pose	RGB	0.205	0.077	0.275	0.349	0.215	0.032	0.200	0.290	0.253
Pix2Pose(w/ICP)	RGB-D	-	-	-	-	-	-	-	**0.675**	**0.410**
DPOD [59]	RGB	0.161	0.169	0.081	0.242	0.130	0.0	0.286	0.222	-

Table 3.7: Results of BOP Challenge 2019.

Method	Test img	AVG	LMO	TLESS	TUDL	ICBIN	ITODD	HB	YCBV
CosyPose(w/ICP) [122]	RGB-D	0.698	0.714	0.701	0.939	0.647	0.313	0.712	0.861
Hybrid-PPF [123]	RGB-D	0.639	0.631	0.655	0.920	0.430	0.483	0.651	0.701
CosyPose-RGB [122] ref.	RGB	0.637	0.633	0.728	0.823	0.583	0.216	0.656	0.821
Pix2Pose(w/ICP)	RGB-D	0.591	0.588	0.512	0.820	0.390	0.351	0.695	0.780
Vidal *et al.* [37]	Depth	0.569	0.582	0.538	0.876	0.393	0.435	0.706	0.450
CDPNv2(w/ICP) [61]	RGB-D	0.568	0.630	0.464	0.913	0.450	0.186	0.712	0.619
PPF(Edges) [12]	RGB-D	0.550	0.515	0.500	0.851	0.368	0.570	0.671	0.375
Leaping from 2D to 6D	RGB	0.471	0.525	0.403	0.751	0.342	0.077	0.658	0.543
EPOS [124]	RGB	0.457	0.547	0.467	0.558	0.363	0.186	0.580	0.499
Félix&Neves [120], [121]	RGB-D	0.412	0.394	0.212	0.851	0.323	0.069	0.529	0.510
AAE(w/ICP) [45]	RGB-D	0.398	0.237	0.487	0.614	0.281	0.185	0.506	0.505

Table 3.8: Results of BOP Challenge 2020. Photo-realistic synthetic images significantly improve the performance on datasets that do not have real training images.

3.4.6 Ablation study

This section analyzes components of Pix2Pose to clarify how each component contributes to the performance of the method.

Transformer loss Figure 3.10 and 3.11 show the outputs of the network using a symmetric object in T-Less after training with different configuration. The second row of Figure 3.10 shows failure predictions around view boundaries, 0 and π, when the view-limitation strategy is applied. However, the transformer loss successfully guides the network to produce consistent values for symmetric poses (see the third row in Figure 3.10. On the other hand, the network trained with the simple L1 loss successfully predicts all poses when objects are fully visible. This is because the network is able to distinguish the side of the object by the crucial clue, the upper center of the object. As shown in Figure 3.11, the network trained with the L1 loss fails to predict correct outputs when this crucial part is occluded while the network trained with the transformer loss produces precise estimations regardless of occlusion. As the transformer loss does not penalize estimations for different symmetric poses, the loss implicitly guides the network to be converged to produce consistent values for similar appearances from symmetric poses.

Imprecise 3D models The evaluation on T-Less already shows the robustness to 3D CAD models that have small geometric differences with real objects. However, it is often difficult to build a 3D model or a CAD model with refined meshes and precise geometries of a target object. Thus, a simpler 3D model, a convex hull covering

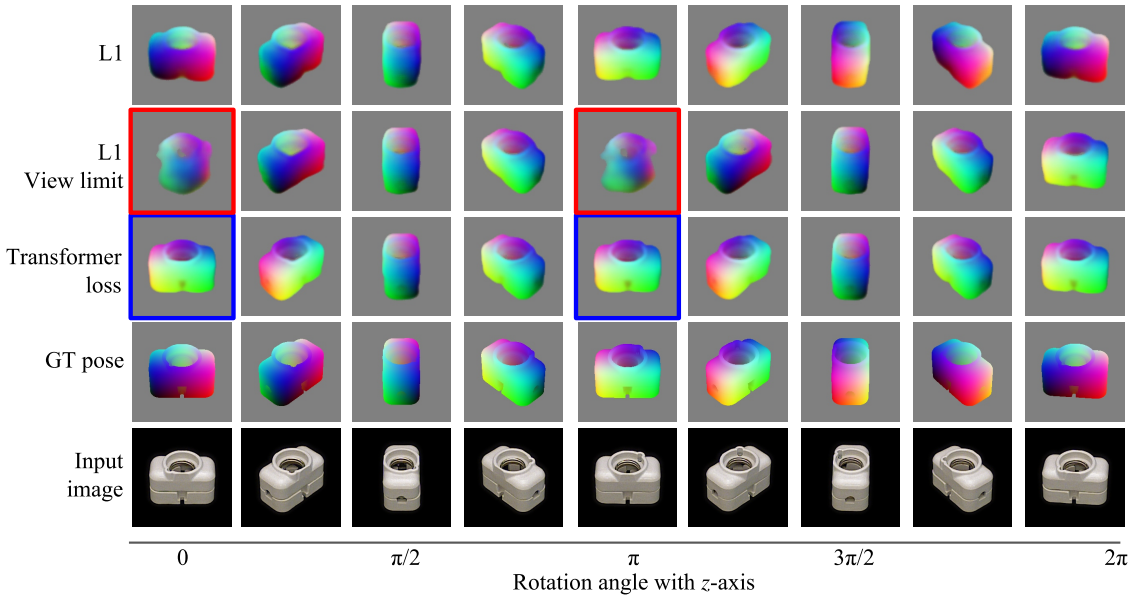

Figure 3.10: Prediction results of varied rotations with the z-axis. The L1 loss with the view limit strategy causes noisy predictions at boundaries, 0 and π, as denoted with red boxes. The transformer loss implicitly guides the network to predict a single side consistently for the symmetric poses (blue boxes.)

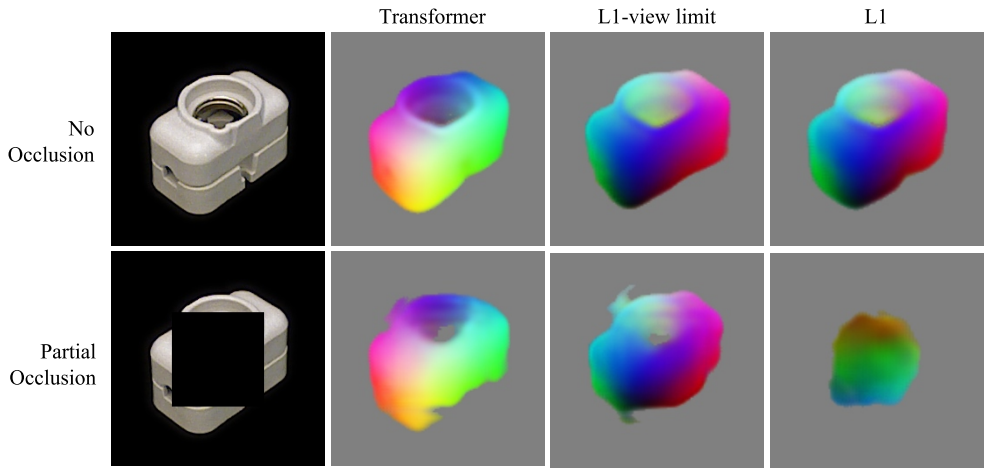

Figure 3.11: Prediction results with/without occlusion for a symmetric object. For the network trained by the L1 loss, it is difficult to predict the exact pose when the upper part, which is a clue to determine the pose, is not visible. The prediction of the network using the transformer loss is robust to this occlusion since the network consistently predicts a single side.

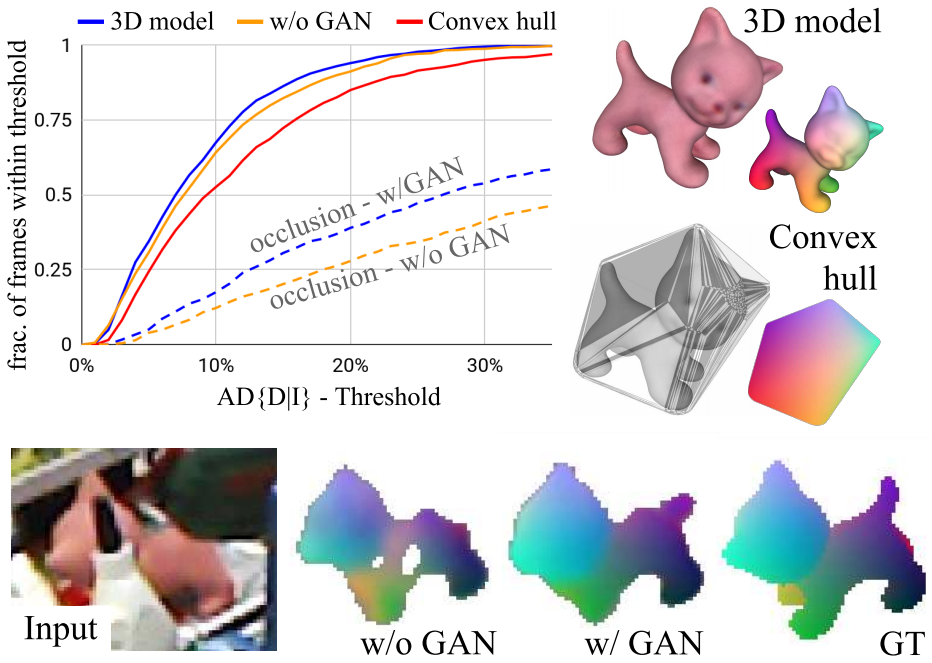

Figure 3.12: Results of the ablation study. Top: the fraction of frames within AD{D|I} thresholds for the `cat` in LineMOD. The larger area under a curve means better performance. Bottom: qualitative results with/without GAN.

out-bounds of the object, is used in this experiment as shown in Figure 3.12. The training and evaluation are performed in the same way for the LineMOD evaluation with synchronization of object masks using annotated masks of real images. As shown in the top-left of Figure 3.12, the performance slightly drops when using the convex hull. However, the performance is still competitive with methods that use 3D bounding boxes of objects, which means that Pix2Pose uses the details of 3D coordinates for robust estimation even though 3D models are roughly reconstructed.

Contribution of GAN The network of Pix2Pose can be trained without GAN by removing the GAN loss in the final loss function in Equation (3.5). Thus, the network only attempts to reconstruct the target image without trying to trick the discriminator. To compare the performance, the same training procedure is performed without GAN until the loss value excluding the GAN loss reaches the same level. Results in the top-left in Figure 3.12 shows the fraction of correctly estimated poses with varied thresholds for the ADD metric. Solid lines show the performance on the original LineMOD test images, which contains fully visible objects, and dashed lines represent the performance on the same test images with artificial occlusions that are made by replacing 50% of areas in each bounding box with zero. There is no significant change in performance when objects are fully visible. However, the performance drops significantly without GAN when objects are occluded. Examples at the bottom of Figure 3.12 also show training with GAN produces robust predictions on occluded parts.

2D Detector	SSD-6D [48]	Retinanet [7]	R-CNN [5]	GT bbox
2D bbox	89.1	97.7	98.6	100
6D pose	64.0	71.1	72.4	74.7
6D pose/2D bbox	70.9	72.4	73.2	74.7

Table 3.9: Average percentages of correct 2D bounding boxes (IoU>0.5) and correct 6D poses (ADD-10%) on LineMOD using different 2D detection methods. The last row reports the percentage of correctly estimated poses on scenes that have correct bounding boxes (IoU>0.5).

Robustness to 2D detectors Table 3.9 reports the results using different 2D detection networks on LineMOD. Retinanet and Faster R-CNN are trained using the same training images used in the LineMOD evaluation. In addition, the public code and trained weights of SSD-6D [48] are used to derive 2D detection results while ignoring pose predictions of the network. It is obvious that pose estimation results are proportional to 2D detection performances. On the other hand, the portion of correct poses on good bounding boxes (those that overlap more than 50% with ground truth) does not change significantly. This shows that Pix2Pose is robust to different 2D detection results when a bounding box overlaps the target object sufficiently. This robustness is accomplished by the refinement in the first stage that extracts useful pixels with a re-centered bounding box from a test image. Without the two-stage approach, the performance significantly drops to 41% on LineMOD when the output of the network in the first stage is used directly for the PnP computation.

Inference time The inference time varies according to the 2D detection networks. Faster R-CNN takes 127ms and Retinanet takes 76ms to detect objects from an image with $640\times480px$. The pose estimation for each bounding box takes approximately 25-45ms per region. Thus, our method is able to estimate poses at 8-10 fps with Retinanet and 6-7 fps with Faster R-CNN in the single object scenario.

3.4.7 Failure cases

There are three primary reasons for failure cases: (1) Poses that are not covered by real training images and the augmentation. (2) Ambiguous poses due to severe occlusion. (3) Not sufficiently overlapped bounding boxes, which cannot be recovered by the bounding box adjustment in the first stage. The second row of Figure 3.13 shows that the random augmentation of in-plane rotation during the training is not sufficient to cover various poses. Thus, the uniform augmentation of in-plane rotation has to be performed for further improvement.

3.5 Discussion

This chapter introduced a novel architecture, Pix2Pose, for 6D object pose estimation from RGB images. Pix2Pose addresses several practical problems that arise during pose

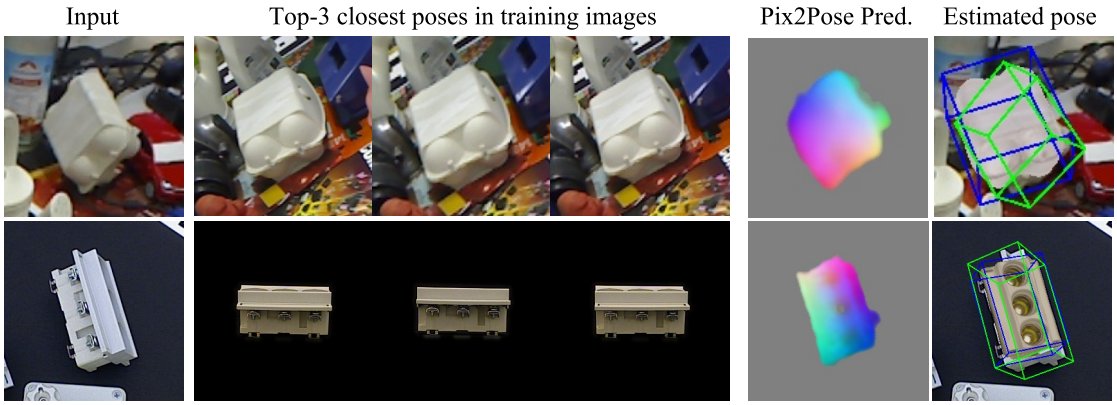

Figure 3.13: Examples of failure cases due to unseen poses. The closest poses are obtained from training images using geodesic distances between two transformations (rotation only).

estimation: the difficulty of generating real-world 3D models with high-quality texture as well as robust pose estimation of occluded and symmetric objects. Evaluations with three challenging benchmark datasets show that Pix2Pose significantly outperforms state-of-the-art methods while solving these aforementioned problems.

Evaluation results reveal that many failure cases are related to unseen poses that are not sufficiently covered by training images or the augmentation process. Since it is difficult to obtain textured 3D models in good quality from the real-world, it is important to synthesize images of objects in unseen poses using a limited number of training images. In this case, a few real images are sufficient for training the Pix2Pose network while reducing the effort for collecting and annotating images. This challenging problem is addressed in Chapter 4. As a summary, this chapter introduced the following contributions.

Hightlights

(1) A novel framework for 6D pose estimation, Pix2Pose, that robustly regresses pixel-wise 3D coordinates of objects from RGB images including invisible areas.
(2) A novel loss function, the transformer loss, for handling symmetric objects that have a finite number of ambiguous views.
(3) Experimental results on three different datasets, LineMOD, LineMOD Occlusion, and T-Less, showing that Pix2Pose outperforms the state-of-the-art methods even if objects are occluded or symmetric.

Chapter 4

Neural Object Learning

4.1 Motivation

In the previous chapter, it is shown that Pix2Pose is able to predict poses of objects reliably when a dataset provides a sufficient number of real training images. However, as discussed in Section 3.4.7, the failure cases are usually caused when a pose in a test image is not observed in real training images. Thus, the real training images should uniformly cover poses of the object to train a pose estimator robustly, which is difficult to achieve in the real environment where objects are often occluded by other objects and movements of cameras are limited. Furthermore, it is difficult to annotate 6D poses of an object to more than 100 images while covering entire viewpoints of the object.

On the other hand, it is possible to create textured 3D models using real images captured from known camera poses. The 3D models are then used to render objects from uniformly sampled viewpoints. The textured 3D models included in standard pose benchmarks are created with special scanning devices, such as the BigBIRD Object Scanning Rig [58] or a commercial 3D scanner [64]. When scanning objects, a single object is usually placed with a simple background and consistent lighting condition without surrounding objects. Camera poses are precisely obtained by using visible markers or multiple cameras with known extrinsic parameters. However, this well-constrained setup is not realizable in the real environment again, where the target objects are placed in a cluttered scene without any marker, which produces imprecise camera pose information. This motivates us to develop a new approach to create images of objects from arbitrary viewpoints using a small number of cluttered images for the purpose of training object detectors and pose estimators.

In this chapter, we address the following research questions.

Research Questions

(1) Is it possible to train object detectors and pose estimators using a few cluttered images of an object?
(2) How to synthesize images of an object in a novel pose using a small number of images?
(3) How to deal with noisy pose annotations of source images?

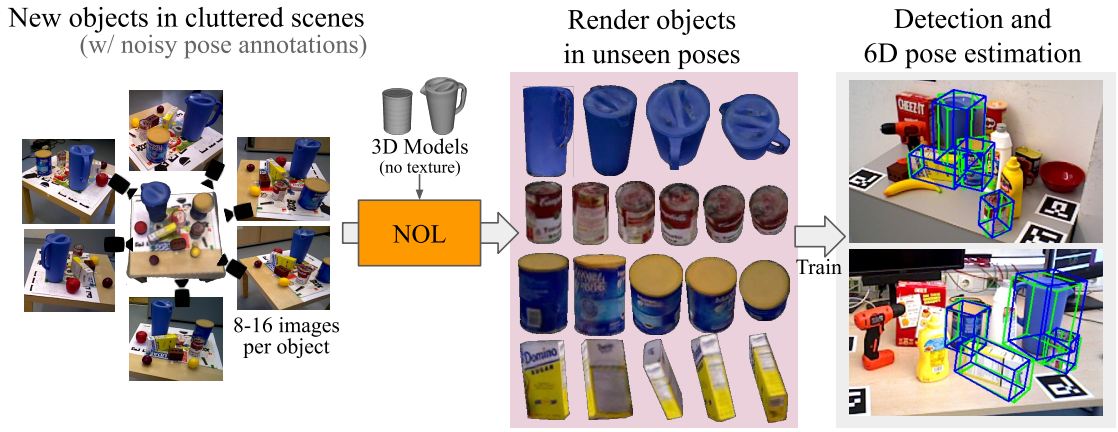

Figure 4.1: The overall concept of Neural Object Learning (NOL) for pose estimation. NOL uses a few cluttered scenes that consist of new target objects of which to render images in arbitrary poses. Rendered images are used to train pipelines for 6D pose estimation and 2D detection of objects

This chapter introduces Neural Object Learning (NOL), a method to synthesize images of an object in arbitrary poses using a few cluttered images with pose annotations and a non-textured 3D model of the object. The purpose of NOL is to train CNN-based pose estimators for new objects while minimizing the effort for obtaining training data from real environments. The knowledge of 3D representation and a few observations of objects are usually sufficient for humans to recognize objects in a new environment. Likewise, NOL composes appearances of objects in arbitrary poses using 3D models and a few cluttered and unconstrained images without using depth images. To overcome pose annotation errors of source images, a novel refinement step is proposed to adjust poses of objects in the source images. Evaluation results show that images created by NOL are sufficient to train CNN-based pose estimation methods and achieves state-of-the-art performance.

4.2 Method

The overall concept of the NOL pipeline is summarized in Figure 4.1. The objective of NOL is to create an image X^D of an object in a target pose T^D using K source images, $\{I^1, I^k ... I^K\}$, with pose annotations, $\{T^1, T^k ... T^K\}$, and object masks that indicate whether each pixel belongs to the object or not in a source image, $\{M^1, M^k ... M^K\}$. We do not assume T^k or M^k to be accurate, which is common if the source images are collected without strong supervision such as marker-based localization or human annotation.

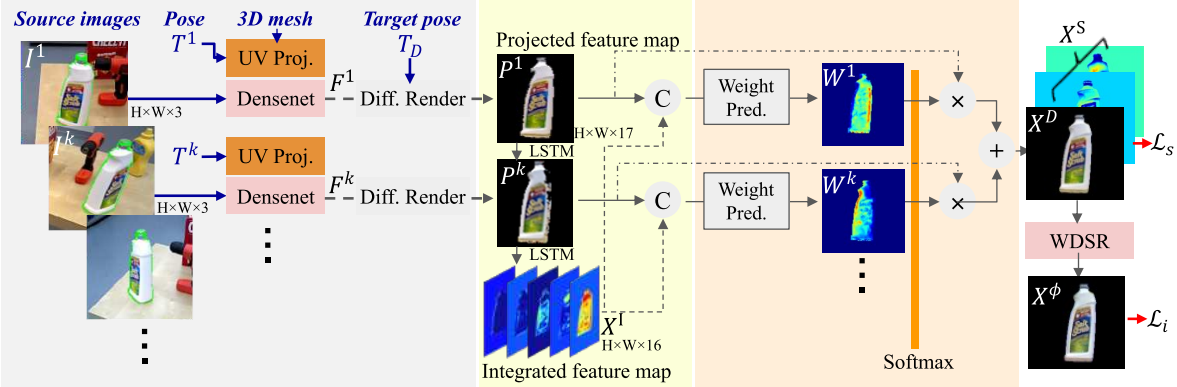

Figure 4.2: An overview of the NOL architecture. X^D is used as a rendered output while X^ϕ is used to compute the image loss for training

4.2.1 Network Architecture

Figure 4.2 depicts an overview of the network architecture. Firstly, source images are encoded and projected to compute an integrated feature map in a target pose. Secondly, the weighted sum of the projected feature maps is computed by predicting weight maps. A decoder block produces a decoded image that is used to compute the image loss.

Integrated Feature Maps Each source image, $I^k \in H \times W \times 3$, is encoded with a backbone network, Densenet-121 [126], to build feature pyramids using the outputs of the first four blocks. Each feature map from the pyramids is processed with a convolutional layer with 3×3 kernels to reduce the number of an output channel of each block to 4, 3, 3, and 3. Each feature map is resized to the size of the original input using bi-linear interpolation. In addition to the feature map with 13 channels, original color images (3 channels) and face angles with respect to camera views (one channel) are concatenated. As a result, the encoded feature map F^k of each input image I^k has 17 channels. The UV coordinates of each vertex are computed using 2D projected locations of each visible vertex in an input pose T^k. These UV coordinates are then projected to the target pose T^D using a differentiable renderer proposed in [74]. Feature values of each pixel in a projected feature map P^k are computed using bilinear interpolation of surrounding feature values obtained from corresponding pixels from the encoded feature map F^k, which is similar to rendering an object with a separate texture image. The projected feature maps $P^{k \in K}$ are compiled by convolutional Long-Term and Short-Term Memory (LSTM) layers to compute the integrated feature map X^{I} with 16 output channels at the same resolution of the projected feature maps (Figure 4.3). This LSTM layer enables the network to learn how to extract valuable pixels from different source images while ignoring outlier pixels caused by pose errors. It is also possible to use a different number of source images without changing the network architecture.

Weight Prediction Block Figure 4.4 presents the architecture of the weight prediction block. The integrated feature map X^{I} is concatenated with each projected feature map P^k to compute a corresponding weight map W^k in the weight prediction block, which implicitly encodes distances between P^k and X^{I} per pixel. The resulting weight maps

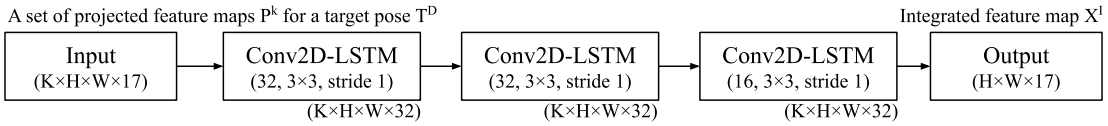

Figure 4.3: The architecture of the LSTM block that integrates projected feature maps.

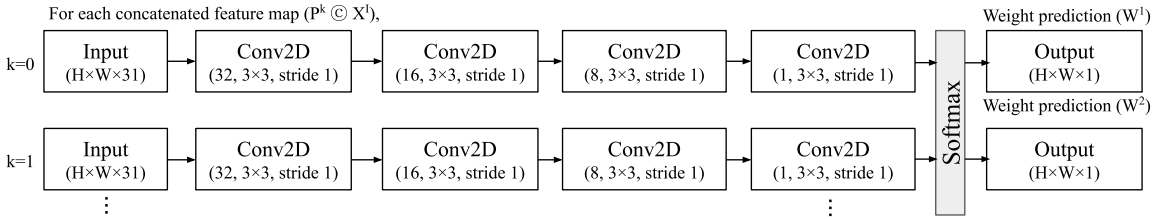

Figure 4.4: The architecture of the weight prediction block.

$W^{k \in K}$ are normalized with the *Softmax* activation function over K projected images. Therefore, the summation of the weighted maps over $W^{k \in K}$ is normalized for each pixel while keeping strong weights on pixels that have remarkably higher weights than others. The weighted sum of projected feature maps using the predicted weight maps produces the weighted feature map X^{S}. Since the first three channels of P^k represent projected color values from source images, the first three channels of X^{S} are a color image, which is referred to as the weighted rendering X^D and the output image of NOL during inference.

Decoder Block Since X^D is obtained by the weighted summation of projected images, color values of pixels X^D are limited to the color range of projected pixels. However, when training the network, the color levels of the source images can be biased by applying randomized color augmentations while maintaining the original colors for the target image. This causes the weight prediction block to be over-penalized even though color levels of X^D are well balanced for given source images. This motivates us to add a module to compensate for these biased errors implicitly during training. An architecture used for the image super-resolution task, WDSR [127], is employed as a decoder block to predict the decoded rendering X^{ϕ} in order to compute the losses during training. A detailed analysis regarding the role of the decoder is presented in Section 4.4.

4.2.2 Training

The objective of training consists of two components. The first component, the *image loss*, renders a correct image $\mathcal{L}_{\mathrm{i}}$ in a target pose. The second component, the *smooth loss*, minimizes the high-frequency noise of the resulting images $\mathcal{L}_{\mathrm{s}}$.

Image Loss The image loss $\mathcal{L}_{\mathrm{i}}$ computes the difference between a target image X^{GT} in a target pose and the decoded output X^{ϕ}. In addition to the standard L1 distance of each color channel, the feature reconstruction loss [128] is applied to guide the predicted

images to be perceptually similar to the target image as formulated by

$$\mathcal{L}_{\mathrm{i}} = \frac{1}{M^D} \sum_{p \in M^D} \lambda_i |X_p^{\phi} - X_p^{GT}|_1 + \lambda_f |\psi(X_p^{\phi}) - \psi(X_p^{GT})|_1, \tag{4.1}$$

where M^D is a binary mask that indicates whether each pixel has at least a valid projected value from any input $I^{k \in K}$, and $\psi(\cdot)$ denotes outputs of a backbone network with respect to the image. The outputs of the first two blocks of DenseNet [81] are used for the feature reconstruction loss. The parameters λ_i and λ_f are used to balance the losses.

Smooth Loss Even if the objective function in Equation (4.1) guides the network to reconstruct the image accurately, the penalty is not strong when the computed image has high-frequency noise. This is the reason why IBR and image in-painting methods [70], [72], [73] usually employ a smooth term. This minimizes the gradient changes of neighboring pixels even if pixel values are obtained from different source images. Similarly, we add a loss function to ensure smooth transitions for neighboring pixels in terms of color values as well as encoded feature values. This is formulated as

$$\mathcal{L}_{\mathrm{s}} = \frac{\lambda_s}{M^D} \sum_{p \in M^D} \nabla^2 X_p^S. \tag{4.2}$$

The loss function creates a penalty when the gradients of color and feature values of each pixel are inconsistent with neighboring pixels. In contrast to the image loss, the weighted feature map X^S is used directly instead of using the decoded output X^{ϕ}. Thus, the weight prediction block is strongly penalized when producing high-frequency changes in the predicted weight maps $W^{k \in K}$ and the weighted feature map X^S.

Training using Synthetic Images with Pose Errors Synthetic images are created to train the NOL network. 3D models from the YCB-Video dataset [49] are used while

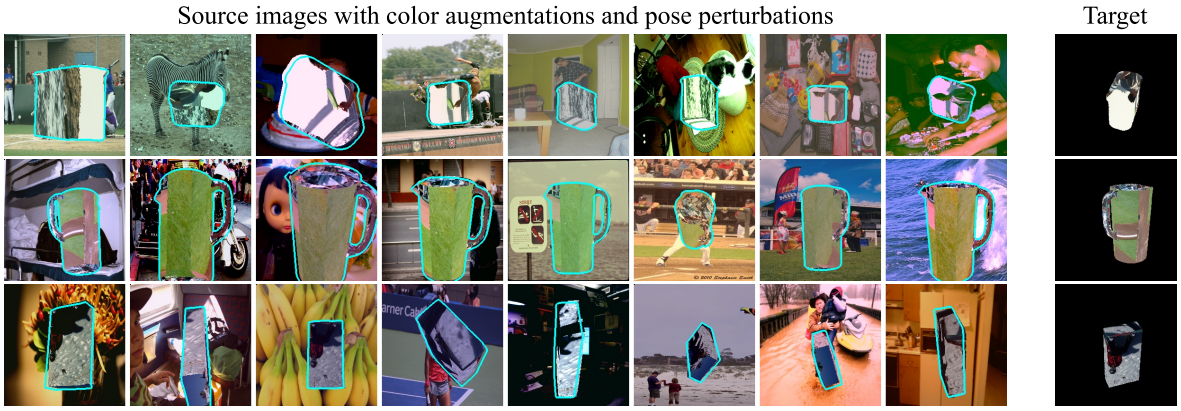

Figure 4.5: Examples of source images and target images. Color augmentations are applied to source images and pose perturbations are applied to pose annotations of source images with the parameters in Table 4.1. No augmentation is applied to target images and target poses.

Color augmentation					Pose augmentation	
Color add	Contrast norm	Multiply	Gaussian blur	Addictive noise	ΔTrans(m)	ΔRot(rad)
$\mathcal{U}$(-15, 15)	$\mathcal{U}$(.8, 1.3)	$\mathcal{U}$(.8, 1.2)	$\mathcal{U}$(0, .5)	$\mathcal{N}$(0, 10)	$\mathcal{U}$(-.01, .01)	$\mathcal{U}$(-.05, .05)

Table 4.1: Parameters of color augmentations and pose perturbations

replacing original textures with randomly sampled images of the MS-COCO [9] dataset. After sampling a 3D model and a texture image, 10 images are rendered as a batch set in different poses with random background images. During training, one image from a batch set is chosen as a target image X^{GT} and its pose is set to a desired pose T^D, and the other K images are assigned as input images $I^{k \in K}$. To simulate different lighting conditions, color augmentations are applied to the input images while no augmentation is applied to the target image. Pose errors are also simulated by applying random perturbations to the actual poses $T^{k \in K}$ of the input images during training. As a result of the perturbations, vertices are projected to wrong 2D locations, which produces wrong UV coordinates per vertex and outlier pixels in projected feature maps at a desired pose. This pose augmentation forces the network to be robust to pose errors while attempting to predict an accurate image in the target pose. A total of 1,000 training sets, consisting of 10 images per set, are rendered for training. The same weights are used to render objects in all evaluations in this chapter after training for 35 epochs. Detailed parameters used for data augmentation are listed in Table 4.1 and Figure 4.5 shows examples of mini-batches after the augmentation.

4.2.3 Gradient Based Pose Refinement and Rendering

The error in an input pose T^k causes crucial outlier pixels in the projected feature map P^k at the desired pose. Figure 4.6 shows an example of wrong pixels in the projected

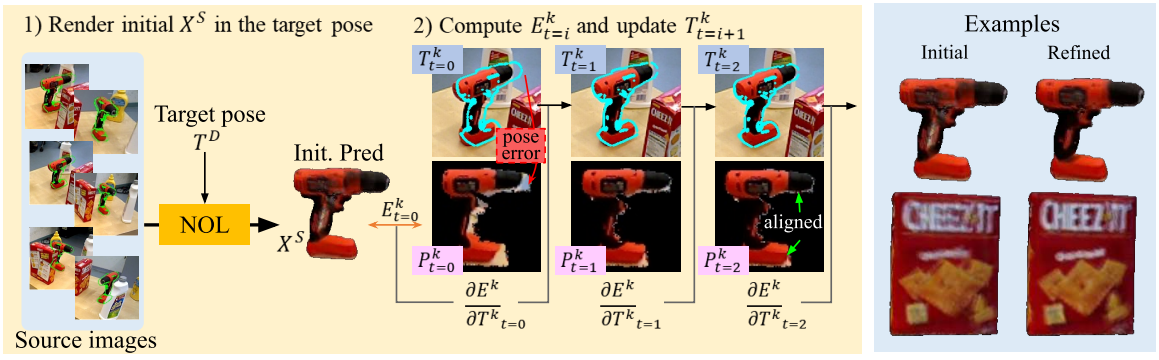

Figure 4.6: An overview of the proposed pose refinement process and example results. The partial derivative of the projection error E^k is used to update each input pose T^k

feature map obtained from the ground plane (blue) in the source image due to the error of the initial pose $T^k_{t=0}$. As discussed in Section 2.3.2, the differentiable renderer enables derivatives of 3D vertices of a 3D model to be computed with respect to the error defined in 2D space. Since 3D locations of vertices and UV coordinates in the desired pose are derived by matrix multiplications, which is differentiable, the gradient of each input pose T^k can be derived to specify a direction that decreases the difference between a desired feature map and each projected feature map P^k. The first prediction of NOL, $X^S_{t_0}$, without refinement is used as an initial desired target. The goal of the refinement step is to minimize the error, E^k, between the initial target and each projected feature map P^k. In every iteration, the partial derivative of the projection error E^k with respect to each input pose $T^k_{t=t_i}$ is computed by

$$\Delta T^k_{t_i} = \frac{\partial E^k}{\partial T^k_{t_i}} = \frac{\partial |X^S_{t_0} - P^k_{t_i}|}{\partial T^k_{t_i}}, \tag{4.3}$$

and the input pose at the next iteration $T^k_{t_i+1}$ is updated with a learning step δ, i.e. $T^k_{t_i+1} = T^k_{t_i} - \delta \Delta T^k_{t_i}$. In our implementation, translation components in T^k are directly updated using $\Delta T^k_{t_i}$. On the other hand, updated values for rotation components, $\mathbb{R}^{3\times3}$, do not satisfy constraints for the special orthogonal group, $SO(3)$. Thus, the rotation component of $\Delta T^k_{t_i}$ is updated in the Euler representation and converted back to the rotation matrix. As depicted in Figure 4.6, the iterations of the refinement step correctly remove the pose error so that the projected image no longer contains pixels from the background, which decreases blur and mismatched boundaries in the final renderings. After refining every input pose $T^k_{t=t_0}$ until the error does not decrease or the number of iterations exceeds 50, the final output X^D is predicted using the refined poses $T^k_{t=t_f}$.

4.3 Evaluation

This section presents the evaluation of the proposed NOL approach in relation to the task of 6D object pose estimation. We introduce datasets used in the evaluation and provide all implementation details of NOL. The evaluation results show that the quality of NOL images created using a few cluttered images are sufficient for pose estimation and leads to outperforming other methods trained with synthetic images using textured 3D models or real images.

4.3.1 Datasets

Three datasets are used for evaluation: LineMOD [15], LineMOD-Occlusion [16], and our new dataset. LineMOD and LineMOD-Occlusion have been used as standard benchmarks for 6D pose estimation of objects. LineMOD provides textured 3D models and 13 test sequences that have an annotated object per image. The 3D models are created by placing each object alone on a plane and performing a voxel-based 3D reconstruction [15]. LineMOD-Occlusion is created by additionally annotating eight objects presented in a test sequence in LineMOD. Previous works reporting results on these dataset have used either synthetic images using given 3D models [45], [48] or 15%

real images obtained from test sequences (150 to 200 images per object) [46], [47], [51] for training. In contrast to previous work, images created by NOL are used to train both a pose estimator and a 2D detection pipeline.

4.3.2 Single sequence-Multi Objects Training Dataset

A new dataset, Single sequence-Multi Objects Training (SMOT), is created to reflect real noise when training images are collected from a real scenario, i.e., a mobile robot with a RGB-D camera collects a sequence of frames while driving around a table to learning multiple objects and tries to recognize objects in different locations. The dataset consists of two training sequences and eleven test sequences using eight target objects sampled from the YCB-Video [49] objects. Figure 4.8 presents the target objects of the dataset. Two training sequences that include four target objects per sequence are collected by following trajectories around a small table (see Figure 4.7). Camera poses of frames are self-annotated by a 3D reconstruction method [129] while building a 3D mesh of the static scene. 3D models provided in YCB-Video are aligned to the reconstructed scenes, and corresponding object poses are computed using camera poses. No manual adjustment is performed to preserve errors of self-supervised annotations. On the other hand, test images are collected with visible markers to compute more accurate camera poses while moving the robot manually in front of different types of tables and a bookshelf. The pose of each object is manually annotated for one reference frame and the poses in other frames are computed using the relative camera poses that are jointly determined using the 2D markers [130] and the 3D reconstruction method. Additional manual adjustments are performed when poses are remarkably wrong. As a result, each object has approximately 2,100 test images. Figure 4.9 shows examples test images of each sequences. The statistics of the dataset is summarized in Table 4.2, which show the range of view points are very limited while test images have the wider ranges.

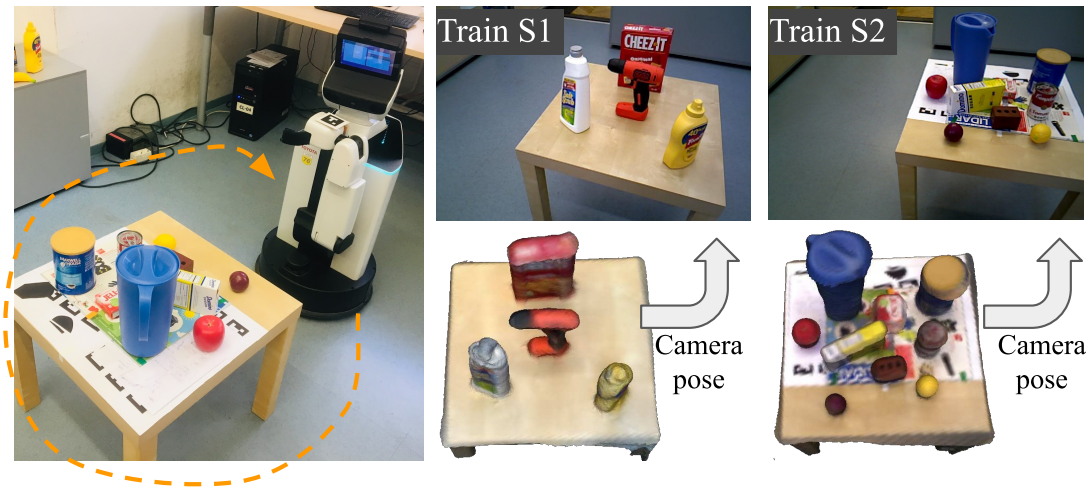

Figure 4.7: Training images of SMOT is collected using a mobile robot driving around the table.

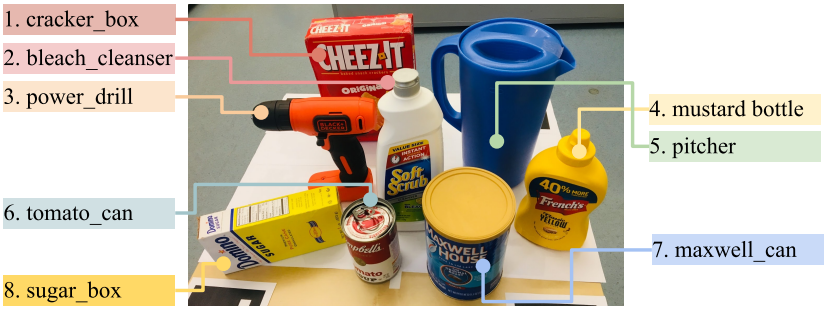

Figure 4.8: Target objects of SMOT.

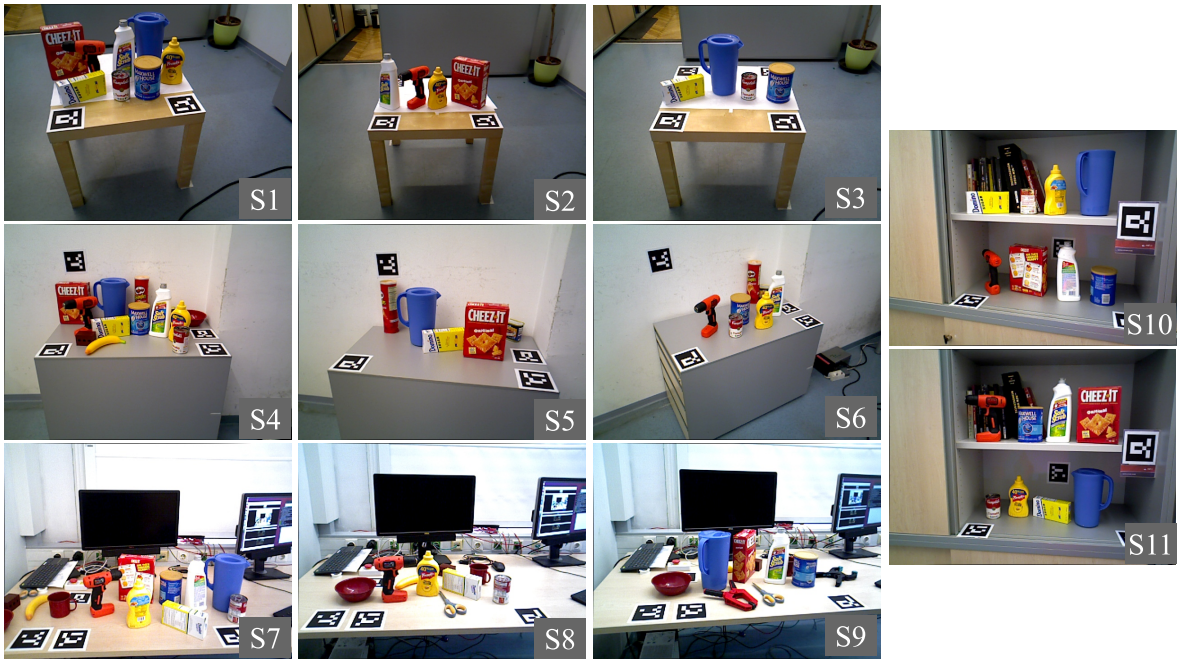

Figure 4.9: Test sequences of SMOT.

Object	cracker box	bleach	driller	mustard	pitcher	tomato can	maxwell can	sugar box
No. Test images	2155	2171	2118	2118	2090	2053	2106	2037
Train-Azimuth	(-180°, 180°)				(-180°, 180°)			
Train-Elevation	(38.3°, 40.0°)				(38.9°, 40.8°)			
Test-Azimuth	(-180°, 180°)							
Test-Elevation	(8°, 42°)							

Table 4.2: Statistics of SMOT. Training images have a limited elevation range in comparison to the range of test images.

4.3.3 Implementation Details

For the NOL network, the resolution of input and target images are set to 256×256. A number of source images, K, is set to 8 for training and 6 for inference. Thus, a mini-batch is assigned to have 8 input images with noisy pose annotations and a target image with an actual pose. The loss weights are set to, λ_i=5, λ_f=10, and λ_s=1.

Sampling of Source Images To render NOL images for training a pose estimator, source images are sampled from the training sequences of a dataset. For LineMOD and LineMOD-Occlusion, a maximum of 16 images per object are sampled from the same training splits of real images used in previous work [46], [47], [51]. Since objects are always fully visible in the training set, images are simply sampled using pose annotations. In each sampling iteration, an image is randomly sampled and images that have similar poses (less than 300mm translation and 45° rotation), are removed. The sampling is terminated when no more images remain. In contrast to LineMOD and LineMOD-Occlusion, the visibility of each object varies in the training set of SMOT. In order to minimize the number of source images, a frame with the highest value is selected at each sampling iteration by counting the number of visible vertices that have not been observed in the previously sampled frames. The sampling iteration is terminated when no frame adds additionally observed vertices.

Rendering NOL Images Each target object is rendered using NOL in uniformly sampled poses defined over an upper-hemisphere for every 5° for both azimuth and elevation. For each target pose, 6 images are chosen from sampled images using the same image sampling procedure while limiting the target vertices to visible vertices in the pose. As a result, 1296 images are rendered and for each image, 2D rotations are applied from -45° to 45° for every 15° to augment the in-plane rotation. For LineMOD and LineMOD-Occlusion, synthetic images are also rendered in the same sampled poses using given 3D models to train a pose estimator for comparison.

Training Recognizers To show whether NOL images are sufficient to estimate poses of objects in arbitrary poses using a recent RGB-based pose estimation method, Pix2Pose introduced in Chapter 3, is used. To increase the training speed and decrease the number of training parameters, the discriminator and the GAN loss are removed. All other aspects are kept the same except for the number of training iterations, which is set to approximately 14K because of the decreased number of trainable parameters. Resnet-50 [118] is used as a backbone for the encoder and weights are initialized with pre-trained weights on ImageNet [38]. Retinanet [7] with Resnet-50 is trained to supplement 2D detection results using NOL images with the same setup in Chapter 3.

4.3.4 Quality of Rendered Images

Figure 4.10 shows the rendered images using a training sequence of SMOT. Renderings in *3DRecont* show object models extracted directly from a reconstructed 3D mesh of the training scene [129]. Both MVS [69] and G2LTex [68] use the same images sampled for rendering NOL images with the same pose annotations. Multi-view texturing methods create less blurry textures than NOL for planar surfaces since they try to map an image

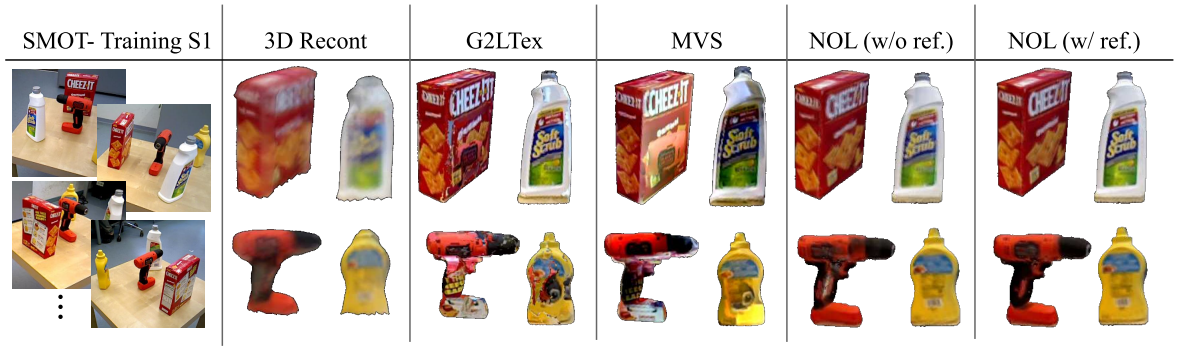

Figure 4.10: Rendered results of SMOT objects using a training sequence. NOL successfully removes pixels from the background and other objects due to pose refinement

to a large area without combining pixels from other images. However, this induces misaligned results when the input poses are inaccurate even if depth images are used to optimize poses as in [68], e.g., doubled letters on the cheeze-it box. On the other hand, results of NOL after pose refinement (last column) removes these doubled textures by correcting pose errors using color images only, which is robust to depth registration errors. Furthermore, NOL successfully rejects outlier pixels from other objects and the background.

4.3.5 Pose Estimation: LineMOD

The left side of Table 4.3 shows the results when RGB images are used for pose estimation. Since no real image is directly cropped and used to train the pose estimator, the results are mainly compared against methods that use synthetic images only for training. The method trained with NOL images outperforms the same method trained with synthetic images using the given 3D models. This verifies that the quality of NOL images is more similar to the appearances of real objects. The results of objects with metallic or shiny surfaces, e.g., `Camera`, `Phone`, and `Can`, show significant improvements against other results obtained with synthetic training images without any real observation. As depicted in Figure 4.12, NOL realizes the details of shiny and metallic materials by optimizing the colors of each view separately. The performance is competitive to the best method that uses real color and depth images of objects for domain adaptation.

NOL images tend to contain noisy boundaries, especially around the lower parts of objects where NOL mistakenly extracts pixels from the background table (see the bottom of `Iron`, `Can`, and `Phone` in Figure 4.12. This limits the translation precision of predictions along the principle camera axis (z-axis). To decrease the translation errors, ICP refinement is applied to refine poses using depth images as reported on the right side of Table 4.3. The method trained with NOL images outperforms state-of-the-art trained with synthetic images. The result is competitive to state-of-the-art results in the last two columns even though the methods [81], [131] use more than 13 times the number of real images for training.

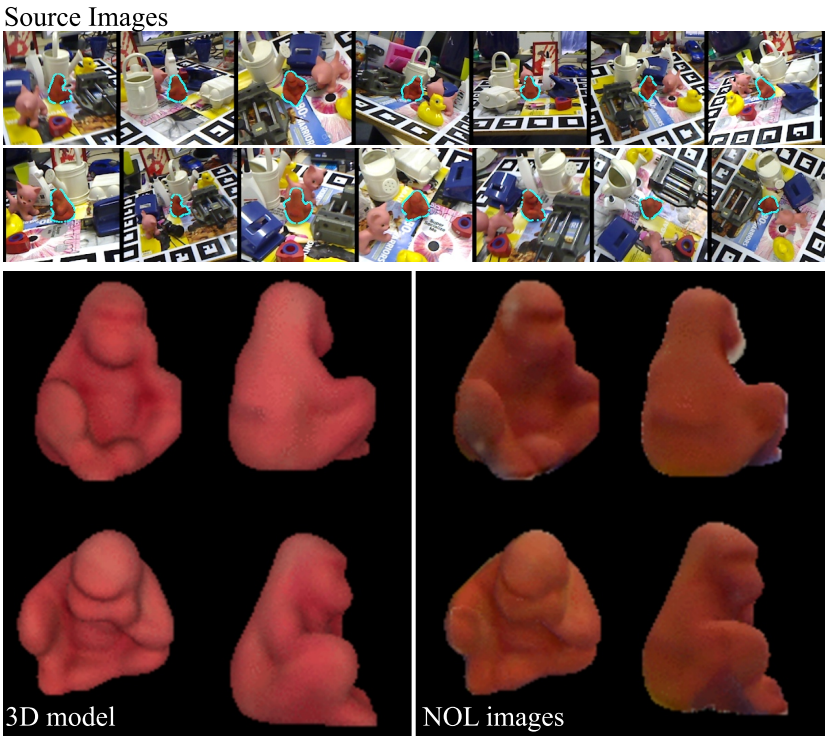

Figure 4.11: Source images and generated images of `ape` in LineMOD using NOL in comparison to the 3D model.

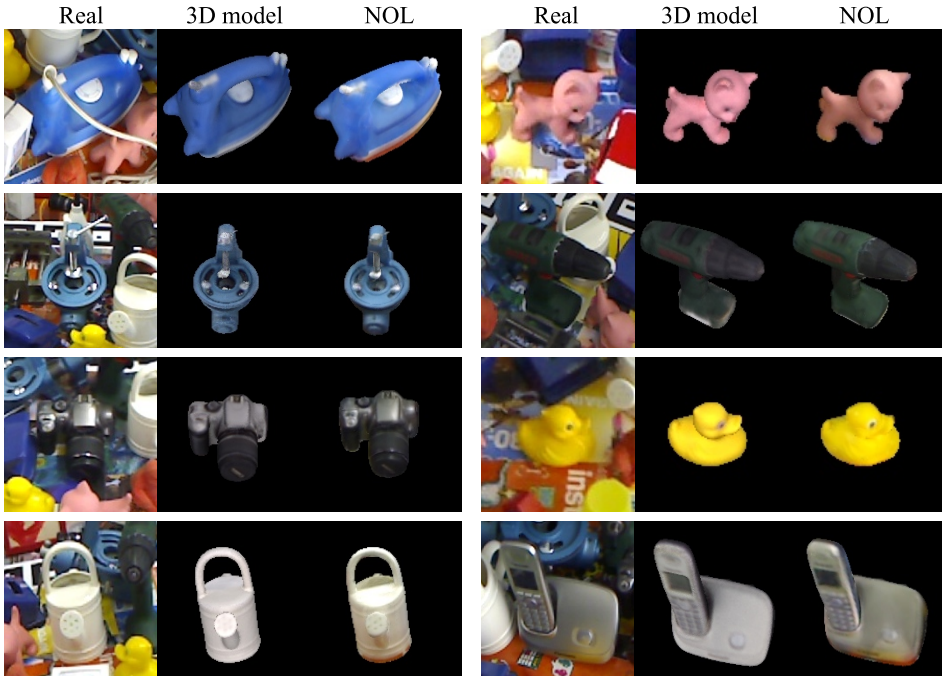

Figure 4.12: Examples of rendered images of objects in LineMOD using NOL.

Types	Required data for training										
Texture	-	✓	✓	✓	✓	✓	-	✓	✓	-	✓
No. real images	14	-	-	-	-	183	14	-	-	183	183
GT 6D pose	✓	-	-	-	-	-	✓	-	-	✓	✓
Depth image	-	-	-	-	-	✓	-	-	-	✓	✓
Test type	RGB w/o refinement						RGB + ICP			RGB-D	
Training on	**NOL**	synthetic images				real+syn	**NOL**	syn		real+syn	
Method	Pix2Pose	Pix2Pose	[48]	[45]	[59]	[132]	Pix2Pose	Pix2Pose	[45]	[81]	[131]
Ape	**35.4**	10.0	2.6	4.0	35.1	19.8	**95.2**	92.7	20.6	92.3	**97.3**
Benchvise	55.6	13.4	15.1	20.9	59.4	**69.0**	**99.0**	90.4	64.3	93.2	**99.7**
Camera	37.5	4.4	6.1	30.5	15.5	**37.6**	**96.6**	77.9	63.2	94.4	**99.6**
Can	**65.5**	26.4	27.3	35.9	48.8	42.3	**97.6**	85.8	76.1	93.1	**99.5**
Cat	**38.1**	24.8	9.3	17.9	28.1	35.4	**98.6**	90.1	72.0	96.5	**99.8**
Driller	52.2	9.1	12.0	24.0	**59.3**	54.7	**98.0**	66.1	41.6	87.0	**99.3**
Duck	14.7	3.7	1.3	4.9	25.6	**29.4**	**89.1**	82.3	32.4	92.3	**98.2**
Eggbox	**93.7**	34.6	2.8	81.0	51.2	85.2	**99.2**	88.7	98.6	99.8	**99.8**
Glue	63.1	35.1	3.4	45.5	34.6	**77.8**	**96.5**	92.2	96.4	**100**	**100**
H.puncher	34.4	3.7	3.1	17.6	17.7	**36.0**	**93.2**	46.3	49.9	92.1	**99.9**
Iron	57.9	30.4	14.6	32.0	**84.7**	63.1	**99.3**	93.5	63.1	97.0	**99.7**
Lamp	54.2	6.7	11.4	**60.5**	45.0	75.1	**96.6**	39.3	91.7	95.3	**99.8**
Phone	41.8	13.8	9.7	33.8	20.9	**44.8**	**92.8**	79.1	71.0	92.8	**99.5**
Average	49.5	16.6	9.1	28.7	40.5	**51.6**	**96.3**	78.8	64.7	94.3	**99.4**

Table 4.3: Evaluation results on LineMOD. The ADD score is used except for `Eggbox` and `Glue` that use the ADI score.

Type	RGB w/o refinement					RGB+ICP3D			Depth
Train source	**NOL**	Syn. using 3D models				**NOL**	Syn		3D model
Method	Pix2Pose	Pix2Pose	[45]	[61]	[59]	Pix2Pose	Pix2Pose	[45]	[37]
BOP Score	**37.7**	20.0	14.6	37.4	16.9	**61.3**	45.3	23.7	58.2

Table 4.4: Evaluation results on LineMOD-Occlusion. The results of other methods are cited from the 6D pose challenge 2019 [18].

4.3.6 Pose Estimation: LineMOD-Occlusion

The same models used in the LineMOD evaluation are used to test on LineMOD-Occlusion as reported in Table. 4.4. Similar to the LineMOD evaluation, methods trained by synthetic images are mainly compared. The evaluation protocol used in the recent pose challenge [18] is applied with the same test target images. The result of Pix2Pose using synthetic images is obtained by re-training the network with Resnet-50 backbone, which performs better than the official result in the challenge [18].

The performance of this method is significantly improved by using images created by NOL for training with RGB inputs and with the inclusion of ICP refinement using depth images. Furthermore, using NOL images leads to the method outperforming state of the art using color images [61] and the best performing method on this dataset [37].

4.3.7 Pose Estimation: SMOT

The Pix2Pose network and the 2D detection method [7] are trained using crops of entire real images where each object is visible more than 50%. This is an average of

Type	RGB				RGB-D (ICP3D)			
3D Model	Precise			Recont	Precise			Recont
Train source	Real	G2Ltex	**NOL**	**NOL**	Real	G2Ltex	**NOL**	**NOL**
cracker_box	30.8	24.8	49.5	45.4	85.2	92.5	96.3	88.6
bleach_cleanser	19.1	27.5	32.7	12.8	94.0	89.9	93.6	64.9
driller	23.8	2.3	19.8	26.0	87.8	53.9	96.4	91.2
mustard	2.0	33.2	25.9	19.0	88.3	73.8	89.7	82.0
pitcher*	25.9	21.7	30.9	34.8	93.3	92.9	88.7	96.1
tomato_can*	36.7	17.5	41.3	11.1	86.9	79.0	84.9	71.7
maxwell_can*	37.6	40.9	54.7	18.3	95.3	94.2	93.3	84.6
sugar_box	23.7	37.9	29.0	12.3	60.8	79.9	76.9	55.2
Average	25.0	25.7	35.5	22.5	86.5	82.0	90.0	79.3

Table 4.5: Object-wise results of the SMOT evaluation. The ADD metric is used except for symmetric objects, marked with (*), that are evaluated with the ADI metric.

364 images per object. For G2Ltex and NOL, up to 16 images per object are sampled as explained in Section 4.3.3 to render training images.

Table. 4.5 shows pose estimation and 2D detection results in terms of the AD{D|I} score and the mean Average Precision (mAP) [133]. The results using NOL images outperform other methods using real images and models textured by G2Ltex for both RGB and RGB-D inputs. This is because real images do not fully cover target poses and objects are often occluded by other objects in training images. The comparison with G2LTex provides a quantitative verification regarding the better quality of renderings created by NOL using the same source images.

4.3.8 Dynamic Objects

It is possible to render an object if poses of the object are known even though objects in source images are dynamically moving. To show how the method performs with this challenging scenario, the HO3D dataset [134] is used. The dataset is collected by manipulating objects in front of multiple cameras while automatically annotating hand and object poses. Thus, annotated poses of objects potentially include errors, and objects are often occluded by fingers. We sampled source images from the training set by following the same sampling procedure. As a portion of the training set has significantly wrong pose annotations, we rejected images when pose errors are remarkable. Figure 4.13 presents source images and example results of two objects in the dataset, which shows acceptable results in this challenging configuration. Therefore, NOL is able to compose different observations of objects even though objects are captured from dynamic scenes.

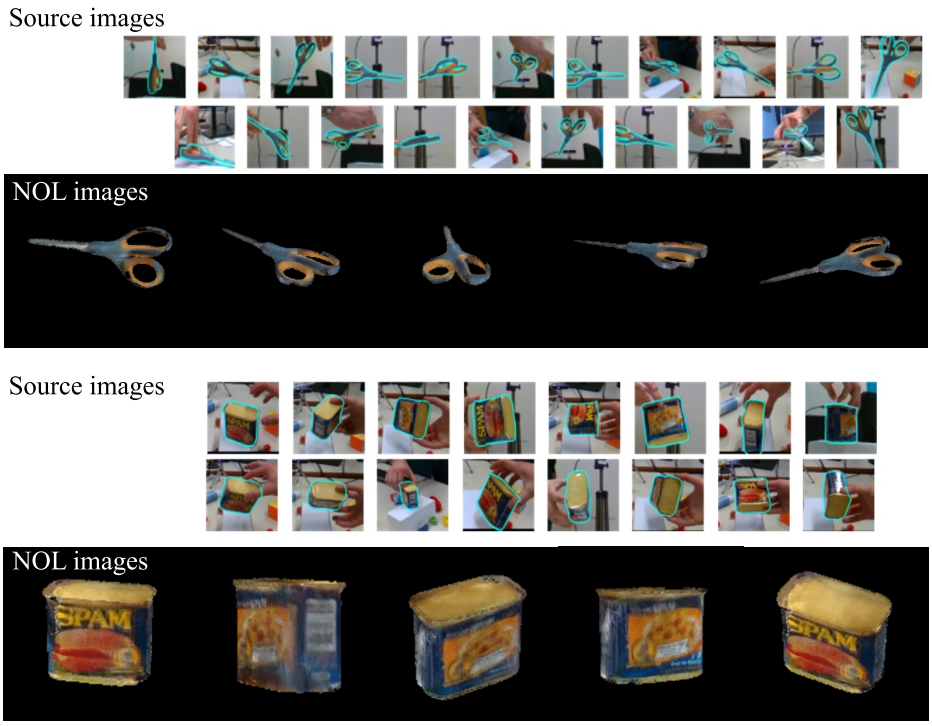

Figure 4.13: Example results using HO3D.

4.4 Ablation Study

This section analyzes factors that influence the quality of NOL images. The perceptual similarity [135] is used to measure the quality of generated images in comparison to the real images. We sample 10 test images per object in SMOT (80 images), render the objects at GT poses, and compare them with real images.

4.4.1 Each component

Table 4.6 shows the most significant improvement comes from the decoder. Figure 4.14 shows qualitative results of weighted renderings X^D, decoded renderings X^ϕ, and results after training the network without the decoder. As discussed in Section 4.2.1, the network trained with the decoder converges to produce X^D in a neutral color level as a reference image while the decoder absorbs over-penalized errors caused by randomly biased colors. The results denoted as *w/o LSTM* are derived by replacing the LSTM module with a simple average over projected features P^k. In this case, the results drop significantly since the LSTM module highlights valuable pixels among projected pixels. The refinement step consistently improves the image quality for all configurations.

4.4.2 Loss functions

The best results are made with all proposed losses $\mathcal{L}_\text{i}$+$\mathcal{L}_\text{s}$. The perceptual loss in addition to the standard L1 loss significantly improves the performance by guiding the

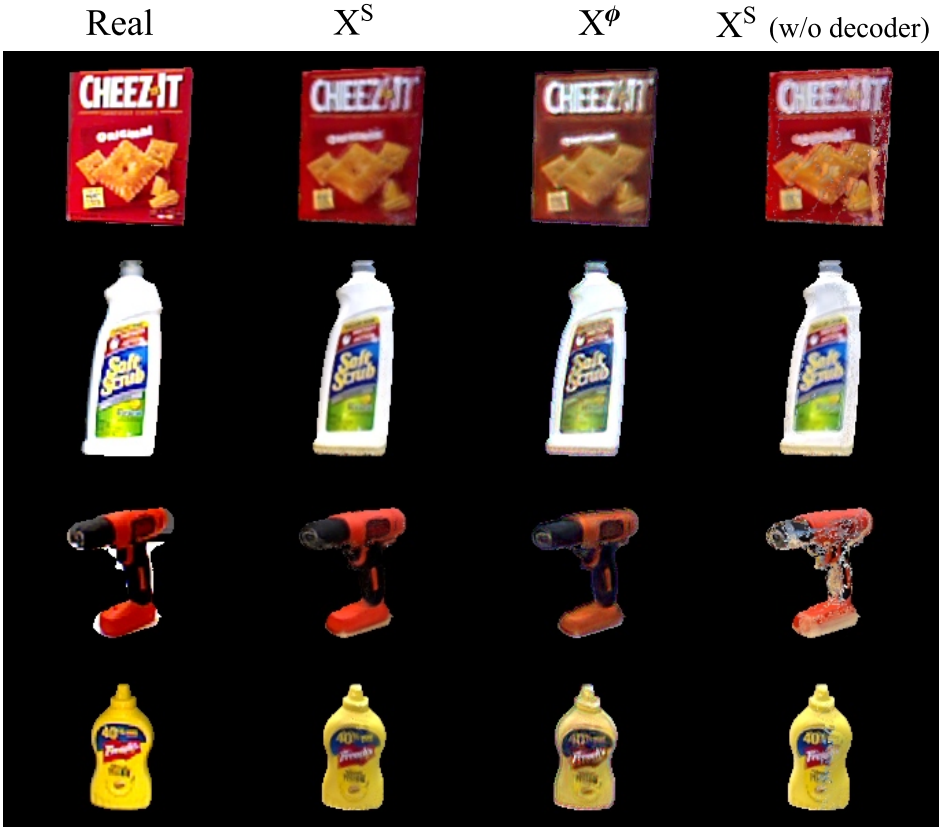

Figure 4.14: Outputs of weighted renderings, decoded renderings, and weighted renderings after training without the decoder block.

	Components			Loss functions			
Setup	All	w/o Decoder	w/o LSTM	$\mathcal{L}_1$ (RGB)	$\mathcal{L}_i$	$\mathcal{L}_i + \mathcal{L}_s$	$+\mathcal{L}_{GAN}$
w/o Ref	**0.181**	0.289	0.247	0.194	0.184	**0.181**	0.188
w/ Ref	**0.173**	0.279	0.241	0.184	0.177	**0.173**	0.183

Table 4.6: Perceptual similarity (smaller is better) of rendered images with different configurations

network to preserve perceptual details, like edges, with less blurry images while the smooth loss $\mathcal{L}_s$ additionally reduces the high-frequency noise. As the adversarial loss [54] provides better performance for image reconstruction tasks, the adversarial loss $\mathcal{L}_{GAN}$ is added to our loss function, which does not improve the result in our implementation.

4.4.3 Geometrical Errors in 3D Models

Geometrical errors of 3D models cause outlier pixels in projected feature maps that are not adjusted by the pose refinement. To analyze the effect of geometrical errors, the same evaluation with SMOT is performed using object models obtained from the 3D

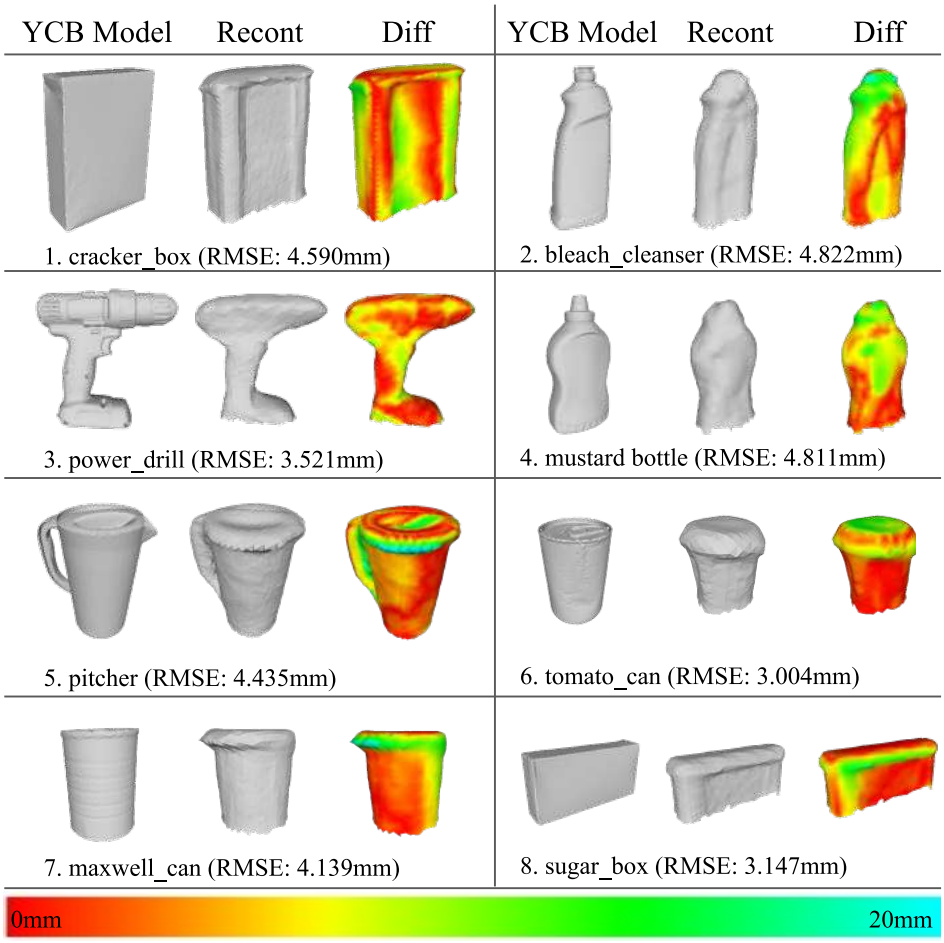

Figure 4.15: Visualization of geometrical errors measured with the Hausdorff distance.

reconstruction of training scenes. Models denoted with *3DRecont* in Figure 4.10 show examples of four SMOT objects and the geometrical error of each object is visualized in Fig 4.15. Table 4.5 includes the evaluation results denoted with *Recont* for 3D models. The results show that the performance drops since the NOL images are noisier and blurrier due to the imprecise 3D models. This indicates that precise 3D models are important for NOL to generate images with sufficient quality. Thus, the further optimization of 3D vertices is required using a few cluttered images when geometries of models are not accurate.

4.5 Discussion

This chapter introduced a novel method, Neural Object Learning, that creates training images for pose estimators using a small number of cluttered images. To the best of our knowledge, this is the first attempt to learn multiple objects from a cluttered scene for 6D pose estimation, which minimizes the effort for collecting and annotating data for training. In the evaluations, it has been shown that estimation methods trained by NOL images outperform the same methods trained by synthetic images and 13 times

the number of real images. This work highlights the fact that not only estimation methods but also the creation of training images using a few observations are important for real applications.

For future work, the method can be extended to optimize 3D models for reducing geometrical errors. This accomplishes the fully self-supervised learning of objects from cluttered scenes in real environments without any human supervision.

Hightlights

(1) Neural Object Learning that uses non-textured 3D models and a few cluttered images of objects to render synthetic images of objects in arbitrary poses without re-training the network.
(2) A novel refinement step to adjust annotated poses of an object in source images to project features correctly in a desired pose without using depth images.
(3) A new challenging dataset, Single sequence-Multi Objects Training (SMOT), that consists of two sequences for training and eleven sequences for evaluation, collected by a mobile robot, which represents a practical scenario of collecting training images of new objects in the real world.
(4) Evaluation results that show images rendered by NOL, which uses 8 to 16 cluttered images per object, are sufficient to train 6D pose estimators with state-of-the-art performance in comparison to methods that use textured 3D models and 13 times the number of real images.

Chapter 5

Mult-Task Template Matching using Depth Images

5.1 Motivation

In the previous chapters, it has been shown that using color images is sufficient for pose estimation even though only a few training images are available for a new object. However, geometric information of an object is more important than texture in robotic manipulation tasks [136], [137]. For instance, the way of grasping should be similar for boxes with similar sizes regardless of their textures and colors. In other words, many items in daily life can be manipulated properly by using previous experiences of grasping different objects that have similar geometries but different textures, e.g., mugs, bowls, and plates. Furthermore, objects in industrial domains are often texture-less without specific texture information in CAD models. Therefore, it is worth using depth information to obtain geometrical information of objects for manipulating the objects. Furthermore, depth images are also comparably easy to simulate [138], which allows the use of synthetically rendered images for training without additional domain adaptation techniques.

A drawback of CNN-based methods that directly regress poses of objects is that a network trained for a specific set of objects has to be re-trained when new objects are added. However, this is inefficient for domains that face new objects very often. In this case, training a network for each instance is not feasible. As introduced in Section 2.3.1 in Chapter 2, the same network can be used to estimate poses of different objects by matching corresponding samples that has the nearest feature values in a learned metric space [42], [43]. Real or synthetic training images of the object can be used to create a codebook of features of the object in different poses. Thus, fine-tuning of the network is not necessary for a new object, which is very efficient for domains that have to face new objects every day. This concept is also applicable to objects in a class with similar shapes when the features are trained to have closer distances to similar geometries.

This chapter introduces a novel framework, Multi-Task Template Matching (MTTM), for 6D pose estimation and segmentation of objects using a template set of depth images, which does not require further training of the network for a new object. The outputs of MTTM are the NN template with the closest pose, pixel-wise segmentation masks, and

the pose transformation from the pose of the NN template to that of the object in the test image. In summary, this chapter addresses the following research questions.

Research Questions

(1) How to design a pipeline that matches samples of new objects for detection, segmentation, and pose estimation using depth images?
(2) How to make the pipeline more robust to noisy depth images, simulate noise in synthetic images?
(3) How does the pipeline working with public benchmarks and real scenarios?

5.2 Method

This section presents the method for rendering synthetic depth images and the details of MTTM including the network architecture, the training method for multiple tasks, and the process for deriving and evaluating pose hypotheses.

5.2.1 Rendering of Noisy Depth Images

MTTM uses only synthetic depth images for training as depth images are easier to simulate than color images. Therefore, MTTM can be applied to any other domain that does not have sufficient training images. As described in [138], noisy depth images are rendered by simulating a typical stereo camera. This motivates us to implement a rendering with a sensor simulation that creates realistic depth images. Figure 5.1 presents the process of sensor simulation and examples. As shown in the resulting image, the outputs of the rendering pipeline have invalid pixels at the boundaries of each object,

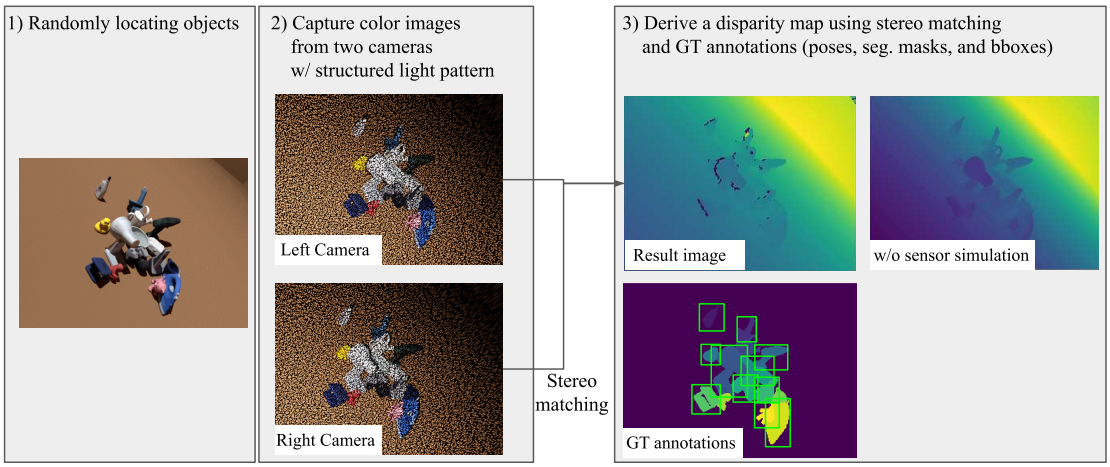

Figure 5.1: The process of generating realistic depth images by a sensor simulation. In contrast to the ideal depth image, the generated image includes pixels with invalid depth values and noise, which is common in real depth images.

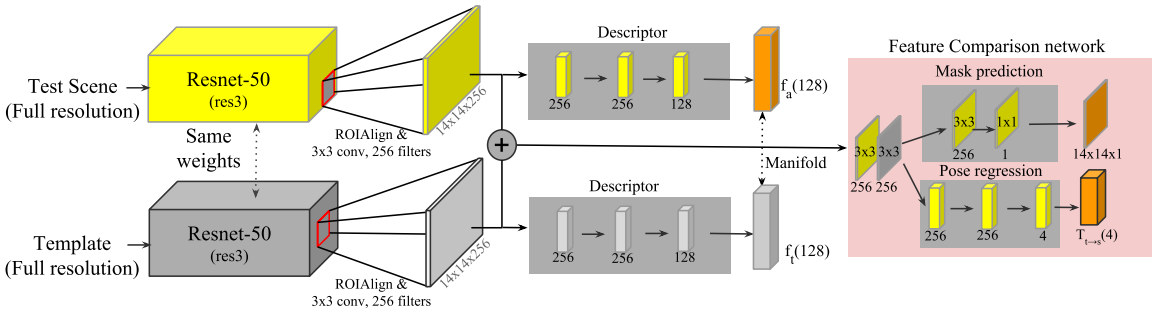

Figure 5.2: The network architecture of MTTM. Layers annotated with yellow denote the computations that are required during test time since the features of templates are precomputed and stored in the database (gray boxes).

which is the most significant difference between synthetic and real depth images. The realistic depth images enable us to train the network of MTTM without any additional augmentation of sensor noise. Training images are rendered from arbitrary camera poses after randomly placing target objects on a plane that simulate cluttered scenes. As a result of the sensor simulation, no domain adaptation technique is required to use both real and synthetic images as a template set at test time.

5.2.2 Network Architecture

The network extracts feature descriptors of given regions of interest (ROI) in a test scene to retrieve the nearest template while predicting the segmentation mask of a target object and pose transformation from the nearest template to the object in the scene. As shown in Figure 5.2, the Resnet-50 [118] architecture is used as a backbone network and initialized by the weights that are trained on the Imagenet dataset [38]. Since the original network needs a color image as an input, a depth image, which has one channel, must be converted into a three-channel image. We apply one typical technique, using x, y, and z components of surface normals as each channel of image pixels. The output of the third stage of Resnet-50 is used as a feature map of the given input image. An additional 3x3 convolution layer with 256 filters is applied to reduce the dimension of the feature map. Hence, the dimension of the shared feature map of an image is 256 channels with eight times smaller resolution than the input image. Like the method used in [6], the feature map of a ROI is cropped using bi-linear interpolation of the feature map. Different from the original ROI-Align method of [6], a simplified method is applied that crops the propositional ROI area in the feature map and resizes it to the desired output size (14x14), which works without a remarkable difference as mentioned in [6]. The ROI-Align enables the network to have any arbitrary sizes of images as inputs.

Each ROI feature map in the fixed size is used for multiple tasks: extraction of descriptors for manifold learning, mask prediction and pose regression using pairs of ROI feature maps. The descriptor is calculated by fully connected layers with filter sizes of 256, 256, and 128. The last layer has the *linear* activation while all the others

use the *elu* activation. Therefore, the dimension of the feature vector of each ROI is 128, $f \in \mathbb{R}^{128}$. This feature vector is calculated independently so that feature vectors of templates are pre-calculated and stored in a database for efficient tree search at test time.

A pair consisting of ROI feature maps from a scene and a template proceeds to 3x3 convolution layers with 256 filters separately before channel-wise concatenation of feature maps. Hence, the output dimension of the concatenated feature map is 14x14x512. This merged feature map is used for mask prediction and pose regression separately in the feature comparison network as shown in Figure 5.2. For mask prediction, a 3x3 convolution layer with 256 filters and a 1x1 convolution layer with single channel output with the *sigmoid* activation is used to represent per-pixel mask prediction. For pose regression, fully connected layers with 256, 256, and 4 output channels are applied with the hyperbolic tangent (*tanh*) activation for the last layer, which gives the pose difference of the pairs in quaternions, $q \in \mathbb{R}^4$.

5.2.3 Training Networks for Multiple Tasks

Distances of feature vectors between similar poses should be smaller than distances to different poses or different objects. Triplet loss gives reliable performance for this purpose [42]–[44]. Compared to previous work that use cropped image patches, the feature map of the entire scene is computed to crop a feature map of each ROI. For each training scene that contains multiple objects, $a_1, a_2 ... a_i$, the positive and negative templates are assigned to each object. A positive template, $t_{p,i}$, is selected from the top five closest pose templates in the same class while a negative template, $t_{n,i}$, is randomly picked from a different class or a different pose of the same object. One-half of the negative templates are from the same class while the other half is from different classes.

Symmetric objects can have close shapes even though the poses of objects are different. Balntas et al. [43] introduce a function to measure the pose similarity of an object using rendered depth images of each pose q_1, q_2 :

$$\phi(q_1, q_2) = \frac{1}{P} \sum_{P} ||s(q_1)_p - s(q_2)_p||, \tag{5.1}$$

where $s(q)_p$ denotes the depth value of the rendered image at pixel p after placing the object in the pose q. ϕ value is set to zero if the value is smaller than ϕ_l and set to one if the value is larger than ϕ_u. The pose distance of two views is defined as:

$$\delta(q_1,q_2) = \arccos(|q_1 \cdot q_2|)\phi(q_1, q_2), \tag{5.2}$$

where the pose q is the rotational pose of an object in quaternions. The negative template is accepted to be assigned when the pose distance $\delta(q_1,q_2)$ is larger than that of the positive template. The loss function for manifold learning is defined as

$$L_{\text{triplet}} = \max\left(0, 1 - \frac{||f_a - f_n||_2^2}{||f_a - f_p||_2^2 + m}\right), \tag{5.3}$$

where f_a, f_p and f_n represent feature vectors of ROIs from the anchor image, positive and negative templates. m is the dynamic margin introduced by [44] that forces the features to have larger distances to different classes, which is formulated by

$$m = \begin{cases} 2\delta(q_a,q_n) & \text{if } c_a = c_n, \\ \gamma & \text{otherwise,} \end{cases} \tag{5.4}$$

where c_a and c_n denote the class of the object in the anchor ROI and the negative template. The margin m is comparably large when the negative template is not the same class. To guide the features to have a closer distance for the similar pose [137], an additional loss for the positive pair is also applied as follows.

$$L_{\text{pose\_pair}} = ||f_a - f_p||_2^2. \tag{5.5}$$

Segmentation masks and pose transformations are computed only for positive pairs. Since the ground truth segmentation of templates is assumed to be given, the ROI feature map of a template is filtered by its segmentation mask to minimize the effect of the background. Thus, feature values that do not belong to the object are set to zero for all channels. This filtered ROI feature map is also applied when the feature descriptor is pre-calculated for templates while the ROI feature map of the test scene keeps all the values. The loss function L_{mask} for mask prediction is a mean value of *binary cross entropy loss* over the pixels of the resized ROI feature map. The pose regression loss L_{pose_reg} is the Euclidean distance between the ground truth transformation and the predicted transformation in quaternions. Finally, the total loss for all the tasks is given by

$$L = L_{triplet} + L_{pose\_pair} + L_{mask} + L_{pose\_reg}. \tag{5.6}$$

5.2.4 Object Detection and Pose Hypotheses Generation

Figure 5.3 summarizes the inference process that detects an object and creates pose hypotheses. For an input scene, center pixels are uniformly sampled and generate ROI proposals with fixed spatial size. Thus, the width and height, w, of the ROI proposal of the sampled point p is $w_p = s_{size} f / d_p$, where d denotes the depth value at each sampled pixel, f is the focal length, and s_{size} is set to cover the maximum size of target objects in 3D space. These boxes keep the aspect ratio and the spatial scale of the object in the feature map. For each ROI, a feature vector of the ROI is computed to find the NN template in the database using the efficient Kd-Tree search algorithm in Euclidean space, and the feature distance is also calculated.

From this first matching step, ROIs that have closer distances to their nearest template are selected to estimate segmentation masks. Then, the first mask prediction is performed for the selected ROIs with pre-calculated ROI feature maps of templates. Each segmentation mask from the feature comparison network is resized to its original size. To remove the redundant masks of the same object, overlapping masks are merged by the non-maximum suppression algorithm. For every iteration, the seed mask is set to the mask that has the minimum feature distance and merges masks that overlap by more than 50%. The remaining masks are used to specify the new centers of new ROI proposals.

Figure 5.3: The overview of the pose estimation process with an input depth image. Color images are used only for visualization.

In the second matching step, predicted segmentation masks are used to filter out backgrounds in feature maps to match the refined NN templates again. Then, final segmentation masks and poses are estimated. As a result, five pose hypotheses are generated for each region, R_k, using the five nearest templates T_i, a predicted segmentation mask M and rotation matrices M_i, $R_k = \{(T_k^1, M_k, Q_k^1), (T_k^2, M_k, Q_k^2) ... (T_k^5, M_k, Q_k^5)\}$. The segmentation mask is computed once for each region using the result of the first hypothesis for computational efficiency.

5.2.5 Post-Processing

Post-processing is an essential step to refine the pose and remove false detections from the pool of hypotheses. Post-processing is made easier if the CAD model of the target object is given since an exact rendering of each hypothesis is possible. It is challenging to reject and derive the best result using depth images and a set of templates without CAD models, which is going to be discussed as future work. Therefore, CAD models are used only for evaluating generated hypotheses and refining pose predictions.

In the first rejection step, regions are removed if the overlap between predicted segmentation masks and the rendered area of the first pose hypothesis is lower than 30% after three iterations of ICP refinement with down-sampled points. After removing false regions, hypotheses in the accepted regions are refined and evaluated by rendering the object in refined poses. As with previous work that uses depth images for verification [15], [36], the difference between the rendered depth image and the scene is calculated to derive the number of inlier points, N_i, the number of occluded points, N_{occ}, and the number of outliers, N_{out}, with respect to the number of rendered model points, N_m. In contrast to the depth fitness score used in [36], the occlusion penalty term is removed to detect an occluded object without penalty. Instead, the penalty term for outliers $P_O = 1 - N_{out}/N_m$ is added. Thus, the depth fitness, S_D, is derived by $S_D = P_O N_i/(N_m - N_{occ})$. The remaining parts of the evaluation such as the ratio of overlapped boundary points, S_B, and the ratio of matched surface normals, S_N, are used in the same formula of [36]. The final score is a simple multiplication of these scores, $S_{final} = S_D S_B S_N$, which is used to filter out false detections and select the best prediction.

5.3 Evaluation

This section gives implementation details and experimental results. The experiments are performed using a dataset that has heavily occluded objects in the scene, the LineMOD-Occlusion [16]. Experiments using The T-Less dataset [63] show how MTTM performs with unseen objects.

5.3.1 Implementation Details

To train the network of MTTM, both cluttered scenes with multiple objects and template images are rendered using 15 objects of the original dataset. The total number of cluttered scenes used for network training is approximately 20,000. For each training scene, objects that are occluded less than 50% are included for training. The center point of each ROI is randomly shifted for the robustness of misaligned center points. The same method of [15] is applied to sample the viewpoints of synthetic templates except for different scales since surface normals are invariant to the distance to the object from the camera. After two subdivisions of a regular icosahedron, objects are sampled from 301 viewpoints with additional in-plane rotation from $-45°$ to $45°$ with a step size of $15°$. As a result, each object's template contains 2,107 samples. The parameters are set to $\phi_l = 0.1m$, $\phi_u = 0.5m$, $\gamma = 10$ and $s_{size} = 0.3m$. All the experiments use the same network with the same weights after training of 25 epochs.

Pixels with missing values are in-painted using values of surrounding pixels for both rendered and real depth images before calculating surface normals. Surface normals of an input image are computed efficiently by calculating depth gradients [139]. The center points used in the initial detection step are uniformly sampled from every 20 pixels. During the hypothesis generation step, a hypothesis that already has a similar hypothesis in the same region is removed, and the next NN template is selected to avoid redundant hypotheses. In the post-processing step, the depth inlier threshold is set to $10mm$ except for small objects, the ape and duck, that need a smaller depth inlier threshold of $5mm$. All experiments are conducted on an Intel i7-6700K CPU and an NVIDIA GTX1080 GPU.

5.3.2 Evaluation of Segmentation

Since an advantage of MTTM is to predict segmentation masks without aligning the object to the scene, the segmentation performance is evaluated. To annotate the segmentation masks on the dataset, ground truth poses are used to place objects and calculate the difference between test images and rendered images to decide which pixel belongs to each object. A pixel is marked as a part of an object if its depth difference is less than $0.02m$. The segmentation results used in this evaluation are refined segments of each object after the second matching step in Section 5.2.4. Thus, no alignment of templates or CAD models are involved in this evaluation.

The performance is compared to other segmentation methods: using attention points of objects to correctly segment the target object from the scene [140] and an edge-based segmentation method [141]. Centers of objects in test scenes are provided as attention

Approach	**MTTM**	Attention [140]	FastGraph [141]
Object	% of correctly segmented objects (IoU>0.5)		
Ape	**80.1**	65.4	38.1
Can	**79.6**	72.8	55.1
Cat	**64.9**	59.9	33.6
Driller	78.8	**85.2**	76.3
Duck	**85.0**	60.9	58.0
Box	**75.8**	67.7	36.8
Glue	71.3	**80.6**	33.3
Hole P.	**94.9**	69.8	47.0
Average	**78.8**	70.3	47.0

Table 5.1: Results of Segmentation.

Figure 5.4: Matching results of two similar ROIs in the same image. Mask predictions are dramatically changed by retrieving NN templates from different classes. Colors are used only for visualization.

points to the method of [140]. Public code[1] that is implemented by the authors of [140] and code[2] of [141] are used for this evaluation with their default parameters. For the method of [141], we derive results with and without blurring and take the best result between them for each object.

As shown in Table 5.1, segmentation results of MTTM outperforms other methods even though they use color and depth values. This shows that MTTM uses features of NN templates to predict corresponding segmentation masks of the target objects rather than using general boundaries of arbitrary objects. The examples in Figure 5.4 show the mask predictions are dramatically changed with the retrieved templates even though the ROIs are very close. This confirms that the advantage of MTTM that matches features of templates not only for detecting an object but also for segmenting the specific object without rendering and comparing depth images. Additional results for entire images are depicted in Figure 5.5.

[1]https://rgit.acin.tuwien.ac.at/v4r/v4r
[2]https://github.com/rrg-polito/graph-canny-segm

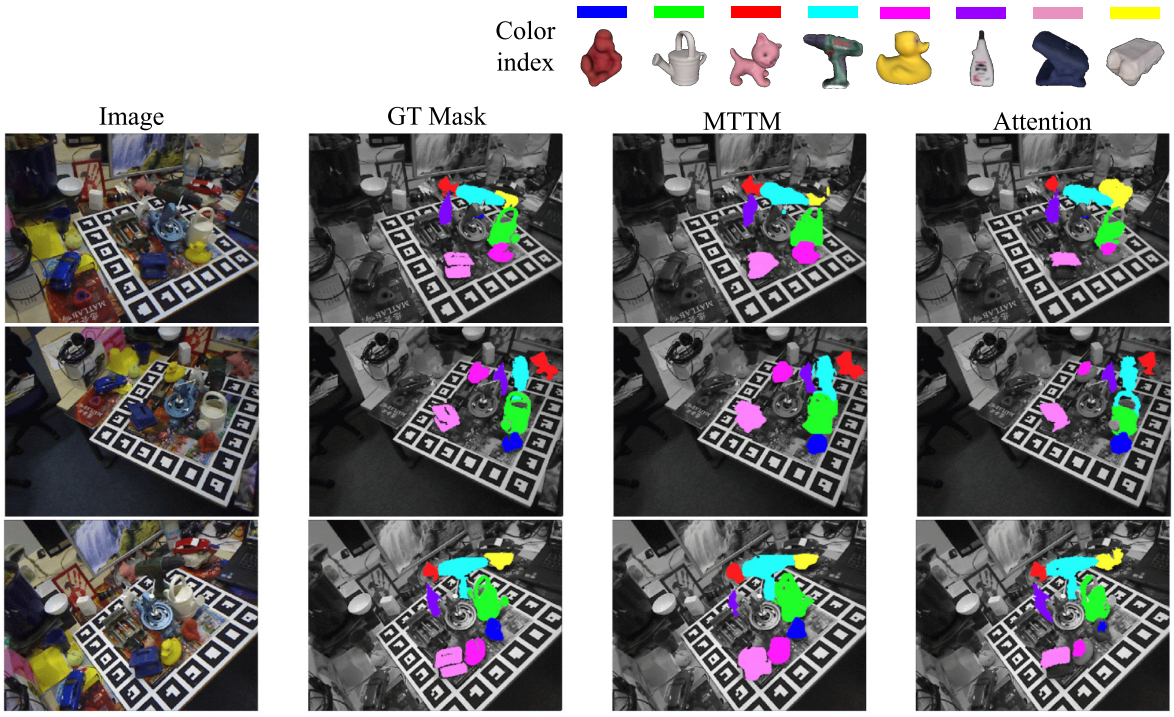

Figure 5.5: Examples of segmentation results using the LineMOD-Occlusion dataset. Results of the baseline [140] are included for comparison.

5.3.3 Evaluation of Object Detection and Pose Estimation

In this evaluation, the target object is assumed to be visible in the scene. The maximum of 50 regions that has a larger portion of inlier points is selected to calculate the final score, S_{final}, after the first rejection step in Section 5.2.5. The top 15 hypotheses that have higher scores are refined by maximum 30 iterations of ICP. Then, the final scores are re-computed to decide on the best prediction. A set of templates of an object is only included for each recognition step. Thus, each region does not compete with other object classes. To analyze the effect of the pose regression network, the same experiment is performed without pose regression, which directly uses poses of templates for initial alignment. The metric is the $AD\{D|I\}$ error that has been used to measure the pose error for the dataset [15]. The `box` and `glue` are regarded as symmetric objects so that the ADI metric is applied. Since there is no previous work that matches templates using depth images only for detection and pose estimation, we compare our method with a template-based method that has been widely used as a baseline for the original dataset [15].

Figure 5.6 shows examples of pose estimation results. As shown in Table 5.2, MTTM performs better on six out of eight objects despite the baseline method using color and depth information together. The second column reports the results without pose prediction, which shows worse performance than using pose prediction. It is clear that the pose regression network predicts a more similar pose than the original pose of the NN template except for the symmetric objects, the `box` and the `glue`. Since multiple

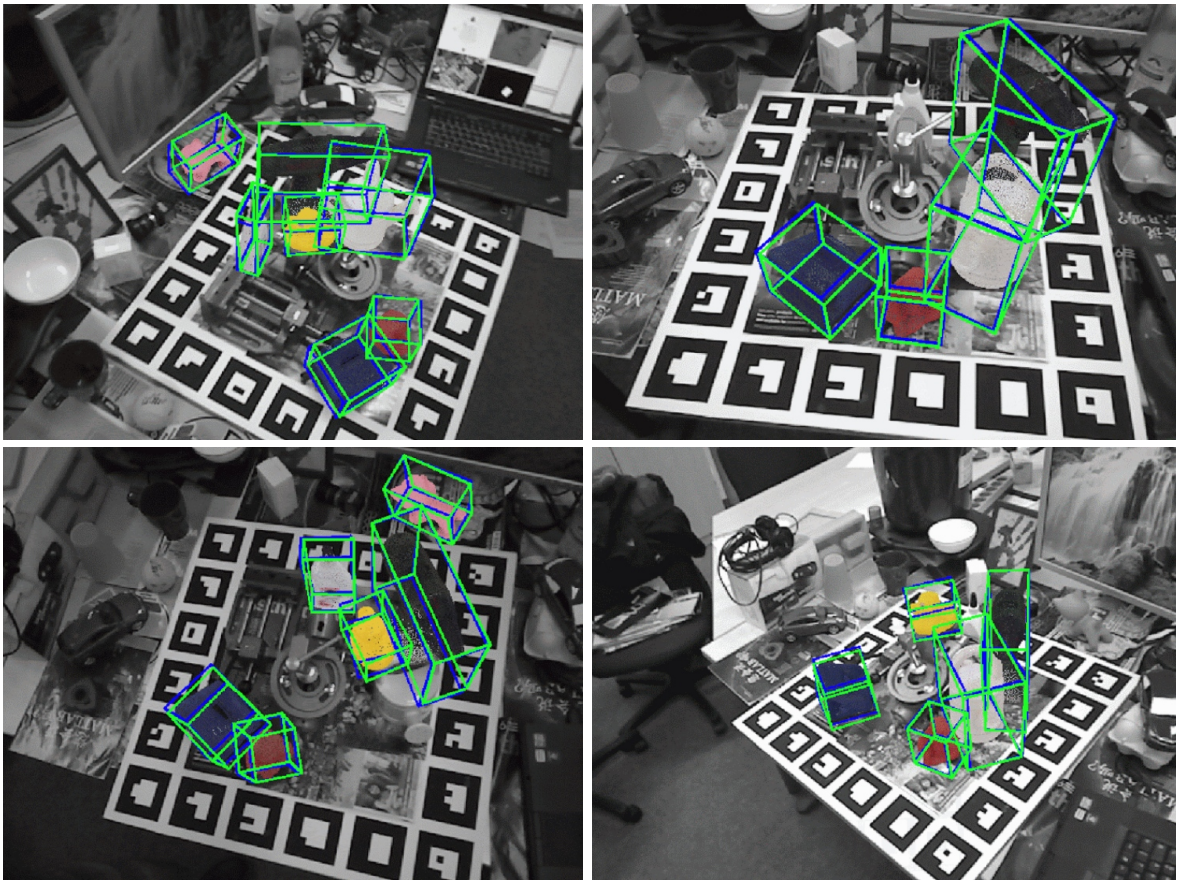

Figure 5.6: Examples of pose estimation results using LineMOD-Occlusion with depth images. Green: ground truth poses, blue: estimated poses. Color images are used only for visualization.

Approach	**MTTM**	**MTTM-NN**	Linemod [15]
Object	% of correct poses ($AD\{D\|I\} < 0.1k_d$)		
Ape	**56.7**	54.4	49.8
Can	**54.5**	50.6	51.2
Cat	**38.2**	37.9	34.9
Driller	55.4	54.2	**59.6**
Duck	56.3	55.6	**65.1**
Box	48.9	**50.7**	39.6
Glue	41.7	**42.9**	23.3
Hole P.	**71.2**	69.8	67.2
Average	**52.9**	52.0	48.8

Table 5.2: Results of Detection and Pose estimation.

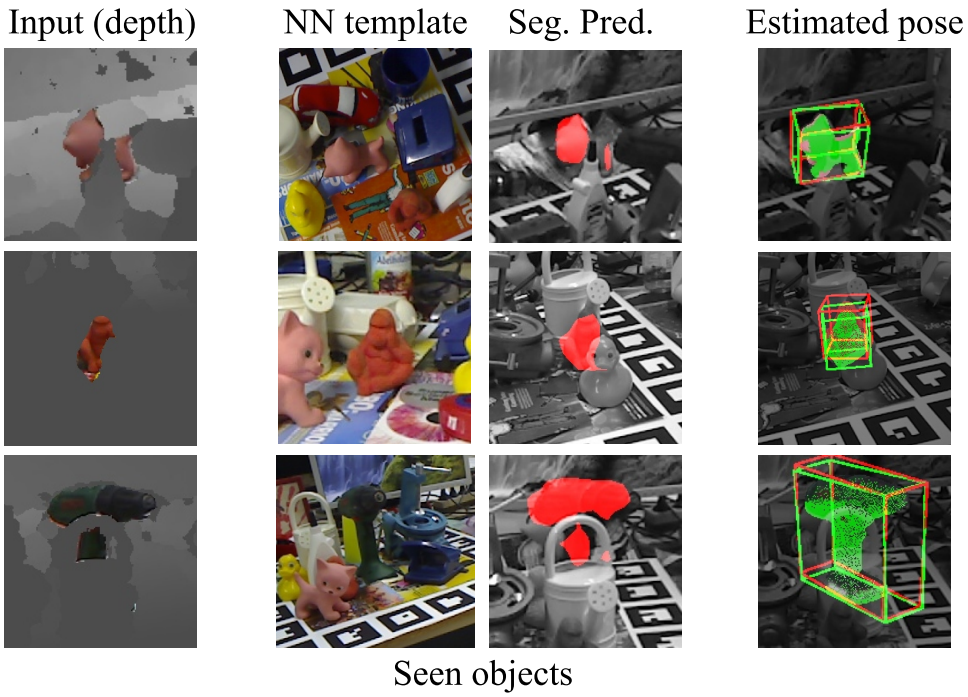

Figure 5.7: Sample results of MTTM using real templates. The database is simply replaced by real images and no further training is performed. Color images are used only for better visualization. Green: ground truth poses, red: predicted poses.

transformations exist for symmetric objects, the pose regression network sometimes tries to predict a transform from the template to a pose that is not the closest among possible candidates, which results in a less accurate estimate. This result shows it is worth using the pose regression network for non-symmetric objects. Most of the failure cases are caused by the wrong estimation of translations since MTTM predicts a rotation of an object only and estimates the center points from visible parts of objects. Thus, the initial alignments are shifted when objects are partially occluded.

5.3.4 Real Templates and New Objects

The qualitative results of using real templates and novel objects are shown in Figure 5.7. Templates of target objects are supplied by real test images of LineMOD [15] that do not overlap with test images of LineMOD-Occlusion [16]. Approximately 1,000 images per object are used as templates without heavy occlusion, and segmentation masks are derived by the same method used in the segmentation experiment. The results show the method produces good results with real templates without additional domain adaptation techniques, which is an advantage of using synthetic depth images with the sensor simulation for training.

Figure 5.8 shows recognition results of three new objects in the T-Less dataset [63] and Figure 5.9 shows examples of pose estimation results of all objects in test images. Templates are replaced by the given training images for the objects in the scene while

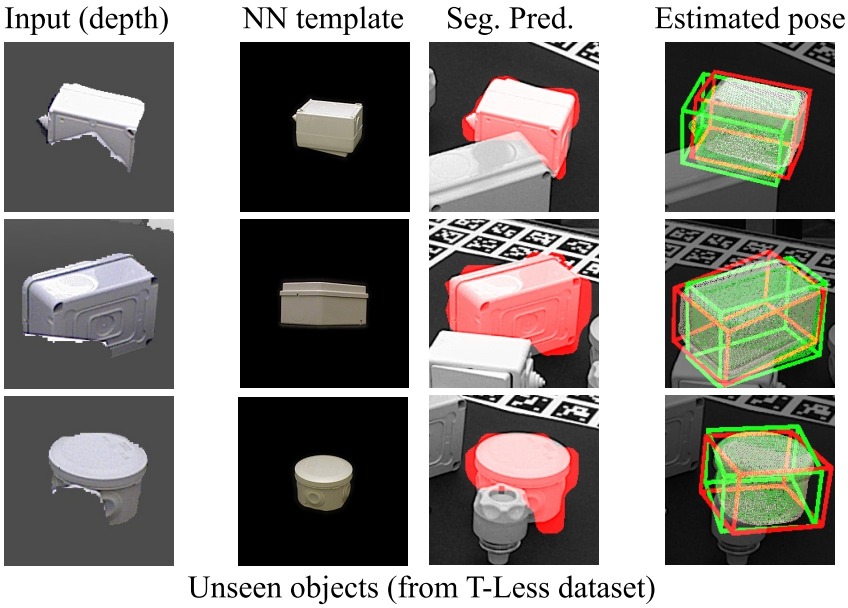

Figure 5.8: Sample results of MTTM using real templates for unseen objects.

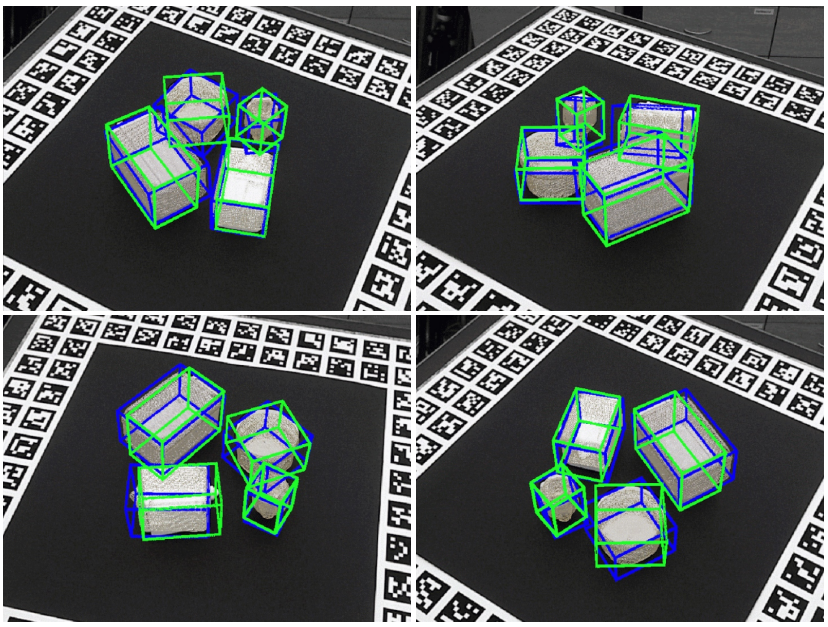

Figure 5.9: Examples of pose estimation results using T-Less with depth images. Green: ground truth poses, blue: estimated poses. Color images are used only for visualization.

the network remains the same. The results show that MTTM successfully retrieves the nearest templates, predicts segmentation masks, and estimates poses of novel objects that have not seen during the network training. Since segmentation results rely on the features of matched templates, it is more crucial for the novel objects to match correct

templates to predict precise segmentation masks.

5.4 Discussion

Learning new objects in real environments requires features derived directly from a limited number of sample scenes rather than CAD models or a large amount of training data. Furthermore, detecting objects without texture is important in order to cope with objects having similar geometry but different texture. MTTM handles this challenging problem by using learned features of templates with a multi-task architecture. Consequently, adding sample images of target objects to the template database is the only step that is necessary for a new object.

The limitation of MTTM is the lack of proper evaluation of generated pose hypotheses without CAD models. Since MTTM does not use color information and does not produce any local correspondence, it is difficult to reject false detections without ICP for computing the fitness of each hypothesis. This also causes longer computation time for evaluation, which takes up to 20 seconds while the generation of hypotheses takes between 1s and 2s per image including the retrieval of NN templates. Thus, the verification part should be improved to select the best hypothesis using matched templates directly instead of CAD models while rejecting false detections quickly.

Hightlights

(1) A novel depth-based framework, MTTM, that matches the NN template as well as predicts the segmentation mask and the pose of an object using a shared feature map, which does not require additional training for a new object.
(2) Derivation of segmentation masks without any alignment of an object to the scene, which enhances the robustness of the pose estimation performance.
(3) A rendering pipeline that creates realistic depth images by a sensor simulation.
(4) Experimental results showing that MTTM outperforms baseline methods with a more challenging configuration with the standard benchmarks, and MTTTM can be extended to new objects.

Chapter 6

Dense Correspondence Matching for Experience-based Grasping

6.1 Motivation

The previous chapter introduces MTTM that matches the nearest samples for segmentation and pose estimation of a target object. MTTM is applicable to objects with the same shape as well as slightly different shapes with similar sizes. Thus, it is possible to transfer a grasp pose of a previous successful attempt to a new object that has a similar shape by applying the estimated transformation on the grasp pose. However, predicting a global transformation is not sufficient when the target object's size is significantly different from that of the nearest sample. As shown in Figure 6.1, the location of the grasp pose can be assigned to a wrong location when the size of two objects are different, which means that computing transformations of local parts are more important. This fact has motivated previous work that estimates grasp poses based on local shapes [86],

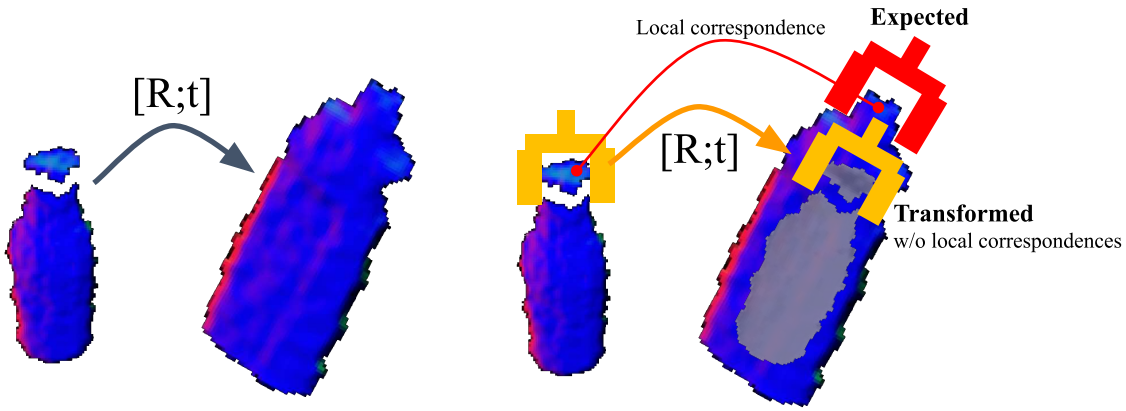

Figure 6.1: Global transformation between two objects are not sufficient to estimate good grasps when the scales of the objects are significantly different. Estimating local correspondences under global contexts is necessary to suitably locate the grasp pose.

[96], [97]. Although the local shape-based methods are good for grasping a new object, the methods do not deploy successful grasp experience with a similar object such that every attempt is independent of previous experience. Furthermore, grasp proposals are obtained without understanding global contexts that are helpful to manipulate objects for further tasks after picking them (e.g., handles of drills). Therefore, when we estimate a grasp pose of a new object, it is beneficial to leverage grasp experience if the shape of the new object is similar to one of objects in a database of successful grasp history. It is also beneficial to transfer grasp poses that are relevant to local parts of familiar objects while estimating locations of corresponding local parts of the new object based on global semantics. This chapter explores the benefits of using successful grasp experiences by addressing the following research questions.

Research Questions

(1) How to use successful grasp experience to estimate a grasp pose of a new object with a similar shape?
(2) How to predict dense geometrical correspondences between two depth images or two 2.5D point clouds while understanding the global context of geometries?
(3) Is it possible to make robots collect good grasp experience of a novel object by themselves without supervision?
(4) Is it possible to guide robots to grasp an object in a specific way (e.g., grasping a handle of a mug)?

This chapter introduces a new method for incremental grasp learning from experience. The key idea is to apply dense geometrical correspondence matching. Familiar objects are identified through global geometric encoding and associated grasps are transferred through local correspondence matching. The Dense Geometrical Correspondence Matching Network (DGCM-Net) is proposed to encode the global geometry of objects in depth images such that similar geometries are represented nearby in feature space to allow accurate retrieval of experience. DGCM-Net additionally reconstructs dense geometrical correspondences between pairs of depth images using a variant of normalized object coordinate (NOC) values, View-Dependent Normalized Object Coordinates (VD-NOC). VD-NOC values are used to represent correspondences of similar objects regardless of their poses in a camera frame. These values are used to compute the rigid transformation between the local region around the grasp of a stored experience and the corresponding region on an object in a new scene.

Advantages of DGCM-Net are presented by experiments with practical scenarios. DGCM-Net is applied in an incremental grasp learning pipeline, in which a robot self-supervises grasp learning from its own experience. Experimental results show that a robot learns to repeatably grasp the same object after one or two successful experiences and also to grasp novel objects that have comparable geometry to a known experience. As an extension, we show that the predictions from DGCM-Net improve the performance of baseline grasping methods by combining their quality measures with our experience-based measure. The incremental learning pipeline is also flexible in that grasp success is not the only measure to constitute experience. Specific positions or configurations of grasps can be preferred and therefore used in future situations. In

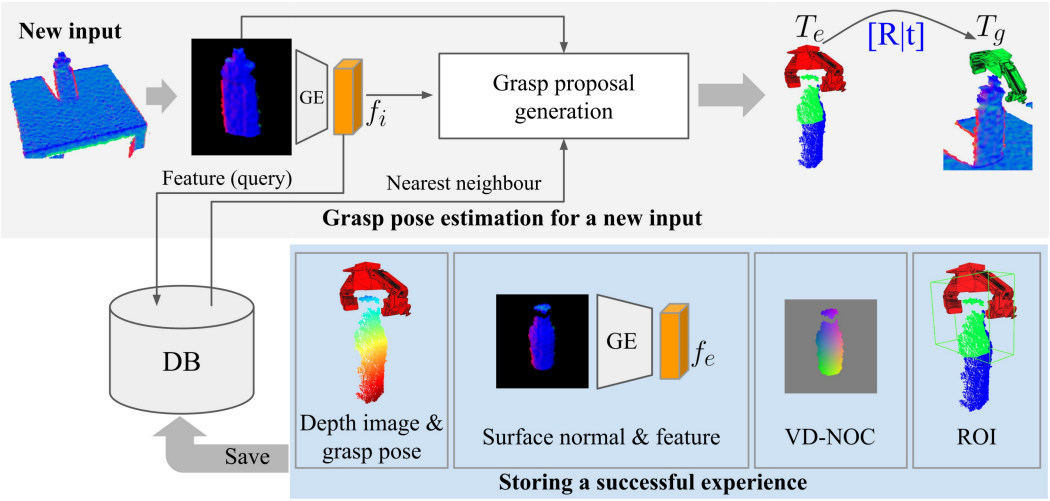

Figure 6.2: Overview of storing and retrieving experience with the incremental grasp learning framework.

particular, semantic grasps, such as grasping the handle of a mug, are prioritized as they are more relevant for the subsequent manipulation of the object [142]–[145]. As a result, task-oriented grasps are quickly learned, allowing a robot to perform meaningful actions with objects.

6.2 Method

This section describes our methodology for incremental experience-based grasp learning. We begin with an overview of the framework. We then describe the dense geometrical correspondence matching network for retrieving experience samples and generating dense 3D-3D correspondences. Lastly, we outline how grasps are transferred between local regions using the predicted correspondences.

6.2.1 Incremental Grasp Learning Framework

The main components of the incremental experience-based grasp learning framework are shown in Figure 6.2. The input is a depth image $D_i \in \mathbb{R}^{W \times H}$ and a segmentation mask $M_i \in \mathbb{R}^{W \times H}$ that has entries 1 for pixels belonging to the target object and 0 otherwise. The goal of the framework is to generate a pose for the gripper that will result in a successful grasp. This is represented as a rigid transformation $T \in SE(3)$ of the gripper pose in the camera coordinate frame.

The first step is to match the target object to samples stored in an experience database $\mathcal{E}$. Matching is done using the global geometric encoding from DGCM-Net, where the feature map f_i of the input image is compared to the feature maps of the database samples. Feature maps are the output of a geometry encoder that takes a surface normal image derived from the initial depth image as input. The set of samples with high

matching scores are used to propose a candidate grasp. For each database match in $\mathcal{E}$, the output of the VD-NOC encoder c_e and the geometry feature encoding f_e as well as f_i from the input are passed to the decoder of DGCM-Net to reconstruct the VD-NOC values $V_i \in \mathbb{R}^{W \times H \times 3}$. This represents a dense mapping between the pixels of the sample and the input and thus a transformation of the points in the 3D coordinates can be computed. Each experience has an associated grasp pose, therefore, the transformation between the images is applied to transform the experience grasp to the target object. Sensitivity to the difference in geometry between the input and sample is reduced by confining the alignment to the region around the grasp pose. The region of interest (ROI) on the sample $\mathcal{R}_e$ is determined from the overlap of the gripper with the 3D coordinates of the segmented object in the depth image. The corresponding ROI on the target object $\mathcal{R}_i$ is derived through the matches between the VD-NOC values. The ROIs are aligned by finding the optimal rotation and translation. The outcome is a proposal for a full 6D grasp pose for the target object.

Incremental learning operates by executing a selected grasp proposal and updating the database online with a new exemplar if the grasp is successful. Specifically, the depth image, surface normal image, VD-NOC values, ROI and transformation of the grasp pose are stored. Unsuccessful grasp attempts do not provide any information for replicating past experience, therefore, no data is stored for failed grasps. As more successful experience is accumulated, the likelihood of finding a nearby match for a new input increases. The method is not restricted to only finding samples of exact object instances, but can match to new or unseen objects if they have geometry resembling those from experience.

6.2.2 Dense Geometrical Correspondence Matching

View-dependent Normalized Object Coordinate Space Predicting dense correspondences between two depth images (i.e., the depth image of the object to grasp and an experience in the database) is done by predicting a variant of NOC values. The NOC values used in Chapter 3 represent the correspondence between the target object and another one in the target object's local frame. Typically this has been applied for object pose estimation where the target object has a reference model and the other object is an observation of the reference model in a scene.

To apply the same methodology without object models, we introduce the view-dependent normalized object coordinate values. The depth images for a reference and an input are converted to surface normal images. The VD-NOC values for the input V_i are computed using the 3D coordinates of each pixel I_i^{3D} from the input segmentation mask in the camera coordinate frame. normalization is performed by setting the origin to the mean coordinate between the maximum and minimum values of I_i^{3D} according to,

$$V_i = \frac{I_i^{3D} - \bar{I}_i^{3D}}{\max |I_i^{3D} - \bar{I}_i^{3D}|}, \text{ where } \bar{I}_i^{3D} = \frac{\max(I_i^{3D}) + \min(I_i^{3D})}{2}. \tag{6.1}$$

normalization is performed separately for each dimension resulting in different normalization factors for each axis. The direction of the z-axis is flipped to produce positive

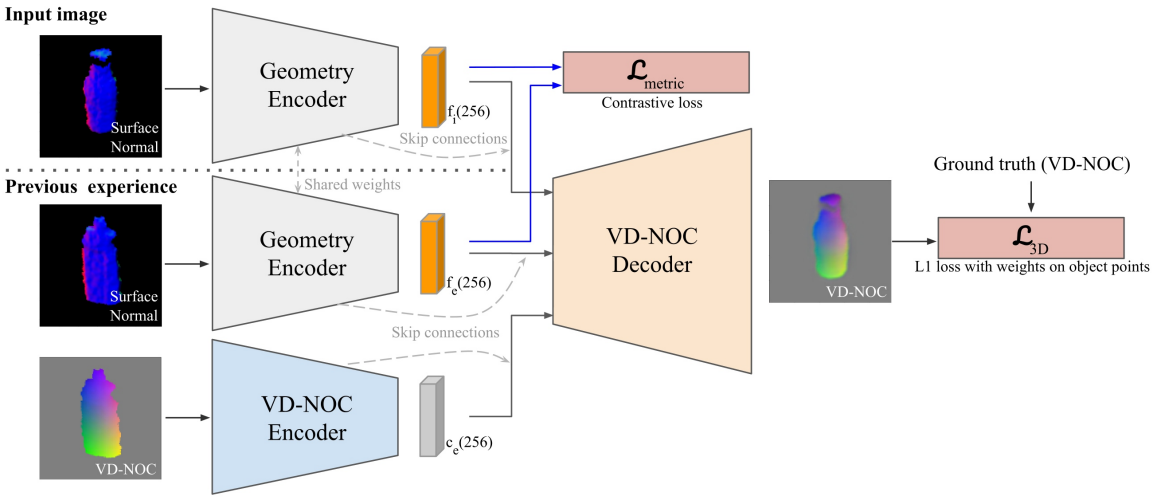

Figure 6.3: Overview of the DGCM-Net architecture and training objectives.

values for points that are nearer.

For grasping, the VD-NOC values are used to estimate the similarity between points on the target object in the input image and the points on the object in the experience database. A smaller distance between values in the VD-NOC values represents closer geometrical correspondence. These can be used to estimate the transformation of a set of points in the grasp pose ROI in order to transfer the grasp experience to the target object.

DGCM-Net Architecture An overview of the dense geometric correspondence network is shown in Figure 6.3. DGCM-Net consists of a geometry encoder, VD-NOC encoder and VD-NOC decoder. The purpose of the geometry encoder is to learn a representation that places images with similar geometry closer in feature space than images with dissimilar geometry. The purpose of the VD-NOC encoder-decoder is to reconstruct the VD-NOC values between a pair of images.

The input to the geometry encoder is a cropped surface normal image derived from the input depth image and segmentation mask. The cropped image is created from a 2D bounding box that is centered at the 2D projected point of the segmentation mask's centroid. The height and width of the image are adjusted to correspond to 30cm spatial size in 3D space. The cropped image is then resized to 128x128 pixels. The first three stages of the Resnet-50 [118] architecture is employed and initialized with the pre-trained weights using the ImageNet dataset [38]. The output of the third block is passed to three convolution layers, kernel sizes = [3, 3, 2] and filter sizes = [256, 256, 128] with strides 2 for all, and two fully connected layers with 256 outputs. The *LeakyReLU* activation is applied to every layer output except the last layer that uses the tanh as an activation to transform feature descriptors to 256 dimensions.

The input to the VD-NOC encoder is a cropped VD-NOC image. The input is passed to five convolution layers, kernel sizes = [5, 3, 3, 3, 3] and filter sizes = [128, 256, 256, 256, 256] with strides 2 for all, and one fully connected layer with 256 outputs. The

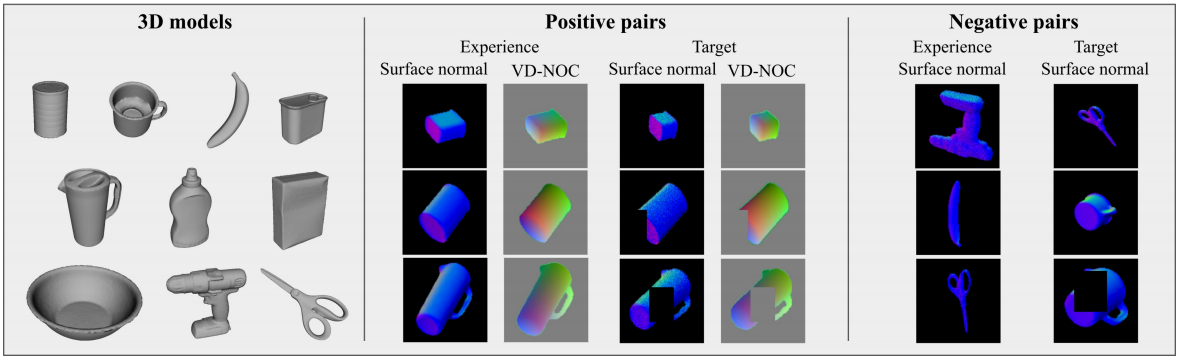

Figure 6.4: 3D models used for training and examples of positive and negative training pairs. Left: Object models from the YCB dataset used to train DGCM-Net. Middle: Examples of positive pairs with online augmentation, Gaussian noise, and partial occlusion. Right: Examples of negative pairs. The pairs are used to train feature vectors to have smaller distances for similar geometries.

Stage	Data generation		Online augmentation	
Parameters	Scale	Dist. to camera	Frac. of Occlusion	Gaussian noise
Range	$\mathcal{U}(0.8, 1.5)$	$\mathcal{U}(1\text{m}, 1.7\text{m})$	$\mathcal{U}(0.0, 0.25)$	$\mathcal{N}(\mu{=}0,\sigma{=}0.01)$

Table 6.1: Overview of the parameters used to generate the training data and for the online data augmentation.

activation of each layer is the same for the geometry encoder. The VD-NOC decoder reconstructs the VD-NOC values for the input image with respect to the camera frame. The input to the decoder is the concatenated features from both geometry encodings of the images and the output of the VD-NOC encoder for the reference. Skip connections [115] are added by concatenating one-half of the output channels of each intermediate layer of the encoders with corresponding layers in the decoder. This helps to predict fine details in local areas. The decoder ends with a fully connected layer with 2048 outputs followed by five blocks of deconvolution and convolution layers. The output of the last convolution layer is the same size as the input image with three channels that represent the x, y and z components of the VD-NOC values.

Training Objective DGCM-Net has two tasks and therefore consists of two objectives in the training process. The first is the metric learning of feature descriptors to perform matching and the second is for reconstructing the VD-NOC values of an input image. For metric learning, the contrastive loss [146] is employed to minimize the Euclidean distance between features of similar geometry (a positive pair) while increasing the distance for a pair of different geometry (a negative pair) as formulated by,

$$\mathcal{L}_{\text{metric}} = \frac{1}{N}\sum_{i=1}^{N}(1-\omega_i)d_i^2 + \omega_i \max(10-d_i,0)^2, \tag{6.2}$$

where ω denotes labels for pairs that are set to 0 for positive pairs and 1 for negative pairs. d denotes the Euclidean distance between encoded feature vectors ($f_i, f_e \in \mathbb{R}^{256}$) of the target and experience images from the geometry encoder. The loss is computed for a mini-batch that consists of N pairs of training images.

For the reconstruction of VD-NOC values, the standard L1 loss is applied for each pixel p. Since background pixels are masked out, their values are easy to predict. Hence, the loss values for pixels on the object masks $M_i \in \mathbb{R}^{W \times H}$ are weighted by a factor of 3 to more precisely predict the values of pixels in the object masks as we performed in Chapter 3. The reconstruction loss is thus given by,

$$\mathcal{L}_{\text{3D}} = \frac{1}{N \times W \times H} \sum_{i=1}^{N} \left[3 \sum_{p \in M_i} ||V_{\text{i}}^p - V_{\text{gt}}^p||_1 + \sum_{p \notin M_i} ||V_{\text{i}}^p - V_{\text{gt}}^p||_1 \right]. \tag{6.3}$$

The reconstruction loss is computed only if the pair of samples is positive. Finally, the objective of the training is the weighted sum of two loss functions,

$$\mathcal{L} = \mathcal{L}_{\text{metric}} + \lambda \mathcal{L}_{\text{3D}}, \tag{6.4}$$

where λ is a weight balancing the two objectives. We set λ to 1 in our experiments.

Training using Synthetic Images Synthetic depth images are created to train the network. 3D models are sampled such that no two models are the same even after a scale change[1]. Objects are selected from the YCB object and model set [58] and listed in Figure 6.4 (left). Depth images are rendered in OpenGL[2] for each object model by uniformly sampling a pose and randomly selecting scale factors for each axis. Five scenes are rendered with different scales for each sampled object pose. To avoid ambiguous views of symmetric objects, view angles are limited between 0 and 45 degrees on each axis. For cylindrical objects, no variation around the rotational axis is applied. Parameters used in the generation process are summarised in Table 6.1. The result for every training sample is a depth image, VD-NOC image, annotated pose, annotated scale factors for each dimension and a look-up table of visible vertices. Approximately 166k images are created and used for training.

Metric learning requires positive and negative pairs. Positive pairs are obtained from samples of the same object in different poses when a pair of images share more than half of the visible vertices. Negative pairs are obtained from different objects or different poses of the same object when images share less than half of the visible vertices. Examples of training samples of both types are given in Figure 6.4 (middle and right). For positive pairs, the target VD-NOC values (i.e., the ground-truth value) is computed using the relative pose of the object, which is known for the training samples. Thus, the VD-NOC values that are defined in the camera frame of the first element of the pair are transformed to the camera frame of the second element. For our grasping framework, this amounts to transforming the VD-NOC values from the object in the input image to the object in the experience database.

[1]The set of 3D models only contains one box because any other box can be constructed just by manipulating the scale in the different dimensions

[2]https://www.opengl.org

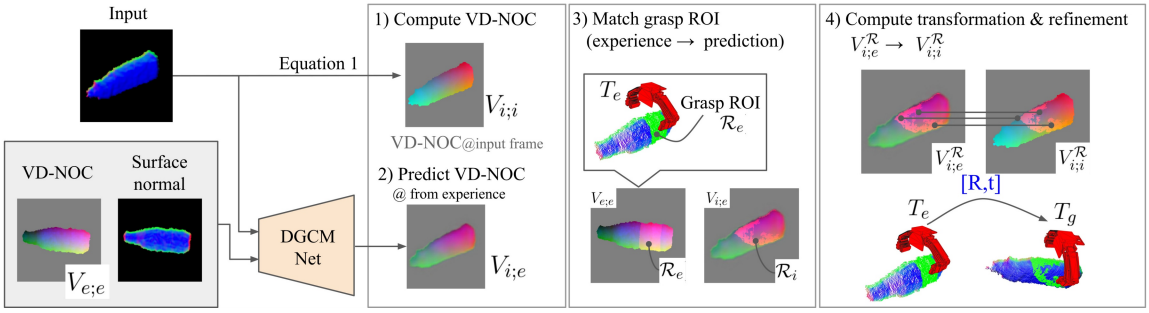

Figure 6.5: Overview of the process for generating grasp proposals from the nearest neighbour experience.

Further augmentation is applied to the image samples to improve robustness against occlusion and noise. Occlusion is simulated by setting a partial area in the surface normal image and the corresponding entries in the VD-NOC values to zero (i.e., the value for the background). This enables the network to learn features that still return good matches between an input and samples in the database even when one is occluded. Gaussian noise is also applied to both images to cope with the expected noise from real sensors. Figure 6.4 (middle and right) presents examples after applying the augmentation. More details about the parameters used for augmentation are provided in Table 6.1.

We train the network for 35 epochs using the ADAM optimiser [117] while assigning 25 positive pairs and 25 negative pairs for each batch. The learning rate is initially set to 0.0001 and divided by a factor of 10 every 5 epochs. After training the network once, the weights are fixed for all experiments in this chapter without any fine-tuning.

6.2.3 Generating Grasp Proposals

The overview in Figure 6.5 shows the process of retrieving and generating grasps given an input depth image. First, the surface normal image of the input is encoded to a feature map f_i by the geometry encoder. This is compared to all feature maps $\{f_e\}\ \forall\ e \in \mathcal{E}$ to find a set of nearest neighbours $\mathcal{N}_i \subset \mathcal{E}$. The stored VD-NOC values V_e of a sample $e \in \mathcal{N}_i$ is loaded to compute the VD-NOC feature map c_e. Given c_e, f_e and f_i, the decoder predicts the VD-NOC values of the input depth image $V_{i;e}$ (VD-NOC values of D_i in the frame of D_e) as shown in Figure 6.5. The ROI of the experience $\mathcal{R}_e$ is used to compute the corresponding ROI for the input $\mathcal{R}_i = \{p \in V_i : \min_e |p - p_e| < \theta_c\ \forall\ p_e \in \mathcal{R}_e\}$, which is the subset of points whose distance to the nearest points in $\mathcal{R}_e$ is below a threshold θ_c. The predicted VD-NOC ROI points is denoted $V^{\mathcal{R}}_{i;e}$ and are defined in the camera frame of D_e. Each pixel of $V^{\mathcal{R}}_i$ forms a 3D-3D correspondence from the VD-NOC values $V^{\mathcal{R}}_{i;e}$ and $V^{\mathcal{R}}_{i;i}$ that are defined in D_e and D_i. Thus, an initial rotation from the camera frame of the experience to the camera frame of the input is derived by aligning the ROI VD-NOC images. The grasp pose T_e is then aligned to D_i by computing the rotation that minimises the summation

of distances of the correspondences given by,

$$R_{\text{init}},t_{\text{init}} = \arg\min_{R,t} \sum_{\mathcal{R}_i} ||(RV^{\mathcal{R}}_{i;e} + t) - V^{\mathcal{R}}_{i;i}||_2. \tag{6.5}$$

The optimised solution for Equation (6.5) is obtained using singular value decomposition. The unit of t_{init} does not correspond to the scale of the 3D space because the VD-NOC values are normalized. Therefore, the translation t_{init} is separately computed using the difference between the mean coordinates between the maximum and minimum values of the ROI points as was applied in Equation (6.1). The computed rotation and translation are used to transform all ROI points of the experience $\mathcal{R}_e$ to the scene and the alignment is refined by applying the iterative closest point algorithm. The grasp poses in the experience T_e is transformed to create the grasp proposal T_g by applying the same refined transformation. Finally, the gripper position is moved to a fixed distance from the object surface by translating along the approach direction with respect to the closest point in the input.

Each match in the database has an associated score in the range (0,1] that represents that similarity of the depth image to the input, which is used as a pseudo-measure for the quality of the grasp. This score is computed as,

$$S(i,e) = e^{-||f_i - f_e||_2}. \tag{6.6}$$

The final output is a set of grasps $\mathcal{G} = \{(T_g, s_g)\}$ where each grasp proposal is composed of a transformation of the gripper into the scene T_g as well as a score value s_g using Equation (6.6).

6.3 Evaluation

This section analyzes the grasp proposal method with a hand-annotated dataset. Real-world grasping experiments with a mobile manipulator presents the grasp performance of our framework using multiple object classes in different scenarios. Code for DGCM-Net is publicly available at `https://rgit.acin.tuwien.ac.at/v4r/dgcm-net`.

6.3.1 Offline Experiments

Offline experiments are performed to first investigate the quality of grasp pose prediction with respect to the size of the grasp experience database and secondly to evaluate the ability to transfer grasps between observations of objects within the same and to different classes. The threshold for matching ROI correspondences θ_c is set to 0.3 for all experiments. This value produces a reasonable separation of ROI areas and other parts of objects. Every stored experience is duplicated with in-plane rotations at angles between -90 and +90 degrees with a step size of 45 degrees. This enables grasp transfer to objects in new poses.

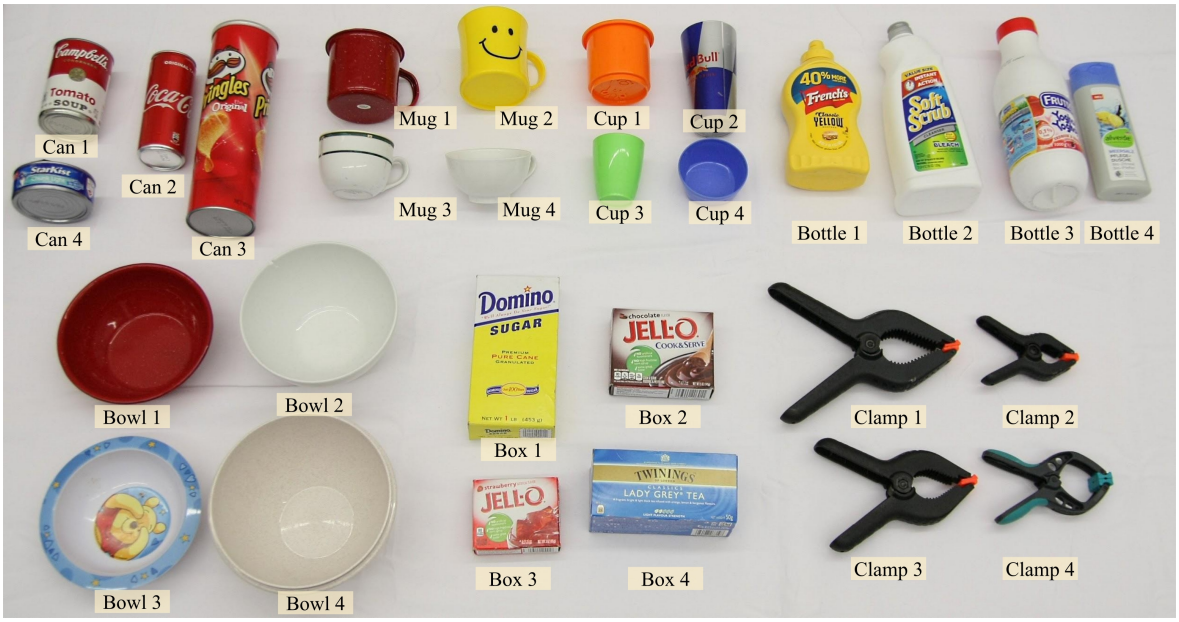

Figure 6.6: Objects in the dataset used for the offline experiments and to supplement past experience during the online experiment.

Dataset A dataset is created to evaluate the quality of grasp prediction comprising depth images of the objects shown in Figure 6.6. These objects are organised into seven classes: `can`, `mug`, `cup`, `bottle`, `bowl`, `box` and `clamp`. Four instances are used for each class and a number of these instances are from the YCB object dataset [58], while other instances are objects commonly found in homes. The dataset is available at `https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/lfed-6d-dataset/`.

Recordings are made by placing each object on a small table and capturing a depth image with an ASUS XTion Pro Live RGB-D camera. Each object is placed in various poses and locations, and the camera is moved between two different heights. The object is segmented in each depth image by detecting the table surface with RANSAC and selecting all points that remain above the table plane. The dataset does not require ground-truth segmentations, but instead should be segmented by the same method that extracts the masks for the input images in order for the entries in the experience database to best resemble the inputs.

Grasp poses for a parallel-jaw gripper are manually annotated in the depth images. Each depth image consists of possibly multiple grasp annotations according to their direction, for example, from the top or from the side. The full dataset used for testing consists of depth images, segmentation masks and grasp poses for 28 objects.

Measuring Grasp Pose Quality Reporting quantitative statistics requires the quality of the estimated grasp poses to be measured. It is possible to execute physics simulation and to check for grasp success, however, to isolate the grasp prediction itself, we measure the difference in grasp pose for an input with respect to the annotated pose. The experiments are simplified by selecting only grasp annotations on the top of the objects

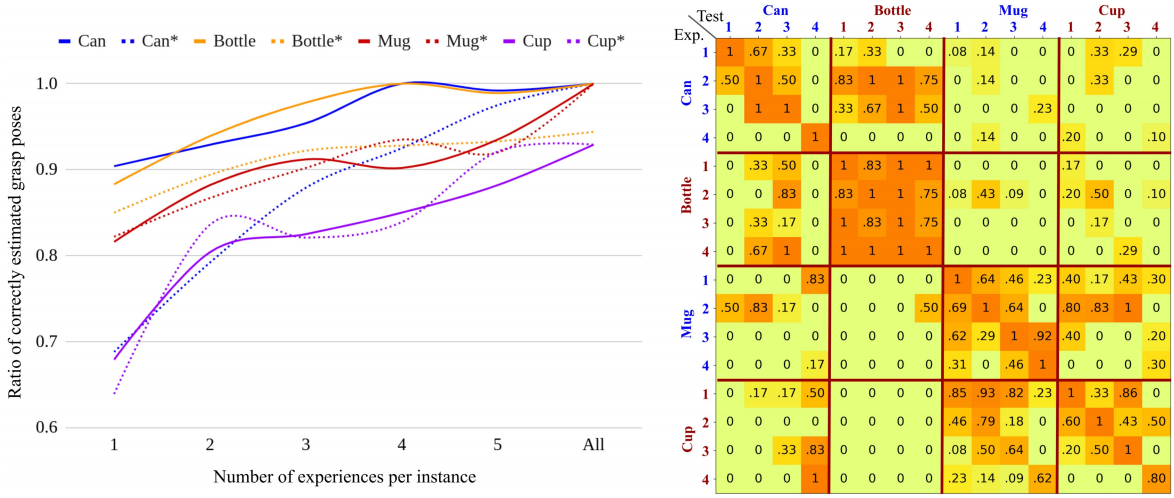

Figure 6.7: Results of the offline experiments. Left: Ratio of accurately estimated grasp pose with an increasing number of experiences per instance. Solid lines show results when class specific experience is used for each test class. Dotted lines show results when experience from all classes is used. Right: Ratio of accurately estimated grasp poses using experience from each instance in all classes.

when they are placed in their upright canonical pose. Even though the grasp proposals are limited to top-down, multiple poses are available, especially for objects that are symmetric or are elongated in the x or y dimension such that translations of a top-down grasp are equivalent. Consistency of top-down grasp poses is ensured by testing with the subset of classes `can`, `mug`, `cup` and `bottle`.

A grasp pose is regarded as correct when the translation error is less than 5cm and the rotational error around the x- and y-axes is less than 15 degrees. A rotation error around the z-axis in the gripper frame, which is parallel to the rotational axis of an object, is ignored since it should be a successful grasp regardless of the rotation with respect to this axis.

Increasing Experience The left plot of Figure 6.7 shows the ratio of correctly estimated grasp poses with increasing numbers of experience per instance. Solid lines show results when only the experience for the relevant class is considered and the dotted lines show the results when experience for all classes is considered. For each configuration (class and number of experience per instance), we perform ten iterations using randomly selected samples in the iteration. The results with class-specific experience demonstrate that the grasp poses are often correctly estimated even if only one experience is included per instance. For the `can` and `bottle` classes, the correct estimation is approximately 90%, while the worst performing class, `cup`, achieves 68%. However, as the number of experiences increases per instance in each class, the grasp pose estimation improves.

The dotted lines show the variation in performance when including other classes for experience, which reflects more practical scenarios in the real world. Except for the `mug` class, the performance slightly drops because the retrieval of experiences from different

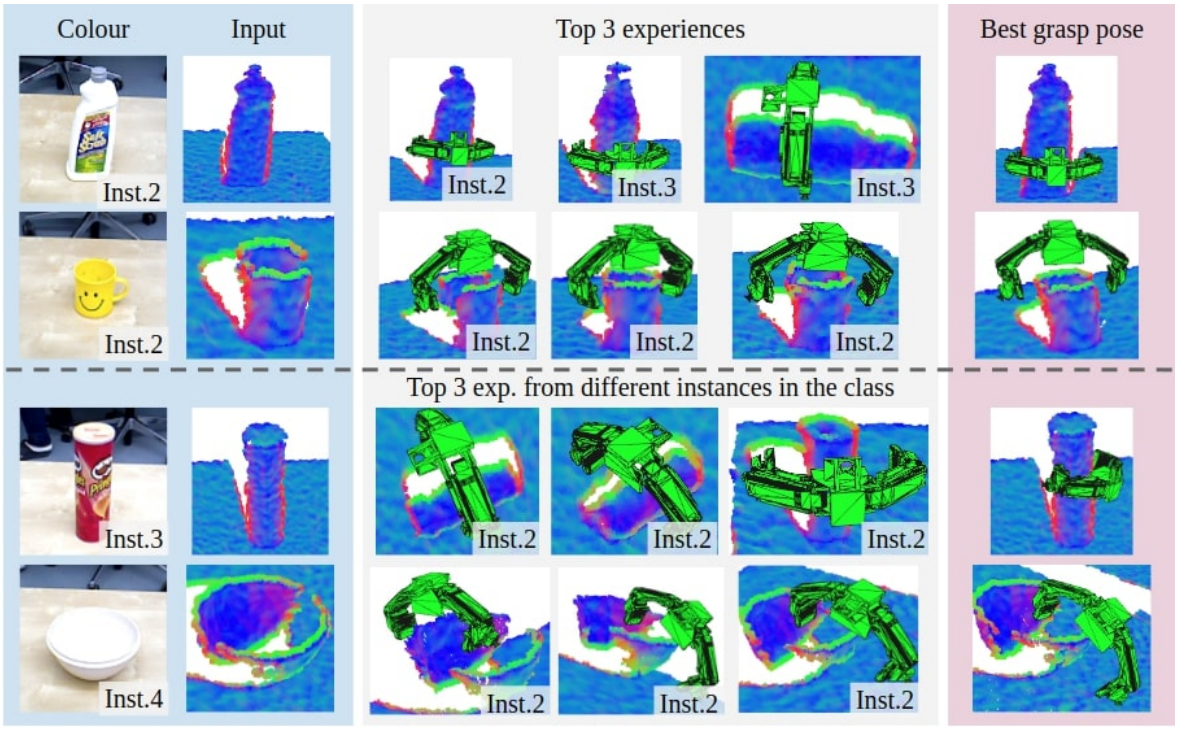

Figure 6.8: Examples of the three nearest experiences and estimated grasp poses for example input images. Top two rows show examples when the same instance is included in the experience database. Bottom two rows show examples when the instance is not in the experience database.

classes can cause inaccurate prediction of VD-NOC values. However, the accuracy still achieves more than 79% when two experiences are included per instance. This implies that the feature space encoded by the geometry encoder is sufficient to distinguish different geometrical shapes. The performance gap for the `can` class, which has the most simple shape, is comparably larger than for the other classes. This is because the mapping from more complex to simpler geometries produces inaccurate estimations by transferring detailed shapes into simpler geometries. We discuss more detail regarding this relationship between classes in the following section.

Figure 6.8 shows qualitative results. The figure shows the three nearest experiences and the best transformed pose for different instances from different classes. The first and second rows are obtained when the experience database contains samples from all classes and include the exact instance in the test image. The third and fourth rows are obtained after excluding the instance in the test image so that different instances in the same class must be retrieved to generate grasp proposals. The results reveal that the grasp poses are transformed to similar locations and directions even if the object poses from experience are different (see the grasp poses for the `can` in the third row).

Transfer Between Instances and Classes These experiments show that experience can be transferred between instances and classes. The experiments are conducted by using all experience from a single object instance while testing on different instances.

The evaluation metric and subset of target grasp poses are the same as in the previous experiment. The matrix on the right of Figure 6.7 shows that the experience of instances transfers well to other instances within the same class. Furthermore, experience also transfers beyond the class. For example, many good grasps are found for `bottle` instances when provided by experience from `can` instances (both types of objects have a closed surface on the top) and that instances of the `cup` class provide sufficient experience for grasping `mug` instances (both types of objects have no surface on the top). However, the results show that it is difficult to transfer the experience of a class to an instance in the same class when the geometry and scale of the instance are different from the other instances in the class (e.g., `Can4` and `Cup4`). Thus, better grasp poses are obtained when the experience is obtained from geometrically similar objects regardless of explicit classes. It is also observed that grasps for simpler geometries (e.g., from `can` to `bottle` and `cup` to `mug`) are more accurately transferred, while the other direction from complex geometry to simpler geometry is more difficult. This is because the network tries to predict VD-NOC values of detailed shapes of objects in the experience set, such as handles of `mug` instances, which potentially causes errors by predicting corresponding points even if the shapes are missing in the new object.

6.3.2 Robot Experiments

This section presents results of real-world grasping experiments with a mobile manipulator. First, we describe the hardware set up used for the experiments. Second, we compare our method to baseline approaches. Third, we evaluate the full pipeline of online incremental grasp learning. Finally, we demonstrate the extension to semantic grasp learning.

Experimental Details The robot experiments are performed with the Toyota Human Support Robot [147]. The platform consists of a 4-DOF arm but motions are computed including the omni-directional base, which effectively offers seven degrees of freedom. Motions for grasp execution are planned using MoveIt [148][3]. The end-effector is a parallel-jaw gripper and grasp success is measured by checking the distance between the tips of the gripper after the target object is lifted. If the distance is non-zero, then the grasp is declared successful, otherwise, it is a failure. Depth images are captured with the onboard ASUS XTion Pro Live RGB-D sensor positioned on the head of the robot.

For all grasping experiments, individual objects are placed on a small table that has a height of 45cm. The robot is approximately positioned 30cm from the table (edge of the robot base to edge of the table). The head of the robot is tilted such that the camera faces the center of the table. The torso of the robot is raised to give an approximate distance from the camera to an object of 1m to suit the optimal range of the sensor. Objects are segmented from the table with the same procedure for generating segmentation masks for the dataset.

All code is written in C++ and Python, and is running on the robot in Ubuntu 16.04. ROS [149][4] is used for process communication. DGCM-Net is implemented in

[3] http://moveit.ros.org
[4] https://www.ros.org/

Tensorflow and is running on an external PC with an NVIDIA GTX 1050 Ti.

Comparison to Baselines Experiments are conducted to measure the grasp performance of our framework. For comparison, experiments are also performed with a number of baselines. The full set of methods is as follows:

- HAF: The approach introduced in [86], where height accumulated features are extracted from point clouds to derive grasp-relevant structure. The features are computed on different regions of the input and a support vector machine is trained to predict the quality of the grasp for each feature. Both top-down and forward-facing grasps are enabled, and the output with the highest score is executed. We use the original code provided[5].

- GPD: The approach introduced in [96], where grasps are sampled using the surface geometry of the input point cloud. Grasp success for each sample is classified using a convolutional neural network (CNN). This takes as input three images: an averaged height map of occupied points, averaged height map of the unobserved region and averaged surface normals. Given this input, the CNN generates a score value. Finally, grasps are clustered and the highest scoring cluster is selected. We use the original code provided[6] and the full 15 channel version.

- DGCM-Net: The grasp proposals from DGCM-Net using pre-collected experience for the relevant object classes in the experiments. Similar to GPD, the set of proposals from DGCM-Net are clustered and the highest scoring cluster is executed. Clustering is performed by grouping all grasps within 5cm translation and 15 degrees rotation. The grasp of the cluster is the mean pose of the proposals that make up the cluster. The cluster with the highest summed score is executed. The number of nearest neighbors to be retrieved by DGCM-Net is set to 10.

- GPD + DGCM-Net: Grasps are proposed using GPD and the scores are modified by the predictions from DGCM-Net. First, the grasps from the GPD method are computed and the scores are normalized to the range $[0, 1]$. Then DGCM-Net is run on the same input and for each GPD candidate, we find all DGCM-Net proposals within 5cm translation and 15 degrees rotation. The experience score is the average of the scores for all DGCM-Net grasps deemed to be nearby. The final score for each GPD candidate is the average of the normalized GPD score and the summed experience score. The grasp with the highest final score is executed.

Many robotic grasping approaches are successful for the bin-picking task, e.g., [95]. However, these are focused on 2D grasping and therefore expect a top-down view of the scene and only generate a grasp parallel to the camera axis. This is unsuitable for our robot platform due to the position of the arm on the front of the body that occludes the scene when facing the camera directly downwards. Additionally, bin-picking methods are at a disadvantage because they only generate grasps for a single approach direction. It is left to future work to extend the evaluation to this type of scenario.

[5] https://github.com/davidfischinger/haf_grasping
[6] https://github.com/atenpas/gpd

Figure 6.9: Test objects used for the online grasping experiments. Three instances in seven classes are used.

	HAF	GPD	**DGCM-Net**	**GPD + DGCM-Net**
Box	0.87	0.80	0.67	1.00
Can	0.87	0.67	0.93	0.73
Bottle	0.87	0.93	0.93	0.93
Mug	0.80	0.80	0.87	1.00
Cup	0.80	0.80	0.73	1.00
Bowl	0.40	0.87	0.80	0.87
Clamp	0.40	0.60	0.60	0.67
Average	0.71	0.79	0.79	0.89

Table 6.2: Grasp success rate of our framework and baseline methods for different target object classes. The bottom row shows the average for all classes.

The experiments are performed for objects from the classes `box`, `can`, `bottle`, `mug`, `cup`, `bowl` and `clamp`. Three instances are chosen per class and five poses are considered per instance. The five poses for each instance are kept constant for the experiments with each grasping method. The objects selected for the experiments are shown in Figure 6.9. These include one object from the YCB dataset for each class from the objects used in Section 6.3.1, in particular, the `sugar box`, `spam can`, `mustard bottle`, `red mug`, `orange cup`, `red bowl` and `XL clamp`. The other two objects for each class are a mixture of YCB objects and common objects found in homes.

The experience used for our method is an extension of the database from Section 6.3.1 that includes instances from the additional classes of `box`, `bowl` and `clamp`. Since we are interested in observing the grasp performance for unseen objects, the YCB objects selected as target objects are removed from the experience database.

Performance is measured by grasp success rate, which is the number of successful grasps divided by the total number of attempts. Table 6.2 reports the average grasp

success rate for each class and the average for all classes (bottom row). The results show that our method performs equivalently to GPD and that both methods outperform HAF (+8%). However, combining experience with GPD achieves a much higher grasp success rate overall. In comparison to the original GPD method, this is an increase of 10%.

For most classes, the combined approach performs either the same as the best performing individual method or better. The only exception is the `can` class, which has 20% lower grasp success rate than our direct method. Our observation during the experiments is that GPD often proposed grasps on and orthogonal to the rim of the `can` objects, which resulted in failures. This exposes the flaw that if the initial candidates are unfavorable, the combined approach cannot improve. For these objects, when grasp experience on the top of the `can` are stored, the grasps on the rim are still similar in position and orientation to warrant their selection.

Surprisingly, the `box` class is the most difficult for our approach despite having easy geometry to compute a grasp as shown by the high success rate of HAF. This can be explained by the fact that all objects of this type can be represented by a single `box` by changing the scale in the different dimensions. Thus, the network has to decide whether to regard a new instance as a scaled version of an experience or as a transformed (i.e., rotated) instance. The ambiguity causes noisy predictions of VD-NOC values. Furthermore, since grasp proposals are transformed from previous experiences and are ideally in a similar grasp location, a scale change may cause the prediction to exceed the range of the gripper, resulting in its rejection due to the collision.

Incremental Learning This set of experiments demonstrate the incremental learning framework. The test objects chosen are the `grey clamp` from our dataset, the `plastic drill` from the YCB object dataset and a `gaming controller`. Past experience is stored in the database, however, not for the classes of the test objects. Therefore, experience from the `clamp` class is removed. Since good grasps may not be generated for the unseen objects, GPD is used in the beginning until DGCM-Net makes reasonable predictions. A threshold of 0.9 is set as the minimum feature distance that must be achieved by the output of DGCM-Net, otherwise, the best grasp from GPD is executed. Typically it only takes one or two successful attempts for the system to switch from GPD to DGCM-Net. The objects are placed randomly on the table at the beginning of each experiment and the system runs autonomously, with the robot grasping the object from where it lies after a successful or failed attempt. The object is only handled by a person if it is unintentionally moved near the edge of the table and presents a risk of falling. After successful grasps, objects are placed on the table by the robot and receive a slight variation in pose; failed grasps typically cause considerable object movement. After any grasp attempt, the robot base returns to the start position and the localization inaccuracy generates further viewpoint variation.

Figures 6.10, 6.11 and 6.12 show the evolution of grasp success for the three objects. In these figures, the first row shows the surface normal image of the input and the second to fourth rows show the nearest three matches in the database. Below this, we plot the minimum feature distance of the nearest neighbor. Lastly, we show the best grasp proposal from DGCM-Net and the actual gripper position during the grasp

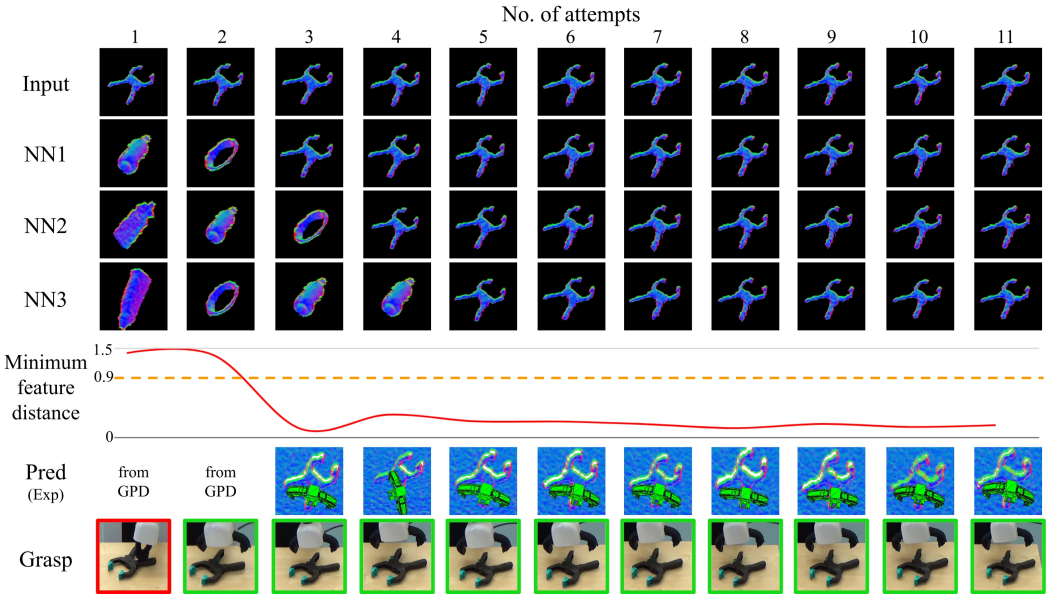

Figure 6.10: Summary of results for the incremental learning experiment with the `grey clamp`.

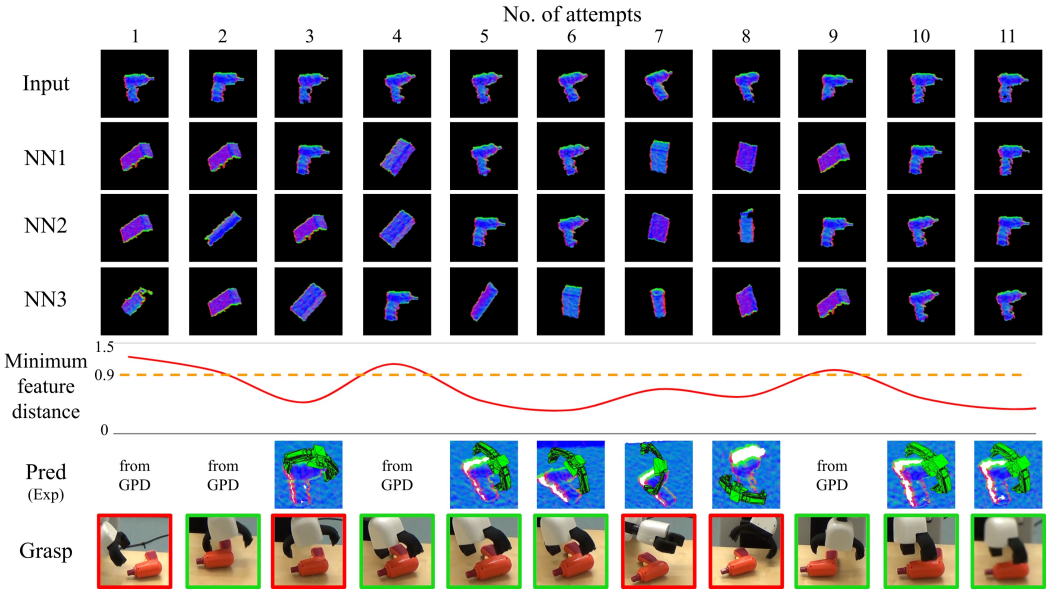

Figure 6.11: Summary of results for the incremental learning experiment with the YCB `plastic drill`.

captured by an external camera (red border indicates failure and green border indicates success).

From these experiments, we make two observations. Firstly, after the first successful attempt from GPD is recorded, the robot typically continues to grasp the target objects successfully. The predicted grasps for each attempt confirms that our method reliably predicts the same successful experience so long as the object and its shape is correctly

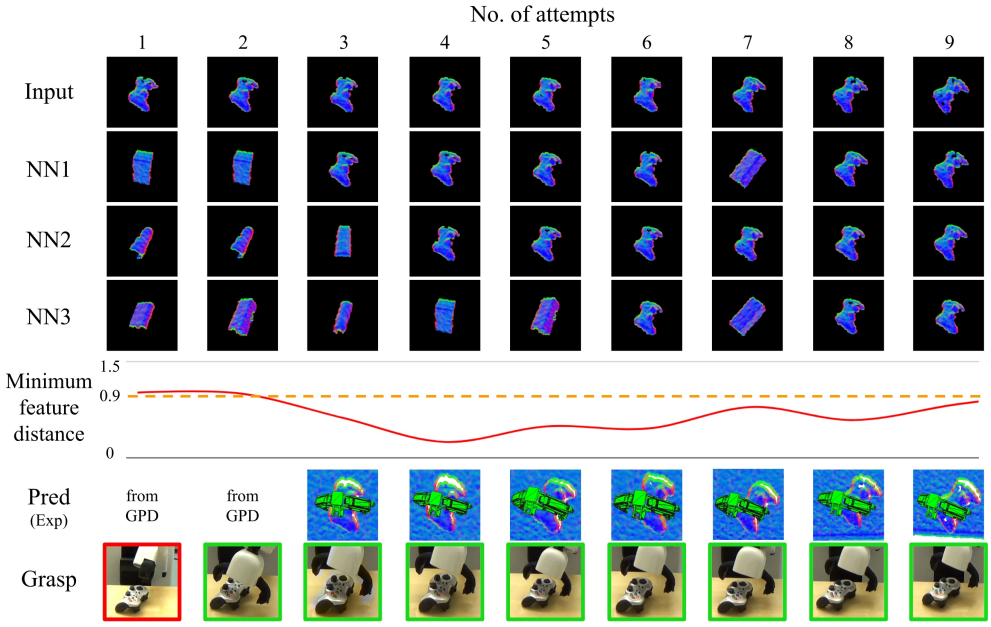

Figure 6.12: Summary of results for the incremental learning experiment with the `gaming controller`.

identified. The second observation is that the minimum feature distance drops below the threshold after as many as one samples is in the database. This is most apparent for the `grey clamp` in which the minimum feature distance is very small for all subsequent trials.

The grasping for the `plastic drill` and `gaming controller` are less reliable than for the `grey clamp`. For the `plastic drill`, the system still exhibits some failures even after accumulating experience. In both cases, the nearest feature distance does not converge to the same low value as was observed for the `grey clamp`. The reason is that the `plastic drill` and `gaming controller` have less distinct shapes and therefore are more difficult to match. This is especially noticeable when the objects have rotated. The objects are often confused as an instance from the `box` class and the grasps for the matching object is executed. Fortunately for the `gaming controller`, the execution still results in success. However, for the `plastic drill`, the predicted grasp is not very good and the grasp fails.

Semantic Grasping A final set of experiments demonstrate the extension of our method to generate semantic grasps for instances belonging to the same functional class. For these experiments, we investigate grasps on the handles of `mug` and `drill` objects. The experience is hand-annotated for test exemplars of instances in the same classes. The semantic grasps can be automatically annotated by affordance detection methods [150] in real applications.

Example grasp proposals generated by our method as well as GPD and HAF for a `mug` and `drill` in various poses are shown in Figure 6.13. Our method reliably generates grasp poses on the relevant object part, while both GPD and HAF fail to

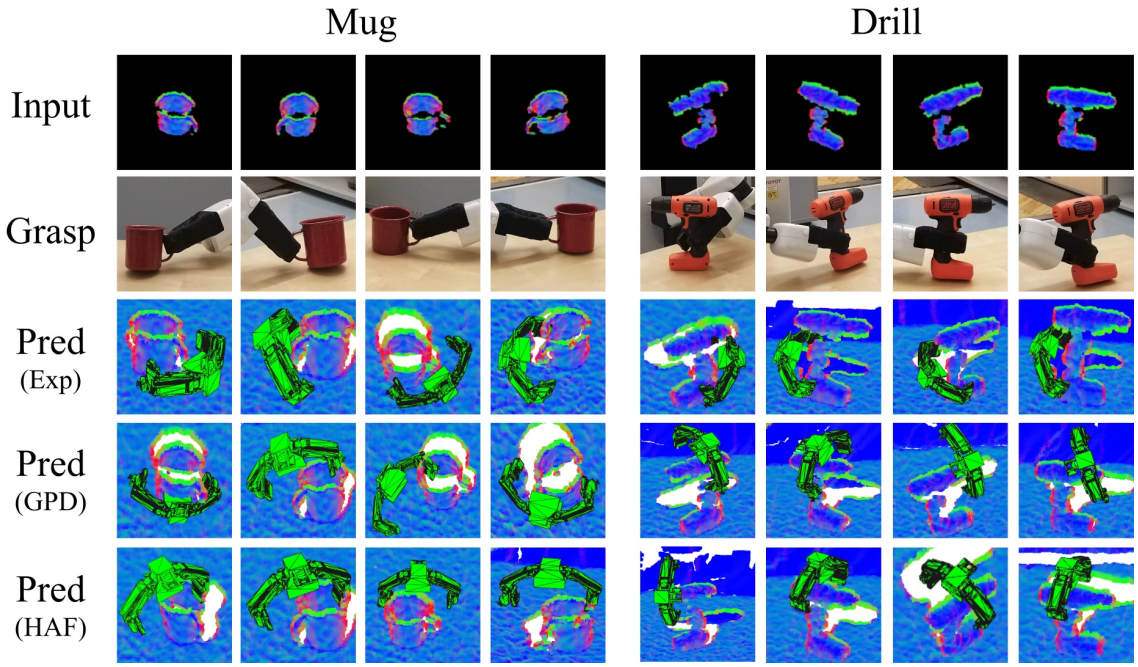

Figure 6.13: Examples of semantic grasping for the YCB red mug and the YCB orange drill in four different poses. First row: Surface normal image. Second row: Grasp pose executed by the robot. Third row: Predicted grasp from DGCM-Net. Fourth row: Predicted grasp from GPD. Fifth row: Predicted grasp from HAF.

do so. Although the grasps from GPD and HAF may result in success, they do not support the functional use of the object. For the `mug`, it is understandable that the handle is not grasped because the quality of the depth data on that part of the object is very poor and does not characterize a stable grasp. Our method, on the other hand, does not only rely on the local structure to estimate the grasp. So long as there is some cue about the handle, as is present in these selected examples, the handle grasp is generated. The `drill` offers more depth data on the handle, but the baselines still prefer to grasp the head. Even though HAF is executed to find both top and front grasps, a top grasp or a front grasp on the head is preferred.

6.4 Discussion

This chapter introduced an approach for incrementally learning grasps by leveraging past experiences. In our system, every successful grasp is stored in a database and retrieved to guide future grasps. This is accomplished with the dense geometric correspondence network that is trained to predict the similarity between newly acquired input depth images and stored experiences as well as to predict 3D-3D correspondences to transform grasp poses. A descriptive feature space is constructed for the retrieval task using metric learning and correspondences are established by predicting view-dependent normalized

object coordinate values.

Offline studies with a dataset showed that our approach precisely recovers grasps from experiences with the same object and also transfers well to unseen objects from the same or different classes. Furthermore, results showed that more experience leads to more reliable grasp proposals. Hardware experiments with a mobile manipulator showed that our experience-based grasping method performs equally successful as the baselines and integration with the baselines show overall superior performance. Additional experiments demonstrated the full online capability to efficiently learn grasps for unseen objects, often needing only one or two successful grasps to reliably re-grasp the same object. Finally, an extension was demonstrated whereby specific grasps, such as those on handles, can be desired in order to achieve semantically meaningful grasps.

Hightlights

(1) The dense geometrical correspondence matching network to encode object geometry for nearest neighbor retrieval and to densely reconstruct 3D-3D correspondences in order to transfer grasps from stored experiences to unseen objects.
(2) An experience-based 6D grasp learning pipeline that accumulates exemplars to guide grasp selection for the same object or novel unseen objects.
(3) Offline experiments with a new dataset showing the capability of DGCM-Net to transfer grasps to unseen objects as well as to steadily improve over time with increasing accumulation of data.
(4) Online grasping experiments showing that combining the baselines with our experience-based predictions significantly improves the reliability of grasping.
(5) Demonstrations showing the extension of our method for semantic grasping by guiding grasp selection to specific parts of objects

Chapter 7

Conclusion

Typical datasets for object pose estimation have reflected challenging scenarios during test time by capturing images from unconstrained and cluttered scenes in different lighting conditions. However, not only test environments but also training environments are unconstrained in real-world scenarios. 3D models with high-quality textures are common in datasets while it is difficult to obtain 3D models in good quality from the real world without a special device and setup. Capturing real images of objects using cameras is easy but annotating poses is a difficult and time-consuming task. The objects in the captured images are often occluded by other objects. There are objects that cannot be represented by a 3D model due to shape variations of individual instances such as vegetables, fruit, and cups. In our daily lives, we face new objects every day while recognizing them after a few observations instead of thinking and learning the objects for a night. Likewise, poses of new objects can be determined by referencing a few observations instead of training a new recognizer for the new objects, which is useful to transfer successful interaction with an object to a new object when they are similar.

This chapter summarizes how the methods introduced in this thesis tackled the aforementioned challenges and discusses further directions of object pose estimation research.

7.1 Summary

In each chapter, different types of the aforementioned practical scenarios were assumed, and the obstacles that make previous work difficult to apply for the scenarios were tackled, which is the biggest contribution of this thesis. Evaluations using our custom datasets and real robots showed the proposed methods in this thesis successfully reduce the gap between the dataset world and the real world. Furthermore, evaluations on public benchmarks showed the methods also achieve state-of-the-art performance in comparison to previous work.

A novel architecture for 6D pose estimation of objects using RGB images, Pix2Pose, was introduced in Chapter 3. As textures of 3D models are difficult to obtain, the network was trained with only real images using 3D models without texture information. Pix2Pose also tackled two common challenges of object pose estimation: occlusion and

symmetry. The occlusion of objects is handled by predicting coordinates of each pixel regardless of visibility. A novel loss function took a minimum error among possible loss values computed from multiple symmetric poses of an object, which made the training process more robust with higher accuracy. Pix2Pose has shown robust and state-of-the-art performance in different types of datasets from various domains using either 3D models without texture or with texture.

Chapter 4 highlighted the issue that it is difficult to obtain either 3D models with high-quality textures or many real images with pose annotations from various viewpoints, which has been assumed in previous work. A novel method, NOL, was introduced to create training images of an object using a few cluttered images (e.g., less than 20 images) with noisy annotations. A new dataset was collected with a mobile robot from a practical scenario, i.e., the robot drives around a table with multiple objects while looking at the center of the table, and poses of objects are self-annotated by a 3D reconstruction method. Experimental results with this new dataset showed NOL successfully synthesizes training images of objects using a few observations. The generated training images were sufficient to train a 2D detection method and a pose estimation method for an object. Evaluations on public benchmarks also showed training images created by NOL led to state-of-the-art performance in comparison to methods that are trained with 13 times the number of real images and synthetic images using textured 3D models.

The methods introduced in Chapter 5 and 6 tackled more challenging scenarios in applications of robot manipulation. The pose estimators that directly predict poses of a closed-set of objects are not applicable when we have to face new objects frequently as the estimation model has to be re-trained. Instead of re-training the model every time, a novel method, MTTM, is proposed in Chapter 5 to estimate poses of objects by matching the nearest sample from a set of object templates. The method compares features of inputs and templates to predict pixel-wise segmentation of target objects. Since geometries of objects are more important than textures for manipulation, depth images were used as inputs. Evaluations on public benchmark showed the method successfully performs detection, segmentation and pose estimation of objects using a set of templates regardless of their types (synthetic, real). The method enables to recognize a new object by adding sample images of target objects immediately without fine-tuning of the network model. Experiments in a practical scenario show that the method is also applicable to objects with similar shapes (e.g., vegetables and fruit) without exact 3D models.

Chapter 6 further improved MTTM to transfer successful grasp experience of an object to a new object that has a similar geometry. As grasp success relies more on local shapes, the features of two inputs, one from a camera and another from an experience, are compared to predict pixel-wise geometric correspondences. The dense correspondences were used to specify a relevant region for grasping and transfer a grasp pose from the experience to the target object. The network is trained to predict consistent values for an object regardless of scale changes of the object by applying random scaling factors. Thus, locations of local parts in the object are globally consistent for the object in the dense correspondences, which enables the method to estimate grasp poses at the semantically same location (e.g., handles of drills). Evaluations performed on a new

dataset showed that the method precisely estimates grasp poses from experiences with the same object and also successfully transfers to unseen objects when the objects have similar geometries. Experiments using a real robot showed outperforming results when the method is integrated with a baseline that supplements grasp poses for unfamiliar objects. Additional experiments demonstrated that grasping of specific parts, such as handles of drills and mugs, can be guided by including corresponding grasp experience on the specific parts as the method predicts local correspondences at globally consistent locations.

7.2 Outlook

The effort for reducing the gap between dataset-based research and real applications should be continued in future work. As this thesis addressed, one of the major directions is to reduce the effort for training since the quality and amount of training data is more important when a data-driven model is employed. The instance-level pose estimation can be extended to estimate poses of objects in the same category. For both robotics and augmented reality applications, it is also important to develop a method that is computationally efficient so that object poses can be estimated using a mobile device with low power consumption. Precisely estimated poses of surrounding objects play a key role in robot manipulation tasks by guiding robots to plan safe trajectories with successful manipulation of target objects.

7.2.1 Training Data for Pose Estimation

Types of training data vary by domains, scenarios, and applications. The most practical and challenging scenario is when no prior knowledge of a target object is given. Although NOL and the SMOT dataset in Chapter 4 tackled this scenario by reconstructing 3D scenes and self-annotating object poses, the pipeline suffers from geometrical errors of reconstructed 3D models. Therefore, future work should aim to reconstruct precise 3D models from a few observations of objects. Furthermore, some parts of objects, such as the bottom parts of the objects in SMOT, cannot be observed without reconfiguration of a static scene. A possible approach will be expanding data incrementally and associating different sequences. For example, an initial model is trained by partial observations of a target object. The initial model is used to associate new observations from different scenes. The same approach can be applied when textured 3D models are available so that synthetically rendered images train the initial model [151], [152] for associating real images. Using robots is a great option for collecting training data from real environments as the robots can actively reconfigure scenes by moving objects and cameras [153].

7.2.2 Category-level Pose Estimation

It is not difficult for humans to presume a shape of a new instance when the instance belongs to a well-known category (e.g., people assume the entire shape of new mugs, bowls, and plates at a glance as they have prior-knowledge of the categories). Likewise,

it is more efficient to train a model for a class than train a model for each instance in the class so that no further training is required for a new instance in the class. As Chapter 6 introduced, it is possible to predict normalized coordinates of objects even though the shapes of objects are not the same. A similar approach using RGB-D images [154] also shows that poses of objects in a category can be represented with normalized coordinates and final poses are derived by depth information. There are a limited number of classes in public datasets since a huge effort is necessary to align all 3D models of a class in the same canonical pose. Hence, a large scale dataset is required to scale research in this direction.

7.2.3 Computational Efficiency

Robots and mobile devices are often powered by batteries and are not equipped with the high-performance Graphic Process Unit (GPU) that is a key component of CNN-based methods. Thus, improving the computational efficiency is important for both robotic and augmented reality applications. Multiple techniques have been proposed for downsizing CNN models for image recognition tasks [155], [156]. As such, CNN-based pose estimation methods can be optimized in terms of computational efficiency while minimizing performance drops.

7.2.4 Robot Manipulation using Object Poses

When poses of objects are precisely estimated from a scene, it is possible to transform grasp poses defined in a local frame of a target object to the scene while knowing the 3D configurations of surrounding objects. Thus, robots can build a digital twin of an environment to compute safe trajectories of manipulators. Furthermore, stable grasp poses for a new object can be determined by hand or automatically annotated by physics simulations and real robot attempts. Thus, an ideal system in the future can be realized by integrating the self-supervised collection of training data and automated grasp pose annotations, which requires no human intervention for safely manipulating new objects in cluttered environments.

Bibliography

[1] D. Fischinger, P. Einramhof, K. Papoutsakis, W. Wohlkinger, P. Mayer, P. Panek, S. Hofmann, T. Koertner, A. Weiss, A. Argyros, *et al.*, "Hobbit, a care robot supporting independent living at home: First prototype and lessons learned," *Robotics and Autonomous Systems*, vol. 75, pp. 60–78, 2016.

[2] 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*, 2016.

[3] T. FÃ¤ulhammer, R. AmbruÅŸ, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, "Autonomous learning of object models on a mobile robot," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 26–33, 2017.

[4] M. F. B. van der Burgh, J. J. M. Lunenburg, R. P. W. Appeldoorn, L. L. A. M. van Beek, J. Geijsberts, L. G. L. Janssen, P. van Dooren, H. W. A. M. van Rooy, A. Aggarwal, S. Aleksandrov, K. Dang, A. T. Hofkamp, D. van Dinther, and M. J. G. van de Molengraft, "Tech united eindhoven @home 2019 champions paper," in *RoboCup 2019: Robot World Cup XXIII*, S. Chalup, T. Niemueller, J. Suthakorn, and M.-A. Williams, Eds., Springer, 2019, pp. 529–539, ISBN: 978-3-030-35699-6.

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

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *The European Conference on Computer Vision (ECCV)*, 2014.

[10] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of human support robot as the research platform of a domestic mobile manipulator," *ROBOMECH journal*, vol. 6, no. 1, p. 4, 2019.

[11] E. Muñoz, Y. Konishi, V. Murino, and A. Del Bue, "Fast 6d pose estimation for texture-less objects from a single rgb image," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 5623–5630.

[12] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, Ieee, 2010, pp. 998–1005.

[13] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, p. 155, 2008.

[14] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *European Conference on Computer Vision*, Springer, 2016, pp. 606–619.

[15] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision (ACCV)*, 2012.

[16] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *The European Conference on Computer Vision (ECCV)*, 2014.

[17] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "Bop: Benchmark for 6d object pose estimation," in *The European Conference on Computer Vision (ECCV)*, 2018.

[18] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbè, E. Brachmann, F. Michel, C. Rother, and J. Matas, "Bop challenge 2020 on 6d object localization," in *European Conference on Computer Vision Workshops*, Springer, 2020.

[19] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[20] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–8.

[21] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[24] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, Springer, 2006, pp. 404–417.

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.

[26] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conference on Computer Vision*, Springer, 2010, pp. 356–369.

[27] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*, IEEE, 2009, pp. 3212–3217.

[28] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981, ISSN: 0001-0782. [Online]. Available: `http://doi.acm.org/10.1145/358669.358692`.

[29] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypothesis verification framework for 3d object recognition in clutter," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1383–1396, 2016.

[30] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto, "Fast 6d pose estimation from a monocular image using hierarchical pose trees," in *European Conference on Computer Vision*, Springer, 2016, pp. 398–413.

[31] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *European Conference on Computer Vision*, Springer, 2014, pp. 462–477.

[32] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 2987–2992.

[33] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 2155–2162.

[34] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, Springer, 2012, pp. 113–122.

[35] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *European Conference on Computer Vision*, Springer, 2016, pp. 834–848.

[36] M. Li and K. Hashimoto, "Accurate object pose estimation using depth only," *Sensors*, vol. 18, no. 4, p. 1045, 2018.

[37] J. Vidal, C.-Y. Lin, X. Lladó, and R. Martí, "A method for 6d pose estimation of free-form rigid objects using point pair features on range data," *Sensors*, vol. 18, no. 8, p. 2678, 2018.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[40] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *The European Conference on Computer Vision (ECCV)*, 2016.

[41] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3d object pose estimation," in *The European Conference on Computer Vision (ECCV)*, 2018.

[42] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3109–3118.

[43] V. Balntas, A. Doumanoglou, C. Sahin, J. Sock, R. Kouskouridas, and T.-K. Kim, "Pose guided rgbd feature learning for 3d object pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3856–3864.

[44] S. Zakharov, W. Kehl, B. Planche, A. Hutter, and S. Ilic, "3d object instance recognition and pose estimation using triplet loss with dynamic margin," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017, pp. 552–559.

[45] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *The European Conference on Computer Vision (ECCV)*, 2018.

[46] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[47] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[48] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

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

[50] T.-T. Do, T. Pham, M. Cai, and I. Reid, "Real-time monocular object instance 6d pose estimation," in *British Machine Vision Conference (BMVC)*, 2018.

[51] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, and C. Rother, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[52] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[53] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, 2012, pp. 341–349.

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[55] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[56] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 107, 2017.

[57] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[58] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[59] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: 6D pose object detector and refiner," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950.

[60] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *The European Conference on Computer Vision (ECCV)*, 2018.

[61] Z. Li, G. Wang, and X. Ji, "CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7678–7687.

[62] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[63] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[64] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects," in *The IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[65] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze, "Rgb-d object modelling for object recognition and tracking," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 96–103.

[66] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modeling," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011.

[67] F. Wang and K. Hauser, "In-hand object scanning via rgb-d video segmentation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 3296–3302.

[68] Y. Fu, Q. Yan, L. Yang, J. Liao, and C. Xiao, "Texture mapping for 3d reconstruction with rgb-d sensor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018, pp. 4645–4653.

[69] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! — Large-scale texturing of 3D reconstructions," in *Proceedings of the European Conference on Computer Vision*, Springer, 2014.

[70] J. Philip and G. Drettakis, "Plane-based multi-view inpainting for image-based rendering in large scenes," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2018, pp. 1–11.

[71] H. Shum and S. B. Kang, "Review of image-based rendering techniques," in *Visual Communications and Image Processing 2000*, International Society for Optics and Photonics, vol. 4067, 2000, pp. 2–13.

[72] T. Thonat, E. Shechtman, S. Paris, and G. Drettakis, "Multi-view inpainting for image-based scene editing and rendering," in *2016 Fourth International Conference on 3D Vision (3DV)*, IEEE, 2016, pp. 351–359.

[73] O. Whyte, J. Sivic, and A. Zisserman, "Get out of my picture! internet-based inpainting.," in *British Machine Vision Conference (BMVC)*, 2009.

[74] P. Henderson and V. Ferrari, "Learning single-image 3d reconstruction by generative modelling of shape, pose and shading," *International Journal of Computer Vision*, 2019, ISSN: 1573-1405. [Online]. Available: `https://doi.org/10.1007/s11263-019-01219-8`.

[75] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[76] J. Thies, M. Zollhöfer, and M. Nieundefinedner, "Deferred neural rendering: Image synthesis using neural textures," *ACM Trans. Graph.*, vol. 38, no. 4, 2019, ISSN: 0730-0301. [Online]. Available: `https://doi.org/10.1145/3306346.3323035`.

[77] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time CAD model matching for mobile manipulation and grasping," in *Proceedinfs of IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 290–296.

[78] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe, "HERB: A home exploring robotic butler," *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, 2010.

[79] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Mobile manipulation in unstructured environments: Perception, planning, and execution," *IEEE Robotics Automation Magazine*, vol. 19, no. 2, pp. 58–71, 2012.

[80] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. T. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proceedings of the Conference on Robot Learning*, 2018, pp. 306–316.

[81] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "DenseFusion: 6D object pose estimation by iterative dense fusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352.

[82] R. B. Rusu, A. Holzbach, R. Diankov, G. Bradski, and M. Beetz, "Perception for mobile manipulation and grasping using active stereo," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 632–638.

[83] A. Makhal, F. Thomas, and A. P. Gracia, "Grasping unknown objects in clutter by superquadric representation," in *Proceedings of the IEEE International Conference on Robotic Computing*, 2018, pp. 292–299.

[84] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisâ€"A survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[85] Yun Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 3304–3311.

[86] D. Fischinger, A. Weiss, and M. Vincze, "Learning grasps with topographic features," *The International Journal of Robotics Research*, vol. 34, no. 9, pp. 1167–1194, 2015.

[87] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[88] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[89] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015, pp. 1316–1322.

[90] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 3406–3413.

[91] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 769–776.

[92] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, pp. 1–12, 2017.

[93] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4461–4468.

[94] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 1957–1964.

[95] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proceedings of Robotics: Science and Systems*, 2017.

[96] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[97] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "PointNetGPD: Detecting grasp configurations from point sets," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 3629–3635.

[98] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.

[99] J. Cai, H. Cheng, Z. Zhang, and J. Su, "MetaGrasp: Data efficient grasping by affordance interpreter network," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4960–4966.

[100] D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *Proceedings of Robotics: Science and Systems*, 2018.

[101] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, pp. 1336–1342.

[102] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 47, no. 4-5, pp. 421–436, 2018.

[103] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of the Conference on Robot Learning*, 2018, pp. 651–673.

[104] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4238–4245.

[105] E. Jang, C. Devin, V. Vanhoucke, and S. Levine, "Grasp2vec: Learning object representations from self-supervised grasping," in *Proceedings of the Conference on Robot Learning*, 2018, pp. 99–112.

[106] A. Morales, E Chinellato, A. H. Fagg, and A. P. D. Pobil, "Using experience for assessing grasp reliability," *International Journal of Humanoid Robotics*, vol. 1, no. 4, pp. 671–691, 2004.

[107] J. Bohg and D. Kragic, "Grasping familiar objects using shape context," in *Proceedings of the International Conference on Advanced Robotics*, 2009, pp. 1–6.

[108] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.

[109] C. Liu, B. Fang, F. Sun, X. Li, and W. Huang, "Learning to grasp familiar objects based on experience and objectsâ€™ shape affordance," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2710–2723, 2019.

[110] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal, "Template-based learning of grasp selection," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 2379–2384.

[111] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A kernel-based approach to direct action perception," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 2605–2610.

[112] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic, "Generalizing grasps across partly similar objects," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 3791–3797.

[113] R. Detry, C. H. Ek, M. Madry, and D. Kragic, "Learning a dictionary of prototypical grasp-predicting parts from grasping experience," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 601–608.

[114] R. Detry and J. Piater, "Unsupervised learning of predictive parts for cross-object grasp transfer," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1720–1727.

[115] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[116] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[117] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[118] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[119] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016.

[120] C. Raposo and J. P. Barreto, "Using 2 point+ normal sets for fast registration of point clouds with small overlap," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5652–5658.

[121] P. Rodrigues, M. Antunes, C. Raposo, P. Marques, F. Fonseca, and J. P. Barreto, "Deep segmentation leverages geometric pose estimation in computer-aided total knee arthroplasty," *Healthcare Technology Letters*, vol. 6, no. 6, pp. 226–230,

[122] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[123] B. D. Rebecca König, "A hybrid approach for 6dof pose estimation," in *Proceedings of the European Conference on Computer Vision Workshopes (ECCVW)*, 2020.

[124] T. Hodaň, D. Baráth, and J. Matas, "EPOS: Estimating 6D pose of objects with symmetries," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[125] C.-Y. Fu, M. Shvets, and A. C. Berg, "Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free," *arXiv:1901.03353*, 2019.

[126] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[127] J. Yu, Y. Fan, J. Yang, N. Xu, Z. Wang, X. Wang, and T. Huang, "Wide activation for efficient and accurate image super-resolution," *arXiv:1808.08718*, 2018.

[128] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *The European Conference on Computer Vision (ECCV)*, 2016.

[129] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[130] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[131] 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*, 2019. arXiv: 1911.04231 [cs.CV].

[132] M. Rad, M. Oberweger, and V. Lepetit, "Domain transfer for 3d pose estimation from color images without manual annotations," in *Computer Vision – ACCV 2018*, C. Jawahar, H. Li, G. Mori, and K. Schindler, Eds., Springer, 2019, pp. 69–84, ISBN: 978-3-030-20873-8.

[133] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[134] S. Hampali, M. Rad, M. Oberweger, and V. Lepetit, "Honnotate: A method for 3d annotation of hand and object poses," in *CVPR*, 2020.

[135] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 586–595.

[136] J. Sock, K. I. Kim, C. Sahin, and T.-K. Kim, "Multi-task deep networks for depth-based 6d object pose and joint registration in crowd scenarios," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[137] M. Bui, S. Zakharov, S. Albarqouni, S. Ilic, and N. Navab, "When regression meets manifold learning for object recognition and pose estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[138] R. Brégier, F. Devernay, L. Leyrit, and J. L. Crowley, "Symmetry aware evaluation of 3d object detection and pose estimation in scenes of many parts in bulk," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2209–2218.

[139] Y. Nakagawa, H. Uchiyama, H. Nagahara, and R. I. Taniguchi, "Estimating surface normals with depth image gradients for fast and accurate registration," in *2015 International Conference on 3D Vision*, 2015, pp. 640–647.

[140] E. Potapova, K. M. Varadarajan, A. Richtsfeld, M. Zillich, and M. Vincze, "Attention-driven object detection and segmentation of cluttered table scenes using 2.5 d symmetry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 4946–4952.

[141] G. Toscana, S. Rosa, and B. Bona, "Fast graph-based object segmentation for rgb-d images," in *Proceedings of SAI Intelligent Systems Conference*, Springer, 2016, pp. 42–58.

[142] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning task constraints for robot grasping using graphical models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1579–1585.

[143] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1311–1317.

[144] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018, pp. 3516–3523.

[145] R. Antonova, M. Kokic, J. A. Stork, and D. Kragic, "Global search with bernoulli alternation kernel for task-oriented grasping informed by simulation," in *Proceedings of the Conference on Robot Learning*, 2018, pp. 641–650.

[146] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, IEEE, vol. 2, 2006, pp. 1735–1742.

[147] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of human support robot as the research platform of a domestic mobile manipulator," *ROBOMECH Journal*, vol. 6, no. 4, pp. 1–15, 2019.

[148] S. Chitta, I. Sucan, and S. Cousins, "Moveit! [ros topics]," *IEEE Robotics Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[149] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proceedings of the IEEE International Conference on Robotics and Automation, Workshop on open source software*, 2009.

[150] T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018, pp. 5882–5889.

[151] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, "Self6d: Self-supervised monocular 6d object pose estimation," in *The European Conference on Computer Vision (ECCV)*, 2020.

[152] J. Sock, P. Castro, A. Armagan, G. Garcia-Hernando, and T.-K. Kim, "Tackling two challenges of 6d object pose estimation: Lack of real annotated rgb images and scalability to number of objects," *arXiv:2003.12344*, 2020.

[153] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *International Conference on Robotics and Automation (ICRA)*, 2020.

[154] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.

[155] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[156] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations (ICLR)*, 2016.

Appendix A: Pix2Pose Parameters

A.1 Pools of Symmetric Poses

I: Identity matrix, R_a^{Θ}: Rotation matrix about the a-axis with an angle Θ.

- LineMOD and LineMOD Occlusion - eggbox and glue: $sym = [I, R_z^{\pi}]$
- T-Less - obj-5,6,7,8,9,10,11,12,25,26,28,29: $sym = [I, R_z^{\pi}]$
- T-Less - obj-19,20: $sym = [I, R_y^{\pi}]$
- T-Less - obj-27: $sym = [I, R_z^{\frac{\pi}{2}}, R_z^{\pi}, R_z^{\frac{3\pi}{2}}]$
- T-Less - obj-1,2,3,4,13,14,15,16,17,18,24,30: $sym = [I]$, the z-component of the rotation matrix is ignored.
- Objects not in the list (non-symmetric): $sym = [I]$

A.2 List of outlier thresholds

	ape	bvise	cam	can	cat	driller	duck	eggbox	glue	holep	iron	lamp	phone
θ_o	0.1	0.2	0.2	0.2	0.2	0.2	0.1	0.2	0.2	0.2	0.2	0.2	0.2

Table A.2.1: Outlier thresholds θ_o for objects in LineMOD

	ape	can	cat	driller	duck	eggbox	glue	holep
θ_o	0.2	0.3	0.3	0.3	0.2	0.2	0.3	0.3

Table A.2.2: Outlier thresholds θ_o for objects in LineMOD Occlusion

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
θ_o	0.1	0.1	0.1	0.3	0.2	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.2	0.2	0.2	0.3	0.3	0.2	0.3	0.3	0.2	0.2	0.3	0.1	0.3	0.3	0.3	0.3	0.3	0.3

Table A.2.3: Outlier thresholds θ_o for objects in T-Less

Appendix B: Open Resources

A.1 Codes and Tools

- Pix2Pose (Chapter 3): `https://github.com/kirumang/Pix2Pose`
- Neural Object Learning (Chapter 4): `https://github.com/kirumang/NOL`
- DGCM-Net (Chapter 6): `https://rgit.acin.tuwien.ac.at/v4r/dgcm-net`

A.2 Dataset

- Single sequence Multi Objects Training (SMOT): `https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/smot`
- Learning from Experience and Demonstration for 6-DOF Grasping Dataset (LfED-6D): `https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/lfed-6d-dataset`

A.3 Videos

- Pix2Pose (Chapter 3): `https://youtu.be/fQJPS01cmac`
- Neural Object Learning (Chapter 4): `https://youtu.be/wnmGmbDn3ZQ`
- Multi-Task Template Matching (Chapter 5): `https://youtu.be/rs4ekmE6SGo`
- DGCM-Net (Chapter 6): `https://youtu.be/iI_P1UVXfjo`